stichting
mathematisch
centrum

MC

AFDELING TOEGEPASTE WISKUNDE TN 57/70 AUGUSTUS

P.J. VAN DER HOUWEN
C.G. VAN DER LAAN

NUMERICAL SOLUTION OF THE DIFFUSION EQUATION BY ONE STEP METHODS

2e boerhaavestraat 49 amsterdam

Contents

1. Introduction

In this note we consider some numerical examples solved by explicit one-step methods as given in [1], One step methods for linear initial value problems I by P.J. van der Houwen. These examples are inital value problems for a system of ordinary differential equations of the type

$$\frac{d\tilde{U}}{dt} = D\tilde{U} + F, \tag{1.1}$$

where $\tilde{U}$ and F are (vector) functions of the variable t, and D is a matrix with constant entries. We shall denote functions of the continuous variable t by capitals and the corresponding discretized functions, i.e. the functions which arise from restructing t to a discrete set of points, by the corresponding lower cases.

The aim is to obtain the relation between the computation time and the accuracy obtained. The computation time c needed to perform the calculations is measured by the number of evaluations K of the right hand side of equation (1.1).
The difference scheme is of the form

$$u_{k+1} = u_k + \tau c_k^{(1)} + \frac{1}{2!}\tau^2 c_k^{(2)} + \ldots + \frac{1}{p!}\tau^p c_k^{(p)} + \beta_{p+1}\tau^{p+1} c_k^{(p+1)} + \ldots + \beta_n \tau^n c_k^{(n)}, \quad k = 0, 1, 2, \ldots, \tag{1.2}$$

where

$$c_k^{(0)} = u_k$$

and

$$c_k^{(i+1)} = Dc_k^{(i)} + \frac{d^i}{dt^i} f_k, \quad i = 0, 1, 2, \ldots, p-1.$$

Further, u_k denotes the difference solution at $t = t_k = k\tau$ and τ the step along the t-axis.

The accuracy ε is defined by

$$\varepsilon = \max_{k=1,2,\ldots,K} ||\tilde{U}_k - u_k||_2, \tag{1.3}$$

where $\tilde{U}_k$ denotes the analytical solution at $t = t_k = k\tau$ and $||\ ||_2$ the euclidean norm.
Scheme (1.2) will be applied to the diffusion equation.

2. Reduction of the diffusion equation to a set of ordinary differential equations.

Consider the following initial boundary value problem for the diffusion equation:

$$\begin{cases} U_t(x,t) = U_{xx}(x,t) + G(x,t), & 0 \le x \le 1,\ 0 \le t \le T, \\ U(x,t) = U(x,0), & 0 \le x \le 1,\ t = 0, \\ U(0,t) = \Phi(0,t), & 0 \le t \le T, \\ U(1,t) = \Phi(1,t), & 0 \le t \le T, \end{cases} \tag{2.1}$$

where $G(x,t)$, $U(x,0)$ and $\Phi(x,t)$ are given functions and $[0,T]$ the integration interval along the t-axis.

First we discretize the partial differential equation with respect to the x variable to obtain a set of linear equations. Here a three point approximation will be used for the second derivative U_{xx} (see [5] ch. 6). Then problem (2.1) reduces to the initial value problem

$$\begin{cases} \dfrac{d\tilde{U}}{dt} = D\tilde{U} + F, \\ \tilde{U} = \begin{pmatrix} U(h,0) \\ U(2h,0) \\ \vdots \\ U(N-2)h,0) \\ U(N-1)h,0) \end{pmatrix} \quad \text{for } t = 0, \end{cases} \tag{2.2}$$

where

$$\tilde{U} = \begin{bmatrix} U_1(t) \\ U_2(t) \\ \cdot\ \cdot\ \cdot \\ U_{N-2}(t) \\ U_{N-1}(t) \end{bmatrix},$$

$$D = h^{-2} \begin{bmatrix} -2 & 1 & 0 & & \cdot & \cdot & \cdot & 0 \\ 1 & -2 & 1 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & -2 & 1 & 0 & & & \cdot \\ \cdot & 0 & 1 & -2 & 1 & 0 & & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & & & 0 & 1 & -2 & 1 \\ 0 & \cdot & \cdot & \cdot & & 0 & 1 & -2 \end{bmatrix}$$

and

$$F = \begin{bmatrix} G_1(t) & + h^{-2}\, \Phi(0,t) \\ G_2(t) & + 0 \\ \cdot\ \cdot\ \cdot\ \cdot & \\ G_{N-2}(t) & + 0 \\ G_{N-1}(t) & + h^{-2}\, \Phi(1,t) \end{bmatrix}.$$

In these expressions h denotes the mesh-widtg along the x-axis, $U_j(t) = U(jh,t)$ and $G_j(t) = G(jh,t)$. Note that D is a symmetrix matrix.

3. Numerical solution of the initial value problem

The problem has been solved numerically on the EL X8 computer of the Mathematical Centre, Amsterdam, by applying scheme (1.2).

It was shown in [1] that scheme (1.2) has an accuracy of order p and and is stable when

$$\tau \leq \beta(n) \,/\, \sigma(D),$$

where $\sigma(D)$ is the spectral radius of D and $\beta(n)$ is determined by the generating polynomial.

$$P_n(\tau D) = A_p(\tau D) + (\tau D)^{p+1} B_q(\tau D),\ n = p+q+1,$$

with

$$A_p(\tau D) = 1 + \tau D + \frac{1}{2!}(\tau D)^2 + \ldots + \frac{1}{p!}(\tau D)^p$$

and

$$B_q(\tau D) = \beta_{p+1} + \beta_{p+2}\,\tau D + \ldots + \beta_n (\tau D)^q.$$

Here we employ the first order exact polynomial $P_n(x) = T_n(1 + \frac{x}{n^2})$ which is appropriate in cases where D is a symmtric matrix (more general in cases where D has distinct real eigenvalues).

The coefficients β_j of this generating polynomial are given by

$$\beta_j = T_n^{(j)}(1) \,/\, (n^{2j} j!),$$

where

$$T_n^{(j)}(1) = \frac{d^j}{dx^j} T_n(x)\Big|_{x=1}$$

and $\beta(n)$ is given by

$$\beta(n) = 2n^2.$$

At the end of this note the values of β_j are listed for some values of j and n in 12 decimals exact.

We distinguish two cases:

1) Fixed timesteps.

The time step τ equals the maximal allowed step, i.e.

$$\tau = \beta(n) \,/\, \sigma(D) = \tau_{stab}\ .$$

2) Controlled timesteps.

The timesteps τ_k are chosen at each level in such a way that the euclidean norm of the local discretization error due to the x-discretization equals the euclidean norm of the discretization error due to time discretization. To be more specific, we put

$$||\tau_k h^2 \frac{\partial^4}{\partial x^4} U_k||_2 = ||(\frac{1}{(p+1)!} - \beta_{p+1}) \tau_k^{(p+1)} c_k^{(p+1)}||_2.$$

This yields

$$\tau_k = \tau_{acc} = h^2/(12((p+1)! - \beta_{p+1})) *$$

$$* ||\frac{\partial^4 U_k}{\partial x^4}||_2 / ||c_k^{(p+1)}||_2 .$$

We shall use the approximation

$$\frac{\partial^4 U_k}{\partial x^4}|_{x=nh} \tilde{=}$$

$$\tilde{=} h^{-4}(u_k((n+2)h)-4u_k((n+1)h)+6u_k(nh)-4u_k((n-1)h)+u_k((n-2)h)).$$

In our calculations we have used only ± 10% of the available values of $\frac{\partial^4 U_k}{\partial x^4}|_{x=nh}$ in order to save computation time. In addition the stability condition, mentioned above, is imposed on the scheme.

4. Results

The problems we have chosen satisfy the boundary conditions:
$U(0,t) = U(1,t) = 1$.
For a survey of the results we give the following table.

Table 1

Survey of results

U(x,t)	U(x,0)	method			results	
		n	$\tau=\tau_{stab}$	$\tau=\tau_{acc}$	diagram	efficiency
$e^{-t}(x-x^{10})+1$	$1+x-x^{10}$	1	x		1(+)	1
"	"	4	x		1(x)	4
"	"	10	x		1(*)	8
"	"	10		x	1(*)	8
$e^{-t}(x-x^{4})+1$	$1+x-x^{4}$	10	x		2(+)	1
		10		x	2(x)	.9
$(1+t)^{-10}(x-x^{4})+1$	$1+x-x^{4}$	10	x		3(+)	1
		10		x	3(x)	5

The diagrams are given at the end of this note. The numbers in the last column indicate the relative efficiency of the method.

5. Concluding remarks

As shown in table 1 the polynomial method is most efficient when:

a) n is large

b) τ equals the maximum allowed timestep i.e. $\tau = \tau_{stab}$, in cases where the analytical solution is smoother on the t dependence than on x dependence (figure 2).

c) τ equals τ_{acc} in cases where the analytical solution is smoother on the x dependence than on t dependence (figure 3).

6. The ALGOL program

Before we deal with the structure of the program we give a list of the parameters which must be specified as input on a paper tape for the program:

inx	length of interval along the x-axis,
int	" " " " " t-axis.
first	number of meshpoints along x-axis to be used in first calculation.
step	increase in number of meshpoints.
last	number of meshpoints along x-axis to be used in last calculation.
I	equals n.
beta	$\beta(n)$
beta 1 . . . beta n	coefficients of the polynomial
-1	to stop the program.

We now describe the functions of the procedures declared in the program:

INPUT	calculates meshwidth h and τ_{stab}. activates ANSOL for calculation of initial values of u.
STEPSIZE	calculates tauacc and activates the formal procedures A, G.
NORM	calculates the euclidean norm of the vector.
Uxxx	calculates $c_k^{(2)}$ and activates CORTERM.
ANSOL	calculates the analytical solution in the meshpoint on the level t.
CORTERM	activates D, IDIT and calculates the vector $c_k^{(i)}$.
IDIT	calculates the i-th derivative of the inhomogeneous term f_k.

D	calculates the vector $D\mu_k$.
OUTPUT	calculates the relative accuracy eps and delivers the values of u_k and eps.
Pr nlcr	activates Pr string, NLCR and punches the code for carriage return.
Pr string	prints and punches the string parameter.

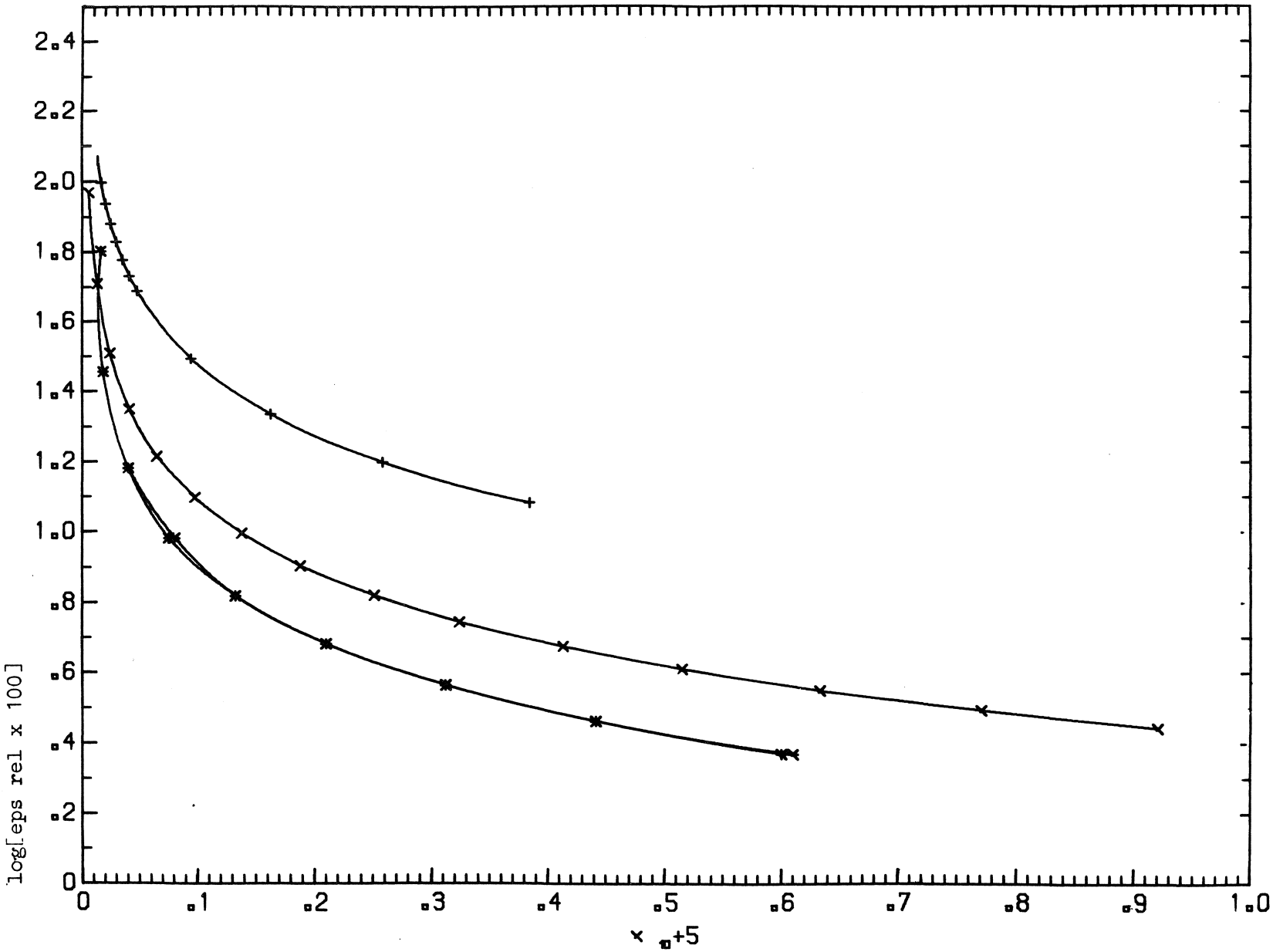
Furthermore, we used the library procedures (e.g. the Input and Output-procedures) as mentioned in [6].

We conclude this section with the complete version of the program in the case of variable timesteps. It will be clear that this program may be used for the same problem with fixed timesteps after some small changes in "cycle" and the procedure "INPUT".

```
begin comment cgl, 2007 diffusian equation with controlled timestep,
  270470, Ut= Uxx + exp(-t) × (x↑4 + 12 × x ↑ 2 - x), U(0,t)=
  U(1,t)= 1, U(x,0)= 1 + x - x↑4, opl: U= 1 + exp(-t) × (x - x↑4);
  integer k, kmax, n, N, Nmin1, Nmin2, i, I, j, first,epst, l, per, J;
  real tau, taustab, h, eps, epsmax, x, t, int, inx, h2de, beta, b2,
  I2de, tauconv, expon, taumax, taumin;

  procedure INPUT(inx, int, U); real inx, int; array U;
  begin h:= inx / N; h2de:= h × h; taustab:= h2de × beta / 4; NLCR;
      PRINTTEXT('N='); ABSFIXT(3, 0, N); TAB; PRINTTEXT('h=');
      FLOT(3, 1, h); TAB; PRINTTEXT('degree='); ABSFIXT(2, 0, I); TAB;
      PRINTTEXT('taustab='); FLOT(10, 1, taustab); NLCR; PRINTTEXT(
'      k tauconv        tau              field in a number of points...
  epsmax ×'); NLCR; t:= epsmax:= taumax:= 0; taumin:= int; k:= 0;
      expon:= 1; ANSOL(U)
  end;

  real

  procedure STEPSIZE(A, C, a, b, c); procedure A, C; array a, b, c;
  begin A(a, b); C(b, c);
      tauconv:= h2de × NORM(a, l) / (NORM(b, l) × abs(b2 × 12 - 6));
      STEPSIZE:= if tauconv ≤ taustab then tauconv else taustab
  end;

  real

  procedure NORM(a)length: (l); array a; integer l;
  begin real s, aj;
      s:= 0;
      for j:= l step - 1 until 1 do
      begin aj:= a[j]; s:= aj × aj + s end;
      NORM:= sqrt(s)
  end eucludian norm ;


  procedure Uxxxx(a, b); array a, b;
  begin integer n;
      real h4dem1;
      h4dem1:= 1 / (h2de × h2de);
      for j:= 1 step - 1 until 1 do
      begin n:= j × 10 - 5;
         a[j]:= (b[n + 2] - b[n + 1] × 4 + b[n] × 6 - b[n - 1] × 4
         + b[n - 2]) × h4dem1
      end
  end;


  procedure C2k(a, b); array a, b;
  begin for J:= 1 step 1 until 2 do CORRECTIETERM(J, a, b);
      for j:= 1 step 1 until 1 do a[j]:= a[j × 10 - 5]
  end;
```

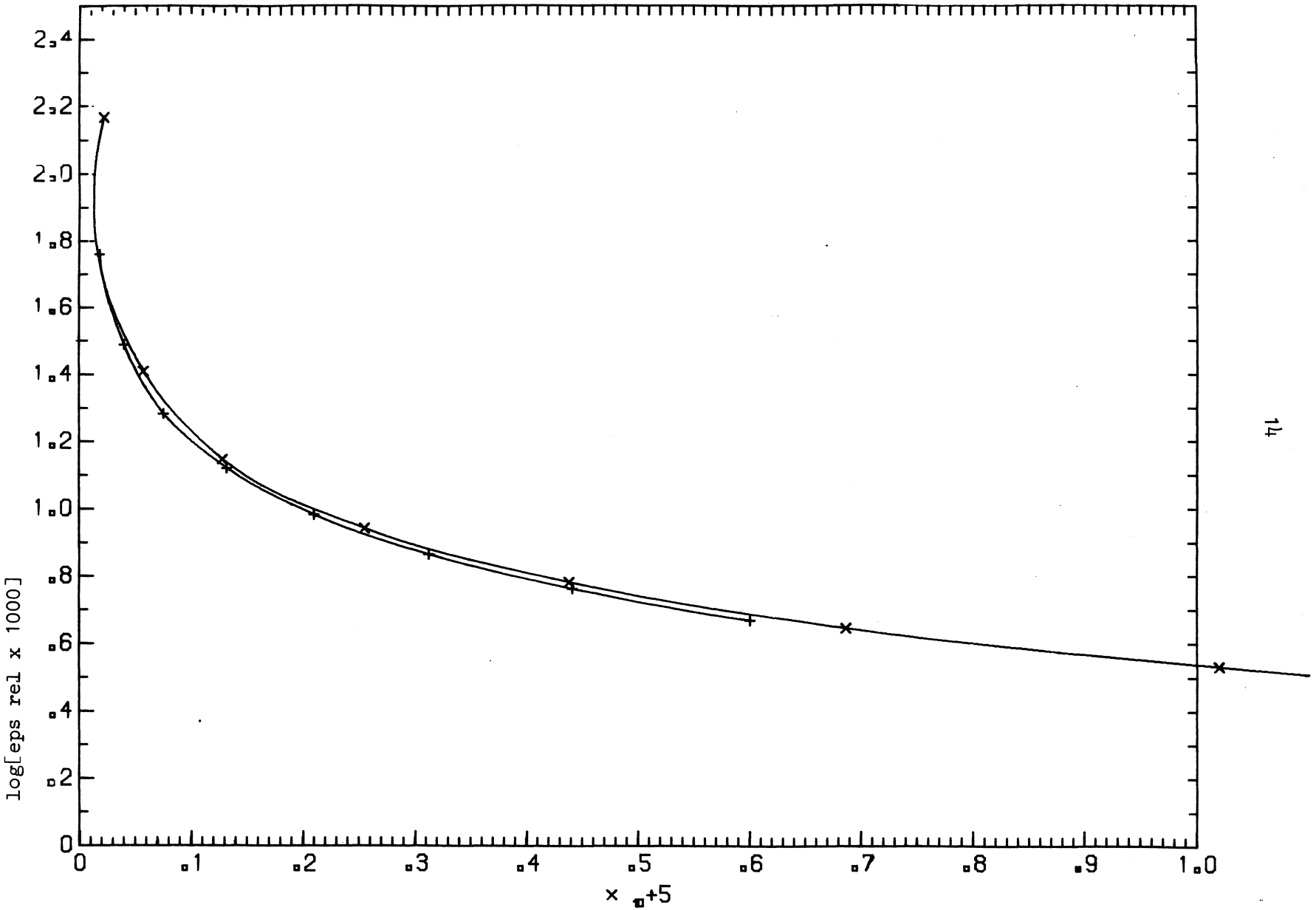
```
procedure ANSOL(a); array a;
for j:= Nmin 1 step - 1 until 1 do
begin x:= j × h; a[j]:= ( - x ↑ 3 + 1) × x × expon + 1 end;


procedure CORRECTIETERM(i, a, b); integer i; array a, b;
begin D(a); IDIT(i - 1, b);
    for j:= Nmin 1 step - 1 until 1 do a[j]:= a[j] + b[j];
end;


procedure IDIT(i)with results in array: (a); integer i; array a;
begin integer sgn;

    real procedure F(x); value x; real x;
    F:= expon × ((x × x + 12) × x - 1) × x;

    if i = 0 then
    begin a[1]:= F(h) + 1 / h2de;
       a[Nmin1]:= F(Nmin1 × h) + 1 / h2de;
       for j:= Nmin 2 step - 1 until 2 do a[j]:= F(j × h)
    end
    else
    begin sgn:= EVEN(i);
       for j:= Nmin 1 step - 1 until 1 do a[j]:= sgn × F(j × h);
    end
end IDIT= Ith time Derivative of Inhomogeneous Term;


procedure D(G); array G;
begin real a, b;
    a:= - 2 × G[1] + G[2];
    for j:= 2 step 1 until Nmin 2 do
    begin b:= G[j - 1] - G[j] × 2 + G[j + 1]; G[j - 1]:= a / h2de;
       a:= b
    end;
    G[Nmin1]:= (G[Nmin2] - G[Nmin1] × 2) / h2de;
    G[Nmin2]:= b / h2de
end;


procedure OUTPUT(A, B, C); array A, B, C;
begin ABSFIXT(6, 0, k); FLOT(10, 1, tauconv); FLOT(10, 1, tau);
    for j:= per step per until Nmin1 do FLOT(3, 1, A[j]); ANSOL(B);
    for j:= Nmin1 step - 1 until 1 do C[j]:= A[j] - B[j];
    eps:= NORM(C, Nmin1) / NORM(B, Nmin1) × 100; FLOT(3, 1, eps);
    if eps > epsmax then
    begin kmax:= k; epsmax:= eps; PRSYM(66) end;
    NLCR; if eps > 1000 then goto ready
end;


procedure PR nlcr; PR string(↓
      ↑);


procedure PR string(s); string s;
begin PRINTTEXT(s); PUTEXT(s) end;
```

```
  PR string(
≮ 'results diffusian equation with inhomogeneous term and controlled timestep

        270470, Ut= Uxx + exp(-t) × (x↑4 + 12 × x ↑ 2 - x), U(0,t)= U(1,t)= 1,
  U(x,0)= 1 + x - x↑4, opl: U= 1 + exp(-t) × (x - x↑4);
'≯
  ); inx:= READ; int:= READ; first:= READ; step:= READ;
  last:= READ;
 new order: I:= READ; if I < 0 then goto ready; beta:= READ;
  begin array B[1:I];
      for i:= 1 step 1 until I do B[i]:= READ; PUNLCR; PUTEXT(
≮ ' N    h                    k     kmax     taumin:          taumax:
 Epsmax:         comptime         degree=≯); ABSFIXP(2, 0, I); PUTEXT(≮ '≯); PUNLCR;
      for N:= first step step until last do
      begin Nmin1:= N - 1; Nmin2:= N - 2;
         per:= if N > 10 then N : 10 + 1 else 1;
         l:= if N > 10 then entier(N / 10) else 1;
         begin array U[0:N], AUxxxx[1:l], C, AIDIT, ARANSOL, AC2k,
            ANSOL min U[1:Nmin1];
            INPUT(inx, int, U); b2:= B[2];
            comment begin cycle;
         next level: for j:= Nmin 1 step - 1 until 1 do
            AC2k[j]:= C[j]:= U[j];
            tau:= STEPSIZE(Uxxxx, C2k, AUxxxx, AC2k, AIDIT);
            if tau > taumax then taumax:= tau;
            if tau < taumin then taumin:= tau;
            for i:= 1 step 1 until I do
            begin real bi, coeff;
               CORRECTIETERM(i, C, AIDIT); bi:= B[i];
               if k = 0 ∧ N = first then
               begin NLCR; FLOT(13, 2, bi); if i = I then
                  begin PRINTTEXT(≮=B(i),i=1,2,3...,I≯); NLCR; NLCR
                  end
               end;
               coeff:= tau ↑ i × bi;
               for j:= Nmin 1 step - 1 until 1 do U[j]:= U[j] + coeff
               × C[j]
            end PI(tauD) Uk;
            t:= t + tau; k:= k + 1; expon:= exp( - t);
            OUTPUT(U, ARANSOL, ANSOLmin U);
            if t < int then goto next level;
            comment end cycle , begin compact results;
            PUNLCR; if N = first then
            begin PUNLCR; PUTEXT(≮ number of mesurepoints=≯);
               ABSFIXP(3, 0, (last - first) : step + 1); PUNLCR;
               PUNLCR
            end;
            ABSFIXP(3, 0, N); PUSYM(118); FLOP(3, 1, h); PUSYM(118);
            ABSFIXP(3, 0, k); PUSYM(118); ABSFIXP(3, 0, kmax);
            PUSYM(118); FLOP(3, 1, taumin); PUSYM(118);
            FLOP(3, 1, taumax); PUSYM(118); FLOP(3, 1, epsmax);
            PUSYM(118); FLOP(3, 1, k × I × N);
         end;
         NEWPAGE
      end;
      goto new order
  end;
ready:
end
```



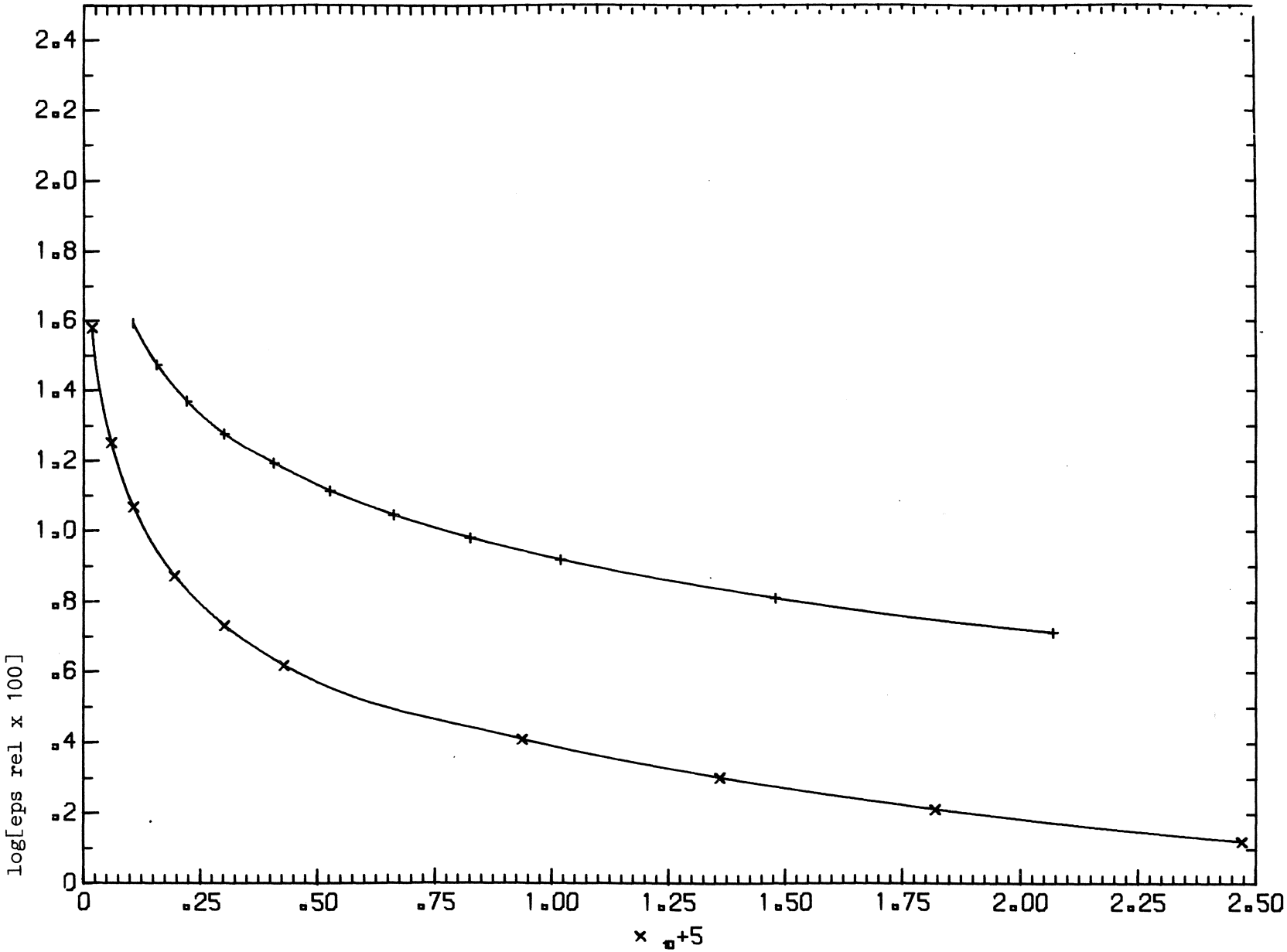
. comp. time

figure 1



comp. time

figure 2



comp. time

figure 3

Numerical values of the coefficients β_j in 12 decimals exact.

n=1 $+.1000000000000_{10}+1$

n=2 $+.1000000000000_{10}+1$ $+.1250000000000_{10}-0$

n=3 $+.1000000000000_{10}+1$ $+.1481481481480_{10}-0$ $+.5486968449929_{10}-2$

n=4 $+.1000000000000_{10}+1$ $+.1562500000000_{10}-0$ $+.7812500000000_{10}-2$ $+.1220703125000_{10}-3$

n=5 $+.1000000000000_{10}+1$ $+.1600000000001_{10}-0$ $+.8960000000002_{10}-2$ $+.2047999999999_{10}-3$ $+.1638400000000_{10}-5$

n=6 $+.1000000000000_{10}+1$ $+.1620370370370_{10}-0$ $+.9602194787377_{10}-2$ $+.2572016460904_{10}-3$ $+.3175328964079_{10}-5$ $+.1470059705593_{10}-7$

n=7 $+.1000000000000_{10}+1$ $+.1632653061224_{10}-0$ $+.9995835068722_{10}-2$ $+.2914237629366_{10}-3$ $+.4361444071156_{10}-5$ $+.3236693188241_{10}-7$
$+.9436423289340_{10}-1$

n=8 $+.1000000000000_{10}+1$ $+.1640625000000_{10}-0$ $+.1025390625000_{10}-1$ $+.3147125244141_{10}-3$ $+.5245208740234_{10}-5$ $+.4842877388000_{10}-7$
$+.2328306436539_{10}-9$ $+.4547473508865_{10}-12$

n=9 $+.1000000000000_{10}+1$ $+.1646090534980_{10}-0$ $+.1043201409000_{10}-1$ $+.3311750504760_{10}-3$ $+.5905727923437_{10}-5$ $+.6186321805323_{10}-7$
$+.3776753238902_{10}-9$ $+.1243375551902_{10}-11$ $+.1705590606174_{10}-14$

n=10 $+.1000000000000_{10}+1$ $+.1650000000000_{10}-0$ $+.1056000000000_{10}-1$ $+.3432000000001_{10}-3$ $+.6406399999999_{10}-5$ $+.7280000000000_{10}-7$
$+.5119999999997_{10}-9$ $+.2175999999999_{10}-11$ $+.5120000000002_{10}-14$ $+.5120000000000_{10}-17$

References

[1] Houwen, P.J. van der, One step methods for linear initial value problems 1, Polynomial Methods. Report TW 119, Math. Centre, Amsterdam (1970).

[2] Houwen, P.J. van der, Finite difference methods for solving partial differential equations. Tract 20, Math. Centre, Amsterdam (1968).

[3] Förch, G.J.R., Houwen, P.J. van der and Riet, R.P. van de, Colloquium stabiliteit van differentieschema's I. Syllabus 2.1., Math. Centrum, Amsterdam (1967).

[4] Dekker, L., Dekker, T.J., Houwen, P.J. van der and Spijker, M.N., Colloquium stabiliteit van differentieschema's II. Syllabus 2.2., Math. Centrum, Amsterdam (1968).

[5] Richtmeyer, R.D., Difference methods for initial value problems. Interscience, London (1957).

[6] Kruseman Aretz, F.E.J., Het MC-ALGOL 60-systeem voor de X8 (MR 81). Math. Centrum, Amsterdam (1966).