On systems analysis and simulation of ecological processes

P.A. Leffelaar (Editor)

Department of Theoretical Production Ecology, Wageningen Agricultural University, P.O. Box 430, 6700 AK Wageningen, the Netherlands

Wageningen 1992
Contents

Preface

Contributors

Part A Fundamentals of dynamic simulation

1 Simulation of living systems - C.T. de Wit
   1.1 Systems, models and simulation
   1.2 Descriptive and explanatory models
   1.3 State-determined dynamic systems and models
   1.4 The usefulness of ecological research using models

2 Basic elements of dynamic simulation - P.A. Leffelaar
   2.1 Introduction
   2.2 State variables, rate variables and driving variables
   2.3 Feedback and relational diagrams
   2.4 Analytical integration and system behaviour in time
   2.5 Numerical integration and the time coefficient
   2.6 An example
   2.7 Summary

3 A simulation language: Continuous System Modeling Program III - P.A. Leffelaar
   3.1 Introduction
   3.2 The structure of the model
   3.3 Integration of rate equations
   3.4 The time loop
   3.5 Data input and output
   3.6 Interpolation
   3.7 Reruns
   3.8 Some elements of CSMP
   3.9 Syntax
   3.10 Concluding remarks

4 The growth of yeast - C.T. de Wit and J. Goudriaan
   4.1 Description of the system
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.10</td>
<td>Effect of the type of boxcar train on population growth</td>
<td>111</td>
</tr>
<tr>
<td>8.11</td>
<td>Concluding remarks</td>
<td>112</td>
</tr>
<tr>
<td>9</td>
<td>Mass flow, numerical dispersion and diffusion</td>
<td>113</td>
</tr>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>113</td>
</tr>
<tr>
<td>9.2</td>
<td>Mass flow and numerical dispersion in a column of soil</td>
<td>113</td>
</tr>
<tr>
<td>9.2.1</td>
<td>The general transport equation</td>
<td>113</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Derivation of the differential equations for mass flow</td>
<td>114</td>
</tr>
<tr>
<td>9.2.3</td>
<td>Analysis of numerical dispersion with the aid of a Taylor series development</td>
<td>116</td>
</tr>
<tr>
<td>9.2.4</td>
<td>Suppression of numerical dispersion</td>
<td>117</td>
</tr>
<tr>
<td>9.3</td>
<td>Diffusion in a soil column</td>
<td>118</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Derivation of the differential equations for diffusive flow</td>
<td>119</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Transport of heat in soil</td>
<td>121</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Derivation of the differential equations for heat flow</td>
<td>121</td>
</tr>
<tr>
<td>9.3.4</td>
<td>An analytical solution for heat flow</td>
<td>123</td>
</tr>
<tr>
<td>9.4</td>
<td>Derivation of the general transport equation</td>
<td>125</td>
</tr>
<tr>
<td>9.5</td>
<td>Concluding remarks</td>
<td>127</td>
</tr>
<tr>
<td>10</td>
<td>Simulation using FORTRAN - C. Rappoldt, P.A. Leffelaar and D.W.G. van Kraalingen</td>
<td>129</td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>129</td>
</tr>
<tr>
<td>10.2</td>
<td>The time loop, the correct calculation sequence and Euler's integration</td>
<td>129</td>
</tr>
<tr>
<td>10.3</td>
<td>Runge-Kutta integration</td>
<td>147</td>
</tr>
<tr>
<td>11</td>
<td>Additional exercises II - P.A. Leffelaar</td>
<td>153</td>
</tr>
<tr>
<td>12</td>
<td>Solutions to the exercises</td>
<td>161</td>
</tr>
<tr>
<td>12.1</td>
<td>Introductory remark</td>
<td>161</td>
</tr>
<tr>
<td>12.2</td>
<td>The answers to the exercises</td>
<td>161</td>
</tr>
<tr>
<td>13</td>
<td>References</td>
<td>245</td>
</tr>
</tbody>
</table>

**Appendix 1** Derivation of the relative error in the rectangular integration method for a given exponential rate curve given as a function of time (i.e. integration without feedback).

**Appendix 2** Derivation of the relative error in the rectangular integration method for an exponential rate curve which is not known as a function of time (i.e. integration with feedback).

**Appendix 3** Summary of the processing of a CSMP program.

**Appendix 4** Derivation of the average concentration (AVC) at the boundary of two consecutive layers of different thickness (TCOM) in connection with the suppression of numerical dispersion.

**Appendix 5** Derivation of the average conductivity of two adjacent layers of unequal thickness with different conductivity or resistance.

**Appendix 6** The main program FORSIM, the Euler and Runge-Kutta drivers, and the adapted
integration routines from Press et al. (1986), including subroutine headers which explain their meaning.
This book has its origin in the course 'Systems analysis and simulation in the theoretical production ecology', that was initiated by Professor C.T. de Wit and coworkers in the early 1970s. A part of the original material from 'Simulation of ecological processes' (de Wit & J. Goudriaan, 1978) and 'Simulation of transport processes in soils' (de Wit & H. van Keulen, 1975) is reworked and supplemented with new material, which often was the result of (international) Post-Graduate Courses. The original thoughts on how systems can be analysed and modelled, are maintained of course: they originate partly from J.W. Forrester's book 'Industrial Dynamics' from 1961.

The book consists of part A and part B. In part A the fundamentals of systems analysis are treated: definitions of the terms system, model, and simulation; state, rate and driving variables; feedback and the simplest numerical integration method; time coefficient; static and dynamic equilibria; and an introduction to the simulation language CSMP. The subject-matter is illustrated with two simple yeast growth models. In part B more complicated and accurate integration techniques are discussed, the use of FORTRAN in CSMP programs, and a simulation method to describe delays and dispersion. Furthermore, the phenomenon of numerical dispersion is discussed in examples of mass flow and diffusion in a soil column. Both part A and B contain a chapter with extra exercises, that enable the student to evaluate his or her skills once again. Part B is finished with a treatment of the general calculation language FORTRAN as a possible means for simulation. This has been done because FORTRAN has a greater portability than CSMP. However, it takes more effort to work with FORTRAN than with CSMP.

The novice should start with part A of the book, the more experienced student might directly start with part B of the book.

Though the one trimester course 'Systems analysis and simulation in the theoretical production ecology', is meant for both undergraduate and graduate students of the Wageningen Agricultural University, also PhD students take advantage of the course. The text can be used on its own because of the many exercises. However, the most fruitful results will be obtained by interaction with the lecturer and the computer. The exercises are primarily meant to test the student's knowledge, but the answers often add some new information to the problem at hand. Each exercise is meant to clarify an aspect of a problem, be it a theoretical or a technical aspect, but furthermore exercises have been kept simple. If one has made oneself familiar with the methodology, the difficulty in analysing systems is mainly due to the availability of knowledge about the system at hand.

The dynamic models in the text are available on floppy disk. They can be used on mainframe computers on which the CSMP simulation language is installed or on IBM PC-AT's, or compatibles. On personal computers, the computer simulation language PCSMP (Jansen et al., 1988) should be used. The FORTRAN modules for dynamic simulation can be used on any computer system. These modules are also available from the department of Theoretical Production Ecology.

I wish to thank Jan Goudriaan for reviewing the typescript and Wilma Biesheuvel for critically recalculating all the exercises in the book.
Contributors

Goudriaan, J., Department of Theoretical Production Ecology, Wageningen Agricultural University, P.O. Box 430, 6700 AK Wageningen, the Netherlands.

Kraalingen, D.W.G. van, Agricultural Research Department, Centre for Agrobiological Research, P.O. Box 14, 6700 AA Wageningen, the Netherlands.

Leffelaar, P.A., Department of Theoretical Production Ecology, Wageningen Agricultural University, P.O. Box 430, 6700 AK Wageningen, the Netherlands.

Roermund, H.J.W. van, Department of Entomology, Wageningen Agricultural University, P.O. Box 8031, 6700 PD Wageningen, the Netherlands.

Wit, C.T. de, Department of Theoretical Production Ecology, Wageningen Agricultural University, P.O. Box 430, 6700 AK Wageningen, the Netherlands.

Rappoldt, C., Agricultural Research Department, Institute for Soil Fertility Research, P.O. Box 30003, 9750 RA Haren, the Netherlands.
Part A Fundamentals of dynamic simulation
Simulation of living systems

C.T. de Wit

1.1 Systems, models and simulation

Since the 1950s, engineers have paid much attention to the study of complex, dynamic systems. Their successes inspired biologists to apply similar techniques in their disciplines. The approach is characterized by the terms: system, model and simulation.

A system is a limited part of reality that contains interrelated elements. The totality of relations within the system is known as the 'system structure': both systems and models have a structure. A model is a simplified representation of a system. Simulation is the building of mathematical models and the study of their behaviour in reference to those of the systems.

There are many kinds of models. A simple mathematical model is the well-known relation between velocity and distance covered by a falling apple depending on the gravitational acceleration and the time from the moment of release. An example of a non-mathematical model is a map. This is a simplified representation of the earth's surface containing relevant information and it allows measurements of distances or areas. Depending on the objectives, a map will display railway lines, lines of equal rainfall or vegetation. A scale model of a ship in a towing tank enables measurements of its resistance in the water to help predict the behaviour of the real ship when this has been built.

It follows from the definition that a model is a system, but the reverse may also be true. A machine is a model of the conception of the engineer and it certainly performs worse than expected. And when an engineer applies simulation, he develops models that lie in between his conception and reality. The ultimate machine is in fact a model of his simulation model, which in its turn is a simplified representation of his mental conception.

Biological systems are not simplified representations of the conception of the biologist, even though current trends in agriculture might seem to be moving that way. Therefore, it may be that the approach that has been so successful in technology is not as useful in biology. There are, of course, ample examples of biological systems: a membrane, a cell, an organ, a plant, an animal, a field, a woodland and a lake. In this context, fields, woods, lakes, estuaries, pastures and all the rest have been given the general name of 'ecosystems'.

A system is a limited part of reality, so that a border has to be chosen. It is wise to select the boundary so that the system is isolated from its environment. This is hardly ever possible, but then it should be attempted to choose a border so that the environment might influence the system, but that the system affects the environment as little as possible. To this end, it is often essential to select a system that is larger than would seem necessary for the primary objective. Therefore, in ecological systems, the microclimate is often a part of the system, but most people willingly neglect -unjustly- the influence of the processes in the ecosystem on the macroclimate, that is considered to be a measurable environment not influenced by the ecosystem.

1.2 Descriptive and explanatory models

A file with data on an ecosystem might be called a model, but it is one of the most unclear and unusable kinds. Potential uses of the data may be formulated however, and then clarity may be introduced by a treatment of this data. This may result in maps that represent aspects of the ecosystem, or in statistical analysis which summarize some of the interrelations. If the time dimension is also taken into account during collection and treatment of the data, these
models are no longer static but dynamic. However, those models remain descriptive, showing the existence of relations between the elements of a system without any explanation, but, of course, this was not their original purpose.

However, explanatory models are possible in biology, because various levels of organization or knowledge are distinguished in this science, just like in all other natural sciences. These levels of knowledge distinguish themselves by the level of integration at which the processes occur. The different levels of integration may be classified, according to the size of the system, such as molecules, cell structures, cells, tissues, organs, individuals, populations, communities and ecosystems. Explanatory models demand that research has to be carried out for at least two integration levels. The lower integration level will then be the explanatory level and the upper level is the one to be explained. In this way, one might attempt to derive the characteristics of membranes from the characteristics of the molecules of which they consist, or one might try to explain the processes in an ecosystem on the basis of knowledge of the behaviour and physiology of the constituent species. When knowledge at the explanatory level is sufficiently extensive and authoritative, and a model of the system to be explained is designed on this basis, it will not be necessary to test the model by comparing its results with those of the real system. For example, models for space travel are so good that the actual journey into space is unnecessary. But explanatory models in biology are frequently so inadequate that proof of their usefulness is necessary. Even if the results of the model correspond to the observations of the system being modelled, there is room for doubt regarding the correctness of the model. However, good agreement is still more the exception than the rule.

If there are discrepancies between the results of the model and reality, the model may be adjusted to obtain better agreement. Then, something that started as an explanatory model will degenerate into a descriptive model. The term 'degeneration' in this context does not mean that descriptive models are inferior to explanatory models. It is used here to emphasize that in this way inscrutable models are obtained with an unjustified pretention to explain something. This is the reason why many models made in ecological studies to date have done more harm than good.

The proper way of working is heuristic, by the path of methodical improvement. When unacceptable discrepancies between the model and the system are observed, it may be possible to determine by experiments with both the model and the system, which aspects of the model are suspicious. These aspects are then studied at the explanatory level. On basis of this new information, elements of the model may be replaced by improved versions, after which the results of the model and the reality may be compared once more.

Explanatory models can be static or dynamic. An example of a static model is one in which the connection between respiration and growth of organisms is calculated on the basis of knowledge of the biochemical processes involved. Another example is a model in which the light distribution over the leaves of a vegetation is calculated from the canopy architecture, reflection and transmission of the leaves, solar position, and the brightness of the sky. The results of the calculations performed with these static models or the models themselves, often form a part of dynamic models. These dynamic models are simplified versions of dynamic systems, or in other words, systems that change with time. The development of these models and the study of their behaviour is frequently called 'simulation'.

1.3 State-determined dynamic systems and models

Simulation of ecological systems with an explanatory model is based on the assumption that the state of every system at every moment can be quantitatively characterized and that changes in a system can be described by means of mathematical equations. This hypothesis leads to the formulation of state-determined dynamic models, in which state-, rate- and driving variables can be distinguished.

State variables are variables like the amounts of biomass, the number of animals, the amount of nitrogen or water in soil, plant or animal, etcetera.

Driving variables or forcing functions characterize the influence of external factors on the system, and are not influ-
enced by the processes within the system. Their value must be monitored continuously. Examples are macro-meteorological variables such as rain, wind and radiation, or the supply of nutrients or migration of animals over the boundaries of the system. Depending on the system's limits, the same variables can be driving variables, or state or rate variables.

**Rate variables** indicate the rate at which the state variables change. Their values are determined by the state and driving variables according to rules that are based on knowledge of the physical, chemical and biological processes that take place in the system, and not on the basis of a statistical analysis of the behaviour of the system. This is the most important distinction between models that describe and models that attempt to explain.

After the calculation of all rate variables, these are used to calculate the state variables according to the scheme: state variable at time \( t + \Delta t \) is equal to the state variable at time \( t \) plus the rate at time \( t \) multiplied by \( \Delta t \). This procedure, called integration, gives the new values of the state variables, and the calculational procedure may be repeated. Obviously, modern calculators are indispensable here. Later in this book, various examples will be presented of applications in ecology.

Rates are not interdependent in state-determined systems: each rate depends at each moment on the value of state and driving variables and can therefore be calculated independent of any other rate. We can illustrate this mutual independence of rates by means of the following example. It will be clear that the rate of plant growth, as measured by the increase in weight of its structural tissues, is closely related to the net photosynthesis in the leaves. However, in an explanatory model of plant growth, this relation is the result of the simultaneous operation of various processes, and it is thus no direct relation. The rate of photosynthesis contributes to the amount of the plant's reserves and, also dependent upon other state variables, the growth rate of the various organs is connected with the amount of reserves. Although photosynthesis stops when it gets dark, growth does continue until the reserves have been exhausted, and sometimes even further, but then at the cost of previously formed structural tissues.

The number of state variables that can be distinguished in ecosystems is discouragingly large. This does not only involve plants, herbivores, carnivores and micro-organisms, but also their various kinds, and of these the numbers, the size, the age, the development stage, etc. For plants, not only the weight and the surface area of the leaves are of importance, but also their nitrogen and mineral contents, their enzymes and other biochemical characteristics. It is possible to continue in this way, and this is the very reason why attempts at constructing models on the basis of a full knowledge of all the biological, physical and chemical processes are completely unrealistic. Models are nothing more than simplified representations of reality, and this simplification characterizes itself by the limited number of state variables that are taken into account.

Analogous to other scientific approaches, it is assumed that the number of state variables in a model is not only limited by the boundaries of the system to be studied, but also by a clear description of the objectives. Consequently, it can never be a reasonable objective of a model to fully comprehend an ecosystem, but if this is so, what might it then be? For every application of models or their construction, we must start by clearly describing the objective.

In agriculture the formulation of this objective is, initially, simple: to obtain insight into the relation between yield on the one hand and human effort on the other. This objective can be further detailed into questions regarding the relation between the addition of nitrogen and yield for well-defined boundary conditions. For every objective an optimum can be found for the number of state variables that should be included. Initially, the applicability of a model increases with an increasing number of state variables. But then the applicability decreases again because the inclusion of a new state variable diverts the attention from the state variables introduced previously because these were considered more important. The attempt at arranging the state variables in order of importance is very time-consuming, and many modelling efforts in ecology are sometimes explicitly, but more often implicitly, geared towards this goal.
1.4 The usefulness of ecological research using models

The maintenance of the integrity and of options for developments of ecosystems, such as fields, woods, estuaries and the human society as a whole, is one of the greatest problems of society. The question is whether it is also a scientific problem. This will only be the case when the problem has not been solved and when it can be made acceptable on theoretical grounds that the problem can be solved. We need not worry about the former, and as far as solvability is concerned: precisely the existence of techniques that enable us to summarize knowledge in operative simulation models will inspire some people to an optimistic vision. But is this justified? Problems can only be solved when the solution can be falsified or, being rather more positive with regard to modelling, when models can be tested for their usefulness. To investigate the possibilities for this, it is useful to distinguish between verifiable and speculative models.

Verifiable models can only be created from repeatable or recurring systems. Examples of repeatable systems are continuous cultures of bacteria, farms and industrial processes. Recurring systems appear to the observer at different places at the same time in different stages. Examples are: stars, individuals of a species and ecological systems whose development in time is controlled by strong negative feedbacks, such as peat bogs. Of these latter ecosystems no two will ever be the same, but this does not imply that their models might not be identical: after all, a model remains a simplified representation of reality. It is clear that experiments can always be carried out with repeatable systems, but that recurring systems are not or less accessible to experimentation. Ecology develops from a descriptive science to a science in which experiments are carried out with repeatable and recurring systems. The latter is justified because the chance of disturbances is small due to the existence of negative feedbacks in these systems, and because there are many of them. System-analytic research and simulation should be limited to repeatable and recurring systems to even a greater extent than the experiments, because the knowledge of the processes that occur is still so limited that verification of models is needed.

Besides repeatable and recurring systems, unique systems exist. Examples of these are climatic systems and systems determined by geographical circumstances such as the Oosterschelde (Eastern estuary of the river Schelde), the Waddenzee (Wadden Sea to the north of the Netherlands), the world itself and human society, as well as ecosystems whose development is only controlled by weak negative feedbacks within narrow limits, so that the originally identical systems may diverge in space and time. Evolution itself is an example of such a system and consequently also breeding in agriculture. Precisely because the breeding process is irreversible, it is only justified when measures are taken to conserve the gene pool. Models of unique systems are speculative models, since they cannot be tested for their usefulness. After all, analysis and testing should apply to completely independent systems if the results of the test are to lead to the possibility of rejecting the model. Of course, unique systems possess recurring elements and these can be isolated and thus made accessible for experiments and the construction of models. Therefore, unique systems can be partially analysed and sometimes experiments can be performed within these systems. The significance of observation and experimentation should then be weighed against the risk of disturbing the system; in case of doubt no experiments should be done. Therefore, genetic manipulation should be carried out with the greatest care.

Speculative models cannot be verified but they can more or less be trusted. The confidence in these models will grow when analogous methods of analysing repeatable and recurring systems lead to the formulation of verifiable models with useful results. The confidence in models of unique systems whose behaviour is controlled by physical phenomena might even be quite large. Nobody need seriously doubt the calculations regarding the relation between the height of the dikes and the chance of flooding in the Netherlands, but when flooding occurs, confidence will have decreased dramatically.

In ecology, useful models have only been constructed of relatively simple systems, and according to some, only of those systems that also could have been understood without the use of the advanced methods of systems analysis and simulation. Therefore, confidence in speculative models of unique systems is justifiably small. The tragic is that,
scientifically speaking, ecologists are only able to make statements on a growing number of repeatable and recurring systems, but that society is demanding statements on the development of the unique systems. Meeting society's requirements can therefore easily lead to plainly fraudulent comments. Many ecologists are well aware of this and refrain from making statements, but this then results in this blank patch in our knowledge being filled in irresponsibly by so-called instant-ecologists.

'Global models' with any perspective of being useful are more complicated than models of those ecosystems whose usefulness has been demonstrated. Indeed, these models should include knowledge from many different scientific disciplines, and in some of the relevant scientific disciplines, this knowledge is significantly smaller than for those of the physical, chemical and biological processes that will play a role in simple ecosystems. This will mean that the basis for confidence will be missing and in many cases will not even be laid. Consequently, 'global' models remain unusable instruments in the preparation of policy decisions. Therefore, the question is whether research efforts should be made to analyse 'global' systems that may result in speculative, interdisciplinary models. Nevertheless, the answer is in the affirmative. The basis for eventual confidence can be laid by comparative research of verifiable and speculative models. Systems analysis and modelling are the only developing interdisciplinary professional fields that enable us to integrate and oversee our incomplete knowledge, and it is pointless to throw away half-worn-out shoes before new ones have been designed. And long before speculative, interdisciplinary models can be used to support governmental decisions in society, these kinds of models can be used as instruments of research policy in order to indicate research priorities.

Criteria can be formulated to which speculative, interdisciplinary models and considerations must comply in order to contribute to scientific development. This can result in recommendations for research that become increasingly important when more models are evaluated in a comparable manner. Possible criteria are:

- The objectives of the model must be well-described.
- Based on the objectives it should be argued which disciplines are to contribute to the development of the model, and to what extent.
- The contributions from the various disciplines should be reasonably scientifically reliable.
- Verifiable parts of the model should indeed be tested.
- Claims to usefulness should be critically evaluated.
- The model must be sound; this means that it should be known which suppositions have been incorporated, and this should be done in such manner that the consequences of other suppositions can be determined within the framework of the model's objectives.
- The model must be used soundly. This means that the designers will have investigated and commented on the consequences of a reasonable number of suppositions, and not only those suppositions which might confirm presuppositions on the results.
- The model must be clear and others than the designers should be able to use it.
2 Basic elements of dynamic simulation

P.A. Leffelaar

2.1 Introduction

When analysing systems one is usually interested in the status of the system at a certain moment and its behaviour as a function of time. A system, defined in Section 1.1 as a limited part of reality that contains interrelated elements, may be too complex to study in its entirety. However, a model, defined as a simplified representation of a system that contains the elements and their relations that are considered to be of major importance for system behaviour, may be easier to study. The design of such models and the study of the model behaviour in relation to that of the system is called simulation; when the change with time is also included, it is called dynamic simulation.

Dynamic simulation models are based on the assumption that the state of each system at any moment can be quantified, and that changes in the state can be described by mathematical equations: rate or differential equations. This leads to models in which state, rate, and driving variables can be distinguished.

The purpose of this chapter is to introduce the method of constructing models according to the state variable approach by very elementary systems in Section 2.2. The concept of feedback and the possibility of visualizing the available knowledge of a system by means of relational diagrams will be discussed in Section 2.3. Section 2.4 shows how the corresponding differential equations may be integrated analytically to obtain the state variables as a function of time in these simple systems. Slight changes in differential equations make analytical solutions impossible, so solutions must be obtained by numerical integration methods. These solutions are based on the assumption that the rate of change is constant during a small period of time, \( \Delta t \). The principle of numerical integration, the relation between the time interval of integration, \( \Delta t \), and the time coefficient of a system are discussed in Section 2.5. In Section 2.6 a more complex system is analysed using the methods presented. The chapter is summarized in Section 2.7.

2.2 State variables, rate variables and driving variables

To introduce the method of constructing models according to the state variable approach, the following examples of elementary systems are used (Figure 1):

1. a car driving at a constant speed;
2. a number of animals that increases each year by a certain fraction;
3. a tank that is filled by a flow of water through an adjustable valve until a certain water level is reached.

The state variables in these examples are the distance covered by the car, the number of animals and the amount of water in the tank, respectively. Generally, state variables have dimensions of length, number, volume, weight, energy or heat content.

The ultimate status of a system is not the only feature of interest; we are also concerned with its behaviour in time. Thus, the rate of change of the state variables in time must be known, as well as the direction of change. If these rates have a clear pattern, they may be formalized by means of rate equations or differential equations. The rate equations and their graphical representation for the three elementary systems are given in Figure 1. Rate variables, the left hand sides in Equations 1, 2 and 3 in Figure 1, have the dimension of a state variable per unit of time, i.e.
length time\(^{-1}\), number time\(^{-1}\) and volume time\(^{-1}\), respectively. In Equations 2 and 3 the rate variables are functions of state variables, \(A\) and \(W\), respectively, whereas in Equation 1 the state has no effect on the rate. The influence of a state on its rate of change is called feedback and will be discussed in Section 2.4. The proportionality coefficients, \(c\), in Equations 2 and 3 are important for the behaviour of the state variables and often have special names. In biological systems \(c\) is called the relative growth rate; in technical systems the inverse of \(c\) is used, and is called the time coefficient. Time coefficients and their consequences for numerical integration are crucial in dynamic simulation, which is discussed in Section 2.5. The constant \(c\) in Equation 1 is a driving variable with the dimension speed. Driving variables, or forcing functions, characterize the effect of outside conditions on a system at its limits or boundaries, and their values must be monitored continuously. Driving variables may have the dimension of rate variables, as in Equation 1, or state variables, depending on their nature. When the driving variable is temperature, e.g. when the fraction by which the number of animals increases each year depends on temperature, it has the dimension of a state variable. It is imperative to check the dimensions of all variables in any model.

Exercise 1

a. What are the dimensions of \(c\) in Equations 1, 2 and 3 in Figure 1?

b. Which general rules form the basis of dimensional analysis?

2.3 Feedback and relational diagrams

The rate variables in Equations 2 and 3 (Figure 1) are functions of the state variables \(A\) and \(W\), respectively, whereas the rate variable in Equation 1 is independent of the state variable \(D\). When a rate variable, expressed in general terms as the derivative \(dX/dt\), depends on the state variable \(X\), a feedback loop exists, i.e. the state of the variable determines its rate of change, and hence its subsequent state. This feedback takes place continually. Both positive and negative feedback loops exist.

In a negative feedback loop the rate of change may be either positive or negative, but it is a negative function of the value of the state variable. For instance, in the case of the water tank (Equation 3) the rate is positive, but it de-
creases linearly with increasing water level in the tank. An example of a negative rate which decreases as a function of the value of the state variable, is obtained when the sign of the coefficient \( c \) in Equation 2 is reversed. Then the number of animals decreases each year by a certain fraction, denoting exponential mortality. Negative feedback causes the system to approach some stable equilibrium: if the system is disturbed it returns to its equilibrium state. In the case of the water tank, the equilibrium state is the maximum water level, \( W_m \). In the case of exponential mortality the state variable approaches zero.

In case of a positive feedback loop, there is a positive relation between the values of rate and state, so that both continuously increase. The exponential growth of the animals described by Equation 2 is an example of positive feedback. In reality, however, there are limits to growth. Then, the simple Equation 2 no longer describe the system and the model needs revision.

Exercise 2

Give at least three environmental conditions that should always be satisfied to achieve a situation where the relative growth rate, \( c \) in Equation 2, is independent of the amount of organisms and time.

Relational diagrams are used to visualize feedback loops, rate and state variables and, more generally, the available knowledge about a system. They illustrate the most important elements and relationships of a system and represent qualitative models of systems. Relational diagrams may be especially helpful at the start of research to facilitate the formulation of rate and state variables. They also make the content and characteristics of a model easily accessible. Relational diagrams for the three systems are given in Figure 2, following the conventions of Forrester (1961) as shown in Figure 3. Figure 2 shows that feedback is absent in the case of the car, and that there is positive and negative feedback in the case of the animals and the water tank, respectively. When a parameter turns out to be variable, it must be replaced by a table or an auxiliary equation. For instance, if the coefficient \( c \) in Equation 2 for the animals, is dependent on temperature, it can be replaced by a so-called auxiliary variable containing information about this temperature dependence which flows to the rate variable.

Relational diagrams of more complex models can often be analysed in terms of the elementary systems of Figure 2.

![Figure 2](image_url)

Figure 2. Relational diagrams for three elementary systems. Variables and symbols are explained in Figures 1 and 3, respectively.
A state variable, or integral.

Integral symbol for a boxcar train. Small bars indicate that the boxcar train consists of a series of integrals.

Flow and direction of a substance by which an amount, or state variable, is changed. These flows always begin or end at a state variable, and may connect two state variables.

Flow that is discontinuous: when it is closed, it has some numerical value, otherwise it is zero. This flow is controlled by a flow of information.

Flow and direction of information derived from the state variables of the system. Dotted arrows always point to rate variables, never to state variables. The use of information does not affect the information source itself. Information may be delayed and as such be part of a process itself.

Valve in a flow, indicating that the calculation of a rate variable takes place; the lines of incoming information indicate the factors upon which the rate depends.

Source or sink of quantities in whose content one is not interested. This symbol is often omitted.

A constant or parameter.

Auxiliary or intermediate variable in the flow of information.

Sometimes placed next to a flow of information, to indicate whether a loop involves a positive or negative feedback.

Figure 3. Basic elements of relational diagrams. Acronyms of variables represented by these elements are usually written inside the symbols. Note that driving variables are often underlined or placed between parentheses. Intermediate variables are often represented by circles.

2.4 Analytical integration and system behaviour in time

Differential equations summarize existing knowledge of a system, i.e. they relate rate variables to state variables, driving variables and parameters. They thus form a model of that system. When the differential equations are formulated, and if the state of the model at a certain moment is known, its state in the future can be calculated. For this purpose the differential equation must be solved with respect to time. This process is called integration and can be visualized for the simplest case of Equation 1 by determining the distance covered by the car in a certain period. Here, the speed is multiplied by the length of the period. Thus the increase in the value of the state variable in a period \( t_1 - t_0 \) time units (Figure 1), equals the area delimited by the time axis, the line parallel to this time axis at the value \( c \) on the rate axis and the two lines, parallel to the rate axis, at the moments \( t_0 \) and \( t_1 \). For Equations 2 and 3 the situation is different, as the rate variables depend on the state. Then, the formal process to obtain the state variable as a function of time must be applied, i.e. integration of the rate equation. This is shown in Figure 4 for all three models. Integration of Equation 2 produces the well-known exponential growth curve (Equation 5 in Figure 4). The
Car
\[ \frac{dD}{dt} = c \] Equation 1

Animals
\[ \frac{dA}{dt} = c \cdot A \] Equation 2

Water tank
\[ \frac{dW}{dt} = c \cdot (W_m - W) \] Equation 3

\[ \int dD = c \cdot \int dt \]
\[ D = c \cdot t + Q \]

initial value of the state variable at \( t = 0 \):

\[ D = D_0, \; \text{so:} \; Q = D_0 \]

\[ A = A_0, \; \text{so:} \; Q = \ln A_0 \]

\[ \ln A = c \cdot t \]

\[ D = c \cdot t + D_0 \]

\[ A = A_0 \cdot e^{c \cdot t} \]

Equation 4

Equation 5

Equation 6

\[ \int \left( W - W_m \right) = -c \cdot \int dt \]

\[ \ln (W - W_m) = -c \cdot t + Q \]

\[ W = W_m - (W_m - W_0) \cdot e^{-c \cdot t} \]

---

Figure 4. Upper half: analytical solutions (Equations 4, 5 and 6) to the differential Equations 1, 2 and 3 respectively, and their graphs.

Lower half: rate variables (Equations 1, 2a and 3a) as a function of time, derived from Equations 4, 5 and 6 respectively, and their graphs. For an explanation of variables see Figure 1. \( Q \) stands for a general integration constant, and \( D_0, A_0 \) and \( W_0 \) are the initial values of the state variables in the particular models.
relationship between the rate variable, \(dA/dt\), and time is obtained by differentiating Equation 5 with respect to time. This yields Equation 2a, which has the same form as Equation 5. The graph picturing Equation 2a may be used to obtain the state variable. This may seem trivial, since the analytical solution is already available (Equation 5). The graph may, however, be used to illustrate the errors introduced by numerical integration methods when these are used to solve differential equations. This is discussed in Section 6.4.

In the case of the water tank, the rate of inflow of water decreases linearly with the difference between a maximum water level, \(W_m\), and the actual water level, \(W\) (Equation 3). Integration yields Equation 6, expressing that the water level in the tank approaches \(W_m\) exponentially. Equation 3a, obtained by differentiating Equation 6 with respect to time, shows that the rate of inflow decreases exponentially.

Exercise 3
Consider the graphs representing Equations 4, 5 and 6 in Figure 4.

a. What do the slopes of the different lines represent?
b. What is the dimension of the slope in each case?
c. How do the numerical values of the slopes change with time?
d. Does your answer in c) agree with the graphical presentation of Equations 1, 2a and 3a?

As long as differential equations are simple, they may be solved analytically to study the behaviour of the models. Minor changes in the equations, however, e.g. if \(c\) in Equation 2 is a function of temperature, make analytical solutions impossible. This is also the case when model results deviate from the behaviour of the system, and more complex (sub-)models are needed, based on newly gathered knowledge of the system. Then, the resulting set of differential equations should be solved by numerical integration methods.

2.5 Numerical integration and the time coefficient

In numerical integration the assumption is made that the rate of change of a state variable is constant over a short period of time, \(\Delta t\). To calculate the state of a system after that short period one must know the state of the system at time \(t\), \(\text{state}_t\), and the value of the rate variable at that time, \(\text{rate}_t\), calculated from the differential equation. By multiplying the \(\text{rate}_t\) by \(\Delta t\) and adding this product to the value of the state variable according to

\[
\text{state}_{t+\Delta t} = \text{state}_t + \Delta t \cdot \text{rate}_t
\]

Equation 7

the new state, \(\text{state}_{t+\Delta t}\), of the system is determined. From this new state, a new rate is calculated which holds for the next time interval \(\Delta t\), and so on. Much can be said about the 'short period', \(\Delta t\), especially with respect to its relation to the time coefficient of a particular model (see below), and with respect to errors introduced by numerical integration methods (see Section 6.4).

Numerical integration is first applied to the example of the water tank, Equation 3. Assume that the tank is empty at \(t = 0\) s, so \(W_0 = 0\) l, coefficient \(c\) equals 1/4 s\(^{-1}\), and the maximum volume \(W_m = 16\) l. The rate of inflow of water into the tank at \(t = 0\) is calculated from the rate equation

\[
(dW/dt)_t = c \cdot (W_m - W_t)
\]

Equation 8
and is 4 l s\(^{-1}\). If the time interval \(\Delta t\), or the 'short period' equals 2 s, the volume of water at time \(t + \Delta t\) is obtained from the state equation

\[
W_{t + \Delta t} = W_t + \Delta t \cdot (dW/dt)_t
\]

Equation 9

as 8 l. During the next time interval of 2 s, the rate is: \(1/4 \times (16 - 8) = 2\) l s\(^{-1}\). Thus, during this time interval 4 l of water will flow into the tank and the total quantity of water after 4 s equals 8 + 4 = 12 l. The calculations thus proceed according to Equations 8 and 9, and can be represented by the following diagram:

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>(W)</th>
<th>(dW/dt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>2</td>
</tr>
</tbody>
</table>

**Exercise 4**

Complete the numerical calculation and plot the amount of water in the tank against time. Now also calculate analytically the amount of water in the tank using Equation 6, and plot the results in the same graph.

a. What is the difference between the numerical and the analytical solution and what is the reason of this difference?

b. When is the rate of inflow zero?

c. What happens if coefficient \(c\) is 1/8 instead of 1/4?

In the case of the water tank the rate of filling decreases with time (see Figure 4, Equation 3a), so that the numerical integration, where the rate variable is kept constant during the time interval \(\Delta t\), overestimates the amount of water in the tank compared to the analytical solution (Exercise 4). The difference between the value of the state variable obtained by the numerical method and the analytical value will be smaller when \(\Delta t\) is smaller. The lower limit to \(\Delta t\) is set by the technical possibilities (rounding errors) of performing the calculations over long time spans.

**Exercise 5**

The parameter values in Equations 8 and 9 are: \(W_0 = 2\) l; \(W_m = 16\) l; \(c = 1/4\) s\(^{-1}\).

a. Perform numerical integration for the filling of the water tank up to about 30 s using the following time intervals: \(\Delta t = 1 \cdot c^{-1}\); \(\Delta t = 1.5 \cdot c^{-1}\); \(\Delta t = 2 \cdot c^{-1}\); \(\Delta t = 2.5 \cdot c^{-1}\).

b. Plot the results in the graph of Exercise 4.

c. What is the relation between the results of the calculations and the ratio of the time interval and the value of \(c^{-1}\)?

d. What upper limit would you set to this ratio? (Also consider your calculations of Exercise 4).

The upper limit to \(\Delta t\) is determined by the inverse value of coefficient \(c\) in the differential equation. The inverse of \(c\) is called the time coefficient, \(\tau\), which has the dimension time. It is a measure of the reaction rate of a model. In models containing more than one rate variable a first approximation to the time interval is obtained by taking \(\Delta t\) smaller than one tenth of the smallest \(\tau\) in that model. The time coefficient appears equal to the time that would be
needed for the model to reach its equilibrium state, if the rate of change were fixed. This applies to any point on the integrated function, as shown in Figure 5.

Figure 5. The amount of water as a function of time, according to Equation 6, with \( W_0 = 0.1 \), \( W_m = 16 \) and \( c^{-1} = \tau = 4 \) s : \( W = W_m \times (1 - e^{-\tau t}) \). The time interval over which the tangent must be extended to intersect with the equilibrium line is the time coefficient, \( \tau \).

**Exercise 6**

a. Check this last statement by applying Equation 8.

b. The statement also holds for a positive feedback loop, but the formulation is slightly different. Explain this for the case of exponential growth.

**Exercise 7**

a. Calculate the time coefficients when the relative growth rates are 1.5, 0.2 and 0.05 per year, respectively.

b. What time intervals would you use for numerical integration in these cases? Also take into account the practical aspect of numerical calculations.

c. Compute the number of animals after 5 years, when \( c = 0.2 \) and \( A_0 = 100 \), by

1. the analytical solution to the problem, Equation 5;
2. the numerical solution to the problem using Equation 2 and \( \Delta t = c^{-1}/10 \).

d. Make a plot of the results.

e. Explain the difference between the numerical solution and the analytical solution.

A cautionary remark may be appropriate here. The time coefficient is defined as the inverse of the relative growth rate (see Exercise 7). The growth percentage over a fixed period (i.e. the relative increase in the number of animals after one year or annual relative increase, \( ari \)) is often used to calculate \( \tau \), but that gives incorrect results. A growth rate of, for example, 20 % per whole year is not equivalent to an \( rgr \) of 0.2 yr\(^{-1}\). The relative growth rate is less:

when \( A_0 = 100 \), \( A \) equals 120 after one year and Equation 5 can be used to calculate the relative growth rate as follows:

\[ A = 120 = 100 \times e^{rgr \times 1}, \]  
so \( rgr = \ln 1.2 = 0.182 \) yr\(^{-1}\), and \( \tau = 5.48 \) yr instead of \( 1/0.2 = 5 \) yr. The relative growth rate \( (rgr) \) may be expressed in the annual relative increase as follows:

\[ A_0 + A_0 \times ari \times 1 = A_0 \times e^{rgr \times 1} \]  
or \( rgr \times 1 = \ln(1 + ari \times 1) \), or more generally \( rgr \times \Delta t = \ln(1 + ari \times \Delta t) \). For an exponential decline, one can derive
\[ r_{dr} \Delta t = -\ln(1 - r_{ard} \Delta t), \] with \( r_{dr} \) and \( ard \) the relative death rate and the annual relative decrease, respectively. The difference between \( ard \) and \( rgr \), or that between \( ard \) and \( rdr \), will be substantial when the annual relative increase or decrease is high.

Other names for the time coefficient and related concepts are time constant, transmission time (in control-system theory), average residence time, delay time, extinction time and relaxation time. This indicates its significance in various disciplines. Doubling time, the time needed to double an amount, is also used to characterize a system, but it is not synonymous with the time coefficient.

Exercise 8
The relationship between doubling time, \( t(2) \), and the time coefficient in exponential growth is \( t(2) = 0.7 \cdot \tau \). Derive this relationship.

The relaxation time, a term often used in physics, is the time needed in exponential processes to change the state by a factor e; it is equivalent to the time coefficient. For an illustration of average residence time, consider an exponentially decreasing population of animals without birth or migration. Then, the average residence time equals the time coefficient.

Exercise 9
Check this last statement mathematically by using the definition of the average residence time:

\[
-\frac{1}{A_0} \int_0^\infty t \cdot \frac{dA}{dt} \cdot dt = \frac{1}{A_0} \int_0^{A_0} t \cdot dA = \frac{1}{A_0} \int_0^\infty A \cdot dt,
\]

and the analytical equation describing exponential decrease: \( A = A_0 \cdot e^{-t/\tau} \).

The definition states that the average residence time equals the sum or integral (\( \int \)) of the decrease rate at each moment (\( \frac{dA}{dt} \)), weighted by the factor time at each moment (\( t \)) (yielding the dimension of 'number of animals \( \cdot \) time') which is then standardized at a unit number of animals at time zero (\( A_0 \)) (yielding the dimension of 'time'). So, a large number of animals that reside a short time in the area considered contribute differently to the average residence time than the last animals in the area that have resided a rather long period.

In nature many processes occur simultaneously. In simulation models of such processes, however, calculations take place sequentially. But since dynamic simulation is based on the principle that rates of change are mutually independent (i.e. they depend individually on state variables and driving variables), all rates at any one moment can be calculated in series. These rates can then be integrated (sequentially) to obtain the values of the state variables one time interval (\( \Delta t \)) later. In this way the model operates in a semi-parallel fashion, and it simulates simultaneously occurring processes. It is convenient to use special simulation languages to describe parallel processes in a semi-parallel fashion (see Chapter 3). If other computer languages are used, however, this requirement should still be met (see Chapter 10).
2.6 An example

The different steps that may be distinguished in systems analysis of living systems are demonstrated now. A more detailed list of steps is presented in Section 4.5.

Objectives and definition of the system A microbiologist plans to develop a technical system in which yeast can be grown continuously. He wishes to use a vessel of constant volume, through which a sugar solution will flow. To gain insight into the proper technical system parameters, such as the volume of the vessel \((v, \text{ m}^3)\), the required concentration of sugar \((c_s, \text{ kg kg}^{-1})\) and the flow rate of water \((q, \text{ m}^3 \text{ d}^{-1})\), he decides to design a model of the system.

The physiological parameters pertaining to the yeast cannot be adjusted like the technical parameters. Therefore, some experiments are performed which show that the absolute growth rate of the yeast \((\frac{dy}{dt}, \text{ kg d}^{-1})\) is proportional to the amount of yeast present \((y, \text{ kg})\), and to the sugar concentration. At a sugar concentration of 10\%, \(c_{s10}\), the relative growth rate and the amount of sugar in the vessel are termed \(\mu_{10}\) and \(s_{10}\), respectively. The rate of sugar consumption per unit yeast \((s_y, \text{ kg kg}^{-1} \text{ d}^{-1})\) is known. The maximum possible quantity of sugar \((s_m, \text{ kg})\) in the vessel is determined by the incoming sugar concentration and the volume of the vessel.

Exercise 10

The following table gives fictitious data on the amount of yeast as a function of time at different constant sugar concentrations.

<table>
<thead>
<tr>
<th>sugar concentration in water (kg kg(^{-1}))</th>
<th>time (h)</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>yeast (kg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>1950</td>
<td>2119</td>
<td>2304</td>
<td>2958</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>1900</td>
<td>2340</td>
<td>2882</td>
<td>5384</td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>2050</td>
<td>3110</td>
<td>4717</td>
<td>16464</td>
<td></td>
</tr>
</tbody>
</table>

a. Derive the relative growth rate of yeast, \(\mu\), at these four different sugar concentrations.
b. Plot the relative growth rate, in units of day\(^{-1}\), against the sugar concentration, \(c_s\). Express \(\mu\) in terms of \(c_s\), \(c_{s10}\) and \(\mu_{10}\).
c. Rewrite the expression for \(\mu\) in terms of the current amount of sugar, \(s\), and \(s_{10}\).

The relational diagram Figure 6 shows the relational diagram of the model. Note that this figure is constructed from the elementary systems introduced in Section 2.4. For instance, the lower left rate of inflow with the integral of the sugar \((s, \text{ kg})\) is equivalent to the relational diagram of the car from Figure 2, and the upper integral of the yeast with the right rate of outflow represents an exponential decrease. The representation of the model by one integral for \(y\) and one for \(s\) implies that the yeast and the sugar solution are well mixed throughout the vessel. The relational diagram does not contain yeast mortality: only the time coefficient of the vessel \((\tau, \text{ d})\) influences the rate of outflow of yeast. The \(\tau\) of the vessel has a similar influence on the rate of outflow of the sugar, but here a second rate plays a role, as sugar is consumed by the yeast. The time coefficient of the vessel represents the average residence time of yeast and sugar in the vessel, when no sugar consumption would take place. In the real system, this characteristic time can be adjusted, as it is defined as \(\tau = v/q\) (d).
Differential or rate equations  The relational diagram in Figure 6 can help to derive the differential equations. It is immediately clear which variables will appear in a particular rate of flow. For example, for the inflow of yeast:

$$\frac{dy}{dt} = f(y, \mu), \text{ with } \mu = f(\mu_{10}, s_{10}, s); \text{ and for the outflow: } \frac{dy}{dt} = f(y, \tau).$$

This information, combined with information on the proportionalities and the units of variables, yields the net flow rate for yeast:

$$\frac{dy}{dt} = \mu \cdot y - \frac{y}{\tau}$$  \hspace{1cm} \text{Equation 10}

where

$$\mu = \frac{s}{s_{10}} \cdot \mu_{10}$$  \hspace{1cm} \text{Equation 11}

$$s_{10} = c_{s10} \cdot v \cdot 1000$$  \hspace{1cm} \text{Equation 12}

$$\tau = \frac{v}{q}$$  \hspace{1cm} \text{Equation 13}

By analogy, the net flow rate for sugar is

$$\frac{ds}{dt} = c_s \cdot q \cdot 1000 - y \cdot s - \frac{s}{\tau}.$$  \hspace{1cm} \text{Equation 14}

Exercise 11

a. Examine the units of all variables and constants in Equations 10 to 14.
b. What does the number 1000 in Equations 12 and 14 represent?

Further analysis of Equations 10 to 14 To study the dynamic behaviour of the yeast-sugar model, Equations 10 and 14 must be solved by numerical integration. However, a number of model properties can be analysed by studying simplified equations or equilibrium properties.

An example of simplification of equations relates to the time needed to equilibrate a water-filled vessel, which is initially free of sugar, with the sugar solution, in the absence of yeast. The relational diagram for this problem is represented by the lower half of Figure 6 when the outflow of sugar due to consumption by the yeast is omitted. The differential equation for the sugar, when \( y=0 \), is \( ds/dt = c_s \cdot q \cdot 1000 - s/r \), which can be solved analytically. For the condition that at \( t=0, s=0 \), this gives:

\[
s = s_m \cdot (1 - e^{-t/T})
\]

Equation 15

where \( s_m = c_s \cdot v \cdot 1000 \), which is the maximum amount of sugar that can be achieved with given \( c_s \) and \( v \).

Equation 15 is similar in form to Equation 6, when \( W_0 = 0 \), although their differential equations describe quite different systems and express a dynamic and a static flow model, respectively.

Exercise 12

Derive from Equation 15, in general terms, the time needed to reach 95% of the final equilibrium level of sugar in the vessel.

Such equilibrating processes may take a long time when large time coefficients are involved. It is preferable, therefore, to start an experiment by filling an empty vessel with the desired sugar solution.

Exercise 13

How much time, expressed in terms of the time coefficient, is needed to reach 100% of the equilibrium sugar level when an empty vessel is filled at a constant rate with the sugar solution? There is no outflow until the vessel is full.

Equations 10 and 14 can be analysed to determine whether equilibrium levels of yeast and sugar can be reached, and if so, what these levels are. In a dynamic equilibrium, the values of the state variables are constant, and the sum of the rates of inflow is equal to the sum of the rates of outflow. Thus, the net rate of change of the state variable equals zero. In the case of the continuous culture this means that \( dy/dt \) and \( ds/dt \) in Equations 10 and 14, respectively, are equal to zero. The equilibrium levels of sugar and yeast can be calculated from:

\[
s = \frac{s_{10}}{\mu_{10} \cdot \tau}
\]

Equation 16

and
\[
y = \frac{1}{\tau} \cdot (s_m - s).
\]

Equation 17

A special case of dynamic equilibrium is obtained when \( y \) is zero. Such a dynamic equilibrium is established when the yeast culture is washed out because the time coefficient for the vessel is smaller than that for the yeast. This is the case when \( q > c_s \cdot v \cdot \mu_{10}/c_{s10} \).

**Exercise 14**

Derive Equations 16 and 17 from Equations 10 to 14.

Equation 17 shows that the equilibrium level of yeast depends on the time coefficient of the vessel that can be influenced.

The microbiologist will be interested in the combination of manipulable parameters that yields the maximum yeast production with the minimum amount of sugar. The only manipulable variables in Equation 17 are \( s_m \) and \( \tau \). From Equation 16 it follows that \( s \) is a hyperbolic function of \( \tau \), indicating that at very low \( \tau \) values \( s \) will be very large and at high \( \tau \) values \( s \) will be small. So, there is no practical minimum value for the amount of sugar. To investigate whether a maximum exists in the relation of yeast production against the time coefficient, Equation 16 is inserted into Equation 17, which is then differentiated with respect to \( \tau \) to obtain

\[
\frac{dy}{d\tau} = \frac{1}{\tau^2 \cdot s_y} \cdot (\frac{2 \cdot s_{10} - s_m}{\tau \cdot \mu_{10}}).
\]

Equation 18

**Exercise 15**


b. At what value of \( \tau \) is there a maximum or minimum value of \( y \)?

If the second derivative of \( y \) with respect to \( \tau \) is negative at the \( \tau \) value found in Exercise 15b, \( y \) is at a maximum. The second derivative is

\[
\frac{d^2y}{d\tau^2} = \frac{2}{\tau^3 \cdot s_y} \cdot (s_m - \frac{3 \cdot s_{10}}{\tau \cdot \mu_{10}}).
\]

Equation 19

Substitution of the answer from Exercise 15b into Equation 19 does yield a negative value for the second derivative: hence, \( y \) is at its maximum. The maximum yeast level, at that value of \( \tau \), can be calculated from Equation 16 and 17 as

\[
y = \frac{1}{4} \cdot \frac{\mu_{10}}{s_{10} \cdot s_y} \cdot s_m^2.
\]

Equation 20

The amount of sugar is then
Equations 20 and 21 show that at the optimum value of $\tau (= v/q)$, the amounts of yeast and sugar can still be changed by adjusting the inflow concentration of sugar which determines $s_m$.

---

Exercise 16
a. Check the units of Equations 16 to 21.
b. Express the water flux $q$ in terms of the other parameters to calculate the inflow rate resulting in the maximum amount of yeast at a given sugar concentration and volume of the vessel.

2.7 Summary

The terms system, model and simulation were illustrated by introducing three basic elementary systems and their models (car without feedback, exponential growth with positive feedback resulting in an unstable equilibrium, and a water tank with a negative feedback mechanism resulting in a stable equilibrium). The important assumption was that systems and their models are state determined, which means that if the initial conditions and the mathematical description of the rates of change of these state variables are available, the time course of the model can be calculated. In these models state, rate, and driving variables are distinguished. It was shown that the systems could be qualitatively represented by means of relational diagrams where the major state, rate and driving variables and their interrelationships and feedbacks are given.

Analytical integration can sometimes be used to obtain the desired time course of models. These solutions are exact because the time increment $dt$ is infinitely small. However, analytical solutions are restricted to rather simple equations or at least to simple boundary conditions and one needs much skill to use these methods.

Numerical integration can always be used to solve differential equations, however. In this case the time increment of integration, $\Delta t$, is finite and the ratio between $\Delta t$ and the characteristic time in which the system significantly changes, i.e. the time coefficient ($\tau$) of the model, should be chosen so that the solution obtained is sufficiently close to the analytical solution, if it would exist. As a rule of thumb the ratio of $\Delta t/\tau$ is taken as 1/10. Numerical integration methods provide a powerful tool to solve differential equations, also to those who have no strong background in mathematics, provided that one comprehends the calculation procedure given in Equation 7 and the important remark in Section 2.5 about mutual independency of rates and semi-parallel integration of simultaneously occurring processes.

An example about yeast growth was used to illustrate a number of aspects introduced in earlier sections. Furthermore, an analysis of a set of differential equations in terms of equilibrium conditions was given. In this context static and dynamic equilibrium were mentioned. Dimensional analysis was found to be very important in developing and understanding equations.

Though the presented techniques may seem very simple, they are extremely powerful in model building of ecosystems. Most models given in the following chapters are composed of the elementary feedback loops, while the above simple mathematical techniques suffice to solve them. Analysis of and solutions to more complex problems require especially more knowledge about the relationships that may characterize system structure, rather than sophistication of mathematical techniques. Such knowledge forms at present the major restriction to systems analysis and simulation of ecological processes, but it forms also the major challenge of future research.
3 A simulation language: Continuous System Modeling Program III

P.A. Leffelaar

3.1 Introduction

CSMP stands for Continuous System Modeling Program, version III. It has been extensively described in the Program Reference Manual by IBM Corporation, 1975. In this chapter only the essentials of CSMP will be discussed. However, these essentials enable us already to solve many problems.

CSMP is a non-procedural language, which means that the user can write programs in a conceptual order, and that CSMP will sort the statements in a computational order. This implies on the one hand that CSMP takes care of the principle of systems analysis that the rates of change are calculated when the states are known and thus that parallel processes are described in a semi-parallel fashion (see Section 2.5), and on the other hand that the user is allowed to start thinking and programming at a high level of integration while descending to the details later on.

CSMP is a dynamic simulation language primarily designed to integrate rate equations to obtain the state of the model as a function of time. To this purpose CSMP takes care of the integration scheme and keeps track of the independent variable time. CSMP also provides a number of numerical integration routines which are easy to use. In this chapter and in part A of this book, however, only the rectangular method will be dealt with. This approach helps the student to first gain a thorough feeling of the important aspects of the relation between the time step in numerical integration and the different characteristic times or time coefficients in the model.

Furthermore, CSMP provides special functions (e.g. interpolation), data input is simple, and tabular- and/or graphical output can be obtained by just listing the variables on a special label.

As the source program of CSMP is written in FORTRAN (FORmula TRANslation), FORTRAN statements as well as all FORTRAN library functions and libraries written in FORTRAN (e.g. IMSL, 1987; Press et al., 1986) can be used in more advanced models. The possibility to use FORTRAN in combination with CSMP is a major reason to use CSMP as a simulation language. The powerful aspects of using FORTRAN within CSMP models will be postponed to Chapter 7 and Chapter 10.

The general design of a CSMP program is given in Figure 7. It will be used in the following sections to illustrate the structure of CSMP programs, and to indicate where to place the state, rate and auxiliary equations, the data for input and output, the method of integration and the timer. It is also used to show how reruns may be invoked.

3.2 The structure of the model

Figure 7 shows that a program should begin with a TITLE label containing a short identification of the program. The first letter of the title should be in upper case. In a CSMP program 3 segments can be distinguished: INITIAL, DYNAMIC and TERMINAL. These statements (labels) indicate that the computations must be performed before, during and after a simulation run, respectively. If one is using these segments, then each segment label closes the former segment. The entire program is terminated by the statements END, STOP and ENDJOB, each on a separate line. END completes the specifications of the model, STOP terminates the simulation run, and ENDJOB terminates the job. The label ENDJOB must start in the first position.

In the INITIAL segment computations are executed only once per run. The use of the segment is optional. The INITIAL segment can be used to give values to the input data and to the time variables, and to define data output and
TITLE General design of a CSMP program

INITIAL

* Data input:
INCON STAT10 =... , STAT20 =...
CONST PI =3.1416 , ...... =...
PARAM ...... =... , ...... =...
FUNCTION DRV1TB = (,...,...), (,...,...), (,...,...), ...

* Data output:
PRINT STATE1 , STATE2 , NRATE1 , NRATE2 
OUTPUT STATE1 , STATE2 , NRATE1 , NRATE2 
PAGE NTAB=0 , GROUP=2 , WIDTH=80

* The independent variable time:
TIMER TIME=..., DELT =..., PRDEL=..., OUTDEL=..., FINTIM=...

* The integration method:
METHOD RECT

* Initial calculations:

DYNAMIC

* State variables:
STATE1 =INTGRL(STAT10 , NRATE1)
STATE2 =INTGRL(STAT20 , NRATE2)

* Driving and rate variables:
DRV1 =AFGEN(DRV1TB,TIME)
NRATE1 =f(STATE1 , DRV1 , TAU...) 
NRATE2 =f(STATE1 , STATE2 , TAU...) 
VAR2 =AFGEN(VAR2TB,.......)

TERMINAL

* Final calculations:

* Reruns may be defined between 'END' and 'STOP':
END
INCON ..., ...
PARAM ..., ...
FUNCTION DRV1TB=
TIMER ...
END
STOP
ENDJOB

Figure 7. General design of a CSMP program. Bold characters are used to highlight the 3 main program segments that may be distinguished and the beginning and termination of the program; italic characters preceded by an asterisk are used for comments. Regular programs do contain one character type only. Note that an equal-sign means 'is to be replaced by' rather than 'is equal to'.

24
the integration method of the model. Furthermore, the computation of results which are used as input for the dynamic section of the program may be executed here. In the DYNAMIC segment the time course of the state variables is calculated. This segment is therefore executed many times (roughly the simulation period divided by the time step). It is normally the most extensive segment in a model. The segment contains the complete description of the model dynamics, together with any other computation required during the simulation. For some models the program consists of just the DYNAMIC segment. Then, the segment may be declared explicitly by the label DYNAMIC, but, if no INITIAL and TERMINAL segments are identified, one can omit it. The TERMINAL segment can be used for computations and specific output that can only be obtained at the end of the simulation run. This could be a computation based on the final values of one or more variables. As in the INITIAL segment, the computations are executed only once. Also the TERMINAL segment is optional.

CSMP is provided with a sorting algorithm to free the user from the task of correctly sequencing the statements. A program is correctly sorted when all input variables to the right hand side of an equation are known. The statements in the INITIAL and DYNAMIC segments are placed in computational order by CSMP. The statements in the TERMINAL segment, however, have to be sorted by the user.

### 3.3 Integration of rate equations

In Figure 7 the state variables are placed directly after the DYNAMIC label according to the assumption that rates depend at each moment on the values of state and driving variables and that thus the state of each system and the driving variables should be quantified before the rates of change can be calculated. A state variable, e.g. STATE1 in line 21 in Figure 7, is the output of the structure statement INTGRL, in which the net rate of change NRATE1 is integrated over time. To calculate the rate at time zero, the state should be known at that moment. Therefore, the initial state should be supplied, i.e. the first argument in the INTGRL statement (STAT0 in line 21). Usually this will be measured data. The mathematical equivalent of line 21 is

\[ y = y_0 + \int_{t_0}^{t} \frac{dy}{dt} \, dt, \]  

Equation 22

where \( y_0 \) and \( \frac{dy}{dt} \) are the initial value and the net rate of change of the state variable: STAT0 and NRATE1 in line 21.

The driving variables that characterize the influence of the outside world on the system should follow the state variables in Figure 7, so that these data are synchronously available as inputs for the rate calculations. Then, the rate equations can be defined.

The INTGRL statement sets up the integration scheme, but it does not specify which numerical integration method should be used to solve the rate equations. This is done by the METHOD statement. The INTGRL and METHOD statements together fully provide for numerical integration. Although CSMP offers a number of integration methods, only the rectangular method is used in Part A of this book. In CSMP this is invoked by: METHOD RECT (line 17). More sophisticated and accurate integration methods and their selection will be introduced in Chapter 6.

Obviously, the state, rate and driving variables vary in time and should be placed in the DYNAMIC segment of the model. The integration method, however, has to be communicated to the program only once, and can thus be placed in the INITIAL.

The sequence of statements discussed here would also be obtained by the sorting algorithm of CSMP. Sequencing
is discussed, however, because it is good to develop a feeling for how the calculational principle of systems analysis that the rates of change can be calculated when the states are known may directly be reflected in simulation programs. Moreover, statement sequencing in programs often improves their comprehensibility and it facilitates the transition to simulation in FORTRAN, which is dealt with in Chapter 10.

3.4 The time loop

Time is the independent variable in simulation models. After each integration this variable needs to be incremented with the time step, \( \Delta t \), used for the numerical integration. CSMP updates time automatically. The user has only to specify a number of so-called timer variables. These include the start time of the simulation (TIME), the time interval of integration (DELT), the time intervals for tabular (PRDEL) or plotted (OUTDEL) output, and the total simulation period (FINTIM). All of these variables are placed on the TIMER statement (Figure 7, line 15). The timer label has to be communicated to the program only once. Therefore, it is placed in the INITIAL segment.

Exercise 17

Reason that time could be kept track of by the statement: \( T = \text{INTGRL}(0.,1.) \).

3.5 Data input and output

Data input, Figure 7 line 4 to 9, concerns the numerical values of the initial conditions (INCON) of the state variables, the constants (CONST), the parameters (PARAM), and the functions (FUNCTION). The INCON, CONST, and PARAM labels could be used interchangeably, but it is recommended to reserve the label INCON for the values of the state variables at time zero, the label CONST for physical constants, e.g. \( \pi \), the gas constant, Avogadro's number, etc., and the label PARAM for parameters that may differ for each simulation run. Any relationship between \( y \) and some independent variable \( x \), may be communicated to the program by the label FUNCTION. The data pairs are given as numerical values of \((x_1,y_1), (x_2,y_2)\), etc., where \( x \)-values should increase monotonically. The function name, for instance DRV1TB (DRiving Variable 1-TaBle) in line 8 of Figure 7, is later referred to by the selected interpolation method (line 24). Different interpolation methods are discussed in Section 3.6.

Numerical values for the initial conditions of the state variables, constants, parameters, and functions can also be calculated in the initial segment and then stored in the desired data type.

Data output, Figure 7 line 10 to 13, is invoked by listing the variables on the label PRINT or OUTPUT. With the PRINT label the numerical values of up to 55 real variables (see Section 3.8) may be printed in a tabular form at each PRDEL interval. The PRINT label can be used only once in a CSMP program: a second label would override the first. With the OUTPUT label the numerical values of up to 5 real variables may both be printed in tabular form and plotted at each OUTDEL interval. When the number of OUTPUT variables exceeds 5, graphical output is suppressed and printed output (of up to 55 variables) is given alone. More than one OUTPUT label may be used. The output document can easily be modified by the PAGE label. This is illustrated in line 13 of Figure 7: PAGE NTAB=0, GROUP=2, WIDTH=80. This line would cause suppression of the tabular output (NTAB=0) of the variables listed on the OUTPUT label in favor of the largest possible resolution in the graphical output. Furthermore, the first two variables listed would be plotted at the same scale (GROUP=2) while the remaining variables are plotted on their own scales, and the width of the plot is 80 columns (WIDTH=80). The WIDTH label may be used if your computer terminal cannot be set to a width of 132, which is the maximum value of the paper width. Other practical
PAGE parameters are NPLOT and LOG. NPLOT can be used to indicate the number of variables that have to be plotted and tabulated (with a maximum of 5), while the remaining variables will be tabulated only. LOG=n specifies that the first n variables in the output group have to be plotted on a logarithmic scale.

Although CSMP sorts the statements, the place of the PAGE label may affect the output result: a PAGE statement following an OUTPUT statement is assigned to that statement; a PAGE statement preceding all the OUTPUT statements applies to the whole group. The OUTPUT statement is clearly more flexible than the PRINT statement. The practical value of the PRINT statement remains, however, because many applications only require tabular output.

Both the input and output data have to be communicated to or calculated in the program only once. Therefore, they are preferably placed in the INITIAL segment.

3.6 Interpolation

The data pairs given on a FUNCTION label (see Section 3.5), may be interpolated in different ways. The simplest way of interpolating between two points is to connect them by a straight line:

\[ y = y_1 + (x - x_1) \cdot \frac{(y_2 - y_1)}{(x_2 - x_1)} \]  

Equation 23

Exercise 18

Draw a graph with an x- and y-axis and two data pairs \( x_1, y_1 \) and \( x_2, y_2 \), and derive Equation 23.

In CSMP, linear interpolation between data pairs is obtained by means of the AFGEN function (Arbitrary Function GENERator):

\[ Y = \text{AFGEN}(YTB, X) \]

where \( Y \) is the result of interpolating the relationship between the \( y \)- and \( x \)-values, supplied to the program in function \( YTB \), with the independent variable \( x \) as input. Note that in the function definition the independent variable comes first in each data pair, whereas in the interpolation function (AFGEN) the independent variable comes as second argument.

Non-linear interpolation of data pairs is also possible. Then, more than 2 data pairs are required, e.g. three pairs to calculate a parabola: \( y = a \cdot x^2 + b \cdot x + c \), or four points to calculate a cubic relation: \( y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d \). In general one can calculate a \( n \)th order function through \( (n+1) \) data points. The linear, parabolic and cubic interpolation methods are often called first, second and third order interpolation methods, respectively.

Exercise 19

Given is the function

\[ \text{FUNCTION} \quad \text{YTB} = (0.,1.) , (1.,1.) , (1.5,0.) , (3.,0.) , (5.,0.) . \]

Plot the data pairs of the function on a graph, and draw the two parabolae which can be calculated through the pairs \((0.,1.),(1.,1.),(1.5,0.)\) and \((1.,1.),(1.5,0.),(3.,0.)\).
In CSMP, parabolic interpolation between data pairs is obtained by means of the NLFGEN function (Non Linear Function GENerator):

\[ Y = \text{NLFGEN}(YTB, X) \]

Higher order interpolation between data pairs may be obtained by means of the FUNGEN function (FUNction GE-Nerator):

\[ Y = \text{FUNGEN}(YTB, \text{ORDER}, X) \]

where the integer variable ORDER (see Section 3.8) should have a value of 1 to 5. When the variable ORDER equals 1 or 2, the AFGEN or NLFGEN functions are generated, respectively.

Now a number of possibilities of data interpolation are known, the best method should be chosen. Therefore, first plot the data that should be interpolated. The data will often show some scatter and may be approximated by a smooth curve, which could be represented by short straight lines. Linear interpolation is adequate then. Sometimes, a parabolic relationship is expected a priori, e.g. from theoretical considerations. This could be a reason to use parabolic interpolation, but it would also be possible to fit a parabolic or another appropriate mathematical equation to the data. Interpolation is no longer needed then, because the data could be calculated from the auxiliary equation.

Equations are to be preferred to arbitrary functions, since generally they give more insight in the relationship between the data. If interpolation is applied and the data pairs are close enough to be connected by straight lines, especially linear interpolation is recommended, as no unexpected results occur.

Exercise 20

The following CSMP program demonstrates the use of the AFGEN and NLFGEN functions:

```
TITLE Linear and parabolic interpolation
INITIAL
FUNCTION YTB = (0.,1.), (1.,1.), (1.5,0.), (3.,0.), (3.5,0.)
OUTPUT Y1, Y2
PAGE GROUP, WIDTH = 80
TIMER FINTIM = 3.5, OUTDEL = 0.1, DELT = 0.1
METHOD RECT

DYNAMIC
Y1 = AFGEN (YTB, TIME)
Y2 = NLFGEN (YTB, TIME)

END
STOP
ENDJOB
```

a. Run the program and study its output. Compare the results with your calculations of Exercise 19.

b. Which data pairs are used for parabolic interpolation in the interval \(0 \leq t \leq 1\), and which in the subsequent time intervals?

A selection of a number of other useful functions available in CSMP is described in Tables 1 and 2.
Table 1. Some CSMP functions.

<table>
<thead>
<tr>
<th>CSMP III Functions</th>
<th>Equivalent Mathematical Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrator</strong></td>
<td></td>
</tr>
<tr>
<td>$Y = \text{INTGRL}(IC, X)$</td>
<td>$y(t) = \int_{t_0}^{t} x , dt + y(t_0)$</td>
</tr>
<tr>
<td>where: $IC = y_{t_0}$</td>
<td>where: $t_0 =$ start time</td>
</tr>
<tr>
<td>$t =$ time</td>
<td></td>
</tr>
<tr>
<td><strong>Arbitrary function generator</strong></td>
<td></td>
</tr>
<tr>
<td>linear interpolation</td>
<td></td>
</tr>
<tr>
<td>$Y = \text{AFGEN}(\text{FUNCT}, X)$</td>
<td></td>
</tr>
<tr>
<td>quadratic interpolation</td>
<td></td>
</tr>
<tr>
<td>$Y = \text{NLFGEN}(\text{FUNCT}, X)$</td>
<td></td>
</tr>
<tr>
<td><strong>Modulo function</strong></td>
<td></td>
</tr>
<tr>
<td>$Y = \text{AMOD}(X, P)$</td>
<td></td>
</tr>
<tr>
<td>$n$ is an integer value such that</td>
<td></td>
</tr>
<tr>
<td>$0 \leq y &lt; P$</td>
<td></td>
</tr>
<tr>
<td><strong>Limiter</strong></td>
<td></td>
</tr>
<tr>
<td>$Y = \text{LIMIT}(P1, P2, X)$</td>
<td></td>
</tr>
<tr>
<td>$y = P_1$ ; $x &lt; P_1$</td>
<td></td>
</tr>
<tr>
<td>$y = P_2$ ; $x &gt; P_2$</td>
<td></td>
</tr>
<tr>
<td>$y = x$ ; $P_1 \leq x \leq P_2$</td>
<td></td>
</tr>
<tr>
<td><strong>Not</strong></td>
<td></td>
</tr>
<tr>
<td>$Y = \text{NOT}(X)$</td>
<td></td>
</tr>
<tr>
<td>$y = 1$ if $x &lt; 0$</td>
<td></td>
</tr>
<tr>
<td>$y = 0$ if $x &gt; 0$</td>
<td></td>
</tr>
<tr>
<td><strong>Input Switch Relay</strong></td>
<td></td>
</tr>
<tr>
<td>$Y = \text{INSW}(X1, X2, X3)$</td>
<td></td>
</tr>
<tr>
<td>$y = x_2$ if $x_1 &lt; 0$</td>
<td></td>
</tr>
<tr>
<td>$y = x_3$ if $x_1 &gt; 0$</td>
<td></td>
</tr>
</tbody>
</table>
Table 1. Continued.

<table>
<thead>
<tr>
<th>Dead time (DELAY)</th>
<th>Equivalent Laplace Transfer Function: $Y(s) = \frac{e^{-ps}}{X(s)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y = \text{DELAY}(N, P, X)$</td>
<td>$y = x(t-p)$; $t &gt; p$</td>
</tr>
<tr>
<td>where: $P =$ delay time</td>
<td>$y = 0$; $t &lt; p$</td>
</tr>
<tr>
<td>$N =$ number of points sampled in</td>
<td>$\text{Equivalent Laplace Transfer Function: }$</td>
</tr>
<tr>
<td>interval $p$ (integer constant) and</td>
<td>$Y(s) = \frac{e^{-ps}}{X(s)}$</td>
</tr>
<tr>
<td>must be $\geq 3$; and $&lt; 16,378$</td>
<td></td>
</tr>
<tr>
<td>Implicit function</td>
<td></td>
</tr>
<tr>
<td>$Y = \text{IMPL}(\text{IC}, P, \text{FOFY})$</td>
<td></td>
</tr>
<tr>
<td>where: $\text{IC} =$ first guess</td>
<td></td>
</tr>
<tr>
<td>$P =$ error bound</td>
<td></td>
</tr>
<tr>
<td>$\text{FOFY} =$ output name from</td>
<td></td>
</tr>
<tr>
<td>final statement in algebraic loop</td>
<td></td>
</tr>
<tr>
<td>definition</td>
<td></td>
</tr>
<tr>
<td>Impulse generator</td>
<td></td>
</tr>
<tr>
<td>$Y = \text{IMPULS}(P_1, P_2)$</td>
<td></td>
</tr>
<tr>
<td>where: $P_1 =$ time of first pulse</td>
<td></td>
</tr>
<tr>
<td>$P_2 =$ interval between pulses</td>
<td></td>
</tr>
<tr>
<td>$y = 0$; $t &lt; p_1$</td>
<td></td>
</tr>
<tr>
<td>$y = 1$; $(t-p_1) = kp_2$</td>
<td></td>
</tr>
<tr>
<td>$y = 0$; $(t-p_1) \neq kp_2$</td>
<td></td>
</tr>
<tr>
<td>$k = 0, 1, 2, 3, \ldots$</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

```
\begin{align*}
  y &= \begin{cases}
    0 & t < p_1 \\
    1 & (t-p_1) = kp_2 \\
    0 & (t-p_1) \neq kp_2
  \end{cases} \\
  k &= 0, 1, 2, 3, \ldots
\end{align*}
```

$\hat{y}$
Table 2. Some FORTRAN functions, which can be used in CSMP statements.

<table>
<thead>
<tr>
<th>FORTRAN Functions</th>
<th>Equivalent Mathematical Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td></td>
</tr>
<tr>
<td>( Y = \exp(X) )</td>
<td>( y = e^x )</td>
</tr>
<tr>
<td>Trigonometric sine (argument in radians)</td>
<td></td>
</tr>
<tr>
<td>( Y = \sin(X) )</td>
<td>( y = \sin(x) )</td>
</tr>
<tr>
<td>Trigonometric cosine (argument in radians)</td>
<td></td>
</tr>
<tr>
<td>( Y = \cos(X) )</td>
<td>( y = \cos(x) )</td>
</tr>
<tr>
<td>Square root</td>
<td></td>
</tr>
<tr>
<td>( Y = \sqrt{x} )</td>
<td></td>
</tr>
<tr>
<td>Largest value (Real arguments and output)</td>
<td></td>
</tr>
<tr>
<td>( Y = \text{AMAX1}(X1, X2) )</td>
<td>( y = \max(x_1, x_2) )</td>
</tr>
<tr>
<td>Smallest value (Real arguments and output)</td>
<td></td>
</tr>
<tr>
<td>( Y = \text{AMIN1}(X1, X2) )</td>
<td>( y = \min(x_1, x_2) )</td>
</tr>
</tbody>
</table>
3.7 Reruns

If a simulation run is to be repeated with new input data and/or execution control statements (e.g. TIMER data),
the new data could be given between two END-statements (Figure 7, lines 31 to 36). This feature enables the user to
do a number of simulation runs, without having to load the program structure in the computer again and again. All
data enumerated in Figure 7 under the headings Data input, Data output, The independent variable time, and The
integration method may be adapted in a rerun. Data that are not changed in a rerun specification will retain their or­
ginal values. Only data and not structure (e.g. rate equations) can be changed in a rerun.

3.8 Some elements of CSMP

**Numeric constants** There are two types of constants: real and integer. Real (floating-point) constants are numbers
written with a decimal point, with a maximum of 7 digits. A real constant may be followed by a decimal exponent
written as the letter E, followed by a signed or unsigned one- or two- digit integer constant. The decimal exponent E
format forms a real constant that is the product of the real constant portion multiplied by 10 raised to the desired po­
wer. E.g. 213.15 = 2.1315E2 = 2131.5E-01. Real constants are restricted to a total of 12 characters. Integers are
whole numbers with a maximum of 10 digits without a decimal point.

**Variables** The name of a variable can contain one to six characters and the first character must be a letter. No
blanks or special characters (e.g. + – * / = : ( ) $ , . ) are allowed in a name. For so-called 'reserved words' one is
referred to the Program Reference Manual (IBM Corporation, 1975). All names of labels, functions and data state­
ments are reserved words. In CSMP programs, from the label TITLE to the label STOP, all variables are declared
real automatically. When using integer variables in a program, these variables should be explicitly declared at the
beginning of the program by the label FIXED (Figure 7 line 3). For example, the integer variable ORDER in the
FUNGEN function in Section 3.6 should appear on the FIXED label.

**Units** With respect to the use of units of measurement in software, it is advised to use SI units (m, s, g, mol,
Pa). If required, separate conversion routines can be written to provide other commonly-used units of measurement as
the program’s output.

**Operators** The operators and the hierarchy of the operations are similar to FORTRAN, on which CSMP is based:

( ) grouping of variables and/or constants 1st
** exponentiation 2nd
* multiplication } 3rd
/ division
+ addition } 4th
– subtraction
= replacement 5th

Functions and expressions within parentheses are always evaluated first. For operators of the same hierarchy, the
component operations of the expression are performed from left to right. There is an exception for exponentiation,
where the evaluation is performed from right to left. Thus, the expression A**B**C is evaluated as A**B**C.

**Number of positions on a line** All executable statements should be positioned in the first 72 positions of a line.
Positions 73 to 80 may be used for comments.

**Statement continuation** To continue statements or functions on the next line, it should be followed by typing
three dots (...) on the line to be continued (Figure 7, lines 8-9). The last dot may appear in the 72th position.

**Comments** Explanatory remarks may be given in the programs following an asterisk (*) (Figure 7, lines 4, 10,
14, 16, etc.).
3.9 Syntax

Some syntactic rules are proposed to help increase readability of programs. These are:
- split up your program into an INITIAL, a DYNAMIC and if necessary, a TERMINAL segment;
- lump all parameter specifications at the beginning of the INITIAL, to have a easy overview of them;
- place all INTGRL statements together at the beginning of the DYNAMIC;
- use blanks (spaces) in your equations to line up e.g. equal-signs (=) and to distinguish between data pairs in FUNCTION statements;
- use blank lines and ***-lines to distinguish between different program parts;
- make short comments in your program at the proper place;
Most of these syntactic rules have been applied in Figure 7.

3.10 Concluding remarks

Although the essentials of CS:MP that were described in this chapter look simple, they enable us to solve complicated rate equations without a thorough knowledge of numerical mathematics. Even when one would not know what time coefficients are involved, it is possible to get trustworthy solutions by the trial and error method described in Exercise 24b. The CS:MP algorithm is a so-called explicit scheme. This means that new values of the state variables can be calculated from the current values, the driving variables, and the parameters involved. (This complies with the basic assumption in systems analysis as mentioned in the introduction of this Chapter.) Contrary to explicit schemes are the implicit schemes, where a set of \( n \) equations with \( n \) unknowns need be solved simultaneously at each time step, often with iterative methods. Explicit schemes are more easily developed than implicit schemes, and one can thus focus on the representation of ecological problems in terms of rate equations, rather than on the mathematical necessities as involved in implicit schemes.

In the rectangular integration method the time step, \( \Delta t \), is fixed and based on the smallest time coefficient, \( \tau \), in the model. This implies that slow processes (with large time coefficients) are integrated with a higher accuracy than fast processes (with small time coefficients). Moreover, since the time coefficient that determines the time step often varies in time (see e.g. Exercise 24), the accuracy of the calculations in the model also varies, because this is a function of the ratio \( \Delta t/\tau \). It would be more appropriate when the accuracy of the calculations would be constant, by adapting the time step of integration. Integration methods that achieve this are discussed in Chapter 6.

Exercise 21

a. Write a CS:MP program for the 'animals' example from Chapter 2 (Figure 1, Equation 2). Take the relative growth rate (RGR) equal to 0.1, and FINTIM=10.0.
b. What is the time coefficient of the model, and what time increment of integration (DELT) would you select?
c. Run the program and study its output.

Exercise 22

Write a CS:MP program for the 'reservoir' example from Chapter 2 (Figure 1, Equation 3). Take TC=4.0, the maximum water level (WMAX) as 16 and use FINTIM approximately 3 times larger than TC: in 3 times TC, approximately 95 percent of the maximum is reached in this proportional process (compare the result of Exercise 12 from Chapter 2).
Exercise 23

a. Write a CSMP program for the 'car' example in Chapter 2 (Figure 1, Equation 1). Use instead of a constant rate, the following function of speed (RDIST, km h⁻¹) against the independent variable of time (h):

\[
\text{FUNCTION RDIST} = 0.0, 80.0, 2.0, 80.0, 3.0, 120.0, 5.0, 120.0, ... \\
\quad 6.0, 40.0, 10.0, 40.0
\]

b. Plot the function. How would you select the size of the integration interval to properly read the rate function?

c. How would you use the graph to manually calculate the path travelled?

Exercise 24

a. Write a CSMP program for the 'yeast sugar' example from Section 2.6. Try to find reasonable parameter values, and initialize the state variables at 1/10 of the equilibrium values that will be attained in due time.

b. Estimate the time interval for rectangular integration to solve this set of rate equations.

c. Run the program and study its results. Does the model reach an equilibrium, as can be calculated from Equations 16 and 17 of Chapter 2?

Exercise 25

a. Write a CSMP program for the 'animals' example from Chapter 2 (Figure 1, Equation 2), in which the relative growth rate is now a function of temperature:

<table>
<thead>
<tr>
<th>TEMP (°C)</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGR (h⁻¹)</td>
<td>0</td>
<td>0.08</td>
<td>0.16</td>
<td>0.21</td>
<td>0.24</td>
<td>0.25</td>
</tr>
</tbody>
</table>

The temperature is given by a sine function that depends on time:

\[
\text{TEMP} = \text{AVTMP} + \text{AMPTMP} \times \sin( (2\pi/24.0) \times \text{TIME} ),
\]

where time (TIME) is expressed in hours, and the mean (AVTMP) and amplitude (AMPTMP) of the temperature amount to 20 °C and 10 °C, respectively. The angular frequency of the temperature wave, often denoted by \( \omega \), is defined as \( 2\pi / t_c \), with \( t_c \) the time required for the wave to make a complete cycle of \( 2\pi \) radians. Omega (\( \omega \)) is given here as the ratio \( 2\pi/24.0 \).

b. What is the maximum and the minimum time coefficient in this model for the given parameters?

c. What is the dimension of the argument of the sinus function?

d. Estimate from b) the time interval for rectangular integration to solve the rate equation. Does this time interval nicely follow the sinus function?

e. Run the program for two days and study its output.
4 The growth of yeast

C.T. de Wit and J. Goudriaan

4.1 Description of the system

Growth is only exponential as long as the relative growth rate remains constant. This is usually so with yeast when it is grown under aerobic conditions with a sufficient supply of sugar and some other growth essentials. The sugar is then continuously consumed to provide the 'C skeletons' and the energy for the growth of new yeast cells and for maintenance of the yeast. The end-products, CO₂ and H₂O, of the sugar broken down in the respiratory process do not pollute the environment of the yeast. However, if yeast grows under anaerobic conditions, one end-product of the respiratory processes is alcohol which may accumulate in the environment. This slows down and ultimately stops the development of yeast buds even when there is still enough sugar available for growth.

Growth curves for yeast that result under such conditions are given in Figure 8. It should be noted that yeast once formed does not die because only the bud formation is affected by the alcohol. Two of the four growth curves are from an experiment of Gause (1934) with monocultures of the yeast species Saccharomyces cerevisiae and Schizosaccharomyces 'Kephir'. It is obvious that the initial relative growth rate and the maximum volume of yeast that is ultimately formed is highest for the first species.

Gause cultivated both yeast species not only in monoculture, but also in mixture. The results of this experiment are also presented in Figure 8 by the other two curves. A comparison of the growth of both species in mixture with their growth in monoculture shows that both affected each other in the first situation. It was proposed by Gause that this was due to the formation of the same waste product, alcohol, that affected the bud formation of both species. In this Chapter we shall analyse whether this explanation is acceptable by constructing a model that simulates the growth of two species independently and in mixture under the assumption that the production of the same harmful waste product is the only cause of interaction.
4.2 Relational diagram

The relational diagram for the yeast system is presented in Figure 9. There are three state variables; the amount of the first and second yeast species and the amount of alcohol. The lines of information flow show directly that the growth of yeast is supposed to depend on the amount of yeast, a relative growth rate and an auxiliary variable: a reduction factor. This reduction factor, in its turn, is given as a function of the amount of alcohol that is present. The relations are, of course, the same for both yeast species although numerical values of parameters and functions may be different. The amount of alcohol increases by the rate of alcohol production of both species. The alcohol production of each species is supposed to depend on the growth rate of the species and on an alcohol production factor.

Exercise 26

In Section 2.5 it is said that rates do not depend on each other in state determined systems. Why is the line of information flow between the rate of growth and the rate of alcohol production not in contradiction with this principle?

Figure 9. A relational diagram for the growth and interference of two interfering yeast species.

Relational diagrams should contain as few details as possible, otherwise they are very difficult to grasp and so defeat their purpose. In studying them, much emphasis should be given to aspects that are not incorporated. For instance, in the present scheme there are no loops that relate the alcohol production directly to the amount of yeast, indicating that the cost of maintenance of yeast cells is not accounted for. The amount of sugar is also not considered, because it is assumed to be always available in sufficient amounts.
Exercise 27
Incorporate the aspect of limited food supply in the relational diagram.

Exercise 28
Compare the relational diagram of the continuous yeast culture fed by a sugar solution (Section 2.6, Figure 6) with the one for the growth and interference of two interfering yeast species (Figure 9), and note two principal differences between the models.

4.3 Simulation

The growth of the first yeast species (*Saccharomyces*) is now simulated by stating that the amount of yeast equals

\[ Y_1 = \text{INTGRL} (IY_1, RY_1) \]

in which \( \text{INCON} \ IY_1 = 0.45 \) is the initial amount of yeast in the arbitrary units, used by Gause, and the rate of yeast growth is given by

\[ RY_1 = RGR1 \cdot Y_1 \cdot (1 - \text{RED}_1) \]

The relative growth rate is defined with \( \text{PARAMETER} \ RGR1 = \ldots \).

It was observed by Gause that in both species the formation of buds was completely stopped at some maximum alcohol concentration which is given as a percentage by \( \text{PARAMETER} \ MALC = 1.5 \).

The dependence of the reduction factor on the alcohol concentration may now be obtained with an arbitrary function generator: \( \text{RED}_1 = \text{AFGEN(RED}_1\text{T}, \ ALC/MALC) \). The most elementary assumption is that bud formation decreases linearly with increasing alcohol concentration, which is introduced with \( \text{FUNCTION} \ RED1\text{T} = (0.,0.),(1.,1.) \).

Exercise 29
Express \( \text{RED}_1 \) directly in \( \text{ALC} \) and \( \text{MALC} \) without using the function generator.

The alcohol concentration itself is the integral of the alcohol production rate which is zero at the initialization of growth:

\[ \text{ALC} = \text{INTGRL} (\text{IALC}, \text{ALCP}_1) \]

\( \text{INCON} \ IALC = 0. \)

and the alcohol production rate is proportional to the growth rate of yeast:

\[ \text{ALCP}_1 = \text{ALPF}_1 \cdot \text{RY}_1 \]

Two values need to be determined now: the relative growth rate and the alcohol production factor. During the early stages of growth, \( \text{RED} \) is practically zero, so that the growth rate is equal to \( RGR1 \cdot Y_1 \). This allows a first estimate of \( RGR1 \) from the data in Figure 8 for the monoculture. \( \text{ALPF}_1 \) follows from the observation that growth was terminated when the alcohol concentration equalled 1.5 percent and the amount of yeast about 13 units.
Exercise 30
a. What is a first estimate of RGR1 in the correct units, and how would you estimate the time step of integration?
b. What is the value of ALPF1 in the correct units?
c. Is this value only physiologically determined or does it also depend on the volume of water in the vessels with yeast?
d. What is the value of IALC when not only the initial amount of yeast is introduced at initialization, but also the corresponding amount of alcohol?
e. Estimate the same values for *Schizosaccharomyces*, when it is known that the alcohol concentration at which the formation of buds is completely inhibited is also 1.5 percent.
f. Which species has the larger alcohol production factor?

The structural equations that describe the growth of the second species (*Schizosaccharomyces*) are, of course, the same as those for the first, so that in a model for concurrent growth it suffices to write them twice: once with a 1 at the end of the relevant symbols and once with a 2. The equation that describes the alcohol concentration now becomes

\[ \text{ALC} = \text{INTGRL} (\text{IALC}, \text{ALCP1} + \text{ALCP2}) \]

Equation 28

This equation holds on the condition that both species interfere only with each other through the production of the same alcohol.

Figure 10 shows the resulting simulation program with MALC identical for both species and the proper data. In the main program IY1 and IY2 are both set to 0.45 units, so that the growth in the mixture is simulated. The two monocultures are simulated in reruns.

FINITIM is set at 150 hours, but the two lines

\[
\begin{align*}
\text{FINISH ALC} &= \text{LALC} \\
\text{LALC} &= 0.99 \times \text{MALC}
\end{align*}
\]

are inserted to avoid unnecessary 'number-grinding', when the alcohol concentration is close to its maximum. This condition FINISH indicates that the simulation is terminated as soon as the alcohol concentration reaches 99 percent of its maximum value.

The relative growth rates and the alcohol production factors are chosen so that the results of the two experimental monocultures are matched as well as possible. A comparison of the mixtures (Figure 8) shows that the actual growth of *Schizosaccharomyces* is slightly more than the simulated growth.

Barring statistical insignificance, we must conclude that both species do not interfere with each other's growth through the production of alcohol only, as assumed in the model. It may be that *Schizosaccharomyces* produces some other waste product that is harmful for the other or that *Saccharomyces* produces a waste product that stimulates the other. These possibilities cannot be distinguished from each other without additional information. And as long as this is not available it is a futile exercise to simulate such suppositions.

Exercise 31
a. Try to reason whether a similar effect could result from the supposition that the reduction functions for the species would not be given by
FUNCTION RED1T = (0.,0.), (1.,1.)
FUNCTION RED2T = (0.,0.), (1.,1.)

but by, for instance:

FUNCTION RED1T = (0.,0.), (0.5,0.75), (1.,1.)  (Saccharomyces)
FUNCTION RED2T = (0.,0.), (0.5,0.25), (1.,1.)  (Schizosaccharomyces)

If this is too difficult, you may find the answer by simulation.

b. Try to reason why we should not proceed this way, and what way of tackling the problem would be more appropriate. Also reconsider Section 1.2 in this respect.

TITLE Mixed culture of yeast
INITIAL
INCON  IY1 =0.45, IY2 =0.45, IALC =0.0
PARAMETER RGR1 =0.21, RGR2 =0.06
PARAMETER MALC =1.5 , ALPF1=0.12, ALPF2=0.26
FUNCTION RED1T = (0.0,0.0), (1.0,1.0)
FUNCTION RED2T = (0.0,0.0), (1.0,1.0)
TIMER  FINTIM=150., DELT=0.5, OUTDEL=2.0
OUTPUT  Y1,Y2,ALC
PAGE  GROUP =2
METHOD  RECT

LALC = 0.99*MALC

DYNAMIC
Y1  =INTGRL(IY1 , RY1 )
Y2  =INTGRL(IY2 , RY2 )
ALC  =INTGRL(IALC, ALCP1 + ALCP2)
RY1  =RGR1*Y1*(1.0-RED1)
RY2  =RGR2*Y2*(1.0-RED2)
ALCP1  =ALPF1*RY1
ALCP2  =ALPF2*RY2
RED1  =AFGEN(RED1T,ALC/MALC)
RED2  =AFGEN(RED2T,ALC/MALC)

FINISH  ALC=LALC

END
STOP
END
JOB

Figure 10. A simulation program for the growth of two yeast species that interfere through the production of the same waste product (alcohol).

These simulation programs are conveniently amended. For instance, the yeast cultures may be washed continuously with water that contains sufficient sugar.
Exercise 32

How would you reformulate the rate of change in the integral of the alcohol concentration, Equation 28, if the yeast cultures would be washed continuously with water that contains sufficient sugar?

---

Exercise 33

a. Which type of system is represented by the mixed culture of the yeasts *Saccharomyces* and *Schizosaccharomyces*?

b. Which type of model is represented by the FUNCTION RED1T = (0., 0.), (1., 1.)?

c. Can the yeast growth model 'Mixed culture of yeast' be called an 'explanatory dynamic model'? Explain your answer.

---

### 4.4 Logistic growth

The form of the differential equation for the present problem will now be derived from the structural equations of the simulation program, but only for situations where the reduction factor is proportional to the alcohol concentration so that (1. - RED) may be replaced by (1. - ALC/MALC). Since the alcohol concentration is equal to the integral of the rate of change of yeast times the alcohol production factor, according to the Equations 26 and 27, it is then possible to rewrite Equation 25 in differential equation form as

\[
\frac{dY}{dt} = RGR \cdot Y \cdot (1 - \frac{Y}{Y_m}) \tag{Equation 29}
\]

in which *Y* is the amount of yeast, *t* is the time and *Y_m* stands for the maximum amount of yeast. This equation may be integrated and then becomes

\[
Y = \frac{Y_m}{1 + K \cdot e^{-RGR \cdot t}} \tag{Equation 30}
\]

---

Exercise 34

a. Express *Y_m* in MALC and ALPF.

b. What are the values of *Y_m* for both species of yeast?

c. Show by differentiation that Equation 30 is an integrated form of Equation 29.

d. Express the initial amount of yeast in the constant *K* and *Y_m* of Equation 30.

e. Calculate the time course of the growth of *Saccharomyces* and compare the result with the simulated course.

f. Why does the differential equation only hold for situations where the initial amount of yeast is very small, whereas the simulation program is generally valid?

---

The growth curve that is described by the differential equation and also presented by the simulated growth curves for the monoculture of yeast in Figure 8 is called the logistic growth curve. This S-shaped curve is symmetrical, but this symmetry hinges on the assumption of proportionality between the reduction factor of growth and the amount of growth that has been made. Especially Lotka (1925) and Volterra (1931) generalized the logistic differential equation for interfering species with the following set of differential equations:
\[
\begin{align*}
\frac{dY_1}{dt} &= R_1 \cdot Y_1 \cdot (1 - A_1 \cdot Y_1 - B_1 \cdot Y_2) \\
\frac{dY_2}{dt} &= R_2 \cdot Y_2 \cdot (1 - A_2 \cdot Y_1 - B_2 \cdot Y_2)
\end{align*}
\]

Equations 31

In general this set of differential equations cannot be integrated into analytical expressions for \(Y_1\) and \(Y_2\) as functions of time and therefore it is wiser to leave such simplifying approaches alone and to formulate the problem directly in terms of a simulation model to study the dynamic behaviour.

**Exercise 35**

a. Show to what extent the simulation model for mixed growth of yeast is covered by this set of differential equations.

b. Express the constants \(R_1, R_2, A_1, A_2, B_1, B_2\) in the constants RGR1, RGR2, ALPF1, ALPF2 and MALC.

c. Which constants of the differential equations are the same?

d. Do they remain the same in situations where a species produces a waste product which is more harmful for the other species than it is for itself?

The equilibrium situation, however, i.e. the situation where the rates \(dY/dt\) in Equation 31 are zero, can be calculated similarly to the case of the continuous yeast culture fed by a sugar solution (Section 2.6). Figure 11 shows

![Figure 11](image)

Figure 11. Competition between two species according to the equations of Lotka-Volterra (Equation 31). Full lines distinguish the areas of positive and negative growth rates. In the horizontally hatched area the growth rate of species \(Y_1\) is positive, in the vertically hatched area the growth rate of species \(Y_2\) is positive. Both growth rates equal zero where the lines cross. There, equilibrium exists.

The equilibrium lines with \(Y_1\) and \(Y_2\) along the axes. Left of the lines the reduction factors, e.g. \((1 - A_1 \cdot Y_1 - B_1 \cdot Y_2)\), are positive. One can easily investigate whether an equilibrium is stable or unstable. Stability is defined as follows: when a system in equilibrium is disturbed and there is a reaction of that system that is directed towards the equilibrium value, the system is stable. If, however, the reaction of the system is directed away from the equilibrium after the disturbance, the system is unstable. For a thorough treatment of stability, the
reader is referred to May (1973) and Edelstein-Keshet (1988).

Exercise 36
a. Will there exist a stable or an unstable equilibrium according to Figure 11?
b. Make a new figure where the equilibrium is opposite to the one in a).
c. What is the ecological basis of these differences?

4.5 Summary and steps in model development

Experimental results from the literature (Gause, 1934) were analysed in terms of the qualitative (relational diagrams) and quantitative (differential equations, etc., and model building) methods introduced in Chapters 1 to 3.

It was illustrated that sometimes a rate is (stochiometrically) related to another rate. However, it should be born in mind that mutual dependance of rates should not occur in models: it would point to a violation of the hypothesis in systems analysis that the state of every system at every moment can be quantitatively characterized and that changes in a system can be described by means of mathematical equations.

The example clearly illustrates how models can and should be developed ideally, namely by separate calibration of model parameters on experimental results (here: the data of the monocultures of the species *Saccharomyces cerevisiae* and *Schizosaccharomyces 'Kephir' *), then introduction of the hypothesis (here: that the growth of each species is affected by the formation of the same waste product), and finally that results of the model for mixed growth is validated on fully independent experimental results of the mixed culture. This procedure is summarized in Figure 12.

Figure 12. Model development for individual species, calibration of model parameters, coupling of the separate models through the hypothesis that is to be tested, and validation of the resulting mixed species model with fully independent experimental data for the mixed culture of the yeast species *Saccharomyces cerevisiae* and *Schizosaccharomyces 'Kephir'*. 

42
Finally, from the differential equations of the example of 'mixed culture of yeast' the logistic growth equation was derived, that in its turn was generalized for interfering species according to Lotka and Volterra.

Three main phases in the development of models can be distinguished: model conceptualisation, programming and evaluation. Each phase may be elaborated in a number of steps.

For the conceptualisation phase, the following steps are noteworthy:
- definition of the problem;
- definition of the purpose or objectives of the study;
- definition and/or assessment of the boundaries of the system;
- choice of the level of detail to be considered, or choice of problem complexity (this strongly interacts with the objectives);
- development of qualitative relationships between system elements through e.g. relational diagrams. Here, the choice of the state variables and possible feedback loops become clear;
- development of model equations (differential equations, auxiliary equations, and forcing functions);
- explicit statement of model assumptions that underlie the individual model equations and the model as a whole.

Having arrived at this point one should be able to judge if it is still necessary to develop the full mathematical model: sometimes the conceptual model developed so far is sufficient to satisfy the objectives of the study.

For the model programming phase, the following steps are noteworthy:
- choice of the system of units for the different processes in the model (e.g. mol or g, m or cm, s or hour, etc.);
- grouping of the different processes in submodels (e.g. subroutines, see Section 7.3);
- writing the submodels and the main model;
- assessment of the time coefficients of the model equations;
- assessment of data that are necessary to parameterize the model (the number of parameters is strongly dependent on the level of detail considered and thus also depends on the objectives);
- assessing model integrity: does the model correctly represent the mathematical equations? This can be checked to a large extent by a dimensional analysis and by including material balances (conservation of mass).

In the model evaluation phase, the following steps are noteworthy:
- experimenting with the model: choice of parameter values (literature, new experiments), calibration, sensitivity analysis both with respect to model structure (model reaction on different model equations that could describe the same process) and model parameters (model reaction on changes in parameter inputs within the range of their uncertainty), judgement of model output, validation with independent experimental data on the level of the system as an entity;
- assessment of the model assumptions (these are interconnected with the assumptions made in the derivation of the individual equations);
- drawing conclusions from model behaviour with respect to the real system;
- documentation of the model, both with respect to technical aspects (list of abbreviations of symbols, correspondence of computer mnemonics with mathematical equations, description of the different routines) and to scientific aspects. The latter usually is not the problem, but the first is hardly ever done and thus needs attention;
- sometimes: simplification of the model, based on the increased physiological, chemical, physical and mathematical insight.

Though such enumerations can never be exhaustive, it reflects what we feel as the major steps in model develop-
ment and it provides some insight in the skills a modeller should develop. Among other things it is clear that the modeller should endeavor a sound interaction between theoretical and experimental work.
5 Additional exercises I

P.A. Leffelaar

5.1 Introduction

In the following additional exercises the subject matter is sequenced according to its treatment in Chapters 1 to 4. They enable the student to evaluate his or her skills once again. Though it might seem quite unbelievable that it would be possible to tackle a great number of ecological problems by the means presented so far, this is the truth. The reason is that often the difficulty in analysing and synthesizing ecological systems does not concern the modelling techniques, but rather the knowledge on how to characterize the system under study, i.e. what important relationships do exist in the system, and how can they be described using differential equations. It is common experience of teachers that this aspect can hardly be taught: one has to practise to master it.

5.2 The exercises

Exercise 37

The formation of material $B$ from material $A$ proceeds according to:

$$\frac{dB}{dt} = c \cdot (B_m - B)$$

where $B_m$ is the maximum amount of material $B$ that can be formed. $B$ and $B_m$ are expressed in gram.

a. What dimension has the constant $c$?

b. Use the rectangular integration method to calculate the numerical value of $B$ after 2 time units, if $c = 0.1$, $B_m = 10$ and $B = 0$ at time zero. Use the rule of thumb to determine the time increment of integration.

c. Derive the analytical solution for $B$ as a function of time, and then reason why the time increment used might be considered as correct.

Exercise 38

Three reservoirs are connected to one another as shown in the figure below. All outflows are proportional to the amount of material that is present in the respective reservoirs. A constant flow is added to reservoir 1.
Exercise 39

The beet cyst nematode only reproduces during the cultivation of sugar beets. The length of the growing season for beets is 6 months. The relative rate of increase of the beet cyst nematode is $1/6$ month$^{-1}$. If no beets are grown, the relative rate of decrease of the population is $0.25$ year$^{-1}$.

a. What is the doubling time of the beet cyst nematode population in the presence of beets and what is the half-live time of the population when beets are absent?
b. Draw the relational diagram.

c. Write a CSMP simulation program that calculates the time course of the beet cyst nematode over a period of ten years, and estimate the size of the time step for numerical integration.
d. Calculate analytically the time course of the number of nematodes over the same period. For simplicity's sake, show only the maximum and minimum values of the population as a function of time.

A farmer applies a crop rotation of beets once every 5 years. The rotation starts with beets and 1000 nematodes are present.

Exercise 40

A portion of a simulation model and four relational diagrams, one of which represents this model, are given:

$$W = \text{INTGRL}(IW, W*C)$$
$$Y = \text{INTGRL}(IY, A*D*W - U*Y)$$
$$C = P*N / (EPS+N) - R*D$$
$$N = B - A*W - Y$$

PARAM $D = 0.02, R = 0.010, P = 0.060, EPS = 20.$
PARAM $B = 100., A = 0.025, U = 0.005$
TIMER $FINTIM = 500., PRDEL = 10., DELT = ...$
PRINT $W, C, Y, N$
a. W is the amount of dry matter in kg ha⁻¹, Y is the amount of nitrogen in kg ha⁻¹ and time is expressed in day. What are the units of the remaining variables?
b. Which state variables are distinguished?
c. Which relational diagram of the above figure represents this system?
d. For which value of N the net rate of change of W will be zero?
e. What is then the equilibrium value for Y?

**Exercise 41**

Water from a river flows into a lake at a rate equal to that at which it leaves the lake. The river is polluted by sewage from a town upstream. Assuming complete mixing of the pollutant with the water, the concentration of the pollutant in the lake can be described by:

\[
\frac{dc}{dt} = W \cdot V \cdot t - k \cdot c
\]

where,
- \(c\) concentration of pollutant in the water of the lake (kg m⁻³)
- \(t\) = \(V/Q\)
- \(V\) volume of water in the lake (m³)
- \(Q\) rate of water supply (m³ d⁻¹)
- \(W\) rate of discharge of the pollutant (kg d⁻¹)
- \(k\) decomposition constant (d⁻¹) of the pollutant.

a. Which processes are represented by the three terms in the differential equation? Specify the units.
b. Draw a relational diagram for the pollutant in the lake. Specify each flow separately.

c. What is the average residence time of a water molecule in the lake?
d. Will the average residence time of the pollutant in the lake be longer or shorter than that of a water molecule? Explain the answer.
e. What is the average residence time of the pollutant in the lake?
f. What is the equilibrium level of \(c\) eventually?

g. Use the following parameters in answering the following questions: initial \(c=0.0\) kg m⁻³, \(V=1.25 \times 10^9\) m³; \(Q=1.0 \times 10^7\) m³ d⁻¹; \(W=60000\) kg d⁻¹; \(k=0.05\) d⁻¹.

h. Calculate the numerical value of the time constant of the differential equation.

- write a CSMP simulation model for this system, and note the following:
  - calculate the time step of integration and round it to a practical value;
  - stop the program when 98% of the equilibrium concentration is reached, using the FINISH label (see answer Exercise 22 for an explanation of the FINISH label);
Exercise 42

A rust epidemic in wheat grows logistically. The maximum infection is reached when 80% of all leaf area is covered with rust lesions. The Leaf Area Index, LAI, which represents the ratio of leaf area to soil area, is 4. One rust lesion occupies 0.5 mm².

a. Calculate the number of days until 70% of the maximum infection is reached when the relative growth rate \( r = 0.4 \text{ d}^{-1} \) and the initial number of lesions equals 10000 ha⁻¹.

b. Write a simulation program, assuming that the LAI is constant.

c. Change the simulation program to account for a reduction of leaf area due to leaf fall at a relative rate of 0.012 d⁻¹.

Exercise 43

Due to a leak in a gas pipe, the activity of bacteria converting methane increases, but simultaneously their activity is more and more inhibited by a concomitant ethylene production.

a. Give the rate equation describing the time course of the bacteria.

The maximum population size of the bacteria, \( Y_{\text{max}} \), is proportional to the diameter of the leak expressed in metre (\( Y_{\text{max}} = 500 \cdot \text{diameter} \)).

b. Calculate the time in which 2/3 of the maximum population size is attained if the diameter of the leak amounts to 50 cm, the relative growth rate of the bacteria is 0.6 d⁻¹, and the initial number of bacteria is 10.

c. What is the numerical value of the time coefficient of the system in the beginning and how does it change in the course of time?

Exercise 44

The following differential equations describe growth and interaction in a prey-predator system:

prey: \[
\frac{dH}{dt} = (a - b \cdot H) \cdot H - c \cdot H \cdot P
\]

predator: \[
\frac{dP}{dt} = -e \cdot P + d \cdot H \cdot P
\]

\( H \) and \( P \) are the densities of prey (Host) and predator, respectively, in (number \( \text{m}^{-2} \)). All coefficients are positive and time is expressed in days. When the predator is absent, the prey increases according to the figure below.
Questions a through d assume the predator to be absent.

a. Which mathematical equation describes the curve?

b. Express the constants in the differential equation of the prey in those of the equation referred to in question a.

c. Use the figure to calculate the values of the constants in the differential equation of the prey. To this purpose, rewrite the equation referred to in question a) so that a straight line of \( y \) versus time (x-axis) results on semi-logarithmic paper.

d. Calculate \( H_0 \) from the graph of c) and compare the result to the value that can be read directly from the above curve.

Now assume that only the predator is present.

e. What is the time course of the number of predators? Calculate coefficient \( e \) when it is known that the predator population declines to 25% of its initial number within 6 days.

Now consider the whole system, i.e. both prey and predator are present.

f. Draw the relational diagram for the set of differential equations.

g. What are the units of the coefficients \( a, b, c, d, \) and \( e \)?

h. How would you calculate the time coefficient?

i. Write a simulation program. Run the program with the following parameter values: \( H_0 = 0.5, P_0 = 0.5, a = \) answer to question c, \( b = 0.0577, c = 0.0867, d = 0.1540, e = \) answer to question e, TIMER FINTIM = 100., OUTDEL = 1., DELT= results from trial-and-error.

j. Calculate the equilibrium lines and make a graph with the prey on the x-axis and the predator on the y-axis.

Which densities of prey and predator do you expect after a long period of time assuming the system to result in a stable equilibrium?

Indicate in the graph where the rates are positive or negative.

Plot the time series of \((H, P)\), as calculated by the simulation program, in the graph.

k. Does the time course of the curve agree with the resultant of the rates \(dH/dt\) and \(dP/dt\) in the various parts of the graph of question j?

Exercise 45

Lotka and Volterra gave the following equations for a prey-predator system. \( H \) is the density of the prey, \( P \) the density of the predator. All coefficients are positive. Time is expressed in days and distance in metres.

\[
\begin{align*}
\frac{dH}{dt} &= a_1 \cdot H - a_2 \cdot H \cdot P \\
\frac{dP}{dt} &= -b_1 \cdot P + b_2 \cdot H \cdot P
\end{align*}
\]

a. Draw the relational diagram representing these equations and write the corresponding CSMP program. Assume the following values for the parameters: \( a_1 = 1.0, a_2 = 0.5, b_1 = 1.0, b_2 = 0.5, H_0 = 3.0, P_0 = 3.0, FINTIM = 6.5, DELT = 0.05, OUTDEL = 0.25.\)

b. Give the dimensions of coefficients \( a_1 \) and \( b_1 \). What time course is followed by the prey population density when predators are absent and vice versa? Give the numerical value of the half-live time in each case.
c. The density of animals is expressed per unit area. What dimensions have \( a_2, b_2 \) and \( a_2/b_2 \)? What does the ratio \( a_2/b_2 \) signify? Do you consider the numerical value of the ratio \( a_2/b_2 \) reasonable?

d. Draw a \( P \)- and a \( H \)-axis (take the \( H \)-axis horizontal) and calculate the equilibrium lines. In which direction will the population develop in every quadrant?

e. Now run the CSMP program from a) and plot the time series of \((H, P)\) in the figure under d. How does the system behave?

Exercise 46

A continuous culture of yeast is started in a vessel in order to produce alcohol from sugar. A 10\% (by weight) sugar solution is introduced into the vessel, that has a constant volume of 120 litres. The rate of addition of the solution is 10 l h\(^{-1}\). The contents of the vessel are well mixed and this mixture of water, sugar, alcohol and yeast flows out of the vessel. It is assumed that the growth rate of the yeast is not affected by the alcohol. The relative growth rate of yeast is 0 and 5 d\(^{-1}\) at 0 and 10\% of sugar, respectively, and it is proportional to the sugar concentration. Furthermore, 1 g of sugar yields 0.5 g of alcohol upon conversion by the yeast, and the rate of sugar conversion is 10 g sugar per gram of yeast per day. This rate is independent of the growth rate of the yeast and of the sugar concentration. When the vessel contains the equilibrium concentration of sugar, the vessel is inoculated with 1 mg of yeast.

a. Draw the relational diagram.

b. Develop a simulation model.

c. What is the alcohol concentration of the outflowing solution at equilibrium?

d. Calculate the value of \( \Delta t \) for the rectangular integration method.

e. Calculate the approximate length of the period (FINTIM) to reach this equilibrium.
Part B  Advanced themes in dynamic simulation
6 Numerical integration and error analysis

P.A. Leffelaar

6.1 Introduction

Integration of rate equations is crucial in dynamic simulation studies. Therefore, a large number of numerical methods has been developed. In general, however, just two methods are sufficient to solve most problems. These are the rectangular integration method of Euler and the method of Runge-Kutta. The former has been explained in Chapter 2, whereas the latter will be introduced in Section 6.2. Integration methods are developed to integrate continuous equations. However, discontinuities may occur, for instance a harvest of biomass. The actions that are to be taken at such events are treated in Section 6.3. That section also presents a table that helps in selecting an appropriate integration method for a problem. Since in principle in numerical integration it is assumed that rates have a polynomial time trend during the time interval of integration, all numerical integration methods will introduce errors. The severity of these errors and their relation to the ratio of Δt/τ will be discussed in Section 6.4.

6.2 Some numerical integration methods

The principle of numerical integration has been illustrated in Chapter 2 using the simplest and most straightforward, so-called rectangular integration method of Euler. In this integration method, the rate at time t is calculated from the state and driving variables at that time and it is assumed to be constant during a pre-fixed time interval Δt. Euler's method will give satisfactory results when the ratio of the time interval and the smallest time coefficient (τ) is well chosen, say 1:10 or better, and it enables the user to easily check the calculations. Other methods, e.g. the trapezoidal and the Runge-Kutta methods, are generally more accurate at a similar Δt/τ ratio, because they partially take into account the dynamics of the rates of change. However, they are also more laborious, as can be seen from their computation schemes in Table 3. To illustrate the trapezoidal integration method, the example of the water tank

Table 3. Computation schemes of the rectangular, trapezoidal and Runge-Kutta integration methods; t stands for time, R for rate, A for state, f for 'function of' and the equals sign should be read as an assignment.

<table>
<thead>
<tr>
<th>Method</th>
<th>Computation Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler's rectangular method</td>
<td>[ R = f(A,t) ] [ A_{t+\Delta t} = A_t + \Delta t \cdot R ] [ t = t + \Delta t ]</td>
</tr>
<tr>
<td>Trapezoidal method</td>
<td>[ R_1 = f(A_t,t) ] [ A_1 = A_t + \Delta t \cdot R_1 ] [ R_2 = f(A_1,t + \Delta t) ] [ A_{t+\Delta t} = A_t + \Delta t \cdot (R_1 + R_2)/2 ] [ t = t + \Delta t ]</td>
</tr>
</tbody>
</table>
Runge-Kutta method

\[ R_1 = f(A_1, t) \]
\[ A_1 = A_1 + \Delta t \cdot R_1 \cdot 0.5 \]
\[ R_2 = f(A_1, t + 0.5 \cdot \Delta t) \]
\[ A_2 = A_1 + \Delta t \cdot R_2 \cdot 0.5 \]
\[ R_3 = f(A_2, t + 0.5 \cdot \Delta t) \]
\[ A_3 = A_1 + \Delta t \cdot R_3 \]
\[ R_4 = f(A_3, t + \Delta t) \]
\[ A_{t+\Delta t} = A_t + \Delta t \cdot (R_1 + 2 \cdot R_2 + 2 \cdot R_3 + R_4)/6 \]
\[ t = t + \Delta t \]

from Chapter 2 is used. The rate equation for this problem was Equation 8: \( \frac{dW}{dt}t = c \cdot (W_m - W_t) \), and the following parameter values are used: \( W_0 = 0 \) l, \( W_m = 161 \) l, \( c = 1/4 \) s\(^{-1} \) and \( \Delta t = 2 \) s. A first estimate of the rate variable, \( R_1 \), and the intermediate state variable, \( A_1 \), is obtained using the rectangular integration method, yielding: \( R_1 = (\frac{dW}{dt})_{t=0} = 41 \) s\(^{-1} \) and \( A_1 = W_{t=2} = 81 \). The estimated value of the state \( A_1 \) at the end of the time interval is subsequently used to calculate a second rate \( R_2 \): \( R_2 = (\frac{dW}{dt})_{t=2} = 1/4 \cdot (16 - 8) = 21 \) s\(^{-1} \). The final rate that is realized over the time interval is the arithmetic average of \( R_1 \) and \( R_2 \). Thus, the final amount of water after 2 seconds is: \( 0 + 2 \cdot ((4+2)/2) = 61 \). The following diagram clarifies these calculations:

In this diagram, the sequence a to e indicates the computational sequence. At a and c rate Equation 8 is used; at d the arithmetic average is calculated, and at b and e integration takes place (Equation 9 from Section 2.5); est. stands for estimation.
Exercise 47
Complete the calculation above and plot the calculated amount of water in the tank against time. Calculate the
amount of water in the tank using Equation 6, and plot the results in the same graph. (You can also use your graph
from Exercise 4.)
a. What do you notice when comparing the numerical and analytical solutions?
b. Show graphically that the trapezoidal integration method underestimates the analytical solution and explain why;
make use of the graph representing Equation 3a, and the appropriate numerical values.

In using the trapezoidal integration method the differential equation is evaluated twice to obtain the state of the
model after one time interval. The larger computation effort is more than compensated by the larger ratio of \(\Delta t/\tau\)
that is permitted to reach the same accuracy as in the rectangular method (Section 6.4). This is even stronger for the
Runge-Kutta integration method. This method is not explained in detail, but its calculation scheme given in Table 3
shows that four rate estimates are necessary to compute the final rate.

Exercise 48
Calculate the amount of water in the tank after a time interval of 2 s using the numerical integration scheme of
Runge-Kutta. Use the above numerical parameter values, i.e. \(W_0 = 0.1\), \(W_m = 16.1\), \(c = 1/4\ \text{s}^{-1}\) and \(\Delta t = 2\ \text{s}\).

So far, the integration routines used had a fixed time interval, which was, for the time being, set to one tenth of
the smallest time coefficient of the model. If the time coefficient varies during simulation, and its smallest value is
known, one can fix the time interval at 1/10 of that value. This, however, implies that during periods with large \(\tau\)
values, the accuracy of integration will be greater than during periods with small \(\tau\) values. In modelling, it is prefe-
rible to preset the accuracy of integration of the fastest process and to vary the time interval, so that the error made is
constant. This can be achieved by combining the integration methods of Runge-Kutta and Simpson (the Simpson
method, which is of intermediate accuracy to the trapezoidal and the Runge-Kutta method, is discussed in detail in
IBM, 1975). Two integration routines, Runge-Kutta's and Simpson's are used to integrate the differential equations.
Their results are compared, and if they differ by more than a preset error criterion, the time interval of integration,
\(\Delta t\), is halved. If the deviation is much smaller than the error criterion, \(\Delta t\) is doubled for the next time step. This
procedure resembles the trial and error method to find an appropriate \(\Delta t\) as described in the answer to Exercise 24b,
but besides being automatic, it implies that larger time steps will be used when the time coefficient is large during
the simulation, so that the efficacy of the calculations is much increased. Because of the constancy of the integration
error this method is recommended as a standard method.

Though many other numerical integration routines are available, the methods discussed here are quite sufficient to
tackle the problems encountered in many ecological models.

Before presenting a scheme that helps to select the appropriate integration method, discontinuities in state vari-
bles should be discussed.

6.3 Discontinuities

If discontinuities occur in the content of a state variable, no derivative exists at the breaking point. A typical dis-
continuity in the time course of a growing crop would be its harvest. Then, the content of the state variable in the model must be removed instantaneously. Integration methods cannot be applied with confidence when the integration interval overlaps the discontinuity. Before and after a discontinuity, however, the rate equations may be continuous and integration will not be problematic.

There are two ways of changing the contents of state variables. The first is based on the principle in systems analysis, that the values of the states can only be changed by integration of rates over time. At the discontinuity, then, a special rate equation is applied: the rate should get an infinitely high value during an infinitely short period, so that the area of the rate times its duration equals the change of the amount in the integral. Thus,

$$\Delta A_t = R_t \cdot \Delta t$$

Equation 32

where $\Delta t$ can be taken very small. From Equation 32 the rate can be calculated as

$$R_t = \Delta A_t / \Delta t$$

Equation 33

Numerically, an infinitely short period must be replaced by a time interval $\Delta t$, and rewriting Equation 7 for the moment of harvesting yields:

$$A_{t+\Delta t} = A_t - \Delta t \cdot (\Delta A_t / \Delta t).$$

Equation 34

If $\Delta$ in $\Delta A_t$ indicates a fraction of 1, the full amount present in $A_t$ would be removed.

This method will be referred to as the 'infinite rate method'. Equation 34 implies that the rectangular integration method must be used. This is illustrated by Exercise 49 where it is attempted to nullify a state variable in one time step, using the rectangular and the trapezoidal integration method.

Exercise 49

Let $A_t$ be 100, $\Delta t$ be 2, and define the rate of change $R_t$ as given in Equation 33, where $\Delta A_t$ represents the full amount of $A_t$:

a. Compute the amount $A_{t+\Delta t}$ according to the rectangular and trapezoidal integration routines.

b. What do you conclude about the method of integration to be applied when division by $\Delta t$ occurs in a rate variable?

The second possibility to change the contents of a state variable, is by simply replacing its value by another one at the moment of the discontinuity, so integration does not take place. Before and after the discontinuity, the rate equations may be integrated with any integration method. This method will be referred to as the 'replace method'. Both time events and state events may occur in practice. In the case of a time event it is a priori known when the event will take place. In case of a state event the value of the state that triggers the event should be compared during the simulation with the value at which the event should occur. At the event itself, either the special rate equation (Equation 33), or the replacement of the integral contents may be executed. Although the 'infinite rate method' is simple to program, in most instances it is less accurate than the 'replace method'. Therefore, the choice between the two methods depends on the problem at hand.

The implementation of the 'infinite rate method' and the 'replace method' in terms of CSMP is explained in Chapter 7.

Table 4 summarizes the line of reasoning to be followed to select the appropriate integration method.
Table 4. Decision table to choose the appropriate integration method.

<table>
<thead>
<tr>
<th>Discontinuities occur</th>
<th>Do you need a highly accurate solution?</th>
<th>No: use the 'infinite rate method',</th>
<th>choose the rectangular method, because a division by $\Delta t$ occurs.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>If the event is triggered by time (time event): synchronize $\Delta t$ with the discontinuity, and choose $\Delta t$ smaller than $1/10$ of the smallest time coefficient in the model.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the event is triggered by the value of a state (state event): find $\Delta t$ by trial and error. If the rate of change of the triggering state variable is large, $\Delta t$ should be chosen smaller than when this rate of change would be small, so that the overshoot beyond the value where the state event should occur is limited.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes: use the 'replace method',</td>
<td>any integration method can be used, because before and after the discontinuity the model is continuous. In the vicinity of the discontinuity the time step should be adapted (See Section 7.2.4). The $\Delta t$ during the continuous portion of the model is based on considerations when no discontinuities occur (see below).</td>
<td></td>
</tr>
</tbody>
</table>

No discontinuities occur.

| Is the smallest time coefficient known and stable? | Yes: | choose the method of Runge-Kutta (fixed time step). Set $\Delta t$ at about $1/2$ of the time coefficient. |
| No: | choose the combined methods of Runge-Kutta and Simpson (variable time step). Determine by trial and error what $\Delta t$ is chosen by the method to meet the error criterion. |

| Is $\Delta t$ stable (in which case the time coefficient will also be stable)? | Yes: | choose the method of Runge-Kutta (fixed time step). Take $\Delta t$ as determined. |
| No: | Continue to use the combined methods of Runge-Kutta and Simpson. |

6.4 Error analysis; a case study of integration without and with feedback

In numerical integration it is assumed that the rate of change of a state variable does not change materially over the integration interval $\Delta t$. However, during an integration interval, rates will usually change, and numerical integration methods thus introduce errors in the solutions of differential equations. The question is if these errors are severe. Error analysis is applied to quantify the integration errors in terms of $\Delta t$ and $\tau$. As many rates in nature are proportional to the amounts present, the error analysis will be demonstrated for the model of exponential growth (Section 2.4: Equations 2, 2a and 5).

In the analysis of error propagation in integration, two situations should be distinguished. In the first situation, the rate as a function of time is known in advance, e.g. it acts as a driving force. Then, results of integration are in-
dependent of the state variable that is changed by integrating the rate, and the same relative error is made each time interval (situation without feedback). In the second situation, the rate depends at each moment on the state of the system. This situation usually occurs in simulation, and the error in the calculations will accumulate: in exponential growth, underestimation of the value of the state will cause underestimation of the rate of change, and hence of the subsequent value of the state (situation with feedback).

Integration of a driving force (no feedback) When a driving force is integrated, its value is known in advance as a function of time, for instance a series of data of rates of change which may be represented by an exponential curve. Figure 13 shows such a curve (solid line), drawn according to Equation 2a, with the initial velocity \( v_0 = A_0 \cdot c \). Integration by the rectangular and trapezoidal methods yields the hatched areas. In this case, the exact error in the result obtained by the rectangular integration method could be derived, but a good approximation is given by the area of the triangles that are included by the trapezoidal integration method.

The relative error in an integration method of order \( n \) is defined as:

\[
E_{rel,n} = \frac{A_{(n\text{th order method})} - A_{((n+1)\text{th order method})}}{A_{(n\text{th order method})}}
\]

where \( A \) stands for surface area. Examples of first, second, third and fourth order integration methods are the rectangular, trapezoidal, Simpson and Runge-Kutta methods, respectively.

![Figure 13. Graphical representation of the surface areas that are calculated by the rectangular integration method (\( \square \)) and by the trapezoidal integration method (\( \square + \square \)), when the exponential rate curve is known as a function of time (solid curved line).](image)

Applying Equation 35 to calculate the relative error in the first order rectangular integration method gives:

\[
E_{rel,1} = 1 - \frac{(v_0 + v_0 \cdot e^{c \cdot \Delta t}) \cdot \Delta t / 2}{v_0 \cdot \Delta t}
\]

Equation 36

From the numerator and denominator \( v_0 \cdot \Delta t \) cancels, and \( e^{c \cdot \Delta t} \) can be written according to a Taylor expansion (see Appendix 1): \( 1 + c \cdot \Delta t + 1/2 \cdot (c \cdot \Delta t)^2 + ... \). Since \( c \cdot \Delta t \) is much smaller that 1, higher order terms can be neglected and after some algebra, one obtains:

\[
E_{rel,1} \equiv - \frac{1}{2} \cdot c \cdot \Delta t = - \frac{1}{2} \cdot \frac{\Delta t}{\tau}
\]

Equation 37
The derivation of this result is given in detail in Appendix 1. Equation 37 shows that the relative error is proportional to the ratio of the time interval of integration and the time coefficient of the model. The minus sign indicates the underestimation by numerical integration of the surface area under the curved solid line in Figure 13. As the rate of change increases exponentially, the same relative error is made each time interval. Thus, the final absolute error is $E_{abs,1} = -1/2 \cdot (\Delta t/t) \cdot A$. With the trapezoidal integration method the triangles are taken into account, so the error is much smaller. The remaining error is estimated by the area between the straight line formed by connecting the corners of the vertical bars and the parabola constructed through the values of the exponential rate function at times $t, t+1/2 \cdot \Delta t$, and $t+\Delta t$. The relative errors for the trapezoidal and Runge-Kutta integration methods have been calculated by Goudriaan (1982) and are given in Table 5. Note that for integration without feedback, the relative error is independent of the simulation time.

Table 5. Estimates of the relative errors of three integration methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Without feedback</th>
<th>With feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangular</td>
<td>$-(\Delta t/\tau)/2$</td>
<td>$-(t \cdot \Delta t/\tau^2)/2$</td>
</tr>
<tr>
<td>trapezoidal</td>
<td>$(\Delta t/\tau^2)/12$</td>
<td>$-(t \cdot \Delta t^2/\tau^3)/6$</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>$(\Delta t/\tau)^3/2880$</td>
<td>$-(t \cdot \Delta t^4/\tau^5)/120$</td>
</tr>
</tbody>
</table>

**Exercise 50**

a. Use the estimates of the relative errors in the situation without feedback from Table 5 to calculate which value of $\Delta t$ (expressed as a fraction of $\tau$) must be chosen to yield a relative error of 1% in the integration of an exponential curve, for the three methods.

b. Calculate analytically the area under an exponential curve between $t = 0$ and $t = 1$, with $v_0 = 1$ and $c = 1$.

Also, calculate this area using the three numerical methods with $\Delta t = 1$. Since in integration without feedback all rates are known in advance, neither the initial condition nor the intermediate areas need to be known.

c. From the answer to b, calculate the exact absolute and relative errors in the results of integration by the three numerical methods, with respect to the analytical solution. Compare these exact relative errors with the estimates in Table 5.

**Integration of a differential equation with feedback** In dynamic simulation the rate variable is usually not known as a function of time; instead, new values are calculated from the current state, as for example in Equations 2 and 3. In a model of exponential growth of a population of animals, an underestimate of the growth rate at time $t$ will result in an underestimate of the population at time $t + \Delta t$, and thus also in the growth rate at that moment. This phenomenon occurs in numerical integration of differential equations with feedback, and an additional error will be added each time interval. Thus, in contrast to numerical integration of a driving force, relative errors increase in the course of the simulation. The error analysis is slightly different from that for a driving force, because the relative error will refer to the integral value including the initial value.

For the rectangular method this implies the following. The rate at time $t$ equals (Equation 2): $(dA/dt)_t = c \cdot A_t$. So the value of the integral after one time interval is: $A_{t+\Delta t} = A_t + \Delta t \cdot c \cdot A_t$. To calculate the value of the integral according to the trapezoidal method, the rate at $t + \Delta t$ is calculated:

$$(dA/dt)_{t+\Delta t} = c \cdot A_t \cdot (1 + \Delta t \cdot c).$$
It then follows that

\[ A_{t+\Delta t} = A_t + \Delta t \cdot (c \cdot A_t + c \cdot A_t \cdot (1 + \Delta t \cdot c)) / 2. \]

All the terms required to calculate \( E_{rel,1} \) according to Equation 35 are now available. After some algebra, and neglecting higher order terms, the relative error in the rectangular integration method is derived as:

\[ E_{rel,1} = - \frac{1}{2} \cdot (\Delta t \cdot c)^2 = - \frac{1}{2} \cdot (\Delta t)^2 \]

Equation 38

The derivation of this result is given in detail in Appendix 2. This relative error occurs in each integration step and, in contrast to the situation without feedback, the errors accumulate. At time \( t \), when \( t/\Delta t \) integration steps have been performed, the relative error is

\[ E_{rel,1} = - (t \cdot \Delta t / \tau^2) / 2 \]

Equation 39

Interestingly, the relative error is proportional to \( \Delta t \), as for integration without feedback, but it is now also linearly dependent on the elapsed time since the start of the simulation. The relative errors for integration with feedback for the trapezoidal and Runge-Kutta integration methods as derived by Goudriaan (1982) are also given in Table 5.

**Exercise 51**

a. Use the estimates of the relative errors in the situation with feedback from Table 5 to calculate which value of \( \Delta t \) (expressed as a fraction of \( \tau \)) must be chosen to yield a relative error of 1\% in the integration of Equation 2, for the three methods, when the simulation time is equal to \( \tau \).

b. For the situation with feedback, the differential equation is \( \frac{dA}{dt} = v = c \cdot A \) (Equation 2). Calculate \( A \) at \( t = 1 \), when \( A_0 = 1 \), \( c = 1 \), and \( \Delta t = 1 \), using the three numerical methods. Since in integration with feedback rates are not known in advance, the initial conditions are needed and the intermediate areas must be calculated.

Also calculate \( A \) at \( t = 1 \) analytically for this situation.

c. From the answer to b, calculate the exact absolute and relative errors in the results of integration by the three numerical methods, with respect to the analytical solution. Compare these exact relative errors with the estimates in Table 5.

**Exercise 52**

a. Derive an equation that gives the value of \( A \) directly after \( n \) time intervals \( (A_n) \) of size \( \Delta t \), when \( \frac{dA}{dt} = c \cdot A \) is numerically integrated.

b. Convert this equation in \( n \) to a function of time.

c. Show that the analytical solution is obtained, i.e. Equation 5, if the time interval \( \Delta t \) approaches zero.

When an integration interval includes a discontinuity, the error in any integration method will be large, since no derivative exists there. Error analysis as described above can, however, be applied before and after such discontinuities. The numerical error due to the discontinuity itself is avoided by using either the 'infinite rate method' or the 'replace method' from Section 6.3.
6.5 Concluding remarks

The rule of thumb to choose a time increment of integration in relation to the time coefficient, that was used till Chapter 6, has now been further specified in terms of the error estimates in Table 5. It is seen that this rule would give an error of about 5\% after a simulation time of length \( t \), when the rectangular integration method is used. This is often an acceptable result. Although for models containing many different time coefficients most often the trial and error method to determine a reasonable \( \Delta t \) (as explained in the answer to exercise 24b) will be followed in practice, Section 6.4 helps to understand how the time interval of integration affects the error in the calculations.
7 Aspects of structured programming using CSMP and FORTRAN

P.A. Leffelaar

7.1 Introduction

The term 'structured programming' refers to the legibility and the intelligibility, or the 'style' of a program. A program must be legible: it should be attractive to study a program's text. A program with 'style', illustrates that the designer has been aware of what it is intended to do, and that he or she wishes to convey this knowledge to others. Therefore, such programs will contain fewer errors and will be easier to modify at a later stage. Hence, programming is more than simply learning or getting acquainted with a simulation or programming language: good programs start with a thorough analysis of the problem, since this induces thinking about structure.

It is useful to realize that people generally 'think from coarse to fine'; from integrated to differentiated. Therefore, the start of problem analysis will be at a high level of integration. Subsequently, parts of the problem will be identified, that in themselves will also be amenable to analysis. A program, however, must calculate from fine to coarse, because all the auxiliary calculations must have been carried out before, for example, the rate of change of a state variable can be calculated: only then correct integration is possible. This is why simulation languages are designed to sort the equations: then the programmer is in the best position to think about the problem.

In structured programming we make use of the principle of gradual refinement, i.e. the process of dividing the problem into a number of subproblems. Each subproblem is subsequently worked out in more detail until components have been formulated that reach the level of program commands. In fact, this makes 'thinking from coarse to fine' operational. Working from coarse to fine may advantageously be reflected in simulation models resulting in structured programs having 'style'.

An INITIAL, DYNAMIC and TERMINAL segment were distinguished as the main structure of CSMP models (Chapter 3). Within the INITIAL and DYNAMIC segment, the sorting algorithm was used. This chapter shows how these program segments may be further structured by the label NOSORT and the PROCEDURE-ENDPROCEDURE block (Subsection 7.2.1), and by the FORTRAN branching possibility IF-THEN-ELSE-ENDIF and the array calculations of the DO-loop (Subsection 7.2.2). The special array integral possibility of CSMP together with the new integration methods from Chapter 6, and the integration status variable KEEP, are discussed in Subsection 7.2.3. In Subsection 7.2.4 some of the new features are illustrated by applying them to discontinuities in state variables as discussed in Section 6.3: in fact this is the CSMP implementation of the 'infinite rate method' and the 'replace method'. Till then all possibilities were applied within the CSMP main program, i.e. between TITLE and STOP. However, it is possible to extend CSMP with FORTRAN subroutines and functions, implying that the advantages of both languages can be combined. These FORTRAN elements are introduced in Section 7.3. The use of FORTRAN alone as a simulation language will be discussed in Chapter 10. The latter is of importance because FORTRAN is available for most machines, contrary to CSMP.

It is hoped that this chapter will help the programmer to develop an awareness of the stylistic features presented so that new programs will be easier to read.
7.2  The structure of CSMP program segments

7.2.1 NOSORT and PROCEDURE's

Until now the sorting algorithm of CSMP was used to free the user from the task of correctly sequencing the statements within the INITIAL and DYNAMIC segment. Sometimes statements should not be sorted by the compiler, however, for instance to use FORTRAN branching conditions, or to make array calculations (see Subsection 7.2.2). In that case the statements have to stay in a fixed order. Figure 14 shows the two ways to avoid sorting the statements.

```plaintext
TITLE The use of NOSORT in CSMP
INITIAL
  NOSORT
  ...
  ...
DYNAMIC
  NOSORT
  ...
  ...
PROCEDURE
  ...
  ...
PROCEDURE OUT1,OUT2 = PRONAM(IN1,IN2)
  ...
  ...
TERMINAL
  ...
END
STOP
ENDJOB
```

Figure 14. Two possibilities to avoid sorting of statements in a CSMP program.

In CSMP. The simplest method is to declare the INITIAL and DYNAMIC segments NOSORT. The NOSORT labels could have been placed somewhere in the INITIAL or DYNAMIC segment and be closed by the label SORT, but this would split up the program into sections. The program blocks in between the NOSORT-SORT sections would then be sorted individually, but during the translation phase of the program, statements from one SORT section can not be moved to another SORT section. Since this usually gives computational problems, it is recommended to either write full segments in NOSORT or to use PROCEDURE's. FORTRAN-statements can be defined only in non-sortable sections, and always have to be sequenced in computational order. When full program segments are declared NOSORT, there is no need to close the segment by the label SORT.

PROCEDURE's provide a more elegant means to define program sections in which non-sortable statements may be used. The PROCEDURE is treated as an entity. It is sorted as a functional block on the basis of input (IN1,IN2) and output (OUT1,OUT2) variables given by the user in the definition of the PROCEDURE. These names correspond with those of the variables in the functional block. PROCEDURE's may be ended by the word
ENDPROCEDURE, but ENDPRO suffices. The statements within the PROCEDURE are not sorted. Variables defined within a PROCEDURE block and not appearing in the definition, e.g. because they are not needed to sort the block, are not available for data output by means of PRINT or OUTPUT. If these variables are needed for output they should be included as output names in the PROCEDURE definition. The use of PROCEDURE's allows the programmer to combine the sorting algorithm of CSMP with the use of powerful (non-sortable) FORTRAN statements in one program. This is a strong argument to use PROCEDURE's instead of the NOSORT option. However, if one uses the NOSORT option, the modeller must take care that the calculation sequence in the program is 'inverted' after that all the equations 'from coarse to fine' are known, so that all the input data are available before any calculation. The result will then often contain fewer programming errors.

7.2.2 Branching and array calculations

Branching Often decisions need to be made in models, e.g. between different rate equations valid under different environmental conditions. The IF-THEN-ELSE-ENDIF structure may then be used to structure the decision process as illustrated in Figure 15.

```
IF (FOOD .GT. FOODMN) THEN
  Food is present and aerobic or anaerobic growth occurs.
  IF (OXYGEN .GT. OXYMN) THEN
    O_2 is present.
    Rate equations for aerobic growth.
  ELSE
    O_2 is absent, i.e. below a certain minimum value.
    Rate equations for anaerobic growth.
  ENDIF
ELSE
  Food is absent, i.e. below a certain minimum value, and death occurs.
  Rate equations for death.
ENDIF
```

Figure 15. An IF-THEN-ELSE-ENDIF structure.

Figure 15 shows that IF-THEN-ELSE-ENDIF structures may be nested. Therefore, complicated decision schemes may be constructed. To keep these structures clear, it is good practice to avoid programming all computer code directly within the structure, but instead to call subprograms where the actual (rate) equations are executed (Section 7.3). An IF-THEN-ELSE-ENDIF structure may also consist of IF-THEN-ENDIF.

In Figure 15, .GT. stands for Greater Than. Other comparators are: .GE. (Greater Equal), .LT. (Less Than), .LE. (Less Equal) and .EQ. (Equal). CSMP can not sort an IF-THEN-ELSE-ENDIF structure. It must therefore be located within a NOSORT section or within a PROCEDURE-ENDPROCedure, and the sequence of equations must thus be determined by the user.

Exercise 53
A car can drive until it runs out of petrol. The function of the car's speed (RDIST, km h$^{-1}$) against the independent variable of time (h) is given in table RDISTB as:

```
FUNCTION RDISTB = 0.0, 80.0, 2.0, 80.0, 3.0, 120.0, 5.0, 120.0, ...
  6.0, 40.0, 10., 40.0
```
The use of petrol per kilometer (PTUSE, l km) is a function of the car's speed (RDIST) and is given in table PTUSE as:

\[
\text{FUNCTION PTUSE} = 20.0, 0.111, 90.0, 0.062, 120.0, 0.082
\]

where PTUSE is an abbreviation for Petrol USE Table from which PTUSE is read: at a speed of 20 km h\(^{-1}\), the car may travel 9 km h\(^{-1}\), at 100 km h\(^{-1}\), 16.1 km h\(^{-1}\) can be travelled, etc.

If the amount of petrol in the tank falls below the threshold level PTLLV (PeTrol Lowest LeVel), the petrol supply effectively stops, and the car's speed is zero. Note that the speed is therefore only given by FUNCTION RDISTB as long as there is petrol (PTRL).

a. Design a relational diagram for this model.
b. Set up the equation for the rate at which the petrol is used: RPTRL. Perform a dimension analysis for this.
c. Design the IF-THEN-ELSE-ENDIF structure to select the car's speed when ample petrol is available and when it has fallen below the threshold value PTLLV. Especially pay attention to the calculational sequence of the equations.
d. Write a simulation program.
   - Set speed to zero when 2 litres of petrol are left in the tank: PTLLV = 2;
   - Let the program stop if speed is zero, by using the FINISH statement.
e. What integration method would exactly integrate the rate equation RDIST? Explain why.

Array calculations: An array is a series of subscripted variables, i.e. RATE(1), RATE(2), etc. The variables may be referred to as RATE(I), where I is an integer counter that indicates the number in the array. Array data need memory space. This is indicated by including the line STORAGE RATE(26), VAR(6) into the program, which means that 26 memory locations are reserved for the variable RATE and 6 locations for VAR.

To assign values to array variables at the beginning of a simulation, the TABLE statement is used:

\[
\text{TABLE VAR(1-6) } = 6 \times 5.
\]

or

\[
\text{TABLE VAR(1-6) } = 1., 2., 4 \times 5.
\]

In the first TABLE-statement the 6 in \(6 \times 5\) indicates the number of memory locations. Therefore, it should be an integer (a number without a decimal point, Section 3.8). Each location is filled with a real number, so \(5\). In the second TABLE-statement the first two memory locations are assigned the values 1. and 2., whereas the remaining 4 locations have got the same value of \(5\).

The array feature permits data stored in a one-dimensional array to be identified in structure statements. If calculations have a similar structure, e.g. the calculations of transport phenomena through a series of reservoirs (see Exercise 55), the DO-loop may be conveniently used:

\[
\text{DO 10 I = 1,26}
\]

\[
\text{DO 20 J = START,END,INCREM}
\]

\[
\text{RATE(I) = \ldots...}
\]

\[
\text{VAR(J) = \ldots...}
\]

The 'names' of these DO-loops are 10 and 20, respectively, and I and J are counters that indicate the memory location. Counter I takes successively the values 1, 2, 3, ... 26, so that all RATE's will be computed. Counter J starts and ends at the value assigned the variables START and END, respectively, and the numerical values of VAR are calculated at the counter values of START, START+INCREM, START+(2*INCREM), START+(3*INCREM), 66
etc. up to END. If the value of INCREM equals 1, it may be omitted as demonstrated in DO-loop 10. The value of START may be higher than END, but then, of course, INCREM should be negative.

Counters should be whole, integer numbers. Therefore, the line FIXED END, I, INCREM, J, START should appear in the simulation program. A DO-loop can not be sorted by CSMP, and should thus appear in a NOSORT section, or in a PROCEDURE-ENDPROC(EDURE) or in a SUBROUTINE.

7.2.3 Array integrals and higher order integration methods

Array integrals The feature of integrator arrays is very useful in the simulation of distributed systems, where time (Chapter 8) or space (Chapter 9) is divided into classes or layers, respectively. The contents of each class or layer is a state variable and is represented by an integrator. Between classes or layers transport of matter takes place. Array integrals with subscripted variables can be conveniently applied for computations that have a similar structure, as was the case with the DO-loop. The statement:

\[
A = \text{INTGRL}( I, A, IA, NRATE, 25 )
\]

specifies an array of 25 integrators in which \( A(I) \), \( IA(I) \) and \( NRATE(I) \) are the output, initial conditions and input variables, respectively. The third argument, 25, is an integer constant. The initial conditions of an integrator array could be supplied by means of a TABLE statement.

For subscripted variables memory space should be reserved (Subsection 7.2.2). However, there is one exception to this rule: for variables that appear in integrator arrays, memory space is automatically reserved by the CSMP system, and a user declaration of STORAGE \( A(25) \), \( IA(25) \) and \( NRATE(25) \) would generate an error message stating 'multiple declaration of name'.

Higher order integration methods The INTGRL statement merely indicates which state variables are present in the model and which rates have to be integrated. The numerical integration method has still to be specified. To this purpose the label METHOD is used (Section 3.3). Since Chapter 6, three methods besides the one of Euler are known: two fixed time interval \((\Delta t)\) methods, i.e. the trapezoidal method, TRAPZ in CSMP, and the Runge-Kutta method, RKSFX in CSMP, and one method with a variable time interval which is determined by an error criterion. The latter method is due to Runge-Kutta and Simpson and may be selected in CSMP by METHOD RKS.

The integration methods RECT, TRAPZ and RKSFX are sometimes called first, second and fourth order methods, respectively, according to the number of consultations of the rate equations to obtain a final result (Table 3, Section 6.2). This final result is stored in the state variables that occur in the INTGRL statements, when the integration status variable, called KEEP, equals 1. The value of KEEP equals 0 during intermediate calculations. Thus, in method RECT, KEEP=1 all the time. In method TRAPZ, KEEP will be alternate 0 and 1, and for method RKSFX the sequence 0-0-0-1 will occur each step. In method RKS it is not a priori known how many times the user program will be consulted to reach a final result, because the method determines its time interval depending on the changes in the time coefficients in the model during the simulation to meet the error criterion. The integration status variable KEEP may be used in models containing code that should be executed only once per time step using the final results for that step. Then, the value of KEEP can be tested in a FORTRAN IF statement to determine whether it is time to execute the special procedure. The integer variable KEEP should never be changed by the user, however. Subsection 7.2.4 describes an application of the use of KEEP in CSMP programs.

Exercise 54

The number of times the user program will be consulted to reach a final result, can be checked by the introduction
of some counters in the program. In the INITIAL segment of the program these counters, COUNT1 and COUNT2, are set to zero. The DYNAMIC segment of the model should contain the statements COUNT1=COUNT1+1 and COUNT2=COUNT2+KEEP. It should be obvious that these statements can not be sorted by CSMP, because the same variables occur to the left and to the right of the equal sign. (This clearly demonstrates why an equal sign means 'is to be replaced by' rather than 'is equal to', see Section 3.2.) Each time these statements are passed, COUNT1 is incremented by one, and COUNT2 by the value of KEEP. Thus, both the number of program consultations and the number of final integrations can be kept track of.

a. Rewrite the program of the answer to Exercise 25 in NOSORT and add the statements discussed.

Run the program with the methods RECT with DELT=0.5, TRAPZ with DELT=1.0, and RKSFX and RKS with DELT=2.0.

b. Which combination do you prefer?

The actual value of Δt determined in method RKS, should neither become too large nor too small. The upper limit of Δt is automatically taken equal to the output interval (the smallest value of PRDEL or OUTDEL) that is usually supplied by the user, but if this value would be too large, the upper limit could be set by the TIMER parameter DELMAX. The lower limit is automatically taken as 10⁻⁷ * FINTIM, but also this limit can be set by the parameter DELMIN. If DELT would not be specified, RKS would take 1/16 * (the smallest value of PRDEL or OUTDEL) as a first estimate. The full timer specification for method RKS would be:

\[
\text{TIMER TIME = ...}, \ \text{DELMIN=}..., \ \text{DELT = ...}, \ \text{DELMAX=}..., \ ...
\]
\[
\text{PRDEL=}..., \ \text{OUTDEL=}..., \ \text{FINTIM=}...
\]

Both an absolute (ABSERR) and a relative (RELERR) error criterion is considered in method RKS. In this method an error ratio is computed for each integrator according to:

\[
\text{Error ratio for integrator} = \frac{|\text{output} - (\text{separate estimate of output})|}{\text{ABSERR} + \text{RELERR} \cdot |\text{output}|},
\]

where 'output' and 'separate estimate of output' are calculated by the Runge-Kutta and the Simpson integration method, respectively. At the end of an integration step, the array of error ratios is scanned to see if the error criteria have been met. If any integrator's error ratio exceeds +1, showing that its error estimate (the numerator of the error ratio) exceeds the error allowed by the absolute and relative error criteria, the step is considered unsuccessful and is discarded. It can be seen from the formula that if the integrator value is small, the permissible absolute error mainly determines the error ratio. If the integrator value grows in magnitude, the permissible relative error is the most important. The default value for both ABSERR and RELERR is 10⁻⁴. Although this value is usually sufficient, other values may be specified by the lines:

\[
\text{ABSERR} \ A(1) = 1.E-5
\]
\[
\text{RELERR} \ A(2) = 1.E-6
\]

where A is an integrator output. These values are also applied for the integrators that are unspecified. The error criteria should not be set too stringently: it should be related to the accuracy of the computer.

Sometimes the error criterion is not met by decreasing the time interval in RKS, and the error message 'DEL T is less than DELMIN' is produced by the system. Then it is advisable to look for a programming or conceptual error, rather than to resort directly to decreasing the value of DELMIN.
Exercise 55

Oil is transported along a pipeline from a first reservoir to a second one, and from there to a third reservoir, where it is collected. The reservoirs have the same size. The rate of supply of oil to the first reservoir stagnates: it is zero. The outflow rate of oil from each reservoir (except the last) is proportional to its contents. The system described is often called a second order (two similar elements) exponential (proportional outflow) delay. The mean time that the oil remains in the two delay elements of the system is 8 weeks.

a. Draw the relational diagram for this system. Also include the collection reservoir in which the oil is stored.
b. Formulate the rate and the state equations. Formulate the latter analogously to Equation 7 from Chapter 2.
c. What is the average residence time or time coefficient of each reservoir? What is its dimension?
d. Write a simulation program.

- Use the array INTGRL for the two reservoirs that have an outflow of oil, and a separate INTGRL for the reservoir that collects the oil.
- Use two different DO-loops to calculate the flows: one to calculate the individual flow from each reservoir, and another to calculate the net rate of change in each reservoir.
- Use PROCEDURE-ENDPROC(cedure) blocks to let CSMP sort these DO-loops.
- Which integration method is best suited for this problem?
- What time interval of integration do you choose?

7.2.4 An application: the implementation of discontinuities in state variables in CSMP

A typical example of a model with a code that should be executed only once per time step using the final results for that step, is for the 'replace method' (Section 6.3) used to tackle a discontinuity in the content of a state variable.

Suppose a crop is to be harvested. This harvest may be controlled either by the time (time event) or by a certain value of the state variable (state event). In both cases an algorithm should be developed in which decision variables regulate the execution of the statements. Figure 16 shows the reasoning for obtaining an algorithm for a time event.

Figure 16. Steps to obtain an algorithm to replace the contents of a state variable in a time event.
Before reaching the moment that the time event should occur, the current values of the state variables and the time are stored. This is necessary to handle the situation when time has proceeded too much, i.e. beyond the moment that the time event should occur. In that case the program should return to the situation before that moment and the time interval of integration should be equalled to the difference between that moment and the actual time. Although the time event should occur at a certain moment, it is practically impossible to arrive at the exact point in time, due to round off errors in real variables. Therefore, Figure 16 shows a time period during which the event may occur, namely between the values of (time event) and \((1+\varepsilon)\cdot\text{(time event)}\), where \(\varepsilon\) is the fraction overshoot allowed. In this period, which may be very short, the integral contents are replaced. The algorithm derived from Figure 16 is given in the listing of Figure 17.

**TITLE** Discontinuity: time event using the replace method

**INITIAL**

**STORAGE** TEVENT(3)

**FIXED** IEVENT, KEEP

**TABLE** TEVENT(1-3)=2.02, 3.39, 3.8

**INCON** IBIOM =100.

**PARAM** RGR =0.2, EPS =0.0001

**TIMER** FINITIM =3.8, PRDEL=0.2, DELMAX=0.2

**PRINT** BIOM, RBIOM, HARVST, OLDBIO, OLDTIM, DELMAX, DELT

**METHOD** RKS

IEVENT =1

**DYNAMIC**

**NOSORT**

BIOM =INTGRL(IBIOM, RBIOM)

HARVST =INTGRL(0.0, 0.0)

IF(KEEP .EQ. 1) THEN

IF(TIME .LT. TEVENT(IEVENT)) THEN

OLDBIO =BIOM
OLDTIM =TIME

ELSE

IF(TIME .GT. (1.0+EPS)*TEVENT(IEVENT)) THEN

BIOM =OLDBIO
TIME =OLDTIM

DELMAX =TEVENT(IEVENT)-OLDTIM

ELSE

CALL PRINT
HARVST =HARVST + 0.9 * BIOM
BIOM =BIOM - 0.9 * BIOM
RBIOM =RGR*BIO
DELMAX =PRDEL
IEVENT =IEVENT+1

CALL PRINT

ENDIF

ENDIF

ENDIF

RBIOM =RGR*BIO

END

STOP

ENDJOB

---

Figure 17. Implementation of the replace method in CSMP.

70
Lines 19 and 23 in Figure 17 show that the user may manipulate the system variable TIME. This should only be attempted when final results are updated, i.e. when KEEP=1. Good results were obtained with the integration method of Runge-Kutta and Simpson (RKS) with the error (EPS) set to $10^{-4}$. When the method of Euler (RECT) was used, this error had to be taken larger, i.e. $10^{-2}$, so that the program would not get 'stuck' at the discontinuity. This error is still very small, however.

Note that the PRINT label of CSMP in Figure 17 is invoked by a CALL in a non-sortable section. This feature may be used to print variables at moments other than multiples of PRDEL or OUTDEL. If an event has occurred, the counter IEVENT is increased by 1 in line 31 to set the next TEVENT. To prevent array boundary problems with respect to this variable, the last time event is taken equal to FINTIM.

---

Exercise 56

a. Study the listing in Figure 17 by explaining the IF-THEN-ELSE-ENDIF structure in natural language.

b. Execute the program and study its results.

---

Exercise 57

a. How should Figure 16 be adapted to obtain an algorithm for a state event.

b. Write a CSMP program in which a BIOMass is harvested (HARVST) when BIOM is about equal to the parameter BIOHST (amount of BIOMass where HARVST should take place). The event is thus triggered by the value of a state called BIOM.

c. Study the results of the program.

---

The 'replace method' has been applied successfully in models on simultaneous water and gas transport, where the gas flow due to the build up of different total gas pressures in adjacent soil compartments was assumed to even out instantaneously, implying that for each time step a set of linear equations had to be solved (Leffelaar, 1988). Another application was in models on root growth. Here, during the simulation process new state variables had to be introduced, representing newly explored soil volumes by newly grown plant roots. Since it is not possible to create new integrals in CSMP during a simulation, this was mimicked by defining a larger number of integrals than needed at time zero. The excess number of integrals had an initial content of zero and a zero rate of change. During the simulation process, these integrals could then be initialized with the appropriate amount of substance and form part of the process description (Hoffland et al., 1990).

In Chapter 8, an application of the 'replace method' for a state event is discussed, using the integration method of
Euler where KEEP is always equal to 1 (see Exercise 54).

This section is finished by discussing the implementation of the 'infinite rate method' in CSMP. In the 'infinite rate method' (Section 6.3, Equation 34), the content of an integral is changed by integrating an 'infinitely' high rate over a very short time, i.e. one DELT. Clearly, this rate only applies at the moment of harvesting. This can be achieved by using CSMP's INSW function (Table 1, Chapter 3):

\[
PUSH = \text{INSW} \left( TIME - \text{THRVST} , 0.0 , 1.0 \right) \tag{Equation 40}
\]

where THRVST is the moment of harvesting. The value of PUSH will be 1.0 if TIME equals or exceeds THRVST. Rewriting Equation 34 now yields:

\[
A_{t+\Delta t} = A_t - \Delta t \cdot (\Delta A_t/\Delta t) \cdot \text{PUSH}. \tag{Equation 41}
\]

Exercise 58

a. Draw the relational diagram for the 'infinite rate method' as applied to a time event in which the state variable biomass (BIOM) is completely harvested: the harvest is collected in the integral HARVST. Assume that the biomass grows according to the rate equation for exponential growth (Equation 2).

b. Write the state equation in the form of Equation 7 (Chapter 2), thus including indices for time and \( \Delta t \) for the condition that \( \text{PUSH} \) would be 0.0 and 1.0.

c. Write a CSMP program for this problem, using Equation 40 and the differential part of Equation 41.

7.3 Subroutines and functions in CSMP/FORTRAN

7.3.1 Three forms to call subroutines in CSMP

Contrary to the PROCEDURE, subroutines and functions are completely independent (sub)programs, which may be called from a main program more than once. Figure 18 shows how FORTRAN subroutines can be called in a CSMP program, and where the subroutine should be defined.

Examples I and II in Figure 18 make use of the CSMP sorting facility. In the third example the programmer must correctly sort the equations and CALL's.

In examples II and III the calls to the subroutine are identical to the subroutine definition between the labels STOP and ENDOBJ. The subroutine in example II has been made sortable by placing it in a PROCEDURE-ENDPROCEDURE block: sorting is carried out on the basis of the specified input (C,D) and output (A,B) variables as mentioned in Subsection 7.2.1. The subroutine call in example I is a sortable CSMP call. Here too, sorting is carried out on the basis of input (C,D) and output (A,B) variables. In the translation phase of the CSMP program, the line \( \text{A,B = SUBNAM(C,D)} \) is placed in the calculational sequence and rewritten as CALL SUBNAM(C,D,A,B). The output variables (A,B) are thus listed after the input variables in the call of the subroutine. However, the subroutine definition in example I is similar to that of examples II and III.

In the translation phase of the CSMP program, a so-called UPDATE subroutine is produced. In this subroutine the equations programmed by the user between the labels TITLE and STOP are placed in the correct calculational sequence, and sortable CSMP subroutine calls are rewritten as explained. The statements between STOP and ENDOBJ, however, are not sorted by the CSMP compiler, because this part of the program is pure FORTRAN. As a consequence, one should know how the sortable CSMP subroutine call in the main program is interpreted by the CSMP...
translator, and then define the subroutine accordingly after the label STOP. Appendix 3 gives a summary of the processing of a CSMP program by the CSMP translator to form the UPDATE subroutine and a control file CONTRO.SYS, and of the subsequent conversion of UPDATE to machine code by the FORTRAN compiler. The file UPDATE.FOR contains a readable version of the UPDATE subroutine: it may be inspected to see how CSMP has sorted the user-defined program.

Sortable CSMP subroutine calls like example I are correctly interpreted if two or more arguments occur to the left side of the '=' sign. Otherwise, CSMP will interpret the call as a function, such as AFGEN, INSW and LIMIT (see Chapter 3, Table 1), where the result of the calculation is stored in the name of the function. Although subroutines with only one output argument will probably not occur frequently, misinterpretation may be avoided by incorporating an additional (dummy) argument to the left side of the '=' sign. When a subroutine is invoked via a CALL statement (examples II and III in Figure 18) no interpretation by the CSMP translator will occur, and the use of a dummy argument is not needed.

Functions The result of the calculation of function subroutines is stored in the name of the function. Since this name can only contain one numerical value at a time, a function can, in principle, only be used to convey a single value to the program that called it (see also Subsection 7.3.4: 'The blank COMMON and the labelled COMMON').
The numerical value contained in the name of a function will be of a certain type. When, for example, the REAL variable 'A' occurs in the CSMP program in the statement A = FUNNAM(C,D), the definition between STOP and ENDJOB would be REAL FUNCTION FUNNAM(C,D) (see for the use of a function Figure 20 in Subsection 7.3.3). A number of examples of the combined use of FORTRAN and CSMP is given in Leffelaar et al. (1986).

7.3.2 The structure of a subroutine

Several aspects of the structure of a subroutine are presented in Figure 19. The subroutine is defined between the

```
********************************
SUBROUTINE SUBNAM(N,C,D,A,B)
$IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION C(N+1),A(N)
DIMENSION LOCAL(23)
SAVE
IF (.... .GT. ....) THEN
  DO 10 I=1,N
  .
10   CONTINUE
ELSE
  IF (.......) RETURN
  .
ENDIF
LOCAL(1) =
  .
RETURN
END
********************************
```

Figure 19. General structure of a subroutine.

lines 'SUBROUTINE SUBNAM(N,C,D,A,B)' and 'RETURN, END'. The arguments following the name of the subroutine take care of the communication between the main program and the subroutines; they are further discussed in Subsection 7.3.4.

The line IMPLICIT REAL (A-Z) declares all variables as REAL. Subsequently, the exceptions, here integers, are declared in the INTEGER line. This procedure is inspired by CSMP, where by default all variables are real (Section 3.8), whereas the integer exceptions in the CSMP main program are listed on the FIXED label.

Array variables in FORTRAN have to be given memory space by the label DIMENSION, which is the equivalent of the STORAGE label in CSMP. For local array variables, such as LOCAL, the appropriate memory space is reserved by giving it a bracketed number directly following the name of the variable, e.g. LOCAL(23). Variable dimensioning, i.e. C(N+1), A(N), is possible only for array variables that are present in the list of arguments.

The SAVE statement causes the local variables of the subprogram to retain their values between successive calls.

The upper limit of the DO-loop, the variable N, is also transferred from the main program to the subroutine via the list of arguments. By using variable dimensions and DO-loop limitations, calculations in a subroutine can be fully controlled from the main program. Hence, no changes are required in the subroutine if a new run with, for
example, a different number of layers, is to be made, and the risk of making errors will be reduced.

Since subroutines and functions are independent (sub)programs, all variables which are not transferred from the program invoking them to the subroutine or function, are local variables. Hence, the local variable LOCAL in subroutine SUBNAM is different from a variable named LOCAL in the main program.

All usual FORTRAN rules apply to subroutines. Some important rules are demonstrated in Figure 19:
- Statements should begin in column 7 or higher; numbers of continuation labels must be placed in the first 5 columns; column 6 is reserved to indicate if the line is a continuation of the preceding one by placing e.g. a dollar ($) sign there;
- Unless overruled by the 'IMPLICIT REAL (A-Z)' statement, variables beginning with I, J, K, L, M or N are considered integer, while the remaining variables are considered real;
- More than one RETURN statement may be used to return to the place in the CSMP program where the subroutine was called. However, a subroutine is always terminated by the statements RETURN, END.

7.3.3 The use of CSMP HISTORY functions in subroutines and functions

History functions require both past and present values of the input variables to calculate their output. Examples of CSMP history functions are AFGEN, NLFGEN, FUNGEN, IMPULS and SAMPLE, which require 5, 10, 10, 4 and 3 storage locations, respectively (IBM Corporation, 1975). When history functions are used in structure statements in the CSMP model, so between TITLE and STOP, their storage requirements are supplied automatically by the translator. When they are used in a subprogram this is not the case and the user has to take measures. Figure 20 pre-

```
TITLE Using history functions in a...
subroutine in CSMP
INITIAL
HISTORY SUBNAM(15)
DYNAMIC
A, B = SUBNAM(C, D)
TERMINAL
END
STOP
********************************************************************
SUBROUTINE SUBNAM(NLOC,C,D,A,B)
IMPLICIT REAL (A-Z)
INTEGER NLOC
SAVE
RESULT1 = AFGEN(NLOC,C,D)
RESULT2 = FUNGEN(NLOC+5,C,D)
RETURN
END
********************************************************************
ENDJOB
```

```
TITLE Using a history function in a...
nested function subroutine in CSMP
INITIAL
HISTORY SUBNAM(5)
DYNAMIC
A, B = SUBNAM(C, D)
TERMINAL
END
STOP
********************************************************************
SUBROUTINE SUBNAM(NLOC,C,D,A,B)
IMPLICIT REAL (A-Z)
INTEGER NLOC
SAVE
RESULT = NESTED(NLOC,C,D)
RETURN
END
********************************************************************
ENDJOB
```

Figure 20. Two program structures in which subroutines containing CSMP history functions occur.
sents two program structures containing subroutines with a history function. The subroutine call \( A,B = \text{SUBNAM}(C,D) \), together with the information that this subroutine contains a history function, as supplied by the statement \( \text{HISTORY SUBNAM}(15) \) in the first example, is interpreted by the CSMP translator as CALL \( \text{SUBNAM}(1,C,D,A,B) \) and placed in subroutine \( \text{UPDATE} \). Thus, the output \( (A,B) \) again follows the input of the subroutine \( (C,D) \), but an additional first argument does appear in the subroutine call. The value assigned to this first argument by the CSMP compiler, called '1' here for the time being, is the first position in the blank \( \text{COMMON} \) (see Subsection 7.3.4) from where the 15 memory places required will be reserved for the AFGEN and the FUNGEN that appear in \( \text{SUBNAM} \). In the subroutine definition a variable \( NLOC \) is positioned on the first location in the list of arguments. The variable \( NLOC \) must be added by the modeller, because the subroutine definition is placed between \text{STOP} and \text{ENDJOB}, which is not interpreted by the CSMP compiler. When history functions are used in subroutines or functions, the number of variables in the list of arguments of the call in the main CSMP program is thus not identical to the number in the definition. However, the subroutine call is correctly expanded in the \( \text{UPDATE} \) by the CSMP translator.

Many non-history CSMP functions may be used within subroutines and functions, but specifically excluded is the \( \text{INTGRL} \)-function. The use of functions like \( \text{INSW}, \text{LIMIT}, \text{EXP}, \text{and SQRT} \) in subprograms is similar to their use in CSMP. For functions starting with \( I, J, K, L, M \) or \( N \) (e.g. \( \text{INSW} \) or \( \text{LIMIT} \), which are examples of function subroutines) the FORTRAN type conventions of variables apply, and the names need to be declared real either by the line \( \text{REAL} \) \( \text{INSW} \), \( \text{LIMIT} \), or implicitly as explained in Subsection 7.3.2.

7.3.4 Communication between the main program and subroutines and functions

Variables can be transferred between the main program and subroutines, and between subroutines, in two ways: via the list of arguments or via \( \text{COMMON} \) memory blocks.

The list of arguments The list of arguments is the bracketed string containing the variable names following the name of the subroutine. Transfer of variables between the calling program and the subroutine is effectuated via corresponding positions in the list, not via corresponding names in the list:

<table>
<thead>
<tr>
<th>this occurs on transfer</th>
<th>this does not occur</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL ( \text{SUBNAM}(C,D,A,B) )</td>
<td>CALL ( \text{SUBNAM}(C,D,A,B) )</td>
</tr>
<tr>
<td>CALL ( \text{SUBNAM}(C,D,A,B) )</td>
<td>CALL ( \text{SUBNAM}(C,D,A,B) )</td>
</tr>
<tr>
<td>SUBROUTINE ( \text{SUBNAM}(C,D,A,B) )</td>
<td>SUBROUTINE ( \text{SUBNAM}(A,C,B,D) )</td>
</tr>
<tr>
<td>SUBROUTINE ( \text{SUBNAM}(A,C,B,D) )</td>
<td>SUBROUTINE ( \text{SUBNAM}(A,C,B,D) )</td>
</tr>
</tbody>
</table>

Therefore, the number of arguments in the subroutine call and in the subroutine definition must correspond, and corresponding positions must contain the same variable type (e.g. \( \text{REAL} \) or \( \text{INTEGER} \)). The variable names in the call and the subroutine definition need not necessarily be the same. Within a subroutine, identification of variables takes place via the name, like in the main program.

A possibility to help construct the list of input and output variables in an orderly manner is according to the function of the variable (van Kraalingen and Rappoldt, 1989):

\[
\text{SUBROUTINE \( \text{SUBNAM} \) ( control variables,} \\
\text{variables to initialize the routine,} \\
\text{input variables,} \\
\text{output variables )}
\]

Within each category the variables may be ordered e.g. according to the following sequence: integers, single reals and
array variables. In each section the variables can be arranged alphabetically, whereas each category should start on a new line.

The basic principle is that once a choice has been made, the programmer should consistently adhere to that rule and specifies somewhere, for instance in the list of abbreviations, how the list of arguments has been set up. In CSMP programs the list of arguments may not consist of more than 8 continuation lines.

The blank COMMON and the labelled COMMON A COMMON block is an area of memory, that is shared by those parts of the program that declare the block. A COMMON should only be used if it is impossible to use argument lists, since maintaining COMMON blocks is laborious, which may easily introduce errors.

Two types of COMMON blocks exist: the unnamed (blank) COMMON, written as COMMON VAR1, VAR2, VAR3, and the named or labelled COMMON, written as COMMON /COMNAM/ VAR1,VAR2,VAR3. Here, COMNAM is the name of the COMMON block.

CSMP uses a blank COMMON block to communicate the values of the variables between the CSMP user program and the CSMP system that takes care of the time loop, the integration algorithm, etc. Variables occurring in the CSMP program that are automatically included in the blank COMMON are:
- variables on a STORAGE statement;
- variables on a INTGRL statement (because automatic STORAGE is done by the program);
- variables at left-hand side of an equal-sign (including the variables on the CSMP sortable subroutine call or a procedure);
- variables on a TABLE statement;
- variables on a FUNCTION statement;
- variables on a PARAM, INCON and CONSTANT statement;
- variables on a TIMER statement.

Consequently, under no circumstances a blank COMMON should be used, otherwise its organization by CSMP is disrupted. Experienced programmers can at best use the CSMP blank COMMON by typing the word in the first six columns of the line following the subroutine definition. Then, all variables of the main CSMP program are available to the subroutine.

Only the variables occurring in the blank COMMON of CSMP can be printed by the PRINT or OUTPUT labels. This implies that a variable that is calculated in a subroutine and that does not appear to the left hand side of the equal-sign in the CSMP part (e.g. A and B in example III of Figure 18) cannot directly be printed: first care should be taken that the variable appears in COMMON. This can easily be achieved by stating e.g. AA=A after the subroutine call. Then, the value of A can be printed via AA (see also Exercise 91g in Chapter 11). Variables can always be printed by means of FORTRAN WRITE statements (see Subsection 10.2.1).

Figure 21 shows a hierarchical subroutine structure, in which two subroutines at a low level share several common variables. Transfer of variables via lists of arguments would require those variables to appear also in the lists of arguments of the subroutines at the middle and high levels. In such a case it is advisable to use a labelled COMMON. As indicated above, the result of a calculation in a function is returned to the calling program via its name, implying that only single numerical values can be returned. Through the use of a labelled COMMON block, however, it is possible to communicate other variables between a function and the calling program.
7.3.5 Subroutines with various functions

To construct clearly structured programs, it helps to distinguish subroutines with different functions (see also van Kraalingen & Rappoldt, 1989), e.g.:
- Routines that carry out calculations with the input variables to calculate e.g. initial values before the actual simulation can start;
- Routines that generate or read the environmental conditions for the simulation model, i.e. the driving variables for the model. This may be a simple function or it may involve the reading of a file, for example weather data. These driving variables are not affected by the model calculations, because there is no feedback (the system does not influence its environment);
- Routines where it is decided (IF-THEN-ELSE-ENDIF) which (rate) calculations should be selected in relation to the environmental conditions;
- Routines that contain the differential equations;
- Service routines in which, for example, linear interpolation occurs (AFGEN in CSMP) or in which the roots of a function are calculated. These routines will often be found in subroutine libraries (IMSL, 1987). Therefore, they are independent of the actual model;
- Routines that organize the output of the model.

Exercise 59
In Exercise 41 h, a simulation model was developed for a lake that is polluted by river water.
- a. Which integration method would you choose for this model?
- b. Calculate the time interval of integration for this method.
- c. Write a CSMP simulation model for this system, placing the differential equation in a subroutine.

Exercise 60
In Exercise 55 d, a simulation model was developed for oil that was transported along a pipeline from a first reservoir to a second one, and from there to a third reservoir, where it was collected.

Write a CSMP simulation model for this system, placing the differential equations in two subroutines.
7.4 Concluding remark

The programs introduced in this book are all very short and easy to understand. This is done because each program should clearly demonstrate a new feature. However, there is no reason why the presented techniques would not be combined to very large simulation models, which may be as complicated as necessary. When programs are written within the CSMP main program alone, i.e. between TITLE and STOP, the limits of the CSMP system, as reported in the FOR03.DAT file (Appendix 3), might be bypassed. When programs consist mainly of subroutines, however, there is no danger that these limits will be exceeded, because the FORTRAN code is not processed by the CSMP system, while the CSMP model itself may be very small. The main CSMP model thus forms a summary of the processes in the simulation model and it may thus be easier to understand. Therefore, it seems most appropriate, and it is advised, to develop structured programs using subroutines.
8 Modelling of ageing, development, delays and dispersion

J. Goudriaan and H.J.W. van Roermund

8.1 Introduction

In a model for a population of identical, synchronized individuals, the development stage can be treated as a single state variable. When there is a distribution of ages or of stages of development, the so-called boxcar train is a suitable method to simulate the development process of the entire population. Some dispersion (variability) may occur within the rates of development of different individuals. Three types of a boxcar train are possible, differing mainly in this dispersion of development rate. In the first type, the escalator boxcar train, dispersion is virtually absent. In the second type, the fixed boxcar train, the dispersion is quite large, and is rigidly determined by the number of boxcars. The third type, the fractional boxcar train, includes a parameter which allows the dispersion to be varied between these two extremes, and to be altered during the simulation process itself.

The CSMP-statements, and FORTRAN-subroutines used to implement these methods are given, and the approach is illustrated with a simple application.

8.2 Development and delay

A good example of a stable and well measurable rate of development can be found in bird's eggs. The time duration between laying of the egg and its hatching is rather fixed. Both moments are two clearly marked milestones in the life of a bird. Because the duration of time between these moments is rather stable, the rate of hatching in a population of birds can be found as the rate of egg laying, delayed over the period of brooding. Such delays are quite common in the description of biological processes. For instance, the well-known equation for exponential growth:

\[
\frac{dy}{dt} = r \cdot y
\]

Equation 42

can be rewritten more specifically to describe the number of adult birds as:

\[
\frac{dy}{dt} = f \cdot y_{t-p} - \mu \cdot y_t
\]

Equation 43

where \( p \) stands for the duration of the egg and juvenile stages together. In Equation 42 the variable \( r \) means the overall relative growth rate. Equation 43 is written as a combination of the relative rate of egg-laying, \( f \), and the relative death rate \( \mu \) of the adult birds.

When the delay period is always the same, a simple CSMP-function (Table 1, Section 3.6) can be used to implement the delay:

\text{OUTFL} = \text{DELAY}(20, \text{PERIOD, INFLOW})

where the rate INFLOW is delayed over a period PERIOD to produce the outflow rate OUTFL. The number 20 stands for the number of sampling points describing the shape of the inflow rate during the delay period. This number should typically be of the order of \text{PERIOD/DELMAX} (see Subsection 7.2.3 for the meaning of DELMAX). Unfortunately, the first argument in the DELAY function must be a number and not a parameter. A major limitation
of the DELAY-function is that it cannot be used with variable delay periods. Another limitation is that no operations are possible on the quantities that are being delayed, such as mortality or emigration.

---

Exercise 61

a. Write a CSMP-model for the differential equation, Equation 43, using one state variable for birds, and one delay function for eggs. Parameterize as follows: initial number of adult birds ADULT is 1000; delay period PERIOD is 200 days; relative rate of laying eggs RLE is 0.01 egg per day per bird, and relative death rate, RDR is 0.0 per day; simulation period FINTIM is 1000 days.

b. Replace the DELAY function by one state variable JUVS. The relative rate of reaching the adult stage RRA is 0.5% per day, so that the mean residence time is again 200 days. Initialize JUVS at zero.

The structure of the model in b) is almost the same as the one in a), except for a different formulation of the rate of reaching the adult stage.

c. Compare the results of both models.

---

8.3 Simulation of development of a population of identical, synchronized individuals

In warm-blooded animals, development and ageing can hardly be distinguished, but in other organisms their rates can be completely different. For instance, plants of the same species may flower at moments that are more determined by temperature and daylength than by time since emergence. Usually, at low temperatures the developmental processes run much slower than at high temperatures.

Discernible stages of development have received names, for instance 'anthesis', 'dough-ripe' (in wheat), 'silking' (in maize), and are largely species specific. Numbers have been attached to these stages, so that their quantification could become easier. For instance, in the general scale of development, such as used in the crop growth model SUCROS87 (Spitters, et al., 1989), 'anthesis' was given the value 1, and 'maturation' the value 2. In specialized literature, more refined scales have been developed, e.g. for wheat (Zadoks et al., 1974; Reinink et al., 1986), for maize (Hanway, 1963; Groot et al., 1986) and for rape seed (Remmelzwaal & Habekotté, 1986).

Rate of development can be defined as the numerical distance between two stages, divided by the time required to pass from one stage to the other. The difficulty with empirical scales is often that the time intervals between subsequent stages are not equal, even under constant conditions. Then the empirical scale must be projected on a fictive scale that meets this requirement of a homogeneous rate. Alternatively, the rate of development varies with developmental stage.

In the following example (Remmelzwaal & Habekotté, 1986) a simulation approach will be shown for the development of rape seed from the stages 9 (end flowering) to 15 (maturation). Over this range the development rate (DEVR) is proportional to temperature above 6 °C. Above 25 °C it is supposed that no further increase of rate of development occurs. This means that the response between 6 and 25 °C can be quantified in terms of degree-days with a base temperature of 6 °C. To proceed from stage 9 to stage 15, there are 490 degree-days needed.

---

Exercise 62

a. How long would this period be at 10 °C and how long at 20 °C?

b. What would be the rate of development at these temperatures?
The state variable 'stage of development' (STAGE) can now be simulated by the following CSMP-statements:

\[
\begin{align*}
\text{STAGE} & = \text{INTGRL}(9., \text{DEVR}) \\
\text{DEVR} & = \text{AFGEN}(\text{DEVRTB}, \text{TA}) \\
\text{FUNCTION DEVRTB} & = (0., 0.), (6., 0.), (25., 0.233), (30., 0.233)
\end{align*}
\]

where TA stands for air temperature in °C, and DEVRTB for DEVelopment Rate TaBle with the development rate as a function of temperature.

Exercise 63
Check the consistency of this model and the manual calculation of Exercise 62 by completing and running this model for 10 °C and for 20 °C.

8.4 The boxcar train

In the example given above, it was possible to keep track of the stage of development of the entire population merely, because all individuals were synchronized. This situation is quite common in field crops. In insect populations, especially if they are polyvoltine (more than one generation per season), several stages of development occur simultaneously. Each stage would then require its own simulation. Also new generations are continuously born and add to the complexity of the population structure.

The boxcar train technique takes care of all possible developmental stages simultaneously. Before the simulation starts, the developmental axis of one stage is broken up into a number of classes or boxcars, each with identical development width. If necessary, several separate boxcar trains may be chained, for instance, one to allow for the egg stage, one for the juvenile stage and one for the adult stage. This separation may be necessary to fulfill the requirement of homogeneity of rate of development within a boxcar train. Then it is much easier to realize boxcars of identical developmental width, at least within the boxcar train.

After this classification, the contents of each boxcar train is initialized with the number of individuals in it. Now, in principle a histogram can be drawn for the distribution of development of the population in the stage considered (Figure 22a).

![Figure 22. Distribution of number of individuals A or of concentration c with stage of development g. The final value of g is called g_f and y stands for for the width of a single boxcar. For explanation see text.](image-url)
If a higher resolution of the development axis would be required, the number of boxcars in the boxcar train should be increased (Figure 22b). In Figure 22b, the vertical axis is normalized so that the total area is still the same as in Figure 22a. This can be achieved by plotting vertically not simply the number per boxcar, but this number divided by its developmental width $\gamma$. The number obtained in this way is called a concentration ($c$), because of the analogy with a salt solution in a stratified soil. The class-wise distribution can be considered as an approximation of the 'true' continuous distribution which could vary with stage $g$ as given by the dashed line in Figure 22b.

If there is no mortality, this concentration distribution function simply shifts to the right without any change in shape. Of course, new individuals may enter at $g = 0$, and at $g = g_f$ individuals are removed from the scene. Theoretically, it is possible to store the shape of the graph of $c(g)$ with a very high degree of resolution, and, accordingly, to simulate the developmental process. However, computational limitations prevent such a high resolution, and we have to live with a discretized representation such as given by histograms in Figure 22a.

The question now is how to allow for the continuously acting developmental drift, which shifts all individuals to a higher stage of development at the same rate. In principle there are two options available to simulate this process:

1. Shift the entire distribution as given by the histograms in Figure 22 continuously to higher values of development, including the boxcar boundaries. Only the beginning ($g = 0$) and the end ($g = g_f$) are fixed. This system is called the escalator boxcar train.

2. Keep the location of the boundaries of the boxcars (the bars in the histogram) fixed, but let the individuals flow from one boxcar to the next one. The rate of movement is proportional to the rate of development, and also to the concentration (height of the bar). This system is called the fixed boxcar train.

These two types of boxcar trains do not differ in the mean delay time. However, in the escalator boxcar train all individuals are about equally delayed, whereas in the fixed boxcar train some are more and others are less delayed. The reason for this variability in delay in the fixed boxcar train is that for each individual the probability to flow to the next boxcar is the same, whether it arrived just recently or has been waiting quite a while. Due to this stochastic process, variance emerges in the residence time in each boxcar. The variance of duration of through-flow through the fixed boxcar train causes a levelling of peaks and dips originally present in the inflow curve. An intensive but brief pulse will be buffered in the relatively long residence time of the boxcars and result in a dispersion during development. This so-called numerical 'dispersion' is an artefact of the system, but it can be turned into a helpful side-effect of the boxcar train. Whenever such variance is observed in nature, the fixed boxcar train may help to simulate this phenomenon, although it does not add any explanatory value about its causes.

It will be explained later, that it is possible to hybridize both systems into the so-called fractional boxcar train. By this hybrid method the degree of variance of the through-flow can be controlled to match the observed variance.

8.5 The escalator boxcar train (delay; no dispersion)

In the escalator boxcar train the developmental process is simulated by a continuous developmental drift of the boxcar boundaries. It is essential that these boundaries are chosen such that each boxcar covers the same developmental width, that means that the duration between two subsequent boundaries are made equal for all of them. After the developmental process has completed one such sub-unit of development, boxcar width $\gamma$ (Figure 23), the entire population has gradually shifted to the right by exactly one boxcar, and so all boxcar numbers can be reset. The process is schematically given in Figure 23. An escalator boxcar train with 4 boxcars is presented here, so that the total development range covered ($g_f$) is equal to $4 \gamma$. Although the rate of development is not necessarily constant, it should change simultaneously for all boxcar boundaries.
Immediately after the start of the simulation a gap opens between \( g=0 \) and the lower boundary of boxcar 1. This gap is filled by a new boxcar, with number 0, which will receive the newcomers into the boxcar train. For the functioning of the boxcar train it does not matter where these newcomers come from, whether they have been generated as an external driving force, or have been produced as offspring from the boxcar train itself, or simply came out of a preceding boxcar train.

On the other end of the boxcar train, the last boxcar is contained between a fixed end boundary and an upward moving lower boundary. The distance between them \( (\gamma - g') \) keeps shrinking and so without outflow the concentration in the last boxcar, \( c_N \) (here \( c_4 \)) would grow beyond limit. Such unlimited compression is prevented by defining a rate of outflow \( Q_{\text{out}} \), as:

\[
Q_{\text{out}} = v \cdot c_N
\]

with

\[
c_N = A_N / (\gamma - g')
\]

where \( v \) is the rate of development, \( A_N \) is the amount in the last boxcar and \( g' \) the cyclic development stage \((0 \leq g' < \gamma, \text{Figure 23})\).

When there is no mortality, the relative rate of decrease of amount \( A_N \) and of the remaining width \( \gamma - g' \) are equal, and so the concentration \( c_N \) does not change. The rate of outflow \( Q_{\text{out}} \) is then proportional to the rate of development \( v \), exactly as we would want it.

When there is mortality, \( Q_{\text{out}} \) does decrease within each developmental cycle \((\gamma)\). In fact all boxcars, including the zeroth and the last one, may or may not loose individuals due to mortality, but they do not exchange them. The shape of the development distribution curve is thus not disturbed by exchange between the classes.

The resetting event occurs when the developmental process has covered the width of one boxcar. At that moment
each boxcar has reached the position which its successor had at the start of the simulation. The boxcar numbers are then reset, the last boxcar is removed entirely and a new zeroth boxcar is opened.

Exercise 64

The escalator boxcar train with 4 boxcars, is used to describe the development of rape seed from stage 9 to stage 15.

a. What is the value of $\gamma$?

b. How would you formulate the development rate $DEVR$?

c. At how many degree-days does the resetting (or 'shift') event occur, if the degree-day simplification is permitted ($6 \leq Ta \leq 25 \, ^\circ C$), and 490 degree-days are needed to proceed from stage 9 to stage 15.

8.6 The fixed boxcar train (delay; fixed dispersion)

With fixed boundaries between the boxcars, there is a continuous forward flow from each boxcar into the next one to allow for the developmental drift. A cascade of water tanks can be used as a physical model to visualize the operation of the fixed boxcar train. The flow out of a boxcar is proportional to its concentration $c$, and to the development rate, $v$:

$$ Q_{i+1} = v \cdot c_i $$

Equation 44

where $Q_{i+1}$ is the flow rate from boxcar $i$ to boxcar $i+1$. The concentration $c_i$ is given by:

$$ c_i = A_i / \gamma $$

Equation 45

For a good understanding of the behaviour of the population contents in the fixed boxcar train, it is best to consider first the simplified situation under a constant development rate $v$. Then each boxcar in itself will act as a first order delay, which has the property that a single sharp input pulse will give rise to an exponentially declining output flow (Figure 24). Of course, the total area of the input pulse and the output pulse is the same.

![Flow rate over time](image)

**Figure 24.** Output flow of one single boxcar in the fixed boxcar train, in response to a sharply peaked input pulse.
The exponential shape of the decline can be derived as follows. Imagine that at time zero the boxcar considered is empty, and that the output flow is consequently zero as well. Suddenly a very brief, sharply peaked input flow fills the boxcar with an amount $A$ (compare the 'infinite rate method' discussed in Section 6.3). According to Equations 44 and 45 the output flow $Q_{\text{out}}$ is directly proportional to the contents $A$:

$$Q_{\text{out}} = A \cdot v / \gamma$$

In the situation described, the inflow will be zero immediately after the passage of the brief pulse, and so the differential equation for the contents $A$ is:

$$\frac{dA}{dt} = -A \cdot v / \gamma \quad \text{Equation 46}$$

Since $v$ and $\gamma$ are constant in this simplified situation, the solution of this differential equation for $A$ is an exponentially declining function:

$$A = A_0 \cdot \exp(-v \cdot t / \gamma)$$

with $A_0$ as the initial value of $A$. The outflow $Q_{\text{out}}$ is given by:

$$Q_{\text{out}} = \frac{v}{\gamma} \cdot A_0 \cdot \exp(-v \cdot t / \gamma) \quad \text{Equation 47}$$

With regard to the relationship between the inflow peak and the outflow function (Figure 24), two observations can be made: (1) on average the outflow is delayed with respect to the inflow; (2) the shape of the outflow is more dispersed over time than the inflow.

To find the value of the average delay we should remember that the inflow pulse was localized at time zero, and so the average delay is equal to the mean of time $t$ of the outflow in Equation 47. To find this mean time, $\tau$, time should be integrated between $t_{\text{zero}}$ (start) and infinity, weighted with the value of $Q_{\text{out}}$ and taken relative to $A_0$ (see also Exercise 9, Section 2.5):

$$\tau = \int_0^\infty t \cdot Q_{\text{out}} \cdot dt / A_0$$

Substitution of $Q_{\text{out}}$ according to Equation 47 gives:

$$\tau = \int_0^\infty t \cdot \frac{v}{\gamma} \cdot \exp(-v \cdot t / \gamma) \cdot dt$$

which has the solution:

$$\tau = -\exp(-v \cdot t / \gamma) \cdot (t + \frac{v}{\gamma}) \bigg|_0^\infty$$

or $\tau = \gamma / v$
In fact, this answer is not surprising since $\gamma/v$ is the time coefficient in the argument of the exponential function in Equation 47, and also in differential Equation 46.

So far the derivation only concerned a single infinitely sharp inflow pulse, and one might wonder whether the value of the delay is independent of the shape of the inflow. This is indeed the case, since any shape of the inflow can be broken up into a series of sharp pulses, each giving rise to its own exponentially delayed outflow. Since the whole system is linear (that means outflow is proportional to contents, Equation 46), the contents and the outflows due to the subsequent pulses can be simply added. Therefore, the average delay is also equal to $\gamma/v$ (or $\tau$) independently of the shape of the inflow. Each boxcar will add this delay to result in the total delay of the entire boxcar train:

$$T_{\text{total}} = N \cdot \tau$$

Equation 48

With regard to the average value of the total delay, the fixed boxcar train and the escalator boxcar train are similar. The difference between these methods appears in the effect on the shape of the outflow: the outflow is identical in shape to the inflow in the escalator boxcar train, and it is much more levelled (dispersed) in the fixed boxcar train. This means that a considerable amount of variance has been added to the time distribution of the inflow in the fixed boxcar train.

Statistically spoken, the variance $\sigma^2$ is the second order moment of the time of outflow, which can be calculated as the mean value of $(t-\tau)^2$, weighted with $Q_{\text{out}}$ and taken relative to $A_0$:

$$\sigma^2 = \frac{1}{A_0} \int_0^\infty (t-\tau)^2 \cdot Q_{\text{out}} \cdot dt$$

$$= \text{deviation} \quad \text{quantity transferred}$$

With $Q_{\text{out}}$ given by Equation 47, and using $\gamma/v = \tau$, $\sigma^2$ can be written as

$$\sigma^2 = \int_0^\infty (t-\tau)^2 \cdot \frac{1}{\tau} \exp(-t/\tau) \cdot dt$$

By use of a table of indefinite integrals, e.g. Burington (1973) and of some algebra, this expression can be shown to lead to:

$$\sigma^2 = \tau^2$$

Since each boxcar will add this amount of variance, irrespective of the others and of its position in the boxcar train, the total amount of variance added by the entire boxcar train is given by:

$$\sigma_{\text{total}}^2 = N \cdot \tau^2$$

The combination of this expression with the one for the total delay (Equation 48) gives the interesting result for the relative dispersion $RD$:

$$\sigma_{\text{total}}/T_{\text{total}} = 1/\sqrt{N}$$
The relative 'dispersion' RD (or 'coefficient of variation' CV) of the time of outflow in response to a peaked inflow decreases with the number of boxcars N. This result can be understood if one thinks of the better resolution in the representation of the development distribution when there are more (but also narrower) boxcars in the boxcar train.

**Exercise 65**

How many boxcars are needed to simulate a delay of 20 days with a 'dispersion' of 2 days (RD=0.1)?

A system of N boxcars in the fixed boxcar train is often termed an N\textsuperscript{th}-order delay. The dynamic response of the outflow after a stepwise change in inflow is given in Figure 25 for different values of N (Ferrari, 1978). An example of a 2\textsuperscript{nd} order exponential delay was dealt with in Exercise 55.

![Figure 25. Response of outflow to a stepwise change in inflow for a fixed boxcar train with different numbers of boxcars.](image)

**8.7 The fractional boxcar train (delay; controlled dispersion)**

In comparison with the DELAY-function of CSMP, both the escalator and the fixed boxcar train are much more flexible in that they permit variable development rates. However, they still lack flexibility in the relative dispersion ($\sigma_{\text{total}}/T_{\text{total}}$ or RD) of outflow. The escalator boxcar train has almost no dispersion (except the small amount due to the distribution over a single boxcar) and the fixed boxcar train has a fixed RD of $1/N\bar{N}$, which, once chosen, can only be changed during the simulation after a very great effort. Several experimental data sets show evidence that the delay and the dispersion are not equally influenced by e.g. temperature, and so the relative dispersion also varies. To allow for this change during the simulation a more flexible method than that of the fixed boxcar train is needed. Such a flexible method can be obtained by a hybridization of the methods of the fixed and of the escalator boxcar train. This method will be termed the fractional boxcar train, because it is based on a repeated fractional shift.

In the escalator boxcar train a complete shift to the next boxcar occurs at the moment of resetting. In the fractional boxcar train it is not the complete contents that are shifted, but only a fraction $F$ of each boxcar's contents. To
compensate for the smaller amount, the shift must occur more frequently. Whereas in the escalator boxcar train the renumbering (or shift) occurs upon completion of a boxcar width, $\gamma$, in this method the fractional shift occurs upon completion of a fraction $F$ of the boxcar width. This fraction ranges between 0 and 1 and can be changed during the simulation. A possible time path of the boundaries between the boxcars is illustrated in Figure 26. The value of the fraction $F$ determines how often the resetting occurs, and the partial transfer of contents. When $F$ is equal to 1, the escalator boxcar train is effectively restored (Figure 23). On the other hand, when $F$ approaches zero, the sawtooth shape of the boundaries between the boxcars is practically straight so that the fixed boxcar train is approached. During simulation the value of $F$ can be varied anywhere between these two extremes so that gradual adaptations can be made in the value of the desired dispersion.

Since the movement through the boxcars is pulsewise, the differential equations must be replaced by difference equations. The cyclic development stage, $g'$, stands for the amount of development elapsed since the last resetting occurred. In the escalator boxcar train, $g'$ triggers the renumbering as soon as it exceeds $\gamma$. Here, in the fractional boxcar train, the trigger level is set at $F \cdot \gamma$. When this level is exceeded, the fractional shift occurs and $g'$ is decreased by $F \cdot \gamma$. The contents $A$ of boxcar $i$ are also reduced according to the following operation:

\[ A_{i,j+1} = (1 - F) \cdot A_{i,j} \]

where $j$ counts the number of shifts since the start. Here, as for Equation 46, it is assumed that the inflow into $A_i$ is zero. Expressed in the initial contents $A_{i,j}$ then is given by

\[ A_{i,j} = A_{i,0} \cdot (1 - F)^j \]

A special situation occurs in the zeroth boxcar. The contents of this boxcar are entirely transferred to the first one, so that after the shift,

\[ A_{0,j} = 0. \]

*The delay in the fractional boxcar train* The first fractional shift does not occur at time zero, but only when $g'$
equals \( F \cdot y \). When the development rate, \( v \), is constant, this shift occurs at time \( F \cdot y / v \), or at time \( F \cdot \tau \).

The expression for the average residence time, \( \bar{\tau} \), is:

\[
\bar{\tau} = \frac{1}{A_{i,0}} \cdot \sum_{j=1}^{\infty} j \cdot F \cdot \tau \cdot A_{i,0} \cdot (1 - F)^{j-1} \cdot F. \tag{49}
\]

Equation 49

Exercise 66

Derive Equation 49.

Equation 49 can be evaluated by using the general expression for the sum of the series

\[
\sum_{j=1}^{\infty} j \cdot r^{j-1} = 1 / (1 - r)^2 \quad 0 < r < 1.
\]

In this equation \( r \) can be replaced by \( 1 - F \), and Equation 49 yields then

\[
\bar{\tau} = \tau.
\]

This result shows that the delay per boxcar is independent of the value of \( F \). Also the total delay \( T_{total} \) of the boxcar train is independent of \( F \), and equal to \( N \cdot \tau \).

The variance in the fractional boxcar train. The variance can be evaluated from

\[
\sigma^2 = \frac{1}{A_{i,0}} \cdot \sum_{j=1}^{\infty} (j \cdot F \cdot \tau - \tau)^2 \cdot A_{i,0} \cdot (1 - F)^{j-1} \cdot F. \tag{50}
\]

Equation 50

By using the sum of the series given above, and also the following one:

\[
\sum_{j=1}^{\infty} j^2 \cdot r^{j-1} = \frac{1+r}{(1-r)^3} \quad 0 < r < 1,
\]

we find that Equation 50 can be simplified to:

\[
\sigma^2 = \tau^2 \cdot (1 - F).
\]

This result shows that the variance is linearly related to the value of the fraction \( F \). This variance occurs in each boxcar so that the total variance of the boxcar train as a whole is:

\[
\sigma_{total}^2 = N \cdot \tau^2 \cdot (1 - F). \tag{51}
\]

Equation 51
8.8 Implementation of the boxcar train in CSMP

The fixed boxcar train (see Figure 27) First the three types of boxcar train will be applied in a simple example. Consider a boxcar train that consists of 5 boxcars. The boxcar train represents a single development stage, such as the egg stage of birds or insects. An array INTGRL is used to represent the state variables ‘A’ in the 5 boxcars:

\[
A = \text{INTGRL}(AI, RA, 5)
\]

with initial content zero:

\[
\text{TABLE AI}(1-5) = 5*0.
\]

DO-loops are used for the calculation of the rates, and concentrations:

\[
\begin{align*}
\text{DO } 20 & \text{ I } = 1, 5 \\
C(I) & = A(I) / \text{GAMMA} \\
\text{CONTINUE}
\end{align*}
\]

(see Equation 45)

Here GAMMA is the name for the mathematical symbol \( \gamma \), the width of a boxcar in development units, and is defined as \( gf/N \), with \( gf = 1 \). This implies that the full range of development, \( gf \), has the numerical value of unity. This is an arbitrary choice which could have been 100 or any other figure. In case of another value for the full range of development the width of the boxcars is adapted proportionally.

The rate of flow from boxcar \( I \) to boxcar \( I+1 \) is given by the rate of development, \( v \) (DEVR), multiplied by the concentration \( C(I) \):

\[
\text{FLOW}(I+1) = \text{DEVR} \cdot C(I).
\]

(see Equation 44)

The rate of development (DEVR) is integrated as well to yield the physiological time, \( g \), which can be considered as the state of development of an immortal individual born at time zero:

\[
G = \text{INTGRL}(0., \text{DEVR}).
\]

It should be noted that \( G \) always increases.

An inflow pulse similar as in Figure 24, of total size UNITY (=1) occurs at time zero:

\[
\text{INFL} = (\text{UNITY/DELT}) \cdot \text{IMPULS}(0.0, \text{FINTIM}).
\]

This inflow into the boxcar train, is assigned to FLOW(1):

\[
\text{FLOW}(1) = \text{INFL}.
\]

The outflow from the boxcar train is made equal to FLOW(N+1):

\[
\text{OUTFL} = \text{FLOW}(N+1).
\]

The net flow of a boxcar is given by:

\[
\text{NETFLO}(I) = \text{FLOW}(I) - \text{FLOW}(I+1)
\]

and, since there is no mortality so far,

\[
\text{RA}(I) = \text{NETFLO}(I).
\]
Figure 27. Listing of a program, including the fixed boxcar train.

**TITLE** Fixed boxcartrain

**INITIAL**

**FIXED** N,I

**STORAGE** FLOW(6), NETFLO(5), C(5)

**TABLE** AI(1-5)=5*0.

**PARAM** DEVR =0.005, N=5, GF=1.0, UNITY=1.0

**TIMER** FINTIM =1000., PRDEL=100., DELT =1.

**PRINT** A(1-5),AOUT,ATOT,BALANC,ADP,VAR,S

**METHOD** RECT

* Calculations

**GAMMA** =GF/FLOAT(N)

**DYNAMIC**

**NOSORT**

* Operations on the state variables

**G** =INTGRL(0.,DEVR )

**A** =INTGRL(AI,RA ,5)

**AOUT** =INTGRL(0.,OUTFL )

**INCUMM**=INTGRL(0.,INFL )

* Balance should be zero

**ATOT** =0.

DO 10 I=1,N

**ATOT** =ATOT + A(I)

10 CONTINUE

**BALANC** =ATOT + AOUT - INCUMM

* Optional statements for study and checking purposes

**ADPG** =GF/DEVR

**ADP** =INTGRL(0., (TIME * OUTFL) / UNITY)

**VAR** =INTGRL(0., ( (TIME-ADPG)**2 * OUTFL) / UNITY)

**S** =SQRT(VAR)

* Calculation of the rates

**INFL** =(UNITY/DELT)*IMPULS(0.0,FINTIM)

**FLOW(1)** =INFL

DO 20 I=1,N

**C(I)** =A(I)/GAMMA

**FLOW(I+1)=DEVR * C(I)**

**NETFLO(I)=FLOW(I) - FLOW(I+1)**

* no mortality

**RA(I)** =NETFLO(I)

20 CONTINUE

**OUTFL** =FLOW(N+1)

**END**

**STOP**

**ENDJOB**
The inflow into the boxcar train and the outflow from it are also collected in separate integrals so that a balance of material may be set up:

\[
\begin{align*}
A_{\text{OUT}} &= \text{INTGRL}(0., \text{OUTFL}) \\
\text{INCUMM} &= \text{INTGRL}(0., \text{INFL}) \\
\text{BALANC} &= \text{ATOT} + A_{\text{OUT}} - \text{INCUMM}
\end{align*}
\]

where ATOT is the total amount in the 5 boxcars, representing the number of individuals of the development stage considered.

In this example the inflow was kept at zero, except for a single pulse at time zero with height UNITY/DELT. This discontinuous behaviour of the inflow requires the use of Euler's integration method (Section 6.3).

The following optional statements for monitoring purposes will normally not be included in a model but study of their formulation and behaviour will improve the understanding. For this simple situation the average delay period of the boxcar as a whole (ADP) could be calculated by the equation in Exercise 9 or numerically:

\[
\text{ADP} = \text{INTGRL}(0., \text{TIME} \times \text{OUTFL/UNITY}).
\]

The expression for the variance (VAR) is given by:

\[
\text{VAR} = \text{INTGRL}(0., (\text{TIME}-\text{ADPG})^2 \times \text{OUTFL/UNITY}).
\]

ADPG must be a constant, and in fact be equal to the final value of ADP at time FINTIM. Therefore, two runs would be required if ADP would be used: the first one to find ADP and the second one to find VAR. In this simple case it may be calculated from \( g/v \). Fortunately, in regular simulations there is no necessity to calculate both ADP and VAR. Here it is done just to check our model and compare it with the theoretical results.

Here, we would like to have the variables A(1-5), AOUT, ATOT, BALANC, ADP and VAR as printed output on intervals PRDEL. ATOT is an important variable, because this is the total number of individuals in a certain stage, which is also usually measured. Variable ATOT, together with BALANC, is also used as a check of our bookkeeping. Although they may seem trivial, in more complicated models these balance variables are extremely useful to detect omitted or double defined flows.

The development rate (DEVR) is given in a PARAMeter statement as 0.005, N as 5.

---

**Exercise 67**

Use the program of Figure 27 to generate the curves of Figure 25 for \( N = 1, 2, 5 \) and 10.

**Exercise 68**

a. Calculate the average delay period (ADP) and its variance (VAR) by hand for \( N = 1, 2, 5 \), and 10.

b. Compare these values with the simulated (mimicked) values (\( T_{\text{total}} \) respectively \( \sigma^2_{\text{total}} \)).

c. When are the simulated values significant?

---

*The escalator boxcar train (Figure 28)* Most statements are the same as for the fixed boxcar train, so only the differences will be mentioned. The criterion for renumbering (shifting) is that the cyclic development stage, \( g' \), (GCYCL) has reached the value GAMMA (the width of a boxcar):

\[
\text{IF} (\text{GCYCL.GE.GAMMA}) \text{ CALL SHIFT(N, GAMMA, GCYCL, A, A0}).
\]
Figure 28. Listing of a program, including the escalator boxcar train.

TITLE Escalator boxcar train
INITIAL
NOSORT
FIXED N , I
STORAGE C(5)

TABLE AI(1-5)=5*0.
PARAM DEVR =0.005, N =5 , GF =1. , UNITY=1.0
TIMER FINTIM =220.0, PRDEL=20., DELT=1.
PRINT A0,A(1-5),AOUT,ATOT,BALANC,G,GCYCL,ADP,VAR,S
METHOD RECT

* Calculations
GAMMA =GF/FLOAT(N)
* The cyclic development starts halfway
G CYCLI =0.5*GAMMA

DYNAMIC
NOSORT

* Operations on the state variables
G CYCL =INTGRL(G CYCLI,DEVR )
G =INTGRL(0. ,DEVR )
A0 =INTGRL(0. ,RA0 )
A =INTGRL(AI ,RA ,5)
AOUT =INTGRL(0. ,OUTFL )
INCUMM=INTGRL(0. ,INFL )

IF (G CYCL .GE. GAMMA) CALL SHIFT(N,GAMMA,G CYCL,A,A0)

* Balance should be zero
ATOT = A0
DO 10 I=1,N
ATOT = ATOT + A(I)
10 CONTINUE
BALANC = ATOT + AOUT - INCUMM

* Optional statements for study and checking purposes
ADPG =GF/DEVR
ADP =INTGRL(0. , (TIME * OUTFL) / UNITY)
VAR =INTGRL(0. ,((TIME-ADPG)**2 * OUTFL) / UNITY)
S =SQRT(VAR)

* Calculation of the rates
INFL = (UNITY/DELT)*IMPULS(0.0,FINTIM)
RA0 = INFL

DO 20 I=1,N-1
* no mortality
RA(I)=0.0
20 CONTINUE
C(N) = A(N)/(GAMMA-G CYCL)
OUTFL = DEVR * C(N)
RA(N) =-OUTFL

END
STOP
The subroutine 'SHIFT' itself is explained at the end of this Section. The use of the IF-statement requires a preceding NOSORT label. The shift itself is formulated in a FORTRAN subroutine with the necessary arguments N (number of boxcars), and the names of the treated integrals. The zeroth boxcar (A0) appears separately, because unfortunately the indexing of array integrals in CSMP begins with one and not with zero. In the main program A0 is thus also formulated separately:

\[ A_0 = \text{INTEGRAL}(0., R_{A0}) \]

and

\[ R_{A0} = \text{INFL} \]

Since the flows between boxcars are zero (by definition in the escalator boxcar train) the names FLOW and NETFLO are not needed. Only the outflow from the last boxcar must be defined (see Section 8.5):

\[ O_{OUT} = \text{DEVR} \times C(N) \]

and assigned to the outflow rate from the last boxcar:

\[ R_{A(N)} = - O_{OUT} \]

The other rates of change of the boxcars are zero, because there is no mortality assumed yet:

\[ \text{DO } 10 \text{ I = 1, } N-1 \]
\[ R_{A(I)} = 0.0 \]
\[ 10 \text{ CONTINUE} \]

The concentration C(N) of the last boxcar must be calculated in a different way than in the fixed boxcar train (see Section 8.5):

\[ C(N) = \frac{A(N)}{(\text{GAMMA} - \text{GCYCL})}. \]
The cyclic development stage, GCYCL, is formulated in exactly the same way as G itself:

\[ \text{GCYCL} = \text{INTGR}(\text{GCYCL}, \text{DEVR}) \]

It is not equal to G, however, because GCYCL is modified in the subroutine SHIFT.

The subroutine 'SHIFT' The shift is performed in subroutine shift. The process of shifting must be done backwards, because otherwise the contents would be replaced by the preceding contents, which is equal to the first integral value. It is important that GCYCL is decreased by GAMMA, and not reset to zero. The shift is triggered when the state of the cyclic development stage, \( g' \), reaches \( \gamma \), but this will usually not occur at an exact multiple of DELT. The excess value of \( g' \) above \( \gamma \) at the first opportunity for the shift must be retained. The process of shifting is an application of the 'replace method' in a state event, but now the criterion at which the replacement should occur is somewhat looser than in Subsection 7.2.4, and the amount of overshoot of \( g' \) above \( \gamma \) is affected by the time interval of integration. Therefore, this time interval should not be taken too large. Furthermore, method RECT is used, implying that KEEP equals 1 all the time (Section 7.2.3). Therefore, there is no need to test for the integration status.

The escalator boxcar train, applied to a demographic problem For demography the best method is the escalator boxcar train, because age, which is used as a characteristic, does not disperse. To illustrate its use, the same example will be given for the growth of the Dutch population as by de Wit & Goudriaan (1978). For clarity only the female part of the population is simulated, the male part being taken for granted. The fraction of boys at birth (FRBOY), is used as an input variable. The age dependence of relative death rate and of relative birth rate is given in Figure 29. The corresponding fraction of survival (FS) is found by simulating a single cohort from birth onwards. Mathematically the relative death rate (RDR) and the fraction survival (FS) are related by:

\[ \text{RDR} = - \frac{d\text{FS}}{da} / \text{FS} \]

where \( a \) stands for age.

![Figure 29](image_url) Figure 29. The age dependence of the relative death rate RDR, and of relative birth rate RBR. The fraction survival (fs, dashed line) is a function of RDR.

The CSMP-program used is given in Figure 30. Data are supplied about the initial age distribution of the population. This is done by a TABLE specifying the contents of the 20 5-year classes of the population array AI. Then two
TITLE Growth of the Netherlands population
INITIAL
FIXED I,N
STORAGE AGE(20),MORR(20),FRACA(20)
* Initial contents of female ageclasses of 5 years wide, expressed
* in thousands
TABLE AI(1-20)=582.,587.,553.,543.,554.,420.,381.,380.,378., ... 
* Fraction young born boys
PARAM FRBOY =0.512
* 100 years of age (AGETOT=ADP) is covered in 20 classes (N)
PARAM AGETOT =100., N=20
* Annual relative death rate, in promille per year, as a function
* of age, specifically for females.
FUNCTION ARDTB= 0.0,10., 2.5,4.0, 5.,1.8, 7.5,0.8, 10.0,0.5, ... 
    15.0,0.3, 20.,0.3, 30.,0.6, 40.,1.6, 50.0,4.9, ... 
    60.0,8.5, 65.,14., 70.,25., 75.,55., 82.5,180., ... 
* Annual relative birth rate per year, in promille per year, as a
* function of age, for boys and girls.
FUNCTION ARITB= 0.,0., 12.5,0., 17.5,20., 22.5,137., 25.,166., ... 
    27.5,188., 30.,166., 32.5,113., 37.5,55.0, ... 
    42.5,16.0, 47.5,2.0, 50.0,0.0, 105.0,0.0
* Time is expressed in years
TIMER FINTIM =1000., DELT=1., PRDEL=100.
PRINT A0,A(1-20),AOUT,ATOT,FRACAO,FRACA(1-20),TBR,TMORT,BALANC
METHOD RECT
* Initial calculations
* Residence time in one age class
    TC =AGETOT/FLOAT(N)
* Development rate is one year per year and constant
    DEVR =1.0
* Development width covered by one ageclass
    GAMMA =AGETOT/FLOAT(N)
* The fraction girls of the young borns
    FRGIRL =1.-FRBOY
* Initialization of integrals
    ATOTI =0.
    DO 10 I=1,N
        AI(I)=AI(I)*1000.
        ATOTI=ATOTI + AI(I)
    10 CONTINUE

DYNAMIC
NOSORT
* Operations on the state variables
* Development
    GCYCL = INTGRL(0.,DEVR )
* State variables in the boxcar
    A0    = INTGRL(0.,RA0 )
    A     = INTGRL(AI,RA,20)
    AOUT  = INTGRL(0.,OUTFL)
* When GCYCL exceeds GAMMA, the shift is applied.
* In this situation, this is after every 5 years
  IF (GCYCL .GE. GAMMA) CALL SHIFT(N,GAMMA,GCYCL,A,A0)

* Balance should be zero
  ATOT = A0
  DO 20 I=1,N
       ATOT= ATOT+A(I)
  20 CONTINUE

  TOTMOR = INTGRL(0.,TMORT)
  TOTBIR = INTGRL(0.,TBR )
  BALANC = ATOT + TOTMOR + AOUT - ATOTI - TOTBIR

* Calculation of the age-distribution
  FRACA0 = A0/ATOT
  DO 30 I=1,N
       FRACA(I)=A(I)/ATOT
  30 CONTINUE

* Calculation of the rates----------------------------------------------
* The total birth rate of girls is first set to zero
  TBR = 0.
**The zero's boxcar
* The age of the centre
  AGE0 = 0.5 * GCYCL
* The mortality rate
  MORR0 = A0 * 0.001*AFGEN(ARDTB,AGE0)
  TMORT = TVDRR
**The 1 - (N-1)'s boxcar
  DO 40 I=1,N-1
* The age of the centre of each boxcar
  AGE(I) = TC*(FLOAT(I)-0.5)+GCYCL
* Mortality rate of each class
  MORR(I)= A(I)*0.001*AFGEN(ARDTB,AGE(I))
* Total mortality
  TMORT = TMORT + MORR(I)
* The 1 - (N-1)'s class only change by death
  RA(I) =-MORR(I)
* Total birth rate of girls
  TBR = TBR + A(I) * ( FRGIRL * 0.001*AFGEN(ARDTB,AGE(I)) )
  CONTINUE

**The N's boxcar
  AGE(N) = TC*(FLOAT(N)-0.5) + 0.5*GCYCL
  ARDN = 0.001*AFGEN(ARDTB,AGE(N))
  MORR(N) = A(N)*ARDN
  TMORT = TMORT + MORR(N)
* The zero's class increases by birth, and decreases by death
  RA0 = TBR - MORR0
* The N's boxcar decreases by death and by an outflow of people
* older than 100 years old.
  CN = A(N)/(GAMMA-GCYCL)
  OUTFL = DEVR*CN*(1.0-ARDN*DELT)
  RA(N) =-OUTFL - MORR(N)

END
STOP

***********************************************************************
FUNCTIONs with a list of coordinate points of the relationship between relative death and birth rates and age are supplied. In the INITIAL segment some computations are done for the discretization of age and development scale, necessary before the actual simulation in the DYNAMIC segment. The simulation itself requires computation of the rates of change, and of course the integration of these rates, which is done by the INTGRL-statement. Also whole population totals are computed by summation over all age classes in DO-loop 20. The data supplied for the FUNCTIONs are read by an AFGEN-statement.

Exercise 69
Find FS in Figure 29 by simulation, using the program in Figure 30. Make the birth rate constant, for instance 1000 per year, and after 100 years or more the age distribution will have the same shape as the fraction FS.

Exercise 70
In the normal simulation with the birth rate coupled to the fertile women, the population size will not be constant. After long enough a period of simulation, an exponential growth will result with a corresponding age distribution and relative growth rate. To find these, run the model as given here for 1000 years. Compare the age distribution with FS found in Exercise 69.

The model simulating the growth of the Netherlands population (Figure 30) is run with METHOD RECT and with a time interval of integration of 1 year. This choice was made for practical reasons such as easy comparison of the model results with available statistical data. However, there are some effects on the numerical values of input parameters such as the relative death rate and ard, and on the model formulation concerning the integration method and the definition of the outflow from the Nth boxcar, that are quite important.

Formally and normally, rates are defined on an instantaneous basis, and not as a mean value during a time interval. According to this definition, when the relative death rate rdr has a value of 1 per year, this value does not mean that after one year all are dead. It means that after one year a fraction exp(-1) is still surviving.

In this model for the Netherlands population however, the annual relative decrease or annual relative death rate, ard, is used instead of the relative death rate. The ard is the total fraction of the population that dies over the period of
one year in relation to the population at the beginning of the year. Annual relative death rates can never exceed the numerical value of 1. The relation between the annual relative death rate and the formal definition of $rdr$ above, was derived in Section 2.5 as:

$$1 - e^{-rdr \cdot \Delta t} = ard \cdot \Delta t$$

with $\Delta t$ chosen as 1 year. This choice means that the numerical value of the parameter $ard$ is linked to the choice of the value of the time interval and of the integration method, in this case METHOD RECT in connection with a time interval of one year. Although this procedure is dangerous, because both the integration method and the time step should not be changed, it is preferred here because of the practical reasons mentioned.

The outflow from the last boxcar element is defined as

$$OUTFL = DEVR \times CN \times (1 - ARDN \times DELT)$$

where $ARDN$ is the annual relative death rate for the $N^{th}$ class. This definition implies that the outflow is corrected for the natural death of the population over the year. The corrected outflow refers to the outflow concerning the people that have reached the upper limit of the physiological age. This correction is related to the choice of METHOD RECT and a time step of one year.

If $OUTFL$ was calculated from the instantaneous value, the outflow would be defined by development rate times concentration, without a correction (Section 8.5). Then, the correction would be automatically taken care of by the numerical integration and a value of the time step that would not be necessarily one, but related to the smallest time coefficient in the model.

Here, the outflow is calculated from the contents at the beginning of each time interval. As indicated above, $ARDN \times DELT$ stands for the fraction of the people that dies during one year. The remaining fraction is subject to the continuous boxcar shift out of the last element. Without this correction, the fraction that dies during one year would also be included in the developmental outflow.

*The fractional boxcar train (Figure 31)* The general aspects of the fractional boxcar train are treated here. The discussion of the application contained in this program, is given in Section 8.9. This boxcar train can be used when the relative dispersion of through-flow is known to vary during the simulation, so both the desired delay and the dispersion must be computed in the DYNAMIC segment of the main program. Instead of delay, the rate of development, $v$, is used in the program, because it is more linearly related to the temperature:

$$DEVR = AFGEN(DERV, TEMP).$$

At low temperatures, the rate of development is completely halted ($v=0$), which corresponds to an infinitely long delay. When the lower temperature threshold is passed, development takes off proportional to the excess temperature above the threshold. In an AFGEN function, this threshold point must be included.

**Exercise 71**

The table describes the dependence of a development rate on temperature:

<table>
<thead>
<tr>
<th>$T_a$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.03</td>
</tr>
<tr>
<td>8</td>
<td>0.05</td>
</tr>
<tr>
<td>10</td>
<td>0.15</td>
</tr>
<tr>
<td>20</td>
<td>0.15</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

a. Choose the minimum number of data points needed for a proper description of these data in an AFGEN function.
b. What happens if the threshold point is not included? What value of $v$ would then be calculated at 10 °C?
c. Could you design a corresponding AFGEN function for the delay period?
Figure 31. Listing of a program including the fractional boxcar train.

TITLE Development Pandemis heparana 1982
INITIAL
FIXED NA ,NB ,NC ,ND ,NE ,NF ,NG ,NH
PARAM NA=4,NB=10,NC=8,ND=3,NE=10,NF=4,NG=10,NH=10
PARAM STDAY =32.
CONSTANT PI =3.1415926
* Relative Mortality Rate
PARAMETER RMRSL3 =0., RMRSL4 =0., RMRSL5=0., RMRML6=0.,...
     RMRFL6=0., RMRDIA=0., RMRMP=0., RMRFP =0.
* Initial numbers in boxcartrains
TABLE IDIA(1-4) = 4*25.
TABLE IL3(1-10) =10* 0.
TABLE IL4(1-8) = 8* 0.
TABLE IL5(1-3) = 3* 0.
TABLE IML6(1-10)=10* 0.
TABLE IMP(1-10) =10* 0.
TABLE IFP(1-10) =10* 0.
* Fraction male
PARAM SEXR =0.5
* Fraction DIA3/DIA2: this fraction is the ratio of L3 / ( L2 + L3 ) in the
diapause stage DIA. So, in the diapause stage already larvae are present.
* Therefore, when they emerge from the diapause stage, the L3 directly
* flows to the L4 boxcar, while the L2 flows to the L3 boxcar train.
PARAM G =0.7
PRINT  DAY,DIATOT,CUPL3,CUPL4,CUPL5,CUPML6,CUPFL6,...
     CUPMP,CUPFP,CUPMM,CUPFM,BALANC,
     DIATOT,L3TOT,L4TOT,L5TOT,ML6TOT,FL6TOT,MPTOT,FPTOT,CUMMM,CUMFM
TIMER  FINTIM=180.,PRDEL=20.,DELT=0.0416666
METHOD RECT
* Development Rate
FUNCTION DRDIAT =-10.,0.0000, 6.0,0.0000, 11.,0.0181, 13.,0.0251,...
     16.,0.0412, 19.,0.0621, 22.,0.0831, 25.,0.1015,...
     35.,0.2585
FUNCTION DRL3T =-10.,0.0000, 9.4,0.0000, 16.,0.1093, 25.,0.2585,...
     35.,0.2585
FUNCTION DRL4T =-10.,0.0000, 8.0,0.0000, 16.,0.1155, 25.,0.2460,...
     35.,0.2460
FUNCTION DRL5T =-10.,0.0000, 7.2,0.0000, 16.,0.1088, 25.,0.2204,...
     35.,0.2204
FUNCTION DRML6T =-10.,0.0000, 6.3,0.0000, 16.,0.0704, 25.,0.1355,...
     35.,0.1355
FUNCTION DRFL6T =-10.,0.0000, 7.8,0.0000, 16.,0.0594, 25.,0.1246,...
     35.,0.1246
FUNCTION DRMPT =-10.,0.0000, 8.2,0.0000, 16.,0.0551, 25.,0.1190,...
     35.,0.1190
FUNCTION DRFPT =-10.,0.0000, 8.2,0.0000, 16.,0.0551, 25.,0.1190,...
     35.,0.1190
* Relative Dispersion
FUNCTION RDDIAT =-10.,0.00, 6.0,0.0000, 11.,0.23, 13.,0.32, 16.,0.18,...
     19.,0.45, 22.,0.20, 25.,0.24, 35.,0.24
FUNCTION RDL3T = -10., 0.00, 9.4, 0.00, 16., 0.11, 19., 0.11, ...
  22., 0.10, 25., 0.13, 35., 0.13
FUNCTION RDL4T = -10., 0.00, 8.0, 0.00, 13., 0.30, 16., 0.19, 19., 0.18, ...
  22., 0.24, 25., 0.30, 35., 0.30
FUNCTION RDL5T = -10., 0.00, 7.2, 0.00, 11., 0.23, 13., 0.57, 16., 0.15, ...
  19., 0.23, 22., 0.11, 25., 0.13, 35., 0.13
FUNCTION RDML6T = -10., 0.00, 6.3, 0.00, 11., 0.25, 13., 0.16, 16., 0.14, ...
  19., 0.14, 22., 0.07, 25., 0.08, 35., 0.08
FUNCTION RDFL6T = -10., 0.00, 7.8, 0.00, 11., 0.25, 13., 0.06, 16., 0.49, ...
  19., 0.10, 22., 0.13, 25., 0.16, 35., 0.16
FUNCTION RDFMPT = -10., 0.00, 8.2, 0.00, 11., 0.03, 13., 0.06, 16., 0.08, ...
  19., 0.06, 22., 0.22, 25., 0.26, 35., 0.26
FUNCTION RDFDPT = -10., 0.00, 8.2, 0.00, 11., 0.03, 13., 0.06, 16., 0.08, ...
  19., 0.06, 22., 0.22, 25., 0.26, 35., 0.26

* Minimum daily temperature in degree Celsius
FUNCTION MNTT = ...
  1., 2.1, 2., 4.5, 3., 6.4, 4., 7.4, 5., 8.2, ...
  6. , -7.0, 7., -8.5, 8., -5.8, 9., -11.2, 10., -12.9, ...
  etc. weather data Wageningen, 1982

* Maximum daily temperature in degree Celsius
FUNCTION MXTT = ...
  1., 7.0, 2., 11.0, 3., 11.6, 4., 11.4, 5., 9.8, ...
  6. , .1., 7., -1.8, 9., -3.6, 9., -5.3, 10., -3.9, ...
  etc. weather data Wageningen, 1982

* Initial calculations
  GAMMAA=1. / FLOAT(NA)
  GAMMAB=1. / FLOAT(NB)
  GAMMAC=1. / FLOAT(NC)
  GAMMAD=1. / FLOAT(ND)
  GAMMAE=1. / FLOAT(NE)
  GAMMAF=1. / FLOAT(NF)
  GAMMAG=1. / FLOAT(NG)
  GAMMAH=1. / FLOAT(NH)

* Total initial amount of diapause larvae
  TINDIA=DIA(1)+DIA(2)+DIA(3)+DIA(4)

DYNAMIC
NOSORT

*--------------------------------------------------------------

* Operations on the state variables
DIA =INTGRL(IDIA,DDIA, 4)
DIAO =INTGRL(O., DIAO, )
L3 =INTGRL(IL3, DL3, 10)
L30 =INTGRL(O., DL30, )
L4 =INTGRL(IL4, DL4, 8)
L40 =INTGRL(O., DL40, )
L5 =INTGRL(IL5, DL5, 3)
L50 =INTGRL(O., DL50, )
ML6 =INTGRL(IML6, DML6, 10)
ML60 =INTGRL(O., DML60, )
FL6 =INTGRL(IFL6, DFL6, 4)
FL60 =INTGRL(O., DFL60, )
MP =INTGRL(IMP, DMP, 10)
MPO =INTGRL(O., DMP0, )
FP = INTGRL(IFP , DFP , 10)
FP0 = INTGRL(0. , DFP0)

GCYCLA = INTGRL(0. , DRDIA)
GCYCLB = INTGRL(0. , DRL3)
GCYCLC = INTGRL(0. , DRL4)
GCYCLD = INTGRL(0. , DRL5)
GCYCLE = INTGRL(0. , DRML6)
GCYCLF = INTGRL(0. , DRFL6)
GCYCLG = INTGRL(0. , DRMP)
GCYCLH = INTGRL(0. , DRFP)

*Time

DAY = STDAY + TIME

*Temperature

MAXT = AFGEN(MXTT, DAY)
MINT = AFGEN(MNNT, DAY)
AVTEMP = 0.5* (MAXT + MINT)
AMPTMP = 0.5* (MAXT - MINT)
TEMP = AVTEMP + AMPTMP*(-COS(2.*PI*TIME))

*Development Rate

DRDIA = AFGEN(DRDIAT, TEMP)
DRL3 = AFGEN(DRL3T , TEMP)
DRL4 = AFGEN(DRL4T , TEMP)
DRL5 = AFGEN(DRL5T , TEMP)
DRML6 = AFGEN(DRML6T , TEMP)
DRFL6 = AFGEN(DRFL6T , TEMP)
DRMP = AFGEN(DRMP , TEMP)
DRFP = AFGEN(DRFP , TEMP)

*Relative Dispersion

RDDIA = AFGEN(RDDIAT, TEMP)
RDL3 = AFGEN(RDL3T , TEMP)
RDL4 = AFGEN(RDL4T , TEMP)
RDL5 = AFGEN(RDL5T , TEMP)
RDML6 = AFGEN(RDML6T, TEMP)
RDFL6 = AFGEN(RDFL6T, TEMP)
RDMR = AFGEN(RDMP , TEMP)
RDFF = AFGEN(RDF , TEMP)

*Shifting the boxcar train contents, if appropriate
CALL FSHIFT(1, NA, TIME, DELT, DRDIA, RDDIA, DIA, DIA0, GAMMAA, GCYCLA)
CALL FSHIFT(2, NB, TIME, DELT, DRL3, RDL3, L3, L30, GAMMAB, GCYCLB)
CALL FSHIFT(3, NC, TIME, DELT, DRL4, RDL4, L4, L40, GAMMAC, GCYCLC)
CALL FSHIFT(4, ND, TIME, DELT, DRL5, RDL5, L5, L50, GAMMAD, GCYCLD)
CALL FSHIFT(5, NE, TIME, DELT, DRML6, RDML6, ML6, ML60, GAMMAE, GCYCLE)
CALL FSHIFT(6, NF, TIME, DELT, DRFL6, RDFL6, FL6, FL60, GAMMAF, GCYCLF)
CALL FSHIFT(7, NG, TIME, DELT, DRMP, RDMR, MP, MPO, GAMMAG, GCYCLG)
CALL FSHIFT(8, NH, TIME, DELT, DRFP, RDFF, FP, FPO, GAMMAH, GCYCLH)

* Additional calculations for output and checking purposes.
* These should be placed after completion of all operations on the
* state variables, which is the case here
CALL BOXTOT(NA, DIA, DIA0, DIATOT)
CALL BOXTOT(NB, L3, L30, L3TOT)
CALL BOXTOT(NC, L4, L40, L4TOT)
CALL BOXTOT(ND, L5, L50, L5TOT)
CALL BOXTOT(NE, ML6, ML60, ML6TOT)
CALL BOXTOT(NF, FL6, FL60, FL6TOT)
CALL BOXTOT(NG, MP, MF0, MPTOT)
CALL BOXTOT(NH, FP, FP0, FPTOT)

TOTMOR = INTGRL(0., TMORR)
TOTIN = INTGRL(0., PIDIA)
CUML3 = INTGRL(0., PIL3)
CUML4 = INTGRL(0., PIL4)
CUML5 = INTGRL(0., PIL5)
CUMML6 = INTGRL(0., PIML6)
CUML6TOT)
CUMFL6 = INTGRL(0., PIFL6)
CUMMP = INTGRL(0., PIMP)
CUMFP = INTGRL(0., PIFP)
CUMMM = INTGRL(0., PDMP)
CUMFM = INTGRL(0., PDFP)

CUPL3 = 100.*CUML3/(TINDIA*(1.-G))
CUPL4 = 100.*CUML4/TINDIA
CUPL5 = 100.*CUML5/TINDIA
CUPML6 = 100.*CUMML6/(TINDIA*SEXR)
CUPFL6 = 100.*CUMFL6/(TINDIA*(1.-SEXR))
CUPMP = 100.*CUMMP/(TINDIA*SEXR)
CUPFP = 100.*CUMFP/(TINDIA*(1.-SEXR))
CUPMM = 100.*CUMMM/(TINDIA*SEXR)
CUPFM = 100.*CUMFM/(TINDIA*(1.-SEXR))

* Balance should be zero
ATOT = DIAATOT + L3TOT + L4TOT + L5TOT + ML6TOT + FL6TOT + ...
METOT + FPTOT + CUMMM + CUMFM
BALSNC = ATOT - TINDIA + TOTMOR - TOTIN

*-----------------------------------------------------------------------*
* Rate calculations

PIDIA = 0.
CALL BOXRAT(NA, DRDIA, RDDIA, PIDIA, RMRDIA, DIA, DIAO, GCYCLA, ...
GAMMAA, DELT, ...
DDIA, DDIAO, MRDIA, PDDIA)

PIL3 = (1.-G)*PDDIA
CALL BOXRAT(NB, DRL3, RDL3, PIL3, RMRL3, L3, L30, GCYCLB, ...
GAMMAB, DELT, ...
DL3, DL30, MRL3, PDL3)

PIL4 = G * PDDIA + PDL3
CALL BOXRAT(NC, DRL4, RDL4, PIL4, RMRL4, L4, L40, GCYCLC, ...
GAMMAC, DELT, ...
DL4, DL40, MRL4, PDL4)

PIL5 = PDL4
CALL BOXRAT(ND, DRL5, RDL5, PIL5, RMRL5, L5, L50, GCYCLD, ...
GAMMAD, DELT, ...
DL5, DL50, MRL5, PDL5)

PIFL6 = (1.-SEXR) * PDL5
PIML6 = SEXR * PDL5
CALL BOXRAT(NE, DRML6, RDML6, PIML6, RMRL6, ML6, ML60, GCYCLE, ...
GAMMAE, DELT, ...
DML6, DML60, MRML6, PDML6)

CALL BOXRAT(NF, DRFL6, RDFL6, PIFL6, RMRFL6, FL6, FL60, GCYCLF, ...
SUBROUTINE FSHIFT(NUMBER,N,TIME,DELT,DEV,RD,A,AO,GAMMA,GCYCL)
IMPLICIT REAL (A-Z)
INTEGER I,N,NUMBER
DIMENSION A(N)
SAVE

F = 1. - FLOAT(N)*RD*RD
C Check on DELT and on N
IF (DELT*DEV .GT. F*GAMMA) THEN
WRITE(6,'(2A,I3,A,1F12.3)')
$ ' DELT too large or too many boxes N ',
$ ' in boxcar train number ',NUMBER,
$ ' at time ', TIME
CALL EXIT
ENDIF

IF (GCYCL .GE. F*GAMMA) THEN
A(N) = A(N) + A(N-1)*F
DO 10 I=N-1,2,-1
A(I) = A(I)*(1.-F) + A(I-1)*F
106
10 CONTINUE
   A(1) =A(1)*(1.-F) + A0
   A0 =0.

   GCYCL =GCYCL - F*GAMMA
ENDIF
RETURN
END

C**********************************************************************C
SUBROUTINE BOXRAT(N,DEVR,RD,INFL,RMR,A,A0,GCYCL,$
   GAMMA,DELT,$
   DA,DAO,MORFL,OUTFL)
IMPLICIT REAL (A-Z)
INTEGER N ,I
DIMENSION A(N),DA(N)
SAVE
c-----mortality rate (MORR) and total mortality flow (MORFL)
MORR =RMR * A0
MORFL =MORR
DAO =INFL -MORR
DO 10 I=1,N
   MORR = RMR * A(I)
   DA(I)=-MORR
   MORFL= MORFL + MORR
10 CONTINUE
c-----the rate of outflow (OUTFL) is calculated
CN =A(N)/(GAMMA - GCYCL)
OUTFL =DEVR * CN * (1.0 - RMR*DELT)
DA(N) =DA(N) - OUTFL
RETURN
END

************************************************************************
* SUBROUTINE BOXTOT calculates the total contents of a boxcar train
* Inputs: N,A,A0
* Output: ATOT

SUBROUTINE BOXTOT(N,A,A0,ATOT)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION A(N)
SAVE

   ATOT =A0
   DO 10 I=1,N
      ATOT=ATOT+A(I)
10 CONTINUE
RETURN
END

************************************************************************
In a similar way the relative dispersion \( RD (\sigma_{total}/T_{total}) \) is a function of temperature:

\[
RD = AFGEN(RDT, TEMP).
\]

The procedure to calculate the rates in the fractional boxcar train is described in a subroutine called 'BOXRAT' in Figure 31. Note the difference in names between the subroutine itself (general names) and the call in the main program (specific names, see Subsection 7.3.4):

```plaintext
SUBROUTINE BOXRAT(N, DEVR, RD, INF, RMR, A, A0, GCYCL,
$                GAMMA, DELT,
$                DA, DAO, MORFL, OUTFL)
```
called by:

```plaintext
CALL BOXRAT(NA, DRDIA, RDDIA, PIDIA, RMRDIA, DIA, DIA0, GCYCLA, 
            GAMMAA, DELT,  ... 
            DDIA, DDIA0, MRDIA, PDDIA).
```

Inputs for this subroutine are: the number of boxcars in the train (NA), the Development Rate of the insects in the DIApauze stage (DRDIA), the relative dispersion (RDDIA), the inflow into the zeroth boxcar (PIDIA), the relative mortality rate (RMRDIA), the actual contents in the state variables DIA and DIA0, and in the cyclic development stage GCYCLA, the development width (GAMMAA), and the time interval of integration (DELT). These inputs have to be given or calculated in the main program. Outputs of this subroutine are the rates of change of the zeroth boxcar and of the other boxcar elements (DDIA and DDIA0), the total mortality rate in the boxcar train (MRDIA), and the outflow of the last (Nth) boxcar (PDDIA). The number of boxcars NA has to be declared integer

**fixed NA**

In the initial segment of the main program, the initial amount in each boxcar and the number of boxcars in the boxcar train have to be given. For instance, when NA is equal to 4:

```plaintext
TABLE IDIA(1-4) = ....
PARAM NA = 4
```

The fraction \( F \) for the fractional boxcar train is calculated in the subroutine FSHIFT by Equation 51 (Section 8.7), written with \( F \) at the left-hand side:

\[
F = 1. - FLOAT(N) \times RD \times RD
\]

When this computed value of \( F \) is used, the fractional boxcar train will produce the desired dispersion. Although this method is flexible, it cannot be stretched beyond its limits. An undue unbalance between the number of boxcars \( N \) and the desired relative dispersion (RD) will be revealed in nonsense values of \( F \). Theoretically, it is immediately clear that \( F \) must lie between zero and unity. The upper end of this range is approached when the number of boxcars \( N \) is rather small for the low RD that we wish to simulate. To select an appropriate value for \( N \), it should be kept in mind that a minimum amount of dispersion cannot be avoided. Even when \( F \) is set at unity, some dispersion remains within the boxcar width. Usually this amount can be neglected.

The lower end of the range of \( F \) is (theoretically) zero, but a closer inspection tells that \( F \) must also be larger than the fraction that would normally be transferred in each integration interval in the fixed boxcar train: \( \text{DELT} \times \text{DEVR}/\text{GAMMA} \). If \( F \) is smaller than this value, the number of boxcars \( N \) was chosen too high. The numerical value of \( N \) must be smaller than 1 divided by the maximally occurring value of RD squared. Experience has shown
that about 3/4 of this maximum number is usually sufficient. If this does not solve the problems, it may also be
that DELT is too large.

The method presented here cannot deal with RD-values larger than 1. The maximum relative dispersion possible is
obtained with the fixed boxcar train (with \( F \) close to zero), and 1 boxcar in the train only. In this extreme situation
we have returned to the first order exponential delay, for which dispersion is equal to delay. A further increase in
dispersion will require a completely different simulation method, not discussed here.

Exercise 72
a. Show that if \( F \) is smaller than \( \Delta t \cdot v/\gamma \), the number of boxcars \( N \) was chosen too high.
b. Show that \( N \) must be smaller than 1 divided by the maximally occurring value of RD squared.

It is good practice to include finish conditions for these sorts of criteria. This is done in subroutine FSHIFT:

```fortran
IF (DELT*DEVR .GT. F*GAMMA) THEN
  WRITE (6, '(2A, I3, A, 1F12.3)')
  $ ' DELT too large or too many boxes N ',
  $ ' in boxcar train number ',NUMBER,
  $ ' at time ', TIME
  CALL EXIT
END IF
```

(The FORTRAN WRITE statement will be discussed in Chapter 10.) Both the time interval of integration and the
residence time in the boxcars of a boxcar train characterize the degree of temporal resolution that is required in the
model. Therefore, there is little point in choosing \( \Delta t \) much smaller than the residence time of one boxcar \( \gamma/v \). In
this case the general rule that \( \Delta t \) must not exceed one tenth of the time coefficient of the fastest model component
can be loosened a bit. The requirement here is less stringent thanks to the negative feedback within the boxcar train:
a numerical error in the rate of transition only affects the distribution among adjacent boxcars and has hardly an effect
on removal from the boxcar train as a whole. The conclusion is that \( \Delta t \cdot v \) should not exceed \( F \cdot \gamma \) (or \( \Delta t \)
should not exceed \( F \cdot \gamma/v \)) but may well be larger than 1/10 of \( \gamma/v \), leaving aside other reasons to determine
\( \Delta t \).

If the execution of the program is aborted on account of one of these finish conditions, the programmer should first
search for bugs in the data set like the AFGEN-functions or in other parts of the program before he tries to repair the
failure by an adaptation of \( N \) or of \( \Delta T \).

Now that \( F \) has been adequately defined, it is used inside the subroutine FSHIFT:

```
CALL FSHIFT (1, NA, TIME, DELT, DRDIA, RDDIA, DIA, DIA0, GAMMAA, GCYCLA)
```

Inputs for this subroutine are: a number which serves to identify the boxcar train if there is a computing error in \( F \),
the number of boxcars in the train (\( NA \)), the time (\( TIME \)) and the time interval of integration (\( DELT \)), the develop­
ment rate (\( DRDIA \)), the relative dispersion (\( RDDIA \)), the actual contents in the state variables \( DIA \) and \( DIA0 \), the
development width (\( GAMMAA \)), and the actual contents in the state variables cyclic development stage (\( GCYCLA \)).
Outputs of this subroutine are the new contents in the state variables \( DIA \) and \( DIA0 \), and in the cyclic development
stage (\( GCYCLA \)). Thus, the state variables \( DIA \), \( DIA0 \) and \( GCYCLA \) are both input and output variables for the
subroutine, and these variables will have different values upon entrance and return from this routine, if, of course, \( g' \)
\( \geq F \cdot \gamma \). In the subroutine definition FSHIFT itself, the list of arguments is repeated with general names. The cal­
culations in this subroutine, triggered by the statement GCYCL .GE. F*GAMMA, are:
IF (GCYCL .GE. F*GAMMA) THEN
    A(N) = A(N) + A(N-1)*F
    DO 10 I=N-1,2,-1
         A(I) = A(I) * (1.-F) + A(I-1)*F
    CONTINUE
    A(1) = A(1) * (1.-F) + A0
    A0 = 0.
    GCYCL = GCYCL - F*GAMMA
END IF

These lines are sufficient to implement the theory of Section 8.7. There are no further changes in comparison to the escalator boxcar train.

8.9 A practical application using the fractional boxcar train

To simulate the population development of insects, the fractional boxcar train can be used. This will be illustrated with an example of an orchard tortricid moth, *Pandemis heparana* (DENN. et SCHIFF.) which is an important pest in European apple orchards. Control of these insects relies mainly on the use of broad-spectrum insecticides, but since these also kill beneficial insects, research has been directed towards more specific insect growth-regulators. These prevent metamorphosis but are effective only if applied at the right time: at emergence of the last-instar larvae. With a simulation model of population development the best time of application can be better predicted.

*P. heparana* usually has one generation per year. Adult female moths deposit their eggs in August. Second or third larval instars hibernate. At the end of March, larvae become active; the sixth larval instar pupates in June. From June until August, therefore, adult moths are found. The model, presented in the relational diagram of Figure 32 and the listing in Figure 31, is a simplified version of a model of de Reede & de Wilde (1986), which simulates the post-hibernation phenology of *P. heparana* in 1982 in Wageningen, the Netherlands. Each development stage (diapause-stage, 3rd-5th stage larvae, 6th stage male and female larvae, and male and female pupae) is described by a fractional boxcar train, which mimics the mean delay and the temporal dispersion. Experiments showed temperature to be the only important determinant of the development rate. The temperature is computed as a simple sinusoidal

\[
\text{Figure 32. Relational diagram with main elements in fractional boxcar train for *Pandemis heparana*.
}
curve through the daily maximum and minimum temperature, which were measured in a Stevenson screen, 1.5 m high. The time step for numerical integration is one hour (1/24 = 0.0417 day), to allow for diurnal fluctuations in temperature. The model is started by initializing the total number of diapause larvae in the first boxcar train, the numbers of all other stages are set to zero. Input data are the development rates and the relative dispersions of the various stages at different temperatures, the relative mortality rate, the sex ratio, the initial ratio of the numbers of L₃ and L₂ larvae in diapause, the start day and the minimum and maximum daily temperature. For each development stage (except the adult stage) at each time step the subroutines FSHIFT and BOXRAT are called. Their inputs and outputs are as described in Section 8.8. The total contents of each boxcar train is calculated is a small subroutine BOXTOT.

Figure 33 compares the simulation results with field sample data on L₅, L₆, pupae and adult moths (in which the sexes are combined). From this figure, it is clear that phenology can be well simulated using the fractional boxcar train when temperature fluctuates. The effect of temperature on the development rate and the relative dispersion must be determined from experiments.

Figure 33. Simulated (———) and measured phenology of L₅ (x), L₆ (+), pupae (Δ) and adult moths (o) of Pandemis heparana. On the y-axis the accumulated fractions are shown.

8.10 Effect of the type of boxcar train on population growth

With the fractional boxcar train being intermediate, the greatest difference can be expected between the escalator boxcar train and the fixed boxcar train types. To evaluate the effect of the chosen method, the demographic example (Figure 29 and Figure 30) was used. To establish a control, the fixed boxcar train was also run with 1-year classes, to find an accurate estimate of the equilibrium distribution and of the relative growth rate corresponding to the used mortality and fertility distributions. This equilibrium age distribution was then used as input for both the fixed and the escalator boxcar trains with 5-year classes. The 1-year control yielded an equilibrium relative growth rate of 8.29 \(10^{-3}\) yr\(^{-1}\), which was closely approximated by the 5-year escalator boxcar train (8.27 \(10^{-3}\) yr\(^{-1}\)). The 5-year fixed boxcar train resulted in a lower simulated relative growth rate: 7.24 \(10^{-3}\) yr\(^{-1}\). The explanation for this underestimation is the numerical dispersion in the fixed boxcar train method. In a growing population, as simulated here, the younger age groups will contain more individuals, so that the numerical dispersion will cause an apparent artificial 'ageing', slowing down the simulated population growth.

The price for the high accuracy of the escalator boxcar train is a slight irregularity in the simulated relative growth rate RGR, following a 5-year cycle around the mean value of 8.27 \(10^{-3}\) yr\(^{-1}\), and ranging between 7.97 \(10^{-3}\) yr\(^{-1}\)
and $8.59 \times 10^{-3}$ yr$^{-1}$. This irregularity is caused by the sawtooth ageing tendency of the 5-year classes.

8.11 Concluding remarks

In Exercise 61 two state variables were distinguished: Adults and Juveniles. The exercise was to describe the Juveniles subsequently in two different ways to see whether the description would affect the results. In the first method the CSMP DELAY-function was used to give a precise delay of 200 days. In the second method the Juveniles were described as a first order pool, again with a residence time of 200 days. Now that we have gone through this chapter we see that the first part of the exercise in fact was an escalator boxcar train with 20 compartments, and the second part a first order fixed boxcar train. Both boxcar trains were initialized with zero contents, mainly because the CSMP-function does so by default. The comparison did show that the simulated overall relative growth rate of the population was less in the first method. Apparently the variance of outflow implicit in the fixed boxcar train stimulated the overall relative growth rate of the population as a whole. In terms of mean residence time, the fast developing individuals gained exactly as much as the slowly developing ones lost, but for the growth rate the effect was not equilibrated. The stimulation of the overall growth rate by the fast juveniles was stronger than the decelerating effect of the slow ones. The exercise showed that variance can be important for simulation of overall growth.

Clearly, the fixed boxcar train method is the simplest and if possible, should be preferred for that very reason. As shown in the demographic example, it is much easier to use 1-year classes in combination with a time interval of one year than to use the escalator boxcar train with 5-year classes. Simplicity of program formulation is then bought for computer time, which is often a profitable deal. In both methods, numerical dispersion is avoided.

Compelling reasons for using the more complicated boxcar train versions may be found in the desire to simulate fluctuating rates of development and dispersion. However, one must still remain aware of the unpleasant fact that dispersion is simply mimicked by a numerical tool, and that it is not really simulated from underlying processes. This should stimulate further research so that the true reasons for the dispersion can be explained.

The merit of the methods presented here is that they enable effects of the observed characteristics of population dynamics in other situations to be evaluated, and can also serve as a tool for prognostic and management purposes.
9 Mass flow, numerical dispersion and diffusion

P. A. Leijfelaar

9.1 Introduction

If water containing dissolved salts (e.g. fertilizer) infiltrates into the soil, a gradual increase in salt concentration will be observed at a certain depth. The curve of the change in salt concentration against time at a certain depth is called the breakthrough curve. The rounded shape of this curve can partly be explained by diffusion of the salt in the water, partly by the different rates of flow of the water in pores of different size (hydrodynamic dispersion) and partly by interactions between the salts and the solid phase of the soil. This type of process, which causes the salt front to level out, is summarized by the term natural dispersion. A similar phenomenon of dispersion is observed in the development of living organisms. If a number of seeds, with an average germination period of 20 days, are sown at the same time, the germination times of the individual seeds will be distributed around the 20th day. The shape of the cumulative curve of germinated seeds versus time, will be similar to the breakthrough curve. (Compare for example the curves in Figure 33 of Section 8.9.) The distribution of the germination periods can, in principle, be explained by variation in the ambient conditions in which the seeds germinate and variation in the physiological characteristics of the individual seeds.

A simulation model which attempts to provide an explanation for this natural dispersion, be it for salt transport or for development, must do so explicitly. It is also possible to provide only a description of the dispersion in a simulation model, instead of an explanation. However, in all cases, a stumbling block is ever present, because an artificial dispersion (also called numerical dispersion) is introduced due to the structure of the model itself; this applies to both soil models and population models.

In Chapter 8, Goudriaan and van Roermund demonstrated the way the numerical dispersion phenomenon can be put to good use. The aim of this chapter is to provide insight into the way numerical dispersion filters through into the simulation of physical transport phenomena and to indicate several possibilities for suppressing numerical dispersion, without resorting to the escalator or the fractional boxcar train.

9.2 Mass flow and numerical dispersion in a column of soil

9.2.1 The general transport equation

Consider a soil on which a layer of salt (e.g. fertilizer) is present. To keep things simple, it is assumed that the soil is saturated with water, and has a porosity $\varepsilon$ (m$^3$ of air m$^{-3}$ of soil). Rainfall causes a mass flow with velocity $v'$ and the water moves through the pores with velocity $v = v' / \varepsilon$. The salt concentration in the water at a location $x$ and time $t$ is $C(x,t)$; the effective diffusion coefficient is $D$. The uptake of the salt by plant roots is represented by $R$. The partial differential equation which describes the rate of change of $C$, $\partial C / \partial t$, under the influence of water flow, diffusion and uptake, is then given as:

$$\frac{\partial C}{\partial t} = -v \cdot \frac{\partial C}{\partial x} + D \cdot \frac{\partial^2 C}{\partial x^2} - R.$$  
Equation 52
The term \(-v \partial C/\partial x\) describes the displacement of the undistorted profile of \(C\) in the \(x\)-direction, \(\partial C/\partial x\), with velocity \(v\). The term \(D \partial^2 C/\partial x^2\) describes the evening out of peaks and troughs by diffusion and natural dispersion. The units of the variables are: 
\(C: \text{kg m}^{-3}\), 
\(t: \text{s}\), 
\(x: \text{m}\), 
\(v: \text{m}^3 \text{m}^{-2} \text{s}^{-1} = \text{m s}^{-1}\), 
\(D: \text{m}^2 \text{s}^{-1}\), 
\(R: \text{kg m}^{-3} \text{s}^{-1}\).

Exercise 73
Check whether the units of the terms \(-v \partial C/\partial x\), \(D \partial^2 C/\partial x^2\) and \(R\) are correct.

For the time being, the rate of uptake of nutrients by plant roots \((R)\) is set to zero. The derivation of Equation 52 will be presented in Section 9.4.

9.2.2 Derivation of the differential equations for mass flow

The simplest way of studying the effect of simulation on the levelling out of the profile of \(C\) in the \(x\)-direction, \(\partial C/\partial x\), is to set the natural dispersion to zero, i.e. \(D=0\). Equation 52 then becomes:

\[
\frac{\partial C}{\partial t} = -v \frac{\partial C}{\partial x} \tag{Equation 53}
\]

For modelling transport processes in the soil, it is common to divide the soil column into a number of layers (de Wit & van Keulen, 1975), as shown in the following relational diagram. This diagram will be used to derive the differential equations:

![Relational diagram for setting up the differential equations to simulate mass flow.](image)

Figure 34. Relational diagram for setting up the differential equations to simulate mass flow.

The flux equations for layer I can be derived immediately:

\[
\begin{align*}
\text{FL}(I) &= V \cdot C(I-1) \\
\text{FL}(I+1) &= V \cdot C(I) \\
\end{align*}
\]

Equations 54
in which it is assumed that the salt is evenly distributed within each layer and its concentration equals \( C(I) \) throughout, so that the flux into a layer is the product of the velocity of the inflowing water and the salt concentration in the water of the preceding layer. Note that the formulation in Equation 54 is analogous to that of the fixed boxcar train from Section 8.8.

The net flux equation for layer \( I \), is now found as the difference between the inflow and the outflow:

\[
NFL(I) = FL(I) - FL(I+1) \tag{Equation 55}
\]

or

\[
NFL(I) = V \cdot (C(I-1) - C(I)) \tag{Equation 56}
\]

with units of \( \text{kg m}^{-2} \text{s}^{-1} \). In Equation 53, \( dC/dt \) has units \( \text{kg m}^{-3} \text{s}^{-1} \). (In \( dC/dt \), normal d's are used instead of Greek \( \partial \)'s, since this quotient refers to normal differentials and not to partial differentials.)

The relation between Equation 56 and Equation 53 can be derived as follows. To calculate the change in the amount of matter in a soil layer from the net flux equation, \( NFL(I) \) must be multiplied by the surface area, AREA, through which the flux of matter passes: \( \text{AREA} \cdot NFL(I) \). This product has units of \( \text{kg s}^{-1} \). The rate of change in concentration in the soil layer \( dC/dt \) is subsequently obtained by dividing \( \text{AREA} \cdot NFL(I) \) by the volume of the layer, VOL. In a linear system, i.e. a system in which the surface through which the flux of matter passes is the same at every depth in the soil column, \( VOL = \text{AREA} \cdot \text{TCOM} \), assuming that all layers have equal thickness, TCOM. In other words, the change in concentration in the soil layer of a linear system is given by \( NFL(I)/\text{TCOM} \), and the rate of change of the concentration is found as

\[
\frac{dC(I)}{dt} = \frac{NFL(I)}{\text{TCOM}} = \frac{V}{\text{TCOM}} \cdot (C(I-1) - C(I)) \tag{Equation 57}
\]

The time coefficient for mass flow in Equation 57 is \( TC_{\text{mass}} = \text{TCOM}/V \), expressed in seconds. Equation 57 is a so-called ordinary differential equation, because it contains a single derivative (here with respect to time), whereas its derivation started from the partial differential equation, Equation 53, containing two derivatives, both with respect to time and space. In all models in this text, partial differential equation are reduced to ordinary differential equations which are easier to handle.

Equation 57 could directly be used for the simulation of mass transport in soil, but usually Equations 54 and 55 are used for this purpose. This is illustrated in Figure 35 showing a program entitled 'Elementary description of mass flow'.

**Exercise 74**

a. Study the program in Figure 35 by making a dimensional analysis of the variables in the listing.

b. Run the program and study its output.

c. Do you consider the choice of the values of the parameters adequate?
TITLE Elementary description of mass flow

INITIAL
FIXED I,N
STORAGE FLUX(21).C(20)
TABLE IH(1-20)=20*0.0

* kg
PARAM N=20 , CS=10.0. V=0.1 , TCOM=0.05, AREA=1.0
* number kg m**3 m
* of ------ ------
* layers m**3 m**2 day

TIMER FINTIM=20.0,PRDEL=2.0.OUTDEL=1.0
* day
PRINT C(1-20),FLUX(21)
OUTPUT C(20)

PAGE WIDTH=80

METHOD RKS

ABSERR H(1)=1.0E-4
RELERR H(1)=1.0E-4

* Calculations
VOL =AREA*TCOM

DYNAMIC

NOSORT
H =INTGRL(IH,NFLOW,20)

DO 10 I =1,N
C(I) =H(I)/VOL
10 CONTINUE

FLUX(1) =CS * V
DO 20 I =2,N
FLUX(I) =C(I-1) * V
20 CONTINUE

FLUX(N+1) =C(N) * V

DO 30 I =1,N
NFLOW(I)=FLUX(I)*AREA - FLUX(I+1)*AREA
30 CONTINUE

END
STOP
ENDJOB

9.2.3 Analysis of numerical dispersion with the aid of a Taylor series development

With the aid of a Taylor series, it is possible to express \( C(I-1) \) in Equation 57 in \( C(I) \). Taylor's formula, which can be used to express a continuous function of \( x \) in terms of the value of the function in point \( x_1 \), the derivatives of the function in \( x_1 \), and the distance between \( x \) and \( x_1 \) is (see also Appendix 1):

\[
f(x) = f(x_1) + f'(x_1) \cdot (x-x_1) + \frac{f''(x_1)}{2!} \cdot (x-x_1)^2 + \frac{f'''(x_1)}{3!} \cdot (x-x_1)^3 + \ldots + \frac{f^{(n)}(x_1)}{n!} \cdot (x-x_1)^n.
\]

Equation 58
When the Taylor series is applied to Equation 57, the following substitutions are appropriate, see Figure 34:

- $C(l)$: value of the function at point $x_1$;
- $C(l-1)$: value of the function at point $x$;
- $(x-x_1)$: distance between the positions where $C = C(l-1)$ and $C = C(l)$; since $C(l)$ refers to the concentration at a location deeper in the soil column than $C(l-1)$, $x_1$ is larger than $x$ and $(x-x_1)$ is negative and equal to $TCOM$.

Substituting these terms in Equation 58 yields:

$$
C(l-1) = C(l) - TCOM \cdot \frac{dC(l)}{dx} + \frac{TCOM^2}{2!} \cdot \frac{d^2C(l)}{dx^2} - \frac{TCOM^3}{3!} \cdot \frac{d^3C(l)}{dx^3} + \frac{TCOM^4}{4!} \cdot \frac{d^4C(l)}{dx^4} - \ldots \text{Equation 59}
$$

Substitution of Equation 59 in Equation 57 results in:

$$\frac{\partial C(l)}{\partial t} = -V \cdot \frac{\partial C(l)}{\partial x} + \frac{1}{2} \cdot V \cdot TCOM \cdot \frac{\partial^2 C(l)}{\partial x^2} - \frac{1}{6} \cdot V \cdot TCOM^2 \cdot \frac{\partial^3 C(l)}{\partial x^3} + \ldots \text{Equation 60}
$$

The first term of Equation 60 is identical to that of Equation 53 ($-v \cdot \partial C/\partial x$), but the second term causes an undesired dispersion with an apparent diffusion-dispersion coefficient of $V \cdot TCOM/2 \, \text{m}^2 \, \text{s}^{-1}$. Higher order terms are further neglected. If the actual diffusion coefficient and natural dispersion, $D$, were identical to $V \cdot TCOM/2$, Equation 60 would be identical to Equation 52 without the sink term ($R$).

The average retention time of the salt in the soil column will be proportional to the column length, $N \cdot TCOM$, in which $N$ represents the number of layers, and inversely proportional to the flow rate $V$, namely $(N \cdot TCOM) / V = N \cdot TCOM_{\text{mass}}$ (compare Equation 48 from Section 8.6). Therefore, simply by dividing the total soil depth in segments of finite thickness, a numerical dispersion is introduced, with a dispersion coefficient dependent on the thickness of the segment and the flow rate of the water.

### 9.2.4 Suppression of numerical dispersion

It is not desirable that the simulation methodology itself affects the solution of the problem of mass flow, and one would like to suppress the numerical dispersion.

A simple method to do so would be to reduce the apparent diffusion coefficient $V \cdot TCOM/2$ by diminishing the value of $TCOM$. The soil will then have to be divided into a larger number of layers in order to maintain the same column length. However, the number of calculations, and consequently the calculation time of the model, increases disproportionately with the number of layers.

It is also possible to strongly suppress numerical dispersion by reformulating the flux equation. In Equation 54 the flux is proportional to the concentration in the preceding layer. In the new formulation, this flux is considered proportional to the average concentration of two adjacent layers:

$$FL(l) = V \cdot \left( C(l-1) + C(l) \right) / 2. \quad \text{Equation 61}
$$

The change in concentration with time in a layer of soil then becomes:

$$\frac{dC(l)}{dt} = \frac{NFL(l)}{TCOM} = \frac{V}{TCOM} \cdot \frac{C(l-1) - C(l+1)}{2}. \quad \text{Equation 62}
$$

This formulation causes the even numbered derivatives to disappear from the Taylor series.
Exercise 75

Check that the even-numbered derivatives \( \partial^2 C/\partial x^2, \partial^4 C/\partial x^4, \partial^6 C/\partial x^6, \text{etc.} \) disappear from Equation 62, if the Taylor series are developed for \( C(\ell-1) \) and \( C(\ell+1) \). (Use Equation 59 for \( C(\ell-1) \), and derive a new expression for \( C(\ell+1) \).)

Exercise 76

a. Rewrite the simulation program from Figure 35 by using Equation 61, and name the program: Dispersion corrected description of mass flow.
b. How should the relational diagram in Figure 34 be modified to express the approach according to Equation 61?
c. Compare the results of the model with those from Exercise 74.

From the results of the program developed in Exercise 76, it would seem that the salt front does not level off to the same extent as in Exercise 74, but that the higher-order (odd-numbered) terms in the Taylor series still cause such a distortion that the concentration fluctuates around 10 kg m\(^{-3}\), the value that should be achieved. Whenever the natural diffusion-dispersion is larger than the numerical dispersion, the distortions will be overshadowed. Therefore, the thickness of the layers should be chosen in such a way that \( D > (V \cdot TCOM)/2 \), i.e. \( TCOM < 2 \cdot D/V \), apart from other considerations which determine the selection of the thickness of the layers. Up to now, constant thickness of all layers has been considered. If concentrations show a strong gradient with depth, however, it may be desirable to select a smaller layer thickness, whereas with less pronounced concentration gradients, thicker layers may be chosen. Appendix 4 indicates how the average concentration should be calculated for a varying layer thickness.

Exercise 77

a. Derive the second equation in Appendix 4 from the first one.
b. What is the value of AVC(\ell) if the thickness of the layers is the same?

9.3 Diffusion in a soil column

Another important transport mechanism in soil is diffusion. Transport by diffusion occurs if differences exist in the concentrations of dissolved substances in a medium. The familiar Fick's 2nd Law is obtained by setting mass flow in Equation 52 to zero, \( v=0 \), and by omitting the sink term, \( R \):

\[
\frac{\partial C}{\partial t} = D \cdot \frac{\partial^2 C}{\partial x^2}
\]

Equation 63

Crank (1975) gives a large number of analytical solutions to Equation 63 for many different boundary conditions. In this text, however, we will solve the differential equations numerically.
9.3.1 Derivation of the differential equations for diffusive flow

The diffusion flux, $q_d$ (kg m$^{-2}$ s$^{-1}$), of a substance is proportional to the product of its concentration gradient, $dC/dx$ (kg m$^{-4}$), and its diffusion coefficient, $D$ (m$^2$ s$^{-1}$):

$$q_d = -D \cdot \frac{dC}{dx} = -D \cdot \frac{\Delta C}{\Delta x}$$

Equation 64

For an explanation of the minus sign in Equation 64, Figure 36 is used.

The concentration gradient is $(C(I) - C(I-1))/(x_2 - x_1)$, with the $x$-values taken as the distance between a certain reference point and the centre of the layer. The value of this quotient is positive since $C(I)$ is higher than $C(I-1)$. However, the flow of matter is from a high to a low concentration: from the right to the left in Figure 36 (the arrow indicates the positive direction; therefore the flow $q_d$ will be negative here). Thus, the direction of the diffusive flow is opposite to an increasing concentration and the minus sign in Equation 64 takes care of that phenomenon.

The relational diagram in Figure 37 facilitates the derivation of the differential equations for diffusive transport.

Using Equation 64 and Figures 36 and 37, the flux equations for layer I can be written directly:

$$FL(I) = -D \cdot (C(I) - C(I-1)) / TCOM$$
$$FL(I+1) = -D \cdot (C(I+1) - C(I)) / TCOM.$$  

Equations 65

in which it is assumed that within each layer the salt is distributed uniformly and thus that the concentration is $C(I)$ throughout, and that $(x_2 - x_1)$ is equal to the distance between the centres of adjacent layers (TCOM).
The net flux equation for layer I is then:

\[ NFL(I) = FL(I) - FL(I+1) \]  
\[ \text{Equation 66} \]
or

\[ NFL(I) = -D \times \left( -C(I-1) + 2 \times C(I) - C(I+1) \right) / TCOM \]  
\[ \text{Equation 67} \]

with units of kg m\(^{-2}\) s\(^{-1}\). In Equation 63, \(dC/dt\) has units of kg m\(^{-3}\) s\(^{-1}\), however. Equation 67 is related to Equation 63 in the same way that Equation 56 was related to Equation 53 in the description of mass flow. Here, this relation is:

\[ \frac{dC(I)}{dt} = \frac{NFL(I)}{TCOM} = -\frac{D}{TCOM^{2}} \times \left( -C(I-1) + 2 \times C(I) - C(I+1) \right) . \]  
\[ \text{Equation 68} \]

The minus sign in Equation 68 is often eliminated with the following result:

\[ \frac{dC(I)}{dt} = \frac{NFL(I)}{TCOM} = \frac{D}{TCOM^{2}} \times \left( C(I-1) - 2 \times C(I) + C(I+1) \right) . \]  
\[ \text{Equation 69} \]

The time coefficient for diffusive flow in Equation 68 (or 69) is \(TC_{\text{diffusion}} = TCOM^{2}/D\). Hence, the time coefficient is proportional to the square of the thickness of the layer and inversely proportional to the diffusion coefficient.

Equation 68 (or 69) could directly be used for the simulation of diffusive transport in soil, but usually Equations 65 and 66 are used for this purpose.

**Exercise 78**

a. Write a simulation model for diffusive transport in a soil column of similar geometrical properties as the one in Figure 35, according to Equations 65 and 66. Take the boundary condition at the soil surface similar to the one in Figure 35, but assume that the lower boundary of the soil column is impervious. Name the program: 'Elementary description of diffusive flow', and take the diffusion coefficient, \(D\), equal to \(4.0E-5\) m\(^2\) d\(^{-1}\).

b. How did you formulate the differential equation for the first soil layer?

**Exercise 79**

Adapt the program from Exercise 78 so that the fertilizer is applied to the profile at a depth between 0.15 and 0.25 m. Distribute the fertilizer evenly over the corresponding soil layers and calculate the amount from an application of 200 kg ha\(^{-1}\).

a. How do you formulate FLUX(1)?

b. Run the program and plot the concentration of the fertilizer in layers 1 to 10 against depth for \(t=0\), \(t=40\), and \(t=100\) days.

c. Why is the concentration profile of the fertilizer asymmetric?

d. Calculate the amount of fertilizer in the soil at \(t=0\) and \(t=100\) days. Does the balance value correspond to the initial amount of fertilizer?
9.3.2 Transport of heat in soil

Similar to diffusion, transport of heat by conduction is caused by random molecular movement. The diffusion equation can be written as:

\[
\frac{\partial T}{\partial t} = D \frac{\partial^2 T}{\partial x^2} = \frac{\lambda}{C_h} \frac{\partial^2 T}{\partial x^2}
\]

Equation 70

where, \(\lambda\) is the heat conductivity in J/(m s °C), \(C_h\) is the heat capacity in J/(m\(^3\) °C), \(T\) is the temperature in °C as a function of depth and time, \(x\) the spatial coordinate in m, and \(t\) the time in s.

9.3.3 Derivation of the differential equations for heat flow

The flux equation for heat flow can be set up according to Ohm's Law:

\[
\mathcal{V} = I \cdot R,
\]

where \(I\) = \(\mathcal{V}/R\), i.e. current = potential difference times the inverse of resistance. The resistance \(R\) of a medium will be proportional to its length, \(l\) or \(TCOM\), and inversely proportional to the conductivity \(\lambda\) or \(COND\), i.e. \(R = TCOM/COND\). The heat-flux equation, analogous to Ohm's Law, is then:

\[
\mathcal{F}(I) = -\Delta T \cdot \frac{COND}{TCOM}
\]

Equation 71

expressed in J m\(^{-2}\) s\(^{-1}\). The minus sign has the same background as that described for diffusion in Subsection 9.3.1. The temperature difference, \(\Delta T\), over a certain distance, \(TCOM\), is indicated by \(TEMP(I) - TEMP(I-1)\). The complete flux equation for layer I is then:

\[
\mathcal{F}(I) = - ( TEMP(I) - TEMP(I-1) ) \cdot \frac{COND}{TCOM}
\]

Equation 72

The volumetric heat content, \(VHTC\) (Volumetric Heat Content, expressed in Joules), of a layer is obtained by adding the product of the flow, the surface area and the time to the heat content already present. The temperature of the layer can be calculated from:

\[
VHTC = VHCAP \cdot VOL \cdot TEMP.
\]

\[
J = \frac{\mathcal{F}}{m^3 \ °C}
\]

Equation 73

In Figure 38 a program called 'Flow of heat in a homogeneous soil column placed on an insulating layer' is presented, using Equations 72 and 73. The soil surface temperature in the model is exogenously described (forcing function) by a sinusoidal temperature wave over the day.

Exercise 80

a. Study the program in Figure 38 by making a dimensional analysis.
b. Which assumptions underly the model?
c. What is the time coefficient of the model?
d. Which numerical value would you calculate for the time interval if you use the rectangular integration method?
Figure 38. Listing of a program entitled 'Flow of heat in a homogeneous soil column placed on an insulating layer', using Equations 72 and 73.

TITLE Flow of heat in a homogeneous soil column...
    placed on an insulating layer
INITIAL
NOSORT
FIXED I,N
STORAGE FLUX(26), TEMP(25)
PARAM TCOM=0.02, AREA=1.0, COND =0.42, VHCAP=1.05E6
    * m     m**2        J        J
    *              --------------------------
    * m degree-C s m**3 degree-C
PARAM ITMP=20.0, TAV =20.0, TAMPL=10.0, N=25
    degree-C
PARAM TIMER FINTIM=172800.0, OUTDEL=3600.0, PRDEL=3600.0
    s
PAGE WIDTH =80, NTAB =0, GROUP
OUTPUT TEMP(1), TEMP(5), TEMP(10), TEMP(15), TEMP(25)
PRINT TEMP(1), TEMP(5), TEMP(10), TEMP(15), TEMP(25)
METHOD RKS

* calculations
VOL =TCOM*AREA
PI =4.0*ATAN(1.0)
    DO 10 I=1,N
         IVHTC(I)=ITMP*VOL*VHCAP
    10 CONTINUE

DYNAMIC
NOSORT
VHTC =INTGRL(IVHTC, NFLOW, 25)

TMPS = TAV+TAMPL*SIN(2.0*PI*TIME/86400.0)
    DO 20 I=1,N
         TEMP(I) = VHTC(I)/(VOL*VHCAP)
    20 CONTINUE

FLUX(1) =-COND * (TEMP(1)-TMPS)/(0.5*TCOM)
    DO 30 I=2,N
         FLUX(I) =-COND * (TEMP(I)-TEMP(I-1))/TCOM
    30 CONTINUE
FLUX(N+1) = 0.0
    DO 40 I=1,N
         NFLOW(I) = FLUX(I)*AREA - FLUX(I+1)*AREA
    40 CONTINUE

END
STOP
ENDJOB

In the listing of Figure 38 the volumetric heat content, VHTC, is calculated with respect to the, arbitrary, initial temperature (DO-loop 10). Therefore, it is possible that VHTC becomes negative during the simulation. This is, however, not problematic because we are interested in the change in VHTC, which can be calculated with respect to any reference temperature. From the changed heat content, the temperature may be found (DO-loop 20). Though some might like to choose the reference temperature at zero K, this would suggest that the volumetric heat capacity,
VHCAP, would not be a function of temperature. Since this is not the case, one can better take a reference temperature that is close to the average temperature of the system considered.

9.3.4 An analytical solution for heat flow

Equation 70 can also be solved analytically if the thermal properties of the medium are constant. The solution presented here (without derivation) is valid for the following conditions:
- there is a sinusoidal temperature wave at the soil surface;
- that situation exists for quite a while, which means that initial effects can be neglected;
- the heat-conducting medium is homogeneous;
- the temperature is constant at a certain depth, i.e. the soil column is semi-infinite.

The solution to Equation 70 for a soil with a constant average temperature, $T_{av}$ is:

$$T_{x,t} = T_{av} + T_{ampl} \cdot \exp(-x/d) \cdot \sin(\omega \cdot t - x/d)$$

Equation 74

where $T_{x,t}$ ($T$ in Equation 70) is the temperature at depth $x$ and time $t$, $T_{ampl}$ the amplitude of the sinusoidal temperature wave at the soil surface, $\omega$ is the angular frequency of the temperature wave, $t$ is the time and $d$ the attenuation depth. The angular frequency is expressed as:

$$\omega = \frac{2\pi}{t_c}$$

Equation 75

where $t_c$ is the time required for the wave to make a complete cycle. Figure 39 illustrates the different terms of Equation 74.

Figure 39. Cyclic variation of the temperature at the soil surface and at depth $x$ (From Koorevaar, Menelik and Dirksen, 1983)

The sinusoidal temperature wave at the soil surface permeates into the soil via heat conduction. As the rate of heat adjustment in the soil is finite, the amplitude of the heat wave decreases with increasing depth (Figure 39). This is
expressed in Equation 74 by the so-called attenuation depth, \(d\), which is the depth at which the amplitude of the temperature wave has decreased to \(T_{\text{amp}/e} = 0.37 \times T_{\text{amp}}\).

---

**Exercise 81**
Check whether the amplitude of the temperature at \(x=d\) is equal to \(T_{\text{amp}/e}\).

---

The attenuation depth is dependent on the thermal properties of the soil and on the angular frequency of the temperature wave according to:

\[
d = \left( \frac{2 \times \frac{\lambda}{\omega \times C_h}}{\omega \times C_h} \right)^{1/2} = \left( \frac{2 \times D}{\omega \times C_h} \right)^{1/2}.
\]

Equation 76

The finite rate of heat adjustment of the soil also causes a phase shift between the heat wave at the soil surface and the wave at a certain depth \(x\) (Figure 39). This is expressed in Equation 74 by the variable \(-x/d\) in the argument of the sine function.

---

**Exercise 82**
Calculate \(T_{x,t}\) with the aid of Equations 74, 75 and 76 and the parameter values from the program in Figure 38, for the 5th soil layer (thickness of each layer is 0.02 m) at 19, 19.25, 19.50, 19.75 and 20 days. Compare your results with the simulated values below:

<table>
<thead>
<tr>
<th>time, d</th>
<th>19</th>
<th>19.25</th>
<th>19.50</th>
<th>19.75</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulated value, °C</td>
<td>16.782</td>
<td>22.743</td>
<td>23.219</td>
<td>17.257</td>
<td>16.782</td>
</tr>
<tr>
<td>analytical solution</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

(via Equation 74)

The analytical solution can be put to good use in testing the numerical program. Subsequently, the numerical program can be adjusted to analyse more realistic situations. An initial significant adjustment could be that of the forcing function of the temperature cycle over the day: the sinusoidal temperature wave is then replaced by a measured temperature series (see for instance the functions MNTT and MXTT in Figure 31, Section 8.8). A second important adjustment would be the introduction of the thermal properties of the soil as a function of depth. However, this means that the calculation of, for example, the average heat conductivity, \(\lambda\) or \(\text{COND}\), between two adjacent layers must be adjusted. In Appendix 5 an expression is derived to calculate the average conductivity.

---

**Exercise 83**
Derive from the final equation in Appendix 5, what value \(AVCOND(I)\) takes, if the thickness of all layers is the same and if the conductivities are identical for each layer?

---

**Exercise 84**
A consultant is to give horticulturists advice on measures they might take to limit frost damage to seeds present in the soil by means of mulching it with peat. To this purpose he wishes to develop a simulation model, enabling him
to introduce different heat capacities and heat conductivities for the peat mulch and for the soil, and different layer thicknesses. The \( \lambda \) and \( C_h \) for peat are \( 0.25 \ J / (m \ s \ °C) \) and \( 2.5 \times 10^6 \ J / (m^3 \ °C) \), respectively. The average temperature and its amplitude equal 1.5 and 6.5 \( °C \), respectively. The initial temperature profile may be taken equal to the average temperature. Other parameter values can be taken from the listing in Figure 38.

a. Use the listing in Figure 38 as a starting point to develop the desired program.

Some hints may be needed. Naturally, the mulch is applied on top of the soil, and, since the total soil depth should remain 0.5 metre (Figure 38), the number of layers in the model should be increased to include the mulch. Furthermore, if different compartment thicknesses are allowed, the distance between the centres of the compartments is not equal to \( TCOM \) anymore, but should be calculated as the arithmetic average of adjacent compartments. The conductivities and the heat capacities of the different materials can best be read from FUNCTIONs by means of AFGEN.

b. Run the program with a peat layer of 4 cm. At which depth the temperatures remain above the freezing point? And at which depth the temperatures remain above the freezing point when no peat mulch would be present?

### 9.4 Derivation of the general transport equation

In Equation 52 it was assumed that the flow rate, \( v \), and the diffusion-dispersion coefficient, \( D \), are not dependent on location. Therefore, \( v \) and \( D \) are placed in front of the \( \partial / \partial x \):

\[
\frac{\partial C}{\partial t} = - v \frac{\partial C}{\partial x} + D \frac{\partial^2 C}{\partial x^2} - R. \tag{Equation 52}
\]

However, these coefficients usually vary with location. Equation 52 is derived below to show how this partial differential equation would look like if the dependence of \( v \) and \( D \) on location would be included. The sink term, \( R \), is again set to zero for the sake of simplicity.

Equation 52 is obtained by combining three equations:

1. the flux equation for mass flow;
2. the flux equation for diffusive flow;
3. the continuity equation.

The flux equations for mass flow and diffusive flow are \( q_m = v \cdot C \) (Equation 54) and \( q_d = - D \cdot \frac{dC}{dx} \) (Equation 64), respectively.

The fluxes, \( q \), should now be related to the temporal change in concentration per unit volume. To this purpose the continuity equation is needed. Its derivation is facilitated by Figure 40.

![Figure 40. Volume element to derive the continuity equation.](image)
The amount of substance that flows into the volume element \(a \times b \times \Delta x\) (m\(^3\)) in a certain time interval, \(\Delta t\), is:

\[
q \times a \times b \times \Delta t \quad \text{(kg)}
\]

The amount of substance that flows out of this volume element in a certain time interval, \(\Delta t\), is:

\[
(q + \Delta q) \times a \times b \times \Delta t \quad \text{(kg)}
\]

The accumulation of substance in the volume element, or its net change equals:

\[
\Delta Z = - \Delta q \times a \times b \times \Delta t \quad \text{(kg)}
\]

The change in concentration is obtained by dividing \(\Delta Z\) by the volume \(a \times b \times \Delta x\):

\[
\Delta C = \frac{- \Delta Z}{a \times b \times \Delta x} = \frac{- \Delta q \times a \times b \times \Delta t}{a \times b \times \Delta x} = \frac{- \Delta q \times \Delta t}{\Delta x}
\]

or

\[
\frac{\Delta C}{\Delta t} = - \frac{\Delta q}{\Delta x}
\]

And if \(\Delta\) approaches zero:

\[
\frac{\partial C}{\partial t} = - \frac{\partial q}{\partial x}
\]

Equation 77

Combining Equation 77 with the flux equations for mass flow, Equation 54, and diffusive flow, Equation 64, results in:

\[
\frac{\partial C}{\partial t} = - \frac{\partial}{\partial x} (v \times C - D \times \frac{\partial C}{\partial x})
\]

Equation 78

If \(v\) and \(D\) would be invariant with depth, they could be taken outside the brackets, thus producing Equation 52.

Note that in a soil, the rate of water flow will change with depth, since the moisture content is seldom constant with depth and because the soil has a higher conductivity in moist layers than in dry layers. The rate of diffusion of nutrients in the water phase of soil may also be higher at higher moisture contents, since a greater surface area of water is available and because the water phase will be more continuous. On the other hand, gas diffusion proceeds much easier in a dry soil, since there is a larger gas-filled surface area and because the diffusion coefficients in the gas phase are approximately \(10^4\) times higher than in the water phase. This means that in realistic models, the flow rate of water (\(v\)) and the diffusion coefficient (\(D\)) will vary with depth.

Exercise 85

A simulation model should be set up to investigate the behaviour of nutrients in a rooted rice or wetland soil. The transport processes that take place are mass flow, at a rate \(v = 0.001\) m\(^3\) m\(^{-2}\)_d\(^{-1}\) (or m \ d\(^{-1}\)), and diffusion, with a diffusion-dispersion coefficient \(D = 4.0 \times 10^{-5}\) m\(^2\) d\(^{-1}\). Both the transport coefficients apply throughout the profile. The water above the soil has a fertilizer concentration of 10 kg m\(^{-3}\), but the soil is initially free of fertilizer. The roots in the profile can take up nutrients between 5 and 30 cm depth. The uptake process is considered to be proportional with the concentration of nutrient in the soil layers. The relative uptake rates (RUR) for the rooted layers are: from 5 to 10 cm: 0.1 d\(^{-1}\); from 10 to 15 cm: 0.2 d\(^{-1}\); from 15 to 20 cm: 0.3 d\(^{-1}\); from 20 to 25 cm: 0.2 d\(^{-1}\); and from 25 to 30 cm: 0.1 d\(^{-1}\). The profile considered has a depth of one metre. The surface area is taken as 1 m\(^2\).
Numerical dispersion in the description of mass flow is to be suppressed by using Equation 61. At the bottom of the profile mass flow is described by Equation 54, whereas the diffusive flow is assumed to be zero there.

a. What does it mean with regard to the rice or wetland soil, when the transport coefficients can be taken constant throughout the profile?

b. Why is it necessary to consider a saturated soil in this model?

c. What layer thickness do you chose, and what is the number of soil layers then?

d. Under which circumstances is it allowed to assume that the diffusive flow at the bottom of the profile is equal to zero?

e. Develop the required program. Use the SI units: d, m, and kg. Include a calculation of the total (accumulated) amount of fertilizer that is taken up by the roots. What total simulation time do you choose (FINTIM)?

f. Run the program and inspect the results. Till when is the simulation warranted?

g. Now take \( v = 0.005 \text{ m d}^{-1} \) and run the program anew. Inspect the results. What do you notice with regard to the concentration in the top layer of the profile? Explain what is wrong in the parameterization of the simulation program.

9.5 Concluding remarks

In Chapter 8 the numerical dispersion, as introduced by the simulation technique through dividing the time or the space into slices, was put to good use in describing breakthrough curves of, for instance, the germination of seeds. In this Chapter it was discussed how to suppress numerical dispersion in the description of mass flow, because we desire to explain the evening out of peaks and troughs in a given concentration profile as the result of real hydrodynamic dispersion (different flow velocities of water in pores of different sizes) and of real diffusive transport. Then, no artefact of the simulation technique can be accepted. Though numerical dispersion will add to the physical dispersion that is explicitly described in a model, it is now possible to calculate its contribution to the total transport. If this contribution would be too large, say more than 10 percent, one could diminishing it by taking a smaller compartment thickness and/or by reformulating the mass flow equation.

The soil example presented was employed in its simplest form, so that attention would not diverge from the theoretical analysis. Nevertheless, the treatment of the soil example provides a basis to further incorporate detail about the transport, uptake and transformation of nutrients. Then, especially more physiology (root uptake equations and microbiological conversions), chemistry (equilibria that determine the distribution of nutrients or any substance between the solid, the water and the gaseous phase in soil), and physical (porous medium characteristics like porosity and water retention and conductivity) information should be included. These subjects are partly covered by de Wit and van Keulen, 1975.
10 Simulation using FORTRAN

C. Rappoldt, P.A. Leffelaar and D.W.G. van Kraalingen

10.1 Introduction

The examples in the previous chapters illustrate that the use of the CSMP simulation language enormously facilitates model development. However, the portability of CSMP user programs is limited, because CSMP is not always available and it can only run on a limited number of computer systems (main frames: VAX, IBM, or PC's: IBM or compatibles). This is the moment to devote attention to the portability of computer models.

Portability or machine independency of computer models is best assured by the use of a standardized computing language for which compilers are available for most types of computers. Therefore, the procedural language FORTRAN was selected. Furthermore, FORTRAN enables to write well structured programs, as illustrated in Section 7.3 and by e.g. Wagener (1980), and an overwhelming number of tested subroutines designed to support scientific computations is available in the literature, e.g. the IMSL library (IMSL, 1987) and the well-documented routines from 'Numerical Recipes' (Press et al., 1986).

When a general purpose language like FORTRAN is used for simulation, the elements that are normally taken care of by the simulation language, i.e. the data input, time loop, sorting of the equations, integration, interpolation and data output, should be programmed by the modeller in addition to developing the differential equations of the model. Since these elements are similar for each new program, it is desirable to separate technical programming details from the actual user model, so that a simulation environment is created in FORTRAN which partially resembles that of CSMP.

In this chapter descriptions are given of how these different elements can be programmed in the FORTRAN-77 standard. We begin with the correct calculational sequence of states and rates using the integration method of Euler without unraveling the technical programming details from the user model equations. Then, it will become obvious why separation of technical programming details from the actual user model is worthwhile, and how this can be done. It will also become clear that a number of standardized input and output formats return each time in any program. To avoid the programming of these functions over and over again, the subroutine library TTUTIL (Rappoldt & van Kraalingen, 1990) is presented. Similarly to the CSMP functions in Tables 1 and 2 in Section 3.6, there is no need to fully understand how these routines work: what is necessary is to know how they can be used. Subsequently, the rerun facility is introduced in the simulation environment of FORTRAN. At this point the shift from Euler's method to the Runge-Kutta method with a self adapting time step to integrate rates could be made. How this can be done by using some of the algorithms of Press et al. (1986) is described. Finally, a detailed description of the model part that should be made by the user is provided by means of a figure analogous to Figure 7 in Section 3.2.

After studying this chapter, it should be easy to further develop simulation programs in FORTRAN.

10.2 The time loop, the correct calculation sequence and Euler's integration

Figure 41 gives the time loop and the calculation sequence for simulation in FORTRAN. An Initial, Dynamic and Terminal segment is distinguished: in the Initial segment the states are calculated or given to start the simulation; the Dynamic segment should be executed until a finish condition is met, usually finish time (FINTIM); in the Terminal segment final calculations based on the simulation could be performed, or reruns may be initiated. The dynamic loop is directly derived from the basic assumption in systems analysis that systems are state determined and that rates of change can be calculated from the values of the state variables and the environmental conditions or
driving variables. Therefore, the update of all state variables, i.e. 'integration', appears on top of the dynamic segment, similar to modelling in CSMP. In designing the dynamic loop it should be kept in mind that only after calculating the rates the values of state and rate variables are synchronously known and only then consistent output is possible (see Section 2.5). Since also at the termination of the model output of states and concomitant rates is desired, at FINTIM rates of change have to be calculated that are not integrated anymore. Hence, N integration steps require N+1 rate calculations. Basically, there are two ways to do this. Either at the start or at the end of the simulation one integration is skipped. The former option is illustrated in Figure 41 and was chosen by Van Kraalingen (1991). Here, the latter option is chosen, however, which implies that the initial states are directly used to calculate the rates of change at the start of the simulation. This is illustrated in Figure 42 where all terms in the dynamic loop are shifted anti-clockwise over one position with respect to Figure 41. This construction will also be used in the integration according to Runge-Kutta in Section 10.3).

Figure 43 presents the computer code for the example of exponential growth (compare answer Exercise 21, Chapter 12) based on Figure 42. The label 'PROGRAM' is used to identify the program by one to six alphanumeric characters, the first of which must be a letter: this program is simply named DEMO1. All FORTRAN programs end with one and only one END statement. The line IMPLICIT REAL(A-Z) declares all variables real, similar to CSMP. On a PARAMETER (...=...) statement variables can be given values that should retain their value during program execution. The time loop is controlled by an IF-THEN-ENDIF structure (Subsection 7.2.2. Note that ENDIF may also be written as END IF). As long as TIME has not surpassed FINTIM, the loop is executed because the GOTO 10 instruction just before ENDIF returns control to the IF-THEN line that is identified by the label 10. The output of calculated states and rates is obtained by:

```
WRITE (60, ' (3G13.5) ') TIME, A, GR
```

This line writes the calculated data of the values of time (TIME), state (A) and rate (GR) variables to the file with unit number 60 with a format of G13.5. In view of the existence of several reserved unit numbers in the final FORTRAN simulation program presented in this chapter, it is proposed to select unit numbers of 60 and higher. The format of the numbers, specified with 3G13.5, means that 3 numbers will follow, each consisting of 13 positions of which 5 will be printed after the
PROGRAM DEM01
IMPLICIT REAL(A-Z)

*** INITIAL
* parameterization of model
PARAMETER ( RGR= 0.1, DELT= 1.0, FINTIM= 10.0 )
* initialization of model
A = 1.0
* initialization of time
TIME = 0.0
* open file for output and write heading
OPEN (60, FILE='RES.DAT', STATUS='NEW')
WRITE (60, '(9A, 2A13/)') 'TIME', 'A', 'GR'

*** DYNAMIC
10 IF (TIME .LE. FINTIM) THEN
   * driving variables (absent here)
   * calculate rates of change
     GR = RGR * A
   * write output
     WRITE (60, '3G13.5') TIME, A, GR
   * integration:
     TIME = TIME + DELT
     A = A + DELT * GR
   GOTO 10
END IF

*** TERMINAL
* terminal calculations could be performed here
STOP
END

Figure 43. FORTRAN simulation program in its simplest form.
decimal point. The G-format indicates that a real number may be either represented with a decimal point without a power expression (for example 1.00 or 201.34), or with a power expression (for example, 1.00E+00 or 2013.4E-01, where E-01 means $10^{-1}$), depending on the magnitude of the number. The file on unit number 60 must be opened once. This should be done in the initial segment of the program by the line:

```fortran
OPEN (60, FILE='RES.DAT', STATUS='NEW')
```

The file will be given the name RES.DAT, and it is newly created. This implies that the computer memory may not contain a file with the same name. After opening the file a table caption is written to identify the calculated results from the dynamic:

```fortran
WRITE (60, '(A9,2A13/)', 'TIME', 'A', 'GR')
```

Here, the caption names should be given between quotes because these are character type variables, for which the A-format is reserved. The '/' in the format specifies that a blank line must appear between the heading and the numerical values in the output table. Files are automatically closed when the label STOP is reached.

The program in Figure 43 generates output after each integration step. Usually, however, output is desired at intervals larger than the integration step. Therefore, the user should have the possibility to specify output-intervals, similar to PRDEL in CSMP. In the FORTRAN simulation program in Figure 44 output is controlled by the logical flag OUTPUT. The flag should be set commensurate with the value of PRDEL. OUTPUT may be either .TRUE. or .FALSE. and it is calculated in line 26. If OUTPUT is false, the lines 28-32 are executed: output is generated, the next moment that output should occur (TNEXT) is calculated (lines 30 and 31) and the logical variable HALT is set. This variable is used to decide if it is time to leave the dynamic loop (line 21). The use of HALT instead of TIME, as was done in the program of Figure 43, is more general because the program might be stopped by other finish conditions than time alone. The variable TNEXT is calculated from FINTIM and the product of the number of intervals, IP, and PRDEL. The number of intervals is calculated in the initial (line 16). This way of calculating TNEXT is to be preferred above adding each time the value of PRDEL to the previous one, because round off errors may accumulate.

The time loop (lines 21-40) should not only guarantee the generation of final output when a PRDEL is reached, but also when the simulation is interrupted by the model by setting its terminal flag TERMNL. Finish conditions other than time might be induced in the rate section (lines 22-24) or in the state section (lines 36-37) of the program by setting the flag TERMNL true. At the start of the simulation the variables HALT and TERMNL should be initialized as .FALSE.

In the program in Figure 43 the last integration was not skipped, as suggested in Figure 42. In the listing in Figure 44 this is introduced, however, by means of lines 35 and 38.

The requirements that output of results should occur when PRDEL is reached or when the model is terminated by another finish condition induced in either the rate or state section of the model, and that the last integration should be skipped, are all met by the time loop in Figure 44.

Although the listing in Figure 44 may already be of practical use, it has a number of principal drawbacks. First, the user should not be concerned with the time management, the control of output flag, and the actual integration process in simulation models. Second, the model statements in Figure 44 (lines 7-10 and 24) are currently entangled with the technical aspects of solving the rate equation with the result that the program is less legible. It would be best when all user statements were programmed in one highly legible module or subroutine summarizing the equations that describe the system under study. Third, all parameters and initial variables are given values within the program code. This implies that for each change of parameters the program needs to be compiled again. This can easily be avoided when data would be read from external data files by the compiled program. And fourth, technical statements like format specifications for output of results and the opening of files are often needed and cumbersome to program. Therefore, one would like to replace these by simple functions, often called utility functions. These requirements are met in
*** INITIAL
* Initialization and parameterization of model
PARAMETER (RGR = 0.1, STTIME= 0.0, DELT= 1.0, 
$ S $ PRDEL= 1.0, FINTIM= 10.0)
* initialize model
A    = 1.0
* initialize timing
TERMNL =.FALSE.
TIME  = STTIME
TNEXT = STTIME
HALT  = TIME.GE.FINTIM
IP     = 1 + INT ((FINTIM- STTIME) / PRDEL - 0.01)

* Open file for output and write heading
OPEN (60, FILE='RES.DAT', STATUS='NEW')
WRITE (60, I (A9,2A13/) ) 'TIME' I 'A' I 'GR'

*** DYNAMIC
10 IF (.NOT.HALT) THEN
  * driving variables (absent here)
  * calculate rates of change
  GR    = RGR * A
  * output required ?
  OUTPUT = (TNEXT-TIME)/PRDEL.LT.0.0001 .OR. TERMNL
  IF (OUTPUT) THEN
    WRITE (60, '(3G13.5)') TIME,A,GR
  * get next output time ; leave dynamic loop ?
    IF = IP - 1
    TNEXT = FINTIM - IP * PRDEL
    HALT  = TNEXT.GT.FINTIM .OR. TERMNL
  END IF
  * integration:
    IF (.NOT.HALT .AND. .NOT.TERMNL) THEN
      TIME  = TIME + DELT
      A     = A + DELT * GR
    END IF
GOTO 10
END IF

*** TERMINAL
* terminal calculations could be performed here
STOP
END

Figure 44. FORTRAN simulation program that fulfills the requirement that output of results should occur when PRDEL is reached or when the model is terminated by another finish condition.

the program setup shown in Figure 45. The user subroutine MODEL, further called MODEL, contains all the equations for the initial, dynamic and terminal calculations. Furthermore, the data needed in MODEL, like initial conditions and parameters, are read from the file MODEL.DAT, and the calculated results are written in the file RES.DAT. The timer variables are given in a separate file TIMER.DAT, so that the model's time management including the values of these variables is fully located in the program EUDRIV. The abbreviation EUDRIV stands for Euler driver, because
Figure 45. Lay out of FORTRAN simulation modules using the integration method of Euler, showing a main program EUDRIV, the user model subroutine MODEL and their communication, input files (TIMER.DAT and MODEL.DAT) and output files (MODEL.LOG and RES.DAT).

It controls the sequence of calls to the model that is integrated by Euler's method. In fact, the user has only to develop MODEL, and the two data files. There is no need to reprogram the main program EUDRIV. The main program also opens the file MODEL.LOG. This file is used to write messages on program performance during execution.

The program code in Figure 46 is developed according to the lay out in Figure 45, but it only shows the main program. Below we will in turn consider the dynamic, the initial and the terminal part of the program, and the communication between the main program and MODEL. Thereafter, MODEL will be discussed.

The dynamic part (lines 82-117) The major features of the time loop were already discussed in lines 21-40 of Figure 44. The call to MODEL in line 90 replaces lines 22-24 in Figure 44. The driving variables and the rates must thus be evaluated there. In MODEL different state and rate variables may be present, all having their own unique name invented by the modeller. Integration of these rates and the updating of the states is a general computational problem, however, and the user should not be occupied with that problem. Consequently, a general array of states (STATE) is updated in the main program by integrating a general array of rates (RATE), e.g. lines 111-113. These lines thus replace line 37 in Figure 44. These general arrays are communicated to MODEL by the list of arguments. Within MODEL it will now be necessary to copy these general arrays in user specific names, as will be discussed below.

Inspection of the lines 83-93 in comparison with lines 22-29 from Figure 44 shows that the writing of calculated results, which was done in Figure 44 in line 28, has been removed from the main program. Since writing of calculated results is a user specific activity, it will be done in the user's MODEL. However, MODEL then should have information on the necessity of writing results. As a consequence, the OUTPUT flag should be calculated just before calling MODEL, line 84.

Just before the call to MODEL (line 90) the integer variable KEEP, which has the same significance as in CSMP, is equated to 1 since a final integration has just taken place (Subsection 7.2.3). Only at this point in the time loop, the states could be adjusted directly by the 'replace method', if a discontinuity would occur (Section 6.3; Subsection 7.2.4). The adjustment should be done directly at the beginning of MODEL. After returning to the main program loop, KEEP is equated to 0.
PROGRAM EUDRIV

* formal parameters
INTEGER IUL, IUD, IUM

* common /INFO/
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT, PRDEL, DELMAX, FINTIM,
COMMON /INFO/ IULOG, IUMOD, KEEP, TERMNL

** local (non-common) variables
INTEGER I, IP, NEQ, NDEC
REAL STTIME, TIME, TNEXT, DEL, DELT0
REAL DELHLP, MULTI1, MULTI2
REAL STATE, RATE, SCALE
PARAMETER (NDEC=250)
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL HALT, OUTPUT

* unit numbers used for logfile, driver (main program) and model
DATA IUL/20/, IUD/40/, IUM/50/
* get copy of logfile unit number and model unit
* number into /INFO/
IULOG = IUL
IUMOD = IUM
* open logfile
CALL FOPENG (IULOG, 'MODEL. LOG', 'NEW', 'SF', 0, 'DEL')

IF (IULOG.GT.0)
$ WRITE (IULOG, '(A)') ' Initialize model'
WRITE (*, '(A)') ' Initialize model'

* read timer variables
CALL RDINIT (IUD, IULOG, 'TIMER.DAT')
CALL RDSREA ('STTIME', STTIME)
CALL RDSREA ('FINTIM', FINTIM)
CALL RDSREA ('PRDEL', PRDEL)
CALL RDSREA ('DELMAX', DELMAX)
CLOSE (IUD, STATUS='DELETE')
IF (PRDEL.LE.0.0)
$ CALL ERROR ('EUDRIV', 'Illegal value PRDEL')
IF (DELT.GT.PRDEL)
$ CALL ERROR ('EUDRIV', 'Illegal value DELT')
IF (DELMAX.GT.PRDEL)
$ CALL ERROR ('EUDRIV', 'Illegal value DELMAX')

* investigate if DELT is a multiple of PRDEL
MULTI1 = PRDEL/DELT
MULTI2 = FLOAT( INT(PRDEL/DELT) )
IF( MULTI1 .NE. MULTI2 ) THEN
DELT = MIN (DELT, DELMAX)
IF (DELTHLP .LT. DELT ) THEN
DELT = DELTHLP
ELSE
DELT = DELTHLP/2.
GOTO 5
ENDIF
ENDIF

* initialize timing
TERMNL = .FALSE.
TIME = STTIME
TNEXT = STTIME
HALT = TIME.GE.FINTIM

135
* to simplify debugging: set states and rates to 0
DO 10 I=1,NDEC
STATE(I) = 0.0
RATE(I) = 0.0
10 CONTINUE
* initialize model
CALL MODEL (1,.FALSE.,TIME,STATE,RATE,SCALE,NDEC,NEQ)
* error checks
IF (NEQ.LE.0) CALL ERROR ('EUDRIV',
$ 'No value of NEQ was specified in MODEL')
IF (NEQ.GT.NDEC) CALL ERROR ('EUDRIV',
$ 'Too many state variables')
* dynamic loop
IF (IULOG.GT.0)
WRITE (IULOG,'(A)') ' Dynamic loop'
WRITE (*,'(A)') ' Dynamic loop'
20 IF (.NOT.HALT) THEN
* output required ?
OUTPUT = (TNEXT-TIME)/PRDEL.LT.0.0001 .OR. TERMNL
* get rates of change at beginning of time step
* (write output)
* this is the rate call at the start of a new step
* (KEEP=1)
KEEP = 1
CALL MODEL (2,OUTPUT,TIME,STATE,RATE,SCALE,NDEC,NEQ)
KEEP = 0
IF (OUTPUT) THEN
* get next output time ; leave dynamic loop ?
IP = IP - 1
 TNEXT = FINTIM - IP * PRDEL
HALT = TNEXT.GT.FINTIM .OR. TERMNL
END IF
* time step limitation and integration
IF (.NOT.HALT .AND. .NOT.TERMNL) THEN
* limit timestep
DELT = MIN (DELT0,DELMAX)
* IF (TIME+DELT.LT. TNEXT) THEN
* accept advised step
DELT = DELT
ELSE
* reduce time step
DELT = TNEXT-TIME
END IF
* integration
TIME = TIME + DEL
DO 30 I=1,NEQ
STATE(I) = STATE(I) + DEL * RATE(I)
30 CONTINUE
DELT = DELT0
END IF
GOTO 20
END IF
* terminate model
IF (IULOG.GT.0)
WRITE (IULOG,'(A)') ' Terminate model'
CALL MODEL (4,.FALSE.,TIME,STATE,RATE,SCALE,NDEC,NEQ)
STOP
END

Figure 46. FORTRAN code for the main program EUDRIV. Bold characters form part of the changes that should be introduced in the program to transform it into a subroutine.
In the Euler integration method the first integration is also the final one, thus KEEP management is somewhat superfluous. The reasoning, however, is fundamentally correct and used as a preparation for integration problems including discontinuities in Runge-Kutta (Section 10.3). Note that when the OUTPUT flag is true, KEEP always equals 1, which means that only final results are written. If KEEP equals 1, however, OUTPUT is not necessarily true, because a number of final integrations is attained between two consecutive OUTPUTs.

In Figure 46, lines 102-108, it is checked whether the current time (TIME) plus the timestep (DEL(f)) does not exceed the next output moment (TNEXT), because it is desired that output occurs exactly at TNEXT. This procedure is applied because rounding errors might occur in adding up DEL(T) to TIME in each integration step (line 110).

The statements in lines 101 and 114 enable us to program discontinuities in MODEL. From Subsection 7.2.4 it is known that time manipulation in connection with discontinuities should take place via the timer variable DELMAX. The calculations concerning the discontinuity can be well controlled by taking DELT as the minimum value of DELMAX and DELT0. Here, DELMAX is previously read from the file TIMER.DAT in line 35, whereas DELT0 is the value of the time step at time zero that is a multiple of PRDEL (lines 44-57). Though after the discontinuity, DELMAX would be reset to PRDEL in MODEL (see e.g. Figure 17 in Subsection 7.2.4), DELT would nevertheless stay small because of line 101 with the result that the simulation would hardly proceed anymore. Therefore, DELT is reset to DELT0 in line 114 again.

The last value of TNEXT will exceed that of FINTIM by one PRDEL, because IP (lines 65 and 94) will finally get the value of -1. Then, the HALT flag is set in line 96, the integration of the final rates is skipped (HALT=.TRUE. in line 99) and the dynamic loop is left via the lines 116, 82 and 117. Finish conditions other than time might be induced in MODEL by setting the flag TERMNL true, as already remarked in the discussion about Figure 44. The reaction of the main program on setting TERMNL true depends on the value of the output flag (OUTPUT). If OUTPUT is true, output will have been generated in MODEL. On returning to the main program, the HALT flag is set true in line 96, integration is skipped (TERMNL=.TRUE. in line 99) and model execution is terminated as if FINTIM would have been reached. If OUTPUT is false and termination would be requested the reaction is different. On returning to the main program, the HALT flag is not set true (because OUTPUT is false, line 92), integration is skipped (TERMNL=.TRUE. in line 99), OUTPUT is set true in line 84, the rate calculation in MODEL is repeated, after which the HALT flag is set true in line 96, and the loop is left via the lines 99, 116, 82 and 117. This means that indeed the time loop not only guarantees the generation of final output when a PRDEL is reached, but also when the simulation is interrupted by the model by setting its terminal flag TERMNL true.

The initial part (lines 25-81) In the initial part of the program first the file MODEL.LOG is opened and the message 'Initialize model' is written to this file and to the computer screen (lines 25-29, see also Figure 45), second the timer variables are read from the file TIMER.DAT (lines 30-37, see also Figure 45), third some possible error conditions with respect to the timer variables are checked (lines 38-43), fourth an integration time step DELT is calculated that is a multiple of PRDEL and equal to or smaller than the DELT that is provided by the user (lines 44-59), fifth timing and finish conditions are initialized (lines 60-65), sixth the state and rate arrays are nullified to simplify debugging (lines 66-70), seventh MODEL is called for all preparative calculations needed to start the simulation (line 72), and eighth, it is checked if the number of equations in MODEL (NEQ) is specified and whether it does not exceed the maximum number that is possible in the program EUDRIV (NDEQ) (lines 73-77). The last number is now specified in line 15 as 250, but it can be adapted as needed. Only some of these points need explanation.

Consider the call to MODEL in line 72. This call replaces lines 6-10 in Figure 44. It means that only some of the computer instructions in MODEL have to be executed. Similarly, in the dynamic part of the program (line 90 in Figure 46) only the rate calculations have to be executed, and after completion of the dynamic loop, MODEL is called for terminal calculations (line 122). This means that MODEL should have different begin and end points for initial, dynamic and terminal calculations, which is reflected in the first argument of each call: this will be discussed later on.

137
The opening of files (line 26), reading of data (lines 30-37) and reporting of errors (lines 38-43 and 73-77) could be programmed in FORTRAN each time it is needed. However, such program code is needed in each model, its programming diverts attention from the actual problem, it is cumbersome and it is liable to errors. Therefore, a library with utility subroutines was developed that enormously facilitates programming. Table 6 summarizes the most important functions in this library. The formal explanation of the functions in Table 6 combined with their application in program EUDDRIV will provide a full understanding of the use of the functions. The initial part of the program is ended by writing the line 'Dynamic loop' to MODEL.LOG and to the computer screen (lines 78-81).

The terminal part (lines 118-122) The terminal part of the program is entered after leaving the dynamic time loop. First 'Terminate model' is written to the file MODEL.LOG. Then, MODEL is called to enable the user to perform some last calculations based on e.g. the results from the simulation.

The communication between the main program and the user subroutine MODEL (lines 8-9, and the list of arguments in subroutine MODEL) The communication between the main program and MODEL takes place via a small common block (Subsection 7.3.4) called INFO (lines 8-9), and by the list of arguments of subroutine MODEL, e.g. line 72. The common block INFO contains the variables that control the time (DELT, PRDEL, DELMAX, FINTIM), the integration status (KEEP) and the finish or termination flag of the model (TERMNL). The timer variables are included in the common to be able to manipulate time when discontinuities occur, and to have the possibility to print these variables. INFO also contains the unit number of an opened logfile (IULOG) and the first of a series of free unit numbers (IUMOD) that can be used by MODEL. The first argument in the list of arguments in the call to MODEL determines which part of the model should be evaluated (1: initial; 2: dynamic or 4: terminal). The second and third argument control the writing of results and the current time, respectively. The fourth and fifth argument transfer the state and rate arrays and the seventh and eighth argument the maximum and actual array lengths from the main program to MODEL and vice versa. The SCALE parameter is not used in Euler integration. It is included merely to have a similar list of arguments as needed in Runge-Kutta integration as discussed later on. A similar list of arguments is practical because then different integration algorithms can be used to solve the differential equations defined in MODEL.

Finally, the variables that occur in the INFO common block and the local variables in the program are all individually given a type declaration in lines 5-7 and 11-17, respectively.

The user's subroutine MODEL Figure 47 shows the lay out of the user's subroutine MODEL, for the example of exponential growth. This lay out can be used as the starting point to develop FORTRAN simulation programs. The actual programming activity of the user is limited to the bold characters in the listing. The remaining part of the program should not be changed. In the discussion on the initial, dynamic and terminal segment in Figure 46, it was stated that in each of these segments a different part of the user's MODEL had to be executed. These different parts of MODEL are selected by the value of the first argument in the call to MODEL: 1 for initial; 2 for dynamic; 4 for terminal. In MODEL this parameter is called ITASK (line 2, Figure 47). The execution of the different MODEL parts is controlled by means of an IF-THEN-ELSE-END IF structure: lines 29, 53, 73, and 81. The task controlled execution enables us to put the user statements together in one legible subroutine. Of course, the user is free to further structure MODEL by using subroutines within each part (Chapter 7). Below, the dynamic, the initial, the terminal and the communication and declaration of variables in MODEL are shortly considered. The calls to the FORTRAN utility functions in MODEL are best understood in conjunction with Table 6.

The dynamic part (lines 53-72; ITASK = 2) The dynamic part starts with copying the general array of states (STATE, here only one element) in a user specific name, here A (line 57). Then, driving variables can be read and rates can be calculated (lines 58-60). Now, the states and rates are synchronously known, and output might be desired (lines 61-67). At the end of the dynamic, the user specified rate, here GR, is copied in the general rate array (RATE, line 72). However, also the value of state A is copied into the general STATE array (line 70). The latter action is only needed in fact when the values in the user state array, here only element A, is changed in MODEL, for
Table 6. FORTRAN functions from the library TTUTIL (Rappoldt & van Kraalingen, 1990).

**Interactive data entry from screen:**

- **CALL ENTINT('QUEST', I**)
- **CALL ENTREA('QUEST', X)**
- **CALL ENTCHA('QUEST', S)**
- **CALL ENTDIN('QUEST', S)**
- **CALL ENTDRE('QUEST', XDEF, X)**
- **CALL ENTDCH('QUEST', SDEF, S)**

Interactive entry of an INTEGER number IX. Writes the character string QUEST on the screen as a question and returns entered number to the calling program.

Interactive entry of a REAL number X. For further description see ENTINT.

Interactive entry of a CHARACTER string S. For further description see ENTINT.

Interactive entry of an INTEGER number IX with a default IXDEF. Default value is assumed when <RETURN> is given. For further description see ENTINT.

Interactive entry of a REAL number X with a default XDEF. For further description see ENTINT.

Interactive entry of a CHARACTER string S with a default SDEF. For further description see ENTINT.

**Opening of files:**

- **CALL FOPENG(I UNIT, 'FILE', 'STATUS', 'TYPE', 'IRECL', 'PRIV')**

Opens formatted (F), unformatted (U) or binary (B) file with sequential (S) or direct (D) access.

- **I UNIT** unit number used to open the file.
- **FILE** name of the file to be opened.
- **STATUS** status of the file.
- **TYPE** string containing code for format-keyword (F, U, or B), and code for access-keyword (S or D).
- **IRECL** record length of direct access file. Parameter is dummy, e.g. 0, in case of sequential file.
- **PRIV** privilege; in the case that STATUS='NEW' and file exists:
  - 'DEL': old file is overwritten,
  - 'NOD': old file is saved and program stopped,
  - 'UNK': interactive choice.

**Initializing data files:**

- **CALL RDINIT(I UNIT, IULOG, 'DATFIL')**

Initializes the reading of a data file with the routines RDSINT, RDSREA, and RDAREA.

- **I UNIT** unit number to open a temporary random access file for I/O; (UNIT+1) used to open the data file.
- **IULOG**: If > 0, unit number of log file used to report data file syntax errors.
  - If = 0, nothing is done with a log file.
- **DATFIL** name of the data file.

**Reading data from file:**

- **CALL RDSINT('XNAME', I**)

Reads the single integer numerical value IX of the variable named XNAME from a data file. The reading should be initialized with RDINIT.

* Example of a data file:

  A = 3.4 ; IB = 5
  ! A is a real variable, IB is an integer.

  ZTB = 1.0, 3.4, 1.2,
  ! Last comma is needed to continue an array.

  H = 10*0.0, 0.5, 9*0.0
  ! No spaces can be used in expressions like 10*0.0.

* End of example data file.
Routine RDSINT may read the value IB by: CALL RDSINT ('IB', IB).
Reading A would result in the value 3, because the nearest integer function is used.

CALL RDSREA ('XNAME', X)

Reads the single real numerical value X of the variable named XNAME from a data file. The reading should be initialized with RDINIT. This routine may read the value of A from the above data file by:
CALL RDSREA ('A', A).

CALL RDAREA ('XNAME', X, $ ILDEC, IFND)

Reads the real numerical values X of the array named XNAME from a data file. ILDEC is the declared length of X; IFND is the actual number of values found in the file. The reading should be initialized with RDINIT. This routine may read the value of ZTB from the above data file by:
CALL RDAREA ('ZTB', ZTB, 100, ILZ). Here, the value of ILZ will get the value of 6. H may be read from the table by:
CALL RDAREA ('H', H, 25, ILH), and the value of ILH will be 20.

Special functions:

\[ Y = \text{LINT}(\text{TABLE}, \text{ILTAB}, X) \]

Linear interpolation of the one-dimensional array \( \text{TABLE} \) at the value of the independent variable \( X \). \( \text{TABLE} \) contains paired data: \( x_1, y_1 \), \( x_2, y_2 \), \( x_3, y_3 \), \ldots, \( x_n, y_n \). \( \text{ILTAB} \) is the number of elements in the array, i.e. \( 2 \times n \). Extrapolation occurs when the value of \( X \) is beyond the defined region, but this also results in a warning to the screen. The result of the interpolation returns to the calling program in the function name 'LINT'. LINT is the equivalent of the AFGEN function of CSMP in Section 3.6, Table 1.

\[ Y = \text{INSW}(X_1, X_2, X_3) \]

Input switch relay depending on sign of \( X_1 \). \( Y = X_2 \) if \( X_1 < 0 \); \( Y = X_3 \) if \( X_1 \geq 0 \). The result of the switch returns to the calling program in the function name 'INSW'. INSW is the equivalent of the INSW function of CSMP in Section 3.6, Table 1.

\[ Y = \text{LIMIT}(\text{MIN}, \text{MAX}, X) \]

Limiter: returns the value of \( X \), limited within the interval [\( \text{MIN}, \text{MAX} \)], in the function name 'LIMIT'. LIMIT is the equivalent of the LIMIT function of CSMP in Section 3.6, Table 1.

General:

CALL ERROR ('MODULE', 'MESSAG')

Writes an error message to the screen and holds the screen until the <RETURN> is pressed. MODULE contains a string with the name of the (sub)program where the error occurs; MESSAG contains the string with the error message.

CALL UPPERC (STRING)

Converts STRING to uppercase characters.

Output of data:

CALL OUTDAT (ITASK, IUNIT, $ 'RN', R)

Generates output files. The subroutine has three main tasks, namely to initialize the output file (ITASK=1), to store the data gathered during the calculation process (ITASK=2), to generate the output file in a desired format (ITASK=4, 5, or 6), and to delete temporary files (ITASK=99).

ITASK Integer that can take the value 1, 2, 4, 5, 6, or 99.
ITASK=1 initializes the subroutine, opens a temporary file for storage, and stores the name of the independent variable and the unit

ITASK=2 stores the data gathered during the calculation process.

ITASK=4 generates the output file in a desired format.

ITASK=99 deletes temporary files.

IUNIT unit number for writing to output file. If the unit defined during ITASK=1 is open, this is used for output. Otherwise, a file RES.DAT is created, using that unit. IUNIT+1 is used as a temporary file.

RN either a text string or a name of variable (up to 11 characters will be used). If ITASK is 4, 5, or 6, this string will be written to the output file as title (not limited to 11 characters).

R value of variable if ITASK=2. Otherwise, R has no significance.

R value of variable if ITASK=2. Otherwise, R has no significance.

R value of variable if ITASK=2. Otherwise, R has no significance.

RN either a text string or a name of variable (up to 11 characters will be used). If ITASK is 4, 5, or 6, this string will be written to the output file as title (not limited to 11 characters).

R value of variable if ITASK=2. Otherwise, R has no significance.

R value of variable if ITASK=2. Otherwise, R has no significance.
**CALL OUTARR ('NAME', ARRAY, I1, I2)**

Transfers the contents of an array named NAME to the OUTDAT routine. ARRAY the array that has to be transferred. I1 Array element where output should start. I2 Array element where output should finish. Example: CALL OUTARR ('ABC', ABC, 1, 10) causes the transfer of the elements ABC(1) to ABC(10) to the OUTDAT routine, which can then be used to generate an output table.

**CALL OUTPLT (ITASK, RN)**

OUTPLT is used to printplot a selection of the stored variables. By repeated calls to the subroutine with ITASK=1, names of variables for which the plot is wanted can be supplied to the subroutine. Calls with ITASK = 4, 5, 6, or 7, generate printplots with a width of 80 or 132 columns, either with individual scaling or with common scaling (all variables scaled to the smallest and largest value in the data set). OUTPLT is designed to be used in conjunction with OUTDAT, which is used to write a variable name and its value to a temporary file.

ITASK 1: instructs the routine to store variable names for use in the printplot.
4: wide format (132 columns), individual scaling.
5: wide format, common scaling.
6: small format (80 columns), individual scaling.
7: small format, common scaling.

If the unit defined during ITASK=1 of OUTDAT is open, this is used for output. Otherwise, a file 'RES.DAT' is created, with that unit. RN either a string or a name of variable (up to 11 characters will be used). The value of the variable must have been stored by previous calls to OUTDAT.

* Example:
CALL OUTPLT (1, 'DTGA')
CALL OUTPLT (1, 'WSO')
CALL OUTPLT (5, 'Plot title')

**CALL OUTCOM (STRING)**

Stores a text string that is written to the output file generated by subroutine OUTDAT. A maximum of 25 strings of 80 characters can be stored.

---

Time is made the independent variable. It is initialized at 0.

2: stores the name and the value of a variable in a temporary file during the calculations. Example:

CALL OUTDAT(2,0,'TIME',TIME)
CALL OUTDAT(2,0,'RATE',RATE)
CALL OUTDAT(2,0,'D(11)',D(11))

The second argument in OUTDAT has no meaning when ITASK=2. Therefore, 0 is written.

4: generates table output.
5: generates spreadsheet output.
6: generates two column output.

Example:
CALL OUTDAT(4,60,'Title',0.)

After using OUTDAT with ITASK equal to 4, 5, or 6, the subroutine can be used again by initializing with ITASK=1.

99: deletes temporary files. Example:

CALL OUTDAT(99,0,'',0.)

Transfers the contents of an array named NAME to the OUTDAT routine.
Example: Exponential growth (CSMP in Exercise 21)

SUBROUTINE MODEL (ITASK, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)

* The STANDARD (!!) parameter list:

* ITASK - task of model routine I
* OUTPUT = .TRUE. output request (ITASK=2 only) I
* TIME = -time I
* STATE - state array of model I/O
* RATE - rates of change belonging to STATE I/O
* SCALE - size scale of state variables I/O
* NDEC - declared size of arrays I
* NEQ - Number of state variables, for ITASK=1 O otherwise I

IMPLICIT REAL (A-Z)

* formal parameters, =do not change this section !=
INTEGER ITASK, NDEC, NEQ
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL OUTPUT

* common /INFO/, =do not change this section !=
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMCD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT, PRDEL, DELMAX, FINTIM,
$ IULOG , IUMCD, KEEP , TERMNL

* local (non-common) variables
INTEGER IURES, IUDAT
INTEGER N

IF (ITASK.EQ.1) THEN
* initial
* get local unit numbers
IURES = IUMOD
IUDAT = IUMOD + 2
* open input file
CALL RDINIT (IUDAT, IULOG, 'MODEL.DAT')
* initial state
CALL RDSREA ('IA', A )
* model parameters (see data file for meaning of symbols)
* CALL RDSREA ('RGR', RGR)
CALL RDSINT ('N', N )
* scales
CALL RDAREA ('SCALE', SCALE, NDEC, NEQ)
CLOSE (IUDAT, STATUS='DELETE')
* check the size of the SCALE array on file:
IF (NEQ.NE.N) CALL ERROR ('MODEL','SCALE length mismatch')
$ initialize output
CALL OUTDAT (1, IURES, 'TIME', 0.0)
* initialize state variables
STATE(1) = A

ELSE IF (ITASK.EQ.2) THEN
* rates of change
* assign state to local variable names
A = STATE(1)
* driving variables (absent here)
* calculation of rates of change
GR = RGR * A
* output section
  IF (OUTPUT) THEN
    CALL OUTDAT (2,0,'TIME',TIME)
    CALL OUTDAT (2,0,'DELT',DELT)
    CALL OUTDAT (2,0,'A' , A )
    CALL OUTDAT (2,0,'GR' , GR )
  END IF
  * assign result to state array (Needed when
    * discontinuities in states occur)
    STATE(1) = A
  * assign result to rate array
    RATE(1) = GR
  ELSE IF (ITASK.EQ.4) THEN
    * terminal
    $ (4,0,'Results of exponential growth',0.0)
    * delete temporary file
    CALL OUTDAT (99,0,' ',0.0)
  END IF
  RETURN
END

Figure 47. Lay out of user's subroutine MODEL, for the example of exponential growth. Bold characters indicate what should be programmed by the user. The remaining part of the program should not be changed.

example by the 'replace method' (Subsection 7.2.4). The adjustment should be done directly at the beginning of the dynamic part of MODEL, i.e. between the lines 57 and 58.

The initial part (lines 29-52; ITASK=1) The main objective of the initial is to read the input data from the file MODEL.DAT (lines 35-45), to initialize the time in the output file (line 50), and to perform all initial calculations needed. MODEL needs a log file unit number (IULOG, line 36). Further the integer variable IUMOD is the first of a series of free unit numbers. In Figure 47 IUMOD is used for output and IUMOD+2 for input (lines 33, 34, 36, 45 and 50).

The number of equations in the model, N (here 1), should be provided by the user (line 42). It is used to check in lines 47 and 48 if this number is the same as the number of scale values (SCALE) that is determined by subroutine RDAREA (line 44). The SCALE parameter is discussed later on in connection with the Runge-Kutta integration.

The terminal part (lines 73-81; ITASK=4) The terminal part is used here to write a two column output of time versus A and time versus GR (lines 76-78). Thereafter, the temporary file with unit number IURES+1, is deleted (line 80). However, the terminal part may also contain other statements that are to be executed only once.

Communication and declaration of variables (lines 2-28) The communication between MODEL and the main program EUARD (lines 2-3 and 19-24) was discussed earlier and is not repeated here. Non user variables are individually given a type declaration in lines 16-18 and 25-28. Furthermore, use is made of the IMPLICIT REAL statement (line 14), so that the modeller has only to declare exceptions to this data type, analogous to CSMP (see Subsection 7.3.2). MODEL starts with a header, lines 4-13, where the arguments of the list of arguments in the subroutine definition is explained.

In Figures 46 and 47 data are read from the files TIMER.DAT and MODEL.DAT (see also Figure 45). Data files consist of a combination of variable name and associated value(s). Note that variables may appear in any order in the file, comment lines start with '* ' in the first column, or '! ' in any column, different name-value combinations on the same line should be separated by a semicolon ';', no tabs may occur, and each line should end with a return. The MODEL.DAT file for Figure 46 is as follows:
The TIMER.DAT file for Figure 46 reads:

```
<File TIMER.DAT>
* Simulation control variables
STTIME = 0.0 ! start time
FINITIM = 10.0 ! finish time
PRDEL = 1.0 ! output time step
DELMAX = 1.0 ! maximum allowed timestep
DELT = 0.001 ! timestep Euler integration
<End>
```

Another example of a data file for MODEL.DAT is given in Table 6 under the caption 'Reading data from file'.

The four principal drawbacks that were enumerated at the end of the discussion about Figure 44 are fully abolished in the program of Figures 46 and 47. What still lacks in the program, however, is the possibility to make reruns without having to change the parameter values in the file MODEL.DAT for each new run. Therefore, we now first discuss how the data for which the rerun should be performed is read, and then how the program code for the rerun facility looks like.

**Reading data for a rerun** The rerun facility introduced here is based on the idea that the data files as defined above should remain unchanged and that data for reruns should be specified in a separate file called RERUNS.DAT. This file may contain the names and values of variables from both the 'standard' data files (MODEL.DAT and TIMER.DAT) that are read by the program. In the first run, the values from the standard data files will be used. In subsequent runs these values are replaced by those from the rerun file. Execution of the program will continue until all rerun sets defined in RERUNS.DAT have been used. Basically, the file RERUNS.DAT has the same syntax as the standard data files, except that it may consist of identical sets of rerun variables, and, as a consequence, names of variables may appear more than once in the file. The order and number of the rerun variables should be the same in each set. A new data set starts when the first variable is repeated. Unlike in CSMP, each variable whose value is changed somewhere in the rerun file should be assigned a value in each set, even if that value is similar to the one in the previous set. An example of a RERUNS.DAT file where three reruns are defined for the model in Figure 46 reads:

```
<File RERUNS.DAT>
* reruns.dat
IA = 1.0 ; RGR = 0.01 ; PRDEL = 1.0 ! rerun set 1, model run 2
IA = 1.0 ; RGR = 0.1 ; PRDEL = 10.0 ! rerun set 2, model run 3
IA = 2.0 ; RGR = 0.1 ; PRDEL = 10.0 ! rerun set 3, model run 4
<End>
```

This file specifies three reruns, namely rerun set 1 to 3. The first rerun set corresponds with the second model run, since in the first model run the data from MODEL.DAT and TIMER.DAT are used. If the file RERUNS.DAT is absent or empty, the model will execute one single run, using the data from the standard data files, as explained below.

**The program code of the rerun facility** Performing reruns implies that program EUDRIV in Figure 46 should be invoked from time zero onwards using different input data. Thus, the control structure for reruns should be programmed in a loop 'around' the actual model, as can be deduced from Figure 42. This was implemented by transforming program EUDRIV into a subroutine that is invoked by a main program where measures are taken to read the rerun data. The overall arrangement of the program modules is indicated in Figure 48. The new portion in this figure, program
Figure 48. Lay out of FORTRAN simulation modules using the integration method of Euler, showing a main program FORSIM, the driver subroutine EUDRIV (formerly 'program EUDRIV'), the user subroutine MODEL, their communication, input files (RERUNS.DAT, TIMER.DAT and MODEL.DAT) and output files (MODEL.LOG and RES.DAT).

FORSIM, and the minor changes in program EUDRIV from Figure 46 to transform it into subroutine EUDRIV, need some explanation.

The program code of FORSIM is shown in Figure 49. We first consider lines 18-25. The call to RDSETS (line 18) detects if a RERUNS.DAT file is present and if so, it analyses the data file. The variable INSETS is returned and contains the number of rerun sets present in the rerun file; its value is zero if the rerun file is absent. DO-loop 20 (lines 20-25) runs INSETS+1 times, because it starts at the counter value IS=0, which corresponds with the first simulation run. The value of the DO-loop counter is then used in the call to RDFROM to select a parameter set for the simulation. If IS equals zero, the standard data files (MODEL.DAT and TIMER.DAT) will be used by the RD routines (see Table 6). If IS is larger than zero, the RD routines will replace the values in the standard files with those from the rerun file. Therefore, no changes are necessary in the user's MODEL, as these replacements are internal to the RD routines. Results of the first simulation run and the reruns are written to the output file after completion of all runs, since this is done in the terminal section of MODEL (Figure 47, line 77-78). Subroutine RDFROM is also called before a rerun is started to check if all the variables of the preceding set were used. If this is not the case, it is assumed that there is a typing error in the data files and the simulation is halted when the logi-
* PROGRAM FORSIM                                                                 line 1
  * declarations                                                                 line 2
    INTEGER IS, IULOG, IURER, IUDRIV, IUMOD                              line 3
    REAL DUM                                                               line 4
    LOGICAL FATAL                                                        line 5
    EXTERNAL MODEL                                                     line 6
  * unit numbers used for logfile and for rerun facility                line 7
    DATA IULOG/20/, IURER/30/                                         line 8
  * unit numbers used for driver and model                              line 9
    DATA IUDRIV/40/, IUMOD/50/                                         line 10
  * fatal errors on non-used rerun variable values                      line 11
    DATA FATAL/.TRUE./                                                 line 12
  * dummy model call (some machines do not correctly                  line 13
    externalyze MODEL)                                               line 14
    CALL MODEL (0,.FALSE.,DUM,DUM,DUM,DUM,1,IS)                        line 15
  * open logfile and analyse rerun file                                line 16
    CALL FOPENG (IULOG,'MODEL.LOG', 'NEW', 'SF', 0, 'DEL')             line 17
    CALL RDSETS (IURER, IULOG, 'RERUNS.DAT', INSETS)                   line 18
  * model runs                                                        line 19
    DO 20 IS = 0, INSETS                                               line 20
      WRITE (IULOG, '(/A14,,A)') ' Run',IS+1,'                         line 21
      WRITE (*, '(/A14,,A)') ' Run',IS+1,'                           line 22
      CALL RDFROM (IS,FATAL)                                          line 23
    END                                                                 line 24
    CALL EUDRIV (IULOG, IUDRIV, IUMOD, MODEL)                          line 25
  20  CONTINUE                                                        line 26
    IF (INSETS.GT.0) CLOSE (IURER, STATUS='DELETE')                   line 27
    STOP                                                               line 28
END                                                                  line 29

Figure 49. The main program FORSIM that takes care of the rerun facility and calls the driver subroutine EUDRIV.

cal input variable FATAL is set true (Figure 49, line 12 and 5). If this variable is set false, a warning is produced in the logfile MODEL.LOG. Further details about the subroutines RDSETS and RDFROM can be found in Rappoldt & van Kraalingen, 1990. For writing to the log file, it should be opened, line 17 (this line previously occurred in program EUDRIV, Figure 46, line 26). The name of the model subroutine is declared external by means of the EXTERNAL statement in line 6 (Wagener, 1980). This statement allows to pass a program unit name, e.g. a subroutine name, as an actual argument in the list of arguments in a subroutine, here subroutine EUDRIV in line 24. Now, instead of using the name MODEL, another name could be used for the user’s program, while the new name has only to be defined in the main program FORSIM (lines 6, 15 and 24). No further changes have to be made in the driver EUDRIV and the integration routines in the library TTUTIL (Rappoldt & van Kraalingen, 1990). On some computers the linker appears to have problems with handling external subprograms. Problems tend to arise especially when routines with external modules in their parameter list (like EUDRIV) are linked from an object library. Then, inclusion of the driver EUDRIV by means of an include statement in the program source will almost always cure the problem. Also lower level routines may have to be included. With Absoft FORTRAN on Apple Macintosh the problem can be solved by including in the main program a dummy call to the model subroutine (line 13-15):

    CALL MODEL (0,.FALSE.,DUM,DUM,DUM,DUM,1,IS)

It is easily verified that this call does not change anything (see Figure 47). It just takes over the function of the external statement that seems to be incorrectly dealt with.

The minor changes needed to transform the program EUDRIV from Figure 46 into subroutine EUDRIV concern the addition and replacement of some statements, and the transfer of some other
statements from program EUDRIV to program FORSIM. The additions to program EUDRIV concern the line 'EXTERNAL MODEL' between lines 3 and 4, and the line 'SAVE' between lines 17 and 18. The replacements concern line 1 and line 123, where 'SUBROUTINE EUDRIV (IUL, IUD, IUM, MODEL)' and 'RETURN' should be written, respectively. The transfer of statements concerns lines 18-20 and lines 25-26 to program FORSIM, Figure 49. These lines now occur in lines 7-10 and 16-17 of FORSIM, respectively.

Though the simulation program of Figure 43, with which we started, is very different from the final simulation program, which is composed of Figure 49, the adapted Figure 46, and Figure 47, the sequence of the calculations in the initial, dynamic and terminal parts of the time loop are still the same as depicted in Figure 42.

In summary, the modeller has to develop subroutine MODEL, according to the lay out of Figure 47, and to write the data files RERUNS.DAT, TIMER.DAT and MODEL.DAT. If no reruns are planned, the file RERUNS.DAT may be omitted. Furthermore, program FORSIM and the subroutine library TTUTIL should be available.

Exercise 86
a. Adapt the user's subroutine MODEL (Figure 47) for the problem of 'Exponential growth with temperature dependent relative growth rate', using the equations from the CSMP program (answer to Exercise 25) as the starting point.
b. Write the data files TIMER.DAT and MODEL.DAT using the numerical values of the answer to Exercise 25.

Exercise 87
a. Adapt the user's subroutine MODEL (Figure 47) for the problem of 'State event using the replace method', using the equations from the CSMP program (answer to Exercise 57) as the starting point.
b. Write the data files TIMER.DAT and MODEL.DAT using the numerical values of the answer to Exercise 57.

Exercise 88
a. Adapt the user's subroutine MODEL (Figure 47) for the problem of 'Elementary description of diffusive flow', using the equations from the CSMP program (answer to Exercise 78) as the starting point.
b. Write the data files TIMER.DAT and MODEL.DAT using the numerical values of the answer to Exercise 78.

10.3 Runge-Kutta integration

The time loop presented in Figures 42 through 46 is based on Euler's integration method, because the rate equations are evaluated only once per time step. In higher order integration methods, the rates must be evaluated more often. For example, in the second order trapezium method the rates are evaluated twice. Using the first (trial) rate, a (trial) integration is carried out according to Euler's method. Subsequently, a second (trial) rate is calculated. The average of the two trial rates is used for final integration (Section 6.2). Obviously, intermediate results of the calculations should be temporarily stored. In Figure 46 (line 112), or Figures 43 and 44, the results of the integration are not stored, however. There all results are final and they are directly used to proceed the simulation with time. If the first integration would have to be a trial integration, it should be possible to invoke the rate equation from within the integration method, i.e. in line 112 in Figure 46, and store the intermediate results. This is exactly what higher order integration methods do. For example, in CSMP the selected integration routine invokes subroutine UPDATE (Subsection 7.3.1
and Appendix 3) in which only the rate equations appear (the state variables (INTGRL's) are made comment lines by placing a 'C' in the first column. When we want to perform simulations in FORTRAN using higher order integration routines, a similar arrangement of calculations as in CSMP is needed. Figure 50 shows the overall arrangement of the program modules needed for higher order integration. In particular, the Runge-Kutta integration method with self adapting time

Figure 50. Lay out of FORTRAN simulation modules using the integration method of Runge-Kutta, showing the main program FORSIM, the driver subroutine RKDRIV, the user subroutine MODEL, the integrating subroutine RKQCA (and RK4A), their communication, input files and output files.

step will be dealt with. Compared with Figure 48, the new parts in Figure 50 concern the driver (RKDRIV) and the integrating subroutine RKQCA that controls the time step of integration by evaluating the integration error, stores intermediate results, invokes MODEL and integrates the rates via subroutine RK4A. Subroutines RKQCA and RK4A are slightly adapted versions of RKQC and RK4, described in 'Numerical Recipes' by Press et al. (1986, Sections 15.1 and 15.2). For a treatment of the mathematical background of the Runge-Kutta routines, the reader is kindly referred to that book. We restrict ourselves to some main points, namely the calculation of the integration
error and the main differences in the integration section of RKDRIV compared with EUDRIV. Some miscellaneous remarks conclude this Section.

**Calculation of the integration error** The Runge-Kutta integration method with a fixed time interval requires four evaluations of the rate equations (Section 6.2). If adaptive time step control is desired, however, information about the performance of the integration is needed, in particular an estimate of the error. This could be obtained by comparing the results of two integration methods that differ one order in accuracy, e.g. the RKS method from CSMP (Section 6.2), but another possibility is by time step doubling, as applied here (Press et al., 1986). Then each time step is taken twice, once as a full step, and, independently, as two half steps. Each Runge-Kutta integration requires four evaluations of the rates, but since the single step and the double step sequences share a starting point, 11 evaluations are needed. The shared starting point is the main reason that the first call to MODEL takes place from RKDRIV, Figure 50. The number 11 should be compared to 8 (the two half steps), since, apart from the step size control, the accuracy that is achieved will be that of the smaller step size. The difference between the two numerical calculations is an indicator of the integration error. It can be compared to some desired user specified integration error, \( \Delta_{\text{des}} \), to decide about the next time step for solving the model equations. The desired error is calculated from the product of the maximum value that may occur for a state variable \( i \), called SCALE(\( i \)), and an overall tolerance level, called EPS. The accuracy of the \( i \)th equation will then be taken as \( \Delta_{\text{des,}i} = \text{EPS} \times \text{SCALE}(i) \). The maximum values of the state variables or, more loosely formulated, the order of magnitude of the state variables in the course of the simulation, is provided to the program by means of the SCALE(\( i \)) array. For further details about the error handling procedure and the time step calculation, see Press et al. (1986).

**The main differences in the integration section of RKDRIV compared to EUDRIV** The program code for subroutine RKDRIV is shown Figure 51. The lines 84-99 from Figure 51 may be compared with lines 99-115 of Figure 46. In line 86 of Figure 51 the time step for the next integration is taken as the smallest of either the advised time step, DELNXT, from the integration routine or the maximum time step, DELMAX, provided by the user in TIMER.DAT. Then, if \( \text{TIME} + \text{DEL} \leq \text{TNEXT} \), DELT is set equal to DELNXT and the integration routine RKQCA is called (lines 90-91). That routine calls MODEL 11 times using the time step DELT. When the result would not satisfy the accuracy criterion, a smaller time step would be taken and another 11 rate calls made. Since the first rate call is based on the previous final result of integration, it is valid until a new final integration has been achieved. Therefore, it is needed as an input to the RKQCA routine. When the value of \( \text{TIME} + \text{DELNXT} \) exceeds that of the next output time \( \text{TNEXT} \), DELT is reduced to \( \text{TNEXT} - \text{TIME} \) before calling RKQCA. Returning from RKQCA, the new advice for \( \text{DELNXT} \) may be too small a number (as a result of the reduction of DELT in line 95). Therefore, that advice is ignored by means of the variable DUMMY and the current value of DELNXT is used again.

The integration routine RKQCA can only perform rate calls to MODEL when it 'knows' the formal parameters of the user's model subroutine. Therefore, the parameter list of MODEL has been made independent of the problem at hand, by using general names as STATE, RATE, etc. (Figure 47, lines 2-3). By means of the EXTERNAL MODEL statement (Figure 51 line 4) routine RKQCA also 'knows' that the argument MODEL in its list of arguments (Figure 51 line 91 and 97) is in fact a subroutine name.

**Miscellaneous remarks** The writing of results should take place when an integration is completed. This is the case when routine RKQCA returns to the driver routine RKDRIV. Handling of the OUTPUT flag and the integration status (KEEP) is similar to Euler integration (Figure 46), but the statement KEEP=0 (Figure 51, line 76) now gets its meaning, namely that all integrations except the last, that are performed from routine RKQCA, are trial integrations. Therefore, no output of results should take place during these trial calculations. To this purpose the OUTPUT flag is set to .FALSE. in the lists of arguments in the calls to MODEL from RKQCA and RK4A (see Appendix 6). The first call to MODEL from RKDRIV takes place under KEEP=1 because these calculated rates indeed belong to the final results of the state variables.
SUBROUTINE RKDRIV (IUL, IUD, IUM, MODEL)

formal parameters
INTEGER IUL, IUD, IUM
EXTERNAL MODEL

common /INFO/
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT, PRDEL, DELMAX, FINTIM,

local (non-common) variables
INTEGER I, IP, NEQ, NDEC
REAL STTIME, EPS, TIME, TNEXT, DELDID, DELNXT, DUMMY
REAL STATE, RATE, SCALE
PARAMETER (NDEC=250)
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL HALT, OUTPUT
SAVE

get copy of logfile unit number and model unit number
IULOG = IUL
IUMOD = IUM
IF (IULOG GT 0)
  WRITE (IULOG, '(A)') 'Initialize model'
  WRITE (*, '(A)') 'Initialize model'

read timer variables
CALL RDINIT (IUD, IULOG, 'TIMER.DAT')
CALL RDSREA ('STTIME', STTIME)
CALL RDSREA ('FINTIM', FINTIM)
CALL RDSREA ('PRDEL', PRDEL)
CALL RDSREA ('EPS', EPS)
CALL RDSREA ('DELMAX', DELMAX)
CALL RDSREA ('DEL', DELT)
CLOSE (IUD, STATUS='DELETE')
IF (PRDEL LE 0.0)
  CALL ERROR ('RKDRIV', 'Illegal value PRDEL')
IF (DELMAX GT PRDEL)
  CALL ERROR ('RKDRIV', 'Illegal value DELMAX')

initialize timing
TERMNL = .FALSE.
TIME = STTIME
TNEXT = STTIME
HALT = TIME.GE.FINTIM
IF = 1 + INT ((FINTIM - STTIME) / PRDEL - 0.01)
DELNXT = DELT

to simplify debugging: set states and rates to 0 and scales to 1
DO 10 I=1, NDEC
  STATE(I) = 0.0
  RATE(I) = 0.0
  SCALE(I) = 1.0
CONTINUE

initialize model
CALL MODEL (1, .FALSE., TIME, STATE, RATE, SCALE, NDEC, NEQ)

error checks
IF (NEQ LE 0) CALL ERROR ('RKDRIV','
  'No value of NEQ was specified in MODEL')
IF (NEQ GT NDEC) CALL ERROR ('RKDRIV','
  'Too many state variables')
DO 20 I=1, NEQ
  IF (SCALE(I) LE 0.0)
    CALL ERROR ('RKDRIV','
      'At least one SCALE variable is not positive')
CONTINUE
In the initial part of the EUDRIV program (Figure 46, lines 44-59) the time step DELT was calculated as a multiple of PRDEL, so as to obtain equally spaced time steps, similar to simulation models in CSMP. This is not wanted in the Runge-Kutta driver of Figure 51, because the time step of integration should be often adapted to meet the user desired accuracy ($\Delta_{\text{des}}$).

Programming discontinuities in MODEL was made possible by the lines 101 and 114 in program EUDRIV. In the RKDRIV program no precautions have to be taken, because of the automatic adaptation of DELT via DELNXT (Figure 51, lines 86, 89 and 95).

The overall tolerance, EPS, needed to calculate the desired accuracy during the simulations, should be specified in the data file TIMER.DAT. The variable is read in line 31. (The SCALE(I) variable should be specified in the data file MODEL.DAT.) A first estimate of the time step for the driver is provided via DELT in line 33.

Since the lists of arguments of MODEL in RKDRIV and EUDRIV are identical, the user specifi-
ed MODEL may be called by both drivers. It would thus be desirable to combine Figures 48 and 50 to be able to choose between the different integration methods when running the main program FORSIM from Figure 49. This can be attained by a slight change in FORSIM. The full series of modules: FORSIM, EUDRIV, RKDRIV, RKQCA and RK4A are given in Appendix 6.
11. Additional exercises II

P.A. Leffelaar

Exercise 89

In a crop rotation system, the amount of organic matter (ORMAT) in the top soil is enriched with stubble, roots and green manure crop at an annual rate of 4000 kg ha⁻¹. In the model this increase is distributed evenly throughout the year. Furthermore, it is known that the half-live time of the organic matter is about 35 years and that each year 0.5 % of this organic matter is leached from the soil, at a rate RLCHR.

a. Draw a relational diagram for the organic matter in the soil.
b. Calculate the relative decomposition rate (RDECRT) of the organic matter.
c. Calculate the time coefficient of the system.
d. Write a simulation program for this system. Which integration method do you choose?
e. If the organic matter in the soil is mainly found in the first 25 cm and 1 m³ of soil weighs 1500 kg, what will then be the organic matter content after a long period?
f. How would you modify the simulation program to make it suitable for simulating the progress of the organic matter with time if this would not be added evenly throughout the year, but all at once each year? Use both the 'infinite rate method' and the 'replace method'.

Exercise 90

A soil scientist and a phytopathologist are requested to study the effect of a herbicide on the soil fauna. They decide to limit themselves in first instance to the effect of the herbicide on the population of the dung beetle *Typhaeus typhoeus*. According to their information, the dung beetle population shows logistic growth if the amount of herbicide is below 68 kg ha⁻¹. The relative growth rate in the logistic growth equation is 0.5 yr⁻¹. If the amount of herbicide is equal to or larger than 68 kg ha⁻¹, the dung beetle population decreases exponentially with a half-live period of 3.47 years. The original population of dung beetles is 100 ha⁻¹ and the maximum is 10000 beetles ha⁻¹. Herbicide is first applied after 3 years and subsequently every 2 years, in a single application of 30 kg ha⁻¹. In the soil herbicide is degraded through microbial action. The rate of degradation is proportional to the amount of herbicide in the soil. The average residence time of the herbicide in the soil is 10 years.

a. Draw a relational diagram for this system. Identify the different variables and indicate their units.
b. Calculate analytically the dynamics of the herbicide over a period of 11 years, if the soil contains 50 kg ha⁻¹ herbicide at the beginning of that period. Calculate the amount of herbicide only at several characteristic moments.
c. Calculate analytically the dynamics of the beetle population over the same period. Give results only at some characteristic moments.
d. Write a CSMP simulation program without a subroutine for this system.

To this purpose:
- calculate the time increment of the model;
- apply the 'infinite rate method' to deal with the discontinuity due to the herbicide application;
- use the INSW function to select either the logistic growth rate or the exponential death rate;
- only use variables in the program so that all parameters can be given values in the INITIAL segment of the program;
- check your program by making a dimensional analysis.

153
e. Write a CSMP simulation program for this system with two separate subroutines for the differential equations of the herbicide and the beetles, respectively. Remember that the IMPULS function is a history function.

**Exercise 91**

Part of a simulation program is given:

```
INITIAL
FIXED I, N, NN
N = 40
NN = N + 1
DTSYS = 10.
* DTSYS represents the total Delay Time of the SYstem (hours)
TABLE IA(1-41) = 1000., 40*0.

DYNAMIC
A = INTGRL(IA, RA, 41)

PROCEDURE RA = NFLOW (A, T)
    RA(1) = -(A(1) / T)
    DO 100 I = 2, N
        RA(I) = (A(I-1) - A(I)) / T
    100 CONTINUE
    RA(NN) = A(N) / T
```

a. What is the order of the delay described by this program?
b. How would you calculate T in terms of the other variables, in order to achieve the given total delay time?
c. If method RECT would be used for integration, what would be a reasonable numerical estimation for Δt?
d. What is the absolute and relative dispersion introduced by this program?
e. Which value does A(41) approach after a considerable period?
f. Estimate the minimum time required in which A(41) will reach the value calculated under e).
g. Calculate the time required for A(1) to decrease to 20% of its initial value.
h. What can you say about the intersection of the time courses of the amounts in integral I and (I+1), were I = 1, ..., N-1? Now sketch the progress of the amounts in A(1), A(2) and A(3) with time.

Assume that the simulation program is extended by adding a constant flow of X to A(1).

i. Rewrite the equation for the net rate of the 1st integral.
j. Which level of equilibrium will be reached finally by the 2nd integral?
Exercise 92

A researcher has derived the rate distribution presented below from a cumulative germination curve for a batch of seeds. Now he aims to obtain a reasonable corresponding curve by means of a simulation technique.

Rate of germination

\[
\frac{\Delta G}{\Delta t}
\]

values of coordinates: (5,0) (6,1) (7,9) (8,31) (9,63) (10,63) (11,47) (12,20) (13,7) (14,1) (15,1) (16,0)

a. Use the curve to calculate the numerical values for the two data that are needed to calculate the correct number of integrals for the simulation model.
b. Calculate the number of integrals necessary to mimic the curve.
c. What is the value of the time coefficient of each integral?
d. How should the researcher proceed if he wishes to use half the number of integrals calculated in b) in order to mimic the distribution without changing the mean germination time?

Exercise 93

Water is flowing through a series of 100 well-stirred vessels at a rate, Q, of one litre per minute. The volume, \( V \), of each vessel equals a constant 0.1 litres. Initially, pure water flows through the vessels, but at time \( t_1 \) (=0) this water is replaced by water containing \( 10^6 \) bacteria per litre.
a. Calculate the average residence time of the bacteria in each vessel, and in the system as a whole.

Now assume that each vessel contains an organism consuming the bacteria at a rate proportional to the bacteria concentration, with a maximum consumption rate of \( 10^5 \) bacteria per organism per minute at a concentration of \( 10^6 \) bacteria per litre. This maximum consumption rate is called the bacteria consumption factor, BCNSF.

b. Make a graph of the relation between the bacteria consumption rate per organism (y-axis), and the ratio of the current concentration of the bacteria in the vessel and their maximum concentration.
c. Formulate the rate equation for the first vessel, expressed in the following general symbols:

\[
\begin{align*}
\text{IN} : & \quad \text{rate of inflow of bacteria in the vessel} \\
\text{OUT} : & \quad \text{rate of outflow of bacteria from the vessel} \\
\text{CONS} : & \quad \text{rate of consumption of bacteria by the organism in the vessel}
\end{align*}
\]

\[\text{IN, OUT and CONS are all expressed in (number of bacteria)/(minute) and may be formulated in the following}\]
d. Calculate the ratio between the incoming and outgoing concentration of the bacteria for each vessel after a prolonged period of time.
e. Calculate the ratio of the outgoing concentration of the last vessel and the incoming concentration in the first vessel.
f. Summarize the similarities between the system presented here and that in Exercise 41, Chapter 5.
   - Summarize the similarities between the system presented here and that of Equation 52 of Subsection 9.2.1, in which the diffusion coefficient (D) is set to zero.
g. What time interval would you select if you used the integration method RECT?
h. Write a CSMP simulation program without a subroutine for the 100 vessels in series.
   To this purpose:
   - use the PROCEDURE and DO-loops;
   - explain your choice of the integration routine;
   - only use variables in the program so that all parameters can be given values in the INITIAL segment of the program;
   - Calculate in the TERMINAL part of the program the ratio of the outgoing concentration of the last vessel and the incoming concentration of the first vessel. Print the numerical value of this ratio using a WRITE statement.
   - check your program by making a dimensional analysis.
i. Write a CSMP simulation program using subroutines, in which the rate equations are incorporated for the 100 vessels in series.

Assume that the 100 organisms of this system are transferred to one vessel with a volume of 10 litres. The flow rate of the water through the vessel is still one litre per minute with a concentration of $10^6$ bacteria per litre in the incoming water.

j. Calculate the concentration of bacteria in the outgoing water after a prolonged period.
k. Calculate the ratio between the outgoing and incoming concentration of the large vessel.
   What can you conclude about the feeding efficiency of the organisms in the large vessel compared to that in the series of vessels?

Exercise 94

In recent years, the aphid *Sitobion avenae*, has become one of the most serious pests in cereals. Until flowering (decimal code 60), the aphid population doubles every 4 days. After flowering, the aphids migrate to the ear and there the population doubles every 2.5 days. The relative growth rate of the aphids only depends on the crop development stage.

The crop development rate depends on temperature. At 20 °C, crop development from stage 50 to stage 80 lasts 20 days, and at 7 °C development stops. Assume a linear relationship between crop development rate and temperature. At stage 50 (20 June), there is an average of 0.1 aphids per tiller.
The damage threshold, which is the infestation level causing a loss of more than 250 kg ha\(^{-1}\), is 18 aphids per tiller.

a. Calculate the crop development stage on the 1\(^{st}\), 13\(^{th}\), and 20\(^{th}\) of July, if the average daily temperature is 15 °C.

b. Calculate the aphid population on the same dates.

c. When is the damage threshold reached?

d. Draw a relational diagram for this system.

e. - Draw a graph of:
   a. the crop development rate (y-axis) versus temperature,
   b. the relative growth rate of the aphids (y-axis) versus the crop development stage.

- Calculate the time coefficient and the time step of the model.

f. Write a CSMP simulation program without a subroutine for the aphid population.
   - Write the program in NOSORT and use AFGEN functions for the graphs drawn under e).
   - Write the program so that a sinusoidal temperature wave over the day can be simulated by changing one parameter only.
   - Explain the selected integration method.
   - Use only variables in the program and assign them numerical values in the INITIAL.
   - Perform a dimensional analysis of the program.

- Write a CSMP simulation program using one subroutine for the differential equations of *Sitobion* and the crop development. Note that the AFGEN function is a history function.

Exercise 95

Polluted river water is flowing through 2 lakes in series. The in- and outflow rates of the water are identical. The first lake is very deep and it is assumed that no mixing occurs between the top layer and the lower layer. The pollutant is completely mixed with the water in the top layer of the first lake and in the shallow second lake. In the first lake, both sedimentation of the pollutant from the top layer to the lower layer occurs, as well as its decomposition in the top layer. In the lower layer, no decomposition of pollutant occurs. In the second lake, only decomposition of the pollutant occurs. The sedimentation rate and the decomposition rate in the first lake are proportional to the amount of pollutant present in the top layer. The decomposition rate in the second lake is proportional to the amount of pollutant present in the whole lake. Furthermore, the following data are known:

- \( Q \) : rate of water supply to the first lake = \( 5 \times 10^7 \) m\(^3\) d\(^{-1}\)
- \( CP \) : Concentration of Pollutant in the river water = 20 kg m\(^{-3}\)
- \( SA1 \) : Surface Area of lake one = 25 km\(^2\)
- \( SA2 \) : Surface Area of lake two = 9 km\(^2\)
- \( VT1 \) : Volume of Top layer of lake one = 625 \( 10^7 \) m\(^3\)
- \( VL1 \) : Volume of Lower layer of lake one = 1875 \( 10^7 \) m\(^3\)
- \( V2 \) : Volume of lake two = 90 \( 10^7 \) m\(^3\)
- \( SC \) : Sedimentation Coefficient of the pollutant in the first lake = 0.008 d\(^{-1}\)
- \( RDR1 \) : Relative Decomposition Rate of the pollutant in the top layer of the first lake = 0.03 d\(^{-1}\)
- \( RDR2 \) : Relative Decomposition Rate of the pollutant in the second lake = 0.05 d\(^{-1}\)
- \( MT1 \) : Mass of pollutant in the Top layer of lake one
- \( ML1 \) : Mass of pollutant in the Lower layer of lake one
M2 : Mass of pollutant in lake two : kg
SEDL : SEDimentation Layer : m
DS : Density of the Sediment in the first lake = 1300 kg m\(^{-3}\)

Use the given symbols when answering the questions in this exercise.

a. Draw a relational diagram for the system.
b. Develop the equations that describe the rate of change of the mass of pollutant for the two layers of the first lake and for the second lake.
c. Calculate the equilibrium concentrations of the pollutant that will be reached in due time in the top layer of lake one and in lake two.
d. Calculate the average residence time of a pollutant particle in the top layer of lake one.
e. How much time will be required until a layer of 10 mm of sediment will have settled on the bottom of the first lake, after that the equilibrium state has been reached in the top layer of this lake?
f. Write a simulation program for these lakes.
   - Assume that at time zero, the pollutant begins to flow into the first lake.
   - Let the program also calculate the thickness of the sediment layer, assuming that the pollutant drops immediately to the bottom of the first lake.
   - Let the program stop, when 95% of the equilibrium value for M2 has been reached.
   - Let the program print the concentrations of the pollutant material in both lakes.
   - We wish to use the integration method RKSFX. How would you select the time increment DELT?
g. An analytical solution can also be derived for this system of two lakes in series. For the top layer of lake one this is:

\[ MT1 = IN \cdot TC \cdot (1 - e^{-t/TC}) \]

For lake two the solution is:

\[ M2 = \frac{IN}{TC1} \cdot TC2 \cdot \left\{ \frac{1}{TC1/TC2} \cdot \left[ (TC \cdot e^{-t/TC} - TC2 \cdot e^{-t/TC2}) \right] \right\} \]

where: 
\( IN = Q \cdot CP \)
\( TC = 1/(SC + RDR1 + Q/VT1) \)
\( TC1 = 1/(Q/VT1) \)
\( TC2 = 1/(RDR2 + Q/V2) \).

1. Use these equations to check which equilibrium levels will be reached in the two lakes. Compare your answer with that under c).
2. Use the analytical solution to determine the number of days required until 95% of the equilibrium value for MT1 is reached. Does this correspond to your CSMP output?

**Exercise 96**

A soil scientist desires to obtain an impression about the penetration of frost into the soil. To this purpose he develops a simulation model, enabling him to vary the heat capacity and the heat conductivity of the soil at different depths. A part of the model looks as follows:
VHTC = INTGRL(IVHTC, NETFL, 25)

DO 10 I = 1,25
    TEMP (I) = VHTC(I)/VHCAP(I)
10 CONTINUE

FL(1) = (TMPS-TEMP(1))*COND(1)/0.5
DO 20 I = 2,25
    FL(I) = (TEMP(I-1)-TEMP(I))*AVCOND(I)
20 CONTINUE

FL (26) = 0.0
DO 30 I = 1,25
    NETFL(I) = FL(I) - FL(I+1)
30 CONTINUE

The time is expressed in seconds, the distance in cm, heat in Joules, and temperature in °C. The soil has been divided into layers of 1 cm thickness, and a surface area of 1 cm² is considered.

a. What are the units of the heat content of the layers (VHTC), the heat flow between the layers (FL), the heat conductivity of the soil (COND and AVCOND) and the volumetric heat capacity of the soil (VHCAP), respectively?

b. Why are the thicknesses and surface areas of the layers not included in the equations?
   Give a reason why it is better to include these parameters in the equations.

c. Derive an expression for AVCOND.

d. If the first 5 layers of the simulation model would consist of sand and the remaining layers of clay, how are COND and VHCAP to be initiated in the model based on the following data:

<table>
<thead>
<tr>
<th>Soil</th>
<th>COND</th>
<th>VHCAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>peat</td>
<td>0.25</td>
<td>2.5</td>
</tr>
<tr>
<td>sand</td>
<td>8.8</td>
<td>2.0</td>
</tr>
<tr>
<td>clay</td>
<td>2.9</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Exercise 97
A company produces various kinds of concentrates on one production line and has sufficient storage space. The sales pattern of one of these concentrates is represented in the following figure. The production line is started when the stock, $S$, of this concentrate falls below 500 tons. The production line is stopped when the stock rises above

Progress of the sales rate, SR, as a function of time.
500 tons. Production can be halted instantaneously, but starting up the production always occurs with some delay. The production line can produce a maximum of 300 ton day$^{-1}$ ($P_{\text{max}}$) and the time constant (TC) for the delay in the production line is 5 days. The increase in production per unit of time is described by the following differential equation:

$$\frac{dP}{dt} = \frac{1}{TC} \cdot (P_{\text{max}} - P).$$  \hspace{1cm} \text{Equation 79}

The progress of the stock, $S$, of this concentrate can be calculated by means of:

$$\frac{dS}{dt} = P - SR$$  \hspace{1cm} \text{Equation 80}

The production manager wants to investigate the way the production rate, $P$, and stock progress as a function of time.

a. What is the unit and the meaning of $\frac{dP}{dt}$ in Equation 79?
b. Calculate the production rate 5 days after production has started, if the production would not be halted above a stock level of 500 ton.
c. Write a CSMP simulation program to calculate the stock, $S$, and the production rate, $P$, as a function of time if, besides the above data, it is also known that:
   - the initial stock is 1000 tons on monday morning;
   - the sales pattern is repeated every week as represented in the figure;
   - the production manager wishes to monitor $S$ and $P$ over a period of 8 weeks.
d. How would you incorporate into your model the information that the company has sufficient storage space?

Which function is used to express the influence from the outside world on the model?

What is the name of such a function, and does it generally have one type of dimension?
12 Solutions to the exercises

P.A. Leffelaar

12.1 Introductory remark

Some of the solutions to the exercises are worked out in detail, while for others only short answers without further derivations or comments are given. The detailed solutions will help the novice in some instances and add extra information.

12.2 The answers to the exercises

Exercise 1
a. distance • time⁻¹, time⁻¹, time⁻¹.
b. The terms that are added or subtracted should have the same dimensions or units. The dimensions of the expressions on both sides of the equal sign should be identical. In multiplication and division identical dimensions cancel out. The arguments of exponentials, logarithms and angles should be dimensionless.

Exercise 2
For the relative growth rate, c, to remain constant there should always be a sufficient amount of food available to the organisms, while harmful waste products should be absent or kept at a low level. Moreover, environmental conditions such as temperature and acidity must be kept at a constant level.

Exercise 3
a. The slopes of the different lines represent the time derivatives of the variable on the y axis.
b. (the dimension of the state variable) • time⁻¹.
c. The numerical value of the slope in the graph picturing Equation 4 does not change; it is equal to the constant c. The numerical values of the slopes in the graphs depicting Equations 5 and 6 increase and decrease, respectively, with time.
d. The graphs depicting Equations 1, 2a and 3a give the slopes as a function of time. These time derivatives are constant, increasing and decreasing with time, respectively.

Exercise 4
The results of the numerical integration are: The results of the numerical integration are:

| t | 0 2 4 6 8 10 |
|---|---|---|---|---|---|---|
| W | 0 8 12 14 15 15.5 |
| t | 0 2 4 6 8 10 |
| W | 0 6.30 10.11 12.43 13.83 14.69 |

a. The numerical solution overestimates the value of the state variable compared to the analytical solution, because it is assumed that the rate is constant during the time interval of integration, while in fact it decreases (see graph picturing Equation 3a).
b. When the maximum water level is reached. This is the case after a long time, in principle when \( t \) approaches infinity.

c. The process is twice as slow.

Exercise 5

a. For \( \Delta t = 1 \cdot c^{-1} \) the tank is filled after one time step; no inflow occurs thereafter.

For \( \Delta t = 1.5 \cdot c^{-1} \) the amount of water in the tank oscillates around the equilibrium level, but eventually settles at this level.

For \( \Delta t = 2 \cdot c^{-1} \) an oscillation around the equilibrium level between the limits 0 and \( 2 \cdot W_m \) is obtained.

For \( \Delta t = 2.5 \cdot c^{-1} \) the result is an oscillation with divergence.

b. No answer given.

c. For \( \Delta t \geq c^{-1} \) the results do not reflect reality at all.

d. \( \Delta t \) should be less than \( 0.5 \cdot c^{-1} \). As a rule of thumb \( \Delta t \) should be set at about one tenth of \( c^{-1} \).

Exercise 6

a. The rate equation is given by \( dW/dt = 1/\tau \cdot (W_m - W) \). The change during a time interval equal to \( \tau \), and with a constant rate equals \( \tau \cdot 1/\tau \cdot (W_m - W) = W_m - W \). Thus the difference between the equilibrium value, \( W_m \), and \( W \), at any moment, is eliminated within a single time interval equal to the time coefficient.

b. In a positive feedback loop like exponential growth, the change is directed away from the equilibrium state, so the extrapolation of the tangent must be reversed, and directed towards the unstable equilibrium state; the tangent cuts the horizontal equilibrium line (the x-axis), where the value of the state variable equals zero, one time coefficient interval earlier.

Exercise 7

a. The time coefficients are: 0.67, 5 and 20 year.

b. One-tenth of \( \tau \). However, these data should be rounded off to the nearest smaller value that is an integral fraction of the output interval. For instance, 0.067 will become 0.05, but 0.5 and 2 years may be appropriate for output intervals.

c.1. Analytical: 271.83 after 5 years.

c.2. Numerical: 259.37 after 5 years.

No answer given.

d. During \( \Delta t \) the rate of increase is assumed to be constant, but in fact it increases exponentially, as is shown in the graph picturing Equation 2a. After the first time interval the amount is thus underestimated, and consequently the rate for the next time interval is underestimated, and so on. As time proceeds the discrepancy between the numerical and analytical solution becomes larger. This error propagation will be discussed in Section 6.4.

Exercise 8

Use Equation 5. Then, \( t_2 = \tau \cdot \ln 2 \equiv 0.7 \cdot \tau \). Similarly, the half-live is defined as the time necessary to reach half the original amount. For exponential decrease, \( t_{1/2} \equiv 0.7 \cdot \tau \).

Exercise 9

The question is whether the average residence time equals \( \tau \). This is the case:

\[
\frac{1}{A_0} \int_0^\infty A \cdot dt = \int_0^\infty e^{-t/\tau} \cdot dt = \left[ -\tau \cdot e^{-t/\tau} \right]_0^\infty = \tau.
\]
Exercise 10

It is known that with a fixed sugar concentration the yeast grows exponentially, i.e. as described by Equation 5. Expressing \(c(=\mu)\) explicitly in Equation 5 gives: 
\[
\mu = \frac{\ln(y/y_0)}{t}.
\]
A plot of \(\ln(y/y_0)\) versus \(t\) will give a straight line with slope \(\mu\). Plot the data of Exercise 10 in a graph and calculate \(\mu\) from 
\[
\mu = \frac{\Delta(\ln(y/y_0))}{\Delta t}.
\]

(a) The results should be as follows:

<table>
<thead>
<tr>
<th>Sugar concentration, (c_0)</th>
<th>0</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mu) h(^{-1})</td>
<td>0.0</td>
<td>4.17 (10^{-2})</td>
<td>1.04 (10^{-1})</td>
<td>2.08 (10^{-1})</td>
</tr>
<tr>
<td>(\mu) d(^{-1})</td>
<td>0.0</td>
<td>4.17 (10^{-2})</td>
<td>1.04 (10^{-1})</td>
<td>2.08 (10^{-1})</td>
</tr>
</tbody>
</table>

(b) The general equation for a straight line is 
\[\mu = a \cdot c_s + b.\]
The intercept \(b = 0\) and the slope \(a = \mu_{10}/c_{s10}\) 
thus \(\mu = (\mu_{10}/c_{s10}) \cdot c_s.\)

(c) The concentration of sugar is defined as the weight of sugar per weight of water and is thus proportional to the amount of sugar. The same is true for \(c_{s10}\), which is simply a particular value of the sugar concentration. The equation may therefore be written as 
\[\mu = \mu_{10} \cdot \left(s_0/s_{10}\right).\]

Exercise 11

(a) Use the dimension rules of Exercise 1b.

(b) From dimensional analysis it follows that 1000 has the units kg m\(^{-3}\). Thus 1000 denotes the density of water. Note that this number actually varies between 998.2 and 933.7 kg m\(^{-3}\) for 0 and 10 \% glucose, respectively. (Handbook of Chemistry and Physics, 1974-1975 (D205)).

Exercise 12

Expressing \(t\) explicitly gives 
\[t = -\tau \cdot \ln\left(1 - \frac{s}{s_m}\right),\]
so 
\[t_{95} = -\tau \cdot \ln(0.05) \equiv 3 \cdot \tau.\]

Exercise 13

This time is equal to the volume of the vessel divided by the inflow rate of the sugar solution, which is equal to the time coefficient (Equation 13). Thus 100\% of the equilibrium level is reached after time 
\[t_{100} = \tau.\]

Note: In this case the rate is constant during the whole filling period, therefore the increase of the contents is linear.

Exercise 14

Insert Equation 11 into Equation 10 and set \(dy/dt\) equal to zero to obtain Equation 16. Set \(ds/dt\) in Equation 14 equal to zero, and write \(y\) explicitly to obtain

\[y = \frac{1}{s_y} \cdot \left(c_s \cdot q \cdot 1000 - \frac{x}{t}\right).\]

Multiply the first term within brackets by \(v/v\), use Equation 13 to replace the quotient \(q/v\), and replace \(c_s \cdot v \cdot 1000\) by \(s_m\) to obtain Equation 17.

Exercise 15

(a) 
\[\frac{dy}{d\tau} = \frac{d}{d\tau} \left(1 - \frac{s_m}{s_y} \cdot \left(s_m - \frac{s_{10}}{\mu_{10} \cdot \tau}\right)\right)\]
\[= \frac{s_m}{s_y} \cdot \frac{d}{d\tau} \left(\frac{s_m}{s_y} - \frac{s_{10}}{\mu_{10} \cdot \tau} \cdot \frac{d}{d\tau} \left(\frac{1}{s_0}\right)\right).\]
Apply the rule for differentiating a quotient \(y = u/v\) to each term: 
\[
y' = \frac{u' \cdot v - v' \cdot u}{v^2},
\]
where \(u'\) and \(v'\) stand for the first derivative of the numerator and the denominator, respectively, both with respect to \(x\) (of which \(u\) and \(v\) are functions). Take \(1/(s_y \cdot \tau^2)\) outside the brackets to obtain Equation 18.

b. A maximum or minimum occurs when \(dy/d\tau = 0\). This is the case when \(2 \cdot s_{10}/(\tau \cdot \mu_{10}) = s_m\) or when 
\[
\tau = 2 \cdot s_{10}/(\mu_{10} \cdot s_m).
\]

Exercise 16

a. A summary of the units of the individual terms is: \(s_{10}, s_m, s\): kg of sugar; \(\mu_{10}: \text{d}^{-1}; \tau: \text{d}; s_y: \text{kg of sugar} \cdot (\text{kg of yeast} \cdot \text{d})^{-1}\).

b. The optimum \(\tau\) is given in the answer of Exercise 15b. Substituting that expression in Equation 13, and replacing \(s_m\) by \(c_s \cdot v \cdot 1000\), gives the flow rate: 
\[
q = c_s \cdot v^2 \cdot \mu_{10} \cdot 500/s_{10}.
\]

Exercise 17

The rate of change of \(T\) is 1, so that \(T_{t+\Delta t} = T_t + \Delta t\), and because the initial value of \(T\), \(T_0\), is zero, \(T = \text{time}\).

Exercise 18

From the similar triangles in the following figure,

![Figure](image)

it is directly clear that

\[
\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}
\]

Some algebra then yields Equation 23.

Exercise 19

Parabola through \((0,1), (1,1)\) and \((1.5,0)\):

\[
\begin{align*}
1 &= c \\
1 &= a + b + c \\
0 &= a + 1.5^2 + b + 1.5 + c
\end{align*}
\]

\[
\begin{align*}
a &= -b \\
a &= -4/3 \\
b &= +4/3 \\
c &= 1.
\end{align*}
\]
Therefore, \( y = -(4/3) \cdot x^2 + (4/3) \cdot x + 1 \).

Parabola through \((1.,1.), (1.5,0.)\) and \((3.,0.)\) results in \( y = x^2 - 4.5 \cdot x + 4.5 \)

Exercise 20

a. The results of this first CSMP program are given below to illustrate some aspects of the output.

Note the following:

- The title, input- and output- parameters, timer specification, and the integration method are displayed above the results.
- On the timer specification line also the variables DELMIN and DELMAX appear. These variables are not important as long as the rectangular integration method is used. When these variables are not specified by the user, however, the CSMP program automatically assigns values, namely: \( DELMIN = 10^{-7} \cdot FINTIM; \) \( DELMAX \) is taken equal to the smallest of the specified OUTDEL or PRDEL. The significance of \( DELMIN \) and \( DELMAX \) will be discussed in Chapter 6.
- The maximum and minimum values of the plotted variables is printed, too.

```plaintext
$\$$ $CONTINUOUS$ SYSTEM$ MODELING$ PROGRAM$ III$ V1M3$ EXECUTION$ OUTPUT$ $\$$

TITLE$ Linear$ and$ parabolic$ interpolation
FUNCTION$ YTB = (0.,1.), (1.,1.), (1.5,0.), (3.,0.), (3.5,0.)
OUTPUT$ y1, y2
PAGE$ GROUP, WIDTH = 80
TIMER$ FINTIM = 3.5, OUTDEL = 0.1, DELT = 0.1
METHOD$ RECT
END
VARIABLES$ START$ TIME$ = 0.000000E+00
DELT$ DELMIN$ FINTIM$ PRDEL$ OUTDEL$ DELMAX
0.10000 3.50000E-07 3.5000 0.000000E+00 0.10000 0.10000
$\$$ $SIMULATION$ HALTED$ FOR$ FINISH$ CONDITION$ TIME$ 3.5000
```

165
b. In the interval $0 \leq t \leq 1$ the computer uses the coordinate pairs $(0.,1.), (1.,1.)$ and $(1.5,0.)$. To calculate the parabola in the subsequent intervals one coordinate forward and two coordinates backward are chosen.

The calculation is different in the first interval because only one coordinate backward is known. In the interval $(3.,0.)$ to $(5.,0.)$ the calculation of the parabolae results in a straight line through $(1.5,0.), (3.,0.)$ and $(5.,0.)$. Between $t = 1.5$ and $t = 3$ the NLFGEN creates a loop with negative $y$-values. If, for instance, $y$ would be a relative growth rate which should not be negative, this would introduce a relative death rate in the model. Therefore, the NLFGEN function should be used with caution.
Exercise 21

a. TITLE Integration of a positive feedback loop: exponential growth

INITIAL
INCON IA =1.0
PARAM RGR =0.1
TIMER FINTIM=10.0, OUTDEL=1.0, DELT=1.0
OUTPUT A,GR
PAGE WIDTH =80
METHOD RECT

DYNAMIC
A =INTGRL(IA,GR)
GR =RGR*A

END
STOP
ENDJOB

b. The time coefficient of the model is the inverse of the relative growth rate. The rule of thumb introduced in Chapter 2 leads to a time increment of integration of 1 time unit.

c. Note that in the results of the program given in a) the variables A and GR coincide due to automatic scaling.

Exercise 22

a. TITLE Integration of a negative feedback loop: water tank

INITIAL
INCON IWATER=0.0
PARAM WATMAX=16.0, TC =4.0
TIMER FINTIM=24.0, OUTDEL=2.0, DELT=0.4
OUTPUT WATER,RWATER
PAGE WIDTH =80 ,GROUP
METHOD RECT

* Calculations
WATHLP =0.99*WATMAX

DYNAMIC
WATER =INTGRL(IWATER,RWATER)
RWATER =(WATMAX-WATER)/TC

FINISH WATER=WATHLP

END
STOP
ENDJOB

Note that FINTIM was set twice as large as 3*TC, and that a FINISH statement was used to terminate the program. On a FINISH label two variables may be compared each time an integration has been performed. If the time-dependent variable (here WATER) is now equal to or larger than the auxiliary variable (here WATHLP), the program is terminated. (If the time-dependent variable would decrease rather then increase and would become equal to or smaller than the auxiliary variable, the program would stop, too.) The auxiliary variable need only be calculated once. Therefore, it was placed in the INITIAL segment of the model. No calculations may be carried out on a FINISH label. However, up to ten comparisons may be listed on the label, separated by commas.
Exercise 23

a. TITLE Integration of a forcing function that characterizes the influence of the outside world on the state variable: car

INITIAL
INCON IDIST =0.0
FUNCTION RDISTB=0.0,80.0 , 2.0,80.0 , 3.0,120.0 , 5.0,120.0 , ...
6.0,40.0 , 10.0,40.0
TIMER FINTIM=10.0, OUTDEL=0.5, DELT=0.5
OUTPUT DIST,RDIST
PAGE WIDTH =80
METHOD RECT

DYNAMIC
DIST =INTGRAL(IDIST,RDIST)

RDIST =AFGEN(RDISTB,TIME)

END
METHOD TRAPZ
END
STOP
ENDJOB

Note that the CSMP system variable TIME may be used to read the arbitrary function generator.

b. The size of the integration period to properly read the rate function should be chosen so that readings are at least obtained at the moments where the function 'changes direction', i.e. at times 2, 3, 5 and 6. This would result in a Δt of 1 h. The function is not read very nicely with Δt equal to 1 h, however, because the rectangular integration method then neglects the gradual change in the rate between times 2 and 3 and 5 and 6: it will jump from 80 to 120 at time 3 and from 120 to 40 at time 6.

To better estimate the size of the integration period we could require that at the beginning of each new integration interval the rate should not be changed more than 5% from the preceding rate. The strongest rate of change of the rate should be the starting point for the calculation of Δt. This lies between time 5 and 6. The equation of the rate as a function of time in that interval can be calculated as:

RDIST = -80 * TIME + 520.

At time 5 the rate equals 120 km h⁻¹, and 5% of this value is 6 km h⁻¹. Thus a new integration interval should start at a rate of 114 km h⁻¹. The integration interval may be calculated from the equation as 0.075 h. For practical purposes 0.05 would be chosen. In the program DELT=0.5 was chosen to show the large difference between the integration methods RECT and TRAPZ.

Note that the function is always overestimated in case it decreases, and that it is underestimated in case of an increase like in the time interval of 2 to 3.

You are now familiar with two ways of estimating Δt:
1. on the basis of the time coefficients in the model,
2. on the basis of properly reading model functions.

c. We would calculate the surface areas below the straight line sections of the graph. Each section can be described by the general equation y = a * x + b, in which the values of the coefficients are different each time. Manual calculation of the path travelled yields: DIST = 2 (hour) * 80 (km hour⁻¹) + 1 * (80+120)/2 + 2 * 120 + 1 * (120+40)/2 + 4 * 40 = 740 km.

This method is called the trapezoidal integration method (see Chapter 6 for further details).
Exercise 24

a. TITLE Continuous culture of yeast growing on sugar
INITIAL
INCON IYEAST=0.15, ISUGAR=12.0
* kg Y kg S
PARAM M10 =5.0, CSUG =0.1, Q =300.E-3, RHOWAT=1000.0
* kg S m**3 H2O kg H2O
* kg H2O day
PARAM SUGYST=0.2, CSUG10=0.1, VOL=120.E-3
* kg S kg S m**3
* kg H2O
TIMER FINTIM=7.5, OUTDEL=0.5, DELT=0.02
* day day day
PAGE WIDTH =80
OUTPUT YEAST ,SUGAR
OUTPUT DYSTDT,DSUGDT
METHOD RECT

* Calculations
SUG10 =CSUG10*VOL*RHOWAT
TAU =VOL/Q

DYNAMIC
YEAST =INTGRL(IYEAST,DYSTDT)
SUGAR =INTGRL(ISUGAR,DSUGDT)

DYSTDT =MU*YEAST - YEAST/TAU
MU =SUGAR/SUG10*M10
DSUGDT =CSUG*Q*RHOWAT - YEAST*SUGYST - SUGAR/TAU

END
STOP
ENDJOB

Note the following:
- Initially the amount of yeast (YEAST) has been set to 1/10th of the equilibrium that would be attained at the parameter values used (calculated from Equation 17, Section 2.6). The amount of sugar (SUGAR) in the reservoir has been set to its equilibrium value when yeast would be absent and when the sugar solution would already have flowed through the reservoir for a long time (calculated from Equation 15 (sm), Section 2.6).
- Only variables and not numbers are present in the DYNAMIC segment. This way of programming increases the program's legibility and minimizes the chance of error when parameter values are changed, since all the parameters are grouped together in the INITIAL segment and can be easily overseen.
- Comment lines giving the units of the parameters have been added in the INITIAL segment.

b. To calculate the value of \Delta t for the rectangular integration method one should know all the time coefficients in the model. These are: 1) the inverse of the maximum relative growth rate MU, 2) the average residence time TAU of the yeast in the vessel, and 3) the average residence time of the sugar in the vessel.

The numerical values of these coefficients can be calculated as follows: 1) 1/MU10=0.2 d; 2) TAU=0.4 d; 3) the residence time of the sugar is affected by the water flow through the vessel, with a residence time TAU, and by the consumption rate of the sugar by the yeast. The amount of yeast and sugar change in time. The maximum amount of yeast, and thus the maximum consumption rate of sugar through the term (YEAST*SUGYST), will be accompanied by the minimum amount of sugar. Therefore, assume that the equilibrium values of yeast and
sugar, as may be calculated from Equations 16 and 17, are at the same time the maximum and minimum values reached. Then, the average residence time is given by the ratio of the amount of sugar and the sum of the outflow rates:

\[
\frac{\text{SUGAR}}{\Sigma (\text{YEAST} \cdot \text{SUGYST} + \text{SUGAR}/\text{TAU})} = 0.2 \text{ d},
\]

which is half as large as the value based on TAU alone.

(These values are equilibrium values we could have taken the ratio of SUGAR/(CSUG \cdot Q \cdot \text{RHOWAT}).)

The value of \(\Delta t\) for the rectangular integration method can be calculated as \(\Delta t = 1/10 \cdot \) (smallest time coefficient) = 0.02 d.

If this calculation procedure would not lead to satisfactory results, one could resort to the so-called trial-and-error method to estimate the time increment of integration. This method will now be explained.

Basically, the choice of the time increment should not significantly affect the outcome of a model. Therefore, just choose a time increment, for instance derived from MU10 and TAU, and make a run with the model. Then, half this time increment and investigate if this significantly affects the model results by running the model anew. If model results do not significantly change, double the first choice and see how this affects the results. If this does not affect the results, \(\Delta t\) can be doubled again; if it affects the results, the first choice was correct.

With this method the time coefficient, \(\tau\), is not explicitly determined, but an appropriate value of \(\Delta t\) can be found. Note that in the rectangular integration method \(\Delta t\) is fixed, while the \(\tau\) in the model varies in time. This implies that the accuracy of the calculations in the model also varies, because this will be a function of the ratio \(\Delta t/\tau\). It would be more appropriate when the accuracy of the calculations would be constant, by adapting the time increment of integration. Integration methods that achieve this are dealt with in Chapter 6.

Of course, the average residence time could be calculated in the simulation program, and inspected to get a thorough feeling on how it changes during the simulation and on how it is related to our choice of the time increment for integration. The formula to calculate the average residence time was given above as the ratio of the amount of sugar and the sum of the outflow rates.

c. The state variables will no longer change in the equilibrium state. This is indeed the case and the equilibrium state as calculated from Equations 16 and 17 of Chapter 2 is reached. This calculation is a check on the program's structure.

Note that the state variables yeast and sugar oscillate before arriving at their equilibrium values.

Exercise 25

a. TITLE Exponential growth with temperature dependent relative growth rate

INITIAL
INCON IA =1.0
PARAM AVTMP =20.0, AMPTMP=10.
FUNCTION RGRTB =0.0,0.0 , 10.,0.08 , 20.,0.16 , 30.,0.21 , ...
40.,0.24 , 50.,0.25
TIMER FINTIM=48.0, OUTDEL=2.0, DELT=0.4
OUTPUT A, RGR, GR
PAGE WIDTH =80
METHOD RECT

* Calculations
PI =4.0*ATAN(1.0)

DYNAMIC
A =INTGRL(IA,GR)
GR =RGR*A
RGR =AFGEN(RGRTB,TEMP)
TEMP =AVTMP + AMPTMP*SIN((2.0*PI/24.0)*TIME)

END
STOP
END JOB
Note that the value of π (π) can be calculated as $4.0 \cdot \text{ATAN}(1.0)$, since $\tan(\pi/4) = 1$. Therefore, $\text{arctan} \ 1 = \pi/4$.

b. For the given values of AVTMP and AMPTMP, the minimum time coefficient of this model is attained at 30 °C and will be 4.76 hours, and the maximum time coefficient is attained at 10 °C and will be 12.50 hours.

c. The argument of the sinus function is dimensionless, as it should be. (See also answer to Exercise 1 from Chapter 2.)

d. The time interval for rectangular integration should be 0.47 hours, which may be rounded to 0.5.

This time interval reasonably follows the sine function. But the time coefficient of a sine can, in fact, be calculated as the inverse of the angular frequency. So, $\tau = 1/\omega = t_c / 2\pi = 3.82 \text{ h}$. This would yield a time step of integration of 0.38 or 0.4 h.

e. No answer is given.

Exercise 26

It was said that two rates do not depend on each other, but not that one rate cannot depend on the other. Here, the rate of growth and the rate of alcohol production are consequences of the same process: the biosynthesis of yeast material out of sugar. Therefore, there is a fixed ratio between rate of growth and rate of alcohol production. The rate of sugar consumption is stochiometrically related to the above two rates: laws of conservation of matter, energy etc. can be formulated in such a way that some rate of appearance always equals some rate of disappearance.

Exercise 27

The rate of sugar consumption is equal to a sugar consumption factor times the rate of yeast growth for each species. The amount of sugar is an integral which is emptied by both rates. The amount or concentration of the sugar in the medium should feed back on the growth rate of the yeasts. The quantitative aspects of this feedback are not presented in the diagram.

A relational diagram for the growth and interference of two interfering yeast species incorporating the aspect of limited food supply.
Exercise 28

Two important differences between the models are:

1. In the yeast culture fed by a sugar solution, sugar is consumed proportional to the amount of yeast. This indicates that only the costs of maintenance of yeast cells is accounted for, but not the costs of growth of the yeast cells. In the case of the growth and interference of two yeast species the reverse is true: alcohol production is due to the growth of the two species, but not due to maintenance processes.

   In reality both growth and maintenance processes will occur.

2. The yeast culture fed by a sugar solution is a continuous culture (Figure 6), whereas the growth and interference of two yeast species (Figure 8) is a batch culture.

   In both cases an equilibrium is reached where the state variables do not change anymore. A dynamic equilibrium where the net flow rates are zero in the case of the continuous yeast culture fed by a sugar solution; a static equilibrium where the individual flow rates are zero in the case of growth and interference of two yeast species.

Exercise 29

RED1 = ALC/MALC, as here 0 ≤ ALC ≤ MALC.

Otherwise, RED1 should be given by RED1 = LIMIT(0., 1., ALC/MALC).

Exercise 30

a. A first estimate of RGR1 is obtained by presenting the amount of yeast during early growth on a logarithmic scale against time and drawing a straight line through the data. The value is about 0.2 h⁻¹. Application of the rule of thumb yields a time step of 0.5 h.

b. The value of ALPF1 is the alcohol concentration at the end, divided by the amount of yeast newly grown or

\[ \frac{1.5}{(13. - 0.45)} = 0.12 \text{ percentage of alcohol per unit of yeast.} \]

c. The alcohol production factor depends on the size of the vessel. In a larger vessel, the same amount of alcohol would cause a smaller percentage. It would be more elegant not to mix up the influence of physiological aspects (alcohol production rates) with experimental aspects (vessel size), but Gause did not give the latter.

d. The alcohol percentage corresponding with the initial amount of yeast is ALPF1 * IY1, but Gause did not add this alcohol with the yeast at time zero.

e. Relevant figures for Schizosaccharomyces are: RGR2 = 0.05 h⁻¹, ALPF2 = 0.26 (% alcohol) * (unit yeast)⁻¹.

f. Schizosaccharomyces has the largest alcohol production factor.

Exercise 31

a. Here Saccharomyces would grow more slowly and Schizosaccharomyces faster than suggested by a linear dependency of the reduction factor on the alcohol concentration. In the monocultures, this deviation from the linear dependency would not affect the ultimate amount of yeast that is formed, but in the mixture it would lead to less Saccharomyces and more Schizosaccharomyces. However, the growth curves for the two species in the monoculture would also be of different form.

b. In fact, it is assumed that the differences between the simulated and experimental values in the course of time for the growth of the yeasts in Figure 8 in Section 4.1 would be statistically significant, and in order to achieve better agreement between the simulated and the experimental values an incorrect method was applied, namely by varying the parameters in the functions to the model of the mixed cultures so that better agreement is achieved. By following this method, the model degenerates into a descriptive dynamic model, since it no longer explains why growth proceeds as in the experiment: consequently, the model is used as a method for 'curve fitting'. The parameters of the model lose (partly) their physical or physiological significance, and, moreover, the simulation
of the monocultures will become worse.

The correct method is that of methodical improvement, the so-called heuristic method:
- Since differences are statistically significant, it is concluded that the simulation model is not (very) accurate.
- Experiment with both the model and the system to determine which aspects of the model are uncertain or are probably missing completely. For example, there might be a production of other growth-inhibiting materials than alcohol, or growth-stimulating materials by the yeasts.
- Subsequently study these aspects at an explanatory level: use experiments, for example, to determine the effect of an inhibitor besides alcohol (the explanatory level) on the growth (the level to be explained).
- Incorporate these new aspects into the model and compare again the results of the model with the experimental data.

**Exercise 32**

\[
\text{ALC} = \text{INTGRL} \left( \text{IALC} \cdot \text{ALCP1} + \text{ALCP2} - \frac{\text{ALC}}{\text{TAU}} \right)
\]

where TAU could be called the washing constant or the relative dilution rate. It is expressed in hours and represents the average residence time of the alcohol in the vials with yeast. Note that the differential equation for the outflow is similar to the one for exponential decrease.

**Exercise 33**

a. A repeatable system.

b. A descriptive static model.
   - Descriptive, because only the existence of a relation between the reduction of growth rate and alcohol percentage is indicated; the reasons for the reduction are not explained.
   - Static, because the model does not contain time.

c. The yeast growth model 'Mixed culture of yeast' can be called an 'explanatory dynamic model':
   - Explanatory, for the growth of yeast is explained with the knowledge about the relationship between the growth rate of yeast and the alcohol percentage. Therefore, the minimum of two integration levels is present: the level to be explained is that of the growth of yeast, the explanatory level is the relationship mentioned previously.
   - Dynamic, because the model includes time.
   - Moreover, the model can be verified with experiments.

**Exercise 34**

a. \( Y_m \) equals \( \text{MALC/ALPF} \).

b. Since ALPF was calculated from \( Y_m \), it is not surprising that the \( Y_m \) equals 13 and 5.8 for the species.

c. The first derivative of \( c/v \) equals \((- c/v^2) \cdot (dv/dt)\), when \( c \) is a constant, so that the first derivative of Equation 30 is:

\[
\frac{dY}{dt} = \frac{-Y_m}{(1 + K \cdot e^{-RGR \cdot t})^2} \cdot (-K \cdot RGR \cdot e^{-RGR \cdot t})
\]

The two minus signs cancel, and part of the expression can be replaced by \( Y \) itself:

\[
\frac{dY}{dt} = Y \cdot \frac{K \cdot RGR \cdot e^{-RGR \cdot t}}{(1 + K \cdot e^{-RGR \cdot t})}
\]
The fraction \( \frac{K \cdot e^{-RGR \cdot t}}{1 + K \cdot e^{-RGR \cdot t}} \) can also be written as \( 1 - \frac{1}{1 + K \cdot e^{-RGR \cdot t}} \).

Substituting \( Y \) for a second time gives:

\[
\frac{dY}{dt} = RGR \cdot Y \cdot (1 - \frac{Y}{Y_m}).
\]

In this way the differential equation (Equation 29) is again arrived at.

d. The initial amount of yeast can be found by substituting for time the value zero into the integrated Equation 30. This gives:

\[
IY = \frac{Y_m}{1 + K}
\]

e. See the results of your program.

f. The differential equation for the rate of alcohol production can only be replaced by the integral equation for the amount of alcohol if the initial amounts of yeast are small. Otherwise, the appropriate amount of alcohol, \( ALPF \cdot IY \), has to be added together with the initial amount of yeast if the analytical solution is to be used. Such restrictions do not hold in the simulation program because no equations are eliminated there.

**Exercise 35**

a. If we again neglect the initial amounts of yeast, the amount of alcohol in the mixed culture is given by

\[
ALC = ALPF1 \cdot Y1 + ALPF2 \cdot Y2
\]

Assume that RED = ALC/MALC for both species, then the growth rates may be formulated as

\[
\frac{dY1}{dt} = Y1 \cdot RGR1 \cdot (1 - \frac{ALPF1 \cdot Y1}{MALC} - \frac{ALPF2 \cdot Y2}{MALC})
\]

\[
\frac{dY2}{dt} = Y2 \cdot RGR2 \cdot (1 - \frac{ALPF1 \cdot Y1}{MALC} - \frac{ALPF2 \cdot Y2}{MALC})
\]

b. \( R1 \) and \( R2 \) correspond to \( RGR1 \) and \( RGR2 \), \( A1 \) and \( A2 \) are equal to \( ALPF1/MALC \) and \( B1 \) and \( B2 \) to \( ALPF2/MALC \).

c. \( A1 \) equals \( A2 \) and \( B1 \) equals \( B2 \) because it is assumed that \( Y1 \) and \( Y2 \) are equally sensitive to the alcohol produced by \( Y1 \) and/or \( Y2 \).

d. If \( Y1 \) produces some product that is more harmful for \( Y2 \) than for \( Y1 \), \( A2 \) is larger than \( A1 \).

**Exercise 36**

a. An equilibrium exists where both lines cross each other, since the rates of both species are zero there. The question now is whether a disturbance or perturbation of this equilibrium will induce an action to return to the equilibrium (stable) or that this action is directed away from the equilibrium (unstable). We can investigate this by choosing a point in the graph and calculating the sign of the reduction factor \( (1 - A1 \cdot Y1 - B1 \cdot Y2) \) there. Then we can easily see if the resultant will go towards the equilibrium or not.
In Figure 11, a disturbance of the equilibrium will induce an action to return to the equilibrium. So the equilibrium is stable.

b. To go from this stable equilibrium to an unstable one, the lines of the species should be reversed.

c. The ecological basis of a stable equilibrium is niche-differentiation. When the niches are slightly different, the species do not compete exactly for the same resources. Species 2 may then still be growing even when the population density of species 1 is above its equilibrium and declines.

An unstable equilibrium between two competing species means that an initial advantage of one species will be amplified. The final outcome of the competition depends strongly on having a favourable starting position. For this situation the species must compete for the same resources, and individuals of the same species must be less harmful for each other than for the competitors, or even promote each other. Social behaviour in animals may lead to such a situation.

A much clearer example comes from physiology. Growing fruits or pods on one individual plant compete for assimilates and nutrients. An important characteristic of these growing fruits is that they amplify their own sink strength by producing hormones. Fruits that are too small, for instance when set a later stage, appear not to pass a threshold size needed to get their growth off the ground. These fruits do not survive and are aborted. This mechanism makes sure that the plant does not overload its capacity, and can deliver at least a number of viable seeds.

**Exercise 37**

a. Since $B_m$ and $B$ have the unit of gram and since a rate is found to the left of the '=' symbol, the unit of $c$ must be (time)$^{-1}$. $c$ might be called the relative formation rate of $B$.

b. The rule of thumb to determine $\Delta t$ ($\Delta t = 1/10 \cdot \tau$) for the rectangular integration method (method RECT) will result in a time increment of 1 time unit. Integration according to the rectangular integration method will result in $B = 1.9$ g after 2 time units.

c. Since the differential equation is similar to the one for the water tank that is filled by a flow of water through an adjustable valve from Section 2.2, it can be calculated from Equation 6, Section 2.4, that after 2 units of time an amount $B$ of 1.8127 gram is formed. This means that the rectangular method overestimates the analytical method by less than 5 %, which is acceptable.

**Exercise 38**

a. $H_1 = FL_1 \cdot \left( \frac{1}{1/TC_{12} + 1/TC_{13}} \right)$, $H_2 = FL_1 \cdot \left( \frac{1}{1/TC_{12} + 1/TC_{13}} \right) \cdot TC_{23}/TC_{12}$, $H_3 = FL_1 \cdot TC_4$.

b. The average residence time, or time coefficient, for the material in integral $H_1$ is affected by $FL_{12}$ and $FL_{13}$. The associated time coefficient can be calculated by equating the sum of the present outflows from integral $H_1$ to a new outflow that is defined as $H_1 / TC_{NEW}$:

$$FL_{12} + FL_{13} = H_1/TC_{12} + H_1/TC_{13} = H_1 \cdot (1/TC_{12} + 1/TC_{13}) = H_1 / TC_{NEW}$$

from which it follows that $TC_{NEW} = \frac{TC_{12} \cdot TC_{13}}{TC_{12} + TC_{13}}$

If one would found the time increment of integration on $TC_{12}$ or $TC_{13}$ alone, this would lead to an overestimation of $\Delta t$, since $TC_{NEW}$ is smaller than either of them. This can easily be seen by rewriting
Exercise 39

a. 4.159 month; 2.773 yr

b. 

![Diagram of growth and death cycle]

$$\frac{TC_{12} \cdot TC_{13}}{TC_{12} + TC_{13}}$$ as $$\frac{TC_{13}}{TC_{12} + TC_{13}}$$ and $$\frac{TC_{12}}{TC_{12} + TC_{13}}$$

Exercise 40

a. \( W = \text{IW} \equiv \text{amount of dry matter in kg ha}^{-1} \)

b. \( W \) and \( Y \)

c. no. IV

d. \( N = 20 \) (because \( C = 0 \))

e. \( Y = 64 \) (net rate in \( \text{INTGRL} \) \( Y \) equals zero)

\[ A \equiv \text{amount of nitrogen in dry matter in kg kg}^{-1} \]

\[ Y = IY \equiv \text{amount of nitrogen in kg ha}^{-1} \]
Exercise 41

a. The processes represented by the three terms in the differential equation are:

- $W/V$ is the rate of pollution of the lake in kg d$^{-1}$ m$^{-3}$,
- $c/	au_0$ is the rate of discharge of the pollutant from the lake by the water flow in kg m$^{-3}$ d$^{-1}$,
- $k \cdot c$ is the rate of decomposition of the pollutant in the lake in d$^{-1}$ kg m$^{-3}$.

b. The relational diagram for the pollutant in the lake is pictured as follows:

```
1/\tau_0

---

\text{discharge of pollutant via the river}

C

\text{decrease of pollutant material via decomposition}

---

W/V
```

c. The average residence time of a water molecule in the lake is $\tau_0 = V/Q$ d.

d. The average residence time of the pollutant in the lake will be shorter than the average residence time of a water molecule: both are discharged by the river at the same rate, but the pollutant is also decomposed biologically and/or chemically.

e. The average period the pollutant is present in the lake is $V/(Q + V \cdot k)$ days. This can be understood as follows. The pollutant disappears from the lake by transport with water and by decomposition. Both rates are proportional to the concentration in the lake and therefore the overall decrease is an exponential process. The total rate of decrease of the pollutant in the lake is $c \cdot ((1/\tau_0) + k)$, with $((1/\tau_0) + k)$ the relative rate of decrease. The time coefficient, or average residence time, is its inverse. Substituting the expression for $\tau_0$ and multiplying both numerator and denominator by $V$, results in the given solution. The numerical value of $V/(Q + V \cdot k)$ is indeed smaller than that of $V/Q$, as explained qualitatively in the solution to d.

f. Equilibrium implies that the rates of supply and discharge are equal. Therefore $dc/dt = 0$ and:

\[
\frac{W}{V} = \frac{c}{\tau_0} + k \cdot c = c \cdot \left( \frac{1}{\tau_0} + k \right)
\]

Substitution and rearrangement yields for the concentration:

\[
c = \frac{W}{Q + V \cdot k}
\]

g. The numerical value of the time constant of the differential equation is $1/((1/\tau_0) + k) = 17.24$ d.

h. The time step of integration routine for this program would be 1.724 d. A practical value is than 1.0 d, which results actually in a more accurate solution to the problem.

The CSMP simulation program for this model is:

```
TITLE Pollution of a lake by city waste water.
INITIAL
INCON IC = 0.0
PARAM V = 1.25E9, Q = 1.0E7, W = 6.4, K = 0.05
```
* Calculations

\[ T_0 = \frac{V}{Q} \]
\[ C_{MAX} = \frac{(W/(Q+V*K))}{C/\text{TO} - K*C} \]
\[ C_{MAXH} = 0.98 \times C_{MAX} \]

**DYNAMIC**

\[ C = \text{INTGRL}(IC, RC) \]
\[ RC = W/V - C/T0 - K*C \]

**FINISH**

\[ C = C_{MAXH} \]

END
STOP
ENDJOB

**Exercise 42**

a. 41.298 days.

b. **TITLE** Growth!of!rust

**INITIAL**

\[ \text{INCON} \; IA = 0.5E-6 \]
\[ \text{PARAM} \; RGR = 0.4, \; AM = 3.2 \]
\[ \text{TIMER} \; FINTIM = 50, \; \text{PRDEL} = 5, \; \text{DELT} = 0.25 \]

**PRINT** A

**METHOD** RECT

**DYNAMIC**

\[ A = \text{INTGRL}(IA, GR) \]
\[ GR = RGR \times A \times (1.0 - A/AM) \]

END
STOP
ENDJOB

c. **TITLE** Growth of rust with decreasing leaf area

**INITIAL**

\[ \text{INCON} \; IA = 0.5E-6, \; ILAI = 4. \]
\[ \text{PARAM} \; RGR = 0.4, \; RDR = 0.012 \]
\[ \text{TIMER} \; FINTIM = 50, \; \text{PRDEL} = 5, \; \text{DELT} = 0.25 \]

**PRINT** A, LAI

**METHOD** RECT

**DYNAMIC**

\[ A = \text{INTGRL}(IA, GR) \]
\[ LAI = \text{INTGRL}(ILAI, -RDR \times LAI) \]

\[ GR = RGR \times A \times (1.0 - A/AM) - RDR \times A \]
\[ AM = 0.8 \times LAI \]

END
STOP
ENDJOB
The reduction in leaf area directly affects the growth rate of the infection, as expressed by \((- RDR \cdot A\) in the differential equation for GR. It is thus assumed that the rust lesions are homogeneously distributed over the available leaf area. The indirect effect of a decreasing leaf area is through the decrease of the maximum rust infection that can be reached (AM).

**Exercise 43**

a. \(Y = \frac{Y_m}{1 + K \cdot e^{-RGR \cdot t}}\) with \(K = \frac{Y_m}{Y_0} - 1\).

b. From \(Y_m = 250\) and \(K = 24\), it can be calculated that the time in which \(2/3\) of the maximum population size is attained is 6.452 days.

c. \(TC\)-begin = \(RGR^{-1} = 1.667\) days, because hardly any inhibition of growth takes place yet.

The (effective) relative growth rate of the system, \(RGR_{system}\), can be described by

\[
RGR_{system} = RGR \cdot (1 - \frac{Y}{Y_m})
\]

At \(t = 0\): \(Y = 0\), so \(RGR_{system} = RGR = \frac{1}{TC} \cdot \frac{1}{RGR}\)

When \(Y = 1/2 \cdot Y_m\), \(RGR_{system} = RGR \cdot 1/2 = \frac{1}{2TC}\), so \(TC = \frac{2}{RGR}\).

It is not correct to think that the time coefficient continues to increase. When \(Y\) is approximately equal to \(Y_m\), the differential equation

\[
\frac{dY}{dt} = RGR \cdot Y \cdot (1 - \frac{Y}{Y_m})
\]

can be approximated by

\[
\frac{dY}{dt} = RGR \cdot Y_m \cdot (1 - \frac{Y}{Y_m}) = RGR \cdot (Y_m - Y)\] or

\[
\frac{d(Y - Y_m)}{dt} = -RGR \cdot (Y - Y_m).
\]

This differential equation contains a negative feedback with a \(TC\) of \(1/RGR\), and may be compared with the differential equation for the water tank that is filled by a flow of water through an adjustable valve from Section 2.2.

**Exercise 44**

a. The curve can be described by the integrated form of the equation for logistic growth:

\[
H = \frac{H_m}{1 + K \cdot e^{-RGR \cdot t}}
\]

b. The differential equation for logistic growth is:

\[
\frac{dH}{dt} = RGR \cdot H \cdot (1 - H/H_m)\] or \(\frac{dH}{dt} = RGR \cdot H - (RGR/H_m) \cdot H^2\).
The equation for the prey is: \( \frac{dH}{dt} = a \cdot H - b \cdot H^2 \). Therefore, constants \( a \) and \( b \) can be written as: \( a = RGR \) and \( b = RGR/H_m \).

c. Calculation of constants \( a \) and \( b \).
- Rewriting the integrated form of the logistic growth equation for \( a = RGR \) gives \( \ln(H_m/H - 1) = \ln(K) - a \cdot t \). Therefore, when \( (H_m/H - 1) \) is plotted against time on half logarithmic paper, a straight line with slope \( a \) will be found.
- We read from the figure that \( H_m = 3.0 \). Furthermore:

<table>
<thead>
<tr>
<th>( t )</th>
<th>( H ) (number m(^{-2}))</th>
<th>and therefore ( (H_m/H - 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10 )</td>
<td>0.26</td>
<td>10.54</td>
</tr>
<tr>
<td>( 20 )</td>
<td>1.05</td>
<td>1.86</td>
</tr>
<tr>
<td>( 30 )</td>
<td>2.26</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Now draw the graph. Slope \( a \) is calculated from the decline of \( (H_m/H - 1) \) in time:

\[
a = \frac{\ln(10.54) - \ln(0.33)}{20} = 0.1732 \text{ d}^{-1}.
\]

- Now \( b \) can be calculated:

\[
b = \frac{0.1732}{3} = 0.0577 \text{ m}^2 \text{ d}^{-1} \text{ (number } H)^{-1}.
\]

d. \( H_0 \) can be calculated from the intercept on the y-axis of the half logarithmic graph, where \( t=0 \).
- We read: \( K = 60 = (H_m - H_0) / H_0 \), and \( H_0 = 0.0492 \) is calculated.
- This value compares well to the one found from the \( H-t \) curve: 0.05. (More accurate reading of \( K \) would yield a value of 59.)

e. The equation now reads: \( dP/dt = -e \cdot P \), so exponential death. The relative rate of decrease is: \( - (1/6 \cdot \ln 0.25) = 0.2310 \).

f.
g. \( a = e = d^{-1} \)
\( b = d = m^2 d^{-1} \) (number \( H \))
\( c = m^2 d^{-1} \) (number \( P \))
h. The time coefficient can not simply be calculated by applying the rule of thumb for rectangular integration to the inverse of the relative rates \( a \) and \( e \). Firstly, the problem includes a logistic growth equation in which the time coefficient varies in time (see Exercise 43). The associated time coefficient should be calculated from \( a \). Secondly, the predator population may decrease with a relative death rate of \( e \). Thirdly, there is interaction between prey and predator. The associated time coefficients are the inverses of \((c \cdot P)\) and \((d \cdot H)\) with respect to the prey and predator population, respectively. Since the state variables \( H \) and \( P \) vary in time, the time coefficients vary too. Though the prey-predator model may look very simple, the interaction terms make it too complex to explicitly calculate all time coefficients. What remains is the trial-and-error method to get an impression of the order of magnitude of \( \Delta t \). This method was explained in the answer to Exercise 24b.

i. **TITLE Prey predator Lotka Volterra**

**INITIAL**

\( \text{INCON } H = 0.5 \), \( IP = 0.5 \)

**PARAM**

\( A = 0.1732 \), \( B = 0.0577 \), \( C = 0.0867 \), \( D = 0.1540 \), \( E = 0.2310 \)

**TIMER**

\( \text{FINITM} = 100. \), \( \text{OUTDEL} = 1.0 \), \( \text{DELT} = 0.25 \)

**PAGE WIDTH**

80

**OUTPUT**

\( H, P \)

**METHOD RECT**

**DYNAMIC**

\( H = \text{INTGRL}(IH, RH) \)
\( P = \text{INTGRL}(IP, RP) \)

\[ \begin{align*}
RH &= (A - B \cdot H) \cdot H - C \cdot H \cdot P \\
RP &= -E \cdot P + D \cdot H \cdot P
\end{align*} \]

**END**

STOP

ENDJOB

The \( \text{DELT} \) in this program was estimated by the trial-and-error method. Deviations of not more than 5% were accepted with respect to the minimum and maximum in \( H \) and \( P \) that occurred over a period of 100 days. These minima and maxima were for \( H \): 0.5, 2.303, and for \( P \): 0.178, 1.412, respectively. The associated time coefficients calculated from the inverses of \( a, e, (c \cdot P) \) and \((d \cdot H)\) yield 5.8, 4.3, 8.2 and 2.8 days, respectively. From these data \( \Delta t = 0.28 \) would result, which confirms the practical value of 0.25 d.

In Section 6.3 mathematical analysis will show that the rule of thumb to estimate \( \Delta t \) for solving first order differential equations introduces an error of about 5%.

j. - Calculation of equilibrium lines.

Equilibrium means \( dH/dt = 0 \) and \( dP/dt = 0 \).

For the prey \((dH/dt = 0.)\): \( (a - b \cdot H) \cdot H = c \cdot H \cdot P \rightarrow a - b \cdot H = c \cdot P \). If \( H = 0 \), then \( P = a/c = 2 \). If \( P = 0 \), then \( H = a/b = 3 \). This results in the equilibrium line \((y = a \cdot x + b)\):

\[ P = -\left(\frac{a}{a/b}\right) \cdot H + \frac{a}{c} \]

For the predator \((dP/dt = 0.)\): \( e \cdot P = d \cdot H \cdot P \rightarrow e = d \cdot H \). This results in an equilibrium line parallel to predator \( y \)-axis:
\[ H = e/d = 1.5. \]

- Calculation of densities after a longer period.

Substituting Equation 82 into Equation 81 yields:

\[ P = - \left( \frac{a/c}{a/b} \right) \cdot \frac{e}{d} \cdot P = 1.0. \]

And substitution of 1.0 in Equation 81 yields \( H \):

\[ H = (1 - \frac{d}{c}) \cdot (-\frac{a/b}{a/c}) = 1.5. \]

After a long period the densities will thus be: (prey \( H \), predator \( P \)) = (1.5, 1.0).

- Investigation where rates are positive or negative in relation to the equilibrium lines.

Consider the area to the left of the line \( \frac{dP}{dt} = 0 \).

The differential equation for \( P \) is:

\[ \frac{dP}{dt} = -e \cdot P + d \cdot H \cdot P. \]

Take e.g. \( H = 1/2 \cdot (e/d) \) and substitute this in the equation: \( \frac{dP}{dt} = -e \cdot P + 1/2 \cdot (e/d) \cdot d \cdot P = P \cdot (-1/2 \cdot e) \), which is negative. So, to the left of the line \( \frac{dP}{dt} = 0 \), the rate of \( P \) is negative (and to the right of the line the rate of \( P \) is positive).

Consider the area below the line \( \frac{dH}{dt} = 0 \).

The differential equation for \( H \) is:

\[ \frac{dH}{dt} = (a - b \cdot H - c \cdot H \cdot P. \]

Take e.g. \( H = 1/2 \cdot (a/b) \) and substitute this in the equation: \( \frac{dH}{dt} = (a - b \cdot 1/2 \cdot a/b) - 1/2 \cdot (a/b) - c \cdot 1/2 \cdot a/b \cdot 1/3 \cdot a/c = 1/4 \cdot (a^2/b) - 1/6 \cdot (a^2/b) \), which is positive. So, below the line \( \frac{dH}{dt} = 0 \), the rate of change of \( H \) is positive (growth) and above the line the rate of \( H \) is negative.

k. Yes.

Figures between brackets indicate the time elapsed from the start of the simulation.
Exercise 45

a.

\[
\begin{align*}
\text{TITLE Predator prey} \\
\text{INITIAL} \\
\text{INCON } \text{IH } &= 3.0, \text{ IP } = 3.0 \\
\text{PARAM } &\text{A1 } = 1.0, \text{ A2 } = 0.5, \text{ B1 } = 1.0, \text{ B2 } = 0.5 \\
\text{TIMER } &\text{FINTIM } = 10, \text{ DELT } = 0.05, \text{ OUTDEL } = 0.25 \\
\text{PAGE WIDTH } &\text{= 80} \\
\text{OUTPUT } &\text{H, P} \\
\text{METHOD RECT} \\
\end{align*}
\]

\[
\begin{align*}
\text{DYNAMIC} \\
\text{H } &= \text{INTGRL(IH, GRH)} \\
\text{P } &= \text{INTGRL(IP, GRP)} \\
\text{GRH } &= \text{A1*H - A2*H*P} \\
\text{GRP } &= \text{B2*P*H - B1*P} \\
\end{align*}
\]

END
STOP
END JOB

b. \( a_1 = 1.0 \text{ time}^{-1} \), \( b_1 = 1.0 \text{ time}^{-1} \).

If \( P = 0 \), \( H \) grows exponentially with a half-live time of 0.693 time unit.
If \( H = 0 \), \( P \) declines exponentially with a similar half-live time as for \( H \).

c. \( a_2 : \text{time}^{-1} (\text{number } P)^{-1} \text{ m}^2 \)
(\( b_2 : \text{time}^{-1} (\text{number } H)^{-1} \text{ m}^2 \)
(\( a_2 : \text{number of } H \). This ratio expresses the conversion efficiency of prey into predator.
(\( b_2 : \text{number of } P \)

Since \( a_2/b_2 \) is a ratio of numbers, the information is not sufficient to judge if the numerical value is reasonable. If the ratio would be expressed in grams, for instance, its numerical value of 1 would be highly unlikely, because it would imply a conversion efficiency of 100 percent.

d and e. Numbers between parentheses in the following figure indicate the time elapsed since the start of the simulation. Note that the numbers \( H \) and \( P \) have returned to their original position after approximately 6.5 days.

The graph has not been obtained with \( \Delta t = 0.05 \), as given in a). This value was based on a similar reasoning as explained in the answer to Exercise 24b. The 5% criterion thus appears too flexible to let the system properly return to the initial conditions. This is due to the accumulation of roundoff errors resulting in a spiral-like graph (data not shown). The above graph, instead, was obtained by diminishing \( \Delta t \) once more with an order of magnitude (\( \Delta t \) thus was: (smallest time coefficient)/100=0.005). A similar result could be obtained by using a more
sophisticated integration method (see Chapter 6).

e. The system behaves as a so-called limit cycle.

Exercise 46
a.

TC = \frac{VOL}{INFLOW RATE} \cdot \frac{1}{1/\text{day}}
b. TITLE Growth of yeast

INITIAL
INCON IY =1.E-3, IALC=0.0, ISUG=12000.
PARAM MSUG =12000., TC =0.5
TIMER FINTIM=10. , DELT=0.02, OUTDEL=0.2
METHOD RECT
OUTPUT Y, SUG, ALC

DYNAMIC
Y =INTGRL(IY ,RY )
ALC =INTGRL (IALC, RALC)
SUG =INTGRL(ISUG,RSUG)

RY =RGR*Y - Y/TC
RALC =5.0*Y - ALC/TC
RSUG =24000. - Y*10. - SUG/TC
RGR =5. * SUG/MSUG

END
STOP
ENDJOB

c. At equilibrium the net change in amounts is zero. Solving the three differential equations for the three unknowns yields: SUG = 4800 gram, Y = 1440 gram, ALC = 3600 gram. The alcohol concentration is therefore 3%.
d. To calculate the value of $\Delta t$ for the rectangular integration method one should know all the time coefficients in the model. These are: 1) the inverse of the maximum relative growth rate RGR, 2) the average residence time TC of the yeast and the alcohol in the vessel, and 3) the average residence time of the sugar in the vessel.

The problem is similar to the one explained in Exercise 24b. The value of $\Delta t$ for the rectangular integration method can be calculated as $\Delta t=1/10 \cdot$ (smallest time coefficient) $= 0.02$ d.
e. There is exponential increase (growth of yeast) and exponential decrease (outflow of yeast). Assume that the average relative growth rate in the period in which equilibrium is attained, equals the arithmetic average of the values on time zero and when equilibrium is reached. Then, $\overline{RGR} = 3.5$. The initial amount of yeast was $10^{-3}$ g, the equilibrium value was 1440 g, and the exponential decrease of the yeast has a time constant of 0.5. The net relative growth rate in the period in which equilibrium is attained thus equals $(3.5 - 2) \text{ d}^{-1}$. From the analytical solution, $Y = Y_0 \cdot e^{((3.5-2) \cdot t)}$, it follows that $t = 9.45$ d. Take FINTIM somewhat larger for practical purposes.

Exercise 47

The results of the numerical integration are:

<table>
<thead>
<tr>
<th>t (s)</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>0</td>
<td>6</td>
<td>9.75</td>
<td>12.09375</td>
<td>13.5586</td>
<td>14.4741</td>
</tr>
</tbody>
</table>

(Decimals have been given for comparative purposes only.)
a. The numerical solution underestimates the analytical solution.
b. The rate at $t = 0$ is exact, but at $t = 2$ s the rate is underestimated, so the resulting average rate is too small, and the amount of water in the tank after one time interval is thus underestimated.

The rate at $t = 0$ is exact, but at $t = 2$ s the rate is underestimated, so the resulting average rate is too small, and the amount of water in the tank after one time interval is thus underestimated.
Exercise 48

Using the numerical parameter values of $W_0 = 0.1$, $W_m = 16.1$, $c = 1/4 \text{s}^{-1}$ and $\Delta t = 2 \text{s}$, the intermediate rates are $R_1 = 4$, $R_2 = 3$, $R_3 = 3.25$ and $R_4 = 2.375 \text{s}^{-1}$, resulting in a $W$ at $2 \text{s}$ equal to $6.2916 \text{l}$. The analytical solution is $6.2955 \text{l}$.

Exercise 49

a. The results of the rectangular and trapezoidal methods are $A_{t+\Delta t} = 0$ and $A_{t+\Delta t} = 50$, respectively.

b. When a division by $\Delta t$ occurs in a model, the rectangular method must be applied.

Exercise 50

a. $\Delta t = 0.02 \times \tau$, $\Delta t = 0.121/2 \times \tau$, $\Delta t = 28.81/4 \times \tau$

b. 1.7183; 1.0; 1.8591; 1.7189

c. | Method          | Rectangular | Trapezoidal | Runge-Kutta |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>exact absolute error</td>
<td>$-7.18\times 10^{-1}$</td>
<td>$14.08 \times 10^{-2}$</td>
<td>$6.00 \times 10^{-4}$</td>
</tr>
<tr>
<td>exact relative error</td>
<td>$-7.18\times 10^{-1}$</td>
<td>$7.57 \times 10^{-2}$</td>
<td>$3.49 \times 10^{-4}$</td>
</tr>
<tr>
<td>estimated relative error</td>
<td>$-5.0 \times 10^{-1}$</td>
<td>$8.33 \times 10^{-2}$</td>
<td>$3.47 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

(Table 5)

The estimates are good approximations of the relative errors of the numerical integration methods.

Exercise 51

a. $\Delta t = 0.02 \times \tau$, $\Delta t = 0.061/2 \times \tau$, $\Delta t = 1.21/4 \times \tau$

b. 2.0; 2.5; 2.7083; 2.7183. Note that the initial value of the integral must be added to the analytical result to make a proper comparison with the numerical integration methods.

c. | Method          | Rectangular | Trapezoidal | Runge-Kutta |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>exact absolute error</td>
<td>$-7.18\times 10^{-1}$</td>
<td>$-21.83 \times 10^{-2}$</td>
<td>$-10.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>exact relative error</td>
<td>$-3.59 \times 10^{-1}$</td>
<td>$-8.73 \times 10^{-2}$</td>
<td>$-3.69 \times 10^{-3}$</td>
</tr>
<tr>
<td>estimated relative error</td>
<td>$-5.0 \times 10^{-1}$</td>
<td>$-16.67 \times 10^{-2}$</td>
<td>$-8.33 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

(Table 5)

The exact relative errors are overestimated by a factor of two by the expressions in Table 5. This discrepancy is due to the large ratio of $\Delta t$ to $\tau$; in this case omission of the higher order terms in the Taylor expansion (Appendix 2) is not justified. When $\Delta t$ is taken to be about one tenth of $\tau$, or is adjusted to yield a reasonable error by means of the expressions in Table 5, the estimates are in good agreement with the exact relative errors.

Exercise 52

a. The value of $A$ after one integration step can be calculated from the initial value $A_0$ and the differential equation:

$$A_1 = A_0 + \Delta t \cdot c \cdot A_0 = A_0 \cdot (1 + \Delta t \cdot c).$$
Similarly $A_2, A_3,$ etc. can be calculated:

$$A_2 = A_1 + \Delta t \cdot c \cdot A_1 = A_1 \cdot (1 + \Delta t \cdot c),$$

$$A_3 = A_2 + \Delta t \cdot c \cdot A_2 = A_2 \cdot (1 + \Delta t \cdot c).$$

Substituting $A_{n-1}$ in $A_n$ yields the general expression for $A_n$ that gives the value of $A$ directly after $n$ time intervals of size $\Delta t$:

$$A_n = A_0 \cdot (1 + \Delta t \cdot c)^n.$$

b. To convert this equation of $n$ to a function of time, we use the relation $n = t / \Delta t$. Then,

$$A = A_0 \cdot (1 + \Delta t \cdot c)^{t / \Delta t}.$$

c. If the time interval $\Delta t$ approaches zero, the analytical solution $A = A_0 \cdot e^{t \cdot c}$ should be obtained. Therefore, the question is whether

$$\lim_{\Delta t \to 0} (1 + \Delta t \cdot c)^{t / \Delta t} \text{ is equal to } e^c.$$

To solve this limit, take $1/x = \Delta t \cdot c$. Then, $1/\Delta t = x \cdot c$ and $\Delta t \to 0$ is now equivalent to $x \to \infty$. Substitution yields:

$$\lim_{x \to \infty} (1 + 1/x)^x \cdot c, \text{ which indeed equals } e^c.$$

Exercise 53

a. The relational diagram:

![Relational Diagram]

The feedback of the amount of petrol to RDIST is significant only if PTRL is below the threshold level PTLLV. Linking the rate RDIST to RPTRL does not conflict with the principle that two rates should not mutually affect each other (Section 2.5), since only one rate affects the other: this is possible here since the petrol consumption rate is directly related to the rate with which the car moves forward.

b. The rate equation is:

$$\frac{dPTRL}{dt} = \frac{dDIST}{dt} \cdot PTUSE,$$

$$1 \text{ h}^{-1} = \text{ km h}^{-1} \cdot (1 \text{ km}^{-1}).$$
c. IF (PTRL .GE. PTLLV) THEN
    RDIST = AFGEN(RDISTB, TIME )
    PTUSE = AFGEN(PTUSET, RDIST)
    RPTRL = -RDIST * PTUSE
ELSE
    RDIST = 0
    RPTRL = 0
ENDIF

If this line is true, rates are non-zero, and these rates should be calculated.

If the 'IF (PTRL .GE. PTLLV) THEN' line is false, the car's speed should be zero. Note that RPTRL is also set to zero. Otherwise, it would still have the last value from before the moment that PTRL was smaller than PTLLV (one integration earlier), and petrol consumption would continue whilst the car was standing still and the engine turned off due to lack of petrol.

The equations within the IF-THEN-ELSE-ENDIF are sorted by the programmer. It is easily checked whether sorting has been done correctly by starting at the end of a listing and to see whether all the input data (to the right of the '=' sign) are known to calculate the particular variable (to the left of the '=' sign). Subsequently, the program should be run through from bottom to top.

In this case: RDIST and PTUSE must be known for RPTRL. This is why both are defined before RPTRL is calculated. PTUSET and RDIST are required for PTUSE. Functions are always known, but RDIST must be calculated previously in order that PTUSE may be read from the function.

d. TITLE Integration of a forcing function: car with a tank of petrol...
    that is finite
INITIAL
    IDIST =0.0 , IPTRL =60.0
PARAM
    PTLLV =2.0
FUNCTION
    RDISTB=0.0, 80.0 , 2.0, 80.0 , 3.0, 120.0 , 5.0, 120.0 , ...
    6.0, 40.0 , 10., 40.0
FUNCTION
    PTUSET=20.0, 0.111 , 90.0, 0.062 , 120.0, 0.082
TIMER
    FINTIM=10.0 , OUTDEL=0.5 , DELT=0.5
PAGE WIDTH =80
OUTPUT DIST , PTRL
OUTPUT RDIST, PTUSE
METHOD TRAPZ
DYNAMIC
NOSORT
DIST =INTGRL(IDIST,RDIST)
PTRL =INTGRL(IPTRL,RPTRL)

IF(PTRL .GE. PTLLV) THEN
    RDIST = AFGEN(RDISTB, TIME )
    PTUSE = AFGEN(PTUSET, RDIST)
    RPTRL =-RDIST*PTUSE
ELSE
    RDIST = 0.0
    RPTRL = 0.0
ENDIF

FINISH PTRL=PTLLV

END
STOP
END JOB

Note that the DYNAMIC is completely written in NOSORT, because all sorting has been carried out already in the IF-THEN-ELSE-ENDIF structure. It would thus be useless to apply a PROCEDURE-ENDPROCEDURE.

e. The second order integration method TRAPZ would exactly integrate the first order rate equation RDIST of the form \( y = ax + b \), because feedback of the simulated results on this rate does not occur. Since method TRAPZ calculates the rate at the start and at the end of the time increment, i.e. it reads the given function of time exactly, its result will be exact.
Exercise 54

a. TITLE Exponential growth with temperature dependent relative growth rate

INITIAL
FIXED COUNT1,COUNT2,KEEP
INCON IA =1.0
PARAM AVTMP =20.0, AMPTMP=10.
FUNCTION RGRTB =0.0,0.0 , 10.,0.08 , 20.,0.16 , 30.,0.21 , ...
            40.,0.24 , 50.,0.25
TIMER FIN Tim=48.0, OUT DEL=2.0, PRDEL=2.0, DELT=0.4
PRINT CNT1,CNT2
OUTPUT A,RGR,GR

PAGE WIDTH =80
METHOD RECT

* Calculations
PI =4.0*ATAN(1.0)

COUNT1=0
COUNT2=0

DYNAMIC
NOSORT
A =INTGRL(IA,GR)

TEMP =AVTMP + AMPTMP*SIN(2.0*PI*TIME/24.0)
RGR =AFGEN(RGRTB,TEMP)
GR =RGR*A

COUNT1=COUNT1 + 1
COUNT2=COUNT2 + KEEP
CNT1 =FLOAT(COUNT1)
CNT2 =FLOAT(COUNT2)

END
METHOD TRAPZ
TIMER DELT=1.0
END
METHOD RKSFX
TIMER DELT=2.0
END
METHOD RKS
END
STOP
ENDJOB

Columns 73 to 80 may be used for comments (here the word 'Counter'). All lines that were added to the program of Exercise 25 or changed in sequence, due to the introduction of NOSORT, are indicated by the word 'Counter'. Since integer variables cannot be PRINTed (see Section 3.5), the numerical values of the integer counters were transformed into reals by the function FLOAT, and assigned to the variables CNT1 and CNT2.
The results of the program at time 48 are:

<table>
<thead>
<tr>
<th>method</th>
<th>RECT</th>
<th>TRAPZ</th>
<th>RKSFX</th>
<th>RKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>time step</td>
<td>0.5</td>
<td>1.0</td>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1034.1</td>
<td>1330.3</td>
<td>1367.7</td>
<td>1368.6</td>
</tr>
<tr>
<td>COUNT1</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>218</td>
</tr>
<tr>
<td>COUNT2</td>
<td>97</td>
<td>49</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

We must assume that the results obtained with the automatically adapting method RKS are the most accurate. To get this result the program was executed ('UPDATEd') 218 times, and 25 intervals of integration were required to cover the total simulation period of 48 hours. On the average about ten consultations of this program are needed for each time step. This number drops to four with RKSFX, to two with TRAPZ and to one with RECT. Therefore, the number of times the program was consulted was the same in these three runs with DELT = 2, DELT = 1, and DELT = 0.5, respectively. With the same computational effort method RKSFX gives by far the most accurate results, the next is TRAPZ and RECT scores the worst. The deviation with RECT is of the order to be expected. On the average the relative growth rate was \( \ln(1368)/48 \) or 0.15. With DELT = 0.5 and FINTIM = 48 the relative error can be calculated from Table 5 in Section 6.4 as 0.27, so that the result should be about 1000. Although RGR was varying this estimate for the relative error is still quite reasonable.

b. In this case, in spite of the varying time coefficient, method RKSFX is preferred to method RKS. This illustrates that it worthwhile to perform some experiments with the integration method in a particular model before a final choice be made.

Exercise 55

a. The relational diagram:

![Relational Diagram]

b. The rate equations are as follows:

Individual flows:

\[
\begin{align*}
\text{FLOW}1 &= 0.0 \\
\text{FLOW}2 &= \text{H1/TC} = \text{H1/(DTSYS/2)} \\
\text{FLOW}3 &= \text{H2/TC} = \text{H2/(DTSYS/2)}
\end{align*}
\]

Net flows:

\[
\begin{align*}
\text{RH}1 &= \text{FLOW}1 - \text{FLOW}2 \\
\text{RH}2 &= \text{FLOW}2 - \text{FLOW}3 \\
\text{RCLLCT} &= \text{FLOW}3
\end{align*}
\]

Agreement should be reached on numbering the flows. The rule applied here is that a flow is assigned the number of the integral into which it flows: FLOW1 flows into H1; FLOW2 flows into H2.
The state equations are as follows:

\[
\begin{align*}
H_1(t+\Delta t) &= H_1(t) + \Delta t \cdot RH_1(t) \\
H_2(t+\Delta t) &= H_2(t) + \Delta t \cdot RH_2(t) \\
CLLC(t+\Delta t) &= CLLC(t) + \Delta t \cdot RCLLCT(t)
\end{align*}
\]

(Strictly speaking, these state equations are only valid for the rectangular integration method.)

c. The average residence time or time coefficient of one delay element is: 
\[
\text{(Total delay time of the system, } DTSYS) / \text{(number of delay elements)} = 8/2 = 4 \text{ weeks, with the dimension of time.}
\]

d. TITLE Oil transport: second order exponential delay

INITIAL
FIXED I,N
STORAGE FLOW(3)
INCON ICLLCT =0.0
TABLE IH(1-2)=100.0, 0.0
PARAM DTSYS =8.0 , N =2
TIMER FINTIM =25.0 , OUTDEL=0.5, DELT=0.5
PAGE WIDTH =80 \quad GROUP , NTAB=0
OUTPUT H(1-2) ,CLLCT
OUTPUT RH(1-2),RCLLCT
PRINT H(1-2) ,RH(1-2),CLLCT,RCLLCT
METHOD RKSFX

* Calculations
TC =DTSYS/\text{FLOAT}(N)

DYNAMIC
H =INTGRL(IH ,RH ,2)
CLLCT =INTGRL(ICLLCT,RCLLCT)

PROCEDURE RH=NETFLO(FLOW)
DO 10 I=1,N
     RH(I) =FLOW(I) - FLOW(I+1)
10 CONTINUE
ENDPRO

PROCEDURE FLOW,RCLLCT=FLOWS(H,TC)
     FLOW(1) =0.0
     DO 20 I=2,N+1
        FLOW(I)=H(I-1)/TC
20 CONTINUE
RCLLCT =FLOW(N+1)
ENDPRO

END
STOP
ENDJOB

RKSFX is the best integration method for this problem, because no discontinuities occur, and the smallest time coefficient is known (4 weeks) and stable.

The integration interval could be taken as 2 weeks (Table 4, Section 6.3). However, in the program 0.5 week was chosen to obtain graphs with a reasonable degree of time resolution.
Exercise 56

a. The statements in the IF-THEN-ELSE-ENDIF structure of the program describing a time event can be expressed in natural language as follows:

IF (a definitive integration has been carried out) THEN
    IF (time < the moment at which harvesting is to be carried out) THEN
        The discontinuity is not yet reached. Therefore,
        - remember current biomass in the variable OLDBIO;
        - remember current time in the variable OLDTIM;
    ELSE
        The time has increased beyond the moment that the discontinuity should occur. Therefore,
        IF (time > acceptable moment at which harvesting is to be carried out) THEN
            Return to the situation prior to the moment at which the discontinuity should occur. Therefore,
            - reset biomass 'BIOM' to previous value 'OLDBIO';
            - reset time 'TIME' to previous value 'OLDTIM';
            - calculate the last interval of integration before the moment of harvesting as the difference of that moment and the previous value of time, i.e. 'OLDTIM';
        ELSE
            The correct 'moment' of harvesting is reached. Therefore,
            - call the CSMP PRINT statement, so that the situation immediately before the replacement procedure is printed; (This statement is added only for the purpose of inspection.)
            - assign (a fraction of) the value of the state variable BIOMass to that of 'HARVeST';
            - then remove the (same fraction of the) value of BIOMass from the integral 'BIOM' (the remaining amount is used for regrowth);
            (Note that the latter two statements should not be reversed: then the BIOMass would have been changed before it is stored in HARVST.)
            - calculate the rate associated with the new state which has just been reached;
            - reset the value of DELMAX to the smallest value of PRDEL or OUTDEL; (The value of DELMAX may be very small due to a small difference between the moment that the discontinuity should occur and the previous value of time, i.e. 'OLDTIM'. To proceed after the discontinuity, therefore, DELMAX should be reset to the largest possible value.)
            - increase the event counter 'IEVENT' to choose the next time event that should be reached;
            - call the CSMP PRINT statement, so that the situation immediately following the replacement procedure is printed; (This statement is added only for the purpose of inspection.)
    END IF
END IF
ELSE
The time has increased beyond the moment that the discontinuity should occur. Therefore,

END IF
END IF
END IF

b. No answer given.
Exercise 57

a. Steps to obtain an algorithm for a state event.

- Starting point just before discontinuity
- State event
- Maximum tolerable overshoot in state event 
  \((1+\epsilon)^{(state\ event)}\)

<table>
<thead>
<tr>
<th>Discontinuity not yet reached.</th>
<th>Correct value is reached.</th>
<th>Gone too far.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions:</td>
<td>Actions:</td>
<td>Actions:</td>
</tr>
<tr>
<td>- store current values of state variables, including time;</td>
<td>- remove integral contents;</td>
<td>- remove old state from before the required discontinuity (back to starting point);</td>
</tr>
<tr>
<td>- continue the integration process.</td>
<td>- reset maximum integration step (DELMAX) to original value (PRDEL or OUTDEL);</td>
<td>- half the maximum integration step (DELMAX);</td>
</tr>
<tr>
<td></td>
<td>- continue the calculations.</td>
<td>- recalculate rate;</td>
</tr>
</tbody>
</table>

b. TITLE Discontinuity: state event using the replace method

INITIAL
FIXED KEEP
INCON IBiom =100.
PARAM BIOHST=190., STUBBL=10., RGR =0.2, EPS =0.0001
TIMER FINTIM=35. , PRDEL =0.2, DELMAX=0.2
PRINT BIOM, RBIOM, HARVST, OLDBIO, OLDTIM, DELMAX, DELT
METHOD RKS

DYNAMIC
NOSORT
BIOM =INTGRL(IBIOM, RBIOM)
HARVST =INTGRL(0.0 ,0.0 )

IF(KEEP .EQ. 1) THEN
  IF(BIOM .LT. BIOHST) THEN
    OLDBIO =BIOM
    OLDTIM =TIME
  ELSE
    IF(BIOM .GT. (1.0+EPS)*BIOHST) THEN
      BIOM =OLDBIO
      TIME =OLDTIM
      DELMAX =DELMAX/2.0
    ELSE
      CALL PRINT
      HARVST =HARVST + (BIOM-STUBBL)
      BIOM =STUBBL
      RBIOM =RGR*BIOM
      DELMAX =PRDEL
      CALL PRINT
    ENDIF
  ENDIF
ENDIF

END

STOP

ENDJOB
In this program BIOMass is not harvested completely; a STUBBLE remains on the field. This stubble forms the basis for new growth. Therefore, a harvest is the amount of biomass minus the amount of stubble that remains on the field. Each new harvest is added to the previous one.

c. No answer given.

Exercise 58

a. 

![Diagram of the model](image)

b. The state equation if PUSH=0 is:

\[
\text{BIOM}_{t+\Delta t} = \text{BIOM}_t + \Delta t \cdot (\text{RGR} \cdot \text{BIOM}_t)
\]

The state equation if PUSH=1 is:

\[
\text{BIOM}_{t+\Delta t} = \text{BIOM}_t + \Delta t \cdot (\text{RGR} \cdot \text{BIOM}_t - 1.0 \cdot \left( \frac{\text{ BIOM}_t}{\Delta t} + \text{RGR} \cdot \text{BIOM}_t \right)) = 0.0
\]

The rate equation includes the term RGR BIOM. This implies that the amount of growth during harvesting, i.e. \(\Delta t \cdot (\text{RGR} \cdot \text{BIOM}_t)\), is harvested too.

Note that only the integration method of Euler may be used (see Exercise 49 from Section 6.3).

c. TITLE Discontinuity: time event using the infinite rate method

```
INITIAL
INCON IBIOM =100.
PARAM THRST=5.0, RGR =0.1
TIMER FINTIM=7.0, PRDEL=1.0, DELT=1.0
PRINT BIOM, RBIOM, HARVST, RHRVST, PUSH
METHOD RECT
DYNAMIC
BIOM =INTGRL(IBIOM,RBIOM-RHRVST)
HARVST =INTGRL(0.0,RHRVST)
RBIOM =RGR*BIO
RHRVST =PUSH*((BIOM/DELT) + RGR*BIO)
PUSH =INSW(TIME-THRST,0.0,1.0)
END
STOP
END:JOB
```

Note that the program only describes a single harvest.
Exercise 59

a. The best integration routine for this program is the fourth order Runge-Kutta method with a fixed time interval, RKSFX, since the model contains a fixed time coefficient.

b. The appropriate time interval is about half the time constant for this method (Section 6.3 Table 4) which is about 8 days. A practical value would thus be 7 days.

c. TITLE Pollution of a lake by city waste water.

INITIAL
INCON IC = 0.0
PARAM V = 1.25E9, Q = 1.0E7, W = 6.0E4, K = 0.05
TIMER FINITI= 100.0 , OUTDEL= 7.0 , DELT= 7.0
OUTPUT C, RC
PAGE WIDTH = 80
METHOD RKSFX

* Calculations
T0 = V/Q
C MAX = (W/(Q+V*K))
C MAXH = 0.98*C MAX

DYNAMIC
NOSORT
C = INTGR(IC, RC)

CALL RATE(W, V, C, T0, K, RC)

FINISH C = C MAXH

END
STOP

***************************************************************

SUBROUTINE RATE(W, V, C, T0, K, RC)
IMPLICIT REAL (A-Z)
SAVE
RC = W/V - C/T0 - K*C
RETURN
END

***************************************************************

ENDJOB

Exercise 60

TITLE Oil transport: second order exponential delay

INITIAL
FIXED I, N
STORAGE FLOW(3)
INCON ICLLCT = 0.0
TABLE IH(1-2)=100.0, 0.0
PARAM DTSYS = 8.0 , N = 2
TIMER FINITI= 25.0 , OUTDEL= 0.5 , DELT= 0.5
PAGE WIDTH = 80 , GROUP , NTAB=0
OUTPUT H(1-2), CLLCT
OUTPUT RH(1-2), RCLLCT
PRINT H(1-2), RH(1-2), CLLCT, RCLLCT
METHOD RKSFX

* Calculations
TAU = DTSYS/FLOAT(N)

DYNAMIC
NOSORT
H = INTGRL(IH, RH, 2)
CLLCT = INTGRL(ICLLCT, RCLLCT)

CALL FLOWS(N, TAU, H, ...
  FLOW, RCLLCT)

CALL NETFLO(N, FLOW, ...
  RH)

END
STOP

************************************************************************

SUBROUTINE FLOWS(N, TAU, H, FLOW, RCLLCT)
IMPLICIT REAL (A-Z)
INTEGER I, N
DIMENSION H(N), FLOW(N+1)
SAVE
FLOW(1) = 0.0
DO 10 I=2, N+1
  FLOW(I) = H(I-1)/TAU
10  CONTINUE
RCLLCT = FLOW(3)
RETURN
END

************************************************************************

SUBROUTINE NETFLO(N, FLOW, RH)
IMPLICIT REAL (A-Z)
INTEGER I, N
DIMENSION FLOW(N+1), RH(N)
SAVE
DO 10 I=1, N
  RH(I) = FLOW(I) - FLOW(I+1)
10  CONTINUE
RETURN
END

************************************************************************

END JOB

Note that by virtue of the subroutine structure, the main program becomes a summary of everything that is to be calculated, while details are worked out in subroutines. This is more evident in larger programs, of course.
Exercise 61

a. TITLE Juveniles and Adults with one state variable and a delay function

INITIAL
INCON  IADULT=1000.
PARAMETER  RLE  =0.01 ,  RDR =0.0
PARAMETER  PERIOD=200.
TIMER   FINTIM=1000.,  OUTDEL=100.0
OUTPUT  ADULT
METHOD  RKS

DYNAMIC
ADULT   =INTGRL(IADULT, NADULT)
NADULT  =RIADUL - RDADUL
RIADUL  =DELAY(20,PERIOD, EGGLAY)
EGGLAY  =RLE * ADULT
RDADUL  =RDR * ADULT

END
STOP
END JOB

b. TITLE Juveniles and Adults with two state variables

INITIAL
INCON  IJUVS =0.0 ,  IADULT=1000.
PARAMETER  RLE  =0.01 ,  RDR =0.0 ,  RRA =0.005
PARAMETER  PERIOD=200.
PARAMETER  FINTIM=1000.,  OUTDEL=100.0
OUTPUT  JUVS, ADULT
METHOD  RKS

DYNAMIC
JUVS  =INTGRL(IJUVS , NJUVS)
ADULT =INTGRL(IADULT, NADULT)
NJUVS =EGGLAY - RIADUL
NADULT =RIADUL - RDADUL
EGGLAY =RLE * ADULT
RIADUL =RRA * JUVS
RDADUL =RDR * ADULT

END
STOP
END JOB

c. The results of both models shows a large difference in the calculated number of Adults. Since the Juveniles in the model in b) have a similar mean residence time as in the model in a), the difference must be due to the variance in the second model, which implies that a number of Juveniles have more quickly developed than others, contributing significantly to the rate of growth of the entire population.

Exercise 62

a. In the range from 6 °C to 25 °C, the following relation exists:

\[
\text{degree-days} = (\text{temperature} - \text{threshold temperature}) \times \text{duration}
\]

\[
490 = (TA - 6) \times \text{duration}
\]

At 10 °C, the duration will be \(490/(10-6) = 122.5\) days. At 20 °C, this value will be 35 days.
b. The rate of development at 10 °C will be \((15-9)/122.5 = 0.049 \text{ d}^{-1}\). At 20 °C, this value equals 0.171 \text{ d}^{-1}.

Exercise 63
In the same way as in Exercise 62, the development rate at 25 °C can be calculated as 0.233 \text{ d}^{-1}.

```
TITLE Development of rape seed
INITIAL
PARAM TA = 10.
FUNCTION DEVRTB = 0., 0., 6., 0., 25., 0.233, 30., 0.233
TIMER FINTIM = 130., PRDEL = 5.
PRINT STAGE, DEVR
METHOD RKS

DYNAMIC
STAGE = INTGRL(9., DEVR)
DEVR = AFGEN(DEVRTB, TA)
END

PARAM TA = 20.
END
STOP
END
```

Exercise 64
a. The value of \(\gamma\) is \((15-9)/4 = 1.5\).
b. The development rate is the inverse of the duration between stage 9 and stage 15 (in \text{d}^{-1}).
c. The resetting occurs after 1.5 stage units, this is after \(490/4 = 122.5\) degree-days.

Exercise 65
\(T_{\text{tot}} = 20\) and \(\sigma_{\text{tot}} = 2\), so that the relative dispersion \(RD = 0.1\). This amount of relative dispersion could be obtained with 100 boxcars in the boxcar train \((1/(100^{1/2}) = 0.1\).

Exercise 66
To derive Equation 49,

\[
\bar{\tau} = \frac{1}{A_{i,0}} \sum_{j=1}^{\infty} j \cdot F \cdot \tau \cdot A_{i,0} (1 - F)^{j-1} \cdot F
\]

\[
\begin{array}{cc}
\text{time} & \text{quantity transferred} \\
\end{array}
\]

Equation 49

we need to find expressions for the time and the quantity that is transferred after the \(j^{th}\) shift.

**Quantity transferred** Each quantity that is shifted, is a fraction \(F\) of the amount present then. The general expression for the amount remaining in boxcar \(A_i\) after the \(j^{th}\) shift is:

\[
A_{i,j} = A_{i,j-1} - F \cdot A_{i,j-1} = (1 - F) \cdot A_{i,j-1}.
\]

The general expression for the amount transferred from boxcar \(A_i\) after the \(j^{th}\) shift is:

\[
A_{i,j} = F \cdot A_{i,j-1}.
\]
These expressions are used to calculate the amount that flows out of boxcar \( A_i \), and the amount that remains in that boxcar, after \( j \) shifts, and expressed in its initial contents \( A_{i,0} \). The results are given in the following table.

<table>
<thead>
<tr>
<th>shift number</th>
<th>outflow from boxcar, or quantity transferred</th>
<th>amount remaining in boxcar</th>
</tr>
</thead>
<tbody>
<tr>
<td>( j=1 )</td>
<td>( F \cdot A_{i,0} )</td>
<td>( A_{i,1} = (1 - F) \cdot A_{i,0} )</td>
</tr>
<tr>
<td>( j=2 )</td>
<td>( F \cdot A_{i,1} = F \cdot (1 - F) \cdot A_{i,0} )</td>
<td>( A_{i,2} = (1 - F) \cdot A_{i,1} = (1 - F)^2 \cdot A_{i,0} )</td>
</tr>
<tr>
<td>( j=3 )</td>
<td>( F \cdot A_{i,2} = F \cdot (1 - F)^2 \cdot A_{i,0} )</td>
<td>( A_{i,3} = (1 - F) \cdot A_{i,2} = (1 - F)^3 \cdot A_{i,0} )</td>
</tr>
<tr>
<td>( j=4 )</td>
<td>( F \cdot A_{i,3} = F \cdot (1 - F)^3 \cdot A_{i,0} )</td>
<td>( )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( )</td>
<td>( )</td>
</tr>
<tr>
<td>( j=j )</td>
<td>( F \cdot A_{i,j-1} = F \cdot (1 - F)^{j-1} \cdot A_{i,0} )</td>
<td>( A_{i,j} = (1 - F) \cdot A_{i,j-1} = (1 - F)^j \cdot A_{i,0} )</td>
</tr>
</tbody>
</table>

So, the quantity that is transferred after the \( j^{th} \) shift is \( F \cdot (1 - F)^{j-1} \cdot A_{i,0} \).

Time elapsed Each shift occurs after \( F \) of the residence time \( \tau \) in a boxcar: \( F \cdot \tau \). The time elapsed after \( j \) shifts thus is: \( j \cdot F \cdot \tau \).

Note the analogy of Equation 49 with the one presented in Exercise 9, Section 2.5.

**Exercise 67**

Note that the average delay period should be kept the same: only the number of boxcars is changed. Further, compare your curves with those of Figure 25.

**Exercise 68**

a. The average delay period, is calculated by \( 1/DEVR: 1/0.005=200 \). Its variance \( \sigma_{\text{total}}^2 \) is then \( 200^2/N \), which equals 8000 for \( N=5 \). For \( N = 1, 2 \) and 10, the variance is 40000, 20000 and 4000, respectively.

b. The simulated values of \( ADP \) and \( VAR \) at time 1000 are given below:

<table>
<thead>
<tr>
<th>( N )</th>
<th>( ADP )</th>
<th>( VAR )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>192.9</td>
<td>32826.</td>
</tr>
<tr>
<td>2</td>
<td>200.5</td>
<td>19393.</td>
</tr>
<tr>
<td>5</td>
<td>201.0</td>
<td>7800.8</td>
</tr>
<tr>
<td>10</td>
<td>201.0</td>
<td>3801.0</td>
</tr>
</tbody>
</table>

c. Only after all individuals have flown out of the last boxcar, do these quantities mean what we want.

**Exercise 69**

The statements to be added to the program in Figure 30 are given here.

Changes in the INITIAL:
STORAGE FS(20)
TIMER OUTDEL=100.
OUTPUT FS0,FS(1-20)

Changes in the DYNAMIC:
* Calculation of the age-distribution
  FRACA0 = A0/ATOT
  FS0 = A0/5000.0
  DO 30 I=1,N
  FRACA(I)=A(I)/ATOT

199
FS(I) = A(I)/5000.0

CONTINUE

* The total birth rate of girls is now set to 1000.0
TBR = 1000.0
* And the original TBR is cancelled by making it a comment statement.
* Total birth rate of girls
* TBR = TBR + A(I) * (FRGIRL * 0.001*AFTGEN(ARITB,AGE(I)))

If all survived, the contents of each age class would be 5000. The fraction of survival FS(I) can therefore be calculated by A(I)/5000. The fraction of A(I) in the total population at present can be calculated by A(I)/ATOT.

<table>
<thead>
<tr>
<th>Mean age</th>
<th>Age distribution</th>
<th>FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>0.0675</td>
<td>0.987</td>
</tr>
<tr>
<td>7.5</td>
<td>0.0667</td>
<td>0.975</td>
</tr>
<tr>
<td>12.5</td>
<td>0.0665</td>
<td>0.972</td>
</tr>
<tr>
<td>17.5</td>
<td>0.0664</td>
<td>0.971</td>
</tr>
<tr>
<td>22.5</td>
<td>0.0663</td>
<td>0.969</td>
</tr>
<tr>
<td>27.5</td>
<td>0.0661</td>
<td>0.967</td>
</tr>
<tr>
<td>32.5</td>
<td>0.0659</td>
<td>0.964</td>
</tr>
<tr>
<td>37.5</td>
<td>0.0656</td>
<td>0.959</td>
</tr>
<tr>
<td>42.5</td>
<td>0.0651</td>
<td>0.951</td>
</tr>
<tr>
<td>47.5</td>
<td>0.0641</td>
<td>0.937</td>
</tr>
<tr>
<td>52.5</td>
<td>0.0626</td>
<td>0.915</td>
</tr>
<tr>
<td>57.5</td>
<td>0.0605</td>
<td>0.885</td>
</tr>
<tr>
<td>62.5</td>
<td>0.0580</td>
<td>0.848</td>
</tr>
<tr>
<td>67.5</td>
<td>0.0541</td>
<td>0.791</td>
</tr>
<tr>
<td>72.5</td>
<td>0.0475</td>
<td>0.695</td>
</tr>
<tr>
<td>77.5</td>
<td>0.0356</td>
<td>0.520</td>
</tr>
<tr>
<td>82.5</td>
<td>0.0177</td>
<td>0.259</td>
</tr>
<tr>
<td>87.5</td>
<td>0.0039</td>
<td>0.057</td>
</tr>
<tr>
<td>92.5</td>
<td>0.0001</td>
<td>0.001</td>
</tr>
<tr>
<td>97.5</td>
<td>0.0000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Exercise 70**

After 1000 years of simulation the female population has grown from $6.42 \times 10^6$ ($t = 0.0$) to $2.89 \times 10^{10}$.
Exercise 71

a. Since the data points between the temperatures 5 and 20 °C lay on a straight line the values at 8 and 10 °C are not needed for a proper description of the data in an AFGEN function. Therefore, the description reads:

\[
\text{FUNCTION DEVRT} = 0.0, 0.0, 5.0, 0.0, 20.0, 0.15, 30.0, 0.15
\]

b. The threshold point is the coordinate pair 5.0, 0.0. If this point is not included in the function, interpolation between the temperatures 0 and 20 °C will be applied to the line given by the coordinate pairs 0.0, 0.0 and 20.0, 0.15. Then a value of \( \nu \) of 0.075 is calculated at 10 °C, instead of 0.05.

c. To design the corresponding function for the delay period, one has first to calculate the inverse of the development rate:

\[
\begin{array}{cccc}
T_a & 0 & 5 & 8 \\
\text{delay} & \infty & \infty & 33.3
\end{array}
\]

\[
\begin{array}{cccc}
& 20.0 & 6.67 & 6.67
\end{array}
\]

These data can be described by the function:

\[
\text{FUNCTION DELAYT} = 8.0, 33.3, 10.0, 20.0, 20.0, 6.67, 30.0, 6.67
\]

Comparison of a graph of this function with the one for the development rate shows that the latter is more linearly related to the temperature.

Exercise 72

a. \( F \) should be larger than \( \Delta t \cdot \nu / \gamma \). The developmental width of a class is smaller when the number of boxcars is taken larger: \( \gamma = g_f / N \), where \( g_f \) is usually taken equal to 1, e.g. \( \text{GAMMAA} = 1. / \text{FLOAT(NA)} \) in Figure 31. At large \( N \), the ratio \( \Delta t \cdot \nu / \gamma \) will thus be large and consequently \( F \) will not exceed this ratio anymore.

b. \( N \) should be smaller than 1 divided by the maximally occurring value of \( \text{RD}^2 \), because:

\[
N \text{ is calculated from } F = 1. - (N \cdot \text{RD} \cdot \text{RD}) \text{ as derived from Equation 51. If } N = 1/\text{RD}^2, F \text{ would be 0. If } N < 1/\text{RD}^2, \text{ or } N \cdot \text{RD}^2 < 1, F \text{ would be larger that 0. The criterion mentioned under a) remains valid, however.}
\]

Exercise 73

Use the dimensional rules given in the answer to Exercise 1. Further, no answer is given here.

Exercise 74

a. No answer is given with regard to the dimensional analysis, but some remarks about the program listing in Figure 35 are made:

- The integrals represent amounts (kg salt), from which the concentrations have to be derived ( H(I)/VOL ). In principle, it is also possible that \text{INTGRLs} directly contain concentrations. However, this is not recommended since the same concentration in layers of different thickness will represent different values for the amounts in these layers. In other words: if concentrations are used, either the layer thicknesses should be constant, or the differential equations should be adapted. Furthermore, working with amounts means that the geometry of the system must be carefully examined and explicitly defined in the model of the system considered.

- The \text{DYNAMIC} segment only contains variables, and no numerical values, which means that it is only necessary to change the parameters in the \text{INITIAL} segment to use the model for calculations with another number of layers or different concentrations. The third argument in the \text{CSMP INTGRL} statement, however, must be a number. Therefore, that argument must be altered when a different number of layers is needed.

- In FLUX(N+1), the index (N+1) is calculated to the left of the equal-sign. This is an exception to the rule that
calculations should be performed to the right side of the equal-sign.

b. $$$ CONTINUOUS SYSTEM MODELING PROGRAM III V1M3 EXECUTION OUTPUT $$$

TITLE Elementary description of mass flow
TABLE IH(1-20)=20*0.0
PARAM N=20, CS=10.0, V=0.1, TCOM=0.05, AREA=1.0
TIMER FINTIM=20.0, PRDEL=2.0, OUTDEL=1.0
PRINT C(1-20), FLUX(21)
OUTPUT C(20)
METHOD RKS
ABSERR H(1)=1.0E-4
RELSERR H(1)=1.0E-4
END

0TIMER VARIABLES RKS INTEGRATION START TIME = 0.000000E+00
DELT DELMIN FINTIM PRDEL OUTDEL DELMAX
6.25000E-02 2.00000E-06 20.0000 2.0000 1.0000 1.0000

(Tabular results are omitted here.)

OUTPUT VARIABLE RANGES FOR ALL RUNS IN CASE

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
<th>VARIABLE</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>0.000000E+00</td>
<td>20.0000</td>
<td>TIME</td>
<td>0.000000E+00</td>
<td>10.00</td>
</tr>
<tr>
<td>C(20)</td>
<td>0.000000E+00</td>
<td>'+'=C(20)</td>
<td>C(20)</td>
<td>0.000000E+00</td>
<td>9.99822</td>
</tr>
</tbody>
</table>

1$$ CONTINUOUS SYSTEM MODELING PROGRAM III V1M3 EXECUTION OUTPUT $$$

c. 10 kg fertilizer per m³ (CS) is a reasonable value: if 200 kg ha⁻¹ fertilizer is mixed into the first centimetre of soil with a moisture content of 0.2 m³ m⁻³, there will be 0.02 kg fertilizer in 2 litres water per square metre. This corresponds to a concentration of 10 kg m⁻³. Normally speaking, however, the concentration of the fertilizer in the first layer will decrease with time due to rain and/or irrigation, and will finally approach zero. This
could be achieved by assigning a certain value in the program to \( H(1) \) and taking \( C_S \) equal to 0.0, which would represent irrigation with clean water on a soil containing fertilizer in the first layer. This has not been done here, since the program is used to demonstrate how the permanently imposed surface concentration is distorted by numerical dispersion.

A rate of water flow, \( V \), of 0.1 m d\(^{-1}\) is rather high: it actually implies a rate of penetration, \( v' \), of 5 cm d\(^{-1}\) at a porosity \( \varepsilon = 0.5 \text{ m}^3 \text{ m}^{-3} \), since \( v' = v \cdot \varepsilon \).

### Exercise 75

Equation 62 reads:

\[
\frac{dC(I)}{dt} = \frac{NFL(I)}{TCOM} = \frac{V}{TCOM} \cdot \left( C(I-1) - C(I+1) \right) / 2 \tag{Equation 62}
\]

\( C(I-1) \) was expressed in \( C(I) \) already in Equation 59:

\[
C(I-1) = C(I) - TCOM \cdot \frac{dC(I)}{dx} + \frac{TCOM^2}{2!} \cdot \frac{d^2C(I)}{dx^2} + \frac{TCOM^3}{3!} \cdot \frac{d^3C(I)}{dx^3} + \frac{TCOM^4}{4!} \cdot \frac{d^4C(I)}{dx^4} + \ldots
\]

For the application of the Taylor series to \( C(I+1) \), it holds that (Figure 34):

- \( C(I) \) value of the function in point \( x \);
- \( C(I+1) \) value of the function in point \( x_1 \);
- \( (x-x_1) \) distance between the positions where \( C = C(I+1) \) and \( C = C(I) \): since \( C(I+1) \) refers to the concentration at a position deeper in the soil column than \( C(I) \), \( x_1 \) is smaller than \( x \) and \( (x-x_1) \) is positive and equal to \( TCOM \).

Substituting these terms in Equation 58 yields:

\[
C(I+1) = C(I) + TCOM \cdot \frac{dC(I)}{dx} + \frac{TCOM^2}{2!} \cdot \frac{d^2C(I)}{dx^2} + \frac{TCOM^3}{3!} \cdot \frac{d^3C(I)}{dx^3} + \frac{TCOM^4}{4!} \cdot \frac{d^4C(I)}{dx^4} + \ldots
\]

Entering \( C(I-1) \) and \( C(I+1) \) into Equation 62 results in:

\[
\frac{dC(I)}{dt} = \frac{V}{TCOM} \cdot \left( (C(I) - TCOM \cdot \frac{dC(I)}{dx} + \frac{TCOM^2}{2!} \cdot \frac{d^2C(I)}{dx^2} + \frac{TCOM^3}{3!} \cdot \frac{d^3C(I)}{dx^3} + \frac{TCOM^4}{4!} \cdot \frac{d^4C(I)}{dx^4} + \ldots ) - (C(I) + TCOM \cdot \frac{dC(I)}{dx} + \frac{TCOM^2}{2!} \cdot \frac{d^2C(I)}{dx^2} + \frac{TCOM^3}{3!} \cdot \frac{d^3C(I)}{dx^3} + \frac{TCOM^4}{4!} \cdot \frac{d^4C(I)}{dx^4} + \ldots ) / 2 \right)
\]

\[
= \frac{V}{TCOM} \cdot \left( -2 \cdot TCOM \cdot \frac{dC(I)}{dx} - 2 \cdot TCOM^3 \cdot \frac{d^3C(I)}{dx^3} - \ldots \right) / 2
\]

\[
= - \frac{V}{6} \cdot \frac{dC(I)}{dx} - \frac{V \cdot TCOM^2 \cdot d^3C(I)}{dx^3} - \ldots
\]

This expression no longer contains any terms with even numbers.

### Exercise 76

a. The lines of computer code that should be changed in Figure 35 are:

```
TITLE Dispersion corrected description of mass flow
and in DO-loop 20:

FLUX(1) = CS * V
```

203
DO 20 I =2,N
   FLUX(I) = ( (C(I-1) + C(I))/2.0 ) * V
20 CONTINUE
   FLUX(N+1) =C(N) * V

b. The relational diagram in Figure 34 should also include feedbacks from the integrals to the incoming flows, to express the approach according to Equation 61.
c. $$$ CONTINUOUS SYSTEM MODELING PROGRAM III V1M3 EXECUTION OUTPUT $$$

TITLE Dispersion corrected description of mass flow
TABLE IH(1-20)=20*0.0
PARAM N=20 , CS=10.0, V=0.1 , TCOM=0.05, AREA=1.0
TIMER FINTIM=20.0, PRDEL=2.0, OUTDEL=1.0
PRINT C(1-20), FLUX(21)
OUTPUT C(20)
PAGE WIDTH=80
METHOD RKS
ABSERR H(1)=1.0E-4
RELERR H(1)=1.0E-4
END

OTIMER VARIABLES RKS INTEGRATION START TIME = 0.000000E+00
DELT DELMIN FINITIM PRDEL OUTDEL DELMAX
6.250000E-02 2.000000E-06 20.000 2.0000 1.0000 1.0000

(Tabular results are omitted here.)

OUTPUT VARIABLE RANGES FOR ALL RUNS IN CASE

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
<th>VARIABLE</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>0.000000E+00</td>
<td>20.0000</td>
<td>C(20)</td>
<td>0.0000000E+00</td>
<td>11.8874</td>
</tr>
<tr>
<td>TIME</td>
<td>0.000000E+00</td>
<td></td>
<td>C(20)</td>
<td>0.000000E+00</td>
<td>12.00</td>
</tr>
<tr>
<td>C(20)</td>
<td>0.000000E+00</td>
<td></td>
<td></td>
<td>0.000000E+00</td>
<td></td>
</tr>
</tbody>
</table>

1$$ CONTINUOUS SYSTEM MODELING PROGRAM III V1M3 EXECUTION OUTPUT $$
Exercise 77
a. No answer given.
b. If the thickness of the layers is constant, the value of AVC(I) equals \( \frac{C(I) + C(I-1)}{2} \), which is similar to the concentration used in Equation 61 to suppress numerical dispersion.

Exercise 78
a. TITLE Elementary description of diffusive flow
   INITIAL
   FIXED I,N
   STORAGE FLUX(21),C(20)
   TABLE IH(1-20) =20*0.0
   * kg
   PARAM N=20 , CS=1.0, D=4.0E-5, TCOM=0.05, AREA=1.0
   * number kg m**2 m m**2
   * of ---- ----
   * layers m**3 day
   TIMER FINTIM=100.0,PRDEL=5.0,OUTDEL=5.0
   * day
   PRINT C(1-10),C(20),FLUX(1-10),FLUX(20)
   PAGE WIDTH =80,NTAB=0,GROUP
   OUTPUT C(1-4)
   OUTPUT C(5-8)
   OUTPUT C(9-10),C(20)
   METHOD RKS
   ABSERR H(1)=1.0E-4
   RELERR H(1)=1.0E-4

   * Calculations
   VOL =AREA*TCOM
   DYNAMIC
   NOSORT
   H =INTGRL(IH,NFLOW,20)
   DO 10 I =1,N
   \( C(I) = H(I)/VOL \)
   10 CONTINUE
   FLUX(1) = - D \( \times \) ( C(1)-CS ) / (0.5*TCOM)
   DO 20 I =2,N
   FLUX(I) = - D \( \times \) ( C(I) - C(I-1) ) / TCOM
   20 CONTINUE
   FLUX(N+1) = 0.0
   DO 30 I =1,N
   NFLOW(I)= FLUX(I)*AREA - FLUX(I+1)*AREA
   30 CONTINUE
   END
   STOP
   END

b. The differential equation for the first soil layer is:

\[ FLUX(1) = - D \times (C(1)-CS) / (0.5*TCOM) \]
The diffusion distance in this equation equals the distance between the soil surface and the centre of the layer, which is $0.5 \cdot TCOM$. Just above the soil perfect mixing is assumed between the fertilizer and the water in which it is dissolved.

**Exercise 79**

Our results are:

```plaintext
TITLE Elementary description of diffusive flow

INITIAL

FIXED I,N

STORAGE FLUX(21),C(20)

PARAM N=20 , CS=0.0, D=4.0E-5, TCOM=0.05, AREA=1.0

* kg

of layers

m m**2 m**2

number kg

kg

VOL = AREA * TCOM

DYNAMIC

NOSORT

H = INTGRL(IH,NFLOW,20)

BALANH = 0.0

DO 5 I=1,N

BALANH = BALANH + H(I)

5 CONTINUE

DO 10 I =1,N

C(I) = H(I)/VOL

10 CONTINUE

FLUX(1) = 0.0

DO 20 I =2,N

FLUX(I) = -D * ( C(I) - C(I-1) ) / TCOM

20 CONTINUE

FLUX(N+1) = 0.0

DO 30 I =1,N

NFLOW(I) = FLUX(I)*AREA - FLUX(I+1)*AREA

30 CONTINUE

END

STOP

ENDJOB
```
The fertilizer is applied to the layers 4 and 5 of the soil profile. Each layer, having an area of one square metre, is initialized with an amount of 0.01 kg, which corresponds with 200 kg ha\(^{-1}\).

a. FLUX(I) is set to 0.0, since nutrients dissolved in water cannot diffuse out of the soil. Gaseous substances could diffuse through the soil surface, since gas continuity exists with the atmosphere.

The surface flux could not have been set to 0 by equating the parameter CS to 0, since then an infinite 'sink' would be created, causing a continuous diffusive flow of nutrients out of the soil. That situation would represent a rice field continuously flooded with fresh water.

b. The plot of the concentration of the fertilizer in layers 1 to 10 against depth for \(t=0\) \(t=40\), and \(t=100\) days reads (C corresponds to \(t=0\), E to \(t=40\) and H to \(t=100\) days):

![Graph](image)

c. The concentration profile of the fertilizer is asymmetric since the soil surface forms a layer impervious to the nutrients, while in the opposite direction a column of practically infinite length exists. This is demonstrated by the very small increase in concentration calculated after 100 days at a depth of about 1 metre (1.017 \(10^{-11}\) kg m\(^{-3}\)).

d. The amount of fertilizer in the soil, or the mass balance can either be calculated by

\[
\sum_{i=1}^{20} (C_i \cdot \text{VOL}) = 0.05 \cdot \sum_{i=1}^{20} C_i \quad \text{(kg)}
\]

or by

\[
\sum_{i=1}^{20} H_i
\]

depending on the formulation of the program. The latter expression in terms of CSMP reads:

```
BALANH = 0.0
DO 5 I=1,N
      BALANH = BALANH + H(I)
5   CONTINUE
```

and is incorporated in the program listing. The BALANce of the amount H equals 0.02 all the time.
Exercise 80
a. No answer is given.
b. Some of the assumptions underlying the model are:
   - the soil is homogeneous, i.e. its composition of solid, liquid and gaseous phase does not change with depth.
   - the driving force of the temperature cycle over the day has been taken as sinusoidal. This has been done as a first approximation and enables the calculation of an analytical solution that can be used to check the model for calculational errors (the analytical solution is explained in Subsection 9.3.4).
These assumptions will not always be valid. However, it is relatively simple to introduce location-specific parameters, for example via an AFGEN function. The average conductivity should then be calculated in a different way (see Appendix 5). It is also possible to calculate soil conductivity from its component parts (Wierenga & de Wit, 1970; ten Berge, 1990). Such calculations are necessary if, for example, the water content varies with depth and time.
c. Derivation of the time coefficient of the model:
   - If the temperature at the soil surface suddenly changes by a value DELTMP, the maximum heat content that can flow from the first layer can be calculated using Equation 73:
     \[
     \text{DELTMP} \times \text{AREA} \times \text{TCOM} \times \text{VHCAP} \text{ (Joule)},
     \]
     where \( \text{AREA} \times \text{TCOM} = \text{VOL} \).
   - The rate of flow from the first layer at the moment of the sudden temperature change can be calculated using Equation 72:
     \[
     (\text{DELTMP} \times \text{COND} / (0.5 \times \text{TCOM})) \times \text{AREA} \text{ (Joule s\(^{-1}\))},
     \]
   - The time coefficient is found now by dividing the maximum heat content to be removed by the flow rate:
     \[
     \text{TC} = \frac{\text{DELTMP} \times \text{AREA} \times \text{TCOM} \times \text{VHCAP}}{(\text{DELTMP} \times \text{COND} / (0.5 \times \text{TCOM})) \times \text{AREA}} = \frac{0.5 \times \text{TCOM}^2 \times \text{VHCAP}}{\text{COND}} \text{ (s)}
     \]
d. The numerical value of the time coefficient calculated using the data from the program, is \( \text{TC} = 500 \text{ s} \).
   This means that the time interval of integration should be approximately 50 s if integration is carried out by method RECT.

Exercise 81
At depth \( x = d \), Equation 74 becomes:
\[
T_{x,t} = T_{av} + T_{\text{ampl}} \times \exp(-1) \times \sin(\omega \times t - 1)
\]
Therefore, the amplitude of the temperature wave at \( x = d \) is \( T_{\text{ampl/\epsilon}} \).

Exercise 82
For the analytical calculations a value of \( d \) is required. This value is calculated with Equations 76 and 75 and the parameter values from the program in Figure 38 as \( d = 0.104884649 \text{ m} \). The average depth of layer 5 is 0.09 m, so \( x = 0.09 \). Time should be entered into Equation 74 in seconds (or the values for \( \lambda \) and \( C_b \) should first be converted to days). Substituting the calculated values into the table results in:
time, d  :  19  19.25  19.50  19.75  20
simulated value, °C  :  16.782  22.743  23.219  17.257  16.782
analytical solution :  16.792  22.772  23.208  17.228  16.792

(Equation 74)

Exercise 83

If the thickness of all layers is taken to be equal and the conductivities are identical for each layer, AVCOND(I) assumes the value of COND. Up till now, this value has always been used, since the conductivity was assumed to be constant throughout the profile. Note that the same equation could be used to calculate the average diffusion coefficient between two layers: COND should then be replaced by D.

Exercise 84

a. Our results are:

```
TITLE Flow of heat in a non-homogeneous soil column...
INITIAL
NOSORT
FIXED I,N
STORAGE TCOM(27), DIST(27), DEPTH(27), VOL(27)
STORAGE VHCAP(27), COND(27), AVCOND(28), FLUX(28), TEMP(27)
TABLE TCOM(1-27)=27*0.02
FUNCTION CONDT = (0.0,0.25 , (0.0399,0.25) , (0.04,0.42) , (0.54,0.42)
* J
* ----------------
* m degree-C s
FUNCTION VHCAPT=(0.0,2.5E6) , (0.0399,2.5E6) , (0.04,1.05E6) , ...
(0.54,1.05E6)
* J
* ----------------
* m**3 degree-C
PARAM AREA=1.0, ITMP=1.5, TAM=1.5, TAV=6.5, N=27
* m**2 degree-C
TIMER FINTIM=864000.0, OUTDEL=3600.0, PRDEL=3600.0
* s
PAGE WIDTH=80, NTAB=0, GROUP
OUTPUT TEMP(1), TEMP(3), TEMP(5), TEMP(7), TEMP(27)
PRINT TEMP(1-10), TEMP(27)
METHOD RKS

* calculations
PI = 4.0*ATAN(1.0)

DIST(1) = 0.5*TCOM(1)
DEPTH(1) = DIST(1)
DO 10 I=2,N
   DIST(I) = 0.5*( TCOM(I-1)+TCOM(I) )
   DEPTH(I) = DEPTH(I-1) + DIST(I)
10 CONTINUE

DO 20 I=1,N
   VOL(I) = TCOM(I)*AREA
20 CONTINUE

DO 30 I=1,N
   COND(I) = AFGEN(CONDT, DEPTH(I))
```

209
b. When this program is run with a peat layer of 4 cm, no freezing occurs at a soil depth between 8-10 cm (the depth including the peat mulch is 4 cm larger). Clearly, one could also use this program to find out which mulch thickness would be needed to prevent frost at shallower depths. Without the peat mulch no freezing would occur at a soil depth between 16-18 cm.

Exercise 85

a. If the transport coefficients can be taken constant throughout the profile, it means that the profile is homogeneous with regard to its transport properties.

b. The answer in a) can only be true if the arrangement of the soil particles (the solid phase), the water phase and the gas phase is similar at each depth. In a non-saturated soil there will always be a soil water gradient, and since transport properties strongly depend on the moisture content, they will not be constant. Therefore, a permanently inundated rice or wetland soil is considered.

c. It is given that roots take up nutrients between 5 and 30 cm depth. Furthermore, relative uptake rates (RUR) for the rooted layers are given for layers of 5 cm thickness, and the total profile depth equals one metre. Therefore, it seems most appropriate to start the simulation with a layer thickness of 0.05 m. Since the soil has a depth of one metre, this implies that the number of soil layers will be 20.
If the concentration gradients in the soil profile would be steep, it might be necessary to diminish the layer thickness to improve the simulation. In this example the concentration gradients are especially steep in the top of the soil at the onset of the simulation, because then water containing fertilizer is poured onto the soil that is initially free of fertilizer. The appropriate layer thickness could be found by a trial-and-error method similar to the one described in the answer to Exercise 24b for the determination of the time increment of integration. The starting point for the trial-and-error method is that the space increment should not significantly affect the outcome of the model, analogous to the determination of the time increment. By halving and/or doubling the layer thickness and studying the results of the calculations, the best value of the space increment can be found easily. One should be aware that a change in the layer thickness affects the time coefficient of the model. Therefore, it is best to work with the Runge-Kutta-Simpson method with a variable time step, RKS.

d. If it is assumed that the diffusive flow at the bottom of the soil model equals zero, the actual flow at the bottom of the soil system should be negligibly small. This can only be true if the concentration gradient at the bottom of the soil system would be about zero, i.e. the concentrations in adjacent soil layers are about equal, so that hardly any diffusive transport would occur or that the contribution of the diffusive flow to the total flow (diffusive plus mass flow) would be negligible. These conditions are to be expected, since the nutrient concentration that results from the nutrient input at the top of the profile and the nutrient uptake in the first 30 cm of the soil, percolates into the deeper layers and will be rather constant in the long run. See also Exercise 84f.

c. TITLE Elementary description of mass flow, diffusive flow and uptake

INITIAL
FIXED I,N
STORAGE FLXDIF(21),FLXMAS(21),UPTAKE(20),RUR(20),C(20)
TABLE IH(1-20) =20*0.0
* kg
TABLE RUR(1-20) = 0.0 , 0.1 , 0.2 , 0.3 , 0.2 , 0.1 , 14*0.0
* 1/day
PARAM N=20 , CS=10.0, V=0.001, D=4.0E-5, TCOM=0.05, AREA=1.0
* number kg m**3 m**2 m m**2
* of ---- -------- ----
* layers m**3 m**2 day day

TIMER FINTIM=100.0, PRDEL=5.0, OUTDEL=5.0
* day
OUTPUT C(1-20), TOTUPT
OUTPUT FLXDIF (1-21), FLXMAS (1-21), UPTAKE (1-10)
PAGE WIDTH =80, NTAB=0, GROUP
OUTPUT C(1-4)
OUTPUT C(5-8)
OUTPUT C(9-12)
METHOD RKS
ABSERR H(1) =1.0E-6
RELERR H(1) =1.0E-6

* Calculations
VOL = AREA*TCOM

DYNAMIC
NOSORT
H = INTGRL(IH, NFLOW, 20)
TOTUPT=INTGRL(0.0, NUPT)

DO 10 I =1,N
C(I) = H(I)/VOL
10 CONTINUE
* description of diffusion
  FLXDIF(1) =-D * ( C(1)-CS ) / (0.5*TCOM)
  DO 20 I =2,N
      FLXDIF(I)=-D * ( C(I) - C(I-1) ) / TCOM
  20 CONTINUE
  FLXDIF(N+1) = 0.0

* description of mass flow
  FLXMAS(1) = CS * V
  DO 30 I =2,N
      FLXMAS(I)= ( (C(I-1) + C(I))/2.0 ) * V
  30 CONTINUE
  FLXMAS(N+1) = C(N) * V

* description of uptake
  DO 40 I =1,N
      UPTAKE(I)= RUR(I) * C(I)
  40 CONTINUE

* total uptake by roots
  NUPT = 0.0
  DO 50 I =1,N
      NUPT = NUPT + UPTAKE(I) * VOL
  50 CONTINUE

* net rate of change of nutrient in a soil layer
  DO 60 I =1,N
      NFLOW(I) = FLXDIF(I)*AREA- FLXDIF(I+1)*AREA + FLXMAS(I)*AREA - FLXMAS(I+1)*AREA - UPTAKE(I)*VOL
  60 CONTINUE

END
STOP
ENDJOB

The calculation of the total (accumulated) amount of fertilizer that is taken up by the roots is included in the program by the state variable TOTUPT and the total rate NUPT, that is calculated in DO-loop 50.

The total simulation time is chosen as 100 days, about the length of a growing season.

The values for ABSERR and RELERR were taken as 10^-6, because at the default value of 10^-4 unexpected negative diffusive flows occurred after TIME=700 days.

f. As far as can be judged, the results of the program seem satisfactory. In fact one can only judge that the concentrations of the fertilizer in the layers varies between 0.0 and the applied surface concentration, which can never be exceeded physically.

During the period considered the diffusive flow and the mass flow are both very small at the bottom of the profile. Thus, the assumption that the diffusive flow can be set at zero at the bottom of the profile is warranted, and the soil column could be considered as a semi-infinite medium for all practical purposes. (Based on inspection of the results of the program in Exercise 84f and simulation times of 100 and 1000 days, respectively.)

To fully judge the performance of the model, one should compare the results with experimental data.

g. If v = 0.005, the concentration in the top layer is larger than 10 kg m^-3. Physically this is not possible, since the input fertilizer concentration is only 10 kg m^-3. The fluctuations around 10 kg m^-3 have a similar cause as in Exercise 76: the higher-order (odd-numbered) terms in the Taylor series distort the results of the model and

212
the natural diffusion is not large enough to level out these distortions, because the diffusion coefficient $D$ is not larger than $(V \cdot TCOM)/2$. This means that the thickness $TCOM$ should be decreased, i.e. $TCOM < 2 \cdot D/V$ when the flow rate is increased.

Exercise 86

a. Exponential growth with temperature dependent relative growth rate (CSMP in Exercise 25)

SUBROUTINE MODEL (ITASK, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)

* The STANDARD (!!!) parameter list:

* ITASK - task of model routine I
* OUTPUT = .TRUE. output request (ITASK=2 only) I
* TIME - time I
* STATE - state array of model I/O
* RATE - rates of change belonging to STATE I/O
* SCALE - size scale of state variables I/O
* NDEC - declared size of arrays I
* NEQ - Number of state variables, for ITASK=1 0
* otherwise I

IMPLICIT REAL (A-Z)

* formal parameters, === do not change this section !!! ====
INTEGER ITASK, NDEC, NEQ
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL OUTPUT

* common /INFO/, === do not change this section !!! ====
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT, PRDEL, DELMAX, FINTIM,
$IULOG, IUMOD, KEEP , TERMNL$

* local (non-common) variables
INTEGER IURES, IUDAT
INTEGER ILZ, N
DIMENSION RGRTB(12)
SAVE

IF (ITASK.EQ.1) THEN
  initial
  ===============
  * get local unit numbers
  IURES = IUMOD
  IUDAT = IUMOD + 2

  * open input file
  CALL RDINIT (IUDAT, IULOG, 'MODEL.DAT')

  * initial state
  CALL RDSREA ('IA', A )

  * model parameters (see data file for meaning of symbols)
  CALL RDAREA ('RGRTB', RGRTB, 12, ILZ)
  CALL RDSREA ('AVTMP', AVTMP )

  213
CALL RDSREA ('AMPTMP', AMPTMP )
CALL RDSINT ('N', N )
scales
CALL RDAREA ('SCALE', SCALE, NDEC, NEQ)
CLOSE (IUDAT, STATUS='DELETE')

* check the size of the SCALE array on file:
IF (NEQ.NE.N) CALL ERROR ('MODEL', 'SCALE length mismatch')

* initialize output
CALL OUTDAT (1, IURES, 'TIME', 0.0)

* initialize state variables
STATE(1) = A

* initial calculations
PI = 4.0 * ATAN(1.0)
ELSE IF (ITASK.EQ.2) THEN

* rates of change

* assign state to local variable names
A = STATE(1)

* equations of the model
* Derived variables, i.e. the concentrations

* calculation of rates of change
TEMP = AVTMP + AMPTMP*SIN( (2.0*PI/24.0) *TIME )
RGR = RGR * A

* output section
IF (OUTPUT) THEN
    CALL OUTDAT (2, 0, 'TIME', TIME)
    CALL OUTDAT (2, 0, 'DELT', DELT)
    CALL OUTDAT (2, 0, 'A', A )
    CALL OUTDAT (2, 0, 'GR', GR )
    CALL OUTDAT (2, 0, 'RGR', RGR )
END IF

* assign result to state array
STATE(1) = A

* assign result to rate array
RATE(1) = GR
ELSE IF (ITASK.EQ.4) THEN

* terminal

* table output in two columns and delete temporary
CALL OUTDAT (4, 0, 'Results of exponential growth', 0.0)
CALL OUTDAT (99, 0, '', 0.0)
END IF

RETURN
END
b.

* TIMER.DAT
STTIME = 0.0  ! start time
FINTIM = 48.0  ! finish time
PRDEL = 2.0  ! output time step
EPS = 1.E-4  ! tolerable relative integration error
DELMAX = 2.0  ! Maximum allowed timestep
DELT = 0.4  ! timestep Euler integration

* MODEL.DAT
IA = 1.0  ! kg
SCALE = 1.0
N = 1.0  ! number of integrals
AVTMP = 20.0  ! degree centigrade
AMPTMP = 10.0  ! degree centigrade
RGRTB = 0.0,0.0, 10.,0.08, 20.,0.16, 30.,0.21, 
         40.,0.24, 50.,0.25  ! 1/day

Exercise 87

a.

Discontinuity: state event using the replace method (CSMP in Exercise 57)

SUBROUTINE MODEL (ITASK, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)

* The STANDARD (!!!) parameter list:
*
* ITASK - task of model routine  I
* OUTPUT = .TRUE. output request (ITASK=2 only)  I
* TIME - time  I
* STATE - state array of model  I/O
* RATE - rates of change belonging to STATE  I/O
* SCALE - size scale of state variables  I/O
* NDEC - declared size of arrays  I
* NEQ - Number of state variables, for ITASK=1 0
* otherwise I

IMPLICIT REAL (A-Z)

* formal parameters, === do not change this section !!! ====
INTEGER ITASK, NDEC, NEQ
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL OUTPUT

* common /INFO/,  === do not change this section !!! ====
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/: DELT, PRDEL, DELMAX, FINTIM,
                 IULOG, IUMOD, KEEP, TERMNL

* local (non-common) variables
INTEGER IURES, IUDAT
INTEGER N
SAVE

IF (ITASK.EQ.1) THEN
  initial
* get local unit numbers
IURES = IUMOD
IUDAT = IUMOD + 2

* open input file
CALL RDINIT (IUDAT,IULOG,'MODEL.DAT')

* initial state
CALL RDSREA ('BIOM',BIOM)
CALL RDSREA ('HARVST',HARVST)

* model parameters (see data file for meaning of symbols)
CALL RDSREA ('BIOHST',BIOHST)
CALL RDSREA ('STUBBL',STUBBL)
CALL RDSREA ('RGR',RGR)
CALL RDSREA ('EPS2',EPS2)
CALL RDSREA ('N',N)

* scales
CALL RDAREA ('SCALE',SCALE,NDEC,NEQ)
CLOSE (IUDAT,STATUS='DELETE')

* check the size of the SCALE array on file:
IF (NEQ.NE.N) CALL ERROR ('MODEL','SCALE length mismatch')

* initialize output
CALL OUTDAT (1,IURES,'TIME',0.0)

* initialize state variables
STATE(1) = BIOM
STATE(2) = HARVST

ELSE IF (ITASK.EQ.2) THEN
* rates of change
* assign state to local variable names
BIOM =STATE(1)
HARVST =STATE(2)

* equations of the model
IF (KEEP .EQ. 1) THEN
  IF (BIOM .LT. BIOHST) THEN
    OLDBIO = BIOM
    OLDTIM = TIME
  ELSE
    IF (BIOM .GT. (1.0+EPS2)*BIOHST) THEN
      BIOM = OLDBIO
      TIME = OLDTIM
      DELMAX = DELMAX/2.0
    ELSE
      CALL OUTDAT (2,0,'TIME' ,TIME )
      CALL OUTDAT (2,0,'DELT' ,DELT )
      CALL OUTDAT (2,0,'BIOM' ,BIOM )
      CALL OUTDAT (2,0,'RBIOM',RBIOM )
      CALL OUTDAT (2,0,'HARVST',HARVST)
      CALL OUTDAT (2,0,'OLDBIO',OLDBIO)
      CALL OUTDAT (2,0,'OLDTIM',OLDTIM)
      CALL OUTDAT (2,0,'DELMAX',DELMAX)
      CALL OUTDAT (2,0,'DELT' ,DELT )
HARVST = HARVST + (BIOM - STUBBL)
BIOM = STUBBL
RBIOM = RGR * BIOM
DELMAX = PRDEL

CALL OUTDAT (2, 0, 'TIME', TIME)
CALL OUTDAT (2, 0, 'DELT', DELT)
CALL OUTDAT (2, 0, 'BIOM', BIOM)
CALL OUTDAT (2, 0, 'RBIOM', RBIOM)
CALL OUTDAT (2, 0, 'HARVST', HARVST)
CALL OUTDAT (2, 0, 'OLDBIO', OLDBIO)
CALL OUTDAT (2, 0, 'OLDTIM', OLDTIM)
CALL OUTDAT (2, 0, 'DELMAX', DELMAX)
CALL OUTDAT (2, 0, 'DELT', DELT)

ENDIF
ENDIF
ENDIF

RBIOM = RGR * BIOM

* output section
IF (OUTPUT) THEN
CALL OUTDAT (2, 0, 'TIME', TIME)
CALL OUTDAT (2, 0, 'BIOM', BIOM)
CALL OUTDAT (2, 0, 'RBIOM', RBIOM)
CALL OUTDAT (2, 0, 'HARVST', HARVST)
CALL OUTDAT (2, 0, 'OLDBIO', OLDBIO)
CALL OUTDAT (2, 0, 'OLDTIM', OLDTIM)
CALL OUTDAT (2, 0, 'DELMAX', DELMAX)
CALL OUTDAT (2, 0, 'DELT', DELT)
ENDIF

* assign result to state array
STATE(1) = BIOM
STATE(2) = HARVST

* assign result to rate array
RATE(1) = RBIOM
RATE(2) = 0.0

ELSE IF (ITASK.EQ.4) THEN
* terminal
*	able output in two columns and delete temporary
CALL OUTDAT (4, 0, 'Results of discontinuity program', 0.0)
CALL OUTDAT (99, 0, '', 0.0)
ENDIF

RETURN
END

b.
* TIMER.DAT
STTIME = 0.0   ! start time
FINTIM = 35.0   ! finish time
PRDEL = 0.2    ! output time step
EPS = 1.E-4    ! tolerable relative integration error
DELMAX = 0.2 ! Maximum allowed timestep
DELT  = 0.2 ! timestep Euler integration

* MODEL.DAT
BIOM  = 100. ! kg
HARVST = 0.0 ! kg
SCALE  = 2*1.
N      = 2 ! number of equations
BIOHST = 190.0 ! kg
STUBBL = 10.0 ! kg
RGR    = 0.2 ! 1/d
EPS2   = 0.0001 ! -

Exercise 88
a.

Elementary description of diffusive flow (CSMP in Exercise 78)

SUBROUTINE MODEL (ITASK,OUTPUT,TIME,STATE,RATE,SCALE,NDEC,NEQ)

* Diffusion of nutrients through saturated soil. At the top of
* the soil profile water with a constant nutrient concentration
* is flowing.
* The STANDARD (!!) parameter list:
*
* ITASK - task of model routine I
* OUTPUT = .TRUE. output request (ITASK=2 only) I
* TIME  - time I
* STATE - state array of model I/O
* RATE  - rates of change belonging to STATE I/O
* SCALE - size scale of state variables I/O
* NDEC  - declared size of arrays I
* NEQ   - Number of state variables, for ITASK=1 O
* otherwise I

IMPLICIT REAL (A-Z)
*
formal parameters, === do not change this section !! ====
INTEGER ITASK, NDEC, NEQ
DIMENSION STATE(NDEC),RATE(NDEC),SCALE(NDEC)
LOGICAL OUTPUT
*
common /INFO/, === do not change this section !! ====
REAL DELT , PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMINL
COMMON /INFO/ DELT , PRDEL, DELMAX, FINTIM,
$ IULOG, IUMOD, KEEP , TERMINL
*
local (non-common) variables
INTEGER IURES, IUDAT
INTEGER ILZ , I , N
DIMENSION H(20), FLUX(21), C(20), NFLOW(20)
SAVE

IF (ITASK.EQ.1) THEN
*
initial
*
* get local unit numbers
IURES = IUMOD
IUDAT = IUMOD + 2

* open input file
CALL RDINIT (IUDAT,IULOG,'MODEL.DAT')
* initial state
CALL RDAREA ('H',H,20,ILZ)
* model parameters (see data file for meaning of symbols)
CALL RDSREA ('CS' ,CS )
CALL RDSREA ('D ' ,D )
CALL RDSREA ('TCOM',TCOM )
CALL RDSREA ('AREA',AREA )
CALL RDSREA ('N' ,N )
* scales
CALL RDAREA ('SCALE',SCALE,NDEC,NEQ)
CLOSE (IUDAT,STATUS='DELETE')

* check the size of the SCALE array on file:
IF (NEQ.NE.N) CALL ERROR ('MODEL','SCALE length mismatch')

* initialize output
CALL OUTDAT (1,IURES,'TIME',0.0)
* initialize state variables.
DO 10 I=1,N
    STATE(I)= H(I)
10    CONTINUE
* initial calculations
VOL = AREA*TCOM
ELSE IF (ITASK.EQ.2) THEN
* rates of change
* assign state to local variable names
DO 20 I=1,N
    H(I) = STATE(I)
20    CONTINUE
* equations of the model
* Derived variables, i.e. the concentrations
DO 30 I =1,N
    C(I) = H(I)/VOL
30    CONTINUE
* calculation of rates of change
FLUX(1) = -D * ( C(1)-CS ) / (0.5*TCOM)
DO 40 I =2,N
    FLUX(I) = -D * ( C(I) - C(I-1) ) / TCOM
40    CONTINUE
FLUX(N+1) = 0.0
DO 50 I =1,N
    NFLOW(I)= FLUX(I)*AREA - FLUX(I+1)*AREA
50    CONTINUE
* output section
IF (OUTPUT) THEN
    CALL OUTDAT (2 ,0 ,'TIME' ,TIME )
CALL OUTDAT (2,0,'DELT',DELT)
CALL OUTARR ('C',C,1,10)
CALL OUTDAT (2,0,'C(20)',C(20))
CALL OUTARR ('FLUX',FLUX,1,10)
CALL OUTDAT (2,0,'FLUX(20)',FLUX(20))
END IF

* assign result to state array
DO 60 I=1,N
STATE(I) = H(I)
60 CONTINUE

* assign result to rate array
DO 70 I=1,N
RATE(I) = NFLOW(I)
70 CONTINUE

ELSE IF (ITASK.EQ.4) THEN
* table output in two columns and delete temporary
CALL OUTDAT (4,0,'Results of diffusion program',O.O)
CALL OUTDAT (99,0,' ',0.0)
END IF

RETURN
END

b.
* TIMER.DAT
STTIME = 0.0 ! start time
FINTIM = 20.0 ! finish time
PRDEL = 2.0 ! output time step
EPS = 1.0E-4 ! tolerable relative integration error
DELMAX = 2.0 ! Maximum allowed timestep
DELT = 1.0 ! timestep Euler integration

* MODEL.DAT
H = 20*0. ! kg
SCALE = 20*1.
N = 20 ! number of layers
CS = 10.0 ! kg/m**3
D = 4.0E-5 ! m**2/day
TCOM = 0.05 ! m
AREA = 1.0 ! m**2

Exercise 89
a.
b. \( \text{RDECR} = -(1/35) \cdot \ln (1/2) = 0.02 \text{ yr}^{-1} \).

c. The time coefficient is the time required by the system to reach equilibrium when the rate would be kept constant. The total relative rate of decrease of the organic matter is \( \text{RDECR} + \text{RLCHR} = 0.025 \text{ yr}^{-1} \), thus the TC = 40 years.

d. TITLE Organic matter
INITIAL
INCON IORMAT = 200000.
PARAM RSUPPL = 4000., RDECR = 0.02, RLCHR = 0.005
TIMER FINTIM = 500., DELT = 10.0, PRDEL = 10.0
PRINT ORMAT,RORMAT
METHOD RKSFX

DYNAMIC
ORMAT = INTGRL(IORMAT,RORMAT)

RORMAT = RSUPPL - (RDECR + RLCHR) * ORMAT

END
STOP
ENDJOB

RKSFX was chosen as the integration method, because the time coefficient of the model is constant. The DELT may then be taken as half the time coefficient, yielding 20 yr. However, if output is desired each decade, the time step should be adapted to this interval.

e. When a considerable period has elapsed, an equilibrium will develop in the soil with regard to the organic matter content:

\[
\text{ORMAT} = \frac{\text{RSUPPL}}{\text{RDECR} + \text{RLCHR}} = 160000 \text{ kg organic matter ha}^{-1}.
\]

This amount is contained in \(375.10^4\) kg of soil, yielding an organic matter content of 4.27%.

f. In the 'infinite rate method' a sudden change in the integral content can be achieved by defining the inflow rate as follows (see Section 6.3 and Subsection 7.2.4, Equation 41):

\[
\text{RSUPPL} = (4000.0/\text{DELT}) \times \text{PUSH}
\]

\[
\text{PUSH} = \text{IMPULS}(0.0,1.0)
\]

Because of the discontinuity, the integration method RECT should be used. The time step should be decreased too: it should be one year at the most. However, if \(\Delta t\) would be equal to 1 year, the program would still be continuous, because the \text{IMPULS} function would always be 1 at multiples of \(\Delta t\). This means that the discontinuity will only affect the results of the model, if the \(\Delta t\) would be taken smaller than 1 year. When \(\Delta t\) was taken 0.01, there was about 1% difference between the results of the different models. This illustrates that the initial assumption that the addition of organic matter is evenly distributed throughout the year is good.

The program using the 'replace method' is:
TITLE Organic matter
INITIAL
NOSORT
FIXED IEVENT, KEEP, N
STORAGE TEVENT(501)
INCON IORMAT = 200000.
PARAM APPLIC = 4000.0, RDECR = 0.02, RLCHR = 0.005, EPS=1.E-4
TIMER FINTIM = 500.0, DELMAX= 0.01, PRDEL = 10.0
PRINT ORMAT, RORMAT
METHOD RKS

* Calculations

N =INT(FINTIM+1)
DO 10 IEVENT=1,N
    TEVENT(IEVENT)=FLOAT(IEVENT-1)
10 CONTINUE

IEVENT =1
MAXDEL =DELMAX

DYNAMIC
NOSORT
ORMAT = INTGRL(IORMAT, RORMAT)

IF(KEEP .EQ. 1) THEN
    IF(TIME .LT. TEVENT(IEVENT)) THEN
        OLDORM = ORMAT
        OLDTIM = TIME
    ELSE
        IF(TIME .GT. (1.0+EPS)*TEVENT(IEVENT)) THEN
            ORMAT = OLDORM
            TIME = OLDTIM
            DELMAX = TEVENT(IEVENT) - OLDTIM
        ELSE
            ORMAT = ORMAT + APPLIC
            DELMAX = MAXDEL
            IEVENT = IEVENT +1
        ENDIF
    ENDF
ENDIF

RORMAT = RSUPPL - (RDECR + RLCHR) * ORMAT

END
STOP
ENDJOB

The time step of integration in this program was taken similar to the one in the program containing the 'infinite rate method'. It appears that the amount of organic matter in the soil fluctuates around the equilibrium value of 1.6 $10^5$ kg per hectare in the two programs containing the discontinuous application.
Exercise 90

a. A relational diagram for this system is:

The various variables with their units are:
- HERB: amount of herbicide, kg ha⁻¹
- BEET, BEEMAX: number of beetles ha⁻¹
- LHERB: level of herbicide above which beetle growth decreases exponentially, kg ha⁻¹
- DRES: herbicide dose applied, kg ha⁻¹
- TCHERB: time coefficient of herbicide in soil, yr
- RGR, RDR: relative rates of increase and decrease of the beetles, yr⁻¹.

b. The rate of decrease of the herbicide in the soil is proportional to the amount present. Hence, decomposition proceeds exponentially and can be described by the equation: \( Y = Y_0 \cdot e^{-\left(\frac{t}{TCHERB}\right)} \). Characteristic moments are just before and just after application:

- \( Y \) (na 3 jaar) = \( 50 \cdot e^{-\left(\frac{3}{10}\right)} = 37 \) 67
- \( Y \) (na 5 jaar) = \( (37.0 + 30) \cdot e^{-\left(\frac{2}{10}\right)} = 54.9 \) 84.9
- \( Y \) (na 7 jaar) = \( (54.9 + 30) \cdot e^{-\left(\frac{2}{10}\right)} = 69.5 \) 99.5
- \( Y \) (na 9 jaar) = \( (69.5 + 30) \cdot e^{-\left(\frac{2}{10}\right)} = 81.5 \) 111.5
- \( Y \) (na 11 jaar) = \( (81.5 + 30) \cdot e^{-\left(\frac{2}{10}\right)} = 91.3 \) —

c. The dynamics of the beetle population are controlled by the level of herbicide: below 68 kg ha⁻¹ the beetle population follows a logistic growth pattern; above 68 kg ha⁻¹ the population decreases exponentially. From the answer to b) it follows that logistic growth occurs during the first 5 years. The equation for logistic growth is:

\[
Y = \frac{Y_m}{1 + K \cdot e^{-(rgr\cdot t)}}
\]
Using the following data: \( Y_i = 100, Y_m = 10000, rgr = 0.5, \) and \( K = (Y_m - Y_i)/Y_i = 99, \) yields \( Y_t=5 = 1096. \)

In the remainder of the 11-year period the level of herbicide remains above 68 kg ha\(^{-1}\), and therefore beetles only die. The half-live period is 3.47 years, from which the relative death rate (rdr) follows as 0.2 yr\(^{-1}\). After 6 years (at \( t = 11 \) years) the number of beetles left are \( Y = 1096 \cdot e^{-0.2 \cdot 6} = 330 \) ha\(^{-1}\).

d. The time constants of the model are the inverse values of rgr and rdr, hence 2 and 5 years, respectively. In Exercise 43 the effective relative growth rate in logistic growth was shown to vary in time. This would result in selecting the variable time step integration method of Runge-Kutta-Simpson. However, since discontinuities occur in this program, the method of Euler should be used. The time increment, \( \Delta t \), on the basis of the 'rule of thumb' is then 0.2 years. It appears from the results that under these conditions the maximum number of beetles is underestimated by about 9% compared to the analytical solution (1096).

The CSMP simulation program is:

```
TITLE Beetles and herbicides
INITIAL
INCON IHERB =50.0, IBEET =100.
PARAM TCHERB=10.0, DRES =30.0, LHERB =68.0
PARAM RGR =0.50, RDR =0.20, BEEMAX=10000.0
PARAM FIRST =3.0 , NEXT =2.0
TIMER FINTIM=11.0, PRDEL =1.0 , DELT =0.2
OUTPUT HERB,BEET
PAGE WIDTH =80 , NTAB =0
PRINT HERB, BEET
METHOD RECT

DYNAMIC
HERB = INTRGL(IHERB,RHERB)
BEET = INTRGL(IBEET,RBEET)

RHERB = INPUT - HERB/TCHERB
INPUT = (DRES/DLT) * IMPULS(FIRST,NEXT)
RBEET = INSW(HERB-LHERB,GROWTH,DEATH)
GROWTH = RGR * BEET * (1.0 - BEET/BEEMAX)
DEATH =-RDR * BEET

END
STOP
ENDJOB
```

e. The CSMP simulation program for this model with the differential equation in a subroutine is:

```
TITLE Beetles and herbicides
INITIAL
HISTORY HERBI(4)
INCON IHERB =50.0, IBEET =100.
PARAM TCHERB=10.0, DRES =30.0, LHERB =68.0
PARAM RGR =0.50, RDR =0.20, BEEMAX=10000.0
PARAM FIRST =3.0 , NEXT =2.0
TIMER FINTIM=11.0, PRDEL =1.0 , DELT =0.2
OUTPUT HERB,BEET
PAGE WIDTH =80 , NTAB =0
PRINT HERB, BEET
METHOD RECT
```

224
Exercise 91  
  a. 40th order delay.  
  b. \( T = \frac{DTSYS}{N} \).  
  c. \( T = \frac{10}{40} = 0.25 \), therefore \( \Delta T = 1/10 \times 0.25 = 0.025 \) h is a first estimate of the time step.  
  d. \( N = \left( \frac{DTSYS}{\sigma} \right)^2 \). Therefore, the absolute dispersion, \( \sigma \), is \( DTSYS/N^{1/2} = 1.58 \) h and the relative dispersion is \( \sigma/DTSYS = (1/N)^{1/2} = 0.158 \).  
  e. The whole contents of \( A(1) \) eventually find its way to \( A(41) \). Therefore, \( A(41) \) becomes 1000.  
  f. The order of the delay is high and the total delay time is short. Therefore, the amount in the last integral will be normally distributed in time, and approximately 99% of \( A(1) \) will have found its way to \( A(41) \) in a \( \text{FINTIM} = DTSYS + 3 \times \sigma = 14.74 \) h.  

225
g. $A(1)$ decreases exponentially: $\frac{200}{1000} = e^{-t/0.25}$. Therefore, $t = -0.25 \cdot \ln 0.2 = 0.40$ hours.

h. The time course of the amount in integral I will intersect the time course in integral (I+1) when the latter is at its maximum. This can be seen as follows: when (I+1) is at its maximum, it means that the rate of inflow equals the rate of outflow. Consultation of the rate equations in the PROCEDURE shows that the contents of integral (I+1) must be equal then to the one in number I.

![Graph showing the amounts in A(1), A(2), and A(3) in time.](image)

The course of the amounts in $A(1)$, $A(2)$ and $A(3)$ in time.

i. $RA(1) = X - (A(1)/T)$

j. $RA(2) = 0$ (equilibrium) = $A(1)/T - A(2)/T$; $RA(1) = 0$ (equilibrium). Combining this information with the answer to i), it follows that $A(1) = X \cdot T$; $A(2) = X \cdot T = X \cdot 0.25$.

Eventually, all integrals except $A(41)$ reach this final level.

Exercise 92

a. The first datum required is the average germination period (or: delay time of the system; average residence time; total delay ($T_{total}$; Section 8.6)). The general (analytical) equation to calculate $T_{total}$ was provided in Section 2.5, Exercise 9, and an application was given in Section 8.8, Figure 27.

$$AGP = T_{total} = \frac{\Sigma (\Delta G \cdot \Delta t \cdot t)}{\Sigma (\Delta G \cdot \Delta t)} = \frac{2494}{249} = 10.016 \text{ h.}$$

The second datum required is the variance, $\sigma^2$, or the standard deviation (see Sections 8.6 and 8.8, Figure 27):

$$VAR = \sigma^2 = \frac{\Sigma (\Delta G \cdot \Delta t \cdot (t - AGP)^2)}{\Sigma (\Delta G \cdot \Delta t)}$$
These data may also roughly be read from the curve because it appears to be quite symmetrical. Then the $T_{\text{total}}$ may be estimated as 10, and to estimate $\sigma$ use may be made of the fact that more than 99% of the clock lies within the time period of $T_{\text{total}} \pm 3 \cdot \sigma$. Therefore: $3 \cdot \sigma \equiv 5$ and $\sigma \equiv 1.67$ hours.

b. $N = (T_{\text{total}} / \sigma)^2 = (10.016 / 1.573)^2 = 40.54$, thus 40 or 41. The rough estimate yields $(10 / 1.67)^2 = 35.86$, thus 36.

c. $\tau = T_{\text{total}} / N = 10.016 / 40$ or $10.016 / 41 = 0.250$ or 0.244 h. The rough estimate yields $(10 / 36) = 0.278$ h.

d. Use the method of the 'fractional boxcar train' (Section 8.7, Equation 51):

\[
F = 1 - N \cdot \left(\frac{\sigma}{T_{\text{total}}}\right)^2,
\]

with $N = 20$, $\sigma = 1.573$, and $T_{\text{total}} = 10.016$, this yields $F = 0.5067$.

Note that the germination curve in this exercise was derived from the program in Exercise 91.

Exercise 93

a. The average residence time of the bacteria in each vessel is equal to the ratio of the volume of the vessel and the flow rate, i.e. $V/Q = 0.1$ minute.

The average residence time in the system as a whole is the sum of the residence times in each of the vessels, i.e. $100 \cdot V/Q = 10$ minutes.

b. The graph giving the relationship between the rate of consumption of bacteria per organism (y-axis), and the ratio of the current concentration of the bacteria in the vessel and their maximum concentration is:

c. The rate equation for the first vessel:

\[
\frac{d\text{BACT}}{dt} = \text{IN} - \text{OUT} - \text{CONS}
\]

where:
IN = Q • BCONC

\[ \text{litre} \times \frac{\text{bacteria}}{\text{minute}} \times \frac{\text{litre}}{\text{bacteria}} = \frac{\text{bacteria}}{\text{minute}} \]

OUT = \( \frac{\text{BACT}}{\text{TAU}} \)

\[ \text{bacteria} \times \frac{\text{minute}}{\text{bacteria}} = \frac{\text{bacteria}}{\text{minute}} \]

CONS = BCNSF • ORGNSM • \( \frac{\text{BACT/VOL}}{\text{BCONC}} \)

\[ \frac{\text{bacteria}}{\text{organism} \times \text{minute}} \times \text{No. organisms} \times \frac{\text{bacteria}}{\text{litre}} = \frac{\text{bacteria}}{\text{minute}} \]

with TAU equal to (Volume of vessel)/(flow rate) = VOL/Q.

d. The average residence time of the bacteria is constant for all vessels, and the rate of the bacteria consumption is proportional to the number of bacteria. The ratio of the bacteria concentration in the incoming and outgoing water will be constant for each vessel after a prolonged period of time, because an equilibrium will be established. In that situation the net rates are zero.

Substitution of the data in the differential equation of c) yields the following numerical result:

\[ \frac{dBACT}{dt} = IN - OUT - CONS = 1 \times 10^6 - \frac{\text{BACT}}{1/10} - 10^5 \times 1 \times \frac{\text{BACT/(1/10)}}{10^6} = 0.0 \]

\[ \frac{dBACT}{dt} = 10^6 - 10 \times \text{BACT} - \text{BACT} = 10^6 - 11 \times \text{BACT} = 0.0 \]

The bacteria concentration in the outgoing water is now:

\[ \text{BACT/VOL} = \frac{(10^6/11)}{(1/10)} = \frac{10^6}{10/11} \]

and the ratio of the concentrations is \( \frac{10^6}{(10/11) \times 10^6} = 11/10 \).

e. The ratio between the bacteria concentrations in the incoming and outgoing water of a vessel will be 10/11. This holds for each vessel.

A concentration of (10/11) • (inflow into the first vessel) flows out of the first vessel. A concentration of (10/11) • (inflow into the second vessel) flows out of the second vessel, or (10/11) • (10/11) • (inflow into the first vessel) = (10/11)^2 • (inflow into the first vessel).

A concentration of (10/11)^100 • (inflow into the first vessel) will flow from the 100th vessel.

Therefore, the ratio between the concentration in the outgoing water of the 100th vessel and that of the incoming water in the first vessel is (10/11)^100 = 7.257 \times 10^{-5}. This means that at the end of the series of vessels, approximately 72 bacteria per litre are left in the system.

f. A number of similarities between the system presented here and that of Exercise 41 are:

- The inflow of polluted river water corresponds to the inflow of bacteria with the water;
- In both cases, the volumes (of the lake and of the vessel, respectively) remain constant;
- Both cases are examples of exponential delays;
- The decomposition of the pollutant corresponds to the consumption of the bacteria by the organism: both are
proportional to the amount of material present.

The decomposition constant of the pollutant in the lake \( (k) \) corresponds to the constant:

\[
\frac{BCNSF \cdot ORGNSM \cdot \frac{1}{\text{VOL}}}{BCONC} = 10^5 \cdot 1 \cdot \frac{1/(1/10)}{10^5} = 1.0 \text{ (minute}^{-1})
\]

in the rate equation of the bacteria in c).

Some similarities between the system presented here and that of Equation 52 of Subsection 9.2.1, in which the diffusion coefficient \((D)\) is set to zero, are:

- The inflow of bacteria with the water corresponds to the inflow of dissolved fertilizer with the rain water;
- Consumption of the bacteria by organisms corresponds to \(R\) (the rate of uptake of nutrients by plant roots). However, no additional information on \(R\) is supplied. It could be a term proportional to the concentration (1st order process like exponential growth or decrease), but it could also be formulated independent of concentration (as a zero order process).
- The outflow of bacteria (OUT) and of fertilizer have been set proportional to the amount and the concentration in the preceding vessel or layer, respectively. The 'breakthrough curves' introduced in this way (Chapter 8) or the numerical dispersion (Chapter 9) is an artefact in the case of the fertilizer. For the vessels, however, that are physically separated, the formulation is physically correct if they are properly stirred; in fact they form a cascade.

\(g\). The time increment must be based on the smallest time coefficient. The time coefficient or the average residence time of the bacteria in the series of vessels is determined both by the rate of outflow via the water and by the rate of consumption by the organisms (compare f and g of Exercise 41).

Reformulation of the differential equation yields:

\[
\frac{d\text{BACT}}{dt} = Q \cdot BCONC - \frac{\text{BACT}}{\text{TAU}} - \frac{BCNSF \cdot ORGNSM \cdot \text{BACT/VOL}}{BCONC}
\]

\[
= Q \cdot BCONC - \frac{\text{BACT}}{\text{TAU}} \cdot (\frac{1}{\text{TAU}} + \frac{BCNSF \cdot ORGNSM \cdot \frac{1}{\text{VOL}}}{BCONC}).
\]

The time coefficient can thus be calculated from \((\frac{1}{\text{TAU}} + \frac{BCNSF \cdot ORGNSM \cdot \frac{1}{\text{VOL}}}{BCONC})^{-1}\) as \((\frac{1}{0.1} + 1.0)^{-1}\) or 0.0909 minutes.

If integration method RECT would be used, the time increment should be set at 0.1 of the time coefficient, i.e. 0.00909, resulting in a practical value of 0.005 minutes.

\(h\). The best integration method in this continuous model is RKSFX, since the time coefficient does not vary in the course of the simulation.

- The time increment now becomes \((1/2) \cdot 0.09 = 0.045\), or practically 0.02 minutes.
- In the TERMINAL segment of a program, the equations must always be in the correct order, as the CSMP program does not sort there. The ratio of the concentration in the outgoing water of the last vessel and the ingoing water of the first vessel was calculated in \(e\) and can be programmed using

\[
\text{OUTIN} = (\text{BACT}(100)/\text{VOL})/\text{BCONC}.
\]

In CSMP, use should be made of the FORMAT statement:

```
WRITE(6,40) TIME,OUTIN
40 FORMAT('TIME=',E9.4,'OUTIN=',E9.4)
```
since the notation in which the format is incorporated in the WRITE-statement:

```
WRITE(6,'(1X,A,E9.4,A,E9.4)') 'TIME=',TIME,'OUTIN=',OUTIN
```

is not accepted by the language. However, this notation can be applied in FORTRAN subroutines, that are invoked by CSMP.

The CSMP simulation program for this model is:

```
TITLE Bacteria and organisms
INITIAL
FIXED I,N
STORAGE FLOW(101),CONS(100)
TABLE IBACT(1-100)=100*0.0
PARAM ORGNSM=1.0 , Q =1.0 , VOL =0.1 , N =100
PARAM BCNSF =1.0E5 , BCONC =1.0E6
TIMER FINTIM =15.0 , OUTDEL=0.25 , PRDEL=0.5 , DELT=0.02
PAGE WIDTH =80 , NTAB =0
OUTPUT NBACT(50),NBACT(100)
OUTPUT BACT(1) ,BACT(50),BACT(100)
PRINT NBACT(1) ,BACT(1) ,NBACT(50),BACT(50),NBACT(100),BACT(100)
METHOD RKSFX

* Calculations
TAU =VOL/Q

DYNAMIC
BACT =INTGRL(IBACT,NBACT,100)

PROCEDURE NBACT=NETROC(FLOW,CONS)
* Calculation of NET Rates Of Change in each vessel.
  DO 10 I=1,N
     NBACT(I)=FLOW(I)-FLOW(I+1)-CONS(I)
 10 CONTINUE
ENDPRO

PROCEDURE FLOW=INDFLO(BCONC,Q,BACT,TAU)
* Calculation of INDIVidual FLOws from one vessel to the other.
  FLOW(1) =BCONC*Q
  DO 20 I=2,N+1
     FLOW(I) =BACT(I-1)/TAU
 20 CONTINUE
ENDPRO

PROCEDURE CONS=CONSUM(BCNSF,ORGNSM,BACT,VOL,BCONC)
* Calculation of CONSUMption terms in each vessel.
  DO 30 I=1,N
     CONS(I) =BCNSF * ORGNSM * BACT(I)/(VOL * BCONC)
 30 CONTINUE
ENDPRO

TERMINAL
OUTIN =BACT(100)/VOL) / BCONC
WRITE(6,40) TIME,OUTIN
40 FORMAT('TIME=',E9.4,'OUTIN=',E9.4)
END
STOP
ENDJOB
```
The CSMP simulation program for this model with the equations in a subroutine is:

```
TITLE Bacteria and organisms
INITIAL
FIXED I,N
STORAGE FLOW(101),CONS(100)
TABLE IBACT(1-100)=100*0.0
PARAM ORGNSM =1.0 , Q =1.0 , VOL =0.1, N =100
PARAM BCNSF =1.0E5, BCONC =1.0E6
TIMER FINTIM =15.0 , OUTDEL=0.25 , PRDEL=0.5, DELT=0.02
PAGE WIDTH =80 , NTAB =0
OUTPUT NBACT(50),NBACT(100)
PAGE GROUP
OUTPUT BACT(1) ,BACT(50),BACT(100)
PRINT NBACT(1) ,BACT(1) ,NBACT(50),BACT(50),NBACT(100),BACT(100)
METHOD RKSFX

* Calculations
TAU =VOL/Q

DYNAMIC
NOSORT
BACT =INTGRL(IBACT,NBACT,100)

CALL INDFLO(N,BCONC,Q,BACT,TAU,FLOW)

CALL CONSUM(N,BCNSF,ORGNSM,BACT,VOL,BCONC,CONS)

CALL NETROC(N,FLOW,CONS,NBACT)

TERMINAL
OUTIN =(BACT(100)/VOL) / BCONC
WRITE(6,40) TIME,OUTIN
40 FORMAT(' TIME=',E9.4,' OUTIN=',E9.4 )

END
STOP

*********************************************************************
SUBROUTINE INDFLO(N,BCONC,Q,BACT,TAU,FLOW)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION BACT(N),FLOW(N+1)
SAVE

* Calculation of INDividual FLOws from one vessel to the other.
FLOW(1) =BCONC*Q
DO 10 I=2,N+1
     FLOW(I)=BACT(I-1)/TAU
10 CONTINUE
RETURN
END

*********************************************************************

231
```
SUBROUTINE CONSUM(N,BCNSF,ORGNSM,BACT,VOL,BCONC,CONS)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION BACT(N),CONS(N)
SAVE

* Calculation of CONSUMption terms in each vessel.
DO 10 I=1,N
    CONS(I)=BCNSF * ORGNSM * BACT(I)/(VOL * BCONC)
10 CONTINUE
RETURN
END

*********************************************************************
SUBROUTINE NETROC(N,FLOW,CONS,NBACT)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION FLOW(N+1),CONS(N),NBACT(100)
SAVE

* Calculation of NET Rates Of Change in each vessel.
DO 10 I=1,N
    NBACT(I)=FLOW(I)-FLOW(I+1)-CONS(I)
10 CONTINUE
RETURN
END

*********************************************************************
ENDJOB

j. For the situation with the 100 organisms in one vessel, the concentration of bacteria in the outgoing water after a prolonged period is calculated similarly to that in d):

\[
\frac{dBACT}{dt} = \text{IN} - \text{OUT} - \text{CONS} = 1\times10^6 - \frac{BACT}{10} - 10^5 \times 100 \times \frac{BACT(10)}{10^6} = 10^6 - 1.1 \times BACT = 0.0.
\]

Hence, the concentration of bacteria in the outgoing water is: \(\text{BACT/VOL} = (10^6/1.1)/10 = (1/11) \times 10^6 = 0.909 \times 10^5 \text{ bacteria litre}^{-1}\).

k. The ratio of the concentrations in the outgoing and incoming water of the large vessel is now \((1/11) \times 10^6/10^6 = 1/11 = 0.090909\). For the series of vessels, this ratio was 0.00007257. Hence, the ratio out/in for the large vessel is much higher than for the 100 vessels in series, even though the total volume and the total number of organisms are identical. Consumption of the bacteria by the organisms in the large vessel is thus far less efficient than in the series of vessels.

Exercise 94
a. The crop development rate is \((80-50)/20 = 1.5\) units \(d^{-1}\) at 20 °C, and zero at 7 °C. The rate at 15 °C can be determined via linear interpolation from \(y = y_1 + (x-x_1) \times (y_2-y_1) / (x_2-x_1)\), in which \(x\) is the independent variable and \((x_1,y_1)\) and \((x_2,y_2)\) are the coordinate pairs (7, 0) and (20, 1.5). This results in a crop development rate of 0.9231 units \(d^{-1}\). The crop development stage on the 1st, 13th, and 20th of July, i.e. 11, 23 and 30 days after 20 June, respectively, is therefore:
b. Calculation of the aphid population density on the same dates:

Until flowering (crop stage 60), the aphids have a doubling time of 4 days. The relative growth rate, $r_{gr}$, is therefore $0.1733 \, \text{d}^{-1}$. After 11 days of exponential growth with an initial density of 0.1, the number of aphids per tiller will be $0.1 \times e^{(0.1733 \times 11)} = 0.67$.

After flowering, the doubling time of the aphids is 2.5 days. The relative growth rate, $r_{gr}$, is then $0.2773 \, \text{d}^{-1}$. After 12 days of exponential growth with an initial density of 0.67, the number of aphids found per tiller will be $0.67 \times e^{(0.2773 \times 12)} = 18.75$.

A summary of the calculated results is:

<table>
<thead>
<tr>
<th>Date</th>
<th>Crop Stage</th>
<th>Number of Aphids per Tillar</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 June</td>
<td>50</td>
<td>0.1</td>
</tr>
<tr>
<td>1 July</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>13 July</td>
<td>71.23</td>
<td>18.75</td>
</tr>
<tr>
<td>20 July</td>
<td>77.69</td>
<td>130.64</td>
</tr>
</tbody>
</table>

c. The damage threshold was established at 18 aphids per tiller. This level is thus reached in the course of July 13.

d. A relational diagram for this system is:
The time coefficient and the time step of the model are determined by:
- the highest $r_{gr}$ attained ($0.2773 \, \text{d}^{-1}$). The time coefficient is thus 3.61 d. The time increment for method RECT is therefore 0.36 d, yielding for practical purposes 0.25 or 0.1 d.
- Temperature in the course of the day should be neatly read. The time constant of a sine can be calculated as the inverse of the angular frequency (Exercise 25d): $\tau = \frac{1}{\omega} = \frac{t_c}{2\pi} = 0.16 \, \text{d}$. This would yield a time step of integration of 0.016 d, or, say, 0.02 d. Since this time increment is smaller than the one resulting from the relative growth rate, the time increment is determined by the requirement of following the temperature cycle. It remains to be concluded from trial and error to see whether the RCD function, a forcing function or driving variable affected by the temperature, is neatly followed.

f. The program should be written in NOSORT, i.e. the calculations must be in the correct calculational sequence.

A sinusoidal temperature wave over the day can be obtained by incorporating into the program:

$$\text{TEMP} = \text{AVTMP} + \text{AMPTMP} \times \sin(6.2832 \times \text{TIME}).$$

Note that TIME is expressed in days and that therefore $2\pi$ radians should pass in one day. By setting parameter AMPTMP to zero, a constant temperature AVTMP is obtained; other values for AMPTMP yield sinusoidal waves.

Integration method RKS can be used, since there are no divisions by DELT in the model. The marked change in $r_{gr}$ at crop stage 60 is taken into account (automatically) by using a small time increment then. This can be investigated by including the line 'IF (KEEP .EQ. 1) CALL PRINT', by assigning the value of FINTIM to the PRDEL and OUTDEL, and by including DELT on the PRINT label.

The CSMP simulation program for this model is:

```
TITLE Sitobion in wheat
INITIAL
INCON ISITOB = 0.1, ICDST = 50.0
PARAM AVTMP = 15.0, AMPTMP= 0.0
FUNCTION RCDTB = 0.0,0.0 , 7.0,0.0 , 20.0,1.5
FUNCTION RSITB = 0.0,0.1733, 59.99,0.1733, 60.0,0.2773, 100.,0.2773
TIMER FINTIM = 30.0, OUTDEL= 1.0, PRDEL= 1.0
OUTPUT SITOB, CDST, RGRSTB
PAGE WIDTH = 80, NTAB = 0
PRINT SITOB, CDST, RGRSTB
METHOD RKS
DYNAMIC
NOSORT
SITOB =INTGRL(ISITOB,RSITB)
CDST =INTGRL(ICDST ,RCD )
```
* Description of rate of crop development (RCD).
TEMP =AVTMP + AMPTMP * SIN(6.2832*TIME)
RCD =AFGEN(RCDTB,TEMP)

* Description of rate of growth of Sitobion (RSITOB).
RGRSTB =AFGEN(RSITB,CDST)
RSITOB =RGRSTB * SITOB

END
STOP
ENDJOB

g. The CSMP simulation program for this model with the equations in a subroutine is:

TITLE Sitobion in wheat
INITIAL
HISTORY RATES(10)
INCON ISITOB = 0.1 , ICDST = 50.0
PARAM AVTMP = 15.0, AMPTMP = 0.0
FUNCTION RCDTB = 0.0, 0.0, 7.0, 0.0 , 20.0, 1.5
FUNCTION RSITB = 0.0, 0.1733, 59.99, 0.1733, 60.0, 0.2773,...
100. , 0.2773
TIMER FINTIM = 30.0, OUTDEL= 1.0, PRDEL= 1.0
OUTPUT SITOB, CDST, RGRSTB
PAGE WIDTH = 80 , NTAB = 0
PRINT SITOB, CDST, RGRSTB
METHOD RKS
DYNAMIC
NOSORT
SITOB =INTGRL(ISITOB,RSITOB)
CDST =INTGRL(ICDST ,RCD)
CALL RATES(AVTMP,AMPTMP,TIME,RCDTB,RSITB,CDST,...
SITOB,TEMP,RCD,RGRSIT,RSITOB)
RGRSTB =RGRSIT

END
STOP

***********************************************************************
SUBROUTINE RATES(NLOC,AVTMP,AMPTMP,TIME,RCDTB,RSITB,CDST, 
RSITOB,TEMP,RCD,RGRSTB,RSITOB)

* Description of rate of crop development (RCD).
TEMP =AVTMP + AMPTMP * SIN(6.2832*TIME)
RCD =AFGEN(NLOC,RCDTB,TEMP)

* Description of rate of growth of Sitobion (RSITOB).
RGRSTB =AFGEN(NLOC+5,RSITB,CDST)
RSITOB =RGRSTB * SITOB
RETURN
END

***********************************************************************
ENDJOB
Using the function AFGEN twice in the RATES subroutine implies that 10 memory places must be reserved via the HISTORY label.

The variable RGRSTB is calculated in the subroutine. To print the variable by the PRINT label, it must be included in the CSMP common area. Variables are automatically included in the CSMP common only, if the requirements specified in Section 7.3.4 are met; therefore, under the current conditions RGRSTB will not be included automatically in the CSMP common. To achieve that, the line RGRSTB=RGRSIT is incorporated after the subroutine call. The CALL itself contains RGRSIT (an auxiliary variable) at the position of RGRSTB in the subroutine definition. (The variable RGRSTB could also have been printed directly using the FORTRAN WRITE statement; then, the auxiliary variable RGRSIT would not have been necessary.)

Variable TEMP also can not be printed directly using PRINT. However, printing is possible for the variables RCD and RSITOB.

Exercise 95  

a. Relational diagram

b. The rate equation for the top layer of the first lake is:

\[
\frac{dM_{T1}}{dt} = Q \cdot CP - MC \cdot M_{T1} - RDR1 \cdot M_{T1} - \frac{M_{T1}}{TC1}
\]

Substituting the expression for TC1 and placing MT1 outside the brackets yields:

\[
\frac{dM_{T1}}{dt} = Q \cdot CP - MC \cdot M_{T1} \cdot (SC + RDR1 + \frac{Q}{VT1})
\]

The rate equation for the lower layer of the first lake is:
dML1 = SC • MT1

dM2 = MT1/TC1 - RDR2 • M2 - M2/TC2

Substituting the expressions for TC1 and TC2, and placing M2 outside the brackets yields:

dM2 = MT1 • Q - M2 • (RDR2 + Q)

VT1 V2

c. The concentrations at equilibrium can be calculated from the differential equations if the rates are set to zero.

Lake number one, top layer:

Q • CP = MT1 • (SC + RDR1 + Q)

VT1

MT1 = 

Q • CP

(SC + RDR1 + Q) 0.008 + 0.03 + \frac{5.10^7}{625.10^7} = 2.173913043.10^{10} \text{ kg.}

The concentration will then be \frac{MT1}{VT1} = 3.478 \text{ kg m}^{-3}.

Lake number two:

MT1 • Q = M2 • (RDR2 + Q)

VT1 V2

M2 = \frac{MT1 • Q}{VT1} = 1.647597254 10^9 \text{ kg.}

(RDR2 + Q)

V2

The concentration will then be \frac{M2}{V2} = 1.831 \text{ kg m}^{-3}.

d. The average residence time of a pollutant particle in the top layer of lake number one can be calculated from the differential equation derived in b):

\frac{dMT1}{dt} = Q • CP - MT1 • (SC + RDR1 + Q)

VT1

The time constant of the pollutant particle is

1

SC+RDR1+Q

0.008 + 0.03 + 0.008 = 21.74 d.

At equilibrium, MT1 = 2.173913043.10^{10} \text{ kg. Therefore, } \frac{dML1}{dt} = 173913043.4 \text{ kg d}^{-1}. 237
Sedimentation will occur homogeneously over the surface and 'falling' from the well-mixed top layer. The surface area of the sediment that will precipitate in kg per m² and per day is:

\[ \frac{dM_{L1}}{dt} = \frac{6.957 \, \text{kg} \, \text{m}^{-2} \, \text{d}^{-1}}{SA_{1}} \]

The density of the sediment (DS) was given as 1300 kg m⁻³. be:

\[ \frac{dM_{L1}}{dt} \cdot \frac{1}{SA_{1} \cdot DS} = 6.957 \cdot 5.3515 \times 10^{-3} \, \text{m} \, \text{d}^{-1}, \]

and 10 mm sediment will precipitate in:

\[ \frac{0.01}{5.3515 \times 10^{-3}} = 1.87 \]

**Title: Lake pollution**

**INITIAL**

**** See remark 1.

INCON IMT1 = 0., IML1 = 0., IM2 = 0.,

PARAM Q = 5.E-7, CP = 20., SC = 0.008, 1

PARAM VT1 = 625.E-7, RDR2 = 0.05, V2 = 90.E-7

PARAM SA1 = 25.E-6, DS = 1300.

**** See remark 2.

TIMER FINTIM = 1.E5, DELT = 5., PRDEL = 10.

PRINT MT1CON, M2CON, SEDL, M1L1

METHOD RKSFx

* Calculation

M2STOP = 0.95*1.647597E9

**DYNAMIC**

MT1 = INTGRL (IMT1, RMT1)
ML1 = INTGRL (IML1, RML1)
M2 = INTGRL (IM2, RM2)
SEDL = INTGRL (ISED1, RSEDL)

RMT1 = Q*CP - MT1*(SC + RDR1 + Q/VT1)
RML1 = MT1*SC
RM2 = MT1*Q/VT1 - M2*(RDR2 + Q/V2)
RSEDL = RML1/(SA1*DS)
MT1CON = MT1/VT1
M2CON = M2/V2

FINISH M2 = M2STOP

END

STOP

END JOB

Remark 1: The initial amount of pollutant is zero since in q when t = 0.

Remark 2: Estimation of DELT.

The residence time of a pollutant particle in the first
Let us assume that there is no supply of pollutant t
\( \frac{dM_2}{dt} = -M_2 \cdot (RDR_2 + \frac{Q}{V_2}) \)

Here, the time coefficient is:

\[
TC = \frac{1}{(RDR_2 + \frac{Q}{V_2})} = 9.4737 \text{ days.}
\]

On this basis, a DELT of 9.4737/2 \(=\) 5 days could be selected, since for method RKSFX, \(\Delta t \leq 1/2 \times TC\) may be chosen.

- FINTIM = 1.65 days is very high. However, because a FINISH condition has been included, the simulation will stop earlier. This means that FINISH should not be forgotten, otherwise long computer times would be used to produce insignificant output.

**THEREFORE:** always check your programs carefully before running them.

g1. Equilibrium means that the net rates of change are zero. It is reached after a long period: \( t \rightarrow \infty \).

For the first lake this means: \( t \rightarrow \infty \), \( e^{-t/TC} \rightarrow 0 \), \( MT_1 \rightarrow IN \times TC \), which is identical to the equation in answer c) after substituting \( IN \) and \( TC \).

For the second lake this means \( t \rightarrow \infty \), \( M_2 = IN \times TC \times TC_2 \), which, upon substitution, produces the equation in answer c).

g2. Write \( t \) explicitly: \( t = -TC \cdot \ln\left(1 - \frac{0.95 \cdot MT_1}{IN \times TC}\right) = 65.12 \text{ days.} \) After a little experimentation, it is possible to find \( t_{05} = 77.47 \text{ days for lake number two.} \)

*Derivation of analytical solution as presented in question g) for the time course of pollution in two lakes in series:*

**Conditions:**

- \( t = 0 \) no pollution in lake.
- \( t > 0 \) - constant supply of pollution with a rate: (discharge of river \((m^3)\) \times concentration of pollution \((\text{kg})\)) \(=\) \(\text{kg} \)
- constant outflow of pollution by:
  - a proportional outflow of perfectly mixed pollution plus water, with a relative outflow rate of:
    \[ \frac{Q}{VT_1} \text{ with units } \frac{m^3}{d} = d^{-1}; \]
  - b. a proportional sedimentation with a constant \(SC\): \(d^{-1}\) (only for lake number one);
  - c. a proportional decomposition with constant \(RDR_1\): \(d^{-1}\).

The combined \(TC\) for lake number one can be given as:

\[
TC = \frac{1}{RDR_1 + SC + \frac{Q}{VT_1}} \text{ (day).}
\]

**Lake number one**

The rate equation for the pollution in lake number one is:

\[
\frac{dMT_1}{dt} = IN - \frac{MT_1}{TC} \quad \text{where } IN = \frac{Q \cdot CP}{\frac{m^3 \times \text{kg}}{d \times m^3 \times d}} = \frac{\text{kg}}{d}
\]
Elaboration:

\[
\frac{dM_{11}}{dt} + \frac{M_{11}}{TC} = IN.
\]

Multiply both sides of the equation with the integration factor \(e^{\frac{MT1}{TC}}\), to make it possible to integrate the left-hand side of the equation. This results in:

\[
\frac{dM_{11}}{dt} \cdot e^{\frac{MT1}{TC}} + \frac{M_{11}}{TC} \cdot e^{\frac{MT1}{TC}} = IN \cdot e^{\frac{MT1}{TC}}.
\]

The sum of the terms to the left of the equal-sign is the result of \(\frac{d}{dt} (MT1 \cdot e^{\frac{MT1}{TC}})\). Therefore, \(\frac{d}{dt} (MT1 \cdot e^{\frac{MT1}{TC}}) = IN \cdot e^{\frac{MT1}{TC}}\), and it follows that \(d (MT1 \cdot e^{\frac{MT1}{TC}}) = IN \cdot e^{\frac{MT1}{TC}} \cdot dt\), which after integration gives:

\[
MT1 \cdot e^{\frac{MT1}{TC}} = IN \cdot TC \cdot e^{\frac{MT1}{TC}} + Q.
\]

Determination of the integration constant \(Q\):

for \(t=0\), \(MT1 = 0\), substitution gives: \(0 = IN \cdot TC + Q\). Therefore, \(Q = -IN \cdot TC\).

Substituting \(Q\), dividing by \(e^{\frac{MT1}{TC}}\), and taking \(IN \cdot TC\) outside the brackets gives:

\[
MT1 = IN \cdot TC \cdot (1 - e^{-\frac{MT1}{TC}}),
\]

where \(IN = Q \cdot CP\)

\(TC = 1/(RDR1 + SC + Q/VT1)\).

**Lake number two**

The rate equation for the pollution in lake number two is:

\[
\frac{dM_{22}}{dt} = MT1 - \frac{M_{22}}{TC1 TC2}.
\]

- The amount of pollution flowing from lake 1 into lake 2 depends on the discharge of the river and the concentration of the pollution in lake one, i.e. \(TC1 = 1/(Q/VT1)\).
- The residence time of the material in lake number 2 is a function of decomposition and outflow, both of which are proportional to the amount of pollution in the lake. Therefore, \(TC2 = 1/(RDR2 + Q/V2)\).
- The TC which is found in the equation for \(MT1\) remains: \(1/(RDR1 + SC + Q/VT1)\).

Substituting \(MT1\) in the rate equation gives:

\[
\frac{dM_{22}}{dt} + \frac{M_{22}}{TC2} = \frac{IN \cdot TC \cdot (1 - e^{-\frac{MT1}{TC}})}{TC1}.
\]

We again multiply by \(e^{\frac{MT1}{TC2}}\) to make it possible to integrate the equation:

\[
\frac{dM_{22}}{dt} \cdot e^{\frac{MT1}{TC2}} + \frac{M_{22}}{TC2} \cdot e^{\frac{MT1}{TC2}} = \frac{IN \cdot TC \cdot (1 - e^{-\frac{MT1}{TC}}) \cdot e^{\frac{MT1}{TC2}}}{TC1}.
\]

Integration proceeds as follows:

\[
M_{22} \cdot e^{\frac{MT1}{TC2}} = \int \frac{IN \cdot TC \cdot e^{\frac{MT1}{TC2}} \cdot dt}{TC1} - \int \frac{IN \cdot TC \cdot e^{-\frac{MT1}{TC}} \cdot e^{\frac{MT1}{TC2}} \cdot dt}{TC1}.
\]
\[
M_2 \cdot e^{\frac{1}{T_C}} = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot e^{\frac{1}{T_C}} - \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot \int e^{\left(\frac{1}{T_{C2}} - \frac{1}{T_C}\right) dt}
\]

\[
M_2 \cdot e^{\frac{1}{T_C}} = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot e^{\frac{1}{T_C}} - \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot \frac{T_C}{T_{C2}} \cdot e^{\left(\frac{1}{T_{C2}} - \frac{1}{T_C}\right)} + Q.
\]

Determination of the integration constant Q:

at \( t = 0, M_2 = 0 \). Substitution gives:

\[
0 = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} - \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot \frac{T_C}{T_{C2}} + Q, \text{ from which } Q \text{ can be calculated.}
\]

Substitute Q, move \( e^{\frac{1}{T_C}} \) to the right, and take \( \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \) outside the brackets to obtain:

\[
M_2 = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot \left\{ 1 - \frac{T_C}{T_{C1}} \cdot e^{\left(\frac{1}{T_{C2}} - \frac{1}{T_C}\right)} \cdot e^{\frac{1}{T_C}} + \frac{T_C}{T_{C1}} \cdot e^{\frac{1}{T_C}} - e^{\frac{1}{T_{C2}}} \right\}
\]

or

\[
M_2 = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot \left\{ 1 - \frac{T_C}{T_{C1}} \cdot e^{\frac{1}{T_C}} + e^{\frac{1}{T_C}} \cdot \left(\frac{T_C}{T_{C1}} - 1\right) \right\}.
\]

Since \( \frac{T_C}{T_{C2}} - 1 = \frac{T_C}{T_{C1}} - \frac{T_C}{T_{C2}} = \frac{T_{C2}}{T_{C1}} \), \( M_2 \) can be written as:

\[
M_2 = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot \left\{ 1 - \frac{T_C}{T_{C1}} \cdot e^{\frac{1}{T_C}} + \frac{T_C}{T_{C1}} \cdot e^{\frac{1}{T_C}} \right\}
\]

or

\[
M_2 = \text{IN} \cdot \frac{T_C}{T_{C1}} \cdot T_{C2} \cdot \left\{ 1 - \frac{T_{C2}}{T_{C1}} \cdot (T_C \cdot e^{\frac{1}{T_C}} - T_{C2} \cdot e^{\frac{1}{T_{C2}}}) \right\},
\]

where:\n\( \text{IN} = Q \cdot CP \)
\( T_C = \frac{1}{(RDR1 + SC + Q/VT1)} \)
\( T_{C1} = \frac{1}{Q/VT1} \)
\( T_{C2} = \frac{1}{(RDR2 + Q/V2)} \)

Finally: \( MT1_{\text{CON}} = MT1/VT1 \) and \( M2_{\text{CON}} = M2/V2 \).

By means of these equations, the concentrations of pollution in both lakes can be calculated as a function of time. Results of \( MT1_{\text{CON}} \) and \( M2_{\text{CON}} \) as a function of time are given in the figure below using the numerical values indicated in the exercise.
Exercise 96

a. Units:
- \( \text{VHTC} \equiv J \)
- \( \text{FL} \equiv J \text{ s}^{-1} \)
- \( \text{COND (AVCOND)} \equiv J \text{ cm}^{-1} \text{ s}^{-1} \text{ oC}^{-1} \)
- \( \text{VHCAP} \equiv J \text{ cm}^{-3} \text{ oC}^{-1} \).

b. The thickness and the surface area of each layer is 1 cm and 1 cm\(^2\), respectively. Since multiplication and division by one does not change the numerical value, this can be omitted. However, this method of working is not recommended because:
- the model is more difficult to read;
- the model is more difficult to change, and it is particularly easy to make errors, because not all the parameters are explicitly given;
- the units would no longer seem to be correct. For example, the calculation of \( \text{TEMP}(I) \):

\[
\text{TEMP} = \frac{\text{VHTC}(I)}{\text{VHCAP}(I)} \equiv \frac{J}{\text{cm}^3 \text{ oC}} = \frac{\text{J}}{\text{cm}^3 \text{ oC}}
\]

However, since there is still a division by the volume of 1 cm\(^3\), the dimension of \( \text{TEMP}(I) \) is indeed \( \text{oC} \). The correct way of working was demonstrated in Figure 38 (Subsection 9.3.3) and in Exercise 84 (Subsection 9.3.4).

c. The derivation of the expression for \( \text{AVCOND} \) was given in Appendix 5.

d. \( \text{STORAGE COND(25), VHCAP(25)} \)

<table>
<thead>
<tr>
<th>TABLE</th>
<th>COND(1-25)</th>
<th>5<em>8.8E-2, 20</em>2.9E-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>VHCAP(1-25)</td>
<td>25*2.0</td>
</tr>
</tbody>
</table>
Exercise 97

a. \( P, P_{\text{max}}: \text{tons d}^{-1}, \) TC: \( d. \) Therefore, \( \frac{dP}{dt}: \text{tons d}^{-2}. \) This is the acceleration with which the maximum production level is reached. This acceleration equals zero if the maximum level, \( P_{\text{max}} \), is reached.

b. Equation 79 can be integrated analogously to the problem with the water tank:

\[
P = P_{\text{max}} \cdot (1 - e^{-\sqrt{TC}}).
\]

The production rate at \( t=0, P_0 \), does not occur in the equation, because it will be zero when starting at \( t = 0. \) After 5 days \( P = 189.64 \) tons \( \cdot d^{-1}. \)

c. TITLE Stock control

INITIAL

INCON IS =1000.0
PARAM PMAX=300.0 , TC=5.0
FUNCTION SRTB=0.0,50.0, 0.9,50.0, 1.0,100., 2.9,100., ...
3.0,150., 3.9,150., 4.0,200., 4.9,200., ...
5.0,0.00, 7.1,0.
TIMER FINTIM=56.0, PRDEL=0.1, OUTDEL=0.5, DELT=0.1
PRINT S,P,SR,TAC
OUTPUT S,P,SR
PAGE WIDTH =80 , NTAB=0
METHOD RECT

DYNAMIC

P =INTGRL(0.,DPDT)
S =INTGRL(IS,DSDT)
TAC =INTGRL(0.,RTAC)

DPDT =INSW( 500.0-S, -(P/DELT), (PMAX-P)/TC )
DSDT =P-SR
SR =AFGEN(SRTB,TAC)
RTAC =1.0-(7.0/DELT)*PUSH
PUSH =INSW( TAC-7.0, 0.0, 1.0 )

END
STOP
ENDJOB

- Method RECT is used because the program contains a discontinuity triggered by the amount of concentrate stored.
- \( \Delta t \) is taken as 0.1 instead of 0.5 (= TC/10), so that the FUNCTION SRTB can be easily followed.
- \( \text{FINTIM} = 56 \) days, because the time is expressed in days and the production manager wishes to make calculations for 8 weeks.
- The last day number in FUNCTION SRTB is 7.1 instead of 7.0, because TAC (a mnemonic for \( t \), a help time that is reset after each week) is finally one DELT larger than 7.0. If 7.0 would be used in the function an error message would result at the end of each week. This error message would be given in the CSMP-file containing the results of the model.
- The rate DPDT is defined as \(-P/DELT\) if \( S \) is larger than 500 tons, to fully interrupt the production rate.

d. This message implies that the storage space has no influence on the rate of change of the stock. Therefore, the system has no spatial limitation. The influence of the outside world on the model is imposed by the function of the sales rate, SRTB. This type of function is called a forcing function or a driving variable. In this case, the function has the dimension of a rate. However, forcing functions can also have other dimensions.
13 References


Leffelaar, P.A., E.W. Wolbeer & R.T. Dierkx, 1986. Some hints to write more readable simulation programs by the combined use of CSMP and FORTRAN- subroutines. Simulation Report CABO-TT 9,
Centre for Agrobiological Research, P.O. Box 14, 6700 AA Wageningen, 34 pp.
Appendix 1

Derivation of the relative error in the rectangular integration method for a given exponential rate curve given as a function of time (i.e. integration without feedback).

Cancelling $v_0\Delta t$ from Equation 36 yields:

$$E_{rel,1} = \frac{1}{2} - \frac{1}{2} e^{c\Delta t}$$

The formula for the Taylor expansion is:

$$f(x) = f(x_1) + f'(x_1)(x-x_1) + \frac{f''(x_1)}{2!}(x-x_1)^2 + \frac{f'''(x_1)}{3!}(x-x_1)^3 + ...$$

where $x_1$ indicates the reference point, with regard to which the function of $x$, $f(x)$, is expanded; $(x-x_1)$ is the distance between the value of $x$, for which the function is being calculated, and $x_1$; and $f'(x_1)$, $f''(x_1)$, etc. are the first, second and higher derivatives of the function of $f(x)$ with respect to $x$, at point $x_1$.

Taking $x_1 = 0$, $e^x$ gives the following result after expansion:

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + ...$$

After some algebra, $E_{rel,1}$ is:

$$E_{rel,1} = -\frac{1}{2} c \cdot \Delta t - \frac{1}{4} c^2 \cdot \Delta t^2 - \frac{1}{12} c^3 \cdot \Delta t^3 - ...$$

Second and higher order terms may be omitted because $c\Delta t \leq 1/10$, which yields Equation 37 of Section 6.4.
Appendix 2

Derivation of the relative error in the rectangular integration method for an exponential rate curve which is not known as a function of time. (i.e. integration with feedback).

The surface areas obtained by the rectangular and trapezoidal integration methods are substituted in Equation 35:

\[
E_{rel,1} = 1 - \frac{A_T \cdot (1 + \Delta t \cdot c + (\Delta t \cdot c)^2/2)}{A_T \cdot (1 + \Delta t \cdot c)}
\]

Cancelling \( A_T \) and setting 1 over the common denominator gives:

\[
E_{rel,1} = \frac{-(\Delta t \cdot c)^2/2}{1 + \Delta t \cdot c} \equiv -\frac{1}{2} \cdot (\Delta t \cdot c)^2,
\]

because \( \Delta t \cdot c \leq 1/10 \), which is similar to Equation 38 of Section 6.4.
Appendix 3. Summary of the processing of a CSMP program.

Symbols library

<table>
<thead>
<tr>
<th>Input processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation stage</td>
</tr>
<tr>
<td>CSMP → FORTRAN</td>
</tr>
<tr>
<td>creates → FOR03.DAT, listing of the model, indication of memory space used, and possible error messages in the program written by the user. If programming errors occur, no results will be printed in the FOR06.DAT.</td>
</tr>
<tr>
<td>Results in a subroutine</td>
</tr>
<tr>
<td>UPDATE</td>
</tr>
<tr>
<td>is → a subprogram which can be run under CSMP. Contains the sorted equations (if the sorting algorithm of CSMP has been used). The file UPDATE.FOR can be inspected.</td>
</tr>
<tr>
<td>Translation stage</td>
</tr>
<tr>
<td>FORTRAN → Machine code</td>
</tr>
<tr>
<td>creates → FOR06.DAT, lists all parameters used, and contains the calculated results of the model. If calculational errors occur, these may be reported. Only results up to the occurrence of the error will be printed in the FOR06.DAT.</td>
</tr>
</tbody>
</table>

Controlling of the progress of the program. Lists all parameters used in the file (CONTRO.SYS).
Appendix 4

Derivation of the average concentration (AVC) at the boundary of two consecutive layers of different thickness (TCOM) in connection with the suppression of numerical dispersion.

Consider the following drawing:

```
C(I-1)  C(I)
  
AVC(I)  C(I)
  
C(I-1)
  
0.5*TCOM(I-1)  0.5*TCOM(I)
```

Where C(I) is the concentration in layer I.

Congruence in triangles results in the following relationship:

\[
\frac{AVC(I) - C(I-1)}{0.5 \cdot TCOM(I-1)} = \frac{C(I) - C(I-1)}{0.5 \cdot (TCOM(I-1) + TCOM(I))}
\]

Writing AVC(I) explicitly results in:

\[
AVC(I) = \frac{C(I) \cdot TCOM(I-1) + C(I-1) \cdot TCOM(I)}{TCOM(I-1) + TCOM(I)}
\]
Appendix 5

Derivation of the average conductivity of two adjacent layers of unequal thickness with different conductivity or resistance.

The substitution resistance \( R_s \), where index \( s \) stands for substitution, of two resistances in series is simply the sum of the two resistances:

\[
R_s = R_1 + R_2
\]

The resistance of a medium is directly proportional to its length \( L \), and inversely proportional to its surface area \( A \) and to its specific conductivity \( C \). This implies the following for the three resistances:

\[
R_1 = \frac{L_1}{A_1 \cdot C_1}, \quad R_2 = \frac{L_2}{A_2 \cdot C_2}, \quad R_s = \frac{L_s}{A_s \cdot C_s}.
\]

Now, assume that the surface area through which the matter will flow, has a common value, so \( A_1 = A_2 = A_s \). Consequently, the surface area may be omitted. Furthermore, \( L_s = L_1 + L_2 \). Therefore, substitution of the three expressions for the resistances in the first equation, and writing \( C_s \) explicitly, results in:

\[
C_s = \frac{L_s}{L_1/C_1 + L_2/C_2} = \frac{L_1 + L_2}{L_1/C_1 + L_2/C_2}.
\]

Consider the following drawing for the application of this equation:

```
<table>
<thead>
<tr>
<th>TCOM(I-1) ; COND(I-1)</th>
<th>AVCOND(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCOM(I) ; COND(I)</td>
<td>AVCOND(I+1)</td>
</tr>
<tr>
<td>TCOM(I+1) ; COND(I+1)</td>
<td></td>
</tr>
</tbody>
</table>
```

The index convention for the average conductivity, is similar to that for fluxes: the index is taken similar to the number of the layer into which it enters.

The following holds for a layered medium:

\[
L_1 = TCOM(1) = TCOM(I-1)
\]
\[
C_1 = COND(1) = COND(I-1)
\]
\[
L_2 = TCOM(2) = TCOM(I)
\]
\[
C_2 = COND(2) = COND(I)
\]
\[
C_s = AVCOND(2) = AVCOND(I)
\]

Which results in the following equation:

\[
AVCOND(I) = \frac{TCOM(I-1) + TCOM(I)}{TCOM(I-1)/COND(I-1) + TCOM(I)/COND(I)}
\]
The main program FORSIM, the Euler and Runge-Kutta drivers, and the adapted integration routines from Press et al. (1986), including subroutine headers which explain their meaning.

Notes:
To run the programs in this appendix, the utility library TTUTIL should be linked. The library may be obtained from the department of Theoretical Production Ecology of the Wageningen Agricultural University. With the library goes a technical description on how to create and use an object library for the VAX mainframe computer, the Apple Macintosh and the IBM-PC.

In the main program FORSIM the possibility is introduced to choose between the integration methods of Euler and Runge-Kutta.

If more than 250 state equations are needed PARAMETER (NDEC=250) in EUDRIV (Figure 46, line 15) and RKDRIV (Figure 51, line 15), and PARAMETER (NL=250) in RKQCA and RK4A should be adapted.

PROGRAM FORSIM
* Main programming for calling the rerun facilities in the TTUTIL library and for calling a driver routine for the application of Euler or Runge-Kutta integration to a user specified model (here called MODEL).
*
variables:
* IULOG - Unit number of logfile. Used by the rerun facility for messages and available in the user program via COMMON /INFO/ declared in the driving routine(s).
* (When IULOG=0 no logfile is used by utility).
* IURER - Unit number used for the temporary file RERUNS.TMP
* (IURER+1 is used for the data file RERUNS.DAT itself)
* IUDRIV - Unit number used for the reading the timer routine in routine RKDRIV and EUDRIV, also IUTIM+1 is used.
* IUMOD - First of a series of free unit numbers to be used by the user supplied model. It is suggested to keep a decade of unit numbers free.
* INSETS - Number of parameter sets in RERUNS.DAT.
* IS - Set number in rerun loop.
* FATAL - Flag, determines reaction on non-used variables in rerun sets; currently set .true.
* MODEL - Name of external user model subroutine, (see this example and the header of RKDRIV or EUDRIV for a description of the form of this routine).
* Subroutines and/or functions called:
* - from library TTUTIL: DECCHK, DECENT, DECREA, ENTDCH, ERROR,
* EUDRIV, EXTENS, FOPENG, IFINDC, ILEN,
* ISTART, RDDATA, RDFROM, RDINDX, RDINIT,
* RDSETS, RDSREA, RK4A, RKDRIV, RKQCA,
* UPERC
*
Author: Kees Rappoldt
* Date: October 1990
* declarations
INTEGER IS,IULOG,IURER,IUDRIV,IUMOD

252
Appendix 5

Derivation of the average conductivity of two adjacent layers of unequal thickness with different conductivity or resistance.

The substitution resistance \( R_s \), where index \( s \) stands for substitution, of two resistances in series is simply the sum of the two resistances:

\[
R_s = R_1 + R_2
\]

The resistance of a medium is directly proportional to its length \( (L) \), and inversely proportional to its surface area \( (A) \) and to its specific conductivity \( (C) \). This implies the following for the three resistances:

\[
R_1 = \frac{L_1}{A_1 \cdot C_1}; \quad R_2 = \frac{L_2}{A_2 \cdot C_2}; \quad R_s = \frac{L_s}{A_s \cdot C_s}
\]

Now, assume that the surface area through which the matter will flow, has a common value, so \( A_1 = A_2 = A_s \). Consequently, the surface area may be omitted. Furthermore, \( L_s = L_1 + L_2 \). Therefore, substitution of the three expressions for the resistances in the first equation, and writing \( C_s \) explicitly, results in:

\[
C_s = \frac{L_s}{L_1/C_1 + L_2/C_2} = \frac{L_1 + L_2}{L_1/C_1 + L_2/C_2}
\]

Consider the following drawing for the application of this equation:

The index convention for the average conductivity, is similar to that for fluxes: the index is taken similar to the number of the layer into which it enters.

The following holds for a layered medium:

\[
L_1 = \text{TCOM}(1) = \text{TCOM}(I-1)
\]

\[
C_1 = \text{COND}(1) = \text{COND}(I-1)
\]

\[
L_2 = \text{TCOM}(2) = \text{TCOM}(I)
\]

\[
C_2 = \text{COND}(2) = \text{COND}(I)
\]

\[
C_s = \text{AVCOND}(2) = \text{AVCOND}(I)
\]

Which results in the following equation:

\[
\text{AVCOND}(I) = \frac{\text{TCOM}(I-1) + \text{TCOM}(I)}{\text{TCOM}(I-1)/\text{COND}(I-1) + \text{TCOM}(I)/\text{COND}(I)}
\]
The main program FORSIM, the Euler and Runge-Kutta drivers, and the adapted integration routines from Press et al. (1986), including subroutine headers which explain their meaning.

Notes:

To run the programs in this appendix, the utility library TTUTIL should be linked. The library may be obtained from the department of Theoretical Production Ecology of the Wageningen Agricultural University. With the library goes a technical description on how to create and use an object library for the VAX mainframe computer, the Apple Macintosh and the IBM-PC.

In the main program FORSIM the possibility is introduced to choose between the integration methods of Euler and Runge-Kutta.

If more than 250 state equations are needed PARAMETER (NDEC=250) in EUDRIV (Figure 46, line 15) and RKDRIV (Figure 51, line 15), and PARAMETER (NL=250) in RKQCA and RK4A should be adapted.

```
PROGRAM FORSIM

* Main programming for calling the rerun facilities in the
* TTUTIL library and for calling a driver routine for the
* application of Euler or Runge-Kutta integration to a
* user specified model (here called MODEL).

* variables:
* IULOG - Unit number of logfile. Used by the rerun facility
  for messages and available in the user program via
* COMMON /INFO/ declared in the driving routine(s).
* IULOG=0 no logfile is used by utility).
* IURER - Unit number used for the temporary file RERUNS.TMP
* (IURER+1 is used for the data file RERUNS.DAT itself)
* IUDRIV - Unit number used for the reading the timer routine
* in routine RKDRIV and EUDRIV, also IUTIM+1 is used.
* IUMOD - First of a series of free unit numbers to be used
* by the user supplied model. It is suggested to keep
* a decade of unit numbers free.
* INSETS - Number of parameter sets in RERUNS.DAT.
* IS - Set number in rerun loop.
* FATAL - Flag, determines reaction on non-used variables in
* rerun sets; currently set .true.
* MODEL - Name of external user model subroutine, (see this
* example and the header of RKDRIV or EUDRIV for a
* description of the form of this routine).

* Subroutines and/or functions called:
* - from library TTUTIL: DECHK, DECINT, DECREA, ENTDCH, ERROR,
* EUDRIV, EXTENS, FOPENG, IFINDC, ILEN,
* ISTART, RDDATA, RDFROM, RDINDEX, RDINIT,
* RDSETS, RDSREA, RK4A, RKDRIV, RKQCA,
* UPERC

* Author: Kees Rappoldt
* Date: October 1990

* declarations
INTEGER IS,IULOG,IURER,IUDRIV,IUMOD
```
REAL DUM
CHARACTER METHOD*2
LOGICAL FATAL
EXTERNAL MODEL

unit numbers used for logfile and for rerun facility
DATA IULOG/20/, IURER/30/
unit numbers used for driver and model
DATA IUDRIV/40/, IUMOD/50/
fatal errors on non-used rerun variable values
DATA FATAL/.TRUE./

choose integration method
CONTINUE
CALL ENTDCH ('Euler (E) or Runge-Kutta (RK) method', 'RK', METHOD)
CALL UPPERC (METHOD)
IF (.NOT. (METHOD.EQ. 'E' .OR. METHOD.EQ. 'RK')) GOTO 10

dummy model call (some machines do not correctly externalyze MODEL)
CALL MODEL (0,.FALSE.,DUM,DUM,DUM,DUM,1,IS)

open logfile and analyse rerun file
CALL FOPENG (IULOG,'MODEL.LOG','NEW','SF',O,'DEL')
CALL RDSETS (IURER,IULOG, 'RERUNS.DAT',INSETS)

model runs
DO 20 IS = 0,INSETS
   WRITE (IULOG, '(/,A,I4,/,A)') ' Run',IS+1,'       ====='
   WRITE ( * (I I I A I I I A) ) '
   CALL RDFROM (IS,FATAL)
   IF (METHOD.EQ. 'E') CALL EUDRIV (IULOG,IUDRIV,IUMOD,MODEL)
   IF (METHOD.EQ. 'RK') CALL RKDRIV (IULOG,IUDRIV,IUMOD,MODEL)
20 CONTINUE
IF (INSETS.GT.0) CLOSE (IURER,STATUS='DELETE')
STOP
END

************************************************************************
*
*
*
*
*
************************************************************************

SUBROUTINE EUDRIV (IUL,IUD,IUM,MODEL)

Solves an initial value problem with the simple Euler method.
This driver routine initializes the model, reads a control
file TIMER.DAT and drives the user supplied model until the
finish time FINTIM given in TIMER.DAT is reached.

IUL - logfile unit number
IUD - first of two free unit numbers used by this driver
IUM - first of a series of free unit numbers for model
MODEL - external model routine

Subroutines and/or functions called:
- from library TTUTIL: DECCHK, DECINT, DECREA, ERROR, EXTENS,
  FOPENG, IFINDC, ILEN, ISTART, RDDATA,
  RDINDEX, RDINIT, RDSREA, UPPERC

Author: Kees Rappoldt
Date : October 1990

The user supplied routine MODEL:
The differential equations are actually contained in the user supplied subroutine MODEL which is called by this driver as:

```
CALL MODEL (ITASK, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)
```

Note that the user routine may have an arbitrary name which is given as an EXTERNAL in the CALL to this driver EUDRIV. The action of the user supplied model subroutine depends on the value of ITASK in the following way.

**ITASK = 1** The model is initialized. The number of state variables (differential equations) NEQ is set. Model parameters are set or are read from file. Time and states are set to their initial values. Also the corresponding scales have to be set. The scale array SCALE contains the order of magnitude of each state variable in STATE. A scale needs to be a positive number, for instance 0.001, 0.5 or 30000.0. Also output should be initialized (files opened, headers etc.)

The SCALE array is not used in the EULER integration method.

**ITASK = 2** Values in the STATE array and the current TIME are used to calculate rates of change for each status variable. In order to prevent confusion it is advised to use local and more meaningful names for state variables than just the input array elements STATE(1), STATE(2), etc. Then, at first, the state array is copied into the local variables, then the rates are calculated which are finally copied into the output array RATE.

**ITASK = 4** Terminal call to the model. Final output may have to be generated, files closed, etc.

Some CALL's with ITASK=2 ("rate calls") take place with the logical OUTPUT set to .TRUE. Then the user supplied model produces output to file and/or screen. The period between successive output times is PRDEL, a variable read from the control file TIMER.DAT.

At the start of a new time step, the state array STATE contains a valid (new) status of the system. If anything has to be changed in the state array in order to account for discontinuities, for instance, that should be done at such moments. Therefore, the rate CALL's to MODEL at the beginning a new time step are carried out with common variable KEEP equal to 1. Otherwise KEEP is 0.

This common variable, KEEP, is part of a small common block /INFO/. INFO also contains the unit number of an opened logfile and the first of a series of free unit numbers that can be used by the model. Further /INFO/ contains a logical TERMNL. This logical can be set to .TRUE. by the model routine at any moment. It causes the termination of the current simulation run. Note that, after setting TERMNL to .TRUE. a number of CALL's to MODEL will follow in order to terminate the current time step and to produce final output. Hence, when the flag TERMNL is set, it should never be reset by the model. Then its status could be missed by the driver. Also the timer variables DELT, PRDEL, DELMAX, FINTIM are included in the common INFO to be able to manipulate time when discontinuities occur, and to have the possibility to print these variables.

formal parameters
INTEGER IUL,IUD,IUM
EXTERNAL MODEL

common /INFO/ DELT , PRDEL, DELMAX, FINTIM
INTEGER IULOG , IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT , PRDEL, DELMAX, FINTIM,
$ IULOG , IUMOD, KEEP , TERMNL
**local (non-common) variables**

```plaintext
INTEGER I , IP , NEQ , NDEC
REAL STTIME, TIME , TNEXT , DEL , DELT
REAL DELHLP, MULTI1, MULTI2
REAL STATE , RATE , SCALE
PARAMETER (NDEC=250)
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
LOGICAL HALT, OUTPUT
```

* get copy of logfile unit number and model unit number into /INFO/

IULOG = IUL
IUMOD = IUM

IF (IULOG.GT.0)
$ WRITE (IULOG, '(A)') ' Initialize model'
WRITE ( *(A) ) ' Initialize model'

* read timer variables

CALL RDINIT (IUD,IULOG,'TIMER.DAT')
CALL RDSREA ('STTIME',STTIME)
CALL RDSREA ('FINITIM',FINITIM)
CALL RDSREA ('PRDEL',PRDEL)
CALL RDSREA ('DELMAX',DELMAX)
CALL RDSREA ('DELT',DELT)
CLOSE (IUD,STATUS='DELETE')
IF (PRDEL .LE. 0.0 ) CALL ERROR ('EUDRIV', 'Illegal value PRDEL')
IF (DELT .GT. PRDEL) CALL ERROR ('EUDRIV', 'Illegal value DELT')
IF (DELMAX .GT. PRDEL)
$ CALL ERROR ('EUDRIV', 'Illegal value DELMAX')

* investigate if DELT is a multiple of PRDEL

MULTI1 = PRDEL/DELT
MULTI2 = FLOAT( INT(PRDEL/DELT) )
IF (MULTI1 .NE. MULTI2 ) THEN

* DELT is no multiple of PRDEL; a new DELT should be calculated
DELT = DELHLP/2.
ELSE
DELT = DELHLP/2.
GOTO 5
ENDIF
ENDIF

DELT = MIN (DELT, DELMAX)
DELT0 = DELT

* initialize timing

TERMNL = .FALSE.
TIME = STTIME
TNEXT = STTIME
HALT = TIME.GE.FINITIM
IP = 1 + INT ((FINITIM - STTIME) / PRDEL - 0.01)

* to simplify debugging: set states and rates to 0

DO 10 I=1,NDEC
  STATE(I) = 0.0
  RATE(I) = 0.0
10 CONTINUE

* initialize model

CALL MODEL (1,.FALSE.,TIME,STATE,RATE,SCALE,NDEC,NEQ)

* error checks
IF (NEQ.LE.0) CALL ERROR ('EUDRIV',
$ 'No value of NEQ was specified in MODEL!')
IF (NEQ.GT.NDEC) CALL ERROR ('EUDRIV',
$ 'Too many state variables')

* dynamic loop
IF (IULOG.GT.0)
$ WRITE (IULOG, '(A)') ' Dynamic loop'
WRITE ( *, '(A)') ' Dynamic loop'

20 IF (.NOT.HALT) THEN
* output required ?
OUTPUT = (TNEXT - TIME)/PRDEL.LT.0.0001 .OR. TERMNL

* get rates of change at beginning of time step (write output)
* this is the rate call at the start of a new step (KEEP=1)
KEEP = 1
CALL MODEL (2,OUTPUT,TIME,STATE,RATE,SCALE,NDEC,NEQ)
KEEP = 0
IF (OUTPUT) THEN
* get next output time ; leave dynamic loop ?
IF = IP + 1
TNEXT = FINTIM - IP * PRDEL
HALT = TNEXT.GT.FINTIM .OR. TERMNL
END IF

* time step limitation and integration
IF (.NOT.HALT .AND. .NOT.TERMNL) THEN
* limit timestep
DELT = MIN (DELTO,DELMAX)
IF (TIME+DELT.LT.TNEXT) THEN
* accept advised step
DEL = DELT
ELSE
* reduce time step
DEL = TNEXT-TIME
END IF

* integration
TIME = TIME + DEL
DO 30 I=1,NEQ
STATE(I) = STATE(I) + DEL * RATE(I)
30 CONTINUE
DELT = DELTO
GOTO 20
END IF

* terminate model
IF (IULOG.GT.0)
$ WRITE (IULOG, '(A)') ' Terminate model'
WRITE ( *, '(A)') ' Terminate model'
CALL MODEL (4,.FALSE.,TIME,STATE,RATE,SCALE,NDEC,NEQ)
RETURN
END

**************************************************************************************************
SUBROUTINE RKDRIV (IUL,IUD,IUM,MODEL)
* Solves an initial value problem with the fourth order Runge Kutta
method described by Press et al. (1986). This driver routine initializes the model, reads a control file TIMER.DAT and drives the user supplied model until the finish time in TIMER.DAT is reached.

IUL - logfile unit number
IUD - first of two free unit numbers used by this driver
IUM - first of a series of free unit numbers for model
MODEL - external model routine

Subroutines and/or functions called:
from library TTUTIL: DECCHK, DECINT, DECREA, ENTDCH, ERROR,
EXTENS, FOPFNG, IFINDC, ILEN, ISTART,
RD DATA, RDINDX, RDINIT, RDSREA, RK4A,
RKQCA, UPPERC

Author: Kees Rappoldt
Date: October 1990

The user supplied routine MODEL:
The differential equations are actually contained in the user supplied subroutine MODEL which is called by this driver as:
CALL MODEL (ITASK, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)

Note that the user routine may have an arbitrary name which is given as an EXTERNAL in the CALL to this driver RKDRIV. The action of the user supplied model subroutine depends on the value of ITASK in the following way.

ITASK = 1 The model is initialized. The number of state variables (differential equations) NEQ is set. Model parameters are set or are read from file. Time and states are set to their initial values. Also the corresponding scales have to be set. The scale array SCALE contains the order of magnitude of each state variable in STATE. A scale needs to be a positive number, for instance 0.001, 0.5 or 30000.0. Also output should be initialized (files opened, headers etc.)

ITASK = 2 Values in the STATE array and the current TIME are used to calculate rates of change for each status variable. In order to prevent confusion it is advised to use local and more meaningful names for state variables than just the input array elements STATE(1), STATE(2), etc. Then, at first, the state array is copied into the local variables, then the rates are calculated which are finally copied into the output array RATE.

ITASK = 4 Terminal call to the model. Final output may have to be generated, files closed, etc.

Some CALL's with ITASK=2 ("rate calls") take place with the logical OUTPUT set to .TRUE. Then the user supplied model produces output to file and/or screen. The period between successive output times is PRDEL, a variable read from the control file TIMER.DAT.

At the start of a new time step (taken by the Runge Kutta routine RKQCA), the state array STATE contains a valid (new) status of the system. If anything has to be changed in the state array in order to account for discontinuities, for instance, that should be done at such moments. Therefore, the CALL's to MODEL at the beginning a new time step are carried out with the common variable KEEP equal to 1. Otherwise KEEP is 0.

This common variable, KEEP, is part of a small common block /INFO/. INFO also contains the unit number of an opened logfile and the first of a series of free unit numbers that can be used by the model. Further /INFO/ contains a logical TERMINL This logical can be set to .TRUE. by the model routine at any moment. It causes the
termination of the current simulation run. Note that, after setting
TERMNL to .TRUE., a number of CALL's to MODEL will follow in order
to terminate the current time step and to produce final output.
Hence, when the flag TERMNL is set, it should never be reset by
the model. Then its status could be missed by the driver.
Also the timer variables DELT, PRDEL, DELMAX, FINTIM are included in
the common INFO to be able to manipulate time when discontinuities
occur, and to have the possibility to print these variables.

formal parameters
INTEGER IUL, IUD, IUM
EXTERNAL MODEL

common /INFO/
REAL DELT, PRDEL, DELMAX, FINTIM
INTEGER IULOG, IUMOD, KEEP
LOGICAL TERMNL
COMMON /INFO/ DELT, PRDEL, DELMAX, FINTIM,
$ IULOG, IUMOD, KEEP, TERMNL

**
local (non-common) variables
INTEGER I, IP, NEQ, NDEC
REAL STTIME, EPS, TIME, TNEXT, DELIDL, DELNXT, DUMMY
REAL STATE, RATE, SCALE
PARAMETER (NDEC=250)
DIMENSION STATE (NDEC), RATE (NDEC), SCALE (NDEC)
LOGICAL HALT, OUTPUT
SAVE

get copy of logfile unit number and model unit number into /INFO/
IULOG = IUL
IUMOD = IUM

IF (IULOG.GT.0)
$ WRITE (IULOG, '('A')') 'Initialize model'
WRITE ( *, '('A') ') 'Initialize model'

read timer variables
CALL RDINIT (IUD, IULOG, 'TIMER.DAT')
CALL RDSREA ('STTIME', STTIME)
CALL RDSREA ('FINTIM', FINTIM)
CALL RDSREA ('PRDEL', PRDEL)
CALL RDSREA ('EPS', EPS)
CALL RDSREA ('DELMAX', DELMAX)
CALL RDSREA ('DELT', DELT)
CLOSE (IUD, STATUS='DELETE')
IF (PRDEL .LE. 0.0)
$ CALL ERROR ('RKDRIV', 'Illegal value PRDEL')
IF (DELMAX .GT. PRDEL)
$ CALL ERROR ('RKDRIV', 'Illegal value DELMAX')

initialize timing
TERMNL = .FALSE.
TIME = STTIME
TNEXT = STTIME
HALT = TIME .GE. FINTIM
IP = 1 + INT ((FINTIM - STTIME) / PRDEL - 0.01)
DELNXT = DELT

to simplify debugging: set states and rates to 0 and scales to 1
DO 10 I=1,NDEC
STATE(I) = 0.0
RATE(I) = 0.0

258
* initialize model
  CALL MODEL (1., .FALSE., TIME, STATE, RATE, SCALE, NDEC, NEQ).

* error checks:
  IF (NEQ.LE.0) CALL ERROR ('RKDRIV',
  $ 'No value of NEQ was specified in MODEL')
  IF (NEQ.GT.NDEC) CALL ERROR ('RKDRIV',
  $ 'Too many state variables')
  DO 20 I=1, NEQ
    IF (SCALE(I).LE.0.0) CALL ERROR ('RKDRIV',
    $ 'At least one SCALE variable is not positive')
  20 CONTINUE

* dynamic loop
  IF (IULOG.GT.0)
    WRITE (IULOG,' (A)') ' Dynamic loop!
    WRITE ( *, '(A)') ' Dynamic loop'

30 IF (.NOT.HALT) THEN
  output required ?
  OUTPUT = (TNEXT-TIME)/PRDEL.LT.0.0001 .OR. TERMNL

  get rates of change at beginning of time step (write output)
  this is the rate call at the start of a new step (KEEP=1)
  KEEP = 1
  CALL MODEL (2, OUTPUT, TIME, STATE, RATE, SCALE, NDEC, NEQ)
  KEEP = 0
  IF (OUTPUT) THEN
    get next output time ; leave dynamic loop ?
    IP = IP - 1
    TNEXT = FINTIM - IP * PRDEL
    HALT = TNEXT.GT.FINTIM .OR. TERMNL
  END IF

* one step with accuracy control using RKQCA
  IF (.NOT.HALT .AND. .NOT.TERMNL) THEN
    limit timestep
    DELNXT = MIN (DELNXT, DELMAX)
  IF (TIME+DELNXT .LT. TNEXT) THEN
    accept advised step
    DELT = DELNXT
    CALL RKQCA (STATE, RATE, NDEC, NEQ, TIME, DELT, EPS,
    $ SCALE, DELDID, DELNX'T, MODEL)
  ELSE
    reduce time step, but do not overwrite previous advise
    DELT = TNEXT-TIME
    CALL RKQCA (STATE, RATE, NDEC, NEQ, TIME, DELT, EPS,
    $ SCALE, DELDID, DUMMY, MODEL)
  END IF
  END IF
GOTO 30
END IF

* terminate model
  IF (IULOG.GT.0)
    WRITE (IULOG,' (A)') ' Terminate model'
    WRITE ( *, '(A)') ' Terminate model'
    CALL MODEL (4, .FALSE., TIME, STATE, RATE, SCALE, NDEC, NEQ)
RETURN
SUBROUTINE RKQC (STATE, RATE, NDEC, NEQ, TIME, 
DELTRY, EPS, SCALE, DELDID, DELNXT, MODEL)

* Runge Kutta integration with stepsize control.
* Adapted from routine RKQC from Press et al. (1986)

STATE - state array of model I/O
RATE - rates of change for (TIME,STATE) I/O
NEQ - Number of state / rate variables I
TIME - time I/O
DELTRY - time step tried I
EPS - relative accuracy of criterion I
SCALE - size scale of state variables I
DELDID - time step taken O
DELNXT - advise for new step O
MODEL - external model called with ITASK=2 I
* for rate calculation

Subroutines and/or functions called:
- from library TTUTIL: ENTDCH, ERROR, ILEN, ISTART, RK4A

Author: Kees Rappoldt, adapted from Press et al. (1986)
Date: February 1990

** formal parameters
INTEGER NDEC, NEQ
REAL STATE, RATE, TIME, DELTRY, EPS, SCALE, DELDID, DELNXT
DIMENSION STATE(NDEC), RATE(NDEC), SCALE(NDEC)
EXTERNAL MODEL

** local variables
INTEGER I, NL
PARAMETER (NL=250)
REAL TSTATE, SSTAT, SRATE, FCOR, ONE, SAFETY, ERRCON
REAL PGROW, PSHRNK, TSAV, DELT, DELT2, ERMAX
DIMENSION TSTATE(NL), SSTAT(NL), SRATE(NL)
CHARACTER*1 ANS
LOGICAL GOOD
PARAMETER (FCOR=0.666666667, ONE=1.0, SAFETY=0.9, ERRCON=6.E-4)
SAVE

PGROW = -0.20
PSHRNK = -0.25

* check size of local arrays
IF (NEQ.GT.NL) CALL ERROR ('RKQC', 'Local arrays too short')

* save current time, state and rate
TSAV = TIME
DO 10 I=1,NEQ
   SSTAT(I) = STATE(I)
   SRATE(I) = RATE(I)
10 CONTINUE

* accept suggested value
DELT = DELTRY
GOOD = .FALSE.
* do while
20   IE (.NOT.,GOOD). THEN
   at first two integration steps are made, each with half DELT
   DELT2 = 0.5*DELT
   CALL RK4A (SSTAT, SRATE, NDEC, NEQ, TSAV, DELT2, TSTATE, SCALE, MODEL)

* the second half, with result stored in STATE
   TIME = TSAV + DELT2
   CALL RK4A (TSTATE, RATE, NDEC, NEQ, TIME, DELT2, STATE, SCALE, MODEL)

* now the full step is taken: check significance
   TIME = TSAV + DELT
   IF (TIME.EQ.TSAV) THEN
     WRITE (*, ' (A) ') 'Stepsize not significant in RKQCA'
     CALL ENTDCH ('Do you want to STOP? ', 'W')
     IF (ANS.NE.'N') CALL ERROR ('RKQCA', 'Execution terminated')
   END IF

* integration over full step
   CALL RK4A (SSTAT, SRATE, NDEC, NEQ, TSAV, DELT, TSTATE, SCALE, MODEL)

* from the difference the error criterion is found
   ERRMAX = 0.0
   DO 30 I=1, NEQ
      TSTATE(I) = STATE(I) - TSTATE(I)
      ERRMAX = MAX (ERRMAX, ABS(TSTATE(I)/SCALE(I)))
   CONTINUE

* get new time step when not OK
   ERRMAX = ERRMAX/EPS
   GOOD = ERRMAX.LT.ONE
   IF (.NOT.,GOOD) DELT = SAFETY*DELT*(ERRMAX**PSHRNK)
   GOTO 20
   END IF

* accept step and calculate advise for new DELT
   DELID = DELT
   IF (ERRMAX.GT.ERRCON) THEN
     DELNXT = SAFETY*DELT*(ERRMAX**PGROW)
   ELSE
     DELNXT = 4.0*DELT
   ENDIF

* correct the half step result with the difference in TSTATE
   DO 40 I=1, NEQ
      STATE(I) = STATE(I) + TSTATE(I)*FCOR
   CONTINUE

RETURN
END

************************************************************************************

SUBROUTINE RK4A (STATE, RATE, NDEC, NEQ, TIME, DELT, STATE2, SCALE, MODEL)
* Fourth order Runge Kutta integration over DELT
* Adapted from routine RK4 from Press et al. (1986)

* STATE - state array of model I
* RATE - rates of change for (TIME,STATE) I
* NDEC - declared size of input arrays I
* NEQ - Number of state / rate variables I
* TIME - time I

261
** Subroutines and/or functions called: 
* - from library TTUTIL: ERROR 

** Author: Kees Rappoldt, adapted from Press et al. (1986) 
** Date: February 1990 

* formal parameters 
INTEGER NDEC, NEQ 
REAL STATE, RATE, TIME, DELT, STATE2, SCALE 
DIMENSION STATE(NDEC), RATE(NDEC), STATE2(NDEC), SCALE(NDEC) 
EXTERNAL MODEL 

** local variables 
INTEGER I, NL 
REAL STATET, RATET, RATEM, DELT2, DELT6, TIME2 
PARAMETER (NL=250) 
DIMENSION STATET(NL), RATET(NL), RATEM(NL) 
SAVE 

* check size of local arrays 
IF (NEQ.GT.NL) CALL ERROR ('RK4A','Local arrays too short') 

DELT2 = DELT*0.5 
DELT6 = DELT/6.0 
TIME2 = TIME + DELT2 

* get midpoint state using initial rate 
DO 10 I=1,NEQ  
  STATET(I) = STATE(I) + DELT2*RATE(I)  
CONTINUE 

* find midpoint rates and a new midpoint state 
CALL MODEL (2,.FALSE.,TIME2,STATET,RATET,SCALE,NDEC,NEQ) 
DO 20 I=1,NEQ  
  STATET(I) = STATE(I) + DELT2*RATET(I)  
CONTINUE 

* get a second estimate of midpoint rates, 
* estimate final state, get sum of the two midpoint rates 
CALL MODEL (2,.FALSE.,TIME2,STATET,RATEM,SCALE,NDEC,NEQ) 
DO 30 I=1,NEQ  
  STATET(I) = STATE(I) + DELT*RATEM(I)  
  RATEM(I) = RATET(I) + RATEM(I)  
CONTINUE 

* get final rates, calculate new state from all calculated rates 
CALL MODEL (2,.FALSE.,TIME+DELT,STATET,RATET,SCALE,NDEC,NEQ) 
DO 40 I=1,NEQ  
  STATE2(I) = STATE(I) + DELT6 * (RATE(I)+RATET(I)+2.0*RATEM(I))  
CONTINUE 

RETURN 
END 

************************************************************************