stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE
(DEPARTMENT OF NUMERICAL MATHEMATICS)

NN 16/78

SEPTEMBER

B.P. SOMMEIJER

PARABOLIC PDE, AN ALGOL 68 IMPLEMENTATION FOR THE TIME INTEGRATION OF SEMI-DISCRETIZED PARABOLIC DIFFERENTIAL EQUATIONS

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 65-04, 65M20

PARABOLIC PDE, an ALGOL 68 implementation for the time integration of semi-discretized parabolic differential equations

by

B.P. Sommeijer

ABSTRACT

This note describes the implementation in ALGOL 68 of an algorithm based on a class of stabilized, explicit three-step Runge-Kutta formulas. This implementation is performed in such a way that it is quite simple to deal with differential equations arizing from systems of parabolic equations in one, two or three space dimensions.

KEY WORDS & PHRASES: *parabolic partial differential equations, semi-discretization, numerical software.*

CONTENTS

1. INTRODUCTION

This note describes an ALGOL 68 implementation of a class of explicit, three-step Runge-Kutta methods for the time integration of semi-discretized (systems of) parabolic equations.

In [2] VERWER constructs an algorithm and implements it in FORTRAN. The main purpose of this note is to convert the above implementation into ALGOL 68 with the aim to facilitate the user to define his problem as close as possible to the mathematical formulation. This ALGOL 68 implementation especially gives considerable advantage when dealing with systems of parabolic equations in more than one space dimension. This class of problems is the very one that can be integrated adequately with an explicit method.

On the basis of a simple example the usage of the ALGOL 68 implementation will be illustrated. Numerical results of the program, applied to three semi-discretized problems are reported.

2. DEFINITION OF THE INTEGRATION FORMULAS; SHORT DESCRIPTION OF THE ALGORITHM

We consider a system of ordinary differential equations of the form

$$\frac{du}{dt} = f(t,u), \tag{2.1}$$

resulting from semi-discretization of a system of parabolic equations. During the construction of the algorithm to integrate (2.1) we base ourselves on the assumption that the spectrum of the Jacobian matrix consists of a long narrow strip around the negative axis of the complex plane (this is a reasonable assumption). The integration will be performed with explicit, three-step Runge-Kutta schemes; the degree of these schemes varies from two to twelve and the order is one or two. The integration formulas may be represented in the following form:

$$
\begin{aligned}
u_{n+1}^{(0)} &= u_n, \\
u_{n+1}^{(j)} &= (1-b_j)u_n + b_j u_{n-1} + c_j h f(t_{n-1},u_{n-1}) + \\
&\qquad \lambda_j h f(t_n+\mu_{j-1}h, u_{n+1}^{(j-1)}), \qquad j = 1,\ldots,m \\
u_{n+1} &= d u_{n+1}^{(m)} + (1-d)u_{n-2}, \quad m \geq 2, \quad n = 2,3,\ldots
\end{aligned} \tag{2.2}
$$

where $\mu_0 = 0$, $\mu_j = -b_j + c_j + \lambda_j$, $j = 1,\ldots,m-1$. The vector u_n denotes the approximation to the exact solution $u(t)$ at $t = t_n$. The points t_j, $j = n-2,\ldots,n+1$, are the reference points of the three-step formula and $h = t_{n+1}-t_n$ denotes the step size.

Concerning the constraints the coefficients b_j, c_j, λ_j and d have to meet to obtain schemes of first and second order, we refer to [1].

For an extensive description of the algorithm we refer to [2]. Here we confine ourselves to a short summary, which serves merely as a guide to read the program.

When applied to the scalar testmodel

$$
\frac{du}{dt} = \delta u, \qquad \delta \in \mathbb{C}, \qquad \mathrm{Re}\,\delta < 0 \tag{2.3}
$$

scheme (2.2) yields a linear recurrence relation between u_{n+1}, u_n, u_{n-1} and u_{n-2}. For schemes of degree m and order p this relation delivers a boundary of absolute stability $\beta_p(m)$. There holds

$$
\begin{aligned}
\beta_1(m) &\simeq 5.15\ m^2 \\
\beta_2(m) &\simeq 2.29\ m^2\ .
\end{aligned} \tag{2.4}
$$

To obtain absolute stability it is necessary that

$$
h.\sigma \leq \beta_p(m), \tag{2.5}
$$

where σ denotes the spectral radius of the Jacobian of the right-hand side of (2.1).

Because of the relatively high degree and large absolute stability region of these schemes there must be reckoned with a propagation of rounding errors per integration step. This phenomenon is called internal instability. While the absolute stability criterion (2.5) fixes a lowerbound for the degree when h and σ are given, the internal stability prescribes an upperbound for the degree. This upperbound depends on the machine precision and on the accuracy with which the solution is desired.

To be able to start the process the vectors u_1 and u_2 are necessary. They are calculated by means of a one-step scheme (of order 2) which is also formulated as a three-step scheme by choosing some parameters of (2.2) equal to zero. The boundary of absolute stability $\tilde{\beta}_2(m)$ of these one-step formulas is given by

$$\tilde{\beta}_2(m) \simeq 0.44\, m^2 + 0.03\, m^3. \tag{2.6}$$

Besides, a suitable initial value for the step size is calculated.
After every integration step the local truncation error LTE_p, depending on the order p of the scheme used, is estimated by

$$\begin{aligned} LTE_1 &= \frac{\frac{1}{2}-1.27}{0.27}\,[u_{n+1} - 2u_n + u_{n-1}], \\ LTE_2 &= \frac{\frac{1}{6}-0.44}{0.56}\,[u_{n+1} - 3(u_n-u_{n-1})-u_{n-2}]. \end{aligned} \tag{2.7}$$

Here we use the so called mixed error criterion; that means that the following relation has to be satisfied

$$\|LTE_p\| \le TOL * (1+\|u_{n+1}\|), \tag{2.8}$$

where TOL stands for the user specified tolerance and $\|\cdot\|$ denotes the divided Euclidean norm. During the start of the process (n=1,2) no error control is performed. If the third step is rejected the integration process is restarted with a ten times smaller step size.

When the error estimation prescribes a change of the stepsize new values for u_{n-1}, u_{n-2} and $f(t_{n-1},u_{n-1})$ has to be calculated. u_{n-1} and u_{n-2} are calculated by means of quadratic interpolation. $f(t_{n-1},u_{n-1})$ is calculated by evaluating the right-hand side of (2.1). For the variation of the step size the following formula is used:

$$\bar{\alpha} = \left(\frac{TOL+TOL*\|u_{n+1}\|}{\|LTE_p\|}\right)^{\frac{1}{p+1}} , \tag{2.9}$$

where p is the order.
The new step size is obtained by multiplying the old one by α, where α is defined as

$$\alpha = \begin{cases} \bar{\alpha}/2.0 & \text{if } p = 1 \\ \bar{\alpha}/1.6 & \text{if } p = 2 \end{cases} . \tag{2.10}$$

In the implementation some provisions are made to avoid marginal changes of the step size on the one hand and too rigorous changes on the other hand.

Concerning the implemented order control we remark that the integration process is continued with a second order scheme until the step size h has reached the value $hmax_2$, being the maximum step size with relation to stability. Next the order - except for some restrictions - is changed to one to let h grow (see (2.4) and (2.5)). When during the integration process with a first order scheme the step size h becomes smaller than $hmax_2$, the order is reset to two.

Finally we mention that we need an estimate of the spectral radius of the Jacobian of the right-hand side of (2.1). This is necessary because of the choice of a maximal h based on stability considerations and because of the calculation of the initial step size. The estimation of the spectral radius is performed by means of an adjusted power method.

3. PROGRAM, SPECIFICATION AND USAGE

The program text (the entire listing can be found in the Appendix) consists of a number of identity declarations; they represent the definition of:

a) the mode vec : mode vec = ref [ ][ ] real,
   the mode mat2 : mode mat2 = ref [ ][,] real,
   the mode mat3 : mode mat3 = ref [ ][, ,] real,

b) the mode vmm : mode vmm = union(vec,mat2,mat3),

c) the routine *fatal error* with specification:

   proc(int *error number*) void:

   This routine is called in case of a fatal error; it prints a suitable text, depending on the value of *error number*, and calls the CDC ALGOL 68 routine *error*, which causes a normal termination, triggers trace-back and performes a symbolic dump,

d) the dyadic operators *,/,- and + working on operands of mode vmm, possibly combined with operands of mode real,

e) the monadic operators normn and norm, working on operands of mode vmm delivering the divided Euclidean norm and the Euclidean norm, respectively,

f) the dyadic operator initial to initialize an operand of mode vmm to a given value of mode real,

g) the dyadic operator ** working on operands of mode real; this operator performes an exponentiation,

h) identifiers of mode [ ]ref[ ] real, containing the parameters c and λ of the integration schemes (see (2.2)),

i) The datastructure

```
mode info = struct (real h,hmin,sigma,inacc sigma,tol,
                    vmm  u0,u1,u2,du0,du1,
                    bool first call,
                    int  errorflag,sigma option,
                         max evals,steps,failures,
                         restarts,evals,sigma evals,
                         degree,maxdegree1,maxdegree2,
                         order,
                         steps after start,
                         steps after h,
                         steps after sigma),
```

j)

```
info default = (skip,skip,skip,skip,1.0E-4,
                skip,skip,skip,skip,skip,
                true,
```

```
                    0,2
                    10000,skip,skip,
                    skip,skip,skip,
                    skip,skip,skip,
                    skip,
                    skip,
                    skip,
                    skip),
```

k) the routine PARABOLIC PDE with specification:

```
proc(ref real t,real t end,vmm u,
     union(proc(real,vec)vec,proc(real,mat2)mat2,
           proc(real,mat3)mat3)uf,
     ref info info)void:
```

in which the parameters have the following meaning:

t independent variable of the differential equation (2.1).

input: initial value of the independent variable.

output: last point reached in the integration; when the integration process is succesful *t* is slightly beyond *t end*.

t end input: point at which the solution is desired.

u dependent variable of the differential equation (2.1). The correct mode for *u* must be chosen, depending on the space dimension (1, 2 or 3) of the problem. There are three different modes available: vec,mat2 and mat3 (see a)). The use of this mode will facilitate supplying the derivative (parameter *uf*).

For each discretized partial differential equation the components of the dependent variable associated to gridpoints in the discretized space, can be arranged into a one-, two- or three-times subscripted array, the mode of which we call P. Now a *system* of partial differential equations can be combined conveniently into an object of mode [ ]P. A reference to such a row, properly initialized with the initial value in the point *t*, must be given as parameter *u* (the lowerbound of this row must be one).

output: solution in the point *t end*.

uf a user supplied routine delivering the right-hand side of the differential equation (2.1). When *u* is of mode vec, *uf* must have

the mode proc(real,vec)vec and analogous for the modes mat2 and mat3.

info datastructure by which a number of options and control mechanisms can be specified in case of a first call. On subsequent calls *info* contains the information necessary for continuing the integration. On return of PARABOLIC PDE *info* contains information about the course of the integration.

Now we will discuss the fields of the structure *info*, in case of a first call:

h	output:	initial step size for a subsequent call.
hmin	output:	minimal step size used by PARABOLIC PDE.
sigma	input :	an upper estimate of the spectral radius of the Jacobian in case of *sigma option* = 1. No initialization is required in case of *sigma option* = 2 or 3.
	output:	an upper estimate of the spectral radius initialized by the user or by PARABOLIC PDE.
inacc sigma	output:	inaccurate estimate of the spectral radius used for its control.
tol	input :	tolerance for the local truncation error.
	output:	unchanged.
u0,u1,u2	output:	solution in $t, t-h$ and $t-2h$, respectively.
du0,du1	output:	derivative in t and $t-h$, respectively.
first call	input :	true
	output:	false
errorflag	output:	=0 normal exit, i.e. *t end* is reached.
		=1 *t end* is not reached because the total number of f-evaluations is greater than *max evals*.
		=2 maximal degree is too small because *tol/ small real* is too small . The process is not started.
		=3 the iteration method to estimate the spectral radius fails. The integration process is discontinued.
sigma option	input :	=1 the user must initialize *sigma*.
		=2 *sigma* is estimated and updated by PARABOLIC PDE.

=3 PARABOLIC PDE only estimates *sigma* at the first call.

output: =1 if *sigma option* is 1 or 3 at input.
=2 if *sigma option* is 2 at input.

max evals input : maximum number of f-evaluations to be spent.
output: unchanged.

steps output: total number of integration steps performed.

failures output: number of rejected integration steps.

restarts output: number of times the integration process is restarted.

evals output: total number of f-evaluations performed.

sigma evals output: number of f-evaluations used for the estimation of the spectral radius.

degree output: degree of the scheme used when reaching *t end*.

maxdegree 1 output: maximal degree for first order schemes.

maxdegree 2 output: maximal degree for second order schemes.

order output: order of the scheme used when reaching *t end*.

steps after start output: number of steps performed after start or restart.

steps after h output: number of steps performed after a change of the step size or a change of the order.

steps after sigma output: number of steps performed after an estimation of the spectral radius.

In case of a first call only the fields *first call, tol, sigma option, max evals* and - if *sigma option* =1 - *sigma* require an initialization. The user can confine himself to declaring a variable of mode info and assigning to it the variable *default*, described at j). The result is that the fields *first call, tol, sigma option* and *max evals* get the values true, 1.E-4,2 and 10000, respectively.

If on return *errorflag* equals 0 or 1 all fields of *info* are ready for a next call. If *errorflag* =0 the user needs only to define a new output point *t end* and call again. If on return *errorflag* =1 and the user wants to continue the integration process he must increase the field *max evals* before calling again. If *errorflag* =2 the process has not been started; usually a too high precision is required. If *errorflag* =3 the process can be restarted in the point *t* provided the user is able to take action with respect to

the estimation of the spectral radius. We notice that the points *t end* in the subsequent calls should be conceived as output points; the choice of these points does not influence the integration process.

We shall illustrate the use of PARABOLIC PDE on the basis of the following

EXAMPLE. Consider the nonlinear system of parabolic equations:

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{2}{x} v \frac{\partial u}{\partial x} + \sin(tx) \\
&\qquad\qquad\qquad\qquad\qquad\qquad , \quad 0 < x < 1,\ t > 0 \\
&\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2} + \frac{2}{x} u \frac{\partial v}{\partial x} + \cos(tx) \\
&u(0,x) = 1 - x^2, \quad v(0,x) = 1 + x - \tfrac{1}{2}x^2, \quad 0 \le x \le 1, \\
&\frac{\partial u}{\partial x}(t,0) = 0, \quad v(t,0) = 1 \quad , \quad t > 0 \quad , \\
&u(t,1) = 0 \quad , \quad \frac{\partial v}{\partial x}(t,1) = 0 \quad , \quad t > 0 \quad .
\end{aligned}
\tag{3.1}
$$

Discretizing the space dimension of (3.1), using central differences, on the grid $\{x_j \mid x_j = j/10,\ j = 0,\ldots,10\}$ we find the system of ordinary differential equations

$$
\frac{d}{dt}\begin{pmatrix} U_0 \\ \vdots \\ U_j \\ \vdots \\ U_{10} \\ V_0 \\ \vdots \\ V_j \\ \vdots \\ V_{10} \end{pmatrix} = 100 \begin{pmatrix} -6U_0+6U_1 \\ \vdots \\ U_{j-1}(1-V_j/j)-2U_j+U_{j+1}(1+V_j/j) \\ \vdots \\ 0 \\ 0 \\ \vdots \\ V_{j-1}(1-U_j/j)-2V_j+V_{j+1}(1+U_j/j) \\ \vdots \\ 2V_9-2V_{10} \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ \sin(tj/10) \\ \vdots \\ 0 \\ 0 \\ \vdots \\ \cos(tj/10) \\ \vdots \\ \cos(t) \end{pmatrix}
\tag{3.2}
$$

where U_j and V_j are approximations to $u(t,x_j)$ and $v(t,x_j)$, respectively. We ask for the solution of (3.2) in the points t = .01,.1,.5,1,5,10 while

the integration process has to be performed with the local tolerance equal to 10^{-3} and the number of f-evaluation does not exceed 1000. The "calling program" for doing this can look like:

```
begin
   proc derivative = (real t, vec u) vec:
      begin heap [1:2][0:10] real du;
         du [1][10] := du [2][0] := 0;
         du [1][0] := 600*(-u[1][0] + u[1][1]);
         du [2][10] := 200*(u[2][9] - u[2][10]) + cos(t);
         for j to 9
         do real vjj = u[2][j]/j, wjj = u[1][j]/j;
            du [1][j] := 100*(u[1][j-1]*(1-vjj) - 2*u[1][j] +
                         + u[1][j+1]*(1+vjj)) + sin(t*j/10);
            du [2][j] := 100*(u[2][j-1]*(1-wjj) - 2*u[2][j] +
                         + u[2][j+1]*(1+wjj)) + cos(t*j/10)
         od;
         du
      end;

[1:2][0:10] real u; real t := 0, tend;
for j from 0 to 10
do real xj = j/10.0;
   u[1][j] := 1.0-xj*xj; u[2][j] := 1.0+xj*(1-xj/2)
od;
info c := default; tol of c := 1.0E-3; max evals of c := 1000;
for i to 6 while error flag of c = 0
do tend := case i
           in 0.01, 0.1, 0.5, 1.0, 5.0, 10.0
           esac;
           parabolic pde(t, tend, u, derivative, c);
           if error flag of c = 0
           then print((newline, tend, u, steps of c, evals of c))
           elif error flag of c = 2
           then print((newline, "the integration process is not started"))
```

```
            else print((newline, t, u, evals of c))
            fi
        od
    end
```

REMARKS

(i) As may be clear from the foregoing the particular choice of the mode of the dependent variable has the aim to make it the user comfortable in defining his problem. Because of this reason and because of the readability of the program we sacrificed efficiency if necessary. This choice, of course, causes some overhead. The alternative is to make a program where the user has to store all components of the semi-discretized system of partial differential equations into one, long vector. The proportional profit in execution time decreases with increasing complexity of the differential equation. For the above, non-complex example the profit is about 32 p.c.

(ii) If a domain contains "holes", the user has to add differential equations

$$\frac{d}{dt}\, u[\ldots][\ldots] = 0$$

and initial conditions

$$u[\ldots][\ldots] = 0$$

at the gridpoints in the "hole".

(iii) Concerning the error control, it is emphasized that the error estimates are not strict bounds, but they are fairly reliable over one step. Over a large number of steps the errors may, at most linearly, accumulate depending on the problem and there is no guarantee that the overall error will be less than the bound specified. The user can check the results by repeating the calculation with a different value of *tol*.

In addition, the user has to take care for a realistic proportion between *tol* and the accuracy of the space discretization.

4. NUMERICAL EXAMPLES

To test PARABOLIC PDE, we applied three problems to it. We confine ourselves to define the *partial* differential equations. The first and second problem come from [2] and are semi-discretized by means of a Galerkin method. For the resulting systems of ordinary differential equations we refer to [2]. The third problem is semi-discretized using finite differences. The results of this last example will be compared with the exact solution.

PROBLEM 1. We consider a one-dimensional, non-linear system of equations from electricity theory:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \varepsilon\rho \frac{\partial^2 u}{\partial x^2} - g(u-v), \\ \frac{\partial v}{\partial t} &= \rho \frac{\partial^2 v}{\partial x^2} + g(u-v), \end{aligned} \qquad 0 \le x \le 1, \quad t \ge 0, \tag{4.1}$$

where $g(z) = \exp(\mu z/3) - \exp(-2\mu z/3)$, $\mu = 17.19$, $\varepsilon = 0.143$, $\rho = 0.1743$. The corresponding initial and boundary conditions read:

$$u(0,x) = 1, \quad v(0,x) = 0, \qquad 0 \le x \le 1,$$

$$\frac{\partial u}{\partial x}(t,0) = v(t,0) = 0, \qquad t > 0,$$

$$u(t,1) = 1, \quad \frac{\partial v}{\partial x}(t,1) = 0, \qquad t > 0.$$

The x-interval is subdivided into 31 equidistant grid points. The system is integrated over the interval [0,20], with output points in t = .01, .1, 1, 5, 10 and 20, for three values of *tol*, viz. 10^{-3}, 10^{-4}, 10^{-5}. Results of the integration are listed in table 4.1.

We give the approximations to u(t,x) for the specified number of output points and some grid points x_j. To give some information about the course of the integration processes we list for each output point the values of the following fields of the structure *info: steps, failures, evals, sigma evals* and *sigma*.

	x / t	0	0.2	0.4	0.6	0.8	0.9	steps	fail-ures	evals	sigma evals	sigma
	0							0	0	19	19	4453
	10^{-2}	.5283	.6879	.6879	.6879	.6879	.6883	38	0	124	37	1277
	10^{-1}	.2193	.4742	.5105	.5124	.5242	.5773	58	0	178	40	1277
TOL =	1	.0422	.1988	.3676	.5122	.6516	.7377	79	0	306	56	1036
10^{-3}	5	.0330	.1637	.3227	.4812	.6398	.7319	96	0	484	56	1036
	10	.0327	.1623	.3203	.4785	.6374	.7300	109	0	627	59	1036
	20	.0327	.1624	.3204	.4785	.6375	.7301	147	1	1059	83	1036
	0							0	0	19	19	4453
	10^{-2}	.5271	.6870	.6870	.6870	.6870	.6874	66	0	204	54	1226
	10^{-1}	.2184	.4738	.5099	.5118	.5236	.5767	103	0	327	94	1009
TOL =	1	.0419	.1981	.3671	.5123	.6521	.7381	142	0	476	97	1009
10^{-4}	5	.0330	.1637	.3227	.4811	.6398	.7319	170	0	714	100	1009
	10	.0327	.1623	.3204	.4785	.6375	.7301	188	0	886	103	1009
	20	.0327	.1623	.3204	.4785	.6375	.7301	203	0	1069	106	1009
	0							0	0	19	19	4453
	10^{-2}	.5268	.6867	.6867	.6867	.6867	.6871	122	0	418	142	1172
	10^{-1}	.2181	.4737	.5098	.5117	.5234	.5766	195	0	591	151	1172
TOL =	1	.0419	.1979	.3670	.5124	.6523	.7383	273	0	844	160	1172
10^{-5}	5	.0330	.1637	.3226	.4811	.6397	.7318	321	0	1177	166	1172
	10	.0328	.1624	.3205	.4786	.6376	.7302	350	0	1456	183	1032
	20	.0327	1623	.3204	.4785	.6375	.7301	377	0	1710	186	1032

Table 4.1.

PROBLEM II. The second problem we consider is a two-dimensional, non-linear diffusion problem. The temperature distribution in a filement can be described by:

$$\frac{\partial u}{\partial t} = \beta(u)\left[\frac{\partial^2 u}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right] + \gamma(u), \qquad 0 \le r \le re,\ 0 \le z \le ze,\ t \ge 0,$$

(4.2)
$$u(0,r,z) = 500, \qquad 0 \le r \le re,\ 0 \le z \le ze,$$

$$u(t,r,0) = u(t,r,ze) = 500, \qquad 0 \le r \le re,\ t \ge 0,$$

$$\frac{\partial u}{\partial r}(t,0,z) = 0, \quad \frac{\partial u}{\partial r}(t,re,z) = \mu(u), \quad 0 \le z \le ze,\ t \ge 0,$$

where $re = 10^{-4}$, $ze = 0.15$, $\beta(u) = \lambda(u)/\rho(u)$, $\gamma(u) = \eta(u)/\rho(u)$, $\mu(u) = \psi(u)/\lambda(u)$. The functions λ, ρ, η and ψ are defined by

$$\lambda(u) = 418.4*(0.1287496_{10}-13\ u^4 - 0.116873_{10}-9\ u^3 + 0.384891_{10}-6\ u^2 - 0.569812_{10}-3\ u + 0.57185303),$$

$$\rho(u) = 193*(0.14644_{10}5 + 1.8513*(u - 1040)),$$

$$\eta(u) = 7.1486^2*\pi^{-2}*(-0.378808_{10}9 + 0.267688_{10}7\ u + 0.1724588_{10}3\ u^2),$$

$$\psi(u) = -0.56696_{10}-7\ u^4(-0.270491_{10}-17\ u^5 + 0.3438691_{10}-13\ u^4 - 0.163905_{10}-9\ u^3 + 0.3305647_{10}-6\ u^2 - 0.1194_{10}-3\ u + 0.2667112_{10}-1).$$

We discretize the internal [0,re] by choosing 5 equidistant points. To discretize the interval [0,ze] we define a partition of the z-axis: $\{z_j \mid z_0 = 0,\ z_j < z_{j+1},\ j = 0,\ldots,19,\ z_{20} = ze\}$. We now choose:

$$z_j = 0.06 * j/20 \qquad \text{for } j = 0,\ldots,5$$

$$z_j = z_{j-1} + 0.012 \qquad \text{for } j = 6,\ldots,15$$

$$z_j = z_{j-1} + 0.003 \qquad \text{for } j = 16,\ldots,20.$$

The system is integrated over the internal [0,1] with output points in t = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9 and 1.0, for three values of *tol*, viz. 10^{-3}, 10^{-4}, 10^{-5}. The results of the integration are listed in table 4.2.

From the calculations we find that the solution hardly varies in the radial (r-) direction and that the solution is approximately symmetric with respect to the plane z = ze/2. So we suffice to give the results for some values of $z_j \in [0,ze/2]$ and for r = 0. Table 4.2 yields the same type of information as table 4.1.

	t \ z	0.003	0.006	0.009	0.015	0.075	steps	fail-ures	evals	sigma evals	sigma
	0						0	0	9	9	434068
	0.1	700.7	748.6	756.8	758.0	758.0	137	5	1464	54	367399
	0.2	951.8	1123.3	1171.5	1184.6	1184.8	270	11	2937	185	325713
	0.4	1744.8	2314.4	2511.9	2590.8	2593.8	460	16	5024	292	249563
TOL = 10^{-3}	0.6	2551.1	3154.3	3285.8	3324.9	3326.4	617	18	6760	363	230840
	0.8	2817.9	3304.7	3382.2	3397.4	3397.8	713	19	7858	401	199143
	0.9	2847.5	3317.6	3388.3	3400.8	3400.9	743	19	8215	407	199143
	1.0	2859.0	3322.0	3390.1	3401.5	3401.6	770	19	8542	410	199143
	0						0	0	9	9	434068
	0.1	700.7	748.7	756.9	758.1	758.1	178	0	1447	38	360024
	0.2	952.2	1124.1	1172.6	1185.8	1186.0	329	0	2991	70	292102
	0.4	1746.5	2317.2	2515.3	2594.8	2597.7	545	0	5077	134	252207
TOL = 10^{-4}	0.6	2552.0	3153.8	3285.3	3324.3	3325.9	747	0	7057	175	233241
	0.8	2817.0	3304.4	3382.0	3397.3	3397.6	859	0	8184	190	233241
	0.9	2847.7	3317.5	3388.2	3400.7	3400.9	894	0	8603	193	233241
	1.0	2858.7	3321.9	3390.1	3401.5	3401.6	925	0	8981	199	233241
	0						0	0	9	9	434068
	0.1	701.0	749.5	758.0	759.3	759.3	244	0	2172	41	360465
	0.2	952.8	1125.3	1174.2	1187.7	1187.9	429	0	3897	99	315795
	0.4	1748.9	2319.8	2517.7	2597.1	2600.0	741	0	6786	178	263147
TOL = 10^{-5}	0.6	2552.1	3153.3	3284.7	3323.7	3325.2	997	0	9151	237	217459
	0.8	2815.2	3303.5	3381.4	3396.8	3397.2	1221	0	11107	264	217459
	0.9	2846.7	3317.0	3388.0	3400.6	3400.8	1288	0	11683	273	217459
	1.0	2858.3	3321.7	3390.0	3401.5	3401.6	1329	0	12124	279	217459

Table 4.2.

PROBLEM III. The last problem we consider is a three-dimensional, nonlinear system of equations:

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= a_1(\Delta u - 6u/(x^2+y^2+z^2)) - u + b_1(vw - e^{25t/6}u^5) \\
\frac{\partial v}{\partial t} &= a_2(\Delta v - 20e^{t/2}u) - \frac{1}{2}v + b_2(uw - e^{-t/3}v^2) \\
\frac{\partial w}{\partial t} &= a_3(\Delta w - 42e^{t/6}v) - \frac{1}{3}w + b_3(uv - e^{-7t/6}w),
\end{aligned}
\tag{4.3}
$$

$$t \geq 0,\ 0 \leq x,y,z \leq 1.$$

For all values of the constants a_i and b_i, $i = 1,2,3$ the exact solution is given by

$$
\begin{pmatrix} u \\ v \\ w \end{pmatrix}(t,x,y,z) = \begin{pmatrix} e^{-t} & (x^2+y^2+z^2) \\ e^{-t/2} & (x^2+y^2+z^2)^2 \\ e^{-t/3} & (x^2+y^2+z^2)^3 \end{pmatrix} . \tag{4.4}
$$

Initial and boundary conditions (here of Dirichlet type) follow from (4.4). To semi-discretize (4.3) we choose an equidistant grid with increment 0.1 in all directions, resulting in 729 gridpoints. The coefficients a_i and b_i, $i = 1,2,3$ are set to 1, 5, 10 and 10, 5, 1, respectively. The system is integrated over the interval [0,1] with $tol = 10^{-4}$. The results are listed in table 4.3. In the output points t = .001, .01, .1, .5, 1 the solution is compared with the exact solution. The minimal number of correct significant digits of the value $U_{ijk}(t)$ are listed, where $U_{ijk}(t)$ denotes the numerical approximation to u(t,x,y,z) in the gridpoint ($x_i = i/10$, $y_j = j/10$, $z_k = k/10$), i.e.

$$\min_{i,j,k} \{-^{10}\log | u(t,x_i,y_j,z_k) - U_{ijk}(t) |\}. \tag{4.5}$$

The components V and W are treated in a similar way. The remaining part of table 4.3 gives some information about the integration process and is of the same type as table 4.1.

t	minimal significant digits			steps	fail-ures	evals	sigma evals	sigma
	U	V	W					
.001	3.49	3.50	2.21	10	0	70	46	12402
.01	2.87	2.62	1.75	22	0	103	46	12402
.1	2.67	2.67	1.74	41	0	219	49	12402
.5	2.82	2.74	1.78	63	0	443	52	12402
1.0	3.02	2.88	1.85	90	0	716	55	12402

Table 4.3.

5. REFERENCES

[1] VERWER, J.G., *A class of stabilized three-step Runge-Kutta methods for the numerical integration of parabolic equations*, Journal of Computational and Applied Mathematics, Volume 3, Number 3, 1977.

[2] VERWER, J.G., *An implementation of a class of stabilized, explicit methods for the time integration of parabolic equations*, Report NW 38/77, Mathematisch Centrum, Amsterdam, 1977 (Prepublication).

APPENDIX

Here we give a list of the entire program text.

```
PDE:
        '

'BEGIN'

  'MODE'  'VEC' = 'REF'[ ][ ]'REAL',
          'MAT2' = 'REF'[ ][ , ]'REAL',
          'MAT3' = 'REF'[ ][ , , ]'REAL';
  'MODE'  'VMM' = 'UNION'('VEC','MAT2','MAT3');


  'PROC' FATAL ERROR = ('INT' ERROR NUMBER)'VOID':
     ( 'CASE' ERROR NUMBER
       'IN' PRINT((NEWLINE,"DEVISION BY ZERO")),
            PRINT((NEWLINE,"IMPROPER MODES")),
            PRINT((NEWLINE,"EXPONENTIATION OF NEGATIVE REAL VALUE")),
            PRINT((NEWLINE,"LOWERBOUND UNEQUAL ONE"))
       'ESAC';
       ERROR
     );


  'OP' * =('REAL'R,'VMM'U)'VMM':
          'CASE' U
          'IN' ('VEC'V):('INT' UPBV='UPB'V,
                              LWBV1='LWB'V[1], UPBV1='UPB'V[1];
                        'HEAP'[1:UPBV][LWBV1:UPBV1]'REAL'Z;
                        'FOR' I 'TO' UPBV 'DO'
                          'REF'[ ]'REAL' VI=V[I], ZI=Z[I];
                          'FOR' J 'FROM' LWBV1 'TO' UPBV1 'DO'
                          ZI[J]:=VI[J]*R 'OD' 'OD';Z),
               ('MAT2'M2):('INT' UPBM2='UPB'M2,
                              LWBM21= 'LWB'M2[1], UPBM21= 'UPB'M2[1],
                              LWBM22=2'LWB'M2[1], UPBM22=2'UPB'M2[1];
                        'HEAP'[1:UPBM2][LWBM21:UPBM21,
                                        LWBM22:UPBM22]'REAL'Z;
                        'FOR' I 'TO' UPBM2 'DO'
                          'REF'[ , ]'REAL' M2I=M2[I], ZI=Z[I];
                          'FOR' J 'FROM' LWBM21 'TO' UPBM21 'DO'
                          'FOR' K 'FROM' LWBM22 'TO' UPBM22 'DO'
                          ZI[J,K]:=M2I[J,K]*R 'OD''OD''OD'; Z),
               ('MAT3'M3):('INT' UPBM3='UPB'M3,
                              LWBM31= 'LWB'M3[1], UPBM31= 'UPB'M3[1],
                              LWBM32=2'LWB'M3[1], UPBM32=2'UPB'M3[1],
                              LWBM33=3'LWB'M3[1], UPBM33=3'UPB'M3[1];
                        'HEAP'[1:UPBM3][LWBM31:UPBM31,LWBM32:UPBM32,
                                        LWBM33:UPBM33]'REAL'Z;
                        'FOR' I 'TO' UPBM3 'DO'
                         'REF'[ , , ]'REAL' M3I=M3[I], ZI=Z[I];
                         'FOR' J 'FROM' LWBM31 'TO' UPBM31 'DO'
                         'FOR' K 'FROM' LWBM32 'TO' UPBM32 'DO'
                          FOR' L 'FROM' LWBM33 'TO' UPBM33 'DO'
                         ZI[J,K,L]:=M3I[J,K,L]*R 'OD''OD''OD''OD'; Z)
          'ESAC';
```

```
'OP' / =('VMM'U,'REAL'R)'VMM':
   'IF' R/=0.0
   'THEN' 1.0/R * U
   'ELSE' FATAL ERROR(1); 'SKIP'
   'FI';


'OP' - =('VMM'U1,U2)'VMM':
        'CASE' U1
        'IN' ('VEC'V):('INT' UPBV='UPB'V,
                            LWBV1='LWB'V[1], UPBV1='UPB'V[1];
                       'HEAP'[1:UPBV][LWBV1:UPBV1]'REAL'Z;
                       'CASE' U2
                       'IN' ('VEC'W):
                       'FOR' I 'TO' UPBV 'DO'
                          'REF'[ ]'REAL' VI=V[I], WI=W[I], ZI=Z[I];
                          'FOR' J 'FROM' LWBV1 'TO' UPBV1 'DO'
                          ZI[J]:=VI[J]-WI[J] 'OD' 'OD'
                       'OUT' FATAL ERROR(2) 'ESAC'; Z),
            ('MAT2'M2):('INT' UPBM2='UPB'M2,
                            LWBM21='LWB'M2[1], UPBM21='UPB'M2[1],
                            LWBM22=2'LWB'M2[1], UPBM22=2'UPB'M2[1];
                        'HEAP'[1:UPBM2][LWBM21:UPBM21,
                                        LWBM22:UPBM22]'REAL'Z;
                     'CASE' U2
                     'IN' ('MAT2' N2):
                     'FOR' I 'TO' UPBM2 'DO'
                       'REF'[ , ]'REAL' M2I=M2[I], N2I=N2[I], ZI=Z[I];
                        'FOR' J 'FROM' LWBM21 'TO' UPBM21 'DO'
                        'FOR' K 'FROM' LWBM22 'TO' UPBM22 'DO'
                        ZI[J,K]:=M2I[J,K]-N2I[J,K] 'OD''OD''OD'
                     'OUT' FATAL ERROR(2) 'ESAC'; Z),
            ('MAT3'M3):('INT' UPBM3='UPB'M3,
                            LWBM31= 'LWB'M3[1], UPBM31= 'UPB'M3[1],
                            LWBM32=2'LWB'M3[1], UPBM32=2'UPB'M3[1],
                            LWBM33=3'LWB'M3[1], UPBM33=3'UPB'M3[1];
                        'HEAP'[1:UPBM3][LWBM31:UPBM31,LWBM32:UPBM32,
                                        LWBM33:UPBM33]'REAL'Z;
                   'CASE' U2
                   'IN' ('MAT3' N3):
                   'FOR' I 'TO' UPBM3 'DO'
                     'REF'[ , , ]'REAL' M3I=M3[I],N3I=N3[I],ZI=Z[I];
                      'FOR' J 'FROM' LWBM31 'TO' UPBM31 'DO'
                      'FOR' K 'FROM' LWBM32 'TO' UPBM32 'DO'
                      'FOR' L 'FROM' LWBM33 'TO' UPBM33 'DO'
                     ZI[J,K,L]:=M3I[J,K,L]-N3I[J,K,L] 'OD''OD''OD''OD'
                   'OUT' FATAL ERROR(2) 'ESAC'; Z)
        'ESAC';
```

```
'OP' + =('VMM'U1,U2)'VMM':
        'CASE' U1
        'IN' ('VEC'V):('INT' UPBV='UPB'V,
                            LWBV1='LWB'V[1], UPBV1='UPB'V[1];
                      'HEAP'[1:UPBV][LWBV1:UPBV1]'REAL'Z;
                      'CASE' U2
                      'IN' ('VEC'W):
                      'FOR' I 'TO' UPBV 'DO'
                         'REF'[ ]'REAL' VI=V[I], WI=W[I], ZI=Z[I];
                         'FOR' J 'FROM' LWBV1 'TO' UPBV1 'DO'
                        ZI[J]:=VI[J]+WI[J] 'OD' 'OD'
                      'OUT' FATAL ERROR(2) 'ESAC'; Z),
            ('MAT2'M2):('INT' UPBM2='UPB'M2,
                            LWBM21='LWB'M2[1], UPBM21='UPB'M2[1],
                            LWBM22=2'LWB'M2[1], UPBM22=2'UPB'M2[1];
                       'HEAP'[1:UPBM2][LWBM21:UPBM21,
                                       LWBM22:UPBM22]'REAL'Z;
                    'CASE' U2
                    'IN' ('MAT2' N2):
                    'FOR' I 'TO' UPBM2 'DO'
                      REF'[ , ]'REAL' M2I=M2[I], N2I=N2[I], ZI=Z[I];
                       'FOR' J 'FROM' LWBM21 'TO' UPBM21 'DO'
                        FOR' K 'FROM' LWBM22 'TO' UPBM22 'DO'
                       ZI[J,K]:=M2I[J,K]+N2I[J,K] 'OD''OD''OD'
                    'OUT' FATAL ERROR(2) 'ESAC'; Z),
            ('MAT3'M3):('INT' UPBM3='UPB'M3,
                            LWBM31= 'LWB'M3[1], UPBM31= 'UPB'M3[1],
                            LWBM32=2'LWB'M3[1], UPBM32=2'UPB'M3[1],
                            LWBM33=3'LWB'M3[1], UPBM33=3'UPB'M3[1];
                       'HEAP'[1:UPBM3][LWBM31:UPBM31,LWBM32:UPBM32,
                                       LWBM33:UPBM33]'REAL'Z;
                  'CASE' U2
                  'IN' ('MAT3' N3):
                  'FOR' I 'TO' UPBM3 'DO'
                    'REF'[ , , ]'REAL' M3I=M3[I],N3I=N3[I],ZI=Z[I];
                     'FOR' J 'FROM' LWBM31 'TO' UPBM31 'DO'
                     'FOR' K 'FROM' LWBM32 'TO' UPBM32 'DO'
                     'FOR' L 'FROM' LWBM33 'TO' UPBM33 'DO'
                    ZI[J,K,L]:=M3I[J,K,L]+N3I[J,K,L] 'OD''OD''OD''OD'
                  'OUT' FATAL ERROR(2) 'ESAC'; Z)
        'ESAC';
```

```
'OP' 'NORMN' = ('VMM'U)'REAL':
  'NORM' U/SQRT( 'CASE' U
                 'IN' ('VEC'V) : 'UPB'V*('UPB'V[1]-'LWB'V[1]+1),
                      ('MAT2'M2): 'UPB'M2*('UPB'M2[1]-'LWB'M2[1]+1)*
                                          (2'UPB'M2[1]-2'LWB'M2[1]+1),
                      ('MAT3'M3): 'UPB'M3*('UPB'M3[1]-'LWB'M3[1]+1)*
                                          (2'UPB'M3[1]-2'LWB'M3[1]+1)*
                                          (3'UPB'M3[1]-3'LWB'M3[1]+1)
                 'ESAC');


'OP' 'NORM' = ('VMM'U)'REAL':
  ('REAL' S:=0.0;
   'CASE' U
   'IN'  ('VEC' V):'FOR' I 'TO' 'UPB'V 'DO'
                   'REF'[ ]'REAL'VI=V[I];
                   'FOR' J 'FROM' 'LWB'V[1] 'TO' 'UPB'V[1] 'DO'
                    S+:=('REAL'VIJ=VI[J];VIJ*VIJ) 'OD' 'OD' ,
         ('MAT2' M2):'FOR' I 'TO' 'UPB'M2 'DO'
                   'REF'[ , ]'REAL'M2I=M2[I];
                    'FOR' J 'FROM' 'LWB'M2[1] 'TO' 'UPB'M2[1] 'DO'
                    'FOR' K 'FROM'2'LWB'M2[1] 'TO'2'UPB'M2[1] 'DO'
                     S+:=('REAL'MIJK=M2I[J,K];MIJK*MIJK)'OD''OD''OD',
         ('MAT3' M3):'FOR' I 'TO' 'UPB'M3 'DO'
                   'REF'[ , , ]'REAL'M3I=M3[I];
                    'FOR' J 'FROM' 'LWB'M3[1] 'TO' 'UPB'M3[1] 'DO'
                    'FOR' K 'FROM'2'LWB'M3[1] 'TO'2'UPB'M3[1] 'DO'
                    'FOR' L 'FROM'3'LWB'M3[1] 'TO'3'UPB'M3[1] 'DO'
             S+:=('REAL'MIJKL=M3I[J,K,L];MIJKL*MIJKL)'OD''OD''OD''OD'
   'ESAC';
   SQRT(S) );


'OP' 'INITIAL' =('VMM'U,'REAL'R)'VMM':
        'CASE' U
        'IN' ('VEC'V):
                    ('FOR' I 'TO' 'UPB'V 'DO'
                     'REF'[ ]'REAL'VI=V[I];
                     'FOR' J 'FROM' 'LWB'V[1] 'TO' 'UPB'V[1] 'DO'
                     VI[J]:=R 'OD' 'OD'; V),
             ('MAT2'M2):
                    ('FOR' I 'TO' 'UPB'M2 'DO'
                     'REF'[ , ]'REAL'M2I=M2[I];
                     'FOR' J 'FROM' 'LWB'M2[1] 'TO' 'UPB'M2[1] 'DO'
                     'FOR' K 'FROM'2'LWB'M2[1] 'TO'2'UPB'M2[1] 'DO'
                     M2I[J,K]:=R 'OD''OD''OD'; M2),
             ('MAT3'M3):
                    ('FOR' I 'TO' 'UPB'M3 'DO'
                    'REF'[ , , ]'REAL'M3I=M3[I];
                     'FOR' J 'FROM' 'LWB'M3[1] 'TO' 'UPB'M3[1] 'DO'
                     'FOR' K 'FROM'2'LWB'M3[1] 'TO'2'UPB'M3[1] 'DO'
                     'FOR' L 'FROM'3'LWB'M3[1] 'TO'3'UPB'M3[1] 'DO'
                     M3I[J,K,L]:=R 'OD''OD''OD''OD'; M3)
        'ESAC';

'PRIO' 'INITIAL' = 9;
```

```
'OP' ** = ('REAL'A,B)'REAL':
  'IF' A > 0.0
  'THEN' EXP(B*LN(A))
  'ELSE' FATAL ERROR(3); 'SKIP'
  'FI';


[ ]'REF'[ ]'REAL' C ORDER1=

  ('HEAP'[1:2]'REAL':= ( .1961791081531 53E-01,-.8191338398877 96E-01),
   'HEAP'[1:3]'REAL':= ( .1187708332041 69E-01, .2319380149589 62E-01,
                        -.8362249040858 72E-01                        ),
   'HEAP'[1:4]'REAL':= ( .5022357841454 05E-02, .1604034984953 77E-01,
                         .2445499909298 13E-01,-.8417069442354 70E-01),
   'HEAP'[1:5]'REAL':= ( .2565525423254 33E-02, .7326180473587 80E-02,
                         .1792050964244 27E-01, .2499524823170 57E-01,
                        -.8446647958768 64E-01                        ),
   'HEAP'[1:6]'REAL':= ( .1475900868412 80E-02, .3938787866917 68E-02,
                         .8549286606744 26E-02, .1890824093515 79E-01,
                         .2528480454902 99E-01,-.8467480270415 08E-01),
   'HEAP'[1:7]'REAL':= ( .9360090016571 79E-03, .2381613383135 33E-02,
                         .4799841610170 22E-02, .9340934572831 31E-02,
                         .1958372029527 56E-01, .2548902132161 90E-01,
                        -.8467629524244 60E-01                        ),
   'HEAP'[1:8]'REAL':= ( .6212086436750 04E-03, .1529883883949 21E-02,
                         .2939554524977 17E-02, .5308799237957 16E-02,
                         .9779567248235 42E-02, .1991177687823 79E-01,
                         .2557437820416 22E-01,-.8483410409050 63E-01),
   'HEAP'[1:9]'REAL':= ( .4378639416871 00E-03, .1050701049570 03E-02,
                         .1948109393915 03E-02, .3342059087351 29E-02,
                         .5688262703620 38E-02, .1012289569476 10E-01,
                         .2019304003266 75E-01, .2565526644668 04E-01,
                        -.8487581961913 59E-01                        ),
   'HEAP'[1:10]'REAL':=( .3190035516066 34E-03, .7504415521387 59E-03,
                         .1355128438523 41E-02, .2241770519027 81E-02,
                         .3620833930648 34E-02, .5945315127870 39E-02,
                         .1034579383527 73E-01, .2035524358626 69E-01,
                         .2566960052476 16E-01,-.8488287568115 15E-01),
   'HEAP'[1:11]'REAL':=( .2392905748315 34E-03, .5545506057033 32E-03,
                         .9819927709675 27E-03, .1582471169197 96E-02,
                         .2464820573060 43E-02, .3839820716849 31E-02,
                         .6161217374798 52E-02, .1056164965670 79E-01,
                         .2058074146393 80E-01, .2578652085097 10E-01,
                        -.8486108618281 70E-01                        ),
   'HEAP'[1:12]'REAL':=( .1839107700826 14E-03, .4204895887403 60E-03,
                         .7321756954280 57E-03, .1154857361682 40E-02,
                         .1748883042615 75E-02, .2622256596065 03E-02,
                         .3984302513266 84E-02, .6286010484694 27E-02,
                         .1065427889634 70E-01, .2061554786070 57E-01,
                         .2576889495306 74E-01,-.8496870258589 25E-01)
  );
```

```
[ ]'REF'[ ]'REAL' LA ORDER1=

  ('HEAP'[1:2]'REAL':= ( .236114213662133E-01, .809186111261504E+00),
   'HEAP'[1:3]'REAL':= ( .143058304739038E-01, .279630468008499E-01,
                         .810895217681313E+00                      ),
   'HEAP'[1:4]'REAL':= ( .604601593926948E-02, .193222931324168E-01,
                         .294798704381324E-01, .811443421696271E+00),
   'HEAP'[1:5]'REAL':= ( .310308507303124E-02, .885640714653679E-02,
                         .216533580523272E-01, .301941237256040E-01,
                         .811739206860413E+00                      ),
   'HEAP'[1:6]'REAL':= ( .179545954712348E-02, .478549256497673E-02,
                         .103753549279950E-01, .229237084688118E-01,
                         .306233945802681E-01, .811947529976877E+00),
   'HEAP'[1:7]'REAL':= ( .113130766493322E-02, .287826089439570E-02,
                         .580016214292456E-02, .112863654109042E-01,
                         .236595412149794E-01, .307901131589646E-01,
                         .811949022515172E+00                      ),
   'HEAP'[1:8]'REAL':= ( .757132294865622E-03, .186353369611307E-02,
                         .357827309500765E-02, .645748051106268E-02,
                         .118853580483738E-01, .241746594171167E-01,
                         .310101134988965E-01, .812106831363231E+00),
   'HEAP'[1:9]'REAL':= ( .531535941947433E-03, .127560436756909E-02,
                         .236524020440182E-02, .405769435085915E-02,
                         .690588808135792E-02, .122878703679312E-01,
                         .245036225997679E-01, .311098505248339E-01,
                         .812148546891862E+00                      ),
   'HEAP'[1:10]'REAL':=( .388018193257131E-03, .912675201054015E-03,
                         .164783698586841E-02, .272551148183035E-02,
                         .440121292680665E-02, .722471822342052E-02,
                         .125675683895335E-01, .247132280269183E-01,
                         .311366984352272E-01, .812155602953876E+00),
   'HEAP'[1:11]'REAL':=( .291393271477073E-03, .674703224401576E-03,
                         .119376551375253E-02, .192224289253953E-02,
                         .299186843520478E-02, .465774143076075E-02,
                         .746893835164741E-02, .127958247948468E-01,
                         .249199861274321E-01, .312038578351279E-01,
                         .812133813455542E+00                      ),
   'HEAP'[1:12]'REAL':=( .224316509871435E-03, .512691029141716E-03,
                         .892424643530829E-03, .140717446493261E-02,
                         .213033881270347E-02, .319324626489569E-02,
                         .485035466704903E-02, .764969261852752E-02,
                         .129599605413823E-01, .250613004257468E-01,
                         .312921276379243E-01, .812241429858616E+00)
  );


[ ]'REF'[ ]'REAL' C ORDER2=

  ('HEAP'[1:2]'REAL':= (-.208272450838904E-01,-.176251440856983E+00),
   'HEAP'[1:3]'REAL':= (-.521139515771502E-02,-.204537454654857E-01,
                        -.150676811658657E+00                      ),
   'HEAP'[1:4]'REAL':= (-.176147103226697E-02,-.599278115368554E-02,
                        -.197072880138351E-01,-.142474486805669E+00),
   'HEAP'[1:5]'REAL':= (-.107134982667340E-02,-.299826649785043E-02,
                        -.712560734620138E-02,-.199442161680483E-01,
                        -.137422017332368E+00                      ),
```

```
    'HEAP'[1:6]'REAL':= (-.604231670180290E-03,-.159896048375198E-02,
                         -.342559706795734E-02,-.740230723385568E-02,
                         -.198255427211443E-01,-.135145414990056E+00),
    'HEAP'[1:7]'REAL':= (-.278318214941009E-03,-.759118883031255E-03,
                         -.162347074429806E-02,-.330855861378489E-02,
                         -.713176540776259E-02,-.194704888086196E-01,
                         -.134458609951604E+00                      ),
    'HEAP'[1:8]'REAL':= (-.334232361412569E-03,-.797991874144314E-03,
                         -.147651320422721E-02,-.254478755055698E-02,
                         -.441770433556241E-02,-.831853342706601E-02,
                         -.201674003517510E-01,-.131979437775493E+00),
    'HEAP'[1:9]'REAL':= (-.158892315555575E-03,-.393069506998231E-03,
                         -.746927988185854E-03,-.130401770897648E-02,
                         -.223868649325021E-02,-.397090940159806E-02,
                         -.775863209229052E-02,-.197134148378149E-01,
                         -.132298893246350E+00                      ),
    'HEAP'[1:10]'REAL':=(-.121634600402596E-03,-.285848818062940E-03,
                         -.515794575668280E-03,-.852757517498779E-03,
                         -.137638312131043E-02,-.225710356079413E-02,
                         -.391592582882930E-02,-.764252602126261E-02,
                         -.197179875092944E-01,-.130466073638779E+00),
    'HEAP'[1:11]'REAL':=(-.662549438807673E-04,-.156067719540023E-03,
                         -.281948853670529E-03,-.465572994842910E-03,
                         -.746665755072325E-03,-.120341693364751E-02,
                         -.200516410945224E-02,-.357248721108888E-02,
                         -.719819395117924E-02,-.192024247631614E-01,
                         -.132831492930022E+00                      ),
    'HEAP'[1:12]'REAL':=(-.102994642718880E-03,-.233554144657848E-03,
                         -.402675009004956E-03,-.627463855782561E-03,
                         -.935735406620758E-03,-.137525755447902E-02,
                         -.203399874416459E-02,-.308979758208251E-02,
                         -.495345032599209E-02,-.879895286766497E-02,
                         -.202795277896725E-01,-.129842261469204E+00)
  );

[ ]'REF'[ ]'REAL' LA ORDER2=

  ('HEAP'[1:2]'REAL':= ( .1157728502079l7E+00, .146657402150214E+01),
    'HEAP'[1:3]'REAL':= ( .322706885592909E-01, .134046076063337E+00,
                          .144099939230382E+01                      ),
    'HEAP'[1:4]'REAL':= ( .133871778214583E-01, .430088863850882E-01,
                          .140105103325491E+00, .143279706745083E+01),
    'HEAP'[1:5]'REAL':= ( .698672737950515E-02, .199346838452588E-01,
                          .487542837640687E-01, .143571376454619E+00,
                          .142774459797752E+01                      ),
    'HEAP'[1:6]'REAL':= ( .404201634876247E-02, .107707066239925E-01,
                          .233452828887895E-01, .515966013501086E-01,
                          .145212460730911E+00, .142546799563522E+01),
    'HEAP'[1:7]'REAL':= ( .250603246302696E-02, .637655067062040E-02,
                          .128626589598059E-01, .250863264662972E-01,
                          .528138271220564E-01, .145809052032436E+00,
                          .142478119059676E+01                      ),
    'HEAP'[1:8]'REAL':= ( .171740945875112E-02, .422943439361717E-02,
                          .812597847930835E-02, .146732920126255E-01,
                          .270192532661166E-01, .549219749365157E-01,
                          .147340106604082E+00, .142230201842065E+01),
```

```
    'HEAP'[1:9]'REAL':= ( .118497529615894E-02, .284489191348582E-02,
                          .527875153282853E-02, .906579186025930E-02,
                          .154529857439050E-01, .275515487743281E-01,
                          .550704625416572E-01, .147252559486519E+00,
                          .142262147389151E+01                      ),
    'HEAP'[1:10]'REAL':=( .872783455089815E-03, .205324088453152E-02,
                          .370790338925039E-02, .613438873286459E-02,
                          .990882002529492E-02, .162718375233707E-01,
                          .283274752602443E-01, .558390840810137E-01,
                          .148799097042613E+00, .142078865428394E+01),
    'HEAP'[1:11]'REAL':=( .651788683046089E-03, .150791044134571E-02,
                          .266511608463482E-02, .428578241618491E-02,
                          .666023888900999E-02, .103513856309803E-01,
                          .165742099478320E-01, .283755209595168E-01,
                          .553606804108775E-01, .146972053779980E+00,
                          .142315407357518E+01                      ),
    'HEAP'[1:12]'REAL':=( .506971342083145E-03, .115940247548512E-02,
                          .201964144358370E-02, .318762357481359E-02,
                          .483179707553261E-02, .725433442802687E-02,
                          .110418894355304E-01, .174581139505522E-01,
                          .296446922358879E-01, .572907032015069E-01,
                          .148817451454787E+00, .142016484211436E+01)
  );

[ ]'REF'[ ]'REAL' LA START=

  ('HEAP'[1:2]'REAL':= ( .500000000000000E+00, .100000000000000E+01),
    'HEAP'[1:3]'REAL':= ( .126608170000000E+00, .500000000000000E+00,
                          .100000000000000E+01                      ),
    'HEAP'[1:4]'REAL':= ( .469321699961169E-01, .158054716000000E+00,
                          .500000000000000E+00, .100000000000000E+01),
    'HEAP'[1:5]'REAL':= ( .224728454998609E-01, .663203570561843E-01,
                          .171211150000000E+00, .500000000000000E+00,
                          .100000000000000E+01                      ),
    'HEAP'[1:6]'REAL':= ( .125236991593134E-01, .340607016912871E-01,
                          .763290844635254E-01, .178036992000000E+00,
                          .500000000000000E+00, .100000000000000E+01),
    'HEAP'[1:7]'REAL':= ( .770639365836728E-02, .198641861737173E-01,
                          .408535008054067E-01, .821994942335218E-01,
                          .182050816000000E+00, .500000000000000E+00,
                          .100000000000000E+01                      ),
    'HEAP'[1:8]'REAL':= ( .508510799953263E-02, .126230349649284E-01,
                          .245473612167101E-01, .451875345069677E-01,
                          .859462164497931E-01, .184616448000000E+00,
                          .500000000000000E+00, .100000000000000E+01),
    'HEAP'[1:9]'REAL':= ( .353435602181355E-02, .853260075977219E-02,
                          .159562606514131E-01, .277222917290663E-01,
                          .481258801498683E-01, .884868940567993E-01,
                          .186357896000000E+00, .500000000000000E+00,
                          .100000000000000E+01                      ),
    'HEAP'[1:10]'REAL':=( .255731571963207E-04, .604300164829694E-02,
                          .109795602340940E-01, .183220858017351E-01,
                          .299757552073369E-01, .502114708086094E-01,
                          .902904742681478E-01, .187594930000000E+00,
                          .500000000000000E+00, .100000000000000E+01),
    'HEAP'[1:11]'REAL':=( .191070301553967E-02, .443895271219574E-02,
                          .788881113551870E-02, .127799643987028E-01,
                          .200627788700101E-01, .316337674220291E-01,
                          .517459968097393E-01, .916176293139941E-01,
                          .188505622000000E+00, .500000000000000E+00,
                          .100000000000000E+01                      ),
```

```
    'HEAP'[1:12]'REAL':=( .146551036584494E-02, .335795377137199E-02,
                          .586417870669934E-02, .928689073356087E-02,
                          .141436426176712E-01, .213812201947804E-01,
                          .328895812991588E-01, .529082908860623E-01,
                          .926229273207144E-01, .189195696000000E+00,
                          .500000000000000E+00, .100000000000000E+01)
  );


[ ]'REF'[ ]'REAL' C START=

  ('HEAP'[1:2]'REAL':= ( 0.0,0.0),
    'HEAP'[1:3]'REAL':= ( 0.0,0.0,0.0),
    'HEAP'[1:4]'REAL':= ( 0.0,0.0,0.0,0.0),
    'HEAP'[1:5]'REAL':= ( 0.0,0.0,0.0,0.0,0.0),
    'HEAP'[1:6]'REAL':= ( 0.0,0.0,0.0,0.0,0.0,0.0),
    'HEAP'[1:7]'REAL':= ( 0.0,0.0,0.0,0.0,0.0,0.0,0.0),
    'HEAP'[1:8]'REAL':= ( 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0),
    'HEAP'[1:9]'REAL':= ( 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0),
    'HEAP'[1:10]'REAL':= ( 0.0,0.0,0.0,0.0,00,0.0,0.0,0.0,0.0),
    'HEAP'[1:11]'REAL':= ( 0.0,.0,.0,.0,0,0.0,00,0.0,0.0,
                           0.0),
    'HEAP'[1:12]'REAL':= ( 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
                           0.0,0.0)
  );

'MODE' 'INFO' = 'STRUCT'( 'REAL' H,HMIN,SIGMA,INACC SIGMA,TOL,
                          'VMM'  U0,U1,U2,DU0,DU1,
                          'BOOL' FIRST CALL,
                          'INT'  ERRORFLAG, SIGMA OPTION,
                                 MAX EVALS, STEPS, FAILURES,
                                 RESTARTS, EVALS, SIGMA EVALS,
                                 DEGREE, MAXDEGREE1, MAXDEGREE2,
                                 ORDER,
                                 STEPS AFTER START,
                                 STEPS AFTER H,
                                 STEPS AFTER SIGMA );


'INFO' DEFAULT = ('SKIP','SKIP','SKIP','SKIP',1.E-4,
                  'SKIP','SKIP','SKIP','SKIP','SKIP',
                  'TRUE',
                  0,2,
                  10000,'SKIP','SKIP',
                  'SKIP','SKIP','SKIP',
                  'SKIP','SKIP','SKIP',
                  'SKIP',
                  'SKIP',
                  'SKIP',
                  'SKIP' );
```

```
'PROC' PARABOLIC PDE = ('REF''REAL'T,'REAL'TEND,'VMM'U,'UNION'('PROC'(
                       'REAL','VEC')'VEC','PROC'('REAL','MAT2')'MAT2',
                       'PROC'('REAL','MAT3')'MAT3')UF,'REF''INFO'INFO)
                       'VOID':
  'BEGIN'

    'IF'('CASE' U
         'IN' ('VEC'  V ) : 'LWB'V,
              ('MAT2' M2) : 'LWB'M2,
              ('MAT3' M3) : 'LWB'M3
         'ESAC')  /= 1
    'THEN' FATAL ERROR(4)
    'FI';

     'REF' 'REAL' H          = H 'OF' INFO,
                  HMIN       = HMIN 'OF' INFO,
                  SIGMA1     = SIGMA 'OF' INFO,
                  SIGMA2     = INACC SIGMA 'OF' INFO,
           'REAL' TOL        = TOL 'OF' INFO,
                  APR        = SMALLREAL,
     'REF' 'INT'  ERRORFLAG  = ERRORFLAG 'OF' INFO,
     'REF' 'BOOL' FIRST      = FIRST CALL 'OF' INFO,
     'REF' 'INT'  SIG OPT    = SIGMA OPTION 'OF' INFO,
           'INT'  MAX EVALS  = MAX EVALS 'OF' INFO,
     'REF' 'INT'  STEPS      = STEPS 'OF' INFO,
                  FAILS      = FAILURES 'OF' INFO,
                  RESTS      = RESTARTS 'OF' INFO,
                  EVALS      = EVALS 'OF' INFO,
                  SIG EVALS  = SIGMA EVALS 'OF' INFO,
                  DEGREE     = DEGREE 'OF' INFO,
                  MAXDEG1    = MAXDEGREE1 'OF' INFO,
                  MAXDEG2    = MAXDEGREE2 'OF' INFO,
                  ORDER      = ORDER 'OF' INFO,
                  STAFSTART  = STEPS AFTER START 'OF' INFO,
                  STAFH      = STEPS AFTER H 'OF' INFO,
                  STAFSIG    = STEPS AFTER SIGMA 'OF' INFO;

     'REAL' TOLLIP=1.0E4*APR;


  'PROC' NEWSPACE = ('VMM'U)'VMM':
         'CASE' U
         'IN'  ('VEC'  V):'HEAP'[1:'UPB'V]['LWB'V[1]:'UPB'V[1]]'REAL',
               ('MAT2' M2):'HEAP'[1:'UPB'M2]['LWB'M2[1]:'UPB'M2[1],
                                        2'LWB'M2[1]:2'UPB'M2[1]]'REAL',
               ('MAT3' M3):'HEAP'[1:'UPB'M3]['LWB'M3[1]:'UPB'M3[1],
                                        2'LWB'M3[1]:2'UPB'M3[1],
                                        3'LWB'M3[1]:3'UPB'M3[1]]'REAL'
         'ESAC';
```

```
        'IF' FIRST
        'THEN' U0 'OF' INFO:= NEWSPACE(U);
               U1 'OF' INFO:= NEWSPACE(U);
               U2 'OF' INFO:= NEWSPACE(U);
              DU0 'OF' INFO:= NEWSPACE(U);
              DU1 'OF' INFO:= NEWSPACE(U)
        'FI';

           'VMM' U0 = U0 'OF' INFO,
                U1 = U1 'OF' INFO,
                U2 = U2 'OF' INFO,
               DU0 = DU0'OF' INFO,
               DU1 = DU1'OF' INFO;


 'PROC' F =('REAL'T,'VMM'U)'VMM':
   'CASE' U
   'IN' ('VEC' V):(UF!('PROC'('REAL','VEC')'VEC' PVV):PVV(T,V)!
                                                  FATAL ERROR(2);'SKIP'),
        ('MAT2' M2):(UF!('PROC'('REAL','MAT2')'MAT2' PMM2):PMM2(T,M2)!
                                                  FATAL ERROR(2);'SKIP'),
        ('MAT3' M3):(UF!('PROC'('REAL','MAT3')'MAT3' PMM3):PMM3(T,M3)!
                                                  FATAL ERROR(2);'SKIP')
   'ESAC';


'OP' <=  = ('VMM'U1,U2)'VMM':
  'CASE' U1
  'IN' ('VEC' V) : (U2 ! ('VEC'W): V:=W ! FATAL ERROR(2);'SKIP'),
       ('MAT2' M2) : (U2 ! ('MAT2'N2): M2:=N2 !FATAL ERROR(2);'SKIP'),
       ('MAT3' M3) : (U2 ! ('MAT3'N3): M3:=N3 ! FATAL ERROR(2);'SKIP')
  'ESAC';

  'PRIO' <=  = 1;


'PROC' SET HMAX = 'VOID':
  HMAX:=(5.15*MAXDEG1*MAXDEG1/SIGMA1,
         2.29*MAXDEG2*MAXDEG2/SIGMA1);


'PROC' COEFS = ('REAL'MU,'REF''REAL'C0,C1,C2)'VOID':
  ('REAL'H=(MU-1.0)/2.0; C0:=MU*H; C1:=MU*(2.0-MU); C2:=(MU-2.0)*H);


'PROC' H START = 'REAL':
  'BEGIN'
    U<=F(T+1.0/SIGMA1,U0+DU0/SIGMA1); EVALS+:=1;
    'REAL' H=SQRT(TOL*(1.0+'NORMN'U0)/('NORMN'(U-DU0)/SIGMA1+APR))/
                  (10.0*SIGMA1);
    'REAL'BETA=(0.03*MAXDEG2+0.44)*MAXDEG2*MAXDEG2/SIGMA1;
    ( H > BETA ! BETA ! H )
  'END' # HSTART #;
```

```
        ('MAT2' M2 ):('INT' UPBM2='UPB'M2,
                            LWBM21= 'LWB'M2[1],UPBM21= 'UPB'M2[1],
                            LWBM22=2'LWB'M2[1],UPBM22=2'UPB'M2[1];
                       'HEAP'[1:UPBM2][LWBM21:UPBM21,
                                       LWBM22:UPBM22]'REAL'Z;
                       'FOR' I 'TO' UPBM2 'DO'
                       'REF'[ , ]'REAL' M2I=M2[I], ZI=Z[I];
                       'FOR' J 'FROM' LWBM21 'TO' UPBM21 'DO'
                       'FOR' K 'FROM' LWBM22 'TO' UPBM22 'DO'
                       'REAL'RA=(RANDOM*2.0-1.0)*TOLLIP;
                      ZI[J,K]:=('REAL'M2IJ=M2I[J,K];
                                       (M2IJ=0.0!RA!(RA+1.0)*M2IJ))
                       'OD' 'OD' 'OD' ; Z),
        ('MAT3' M3 ):('INT' UPBM3='UPB'M3,
                            LWBM31= 'LWB'M3[1],UPBM31= 'UPB'M3[1],
                            LWBM32=2'LWB'M3[1],UPBM32=2'UPB'M3[1],
                            LWBM33=3'LWB'M3[1],UPBM33=3'UPB'M3[1];
                       'HEAP'[1:UPBM3][LWBM31:UPBM31,LWBM32:UPBM32,
                                       LWBM33:UPBM33]'REAL'Z;
                       'FOR' I 'TO' UPBM3 'DO'
                       'REF'[ , , ]'REAL' M3I=M3[I], ZI=Z[I];
                       'FOR' J 'FROM' LWBM31 'TO' UPBM31 'DO'
                       'FOR' K 'FROM' LWBM32 'TO' UPBM32 'DO'
                       'FOR' L 'FROM' LWBM33 'TO' UPBM33 'DO'
                       'REAL'RA=(RANDOM*2.0-1.0)*TOLLIP;
                      ZI[J,K,L]:=('REAL'M3IJ=M3I[J,K,L];
                                       (M3IJ=0.0!RA!(RA+1.0)*M3IJ))
                       'OD' 'OD' 'OD' 'OD' ; Z)
    'ESAC';

'PROC' POWERMETHOD = 'VOID':
  'BEGIN' 'REAL'SIGM:=0.0,SIGM1,NORM;
          'VMM'AUX1:=NEW SPACE(U), AUX2:=NEW SPACE(U);
    U<=DU0; AUX1<='DISTURB' U0; AUX2<=F(T,AUX1);
    EVALS+:=1; SIG EVALS+:=1; STAFSIG:=0;
    ( SIG OPT = 3 ! SIG OPT:=1 );
    'REAL' NORM0=TOLLIP*( 'REAL'S0='NORM'AUX1; S0 < 1.0 ! 1.0 ! S0 );
    'BOOL' OUT:='FALSE', NO UPDATE:='FALSE';
    'FOR' K 'WHILE'
         ( K = 51 ! ERRORFLAG:=3; ENDPM );
        NORM:='NORM'(U-AUX2); SIGM1:=SIGM; SIGM:=NORM/NORM0;
         ( K = 3 ! ( SIGMA1 = 0.0 ! SIGMA2:=SIGM ));
         'IF' K > 2
         'THEN' 'IF' SIGMA1 /= 0.0
                'THEN' ( SIGM >= SIGMA2*0.9 ! NO UPDATE:='TRUE'
                                            ! SIGMA2:=SIGM; SIGMA1:=0.0 )
                'FI'
         'FI';
         'IF' NO UPDATE
         'THEN' OUT:='TRUE'
         'ELIF'('ABS'(SIGM1-SIGM)/SIGM <= 0.001) 'AND' K > 4
         'THEN'  SIGMA1:=SIGM*1.1;OUT:='TRUE'
         'ELSE'  U<=F(T,AUX1+(U-AUX2)/SIGM);
                 EVALS +:=1; SIG EVALS +:=1
         'FI';
         'NOT' OUT
    'DO'  'SKIP'  'OD'
   'EXIT' ENDPM:'SKIP'
  'END' #POWERMETHOD#;
```

```
'PROC' PARAMETERS = 'VOID':
  'IF' STAFSTART >= 2
  'THEN' 'IF' ORDER = 1
         'THEN' C:=C ORDER1[DEGREE-1];
                LA:=LA ORDER1[DEGREE-1];
                 B[1]:='CASE' DEGREE
                        'IN' 'SKIP',
                            .508727967095290E+00,
                            .509560637580446E+00,
                            .509914825396269E+00,
                            .509857456549888E+00,
                            .509678390414901E+00,
                            .509981771419273E+00,
                            .509637782352229E+00,
                            .509635656502979E+00,
                            .509618959570098E+00,
                            .509880951278163E+00,
                            .509524549723448E+00
                        'ESAC';
                 B[2]:= .545454545454545E+00

         'ELSE' C:=C ORDER2[DEGREE-1];
                LA:=LA ORDER2[DEGREE-1];
                 B[1]:='CASE' DEGREE
                        'IN' 'SKIP',
                            -.268261337736233E-01,
                            -.280884356184020E-01,
                            -.278187239988046E-01,
                            -.286526610855615E-01,
                            -.287331992850827E-01,
                            -.283378889783029E-01,
                            -.295164279931444E-01,
                            -.288902503205175E-01,
                            -.288400811815056E-01,
                            -.282269850319168E-01,
                            -.298918878159484E-01
                        'ESAC';
                 B[2]:= -.580645161290320E+00
         'FI'
  'ELSE' B:=(0.0,0.0);
         C := C START[DEGREE-1];
         LA:= LA START[DEGREE-1]
  'FI';

'OP''DISTURB' = ('VMM'U)'VMM':
     'CASE' U
     'IN' ('VEC'  V ):('INT' UPBV='UPB'V,
                           LWBV1='LWB'V[1],UPBV1='UPB'V[1];
                       'HEAP'[1:UPBV][LWBV1:UPBV1]'REAL'Z;
                       'FOR' I 'TO' UPBV 'DO'
                       'REF'[ ]'REAL' VI=V[I], ZI=Z[I];
                       'FOR' J 'FROM' LWBV1 'TO' UPBV1 'DO'
                       'REAL'RA=(RANDOM*2.0-1.0)*TOLLIP;
                       ZI[J]:=('REAL'VIJ=VI[J];
                                    (VIJ=0.0!RA!(RA+1.0)*VIJ))
                       'OD' 'OD'; Z),
```

```
'PROC' NEWH = 'VOID':
  ALFA:= 'IF' 'REAL' EPSERR = EPS/ERROR; EPSERR > 1 'AND' STAF H < 3
         'THEN' 1.0
         'ELSE' 'REAL' AA=EPSERR**(1/(ORDER+1))/(2-(ORDER-1)*0.4);
                'IF' AA > 0.9 'AND' AA < 1.1
                'THEN' 1.0
                'ELSE' H:=HOLD *
                          ( AA > 3 ! 3 !: AA < 0.1 ! 0.1 ! AA );
                       ( 'REAL'HM2=HMAX[ORDER]; H > HM2 ! H:=HM2 );
                       H/HOLD
                'FI'
         'FI' ;


'PROC' INTER1 = 'VOID':
  'BEGIN'
   'REAL'C10,C11,C12,C20,C21,C22;
   COEFS(2.0-ALFA,C10,C11,C12);  COEFS(2.0-2.0*ALFA,C20,C21,C22);
    U1<=C12*U2+C11*U1+C10*U0;
    U2<=(C22-C21*C12/C11)*U2+C21/C11*U1+(C20-C21*C10/C11)*U0;
   DU1<=F(T-H,U1); EVALS +:=1;
   STAF H :=0
  'END' #INTER1#;


'PROC' INTER2 = ('REAL'A)'VOID':
  'BEGIN'
    'REAL'C0,C1,C2;
    COEFS(2.0-A,C0,C1,C2);
    U<=C2*U2+C1*U1+C0*U0
  'END' #INTER2#;


'PROC' SHIFT = 'VOID':
  'BEGIN'
    U2<=U1; U1<=U0; U0<=U; DU1<=DU0; DU0<=F(T+HOLD,U0); EVALS+:=1;
    T+:=HOLD
  'END' #SHIFT#;


'PROC' RESTART = 'VOID':
  'BEGIN' RESTS+:=1; FAILS+:=3;
     DEGREE:= STAFSTART:= STAF H:=STAF SIG:= 0;
     T-:=H*2.0; H/:=10.0; HOLD:=H; U0<=U2; DU0<=F(T,U0); EVALS+:=1
  'END' #RESTART#;


'PROC' CHECK ORDER 1 TO 2 = 'VOID':
  ( H < HMAX[2] ! DEGREE:=0; ORDER:=2 );


'PROC' CHECK ORDER 2 TO 1 = 'VOID':
  'IF' 'REAL'HM2=HMAX[2]; STAF H >= 3 'AND' HOLD = HM2 'AND' H = HM2
  'THEN' ORDER:=1; ESTIMATE ERROR; NEWH;
         (ALFA <= 1.0 ! ORDER:=2 );
         H:=HM2; STAF H:=-1;
         ( ORDER = 1 ! DEGREE:=0 )
  'FI';
```

```
'PROC' MAXIMAL DEGREE = 'VOID':
  'BEGIN'
  [] 'REAL' Q=(3.E1,1.E2,7.E2,4.E3,3.E4,2.E5,9.E5,5.E6,3.E7,2.E8,1.E9);
     'REAL' E=TOL/APR;
     'INT' M;
     'IF'   Q[1] > E
     'THEN' ERRORFLAG:=2
     'ELSE' M:=11; 'WHILE' Q[M] > E 'DO' M-:=1 'OD';
            MAXDEG1:=M+1;
            'IF'   Q[1]*100.0 > E
            'THEN' ERRORFLAG:=2
            'ELSE' M:=11; 'WHILE' Q[M]*100.0 > E 'DO' M-:=1 'OD';
                   MAXDEG2:=M+1
            'FI'
     'FI'
  'END' #MAXIMAL DEGREE#;


'PROC' MINIMAL DEGREE = 'INT':
  'BEGIN' 'BOOL'START= STAFSTART < 2,
          'REAL'BETA=( ORDER = 2 ! 2.29 ! 5.15 );
          'INT'M:=2;
     'TO' ( ORDER = 2 ! MAXDEG2 ! MAXDEG1 ) - 1
     'WHILE' H > ( START ! M*0.03+0.44 ! BETA)*M*M/SIGMA1
     'DO' M+:=1
     'OD';
     ( M > 12 ! 12 ! M )
  'END' #MINIMAL DEGREE#;


'PROC' STEP = 'VOID':
  'BEGIN'
    'REAL'D= ( STAFSTART < 2 ! 1.0 ! 1.375-(ORDER-1)*0.6 );
    U<=DU0;
    'FOR' J 'TO' DEGREE-2
    'DO' U<=F(T+(C[J]+LA[J])*H,U0+H*(C[J]*DU1+LA[J]*U));
         EVALS +:=1
    'OD';
    U<=('INT'DGM1=DEGREE-1;F(T+(-B[1]+C[DGM1]+LA[DGM1])*H,
                  (1.0-B[1])*U0+B[1]*U1+H*(C[DGM1]*DU1+LA[DGM1]*U)));
    EVALS +:=1;
    U<=D*((1.0-B[2])*U0+B[2]*U1+H*(C[DEGREE]*DU1+LA[DEGREE]*U))+
       (1.0-D)*U2
  'END' #STEP# ;


'PROC' ESTIMATE ERROR = 'VOID':
  'BEGIN'
    EPS:=TOL*(1+'NORMN'U0);
    ERROR:=('CASE' ORDER
            'IN'   2.85 * 'NORMN'(U-2.0*U0+U1),
                   0.49 * 'NORMN'(U-3.0*(U0-U1)-U2)
            'ESAC')
  'END' # ESTIMATE ERROR #;
```

```
  'REAL' HOLD,EPS,ERROR,ALFA,
  'INT' REJECT:=0,
  [1:2] 'REAL' B,HMAX, 'REF' [ ] 'REAL' C,LA;
'IF' 'NOT' FIRST 'AND' T>=T END
'THEN' INTER2((T-TEND)/H)
'ELSE'
  ERRORFLAG:=0;
  MAXIMAL DEGREE; ( ERRORFLAG=2 ! EXIT );
  DEGREE:=0;
  'IF' 'NOT' FIRST
  'THEN' HOLD:=H
  'ELSE' STEPS :=FAILS :=RESTS :=EVALS :=SIG EVALS :=STAFSTART :=
         STAF H :=STAF SIG :=0;
         ORDER:=2; U0<=U; DU0<=F(T,U0); EVALS+:=1;
         DU1 'INITIAL' 0.0; U1 'INITIAL' 0.0; U2 'INITIAL' 0.0;
         'IF' SIG OPT /= 1
         'THEN' SIGMA1:=0.0; POWERMETHOD;
                ( ERRORFLAG = 3 ! EXIT )
         'FI';
          HOLD:=H:=HMIN:=HSTART;
          FIRST := 'FALSE'
  'FI';
  SETHMAX;
  'BOOL' CHECK DEGREE:='TRUE';
  'WHILE' T<T END 'OR' STAFSTART=1
  'DO' 'IF' CHECK DEGREE
       'THEN' 'INT' MOLD=DEGREE; DEGREE:=MINIMAL DEGREE;
              ( DEGREE /= MOLD ! PARAMETERS)
       'FI';
       CHECK DEGREE:='TRUE';
       ( EVALS >= MAX EVALS ! ERRORFLAG:=1; EXIT );
       ( H < HMIN ! HMIN:=H );
       STEP; STAFSTART+:=1; STEPS+:=1;
       'IF' STAFSTART < 3
       'THEN' SHIFT; STAF H+:=1; STAF SIG+:=1;
              'IF' STAFSTART = 1
              'THEN' CHECK DEGREE:='FALSE'
              'ELSE' DEGREE:=0
              'FI'
       'ELSE' ESTIMATE ERROR;
             'IF' EPS < ERROR
             'THEN' 'IF' STAFSTART = 3
                    'THEN' RESTART
                    'ELSE' 'IF' SIG OPT /= 1 'AND' STAF SIG /= 0
                           'THEN' SIGMA1:=0.0; POWERMETHOD;
                                  ( ERRORFLAG = 3 ! EXIT ! SETHMAX )
                           'FI';
                           HOLD:=H; NEWH;
                           ( ORDER = 1 ! CHECK ORDER 1 TO 2);
                           REJECT+:=1; FAILS+:=1;
                           'IF' REJECT = 3
                           'THEN' REJECT:=0; RESTS+:=1; DEGREE:=0;
                                  ORDER:=2; STAFSTART :=STAF H :=0;
                                  HOLD:=H:=HSTART
                           'ELSE' INTER1
                           'FI'
                    'FI'
```

```
                'ELSE' HOLD:=H; NEWH;
                       ( ORDER = 1 ! CHECK ORDER 1 TO 2);
                       ( ORDER = 2 ! CHECK ORDER 2 TO 1);
                       SHIFT; REJECT:=0; STAF H+:=1; STAF SIG+:=1;
                       'IF' SIG OPT /= 1 'AND' STAF SIG = 25
                       'THEN' POWERMETHOD;
                              ( ERRORFLAG = 3 ! EXIT ! SETHMAX )
                       'FI';
                       'IF' T < T END
                       'THEN' 'IF' H /= HOLD
                              'THEN' INTER1
                              'ELIF' DEGREE /= 0
                              'THEN' CHECK DEGREE:='FALSE'
                              'FI'
                       'FI'
                'FI'
        'FI'
    'OD';
    INTER2((T-T END)/HOLD);
    ( H /= HOLD ! INTER1 )
'FI' ;
    EXIT: 'SKIP'
'END' # PARABOLIC PDE #;

'PR' PROG 'PR'

'SKIP'

'END'
```