2.3 Some techniques in dynamic simulation

J. Goudriaan

2.3.1 Introduction

Good quality dynamic simulation can only be attained if a few basic techniques are mastered. CSMP, introduced in Section 2.2, provides the model builder with a package of convenient features. Some of these features, their principles and pitfalls will now be explained so that they can be used more efficiently and lead to more reliable results. For further studies of numerical techniques, good handbooks are available, for example that of Scheid (1968). Sequencing of program statements, how sorting can be suppressed, and how sortable blocks can be constructed that are internally sequential, is discussed in Subsection 2.3.2.

The MACRO feature of CSMP can be used when a piece of program is repeated several times in the same model with different names. Defined once in the beginning, this piece of program is written by the CSMP translator in all places where a MACRO is invoked. This is explained in Subsection 2.3.3. Solution of implicit equations is possible with iteration techniques. The CSMP provided IMPLicit loop will be discussed as well as a self-written method (Subsection 2.3.4). Next, numerical integration is discussed (Subsections 2.3.5, 2.3.6 and 2.3.7). Characteristics of the modelled system, such as occurrence of feedback and discontinuities and the time coefficient determine which integration method should be used, and also the time interval of integration.

2.3.2 Sorting

The basic rule for sorting program statements in a computational order is that any variable used on the right-hand side of the equal sign in a statement should have been defined in an earlier stage of the program. This simple rule is incorporated in the CSMP translator that prepares the source program for proper handling by the FORTRAN compiler (Subsection 2.2.4).

However, situations exist where the programmer would like to cancel this rule. For example, during the night there is no point in performing laborious calculations of light extinction, so one would rather skip this part of a model. That means that a conditional jump must be inserted: an instruction to jump over a number of statements if the irradiation level is zero. Once the programmer has determined which statements must be skipped, he would not like to see the CSMP translator disturbing the well-balanced sorting effort, and so he inserts a command that the translator should not touch these statements. There are two such commands: NOSORT and PROCEDURE.

The label NOSORT is valid until it is cancelled by the label SORT; so a non-sorted block of statements between the labels NOSORT ... SORT is created. Statements outside this block can be freely sorted by the CSMP translator, except that they cannot be moved across the NOSORT block: NOSORT cuts the program into sections.

If complete sortability is desired, the PROCEDURE label must be used. The translator will consider the group of statements between this label and the label ENDPRO as one big sortable statement. Consequently the translator demands a specification of input and output variables of this block, so that it can properly locate it between the other program statements: a PROCEDURE does not cut a program into sections. For example, suppose we need the sum of the integer numbers between N1 and N2, which are calculated somewhere else in the program. The result S is used somewhere else. Then we can make a program with a PROCEDURE:

```
FIXED     I, N1, N2
PARAM     N1=0
          N2= TIME
PRINT     S
TIMER FINTIM=10., PRDEL=3.3
PROCEDURE S=SUM(N1, N2)
          S=0.
          IF(N2.LT.N1)GOTO 10
          I=N1
       11 S=S+I
          I=I+1
          IF(I.GT.N2)GOTO 10
          GOTO 11
       10 CONTINUE
ENDPRO
END
STOP
ENDJOB
```

In the PROCEDURE for summing, a few new statements are used. The variables I, N1 and N2 are declared integer by the label FIXED. In an IF statement the values of two variables are compared with each other to see, for instance, whether the left one is greater than (GT) or less than (LT) the right one. If this is true the computer follows the GOTO command and jumps to the line starting with that number. If it is not true the computer continues with the next line. The unconditional GOTO 11 command will make the computer return to previous lines to repeat a block of statements. In the paragraph about iteration techniques, we shall see an application of a PROCEDURE.

2.3.3 *Programming with CSMP MACRO's*

In a MACRO the structure of a process is described that different species or different model components have in common. In this sense it resembles a FORTRAN subroutine, but the difference is that it is not invoked in the execution phase, but in the translation phase. Every time a MACRO is invoked, the CSMP translator will expand it to the full text with the appropriate symbols. After this has been completed, the CSMP sorting routine will become operative so that inside the definition of a MACRO, the statements need not be in computational order.

As an example, the simple crop growth model (Subsection 2.2.2) is extended to deal with competition. In a mixture, light absorption will supposedly be proportional to the leaf area of each species. First the MACRO definition:

```
MACRO TWT, LAI=GROWTH(TWTI,MC,CVF,ALU,LAR)
      TWT=INTGRL(TWTI,GTW)
      GTW=(GPHOT - MC*TWT)*CVF
      GPHOT=ALU*AVIS
      AVIS=IVIS*(1.- EXP(-0.7*LAIT))*0.9*LAI/LAIT
      LAI=TWT*LAR
ENDMACRO
```

In this MACRO the following new variables and coefficients occur:

ALU	average light use efficiency, expressed as glucose formed per amount of light absorbed	4.E-5	kg m^2 ha^{-1} J^{-1}
AVIS	absorbed visible radiation		J m^{-2} d^{-1}
CVF	conversion factor of glucose into biomass	0.7	kg kg^{-1}
IVIS	incoming visible radiation	10.E6	J m^{-2} d^{-1}
LAR	leaf area ratio	1.E-3	ha kg^{-1}
MC	maintenance coefficient, expressed in glucose per dry weight	0.015	kg kg^{-1} d^{-1}

These numerical values only give an indication of a reasonable number, and are by no means natural constants.

A MACRO definition must be placed before the label INITIAL. It is clear that the sequence of the statements is not computational here. (If desired, the PROCEDURE feature can be included in a MACRO definition). The structure that is given in the MACRO definition will be written in the main program each time the CSMP translator encounters a call to it. Two such calls (for two plant species) are:

```
TWT1,LAI1=GROWTH(TWTI1,MC1,CVF1,ALU1,LAR1)
TWT2,LAI2=GROWTH(TWTI2,MC2,CVF2,ALU2,LAR2)
```

It is good practice to put the input variables of the MACRO in the right hand list. In the main program these variables should either be calculated or be given as parameters and initial constants. In this example TWTI1 and TWTI2 must be

given separately on an INCON line, and likewise MC1, MC2, CVF1, CVF2, ALU1, ALU2, LAR1 and LAR2 on PARAM lines. There may also be variables that are common to both species, here for example LAIT. Therefore this variable does not occur in the argument list and is defined in the main program:

```
LAIT=LAI1 + LAI2
```

Obviously the TIMER and output control statements must be specified in the normal way.

A variable like GTW, which is used only inside the MACRO, is not known in the main program, because its name was not listed as an argument. It can be made common to the main program by including it in the argument list.

Exercise 18
Complete the program that was started here, run it and study the UPDATE, as well as the output.

2.3.4 Iterative techniques

An iterative procedure in a simulation model can be used to calculate the value of a state variable, when its time coefficient (Subsection 2.1.7) is small in comparison with the integration interval. This technique can be useful in complex models that span three hierarchical levels (Subsection 1.4.4). Then we may assume that this state variable is in equilibrium with its environment after each time step. This steady state can be found by an iterative technique that searches the point where the rate of change of the integral is zero. Of course, the sign of the feedback that controls the rate of change must be negative. As an example we shall treat the simulation of production and consumption of assimilates that flow through a reserve pool. The time coefficient of such a reserve pool is in the order of half a day, so that every day must take care of its own demand. When a diurnal course is simulated and the integration step is on the order of minutes, the reserve pool can be treated as a state variable (Subsection 3.3.3). For simulation of the growth throughout a season, a convenient time step is one day and the equilibrium level of reserves must be found by iteration. For comparison, we shall first consider the state variable formulation:

```
TITLE   RESERVES AS A STATE VARIABLE
INCON   RESLI=0.1
        RESI=RESLI*TWT
DYNAMIC
        RESL=RES/TWT
        RES=INTGRL(RESI,GPHRED-MAINT-CGR)
```

```
            MAINT = 0.015 * TWT
            CGR = 0.1 * RES/(RESL + KRESL)
PARAM       KRESL = 0.1, TWT = (2000., 10000.)
            GPHRED = GPHOT * REDF
            GPHOT = GPHST * (1. - EXP(-0.7 * LAI))
            REDF = AFGEN(REDFT, RESL)
            LAI = AMIN1(WSH/500., 5.)
            WSH = 0.7 * TWT
FUNCTION REDFT = 0., 1., 0.2, 1., 0.25, 0., 0.5, 0.
PARAM       GPHST = 400.
TIMER       FINTIM = 20., PRDEL = 1.
PRINT       RES, RESL, GPHRED
END
STOP
ENDJOB
```

In the INITIAL the reserve level is set to 0.1 of the total plant weight TWT, and then the initial amount of reserves RESI is calculated. In the DYNAMIC the amount of reserves RES is integrated with a rate calculated as gross photosynthesis GPHRED, minus maintenance respiration MAINT, minus consumption rate for growth CGR; all rates being expressed in glucose and in kg ha^{-1} d^{-1}. The rate of maintenance respiration is 1.5% of the total dry weight per day and independent of the reserve level. The consumption rate for growth CGR shows a saturation type curve response to the reserve level (Moldau & Sôber, 1981). In this example a hyperbolic relation is used with a Michaelis-Menten constant (KRESL) of 0.1. The maximum rate of consumption is taken as 0.1 of the total dry weight per day.

Exercise 19
What do these numbers mean for the time coefficient of the reserve level?

The rate of gross photosynthesis GPHRED is reduced by some hypothesized inhibiting factor when the reserve level rises above 20%. This reduction is obtained by multiplication with the reduction factor REDF, which drops to zero at and above 25%. The combined action of these processes on the rate of reserve accumulation is given in Figure 18, where GPHRED,MAINT and CGR have been drawn. At the point of equilibrium GPHRED is equal to MAINT + CGR.

In view of the comparison with an iterative solution of the reserve level the total weight of the crop has been fixed during the simulation. The output (Table 4) shows the equilibrium values with TWT = 2000. as 0.22548 for RESL and 168.55 for GPHRED, almost reached in about 2 days. With TWT = 10000. they are 0.0312 and 387.92.

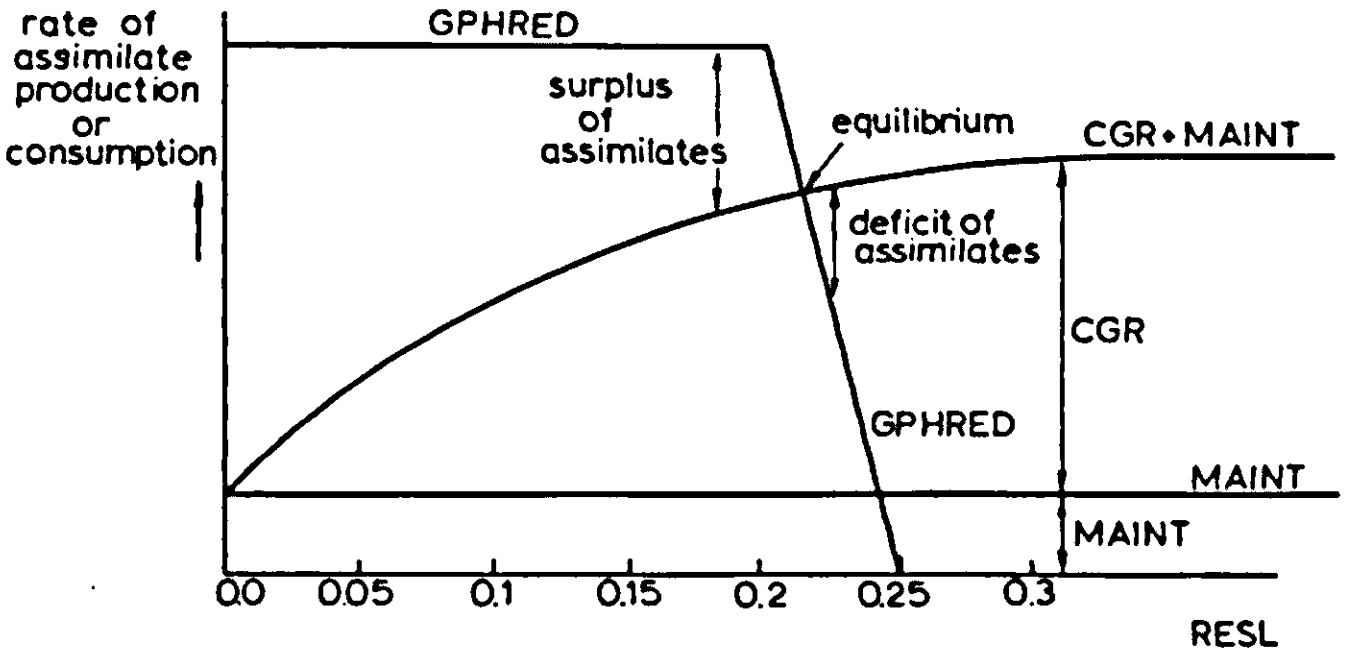

Figure 18. Production and consumption of assimilates as a function of reserve level (RESL).

There are several iterative methods available, of which only two will be discussed. More elaborate methods can be found in handbooks for numerical analysis. All iteration methods have in common that a value for an unknown variable (here RESL) must be found for which two rates are equal. It is then implicitly assumed that sufficient time has passed within the time interval for integration to establish an equilibrium. Here the production rate GPHRED must be equal to the total consumption rate CGR + MAINT. Often this problem is graphically represented as the problem to find the intersect of two lines (Figure 18).

In CSMP, the IMPLicit loop is available as a method to solve this problem. Essentially the method is based on repeated substitution. First a value for RESL is guessed, the corresponding rate of photosynthesis is calculated, then the reserve level is calculated at which the total consumption rate is equal to the previously calculated photosynthesis rate and the procedure is repeated until sufficient convergence is reached. The first drawback of this method is that one of the relations between rate and unknown variable must be inverted: RESL must be written either as a function of total consumption rate or as a function of photosynthesis. When AFGEN functions have been used such an inversion is not possible. Therefore RESL cannot be written as a function of photosynthe-

Table 4. Simulated time course of amount of reserves (RES), the reserve level (RESL) and gross CO_2 assimilation (GPHRED) with the state variable approach.

TIME	RES	RESL	GPHRED
0.	200.00	0.10000	343.66
1.	394.69	0.19735	343.66
2.	449.30	0.22465	174.24
3.	450.90	0.22545	168.72
4.	450.95	0.22547	168.57
5.	450.95	0.22548	168.55

sis, but must be written as a function of CGR by making RESL explicit in the hyperbolic equation. The resulting IMPLicit loop is given in the following simulation program:

```
TITLE   RESERVE LEVEL WITH IMPLICIT LOOP
INCON   RESLI=0.23
        MAINT=0.015*TWT
        GPHOT=GPHST*(1.-EXP(-0.7*LAI))
        LAI=AMIN1(WSH/500.,5.)
        WSH=0.7*TWT

        RESL=IMPL(RESLI,0.0001,RESL1)
        REDF=AFGEN(REDFT,RESL)
        GPHRED=GPHOT*REDF                    } implicit loop
        CGR=GPHRED-MAINT
        RESL1=CGR*KRESL/(0.1*TWT-CGR)

PARAM   GPHST=400.
PARAM   KRESL=0.1,TWT=(2000., 10000.)
FUNCTION REDFT=0.,1.,0.2,1.,0.25,0.,0.5,0.
TIMER   FINTIM=1., PRDEL=1.
METHOD RECT
PRINT   RESL,GPHRED
END
STOP
ENDJOB
```

In this program there is no integral RES. Instead, RESL is calculated as the result of the implicit loop that starts with

```
RESL=IMPL(RESLI,0.0001,RESL1)
```

Here RESLI is the initial guess, 0.0001 is the convergence criterion and RESL1 is the name of the variable that closes the implicit loop. This closing statement is in fact the hyperbolic equation for the dependence of CGR on RESL in which RESL has been made explicit. The three statements in between replace the state variable formulation.

With TWT = 10000., the results of this program correspond with those of the state variable approach.

Exercise 20

Check this by comparing the results of the two methods. Repeat the calculation with both programs for TWT = 2000.

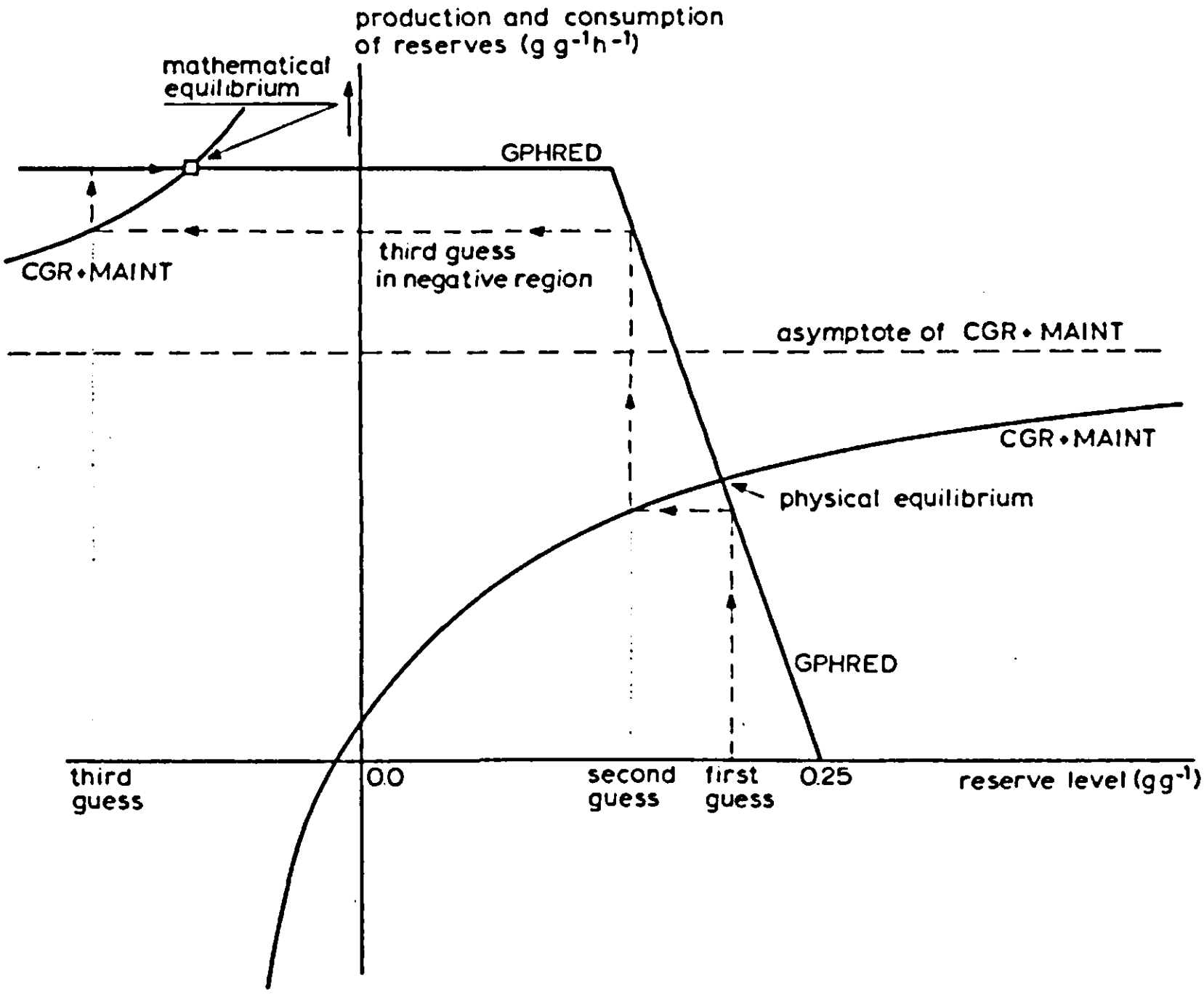

Figure 19. Course of the iteration process followed by the IMPLicit loop to find equilibrium between production and consumption of assimilates.

For TWT = 2000., the results of the implicit loop are disappointing. The calculated reserve level is negative! The reason for this bad behaviour is shown graphically in Figure 19. It is related to divergence of the repeated substitution method. This danger is always present in the CSMP-provided IMPLicit loop and when it strikes it can be resolved by rearrangement of the statements within the implicit loop. However, then the other equation with the unknown variable has to be inverted, which is not always possible. In this example it is hard to write RESL as a function of photosynthesis.

To summarize, the implicit loop may or may not work, dependent on the sequence of the statements. Of course, this is an uncomfortable feature of the implicit loop. In contrast, the halving/doubling method also known as bisection method is absolutely reliable, but its programming requires more work. This method is based on very simple reasoning. First a lower and an upper value are guessed, which certainly comprise the range of the unknown variable. In this example one might choose the values 0. and 0.25. For each of the two guesses the rates of production and consumption are calculated, and, of course, the signs of their differences are opposite. That means that the point where production and consumption match, must lie somewhere in between. Therefore the value halfway is tried and its accompanying production and consumption values.

Figure 20. Listing of CSMP program that employs the halving-doubling iteration method, and its output.

```
TITLE RESERVELEVEL WITH HALVING/DOUBLING ITERATION METHOD
INITIAL
INCON RESL1=0.,RESL2=.25
* LOWER VALUE OF RESL IS ZERO, AND UPPER VALUE IS 25 PERCENT

DYNAMIC
* FOR DEMONSTRATION PURPOSES, TWT IS MADE A FUNCTION OF TIME:
      TWT=20.*TIME

      MAINT=0.015*TWT
      LAI=AMIN1(WSH/500.,5.)
      WSH=0.7*TWT
      GPHOT=GPHST*(1.-EXP(-0.7*LAI))
      GPHRED=GPHOT*AFGEN(REDFT,RESL)


PROCEDURE RESL=ITERA(RESL1,RESL2,GPHOT)
      COUNT=0.
  130 CGR1=0.1*TWT*RESL1/(RESL1+KRESL)
      CGR2=0.1*TWT*RESL2/(RESL2+KRESL)
      AVAIL1=GPHOT*AFGEN(REDFT,RESL1)-MAINT
      AVAIL2=GPHOT*AFGEN(REDFT,RESL2)-MAINT
      COUNT=COUNT+1.
      IF(COUNT.GT.100.)GOTO 150

      IF((CGR1-AVAIL1)*(CGR2-AVAIL2).LT.0.) GOTO 110
* IF THE SOLUTION IS WITHIN THE CHOSEN RANGE GOTO 110

* ELSE DOUBLE THE RANGE INTO THE CORRECT DIRECTION:
      IF(CGR1.GT.AVAIL1) RESL1=2.*RESL1-RESL2
      IF(CGR2.LT.AVAIL2) RESL2=2.*RESL2-RESL1
* AND TRY AGAIN:
      GOTO 130

* RESL3 LIES HALFWAY THE LOWER AND THE UPPER VALUE
  110 RESL3=(RESL1+RESL2)*0.5
      CGR3=0.1*TWT*RESL3/(RESL3+KRESL)
      AVAIL3=GPHOT*AFGEN(REDFT,RESL3)-MAINT
* TEST FOR THE PRESET CONVERGENCE CRITERION:
      IF(ABS(CGR3-AVAIL3).LT.ERROR) GOTO 200

* CRITERION NOT SATISFIED,SO INCREMENT THE COUNT AND HALVE THE RANGE
      COUNT=COUNT+1.
      IF(COUNT.GT.100.) GOTO 150
      IF((CGR3-AVAIL3)*(CGR1-AVAIL1).GT.0.) GOTO 100
* UPPER BOUNDARY IS REPLACED BY THE HALFWAY VALUE:
      RESL2=RESL3
      GOTO 110

* LOWER BOUNDARY IS REPLACED BY THE HALFWAY VALUE:
  100 RESL1=RESL3
      GOTO 110

  150 WRITE(6,800)
  800 FORMAT(' TOO MANY ITERATIONS')
  200 RESL=RESL3

* THE RANGE FOR THE NEXT TIME INTERVAL IS CHOSEN:
      RESL1=RESL3
      RESL2=RESL3+0.001
ENDPRO


PARAM ERROR=0.001
PARAM KRESL=0.1
PARAM GPHST=400.
FUNCTION REDFT=0.,1.,0.2,1.,0.25,0.
TIMER FINTIM=500.,PRDEL=20.,DELT=20.
METHOD RECT
PRINT TWT,RESL,GPHRED
END
STOP
ENDJOB
```

```
1  RESERVELEVEL WITH HALVING/DOUBLING ITERATION METHOD
0  TIME           TWT            RESL           GPHRED
   0.000000D+00   0.00000E+00    0.00000E+00    0.00000E+00
   2.000000D+01    400.00        0.23685         34.125
   4.000000D+01    800.00        0.23434         68.072
   6.000000D+01    1200.0        0.23160         101.81
   8.000000D+01    1600.0        0.22863         135.31
   1.000000D+02    2000.0        0.22548         168.55
   4.000000D+02    8000.0        5.03535E-02     387.92
   4.200000D+02    8400.0        4.53091E-02     387.92
   4.400000D+02    8800.0        4.10078E-02     387.92
   4.600000D+02    9200.0        3.72974E-02     387.92
   4.800000D+02    9600.0        3.40635E-02     387.92
   5.000000D+02    10000.        3.12200E-02     387.92
1$$$ SIMULATION HALTED FOR FINISH CONDITION  TIME      500.00
1$$$ CONTINUOUS SYSTEM MODELING PROGRAM III   V1M3    EXECUTION OUTPUT $$$
```

The sign of their difference is compared with that of the lower value. If it is the same, the halfway value replaces the old lower value in its function as lower value and the procedure is repeated. If the sign is opposite the halfway value becomes the new upper value, and the procedure is repeated. This iteration is continued until the difference between the upper and lower value is reduced to a preset convergence criterion. Every iteration cycle the possible range is halved, so that 10 iterations are required to obtain an accuracy of 1/1000 of the initial range ($2^{10} = 1024$), and by 20 we have a reduction by a factor of 10^6. This method never fails provided there is only one possible solution between the two starting positions.

When we use this method in simulation, most likely the current situation does not differ considerably from that at the previous time interval. It could then be advantageous to use the last solution as one of the starting points, and accept the possibility that the solution is outside the chosen range. We then have to make a provision for being able to double the range in the correct direction until it contains the solution, after which we apply the halving method again. In Figure 20, a self-explanatory listing is given of a CSMP program for the calculation of crop photosynthesis where the halving/doubling iteration method is used. From the output we see that at time 100 when the assumed crop weight is 2000, the result is equal to that of the state variable method. At time 0, when crop weight is assumed zero, of course no result can be obtained. The sturdiness of the method is shown by the absence of zero divisions or overflows for this situation, and resumption of its usual performance at time 20.

A special use of an implicit loop for a crop-water-balance simulation, using one hour time steps, is explained in Subsection 3.3.7.

2.3.5 Some numerical integration methods

Popular numerical integration methods, all available in CSMP, are the rectangular method (METHOD RECT), the trapezoidal method (METHOD TRAPZ), the Runge-Kutta method with fixed integration interval (METHOD RKSFX), and the Runge-Kutta method with a self-adapting integration interval (METHOD RKS). The latter is used by default in CSMP, in the situation that the model builder did not specify a method. To make a proper choice of integration method

for a model, it is worthwhile to consider them a bit further.

The simplest method, the rectangular one, will be discussed first. In this method a rate calculation is followed by an integration step, then time is incremented by the time interval DELT, and the procedure is repeated for the next time interval. Graphically, it is convenient to represent the integral as an area under a curve, where the rate of change corresponds to the function value of the curve. In Figure 21 an example is given. The 'true' area is the area under the continuous curve, and the result of the rectangular method is given by the area of the vertical bars. In this case, where a rising curve is integrated the numerical result is less than the true area; with a decreasing function it would be the opposite. Even if we do not know the exact value of the difference between the numerical result and the 'true' area it is possible to estimate it. In this case the error of the numerical integration is approximately equal to the sum of the areas of the triangles above the vertical bars, formed by connecting their corners.

As a standard example we shall study the integration of the exponential function of time:

$$R = R_0 \cdot EXP(RGR \cdot T) \tag{1}$$

where R is the rate to be integrated, RGR the relative growth rate, T time and R_0 the rate at time zero. Formulated in this way, R is a driving force, that means a function of time only and not dependent on the result of the integration. The dimension of the relative growth rate RGR is T^{-1}, so that the product RGR • T is dimensionless. (It is good practice to verify that arguments of exponentials, logarithms, etc., are dimensionless.) Because the expression for the rate R (Equation 1) can be integrated analytically (which we will consider as the 'true' result), it is possible to find the errors caused by different numerical integration methods. Assuming the value of R_0 is unity, the analytical solution of the integral (A) of R between time T_0 and time T_1 is:

$$A = (EXP(RGR \cdot T_1) - EXP(RGR \cdot T_0))/RGR \tag{2}$$

Using the values RGR = 1, T_0 = 0 and T_1 = 4 we find A = 53.598150.

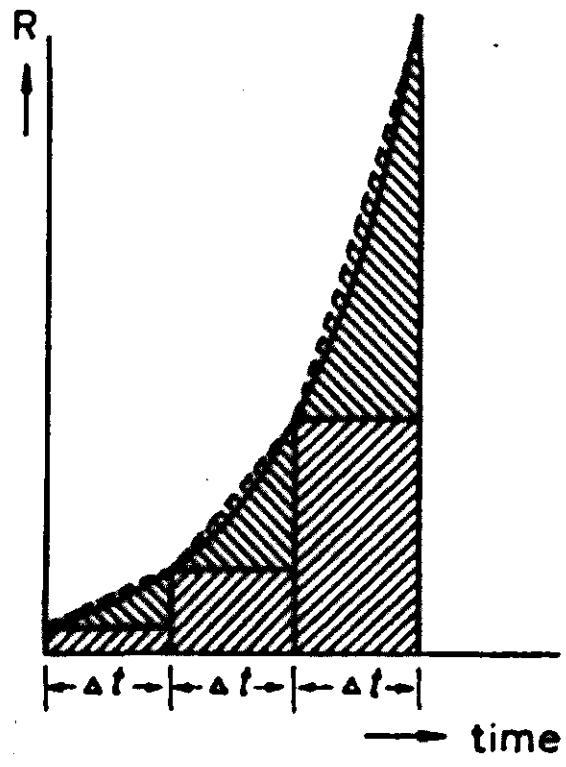

Figure 21. Graphical representation of numerical integration. The rate of change (R) is on the vertical axis, so that areas represent values integrated over time intervals Δt. Exponential function of time (solid line), rectangular integration (▨), trapezoidal integration (▨ + ▧); Δt is time interval of integration.

Exercise 21
Write a CSMP program to find the numerical integral of R with time intervals (DELT) of 1, 0.1 and 0.01. Calculate the error as the difference between the analytical and numerical result. Use the methods RECT, TRAPZ and RKSFX.

2.3.6 Error analysis

Integration of a driving force (no feedback)

Obviously the results are better using TRAPZ, and still more so with RKSFX. The errors of these methods are explained graphically in Figures 21 and 22. By using TRAPZ the triangles above the vertical bars have been included in the integral value, because this method averages the rate at the beginning and at the end of the time interval. Graphically, such averaging can be represented by connecting the corners of the vertical bars by straight lines. We have seen that the error of RECT was just about equal to the area of the triangles formed.

Now we shall try to make a quantitative estimate of the error and how it is related to the time interval of integration. The conclusions of the paragraphs below are summarized in Table 6.

To derive these conclusions, a thorough analysis of the integration procedure is required. For the sake of simplicity this is not done here in full detail. Additional information can be found in Scheid (1968) and Lanczos (1967).

Life is made simple, because R, the rate of change of the integral A of Equation 2, grows exponentially, so that each time interval the same fraction is added to its value and the same relative error is made over and over again. In fact, for RECT the relative error is equal to the ratio of the area of the triangle to that of the vertical bar. This ratio is:

$$E_{rel} = RGR \cdot DELT/2 \qquad (3)$$

Since this ratio does not change, the absolute error at the end is

$$E_{abs} = RGR \cdot DELT \cdot A/2 \qquad (4)$$

Table 5. Numerical integration results (A) and their errors for three values of integration steps with the integration methods RECT, TRAPZ and RKSFX.

DELT	RECT		TRAPZ		RKSFX	
	A	ERROR	A	ERROR	A	ERROR
1	31.193	22.405	57.992	−4.3938	53.616	−0.01807
0.1	50.963	2.6353	53.643	−0.04466	53.598	−0.0000014
0.01	53.331	0.26754	53.599	−0.000446	53.598	−0.0000005

Table 6. Relative errors of the numerical integration results

	With feedback	Without feedback
RECT	$2 \cdot (DELT/TAU) \cdot (TIME/TAU)$	$(DELT/TAU)/2$
TRAPZ	$(DELT/TAU)^2 \cdot (TIME/TAU)/6$	$-(DELT/TAU)^2/12$
RKSFX	$(DELT/TAU)^4 \cdot (TIME/TAU)/120$	$-(DELT/TAU)^4/2880$

We can check with Table 5 that this estimate is quite good, except for DELT = 1, in which case the result is just too far off. Note that A in Equation 4 is numerically not identical to A in Equation 2, as its value depends on method and DELT chosen.

In the trapezoidal method the triangles have been taken into account, so that the error is much smaller. The remaining error is given in Figure 22 as the area between the straight line and the curved solid line. The best estimate of this area is obtained when a parabola is constructed through the function values at time T, T+DELT/2 and T+DELT and the area between the parabola and the straight line is taken. Some algebraic work shows that the remaining relative error is now:

$$E_{rel} = -(RGR \cdot DELT)^2/12 \qquad (5)$$

In Table 5 this result can be checked. Similarly it was derived that for METHOD RKSFX, the relative error is given by

$$E_{rel} = -(RGR \cdot DELT)^4/2880 \qquad (6)$$

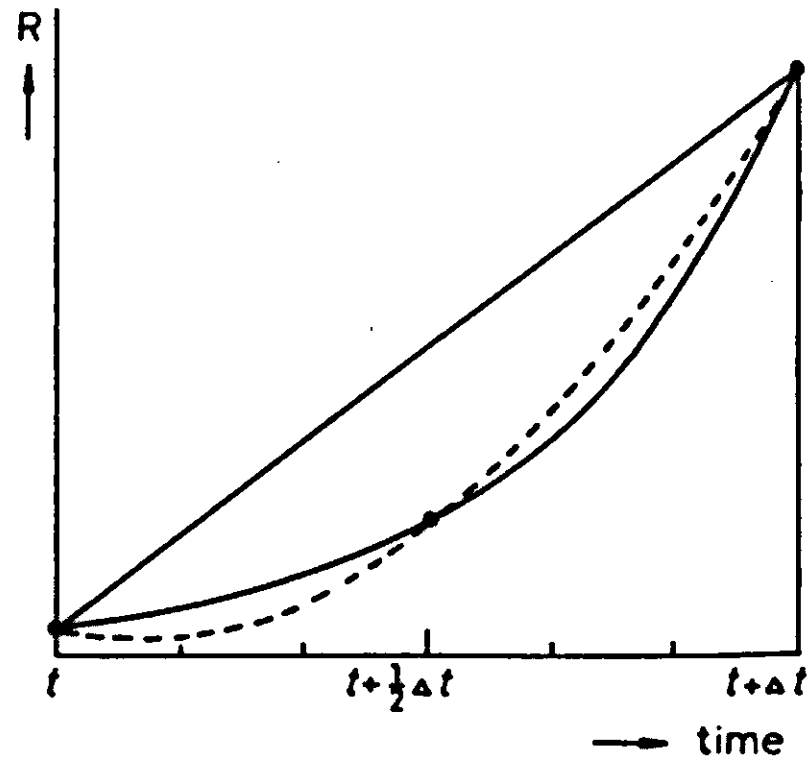

Figure 22. Geometrical representation of the trapezoidal (below straight line) and Runge-Kutta (below curved dashed line) integration methods. The curved solid line is the exponential which has to be integrated. The rate of change (R) is on the vertical axis, so that areas represent integrated values.

For DELT = 0.01 the resulting relative error is drowned in the truncation errors that always occur. The exponent of the dimensionless product (RGR • DELT) that occurs in the expression for the relative errors, reflects that the method RECT, TRAPZ and RKSFX are of the first, second and fourth order, respectively. This result has wider applicability than just this example of integration of the exponential function. However, there are exceptions where with some luck a lower-order integration method does just as well as a higher-order one.

Exercise 22

Repeat Exercise 21 for the integration of

a. R = TIME
b. R = TIME * *2
c. R = SIN(2 * PI * TIME)

between TIME 0. and TIME 0.5, and with PI = 3.141592

Integration with feedback

Influence of feedback on development of errors is considerable. With feedback (Subsection 2.1.6), either positive or negative, the situation is worse than discussed before, because errors will propagate. In exponential growth of a single plant, an underestimation of leaf area will cause an underestimation of photosynthesis, and hence growth, and hence leaf area. With numerical integration the same thing happens, and moreover a new error will be added every time interval. Therefore, in contrast to integration of a driving force, relative errors tend to grow during simulation time, although negative feedback may sometimes help us. The error analysis is slightly different from that for a driving force, because the relative error will refer to the total integral value, which consists of integrated amount and initial value together.

As an example we shall study the integration of the rate R = RGR • A. Ideally the result of this integration is the same as integration of the exponential function EXP(RGR • T), but by linking the rate to the integral value A itself, feedback is introduced. In the rectangular method the value of the integral A after one integration step will equal:

$$A_{T+DELT} = A_T \cdot (1 + RGR \cdot DELT) \qquad (7)$$

so that the rate at time T + DELT is given by

$$R_{T+DELT} = RGR \cdot A_T \cdot (1 + RGR \cdot DELT) \qquad (8)$$

In the trapezoidal method this rate and the one at time T are averaged and used to find a better estimate of A_{T+DELT}. The corrected estimate can be written as:

$$A_{T+DELT} = A_T \cdot (1 + RGR \cdot DELT + (RGR * DELT)^2/2) \qquad (9)$$

As before, we assume that the difference between the result for TRAPZ and RECT is a good estimate for the error involved in RECT. By comparing Equation 7 and Equation 9 we find that for one integration interval the relative error in RECT can be estimated as:

$$E_{rel} = (RGR \cdot DELT)^2/2 \tag{10}$$

In contrast to Equation 3, RGR • DELT occurs here in an quadratic form. This is so because in Equation 3 the error is related to the integral, not considering its initial value. In Equation 10, the initial value is included because with feedback one is interested in the total result. This relative error occurs each integration step, and in contrast to the situation without feedback, these errors accumulate. At time T, when T/DELT integration steps have been performed, the resulting relative error is

$$E_{rel} = RGR^2 \cdot T \cdot DELT/2 \tag{11}$$

Interestingly, the relative error is again proportional with DELT, both with and without feedback. A similar procedure for evaluation of the errors involved in TRAPZ and in RKSFX yields the result presented in Table 6. In this table RGR has been replaced by 1/TAU, where TAU stands for time coefficient. Time coefficient has a more general meaning than relative growth rate and it can be loosely defined as the shortest duration of time in which at least one of the integrals or driving forces of the model system can change considerably (cf. Subsections 1.4.4 and 2.1.7).

In principle the results of Table 6 are also valid when the feedback is negative, but it should be remembered that they refer to the difference between the integral and the equilibrium level. Since that difference decreases in time, the absolute error decreases as well. The relative error of the difference between equilibrium and integral increases all the time, but in most applications that is not disturbing. If we define relative error as absolute error divided by integral itself, negative feedback will help to dissipate old errors. Only in the extreme situation that the time interval exceeds the time coefficient is the dampening effect of the negative feedback insufficient to constrain the growth of the integration errors.

Integration with discontinuities

Discontinuities in forcing functions also need to be considered. The error analysis discussed before is not possible when a discontinuity occurs, because the derivatives do not exist at the breaking point. When an integration interval overlaps a discontinuity, the error in any integration method will be large. To estimate how large, we must first find out what type of discontinuity we are dealing with. In Figure 23 different types of discontinuities have been illustrated. The time course has been given of the content of an integral (state) and its corresponding rate of change is above it. The first discontinuity in this graph is characterized by a jump in the integral content (point t_0). Such a jump can only be realized if the rate gets an infinitely high value at this moment during an

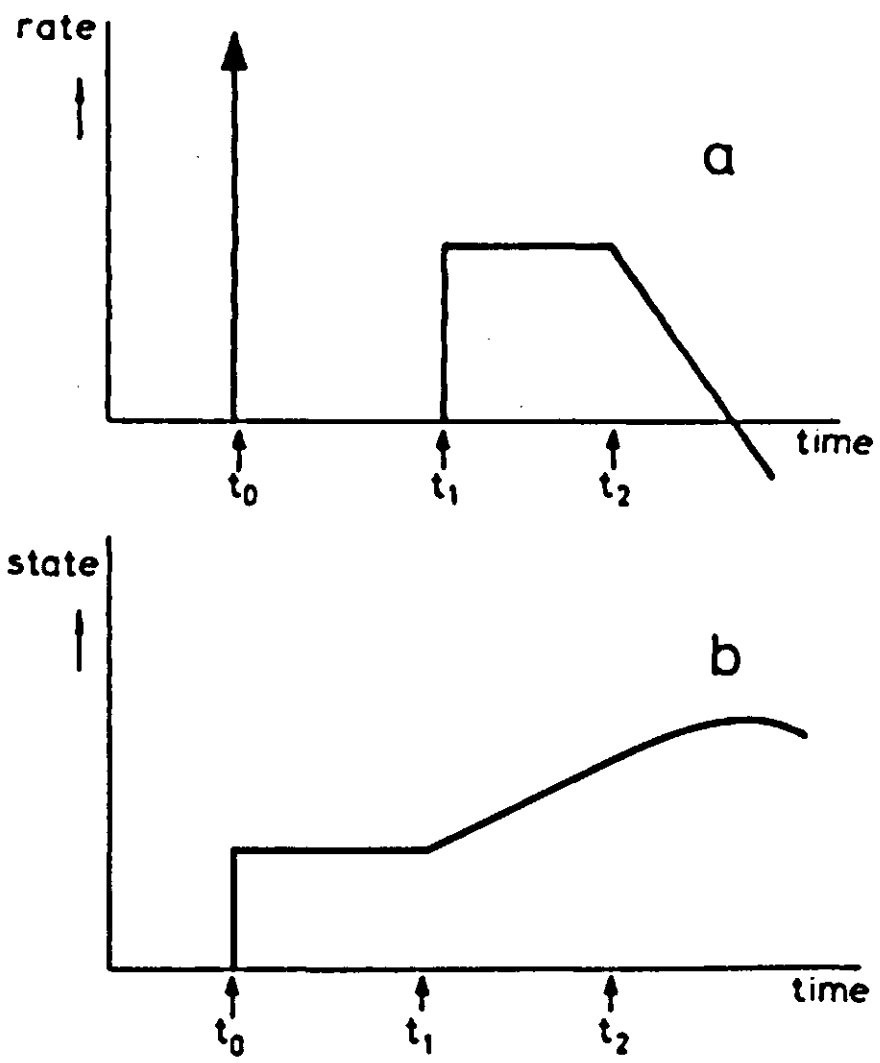

Figure 23. Discontinuities of zero-th, first and second order in a rate variable and their consequences for the state variable.

infinitely short duration of time, in such a way that the area of this pulse is equal to the jump of the integral. In physics such a pulse is called a Dirac pulse and often denoted by $\delta(t_0)$. Let us call this phenomenon a discontinuity of the zero-th order. In the second example, at time t_1 the rate shows a jump, resulting in a change of slope in the state. This is a discontinuity of the first order. The last example shows a discontinuity of the second order at time t_2; it is hardly noticeable in the graph of the state.

It is clear that these discontinuities have a decreasing order of difficulty for simulation. The most serious one, the discontinuity of the zero-th order, occurs when the content of an integral has to be changed instantaneously, e.g. when a crop is harvested. Then the only permissable integration method is METHOD RECT. The Dirac pulse of the rate is approximated by a pulse with width DELT and a height equal to the amount of change divided by the time interval. For instance, if an integral must be emptied we get

$$A_{T+DELT} = A_T - DELT \cdot (A_T/DELT) \qquad (12)$$

It is characteristic for these constructions that a division by DELT occurs somewhere in the program. It ensures that the numerical error because of the discontinuity itself has been corrected, and that the remaining errors can be analyzed in the traditional way.

Discontinuities of the first and second order may occur when weather data are available as daily totals or as daily averages. For instance, the average temperature may jump at midnight. Usually these discontinuities can be handled by a self-adapting method like RKS. Whatever method is used, the error will be proportional to the time interval around the discontinuity; and so it can be made as small as one wishes. In Table 7 a summary is given of the line of reasoning

Table 7. Decision table for integration method and time interval.

Do discontinuities of the zero-th order occur?
Yes: Choose METHOD RECT Synchronize DELT with the discontinuities if they appear at regular intervals. Choose DELT smaller than the smallest time coefficient, usually about 1/10 of it.
No: Is the time coefficient stable and known?
Yes: Choose METHOD RKSFX Choose DELT about ½ of the time coefficient.
No: Specify neither METHOD (RKS is default) nor DELT (is automatically adapted)

that must be followed to arrive at the proper integration method.

2.3.7 Computation schemes of the integration methods

From the scheme in Table 8 it is possible to find how many times the UPDATE is executed for each integration interval DELT. The calculation of the rate(s) R is represented by a call of the UPDATE (see Subsection 2.2.4): R = UPDATE (A_n,T_n) in which A represents the state variable and T time. In RECT this call is done once, in TRAPZ twice and in RKSFX four times. This larger expenditure is more than compensated by the much larger size of time intervals that can be taken to reach the same accuracy. For instance, with a required accuracy of 0.1% for each simulation period of one time coefficient we need 500 computations with RECT, 25 with TRAPZ and only seven with RKSFX. When we know the time coefficient, we can make an estimate of the needed time step beforehand. In complicated systems with many simultaneous processes such an estimate is difficult. It is possible to use an empirical method like running the model twice with different time steps. It is also possible to use the CSMP-provided integration method RKS that chooses the time step itself. In this method two integration routines are executed simultaneously, their results are compared, and when they differ too much the time step is halved. If the deviation is much smaller than required, DELT will be doubled for the next step.

Sometimes the preset error criterion is not met, and DELT is decreased below the minimum value DELMIN. Then the error message 'DELT IS LESS THAN DELMIN' is given, and the simulation is automatically terminated. Such an event usually means a programming or conceptual error, or at least an awkward model structure. Because of the feature of automatic adaption of the time-interval of integration, the method RKS is recommended as a standard method.

In the RKS method, the statements of the computer programs are executed many times, only to obtain a preliminary estimation of the rates. How many

Table 8. Summary of the integration methods RECT, TRAPZ, RKSFX. T stands for TIME.

RECTANGULAR

$R = UPDATE(A_n, T_n)$
$A_{n+1} = A_n + DELT \cdot R$
$T_{n+1} = T_n + DELT$

TRAPZ

$R1 = UPDATE(A_n, T_n)$
$A1 = A_n + DELT \cdot R1$
$T_{n+1} = T_n + DELT$
$R2 = UPDATE(A1, T_{n+1})$
$A_{n+1} = A_n + DELT \cdot (R1 + R2)/2$

RKSFX

$R1 = UPDATE(A_n, T_n)$
$A1 = A_n + DELT \cdot R1 \cdot 0.5$
$T_{n+½} = T_n + DELT \cdot 0.5$
$R2 = UPDATE(A1, T_{n+½})$
$A2 = A_n + DELT \cdot R2 \cdot 0.5$
$R3 = UPDATE(A2, T_{n+½})$
$A3 = A_n + DELT \cdot R3$
$T_{n+1} = T_n + DELT$
$R4 = UPDATE(A3, T_{n+1})$
$A_{n+1} = A_n + DELT \cdot (R1 + 2 \cdot R2 + 2 \cdot R3 + R4)/6$

---

times this execution is done can be checked by introduction of some counters into the program. To this end, an initial segment is introduced, in which the counters COUNT1 and COUNT2 are set to zero. At the very end of the DYNAMIC segment, that is evaluated each time interval, a section is introduced with the card NOSORT to indicate that the statements after this card cannot be sorted.

```
NOSORT
COUNT1 = COUNT1 + 1
COUNT2 = COUNT2 + KEEP
END
```

These statements cannot be sorted because the same variables occur to the left and to the right of the equal sign (see Subsection 2.3.2). Each time this statement is executed, COUNT1 is incremented by 1; and COUNT2 by KEEP. The variable KEEP is an internal CSMP variable and has the value 1 if the integration step is actually executed, and a value 0 if the statements are only executed for a preliminary evaluation. In this way both the number of time intervals and the number of calculations of the whole program can be kept track of.

Exercise 23
Use this NOSORT block with the program introduced in Exercise 21; change the NOSORT block into a PROCEDURE and repeat the calculations.
