STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING TOEGEPASTE WISKUNDE

TW 106

Numerical treatment of a two-component diffusion process in a slab.

by

G.J.R. Förch

December 1968.

The Mathematical Centre at Amsterdam, founded the 11th of February, 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for Pure Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

1. Introduction.

In this paper we consider the drying process of a homogeneous infinite slab of an absorbent (Malto-dextrin) which contains initially about 70% water and 1% acetone.

The aim is to calculate the concentrations of water and acetone, c and c_A, respectively, as functions of time t and the depth x.

For practical purposes we are interested in quantities that can be measured by physical experiments, such as the average concentrations of water and acetone, as functions of t.

In the initial state the water concentration c_0 and the acetone concentration c_{A_0} are assumed to be uniform throughout the slab. The thickness of the slab in the initial state will be 2l.

In this paper the subscript A will always indicate that the quantity mentioned is related with the acetone.

2. Statement of the problem.

In view of the apparent symmetry of the model with respect to the central plane of the slab, we will restrict our considerations to the left half of the slab and take x = 0 at the surface and x = l at the middle of the slab.

It will be assumed that the drying process is a diffusion process satisfying the following system of partial differential equations

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left(D(c)\,\frac{\partial c}{\partial x}\right), \tag{2.1}$$

$$\frac{\partial c_A}{\partial t} = \frac{\partial}{\partial x}\left(D_A(c)\,\frac{\partial c_A}{\partial x}\right), \tag{2.2}$$

where D and D_A are the diffusion coefficients of water and acetone, respectively, which are both functions of the waterconcentration c.

The initial conditions are

(2.3a) $\quad c(0,x) = c_0 \ ,$

(2.3b) $\quad c_A(0,x) = c_{A_0} \ ,$

and the boundary conditions are

(2.4a) $\quad \frac{\partial c}{\partial x} = 0 \qquad$ for $x = 1 \ ,$

(2.4b) $\quad \frac{\partial c_A}{\partial x} = 0 \qquad$ for $x = 1 \ ,$

(2.5a) $\quad D \frac{\partial c}{\partial x} = N \qquad$ for $x = 0 \ ,$

(2.5b) $\quad D_A \frac{\partial c_A}{\partial x} = N_A \qquad$ for $x = 0 \ ,$

where N and N_A are the mass-fluxes at the surface of water and acetone, respectively. The behaviour of D, D_A, N and N_A as functions of c has been determined empirically by L.C. Menting of the Unilever Research Laboratory Duiven.
His calculations resulted in the following functions.

(2.6) $\quad D = a_{11} \exp (-a_{12} / c) \ ,$

(2.7) $\quad D_A = a_{21} \exp (-a_{22} / \sqrt{c}) \ ,$

$$a_{11} = .018 \ , \ a_{12} = 2100 \ ,$$
$$a_{21} = 20 \ , \ a_{22} = 300 \ ;$$

of course all these quantities depend on the the unities of length and concentration we use.

$$(2.8)\qquad N = k\,\{.01888 - .02357 \exp(-.00678\, c(t,0)) - r\,[.0088 \exp .04545\,(T + \frac{2.446 * 10^6\, N}{h}) - .004]\}\ ,$$

$$(2.9)\qquad N_A = k_A c_A(t,0)\ ,$$

where k, k_A, r, h and T are parameters defining the drying circumstances.

3. Numerical approach to the problem.

The problem has been solved numerically on the EL.X8 computer of the Mathematical Centre, Amsterdam.

As equation (2.1) with initial condition (2.3a) and boundary conditions (2.4a) and (2.5a) is independent of the acetone concentration, we first investigate the problem for the waterconcentration alone.

The main difficulty is that $\frac{\partial c}{\partial x}$ is large in the neighbourhood of the boundary $x = 0$, inducing a still much larger spatial derivative for the diffusion coefficent D. This feature of the problem forces us to use a very dense grid in this region of the slab. In order to allow suitable time steps we choose the standard implicit difference-scheme for the numerical calculations. The problem being only one-dimensional, implicit methods have no serious disadvantage, compared to explicit methods, as far as computing time is concerned.

The differential equation (2.6) can be written in the form

$$(2.6a)\qquad \frac{\partial c}{\partial t} = D\,(\frac{\partial^2 c}{\partial x^2} + \frac{a_{12}}{c^2}\,(\frac{\partial c}{\partial x})^2)\ ,$$

from which the following difference analogue may be derived

$$(3.1)\qquad \frac{c_j^{n+1} - c_j^n}{dt} = D_j^n\,\{(\frac{c_{j+1}^{n+1} - 2c_j^{n+1} + c_{j-1}^{n+1}}{(dx)^2}) + \frac{a_{12}}{(c_j^n)^2}\,(\frac{c_{j+1}^n - c_{j-1}^n}{2dx})\,(\frac{c_{j+1}^{n+1} - c_{j-1}^{n+1}}{2dx})\}\ ,$$

where dt and dx are the time- and the space step of the grid and where the subscripts indicate the place in the spatial grid and the superscripts the place in the time grid.

Application of (3.1) gives a set of linear algebraic equations for the c_j^{n+1}, $j = 0,\ldots,N$, that can be solved by the double-sweep method.

The calculations have been preformed with a spatial step of length 1/100. It turned out that with this grid the accuracy at $x = 0$ was not sufficient as the waterconcentration showed a jump up to about 70% of the local value, over the first spatial step.

Therefore, a still much finer grid is required at $x = 0$. To save computing-time a non-uniform grid is used, constructed in the following manner.

The slab is divided into two regions. The first region (region I) contains the smaller values of x, where the waterconcentration varies rapidly. The second region (region II) contains the larger x-values, where the water-concentration has only small variations.

In region I we use a non-uniform grid. At $x = 0$ we take a sufficiently small steplength dx_0 ; the next steps dx_n are defined by $dx_n = \alpha dx_{n-1}$, $n = 1,2,\ldots k$, α and k being suitable numbers.

In region II we use a uniform grid with a steplength that equals approximately dx_k. The choice of the "growing parameter" α and the number k is rather arbitrary. α must be taken so that the growth of the steplengths approximately fits the growth of the diffusioncoefficient D. Both α and k were determined by experiment. It turned out that the choice of these parameters is not very critical.

In the region of the non-uniform grid the derivatives are approximated by

$$\frac{\partial c}{\partial x} \approx \left(-\frac{\alpha}{1+\alpha} c_{j-1} + \frac{\alpha-1}{\alpha} c_j + \frac{1}{\alpha(1+\alpha)} c_{j+1}\right) / dx ,$$

$$\frac{\partial^2 c}{\partial x^2} \approx \left(\frac{2}{1+\alpha} c_{j-1} - \frac{2c_j}{\alpha} + \frac{2}{\alpha(1+\alpha)} c_{j+1}\right) / (dx)^2 .$$

Here the lefthand step is taken to be dx and the righthand step is αdx. Both approximations are of second-order accuracy.

In this way we obtain in region I the following difference analogue for equation (2.1)

$$(3.2)\qquad \frac{c_j^{n+1}-c_j^n}{dt} = D_j^n \{\frac{2}{\alpha(1+\alpha)}\, c_{j+1}^{n+1} - \frac{2c_j^{n+1}}{\alpha} + \frac{2c_{j-1}^{n+1}}{1+\alpha} + \frac{a_{12}}{(c_j^n)^2}\,(\frac{c_{j+1}^n}{\alpha(\alpha+1)} + \frac{\alpha-1}{\alpha}\, c_j^n - \frac{\alpha}{1+\alpha}\, c_{j-1}^n)\,(\frac{c_{j+1}^{n+1}}{\alpha(\alpha+1)} + \frac{\alpha-1}{\alpha}\, c_j^{n+1} - \frac{\alpha}{\alpha+1}\, c_{j-1}^{n+1})\} / (dx)^2 .$$

In region II difference scheme (3.1) is used.

For reasons of completeness we shall now investigate whether the schemes (3.1) and (3.2) are convenient for application of the double-sweep method. This method operates as follows.

The difference schemes are of the form

$$(3.3)\qquad l1_j^{n+1}\, c_{j-1}^{n+1} + l2_j^{n+1}\, c_j^{n+1} + l3_j^{n+1}\, c_{j+1}^{n+1} + l4_j^{n+1} = 0 .$$

Assume that

$$(3.4)\qquad c_j^{n+1} = m0_j^{n+1}\, c_{j+1}^{n+1} + m1_j^{n+1} .$$

Combining (3.3) and (3.4) yields

$$(3.5)\qquad m0_j^{n+1} = -\frac{l3_j^{n+1}}{l2_j^{n+1} + l1_j^{n+1}\, m0_{j-1}^{n+1}} ,$$

and

$$(3.6)\qquad m1_j^{n+1} = -\frac{l4_j^{n+1} + l1_j^{n+1}\, m1_{j-1}^{n+1}}{l2_j^{n+1} + l1_j^{n+1}\, m0_{j-1}^{n+1}} .$$

The boundary conditions make it possible to solve the c_j^{n+1} , $j = 0,1,2,\ldots N$, by first establishing all values for the $m0_j^{n+1}$ and $m1_j^{n+1}$ and afterwards substituting these values into (3.4).

This method is stable if

$$(3.7) \qquad l1_j^{n+1} < 0 \ , \ l2_j^{n+1} > 0 \ , \ l3_j^{n+1} < 0 \ ,$$

and

$$(3.8) \qquad -l1_j^{n+1} - l3_j^{n+1} < l2_j^{n+1} \ , \quad \text{for all } j \ .$$

It is easily seen that for difference scheme (3.1) as well as for difference scheme (3.2) condition (3.8) is always fulfilled.
Condition (3.7) gives rise to the following stability criterions.
For differencescheme (3.1)

$$(3.9) \qquad \left| \frac{a_{12}}{(c_j^n)^2} \left(\frac{c_{j+1}^n - c_{j-1}^n}{4}\right) \right| < 1 \ ,$$

and for differencescheme (3.2)

$$(3.10) \qquad -2 < \frac{a_{12}}{(c_j^n)^2} \left(\frac{c_{j+1}^n}{\alpha(\alpha+1)} + \frac{\alpha-1}{\alpha} c_j^n - \frac{\alpha}{1+\alpha} c_{j-1}^n\right) < \frac{2}{\alpha} \ .$$

In actual computations a check on these stability criterions has been incorporated.

To obtain a convenient boundary condition at $x = 0$ it is necessary to compute N from the implicit equation (2.8), in which c_0^n is substituted for $c(t,0)$.
This transcendental equation is solved by Newtons method.
Boundary condition (2.5) provides the startingvalues $m0_0^{n+1}$ and $m1_0^{n+1}$ necessary to calculate all other $m0_j^{n+1}$ and $m1_j^{n+1}$ from formulas (3.5) and (3.6). Boundary condition (2.4) provides the startingvalue c_N^{n+1} necessary to calculate all other c_j^{n+1} from formula (3.4).

The methods described allow us to compute the waterconcentration as a function of t and x.

The acetone equation (2.2) is taken into account by a completely analogeous approach.
Now the non-uniform grid is prescribed by the expected variation of the diffusioncoefficient of acetone, since this coefficient varies more rapidly than the diffusioncoefficient of water.

The difference analogue of the boundary condition at x = 0 for acetone, turns out to be quite simple

$$(3.4) \qquad D_{A_0}^{n} \left(\frac{c_{A_1}^{n+1} - c_{A_0}^{n+1}}{dx_0} \right) = k_A \, c_{A_0}^{n+1} \, .$$

The stability criterions that have to be fulfilled for correct application of the double-sweep method turns out to be

$$(3.12) \qquad \left| \frac{a_{22}}{2c_j^n \sqrt{c_j^n}} \left(\frac{c_{j+1}^n - c_{j-1}^n}{4} \right) \right| < 1 \, ,$$

for the uniform grid region and

$$(3.13) \qquad -2 < \frac{a_{22}}{2c_j^n \sqrt{c_j^n}} \left(\frac{c_{j+1}^n}{\alpha(1+\alpha)} + \frac{\alpha-1}{\alpha} c_j^n - \frac{\alpha}{1+\alpha} c_{j-1}^n \right) < \frac{2}{\alpha} \, ,$$

for the non-uniform grid region.

In figure 1 some results of the computations are given.

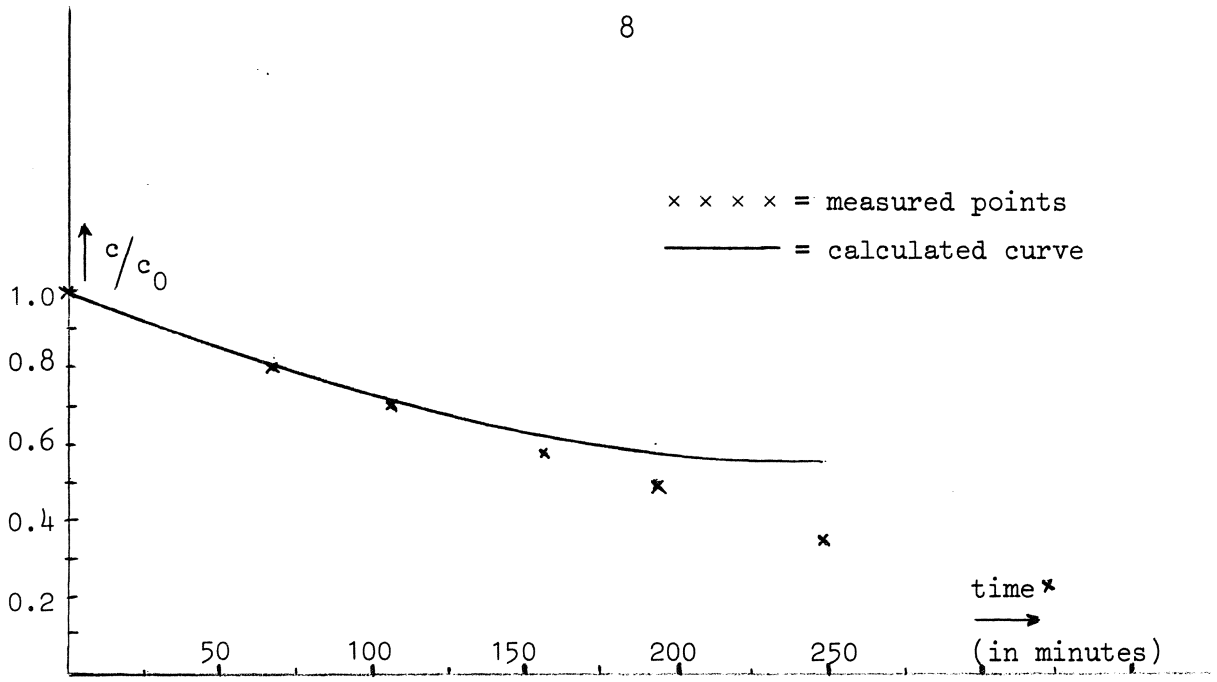

fig. 1

4. The shrinking effect.

The results obtained by the computational methods described above did not agree very well with the empirical results. A reason for this discrepancy is that in reality the slab is shrinking during the drying process which causes higher concentrations. Therefore the shrinkingeffect will be taken into account.

We shall assume the shrinking to be one-dimensional, i.e. only the thickness of the slab reduces during the drying process. To deal with the reducing thickness we replace x by a new coordinate x^*. As new unity of length the amount of slabthickness containing a fixed quantity of absorbent is chosen. We take this standard quantity to be equal to the amount of absorbent that is contained in 1 cm slabthickness per cm^2 surface at the start of the drying process.
Clearly this change of coordinate not only affects the x, but also the concentrations C and C_A and the diffusioncoefficients D and D_A.

The relations between the new functions and the original values are

$$(4.1) \qquad dx^{*} = \frac{1-c}{1-c_0}\, dx \ ,$$

$$(4.2) \qquad c^{*} = \frac{1-c_0}{1-c}\, c \ ,$$

$$(4.3) \qquad c_A^{*} = \frac{1-c_0}{1-c}\, c_A \ ,$$

$$(4.4) \qquad D^{*} = \left(\frac{1-c_0}{1-c}\right)^2 D \ ,$$

$$(4.5) \qquad D_A^{*} = \left(\frac{1-c_0}{1-c}\right)^2 D_A \ .$$

When we express the old values as functions of the new ones we get

$$(4.1a) \qquad dx = (1 - c_0 + c^{*})\, dx^{*} \ ,$$

$$(4.2a) \qquad c = c^{*} / (1 - c_0 + c^{*}) \ ,$$

$$(4.3a) \qquad c_A = c_A^{*} / (1 - c_0 + c^{*}) \ ,$$

$$(4.4a) \qquad D^{*} = a_{11} \exp\left(-a_{12} \frac{1-c_0+c^{*}}{c^{*}}\right) / (1 - c_0 + c^{*})^2 ,$$

$$(4.5a) \qquad D_A^{*} = a_{21} \exp\left(-a_{22}\sqrt{\frac{1-c_0+c^{*}}{c^{*}}}\right) / (1 - c_0 + c^{*})^2 .$$

In this formulas the effect of the changing acetone concentration on the slabthickness is neglected.

I

It is not difficult to show (see [1] chapter XI) that the form of the diffusion differential equations (2.1) and (2.2) can be maintained in the new coordinate system.

So we can use as differential equations

$$(2.1a) \qquad \frac{\partial c^{*}}{\partial t} = \frac{\partial}{\partial x^{*}} \left(D^{*} \frac{\partial c^{*}}{\partial x^{*}}\right) \ ,$$

and

$$\frac{\partial c_A^*}{\partial t} = \frac{\partial}{\partial x^*}\left(D_A^* \frac{\partial c_A^*}{\partial x^*}\right) . \qquad (2.2a)$$

Of course, the boundary conditions have to be transformed correctly into the new coordinate system, but no essential difficulty is encountered here.

After computing the concentrations c^* and c_A^* as funtions of t and x^* the original values c and c_A have to be calculated from (4.1), (4.2) and (4.3).

In fig. 2 the results of the calculations are shown.
At the end of the paper ALGOL 60 program with which the calculations were preformed is given.

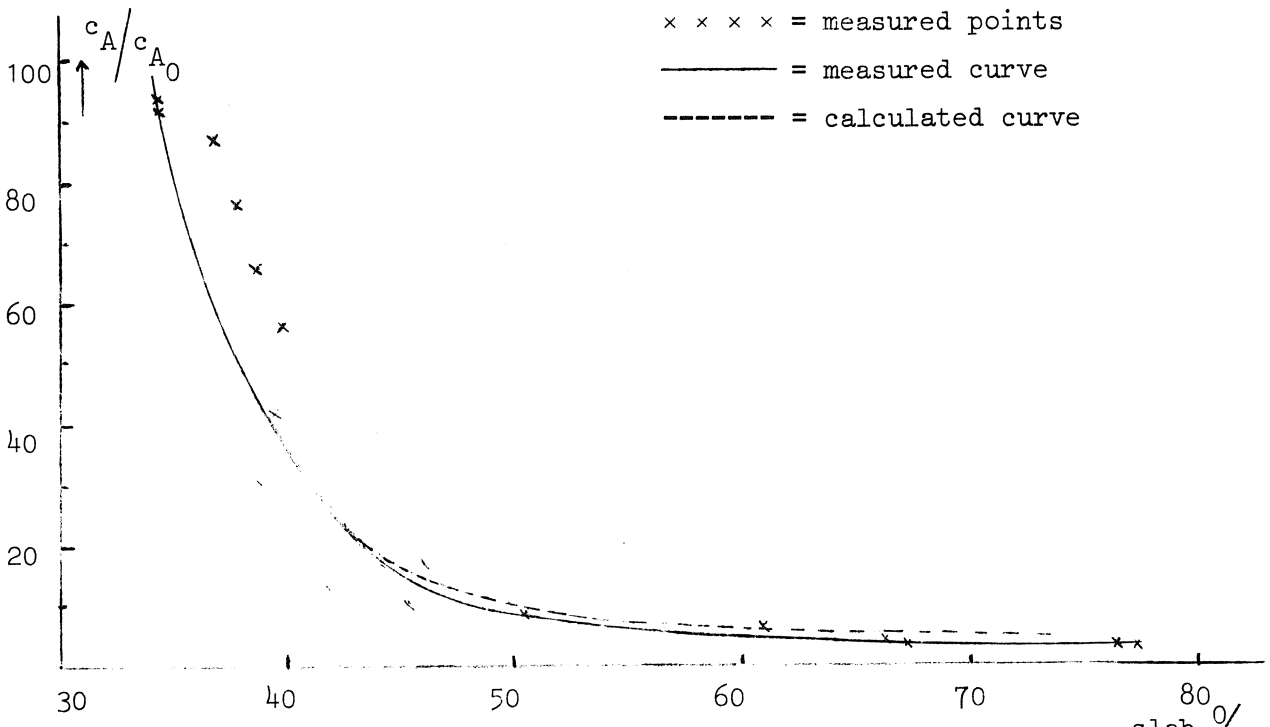

fig. 2 Dry matter content of the slab

5. An alternative approach.

It is also possible to deal with the large coefficients of the differential equations by introducing a coordinate transformation.

We shall sketch this method for the water equation alone. First we introduce dimensionless variables in such a way that the boundary conditions are imposed at $x = 0$ and $x = 1$, and that the diffusion coefficient D becomes the form:

$$D = e^{-\frac{2}{c}} .$$

In this dimensionless equation we introduce the following coordinate transformation.

$$(5.1) \qquad \begin{cases} \xi = x\sqrt{\dfrac{D(1,t)}{D(x,t)}} , \\ \tau = \displaystyle\int_0^t D(1,t)\, dt . \end{cases}$$

This transformation maps the interval $0 \leq x \leq 1$ on $0 \leq \xi \leq 1$.

When we substitute these new variables in the boundary value problem (2.3a), (2.4a), (2.5a) we yield

$$(5.2) \qquad \frac{\partial c}{\partial \tau} = \frac{1}{T}\frac{\partial^2 c}{\partial \xi^2} + \xi \frac{\partial c}{\partial \xi}\left[\frac{\partial c_1^{-1}}{\partial \tau} + \frac{1}{T^2}\left(\left(\frac{\partial c^{-1}}{\partial \xi}\right)^2 + \left(\frac{\partial^2 c^{-1}}{\partial \xi^2}\right)\right)\right] ,$$

with boundary conditions

$$(5.3) \qquad \sqrt{DD_1}\,\frac{\partial c}{\partial \xi} = N \qquad \text{at } \xi = 0 ,$$

$$(5.4) \qquad \frac{\partial c}{\partial \xi} = 0 \qquad \text{at } \xi = 1 .$$

Here $c_1 = c(1,\tau)$, $D_1 = D(1,\tau)$, $T = 1 - \xi \dfrac{\partial c^{-1}}{\partial \xi}$.

We observe that $\frac{\partial c^{-1}}{\partial \xi}$ is negative and, therefore, the coefficients of equation (5.2) are not large.

Though the differential equation has been transformed to a suitable form for numerical treatment, one encounters in this method two basic difficulties.

1. The first one is caused by the fact that the steepness of the concentrationcurve near the boundary $x = 0$ ($\xi = 0$) is not only due to the large coefficients in the differential equation, but is also influenced by the boundary condition at $x = 0$ ($\xi = 0$) ; when the drying process proceeds, this influence increases and in the later stage of the process the boundary condition even overshadows the differential equation.
Now in condition (5.3) we still have a factor $\sqrt{D}$; thus the boundary condition still causes a very large spatial gradient for the concentration.

2. The second difficulty arises when we take the acetone equation into account. Then we have to preform transformation (5.1) with the diffusion coefficient that changes most rapidly with changing waterconcentration. This will be D_A. In this way we can transform the acetone equation quite easily, but now the water equation becomes very complicated.

As transformation (5.1) complicates the calculations and the difficulties mentioned make it very difficult to perform the numerical calculations straight-forward, this approach was rejected.

```
begin comment Opdrachtnr. 8169 , codenr. GJRF240268/VIIa. Impliciet
  schema op ongetransformeerde coordinaten , met verlopend rooster.
  Water en aceton beide . Correcties voor het krimpen van de slab
  aangebracht. Nieuwe diffusiecoeff. en nieuwe randvoorwaarde. ;
  integer K;
  K:= READ;
  begin integer n, n1, n2, K1, K2, I, I1, I2, i, j, J, itereer,
    itereer1, s, p1, p2, p3, p4;
    real alfa, alfa1, beta, beta1, c0, c1, c2, cv1, ca0, ca1, ca2,
    cva1, cv2, cva2, cK, cKa, cK1, cmin, cmax, r, l, tijd, plaats,
    a11, a12, a21, a22, D, DA, d1, dx1, dt, dt1, dt2, dt3, dtmin,
    dtmax, dtmax1, dx, d2, d3, d0, lambda, draaitijd, tijdmax,
    flux0, flux1, fluxa0, fluxa1, integraal0, integraal1,
    integraala0, integraala1, lambda1, l1, l2, l3, la1, la2, la3,
    m10, m11, m20, m21, ma10, ma11, ma20, ma21, flux, fluxa,
    integraal, integraala, eps1, eps2, eps3, fluxmaat, s1, s2, s3,
    s4, sa1, sa2, sa3, d4, d5, lengte, ka, kw, h, gamma, delta,
    tijdmaat, tijdmaat1, tijdmaat2, tijdmaatmin, tijdmaatmax, int1,
    int2, inta1, inta2;
    array stap, c, cv, cvv, ca, cva, cvva, m1, m2, ma1, ma2[0:K];

    procedure LT(r, s); real r; string s;
    begin r:= READ; NLCR; PRINTTEXT(s); FLOT(5, 1, r); NLCR end;


    procedure Input;
    begin PRINTTEXT(≮ resultaten gjrf240268/viia ≯); NLCR; NLCR;
      RUNOUT; PUTEXT(
≮ Resultaten GJRF240268/VIIa. De waarden van de flux en de integraal.≯
      ); PUNLCR; PUNLCR; PRINTTEXT(≮ k = ≯); PRINT(K); NLCR; LT(K1,
      ≮ k1 = ≯); LT(I1, ≮ i1 = ≯); LT(I2, ≮ i2 = ≯); LT(p3,
      ≮ p3 = ≯); LT(p4, ≮ p4 = ≯); LT(alfa, ≮ alfa = ≯);
      LT(itereer1, ≮ aantal iteraties = ≯); LT(l, ≮ l = ≯); LT(r,
      ≮ r = ≯); LT(lambda1, ≮ lambda1 = ≯); LT(cK,
      ≮ beginconc.water = ≯); LT(cKa, ≮ beginconc.aceton = ≯);
      LT(a11, ≮ a11 = ≯); LT(a12, ≮ a12 = ≯); LT(a21, ≮ a21 = ≯);
      LT(a22, ≮ a22 = ≯); LT(eps1, ≮ eps1 = ≯); LT(eps2,
      ≮ eps2 = ≯); LT(eps3, ≮ eps3 = ≯); LT(dt, ≮ dt = ≯); LT(dt1,
      ≮ dt1 = ≯); LT(dtmax, ≮ dtmax = ≯); LT(dtmax1,
      ≮ dtmax1 = ≯); LT(dtmin, ≮ dtmin = ≯); LT(dx, ≮ dx = ≯);
      LT(ka, ≮ ka = ≯); LT(kw, ≮ kw = ≯); LT(h, ≮ h = ≯);
      LT(tijdmax, ≮ tijdmax = ≯); LT(J, ≮ J = ≯); LT(gamma,
      ≮ gamma = ≯); if J < 0 then
      for n:= 0 step 1 until K do
      begin c[n]:= cv[n]:= cvv[n]:= cK;
        ca[n]:= cva[n]:= cvva[n]:= cKa
      end;
      if J > 0 then
      for n:= 0 step 1 until K do
      begin c[n]:= cv[n]:= cvv[n]:= READ;
        ca[n]:= cva[n]:= cvva[n]:= READ
      end
    end Input ;
```

```
procedure Output;
begin RUNOUT; PUNLCR; PUNCH(K2); PUNCH(K1); PUNCH(3); PUNCH(I2);
   PUNCH(p3); PUNCH(p4); FLOP(5, 1, alfa); PUNCH(itereer1);
   FLOP(5, 1, l); FLOP(5, 1, r); FLOP(5, 1, lambda1);
   FLOP(5, 1, cK); FLOP(5, 1, cKa); FLOP(5, 1, a11);
   FLOP(5, 1, a12); FLOP(5, 1, a21);
   FLOP(5, 1, a22 × sqrt(1000)); FLOP(5, 1, eps1);
   FLOP(5, 1, eps2); FLOP(5, 1, eps3); FLOP(5, 1, dt);
   FLOP(5, 1, dt1); FLOP(5, 1, dtmax); FLOP(5, 1, dtmax1);
   FLOP(5, 1, dtmin); FLOP(5, 1, dx); FLOP(5, 1, ka);
   FLOP(5, 1, kw); FLOP(5, 1, h); FLOP(5, 1, tijdmax); PUNCH(1);
   FLOP(5, 1, gamma); RUNOUT;
   for n:= 0 step 1 until K2 do
   begin FLOP(13, 3, c[n]); FLOP(13, 3, ca[n]) end;
   RUNOUT; FLOP(5, 1, draaitijd); PUNLCR;
   FLOP(13, 3, tijdmaatmin); FLOP(13, 3, tijdmaatmax); PUNLCR
end Output ;


real procedure f(c); value c; real c;
begin integer n;
   real k1, k2, k3, k4, k5, k6, k7, f1, f2, N1, N2;
   k1:= kw × .01888; k2:= - kw × .02357; k3:= - .00678;
   k4:= - kw × r × .0088; k5:= .04545 × 21.5;
   k6:= 4545 × 24.45 / h; k7:= .004 × kw × r; N1:= 0; n:= 0;
B: n:= n + 1; f2:= k4 × k6 × exp(k5 + k6 × N1) - 1;
   f1:= k1 + k2 × exp(k3 × c) + k4 × exp(k5 + k6 × N1) + k7 - N1;
   N2:= N1 - f1 / f2;
   if abs(N2 - N1) > 10 - 3 × abs(N2) + 10 ∧ n < 40 then
   begin N1:= N2; goto B end;
   if n = 40 then PRINTTEXT(‹ convergeert niet ›); f:= N2
end;


procedure Beginrooster;
begin plaats:= 0; d3:= dx / alfa;
   for n:= 1 step 1 until K1 do
   begin d3:= d3 × alfa; stap[n]:= d3; plaats:= plaats + d3 end;
   K2:= entier((1 - plaats) / d3); dx1:= (1 - plaats) / K2;
   lambda:= dt / (dx1 × dx1); alfa1:= alfa + 1; beta:= dx1 / d3;
   beta1:= beta + 1; K2:= K2 + K1;
   for n:= K1 + 1 step 1 until K2 do stap[n]:= dx1; NLCR;
   PRINTTEXT(‹ aantal netpunten =›); PRINT(K2); NLCR; PRINTTEXT(
   ‹ dx1 = ›); FLOT(5, 1, dx1); NLCR
end Beginrooster ;


procedure Scheme4;
begin tijd:= tijd + dt; NLCR; PRINTTEXT(‹ tijd = ›);
   FLOT(5, 1, tijd); NLCR; PUNLCR; PUTEXT(‹ tijd = ›);
   FLOP(5, 1, tijd); PUNLCR;
ITER: c1:= c[0]; c2:= c[1]; cK1:= 1 - cK + c1;
   D:= a11 × exp( - a12 × (cK1 / c1) ↑ gamma) / (cK1 × cK1);
   ca1:= ca[0]; ca2:= ca[1];
```

```
DA:= a21 × exp( - a22 / sqrt(c1 / cK1)) / (cK1 × cK1);
d2:= dx; flux:= f(10 3 × c1 / cK1) / D; if p1 < 0 then
begin m10:= 1; m20:= - d2 × flux end
else
begin m10:= 0; m20:= cmin; PRINTTEXT(‹ randcorrectie›); NLCR
end;
if p4 < 0 then
begin ma10:= 1 / (1 + dx × ka / (DA × cK1)); ma20:= 0 end
else
begin ma10:= 0; ma20:= 0 end;
m1[0]:= m10; m2[0]:= m20; ma1[0]:= ma10; ma2[0]:= ma20;
itereer:= itereer - 1; PRINTTEXT(‹ iteratie no.›);
PRINT(itereer1 - itereer); NLCR; comment Hierna BLANK ;
for n:= 1 step 1 until K1 - 1 do
begin c0:= c1; c1:= c2; c2:= c[n + 1]; ca0:= ca1; ca1:= ca2;
  ca2:= ca[n + 1]; d1:= d2; d2:= d1 × alfa; cK1:= 1 - cK + c1;
  delta:= (cK1 / c1) ↑ gamma;
  D:= a11 × exp( - a12 × delta) / (cK1 × cK1);
  DA:= a21 × exp( - a22 / sqrt(c1 / cK1)) / (cK1 × cK1);
  s1:= (a12 × gamma × delta × (1 - cK) / (c1 × cK1) - 2 /
  cK1) × ( - alfa / alfa1 × c0 + c2 / (alfa × alfa1) + (alfa
  - 1) / alfa × c1); s3:= dt × D / (d1 × d1);
  sa3:= dt × DA / (d1 × d1);
  sa1:= (.5 × a22 × (1 - cK) / (c1 × sqrt(c1 × cK1)) - 2 /
  cK1) × ( - alfa / alfa1 × c0 + c2 / (alfa × alfa1) + (alfa
  - 1) / alfa × c1); l1:= s3 × (2 - s1 × alfa) / alfa1;
  if l1 < 0 then
  begin PRINTTEXT(‹ verraad water ›); PRINT(n); NLCR end;
  l2:= - 1 - s3 × (2 - s1 × (alfa - 1)) / alfa;
  l3:= s3 × (2 + s1) / (alfa × alfa1);
  la1:= sa3 × (2 - sa1 × alfa) / alfa1; if la1 < 0 then
  begin PRINTTEXT(‹ verraad aceton ›); PRINT(n); NLCR end;
  la2:= - 1 - sa3 × (2 - sa1 × (alfa - 1)) / alfa;
  la3:= sa3 × (2 + sa1) / (alfa × alfa1);
  s2:= 1 / (l1 × m10 + l2); sa2:= 1 / (la1 × ma10 + la2);
  m11:= - l3 × s2; m21:= - (cv[n] + l1 × m20) × s2;
  m1[n]:= m10:= m11; m2[n]:= m20:= m21; ma11:= - la3 × sa2;
  ma21:= - (cva[n] + la1 × ma20) × sa2; ma1[n]:= ma10:= ma11;
  ma2[n]:= ma20:= ma21
end;
comment Hierna BLANK ;
c0:= c1; c1:= c2; c2:= c[K1 + 1]; ca0:= ca1; ca1:= ca2;
ca2:= ca[K1 + 1]; d1:= d2; d2:= dx1; cK1:= 1 - cK + c1;
delta:= (cK1 / c1) ↑ gamma;
D:= a11 × exp( - a12 × delta) / (cK1 × cK1);
DA:= a21 × exp( - a22 / sqrt(c1 / cK1)) / (cK1 × cK1);
s1:= (a12 × gamma × delta × (1 - cK) / (c1 × cK1) - 2 / cK1)
× ( - beta / beta1 × c0 + c2 / (beta × beta1) + (beta - 1) /
beta × c1); s3:= dt × D / (d1 × d1);
sa3:= dt × DA / (d1 × d1);
sa1:= (.5 × a22 × (1 - cK) / (c1 × sqrt(c1 × cK1)) - 2 / cK1)
× ( - beta / beta1 × c0 + c2 / (beta × beta1) + (beta - 1) /
beta × c1); l1:= s3 × (2 - s1 × beta) / beta1;
if l1 < 0 then
begin PRINTTEXT(‹ verraad water ›); PRINT(K1); NLCR end;
```

```
l2:= - 1 - s3 × (2 - s1 × (beta - 1)) / beta;
l3:= s3 × (2 + s1) / (beta × beta1);
la1:= sa3 × (2 - sa1 × beta) / beta1; if la1 < 0 then
begin PRINTTEXT(‹ verraad aceton ›); PRINT(K1); NLCR end;
la2:= - 1 - sa3 × (2 - sa1 × (beta - 1)) / beta;
la3:= sa3 × (2 + sa1) / (beta × beta1);
s2:= 1 / (l1 × m10 + l2); sa2:= 1 / (la1 × ma10 + la2);
m11:= - l3 × s2; m21:= - (cv[K1] + l1 × m20) × s2;
m1[K1]:= m10:= m11; m2[K1]:= m20:= m21; ma11:= - la3 × sa2;
ma21:= - (cva[K1] + la1 × ma20) × sa2; ma1[K1]:= ma10:= ma11;
ma2[K1]:= ma20:= ma21; comment Hierna BLANK ;
for n:= K1 + 1 step 1 until K2 - 1 do
begin c0:= c1; c1:= c2; c2:= c[n + 1]; cK1:= 1 - cK + c1;
  ca0:= ca1; ca1:= ca2; ca2:= ca[n + 1];
  delta:= (cK1 / c1) ↑ gamma;
  D:= a11 × exp( - a12 × delta) / (cK1 × cK1);
  DA:= a21 × exp( - a22 / sqrt(c1 / cK1)) / (cK1 × cK1);
  s1:= .25 × (a12 × gamma × delta × (1 - cK) / (c1 × cK1) - 2
  / cK1) × (c2 - c0);
  sa1:= .25 × (.5 × a22 × (1 - cK) / (c1 × sqrt(c1 × cK1)) -
  2 / cK1) × (c2 - c0); s3:= lambda × D; sa3:= lambda × DA;
  l1:= s3 × (1 - s1); if l1 < 0 then
  begin PRINTTEXT(‹ verraad water.›); PRINT(n); NLCR end;
  la2:= - 1 - 2 × sa3; la3:= sa3 × (1 + sa1);
  l2:= - 1 - 2 × s3; l3:= s3 × (1 + s1);
  la1:= sa3 × (1 - sa1); if la1 < 0 then
  begin PRINTTEXT(‹ verraad aceton ›); PRINT(n); NLCR end;
  s2:= 1 / (l1 × m10 + l2); sa2:= 1 / (la1 × ma10 + la2);
  m11:= - l3 × s2; m21:= - (cv[n] + l1 × m20) × s2;
  m1[n]:= m10:= m11; m2[n]:= m20:= m21; ma11:= - la3 × sa2;
  ma21:= - (cva[n] + la1 × ma20) × sa2; ma1[n]:= ma10:= ma11;
  ma2[n]:= ma20:= ma21
end;
comment Hierna BLANK ;
c2:= c1:= m21 / (1 - m11); c[K2]:= c[K2 - 1]:= c1;
ca2:= ca1:= ma21 / (1 - ma11); ca[K2]:= ca[K2 - 1]:= ca1;
plaats:= 1 - dx1; FLOT(5, 1, 1); FLOT(5, 1, c2);
FLOT(5, 1, ca2); SPACE(3); FLOT(5, 1, plaats); FLOT(5, 1, c1);
FLOT(5, 1, ca1); SPACE(3); integraal:= 0; integraala:= 0;
int2:= 2 × c1 × dx1; inta2:= 2 × ca1 × dx1;
if itereer > 0 then
begin for n:= K2 - 2 step - 1 until 1 do
  begin c2:= c1; c1:= m1[n] × c2 + m2[n]; c[n]:= c1; ca2:= ca1;
    ca1:= ma1[n] × ca2 + ma2[n]; ca[n]:= ca1;
    d0:= stap[n + 1]; plaats:= plaats - d0; int1:= int2;
    int2:= (c1 + c2) × d0; inta1:= inta2;
    inta2:= (ca1 + ca2) × d0;
    if ((K2 - n) : 2) × 2 = K2 - n then
    begin integraal:= integraal + (int1 + int2 + 2 × d0 × c2)
      / 3;
      integraala:= integraala + (inta1 + inta2 + 2 × d0 ×
      ca2) / 3
    end;
    FLOT(5, 1, plaats); FLOT(5, 1, c1); FLOT(5, 1, ca1);
    SPACE(3); if (n : 4) × 4 = n then NLCR
```

```
  end;
  n:= n - 1; c0:= m1[0] × c1 + m2[0];
  ca0:= ma1[0] × ca1 + ma2[0]; cv1:= cv[0];
  if c0 ≤ cmin + eps1 + abs(cv1 - c0) then
  begin c0:= cmin; p1:= 1 end;
  c[0]:= c0; ca[0]:= ca0; FLOT(5, 1, 0); FLOT(5, 1, c0);
  FLOT(5, 1, ca0); NLCR; int1:= int2; int2:= (c1 + c0) × dx;
  inta1:= inta2; inta2:= (ca1 + ca0) × dx;
  if ((K2 - n) : 2) × 2 = K2 - n then
  begin integraal:= integraal + (int1 + int2 + 2 × dx × c1) /
    3;
    integraala:= integraala + (inta1 + inta2 + 2 × dx × ca1)
    / 3
  end
  else
  begin integraal:= integraal + .5 × int2;
    integraala:= integraala + .5 × inta2
  end;
  PRINTTEXT(‹ integraal water = ›); FLOT(5, 1, integraal);
  NLCR; PRINTTEXT(‹ integraal aceton = ›);
  FLOT(5, 1, integraala); NLCR; FLOP(5, 1, integraal);
  PUSPACE(5); FLOP(5, 1, integraala); PUNLCR; goto ITER;
  comment Na volgende else BLANK ;
end
else
begin cv2:= cvv[K2 - 1]; cv1:= cv[K2 - 1]; cvv[K2 - 1]:= cv1;
  cv[K2 - 1]:= c1; cva[K2 - 1]:= ca1; if s < 0 then
  begin tijdmaat:= 2 × abs(cv2 / (dt2 × dt1 + dt2 × dt2) - cv1
    / (dt2 × dt1) + c1 / (dt2 × dt1 + dt1 × dt1));
    tijdmaat2:= tijdmaat
  end;
  for n:= K2 - 2 step - 1 until 1 do
  begin c2:= c1; c1:= m1[n] × c2 + m2[n]; cv2:= cvv[n];
    cv1:= cv[n]; cvv[n]:= cv1; c[n]:= cv[n]:= c1;
    if s < 0 then
    begin tijdmaat:= 2 × abs(cv2 / (dt2 × dt1 + dt2 × dt2) -
      cv1 / (dt2 × dt1) + c1 / (dt2 × dt1 + dt1 × dt1));
      if tijdmaat2 < tijdmaat then tijdmaat2:= tijdmaat
    end;
    ca2:= ca1; ca1:= ma1[n] × ca2 + ma2[n];
    ca[n]:= cva[n]:= ca1; d0:= stap[n + 1];
    plaats:= plaats - d0; FLOT(5, 1, plaats); FLOT(5, 1, c1);
    FLOT(5, 1, ca1); SPACE(3); if (n : 4) × 4 = n then NLCR;
    int1:= int2; int2:= (c2 + c1) × d0; inta1:= inta2;
    inta2:= (ca1 + ca2) × d0;
    if ((K2 - n) : 2) × 2 = K2 - n then
    begin integraal:= integraal + (int1 + int2 + 2 × d0 × c2)
      / 3;
      integraala:= integraala + (inta1 + inta2 + 2 × d0 ×
      ca2) / 3
    end;
  end;
  comment Hierna BLANK ;
  c0:= m1[0] × c1 + m2[0]; ca0:= ma1[0] × ca1 + ma2[0];
  cv1:= cv[0]; if c0 ≤ cmin + eps1 + abs(cv1 - c0) then
```

```
      begin c0:= cmin; p1:= 1; j:= j + 1 end;
      FLOT(5, 1, 0); FLOT(5, 1, c0); FLOT(5, 1, ca0); NLCR;
      c[0]:= c0; cv2:= cvv[0]; cv1:= cv[0]; cv[0]:= c0;
      cvv[0]:= cv1; ca[0]:= ca0; cva[0]:= ca0; if s < 0 then
      begin tijdmaat:= 2 × abs(cv2 / (dt2 × dt1 + dt2 × dt2) - cv1
        / (dt2 × dt1) + c0 / (dt2 × dt1 + dt1 × dt1));
        if tijdmaat2 < tijdmaat then tijdmaat2:= tijdmaat
      end;
      n:= n - 1; integraal0:= integraal1;
      integraala0:= integraala1; flux0:= flux1; fluxa0:= fluxa1;
      int1:= int2; int2:= (c0 + c1) × dx; inta1:= inta2;
      inta2:= (ca0 + ca1) × dx;
      if ((K2 - n) : 2) × 2 = K2 - n then
      begin integraal1:= integraal + (int1 + int2 + 2 × dx × c1)
        / 3;
        integraala1:= integraala + (inta1 + inta2 + 2 × dx × ca1)
        / 3
      end
      else
      begin integraal1:= integraal + .5 × int2;
        integraala1:= integraala + .5 × inta2
      end;
      PRINTTEXT(‹ gemiddelde waarde water = ›);
      FLOT(5, 1, integraal1 / 1); NLCR; PRINTTEXT(
      ‹ gemiddelde waarde aceton = ›);
      FLOT(5, 1, integraala1 / 1); NLCR;
      FLOP(5, 1, integraal1 / 1); PUSPACE(5);
      FLOP(5, 1, integraala1 / 1); PUNLCR; cK1:= 1 - cK + c0;
      flux1:= 10 - 3 × a11 × exp( - a12 × (cK1 / c0) ↑ gamma) ×
      (c1 - c0) / (dx × cK1 × cK1);
      fluxa1:= 10 - 3 × a21 × exp( - a22 / sqrt(c0 / cK1)) × (ca1
      - ca0) / (dx × cK1 × cK1); PRINTTEXT(‹ flux water =›);
      FLOT(5, 1, flux1); NLCR; PRINTTEXT(‹ flux aceton = ›);
      FLOT(5, 1, fluxa1); NLCR; PRINTTEXT(‹ controle water ›);
      SPACE(5); FLOT(5, 1, .5 × dt × (flux0 + flux1)); SPACE(3);
      FLOT(5, 1, integraal0 - integraal1); NLCR; PRINTTEXT(
      ‹ controle aceton ›); SPACE(5);
      FLOT(5, 1, .5 × dt × (fluxa0 + fluxa1)); SPACE(3);
      FLOT(5, 1, integraala0 - integraala1); NLCR;
      FLOP(5, 1, flux1); PUSPACE(5); FLOP(5, 1, fluxa1); PUNLCR
    end
end Scheme4 ;

Input; a22:= a22 / sqrt(1000); Beginrooster; tijd:= 0; s:= 1;
p1:= - 1; LT(draaitijd, ‹ draaitijd = ›);
flux1:= f(103 × cK) × 10 - 3; fluxa1:= 10 - 3 × ka × cKa;
integraal1:= 1 × cK; integraala1:= 1 × cKa;
cmin:= - 10 - 1 × ln((.1888 + r × .04 - r × .088 × exp(.4545 ×
2.15)) / .2357) / .678; PRINTTEXT(‹ cmin = ›); FLOT(5, 1, cmin);
NLCR; j:= 0; cmin:= cmin × (1 - cK) / (1 - cmin); PRINTTEXT(
‹ cmin gecorrigeerd = ›); FLOT(5, 1, cmin); itereer:= itereer1;
Scheme4; fluxmaat:= fluxa1;
for i:= 1 step 1 until I1 do
begin itereer:= itereer1; Scheme4 end;
tijdmaat:= tijdmaat1:= abs(cv2 + c0 - 2 × cv1) / (dt × dt);
```

```
    dt2:= dt; dt:= dt1; NLCR; if J < 0 then
    begin tijdmaatmin:= tijdmaat × dt2 / dtmax;
       tijdmaatmax:= tijdmaat × dt2 / dtmin
    end
    else
    begin tijdmaatmin:= READ; tijdmaatmax:= READ end;
    PRINTTEXT(≮ tijdmaat1 = ≯); FLOT(5, 1, tijdmaat1); NLCR; s:= - 1;
    itereer:= itereer1; Scheme4; dt2:= dt; itereer:= itereer1;
    Scheme4; if p3 < 0 then
    begin NLCR; PRINTTEXT(≮ zelfzoekende tijdstap begonnen.≯); NLCR;
       for i:= 1 step 1 until I2 do
       begin dt3:= dt × lambda1 × tijdmaat1 / tijdmaat2;
         if j = 10 then
         begin tijdmaatmin:= tijdmaatmin × dtmax / dtmax1;
           dtmax:= dtmax1
         end;
         NLCR; PRINTTEXT(≮ tijdmaat = ≯); FLOT(5, 1, tijdmaat2); NLCR;
         if dt3 > dtmax then dt:= dtmax else
         begin if dt3 < dtmin then
           begin NLCR; PRINTTEXT(≮ gevaar ≯); NLCR; dt:= dtmin end
           else dt:= dt3
         end;
         lambda:= lambda × dt / dt1; dt2:= dt1; dt1:= dt;
         if tijdmaat2 < tijdmaatmax ∧ tijdmaatmin < tijdmaat2 then
         tijdmaat1:= tijdmaat2;
         if tijdmaat2 ≤ tijdmaatmin then tijdmaat1:= tijdmaatmin;
         if tijdmaatmax ≤ tijdmaat2 then tijdmaat1:= tijdmaatmax;
         itereer:= itereer1; Scheme4; if tijdmax ≤ tijd then
         begin Output; EXIT end;
         if fluxa1 ≤ eps3 × fluxmaat then
         begin Output; EXIT end;
         if time > draaitijd then
         begin Output; EXIT end;
       end
    end
    else
    for i:= 1 step 1 until I2 do
    begin itereer:= itereer1; Scheme4; if tijdmax ≤ tijd then
       begin Output; EXIT end;
       if fluxa1 ≤ eps3 × fluxmaat then
       begin Output; EXIT end;
       if time > draaitijd then
       begin Output; EXIT end;
     end
  end
end
```