2.2 Introduction of CSMP by an elementary simulation program

L.J.M. Basstanie and H.H. van Laar

2.2.1 Introduction

The scientific approach toward studying ecological and physiological phenomena often results in their being described by mathematical expressions. A great deal of these phenomena behave in a 'dynamic' fashion which means the 'state' of the ecological system changes with time. With knowlegde of the processes within the system one can develop a mathematical model to study the dynamic behaviour of the system.

De Wit & Goudriaan (1978), Ferrari (1978) and Brockington (1979) have written introductory textbooks on simulation of ecological processes. Although the main topic of this book is modelling of growth processes and related phenomena, those textbooks are useful for their complementary explanations and illustrations.

In this section we introduce the simulation language CSMP (Continuous Simulation Modeling Program) (IBM, 1975), which is used throughout this book. Basic principles of its use are demonstrated by construction of a simple program for the simulation of crop dry-matter production (Subsection 2.2.2). In its first form the program calculates photosynthesis, respiration, dry-matter distribution, and leaf area index during a growth season, assuming a constant environment. Subsequently the program is modified to account for a varying environment and its effect on some of the processes (Subsection 2.2.3). Starting from this basic level has the advantage that the reader who is not familiar with growth modelling and programming in CSMP can find handholds in such a simple program. Gradually, as processes are treated in more detail in the following sections, more elaborate modelling techniques and their programming in CSMP will be discussed.

Subject of this section is also the internal structure of a CSMP program with special attention to the sorting mechanism (Subsection 2.2.4). Finally some basic CSMP programming knowledge is summarized (Subsection 2.2.5) to provide sufficient information for a good comprehension of the following sections. In Section 2.3 some of these aspects will be elaborated. The CSMP manual (IBM, 1975) may be useful for those with some programming experience.

2.2.2 An elementary simulation program for dry-matter production

In a first approximation dry-matter production of a crop can be simulated based on the simple model depicted in Figure 16. Gross CO_2 assimilation feeds a pool of carbohydrates that supplies material for growth and respiration. Part of

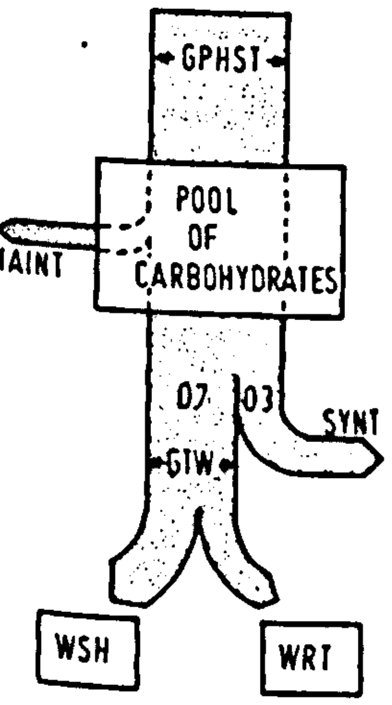


Figure 16. Flow of carbohydrates in a simple crop-production model.

his pool is needed for maintenance of the crop, a process usually defined as naintenance respiration. The substrate for growth is also taken from this pool, nd the conversion into shoot and root biomass involves a loss of carbohydrates, lefined as synthesis respiration. The relational diagram of such a system has een presented in Figure 2, Section 1.2.

Because we would like to describe this model and use it for simulation of rowth on a day-to-day basis, one simplification of the relational diagram has o be made: we will assume that all carbohydrates from photosynthesis are conumed by growth and maintenance processes within one day. As a result, we can mit the pool of carbohydrates as a separate variable from the model. In Secion 3.3 this simplification will be explained more extensively.

The relational diagram reveals the kind of quantitative information that hould be available. A standard value for the gross photosynthetic rate GPHST) of a green and completely closed canopy, well supplied with water nd nutrients is in terms of glucose production on the order of 400 kg $\text{ha}^{-1}\text{d}^{-1}$ n clear summer days. When the canopy is not fully closed the actual value of ne gross photosynthetic rate (GPHOT) is a fraction of GPHST. This fraction orresponds with the fraction of absorbed visible radiation, calculated with an xponential extinction of radiation as function of the leaf area index (LAI, an rea fraction of leaves to ground surface, in m^2m^{-2}). A value of 0.7 for the xtinction coefficient results in a fraction (1. − EXP(−0.7 * LAI)) of absorbed isible light.

Maintenance respiration (MAINT), expressed in glucose, is related to the tal dry matter weight (TWT), a proportionality factor of 0.015 kg $\text{kg}^{-1}\ \text{d}^{-1}$ is fair estimation. As for growth, a conversion factor (CVF, in kilogram of dry atter per kilogram of glucose) of 0.7 quantifies reasonably well the efficiency f synthesis of structural material from the carbohydrates, the remainder being st as respiration. Dry matter is distributed between shoot (WSH) and root VRT), both in kg ha^{-1}; fixed fractions of respectively 0.7 and 0.3 have been osen for the purpose of this example.

With this information the equations characterizing the model can be written.

By using a simulation language, such as CSMP, these equations can be easily coded into a small set of statements comprehensible to reader and computer. Figure 17 represents the actual computer program of the simple model that is developed above. A computer program is normally the last stage in the construction of a 'working version' of a simulation model. When the program is submitted to a computer, the output gives a picture of the behaviour of the model. Numerical values of state variables are computed starting from their initial value at simulation time zero until the end of the simulation period. For defined time intervals, a list of calculated values of variables in which one is interested can be printed.

We will discuss the program of Figure 17 line by line. It is useful to start by identifying the program with a TITLE statement:

```
TITLE DRY MATTER PRODUCTION
```

Subsequently the structure of the model is defined. The goal of the simulation is the total dry matter weight (TWT), which is the sum of the dry matter weight of the shoots (WSH) and the roots (WRT):

```
TWT = WSH + WRT
```

WSH and WRT are state variables in the model that change according to their characteristic growth rates GSH and GRT. Integration of these growth rates gives the actual values of WSH and WRT at any moment. CSMP provides the INTGRL function (Table 2) as a means to integrate numerically a specified rate in time:

```
WSH = INTGRL (WSHI, GSH)
WRT = INTGRL (WRTI, GRT)
```

Initial conditions (WSHI, WRTI) and relevant rates (GSH, GRT) have to be specified as arguments of the INTGRL function and are placed between brackets. At time zero, WSH equals WSHI and the current value of WSH at any time is found by integrating GSH. A similar reasoning is valid for WRT. At the beginning of the growth season, shoot and root dry weight can be estimated at 50 kg ha^{-1}. In an INCON statement we can assign numerical values to these INitial CONditions:

```
INCON WSHI = 50., WRTI = 50.
```

The growth rates GSH and GRT are calculated as

```
GSH = 0.7 * GTW
GRT = 0.3 * GTW
```

in which GTW is the net rate of total dry matter increase (kg ha^{-1} d^{-1}). As illustrated in Figure 16, we can express GTW as

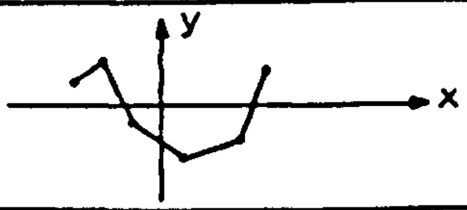
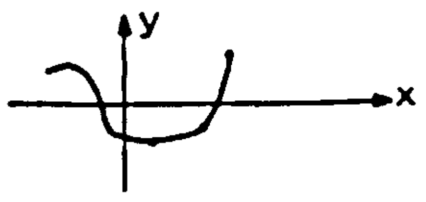
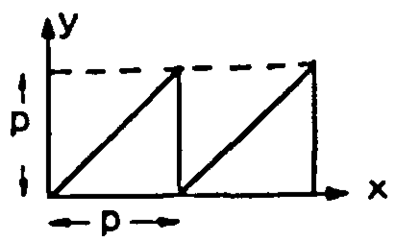
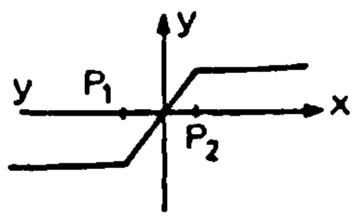
```
GTW = (GPHOT - MAINT) * CVF
```

Figure 17. Listing of the simulation program for calculation of 'dry matter production' and its output.

```
TITLE DRY MATTER PRODUCTION
      TWT   =WSH+WRT
      WSH   =INTGRL(WSHI,GSH)
      WRT   =INTGRL(WRTI,GRT)
INCON WSHI=50., WRTI=50.
      GSH   =0.7*GTW
      GRT   =0.3*GTW
      GTW   =(GPHOT-MAINT)*CVF
      MAINT =(WSH+WRT)*0.015
      GPHOT =GPHST*(1.-EXP(-.7*LAI))
      LAI   =AMIN1(WSH/500.,5.)
PARAM CVF=.7, GPHST=400.
TIMER FINTIM=100., DELT=1., PRDEL=5., OUTDEL=5.
METHOD RECT
PRINT TWT,WSH,WRT,GTW
OUTPUT TWT
END
STOP
ENDJOB
```

```
1  DRY MATTER PRODUCTION
0  TIME            TWT             WSH            WRT            GTW
  0.0000000D+00   100.00          50.000         50.000         17.880
  5.0000000D+00   243.08          150.16         92.924         50.534
  1.0000000D+01   614.06          409.84         204.22         115.80
  1.5000000D+01   1347.1          922.94         424.12         188.94
  2.0000000D+01   2384.7          1649.3         735.40         227.14
  2.5000000D+01   3542.1          2459.5         1082.6         233.86
  3.0000000D+01   4689.0          3262.3         1426.7         222.31
  3.5000000D+01   5777.5          4024.2         1753.2         210.88
  4.0000000D+01   6810.0          4747.0         2063.0         200.04
  4.5000000D+01   7789.4          5432.6         2356.8         189.76
  5.0000000D+01   8718.4          6082.9         2635.5         180.00
  5.5000000D+01   9599.7          6699.8         2899.9         170.75
  6.0000000D+01   10436.          7285.0         3150.7         161.97
  6.5000000D+01   11229.          7840.1         3388.6         153.64
  7.0000000D+01   11981.          8366.7         3614.3         145.74
  7.5000000D+01   12695.          8866.2         3828.4         138.25
  8.0000000D+01   13371.          9340.0         4031.4         131.14
  8.5000000D+01   14014.          9789.5         4224.1         124.40
  9.0000000D+01   14623.          10216.         4406.8         118.01
  9.5000000D+01   15200.          10620.         4580.1         111.94
  1.0000000D+02   15749.          11004.         4744.6         106.19
1$$$ SIMULATION HALTED FOR FINISH CONDITION  TIME      100.00
     TIME         TWT
   0.00000E+00    100.00     +           I           I           I           I
     5.0000       243.08     +           I           I           I           I
     10.000       614.06     I+          I           I           I           I
     15.000       1347.1     I---+       I           I           I           I
     20.000       2384.7     I------+    I           I           I           I
     25.000       3542.1     I---------+ I           I           I           I
     30.000       4689.0     I-----------I-+         I           I           I
     35.000       5777.5     I-----------I----+      I           I           I
     40.000       6810.0     I-----------I-------+   I           I           I
     45.000       7789.4     I-----------I----------+I           I           I
     50.000       8718.4     I-----------I-----------I-+         I           I
     55.000       9599.7     I-----------I-----------I----+      I           I
     60.000       10436.     I-----------I-----------I------+    I           I
     65.000       11229.     I-----------I-----------I---------+ I           I
     70.000       11981.     I-----------I-----------I-----------+           I
     75.000       12695.     I-----------I-----------I-----------I-+         I
     80.000       13371.     I-----------I-----------I-----------I---+       I
     85.000       14014.     I-----------I-----------I-----------I-----+     I
     90.000       14623.     I-----------I-----------I-----------I-------+   I
     95.000       15200.     I-----------I-----------I-----------I---------+ I
     100.00       15749.     I-----------I-----------I-----------I-----------+
1$$$ CONTINUOUS SYSTEM MODELING PROGRAM III    V1M3    EXECUTION OUTPUT $$$
```

Table 2. Some CSMP III functions. From: IBM (1975).

CSMP III Functions	Equivalent Mathematical Expression
Integrator Y=INTGRL(IC,X) where: IC = y_{t_0}	$y(t) = \int_{t_0}^{t} x\,dt + y(t_0)$ where: t_0 = start time t = time
Arbitrary function generator (linear interpolation) Y =AFGEN(FUNCT,X)	y = f(x) 
Arbitrary function generator (quadratic interpolation) Y=NLFGEN(FUNCT,X)	y = f(x) 
Modulo function Y=AMOD (X,P)	$y = x - nP$ n is an integer value such that $0 \leq y < P$ 
Limiter Y= LIMIT(P1,P2,X)	$y = p_1\ ;\ x < p_1$ $y = p_2\ ;\ x > p_2$ $y = x\ ;\ p_1 \leq x \leq p_2$ 
Not Y=NOT(X)	$y = 1$ if $x \leq 0$ $y = 0$ if $x > 0$
Input Switch Relay Y=INSW(X1,X2,X3)	$y = x_2$ if $x_1 < 0$ $y = x_3$ if $x_1 \geq 0$

The maintenance respiration is supposed to be 1.5 percent of the total dry matter per day:

MAINT = (WSH + WRT) * 0.015

The equation respresenting the relation of the gross photosynthetic rate and green surface has an exponential form:

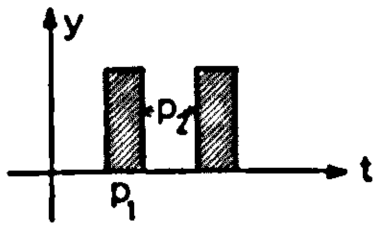
GPHOT = GPHST * (1. − EXP (−0.7 * LAI))

The leaf area index (LAI) is here assumed to be proportional to the shoot dry matter (WSH) to a maximum of 5 ha ha^{-1}. The AMIN1 function (Table 3) can be used to achieve this:

LAI = AMIN1 (WSH/500., 5.)

The AMIN1 function takes the minimum value of its arguments separated by comma's.

Dead time (DELAY) Y=DELAY(N,P,X) where: P=delay time N=number of points sampled in interval p (integer constant) and must be ≥3, and ≤ 16,378	$y = x(t-p)\ ;\ t > p$ $y = 0\ ;\ t < p$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = e^{-ps}$
Implicit function Y=IMPL(IC,P,FOFY) where: IC=first guess P = error bound FOFY=output name from final statement in algebraic loop definition	$y = f(y)$ $\lvert y - f(y) \rvert \leq p \lvert y \rvert$

Impulse generator
 	
Y=IMPULS(P1,P2)
 	
Where: P1=time of first pulse
P2=interval between pulses
	$y = 0\ ;\ t < p_1$
	
	$y = 1\ ;\ (t-p_1) = kp_2$
	
	$y = 0\ ;\ (t-p_1) \neq kp_2$
	
	$k = 0, 1, 2, 3, \cdots$
	
	

The parameters CVF and GPHST have to be specified to complete the numerical information for the program. The PARAMeter statement is used to assign values to variables used as parameters:

PARAM CVF = 0.7, GPHST = 400.

So far the structure of the model has been transformed into a simulation program that can be executed. Only timing, output format and an appropriate integration method must still be specified. A TIMER statement gives the time of finishing the simulation (FINTIM), the printed output interval (PRDEL) and plotted output interval (OUTDEL), and the size of the time step for integration (DELT):

TIMER FINTIM = 100., DELT = 1., PRDEL = 5., OUTDEL = 5.

All TIMER variables are expressed in days, as this is the basic unit of time in this program. A numerical integration method is selected from a set of available

Table 3. Some FORTRAN functions, which can be used in CSMP III statements. From: IBM (1975).

FORTRAN Functions	Equivalent Mathematical Expression
Exponential Y = EXP (X)	$y = e^x$
Trigonometric sine Y = SIN(X)	$y = \sin(x)$
Trigonometric cosine Y = COS (X)	$y = \cos(x)$
Square root Y = SQRT (X)	$y = \sqrt{x}$
Largest value (Real arguments and output) Y = AMAX1(X1, X2)	$y = \max(x_1, x_2)$
Smallest value (Real arguments and output) Y = AMIN1 (X1,X2)	$y = \min(x_1, x_2)$

routines in CSMP (Subsection 2.3.7). For instance, to perform the simulation using the rectangular method after Euler, the next statement has to be used:

```
METHOD RECT
```

Printed output of the variables is obtained by means of:

```
PRINT TWT, WSH, WRT, GTW
```

Plotted out can be obtained by:

```
OUTPUT TWT
```

The END statement defines the end of the simulation model and the STOP statement the end of the simulation program.

```
END
STOP
ENDJOB
```

The ENDJOB statement finishes the computer job. Figure 17 shows a complete listing of the example model and the output after execution.

Exercise 11
Run the program. (Subsection 2.2.5 may be helpful when you are editing your program).

During a simulation run all statements defining the structure of the model are executed several times, the number of which equals FINTIM divided by DELT, when using the rectangular method. DELT, equivalent to Δt in Section 2.1, plays a critical role since it indicates the size of the time interval for integration. For a CSMP statement with an integral function like:

WSH = INTGRL (WSHI, GSH)

the equivalent numerical expression when using a rectangular integration method, as in Subsections 1.1.3 and 2.1.5, is:

$$\mathrm{WSH_{now} = WSH_{previous} + GSH \cdot DELT}$$

From this equation it is clear that the rectangular integration method implicates a constant growth rate during the whole time step DELT. Taking appropriate time steps, such that the rate of change can be regarded as effectively constant, will give good approximations of the value of the state variable (cf. Subsection 2.3.6).

Exercise 12
a. Simplify the program for the case that you are only interested in the total dry matter increase (do not consider biomass distribution into shoots and roots). Assume that the crop is completely covering the soil (GPHOT = GPHST). Run this program.
b. Explain why the results of this simple model are qualitatively similar to the filling of a water tank as discussed in Subsection 2.1.4. Write the governing differential equation for this system and derive from this the time coefficient and equilibrium level.
c. What is the implication of the introduction of the relationships in which gross photosynthesis depends on LAI, and LAI depends on WSH in the form discussed above? What happens with the time coefficient?

2.2.3 Program modifications with forcing functions

Until now we assumed that all external conditions, or driving variables (Subsection 1.1.2), are constant. However, CO_2 assimilation rates are strongly dependent on irradiation and irradiation itself can vary from day to day over a relatively wide range. If we want to make the model realistic, we should introduce the actual irradiation level instead of a fixed rate of gross CO_2 assimilation. Irradiation will drive CO_2 assimilation in the model and is not affected by the state of the system. Therefore we call it a driving or forcing function. More generally, forcing functions are defined as those variables that are not affected by processes within the system, but characterize the influence from outside (Subsection 1.1.3). They are an input to the model. In very simplified models, forcing functions can be introduced in the program as parameters:

```
PARAM GPHST = 400.
```

Very often forcing functions show a characteristic pattern over a certain time period (day or year). An equation describing such regular fluctuations in time may provide approximate values for the use in a general program. For example, the yearly course of daily incoming short-wave irradiation is reasonably well represented by a sinusoidal curve, just like daily gross photosynthesis. For the daily gross photosynthesis (GPHST, expressed as glucose) the equation is

```
GPHST = 300. + 200.*SIN(2*PI*(DAY-(365./4.)+10.)/365.)
PARAM PI = 3.141592
```

DAY stands for the number of a day in the year; counting starts from 1 January. GPHST reaches a minimum value of 100.0 (kg ha^{-1} d^{-1}) on 21 December (DAY = 355) and a maximum value of 500.0 on 21 June (DAY = 172). DAY can be calculated by:

```
DAY = STDAY + TIME
PARAM STDAY = 60.
```

In this way simulation starts on 1 March (STDAY = 60.). TIME is a variable generated by CSMP which expresses the current time during simulation. At the start of simulation TIME = 0., and its value is augmented by DELT when the integration of all state variables is accomplished. (The symbolic name TIME is reserved by CSMP to keep track of time, and cannot be used for other purposes). A second way of keeping track of the number of the day is to give the variable TIME an initial value. This can be done in the TIMER statement:

```
TIMER TIME = 60., FINTIM = 210., DELT = 1., PRDEL = 5.
```

When the time course of a driving variable has been measured, direct use of these values is another option for formulating a forcing function. Measured values, with the corresponding dates can be introduced by a FUNCTION statement:

```
FUNCTION GPHSTB = (60., 300.), (100., 400.), (150., 450.), (210., 500.)
```

In this statement, an ordered set of pairs defines the content of a table, named GPHSTB. The first value in each pair of numbers between parentheses stands for the independent variable (TIME, d), the second expresses the dependent variable (GPHST, kg ha^{-1} d^{-1}). The distance between the coordinates of two pairs does not have to be equal in size; the value of the independent variable should always increase, but the dependent variable can vary in an arbitrary way. The current value of the dependent variable is calculated by means of an AFGEN function (Arbitrary Function GENerator), which performs a linear interpolation in the function table. Function name and independent variable should be specified as arguments of the AFGEN function:

```
   GPHST = AFGEN(GPHSTB, TIME)
```

Exercise 13
The gross photosynthetic rate, expressed as CO_2 assimilated, is proportional to the absorbed short-wave radiation with a proportionality factor of 4.35 10^{-9} kg $Joule^{-1}$; the conversion factor from CO_2 to CH_2O (glucose) is 30.0/44.0. Use following values for daily totals of incoming visible (400-700 nm) irradiation (DTR) in 10^6 Joule m^{-2} d^{-1} on a standard clear day for a latitude of 50° N (Goudriaan & van Laar, 1978):

```
   FUNCTION RADTB = (15., 2.61), (46., 4.80), (74., 8.07), ...
                    (105., 12.20), (135., 15.44), (166., 17.01), ...
                    (196., 16.41), (227., 13.75), (258., 9,80), ...
                    (288., 5.96), (319., 3.19), (349., 2.11)
   DTR = AFGEN (RADTB, TIME) * 1.E6 * 1.E4
```

Assume that 10% of the irradiation is reflected. Formulate the new equations and replace the former equations of the model of Figure 17. Use the program to calculate the dry matter production from Day 60 until Day 210.

Another example of a forcing function that plays a critical role in growth models is temperature. The next exercise illustrates how maintenance respiration can be modelled more realistically by taking the effect of temperature into account.

Exercise 14
Extend the program of Exercise 13 so that maintenance respiration becomes dependent on temperature. Assume a linear temperature course for the simulation period between 10 °C on the first day and 20 °C on the last day. Use the following table:

TEMPERATURE	10.	20.	30.
MAINTENANCE COEFFICIENT	0.008	0.015	0.030

Repeat the simulation for a more realistic course of the temperature.

AFGEN provides a simple linear interpolation between 2 points (x_1, y_1) and (x_2, y_2). The function value y for a certain x is expressed as:

$$y = y_1 + (y_2 - y_1) \cdot (x - x_1)/(x_2 - x_1)$$

In a sense, a broken line is generated by connecting subsequent points. If we want to employ a higher order interpolation we need more than two points: three points for a parabolic and four points for a cubic relation. Generally, through n points fits a curve of order $n-1$. The CSMP function NLFGEN (Non Linear Function GENerator) does this job with $n = 3$. Although theoretically NLFGEN is applicable in many cases, caution should be exercised in its use. For an AFGEN function it is not so difficult to imagine what happens: straight lines connect two points. The situation is much more complicated for a NLFGEN function which fits a parabolic interpolation function to three neighbouring points. For this reason application of NLFGEN is less self-evident as it might be for the AFGEN function. In particular, if the data points imply an abrupt discontinuity, we should take into account large distortion there if using NLFGEN. For such situations it may be preferable to use AFGEN. Exercise 15 illustrates this hidden danger of using NLFGEN.

Exercise 15
Consider the next program:

```
FUNCTION XTB = (0., 1.), (1., 1.), (2., 0.), (3., 0.)
X1 = AFGEN (XTB, TIME)
X2 = NLFGEN (XTB, TIME)
TIMER FINTIM = 3.0, OUTDEL = 0.1, DELT = 0.1
OUTPUT X1, X2
```

Compare the output of the AFGEN and NLFGEN function.

2.2.4 *The structure of the CSMP language*

CSMP is a problem-oriented language designed to facilitate the digital simulation of continuous processes on large-scale digital computers. The advantage of using such a language is that it simplifies the programming. The user is not concerned with the rather difficult programming of numerical integration and interpolation methods, and he need not to worry about the computational order of the statements. (We will return to this important point). A convenient output form is provided by the program itself. The programmer is only responsible for writing the statements that define the model and supplying it with a proper data set. An additional advantage specific for CSMP is that the full capability of the widely used FORTRAN language is available. In addition to FORTRAN facilities, CSMP includes a set of functions that are particularly suited to working with a continuous system (e.g. INTGRL and AFGEN functions).

Every simulation run essentially starts from a well defined initial condition. When needed, one can separate this part from the description of the structure of the model itself. The determination of the final situation of a simulation run and decision for a possible new run can be separated as well in a program. To do so, a CSMP model can be divided into three segments – INITIAL, DYNAMIC and TERMINAL – that describe the computations to be performed before, during and after each simulation run. The TERMINAL segment will not be discussed here.

The INITIAL segment is exclusively used for initialization of variables and computations of variables to be expressed in more basic parameters. Statements in the INITIAL segment are executed only once per simulation run. Assume for example the initial weights of shoot and root depend on the weight of the seed sown, then the simulation model can be modified as:

```
TITLE DRY MATTER PRODUCTION
INITIAL
INCON SEED = 150.
WSHI = SEED * 0.6 * 0.5
WRTI = SEED * 0.6 * 0.5
```

The factor 0.6 is a reasonable value for conversion of seed into plant material; a shoot:root ratio of 1. is used here.

The DYNAMIC segment contains the complete description of the system dynamics, together with any other computations and decisions to be performed for successful simulation. For most simple models, the DYNAMIC segment consists of one section:

```
DYNAMIC
TWT = WSH + WRT
WSH = INTGRL (WSHI, GSH)
.      and so on, as in Figure 17 until:
.
.
END
```

In more complicated systems, the DYNAMIC segment can be divided in several sections, each section dealing with a separate submodel (Subsection 2.3.2).

Specification of an INITIAL segment is optional and is often omitted for small models. In that case the DYNAMIC segment has not to be declared explicitly by the DYNAMIC label.

One of the main advantages of CSMP is its sorting routine. It enables the user to write a simulation program with its statements in the same order as he thinks about the process or system and in which he considers it most lucid and readable. Such an order of statements, however, is often the reverse of what the computational order must be. The CSMP sorting routine finds the proper order from any sequence of statements presented. Sorting of statements is necessary because computations have to be performed in a correct order: calculation of rate variables must always preceed the updating of state variables. This is a consequence of the concept of dynamic simulation (Section 1.1). Also all variables, used to compute rate variables at time t, must have the appropriate value, and not one corresponding with one time interval DELT earlier or later. Sequencing statements in an appropriate computational order is also required if one programs in FORTRAN, but with FORTRAN the sequencing must be done by the programmer (and one is usually not warned when it is done incorrectly).

The FORTRAN program that results from the sorting and some other conversions by the CSMP compiler is called UPDATE. It is accessible like other computer generated files.

Exercise 16

Put the statements of the model of Subsection 2.2.2 in a computational order, and notice the difference with the order presented. Request the translation of the program into FORTRAN (the UPDATE-version) by submitting the program. Compare the results with your own sorting. Spell a name incorrectly, and see what happens. What can occur if a statement is incorrect. For example try

```
MAINT = (GSH + GRT) * 0.015
```

The CSMP programming system sorts the statements in the INITIAL and DYNAMIC segments automatically, independently of each other.

2.2.5 Some basic CSMP programming rules

To write a correct CSMP program, a minimum knowledge of the common expressions of this language is necessary. The intention of this part is to provide a summary of frequently used CSMP statements. Readers who want to know more are referred to a CSMP manual (IBM, 1975).

Data statements

Data statements are used to assign numeric values to parameters, constants and initial conditions. For instance:

```
PARAM P1 = ...., P2 = ....
CONSTANT C1 = ...., C2 = ....
INCON I1 = ...., I2 = ....
```

Parameters specified in a PARAM statement are constant during the simulation run. Variables can be introduced by means of a FUNCTION label (see Subsection 2.2.3).

Structure statements

Structure statements describe the functional relationships between the variables of the model. FORTRAN statements can be used within a CSMP program, and all FORTRAN functions are valid (Table 3). Some examples of structure statements:

```
Y = (A + B) * C
ROOT = SQRT (X * * 2 + Y * * 2)
A = INTGRL (2., X * * 2 + R/D)
```

For more information about available CSMP and FORTRAN functions see Tables 2 and 3 and particularly the Program Reference Manual (IBM, 1975).

Expressions should be written at the right hand side of the equal sign and their numeric value is assigned to the variable at the left. The calculation of an expression is performed according to the standard hierarchy:

- evaluation of brackets (in combination with FORTRAN or CSMP functions)
- exponentiation (* *)
- multiplication and division (*, /)
- addition and subtraction (+, −)

Operators of the same hierarchy are performed from left to right.

Output control statements

The TITLE statement allows the user to specify the program and it appears at the top of each page of the output listing. A PRINT statement is used to specify variables whose values will be printed at each PRDEL interval. For output of some variables in printed graph form the OUTPUT statement is used. For examples see Subsection 2.2.2, and Table 9 and Figure 24 of Section 3.1.

Execution control statements

In a TIMER statement we specify the values of certain system variables.

FINTIM :	finish time for terminating a simulation
OUTDEL:	time interval for print-plot output
PRDEL :	time interval for printing the values of requested variables
DELT :	integration interval (see also Subsection 2.3.5)
TIME :	initial value of time, to be specified only if not zero.

A condition to terminate the simulation, e.g. when TWT exceeds 20 000 kg ha^{-1}, can be introduced by a FINISH label:

```
FINISH TWT = 20000.
```

Also the integration method is specified in an execution control statement, for instance:

```
METHOD RECT
```

If METHOD RECT is not specified, the RKS method is used by default (see Section 2.3).

The statement

END	: completes the specifications of the model.
STOP	: terminates the simulation run(s)
ENDJOB	: terminates the job.

Reruns

If the simulation is to be repeated with new data and/or output and execution control statements, these statements are to be placed between two END-statements:

```
. . . . .
END
PARAM . . .
TIMER . . .
END
STOP
ENDJOB
```

Only data statements with labels such as PARAM, CONST, INCON and FUNCTION can be used to specify reruns, not equations. Running a program a number of times for multiple values of a parameter is induced by writing their values between parentheses. For example:

```
PARAM GPHST = (300., 400., 500.)
```

This feature can be used for only one variable in each rerun.

Exercise 17
Use these features to study in only one program the total dry matter production for all combinations of the values 300., 400., 500. for GPHST and values 400., 500., 600. for the ratio kg (shoot dry matter) ha^{-1} (leaf surface) in the calculation of LAI. Start from the program described in Subsection 2.2.2.

Syntax

Some syntactic rules may be helpful when editing a program.

- maximum 6 characters for names of variables
- each statement on one line
- a statement followed by three dots (...) means that statement will be continued on the next line; this is not allowed within a MACRO! (Subsection 4.2.3)

- spaces between variable names and operators are allowed
- columns 1 to 72 can be used for the program, columns 73-80 are for identification
- statements can begin in any column, except for ENDJOB, which must begin in the first column
- * in the first column stands for comment.

Tracing errors

If the proper Job Control Language is used, the computer will give an answer to your problem. If the program is not free of errors, the computer will give one or more error messages. Sometimes these messages are self-explanatory, otherwise you should consult the CSMP reference manual (IBM, 1975).

If the computer is generating no diagnostics and the results seem to be correct, you still cannot be sure whether your program is free of errors, especially when the program is large and has a complex structure. Some ways of checking are:

- make sure the dimensions are correct. Check them. This can also be quite instructive as one can learn more about the significance of the various coefficients and parameters
- run your program for extreme conditions
- make use of the DEBUG feature. For example, write just before the END statement:

```
NOSORT
         CALL DEBUG (2, 10.)
```

All variables, not only those specified in the output list, are printed twice, first at time 10 and then after one integration step. This allows you to check all computations.