NASA TECHNICAL MEMORANDUM NASA TM-75557

NUMERICAL METHOD FOR SOLUTION OF SYSTEMS OF NON-STATIONARY SPATIALLY ONE-DIMENSIONAL NONLINEAR DIFFERENTIAL EQUATIONS

S. K. Morozov and O. P. Krasitskiy

(NASA-TM-75557) NUMERICAL METHOD FOR SOLUTION OF SYSTEMS OF NON-STATIONARY SPATIALLY ONE-DIMENSIONAL NONLINEAR DIFFERENTIAL EQUATIONS (National Aeronautics and Space Administration) 35 p HC A03/MF G3/64 N79-11804 Unclas 36859

Translation of: "Chislennyy Metod Resheniya Sistem Nestatsionarnykh Prostranstvenno-Odnomernykh Nelineynykh Differentsial'nykh Uravneniy", Academy of Sciences USSR, Institute of Space Research, Moscow, Report Pr-396, 1978, pp 1-37

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION WASHINGTION, D.C. 20546 November, 1978

STANDARD TITLE PAGE

1. Report No. NASA TM-75557	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle NUMERICAL METHOD FOR SOLUTION OF SYSTEMS OF NON-STATIONARY SPATIALLY ONE-DIMENSIONAL NONLINEAR DIFFERENTIAL EQUATIONS		5. Report Date November, 1978
		6. Performing Organization Code
7. Author(s) S. K. Morozov and O. P. Krasitskiy		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address .SCITRAN Box 5456 Santa Barbara, CA 93108		11. Contract or Grant No. NASW-3198
		13. Type of Report and Period Covered Translation
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes

Translation of: "Chislennyy Metod Resheniya Sistem Nestatsionarnykh Prostranstvenno-Odnomernykh Nelineynykh Differentsial'nykh Uravneniy", Academy of Sciences USSR, Institute of Space Research, Moscow, Report Pr-396, 1978, pp. 1-37.

16. Abstract

We propose a computational scheme and a standard program for solving systems of nonstationary spatially one-dimensional nonlinear differential equations using Newton's method. The proposed shceme is valuable because of its universality and the fact that it reduces to a minimum the work of programming. We give a detailed description and present the text of the standard program which realizes this computational scheme. The program is written in the FORTRAN language and can be used without change on electronic computers of type YeS and BESM-6. The standard program described permits us to find nonstationary (or stationary) solutions to systems of spatially one-dimensional nonlinear (or linear) partial differential equations. The proposed method may be used to solve a series of geophysical problems which take chemical reactions, diffusion, and heat conductivity into account; to evaluate nonstationary thermal fields in twodimensional structures when in one of the geometrical directions it can take a small number of discrete levels, and to solve problems in nonstationary gas dynamics.

17. Key Words (Selected by Author(s))		18. Distribution Statement Unclassified - Unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 33	22.

NASA-HQ

Numerical Method for Solution of Systems of Non-Stationary Spatially One-Dimensional Nonlinear Differential Equations

S. K. Morozov and O. P. Krasitskiy

We propose a computational scheme and a standard program for solving systems of nonstationary spatially one-dimensional nonlinear differential equations using Newton's method. The proposed scheme is valuable because of its universality and the fact that it reduces to a minimum the work of programming. We give a detailed description and present the text of the standard program which realizes this computational scheme. The program is written in the FORTRAN language and can be used without change on electronic computers of type YeS and BESM-6. The standard program described permits us to find nonstationary (or stationary) solutions to systems of spatially one-dimensional nonlinear (or linear) partial differential equations. The proposed method may be used to solve a series of geophysical problems which take chemical reactions, diffusion, and heat conductivity into account, to evaluate nonstationary thermal fields in two-dimensional structures when in one of the geometrical directions it can take a small number of discrete levels, and to solve problems in nonstationary gas dynamics.

INTRODUCTION

In contemporary science and engineering practice, one often encounters situations when a specialist not familiar with the methods of computational mathematics and the fine points of programming must solve on an electronic computer one or another mathematical problem. In this case, he turns to a library of standard programs. For example, standard programs have yielded a great extension of the solution to the Cauchy problem for systems of ordinary differential equations by methods of Runge-Kutta type. The level of the developments of present-day computational engineering permits us to pose a question about the creation of standard programs for more complex mathematical problems. In this work, we propose a standard program for solving the boundary problem for a system of nonstationary spatially one-dimensional equations. A very wide class of physical problems can be reduced to such a system. For example, many models of geophysical phenomena, in which diffusion (heat conductivity) is the dominant factor, are often formulated mathematically as mixed problems for equations of parabolic type: either initial or boundary conditions are given. As a rule, the unknowns are related to each other in a nonlinear manner, which significantly complicates the solution.

At the basis of the method is the use of the implicit difference scheme. To solve nonlinear difference equations at each time step, we use Newton's iterative method. At each iteration, a linear system of algebraic equations is solved by a modified method of Gauss, taking into account the sparseness of the matrix, by choosing a pivot element by column and a normalization by row. They behave in an analogous way in problems of mathematical chemistry (see collections [1,2]). In gephysics a similar method was used in work [3]. 4

The system of equations under consideration can be very stiff, i.e. it can contain time constants with essentially different values. For example, in geophysical and astrophysical problems describing the distribution of atmospheric gaseous components by height (the determining processes are chemical reactions and vertical diffusion), the stiffness depends on the great difference in the rates of chemical decay of the various components, and also on the coefficients of diffusion at various heights. It is known [4] that, on account of a stiff restriction intrinsic to them at a step of time, it is unreasonable to use explicit difference schemes to solve stiff systems of differential equations, because the utilization of such schemes requires a large outlay of machine time while, in most cases, such systems in practice cannot be solved by explicit methods on present-day electronic computers.

The proposed scheme is especially advantageous when the determination of an exact solution in time is not required, but we are required to find the steady-state behavior. In cases of the lack or the smallness of the coefficients of the spatial derivatives, as this often occurs in problems of mathematical chemistry, the scheme can be shown to be unstable (the scheme is A-stable [*]). In such cases, we need to watch for negativity of the eigenvalues or, at any rate, for positivity of the solution, and to control the pace of time or to use further iterations, excluding the result of the solution in the negative domain (see, for example, the article of Yu.M. Volin et al. and the article of B.V. Pavlov in collection [1].) /5

Linearization by Newton's method permits us to provide an effective iteration process for solving nonlinear problems, to easily guarantee the conservativity of the difference scheme (a property of the scheme without which it is often impossible to obtain a numerical solution to the original system of differential equations).

It is well-known how much work and time it takes to write down and debug a program. Therefore, by writing down the program, we set up the problem, reducing to a minimum the work of programming and ensuring the possibility of an operational introduction of changes in the original model (for example, in geophysical problems the addition of new components and chemical reactions). This problem is solved by the calculation of a Jacobian matrix by means of numerical differentiation. In this way, we decrease the probability of error in the written program, while the user is freed from tiresome work and can concentrate his attention on the physical formulation of the problem and on the numbers of the experiment.

1. THE MATHEMATICAL METHOD

Let there be given the system of equations

$$\frac{\partial y_k(t,x)}{\partial t} = f_k\left(t, x, y_i, \frac{\partial y_i}{\partial x}, \frac{\partial^2 y_i}{\partial x^2}\right), \qquad (1)$$

where $k,i = 1,\ldots,M$ (i.e. into f_k enter terms with different $i = 1,\ldots,M$), M is the number of equations, y_k is an unknown function, t is time, and x is the spatial coordinate.

We are given some initial conditions

[*] The definition of A-stablity can be found, for example, in the article by L.S. Polak et al. in the collection [2].

$$y_K(0,x) = F_K(x), \quad K = 1,\dots,M . \tag{2}$$

We seek a solution on the interval $a \leq x \leq b$.

The boundary conditions have the form:

$$\psi_K\left(y_i(a), \frac{\partial y_i(a)}{\partial x}\right) = 0 , \tag{3}$$

$$\varphi_K\left(y_i(b), \frac{\partial y_i(b)}{\partial x}\right) = 0 , \tag{4}$$

where $i,k = 1,\dots,M$.

We pass from continuous variables to discrete ones. For this we choose a uniform mesh in x:

$$x_n = a + (n-1)\cdot\Delta x \; ; n = 1,\dots,N; \; \Delta x = (b-a)/(N-1) ; \tag{5}$$

(if it is necessary to introduce a nonuniform mesh, we can usually make a substitution of x by x', so the mesh in x' will be uniform). The time steps will be numbered by means of the index j, while the moments of time corresponding to the steps will be denoted t_j. By means of $y^j_{k,n}$ we denote approximate values of the continuous variable $y_k(t,x)$ at the point (t_j,x_n). The derivatives entering into f_k are approximated by difference relations of not more than three points x_n. For example:

$$\frac{\partial y_i(t_j, x_n)}{\partial x} \approx \frac{y^j_{i,n+1} - y^j_{i,n-1}}{2\,\Delta x} , \tag{6}$$

$$\frac{\partial^2 y_i(t_j, x_n)}{\partial x^2} \approx \frac{y^j_{i,n+1} - 2\,y^j_{i,n} + y^j_{i,n-1}}{\Delta x^2} , \tag{7}$$

while the derivatives entering into ψ_K and φ_K are approximated by the relations

$$\frac{\partial y_i(t_j, a)}{\partial x} \approx \frac{y^j_{i,2} - y^j_{i,1}}{\Delta x} , \tag{8}$$

ORIGINAL PAGE IS
OF POOR QUALITY

$$\frac{\partial y_i(t_j, b)}{\partial x} \approx \frac{y^j_{i,N} - y^j_{i,N-1}}{\Delta x} . \tag{9}$$

In order to avoid restrictions on the time step Δt connected with instability, we make use of a two-tiered implicit scheme: /7

$$\frac{y^j_{k,n} - y^{j-1}_{k,n}}{\Delta t_j} = f^j_{k,n}(y^j_{i,n+1}, y^j_{i,n}, y^j_{i,n+1}), \tag{10}$$

where $f^j_{k,n}$ is an approximate value of f_k at the point n at moment of time t_j, which is obtained by the substitution of expressions(6) and (7) into f_k for $n = 2,\ldots,N-1$.

The boundary conditions can be approximated by the expressions

$$\psi^j_k(y^j_{i,1}, y^j_{i,2}) = 0 , \tag{11}$$

$$\varphi^j_k(y^j_{i,N-1}, y^j_{i,N}) = 0 , \tag{12}$$

where ψ^j_k and φ^j_k are approximate expressions for ψ_k and φ_k, which are used in the substitution of expressions (8),(9) into (3),(4).

Equation (10) has first-order accuracy in time. To increase the accuracy in time, we can put the right-hand side in the form

$$\beta \cdot f^j_{k,n} + (1-\beta) \cdot f^{j-1}_{k,n} , \quad \text{where} \quad \beta \geq 0,5 \tag{13}$$

For $\beta = 0.5$, equation (10) approximates the differential equation (1) with double precision in time. For $\beta < 0.5$, the difference scheme becomes unstable.

To increase the accuracy of approximating the boundary conditions, the derivatives entering into ψ_k and φ_k can be written in the form:

$$\frac{\partial y_i(t_j, a)}{\partial x} \approx \frac{-3y^j_{i,1} + 4y^j_{i,2} - y^j_{i,3}}{2\Delta x} , \tag{14}$$

$$\left.\frac{\partial y_i(t_j, b)}{\partial x}\right/ \approx \frac{y^j_{i,N-2} - 4y^j_{i,N-1} + 3y^j_{i,N}}{2\Delta x} . \qquad (15)$$

In this case the derivatives mentioned above are approximated with double precision in x.

Equations (10), (11), and (12) form a nonlinear system of algebraic equations, consisting of N·M equations and N·M unknowns. We will solve this system by Newton's method, i.e., at each iteration S we solve the following system of linear algebraic equations: /8

$$\frac{y^{j,S+1}_{k,n} - y^{j-1}_{k,n}}{\Delta t_j} = f^{j,S}_{k,n} + \sum_{i=1}^{M}\sum_{l=1}^{3} \frac{\partial f^{j,S}_{k,n}}{\partial y^{j,S}_{i,n+l-2}} \left(y^{j,S+1}_{i,n+l-2} - y^{j,S}_{i,n+l-2}\right), \qquad (16)$$

$$\sum_{i=1}^{M}\sum_{l=1}^{3} \frac{\partial \psi^{j,S}_{K}}{\partial y^{j,S}_{i,l}} \left(y^{j,S+1}_{i,l} - y^{j,S}_{i,l}\right) + \psi^{j,S}_{K} = 0 , \qquad (17)$$

$$\sum_{i=1}^{M}\sum_{l=1}^{3} \frac{\partial \varphi^{j,S}_{K}}{\partial y^{j,S}_{i,N+l-3}} \left(y^{j,S+1}_{i,N+l-3} - y^{j,S}_{i,N+l-3}\right) + \varphi^{j,S}_{K} = 0 . \qquad (18)$$

For S = 0, at the zero-th approximation the values begin with the previous time period: $y^{j,0}_{k,n} = y^{j-1}_{k,n}$. The derivatives of the functions $f_{k,n}$, ψ_K, and φ_K with respect to the unknowns (of the Jacobi matrix) are found by numerical differentiation.

The numerical differentiation is effected by the formula

$$\frac{\partial f_{k,n}}{\partial y_{i,n}} = \left(f_{k,n}(y_{i,n+1}, y_{i,n} + \Delta y_{i,n}, y_{i,n-1}) - f_{k,n}(y_{i,n+1}, y_{i,n}, y_{i,n-1})\right) / \Delta y_{i,n} .$$

For clarity and convenience of our further presentation, we represent the system of equations (16), (17), and (18) in matrix form:

$$\left\|\begin{matrix} A_N & B_N & C_N & 0 & \cdots & 0 & 0 & 0 & 0 \\ A_{N-1} & B_{N-1} & C_{N-1} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & A_{N-2} & B_{N-2} & C_{N-2} & \cdots & 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdots & A_3 & B_3 & C_3 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_2 & B_2 & C_2 \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_1 & B_1 & C_1 \end{matrix}\right\| \cdot \left\|\begin{matrix} F_N \\ F_{N-1} \\ F_{N-2} \\ \cdot \\ \cdot \\ \cdot \\ F_3 \\ F_2 \\ F_1 \end{matrix}\right\| = \left\|\begin{matrix} D_N \\ D_{N-1} \\ D_{N-2} \\ \cdot \\ \cdot \\ \cdot \\ D_3 \\ D_2 \\ D_1 \end{matrix}\right\| \qquad (19)$$

Here A, G, and C are square matrices of size M·M, and F and D are M-dimensional vectors. In this form, the matrix has a block-tridiagonal form under the condition that each of its block elements, in turn, is a square matris. In the upper and lower rows of the matrix equation (19) we write equations (18) and (17), while in the interior rows is equation (16). The vector /9

$$F_n = \left(y_{1,n}^{j,s+1}, \dots, y_{M,n}^{j,s+1} \right), \quad n = 1, \dots, N.$$

Expressions for the elements of the matrices A, B, and C and the components of the vector D have the form:

1) for n = N and i,k = 1,...,M

$$a_{k,i} = \frac{\partial \varphi_k^{j,s}}{\partial y_{i,N-2}^{j,s}}, \quad b_{k,i} = \frac{\partial \varphi_k^{j,s}}{\partial y_{i,N-1}^{j,s}}, \quad c_{k,i} = \frac{\partial \varphi_k^{j,s}}{\partial y_{i,N}^{j,s}},$$

$$d_k = \sum_{i=1}^{M} \sum_{\ell=1}^{3} \frac{\partial \varphi_k^{j,s}}{\partial y_{i,N+\ell-3}^{j,s}} \, y_{i,N+\ell-3}^{j,s};$$

2) for n = 2,...,N-2 and i,k = 1,...,M

$$a_{k,i} = -\frac{\partial f_{k,n}^{j,s}}{\partial y_{i,n+1}^{j,s}}, \quad b_{k,i} = \frac{\delta_{k,i}}{\Delta t_j} - \frac{\partial f_{k,n}^{j,s}}{\partial y_{i,n}^{j,s}}, \quad c_{k,i} = -\frac{\partial f_{k,n}^{j,s}}{\partial y_{i,n-1}^{j,s}};$$

$$d_k = \frac{y_{k,n}^{j-1}}{\Delta t_j} + f_{k,n}^{j,s} - \sum_{i=1}^{M} \sum_{\ell=1}^{3} \frac{\partial f_{k,n}^{j,s}}{\partial y_{k,n+\ell-2}^{j,s}},$$

where $\delta_{k,i}$ is the Kronecher delta symbol;

3) for n = 1 and i,k = 1,...,M

$$a_{k,i} = \frac{\partial \psi_k^{j,s}}{\partial y_{i,3}^{j,s}}, \quad b_{k,i} = \frac{\partial \psi_k^{j,s}}{\partial y_{i,2}^{j,s}}, \quad c_{k,i} = \frac{\partial \psi_k^{j,s}}{\partial y_{i,1}^{j,s}},$$

$$d_k = \sum_{i=1}^{M} \sum_{\ell=1}^{3} \frac{\partial \psi_k^{j,s}}{\partial y_{i,\ell}^{j,s}} \, y_{i,\ell}^{j,s}.$$

The linear system (19) at each iteration S is solved by a modification of the method Gauss, which takes into account the tridiagonality of the matrix, with the choice of a pivot element by row and a normalization by column [5]. The basic idea of the elimination method consists of the fact that we use a linear transformation to make the diagonal elements of the /10 matrix into ones, and the below-diagonal elements zeros. This is done sequentially from top to bottom. To demonstrate the recurrence formula for passing from n to n-1, we introduce the rectangular matrix

$$BZ = \left\| \begin{matrix} \tilde{B}_n & \tilde{C}_n & 0 & \tilde{D}_n \\ A_{n-1} & B_{n-1} & C_{n-1} & D_{n-1} \end{matrix} \right\| .$$

The meaning of the matrices $\tilde{B}$, $\tilde{C}$ and the vector $\tilde{D}$ will become clear from further presentations.

We take the first column of the matrix BZ, find in it the maximal element as a model, and place the row in which it is found in the first position of the matrix BZ. We divide each element of this row by its first element. Now, subtracting the first from the other rows, after multiplying by an appropriate factor, we obtain zeros in the first column of these rows. We find the maximal element in the second column among all rows except the first, and by means of this row we obtain zeros in the second column of all rows except the first. We conduct an analogous transformation, further, with the third column, etc. As a result the matrix BZ is reduced to the form

$$\widetilde{BZ} = \left\| \begin{matrix} E & P_n & Q_n & R_n \\ 0 & \tilde{B}_{n-1} & \tilde{C}_{n-1} & \tilde{D}_{n-1} \end{matrix} \right\| ,$$

where E is the identity matrix.

For n = N, we have

$$BZ = \left\| \begin{matrix} A_N & B_N & C_N & D_N \\ A_{N-1} & B_{N-1} & C_{N-1} & D_{N-1} \end{matrix} \right\| .$$

and the passage from N to N-1 is included in the standard scheme /11 of the passage from n to n-1.

For n = 1, before the standard scheme of passage is applied, A_1 is eliminated by means of the identity matrix obtained for n = 2. In other words, the matrix

$$\left\| A_1 \; B_1 \; C_1 \; \mathcal{D}_1 \right\|$$

by means of the matrix

$$\left\| E \; P_3 \; Q_3 \; R_3 \right\|$$

reduces to the form

$$\left\| 0 \; B_1' \; C_1' \; \mathcal{D}_1' \right\|$$

Then the standard scheme is applied to the matrix

$$BZ' = \left\| \begin{matrix} \tilde{B}_2 & \tilde{C}_2 & 0 & \mathcal{D}_2 \\ B_1' & C_1' & 0 & \mathcal{D}_1' \end{matrix} \right\| .$$

As a result, we obtain

$$\widetilde{BZ}' = \left\| \begin{matrix} E & P_2 & 0 & R_2 \\ 0 & \tilde{C}_1' & 0 & \tilde{\mathcal{D}}_1' \end{matrix} \right\| ,$$

from which

$$F_1 = (\tilde{C}_1')^{-1} \cdot \tilde{\mathcal{D}}_1' ,$$

and then

$$F_2 = R_2 - P_2 \cdot F_1 .$$

And finally the recurrence formula for finding the solution has the form

$$F_n = R_n - P_n \cdot F_{n-1} - Q_n \cdot F_{n-2} , \quad n = 3, \ldots, N .$$

Thus we find an approximate solution to the boundary value problem (1-4) at the moment of time t_j in S+1 iterations. If the iterations correspond to a given precision, then we can continue iterating equation (1) in time.

2. DESCRIPTION OF THE PROGRAM

The program permits us to find a nonstationary solution $y_i(t,x)$ and a steady-state solution $y_i(x)$. The nonstationary solution can be found by /12 integrating equation (1) with a constant step in t or with an automatic choice of step. In the case of an "automatic choice of step", the time step is chosen depending on the number of iterations with respect to nonlinearity at the given step. Thus, the controls on precision do not depend on time.

The steady-state solution can be found by the method of settling down, i.e. we integrate in time up to the moment of time exceeding the maximal characteristic time of the problem (better with an automatic choice of step), or by iterations without settling down in time. In the latter case, we can solve directly a system of stationary equations, i.e. equations (1) without time derivatives.

Into the standard part of the program, we introduce three subroutines: HOK, OUT, and ISCL.

The subroutine HOK controls the iterations, the choice of step in time, the printing of the result, etc.

The subroutine OUT effects the printing of the result.

In the subroutine ISCL we calculate the coefficients for the unknowns in a linear system of difference equations, and this system is solved by Gauss" method.

The user writes the subroutine FF, where the difference equations are written down, and the basic program, where values are given to the physical constants and from which originates the access to the standard part of the program.

The access to the subroutine has the form:

```
CALL HOK(N,M,IB,ITAU,ITER,ITB,EPS,TKOH,TAU,YØ,XØ,X1),
```

Here N is the number of partition points of the x-interval.

M is the number of equations.

IB is the number of steps TAU in time, across which the printout of of the solution is produced. When it is necessary to print out the results at fixed moments of time t_n, we need to sequentially access the subroutine HOK, giving TKOH = t_n. /13

ITAU = 0 is the signal of a calculation with constant step TAU in time.

ITAU ≠ 0 is the signal for a calculation with automatic choice of time step. The step is increased by 1.5 times when the number of iterations at the previous step equals 1 and decreases by 2 times when it is greater than 3. Thus, there is no linear relationship between the choice of step and the accuracy of the solution in time. When we need to obtain a steady-state solution by the method of settling down, then we should set ITAU $\neq 0$.

ITER is the maximum allowable number of iterations at the step TAU; when the number of iterations IT exceeds ITER, there occurs a decrease in the step TAU by two times.

ITB = 1 is the signal to print the solution at each iteration.

EPS is the relative accuracy of the convergence of the iterations. The magnitude of the increment $\Delta y_{i,n}$ in the numerical differentiation is given by the formula

$$\Delta y_{i,n} = EPS/4 \cdot (|y_{i,n}| + |y_{i,n+1}| + |y_{i,n-1}| + |Y\emptyset_{i,n}|).$$

When $\Delta y_{i,n} = 0$ (for example, when the initial approximation $Y\emptyset \equiv 0$) $y_{i,n} = EPS$ (in a program with double precision $\Delta y_{i,n} = (EPS)^2$).

TKOH is the end of the interval of integration in time; when TAU > TKOH/10, the step TAU does not increase; when TAU < TKOH/10^5, the calculation ceases and the statement "THE TAU IS SMALLER THAN ALLOWED" is printed, and the values of the current moment of time and TAU are given.

TAU is the value of the time step: the initial step or constant step, depending on the value of the parameter ITAU. When TAU = 0, we seek a steady-state solution of the system directly by iterations 14 without a settling-down in time. In this case, when the number of iterations IT > ITER, the statement "IT = ITER ITERATIONS DO NOT CONVERGE WITH THE GIVEN ACCURACY EPS" is printed and the values of the next two iterations are given.

Y∅ is the dimensionality file N·M of the initial conditions; for the output of the subroutine HOK, Y∅ is a solution for t = TKOH or in case TAU = 0 — a steady-state solution.

X∅ is the left endpoint of the interval of x's.

X1 is the right endpoint of the interval of x's.

The subroutine HOK works with the subroutines ISCL, OUT, FF, and the library subroutine ABS.

ISCL is the subroutine for solving the system of linear algebraic equations. An accidental result of the form "MATRIX IS DEGENERATE............." is possible. Such a result, as a rule, indicates an improper formulation of the problem (an incorrect assignment of initial or boundary conditions) or errors in the subroutine FF (improperly written equations, errors in the coefficients, etc.)

OUT is the subroutine of printing the solution. The value of the current moment of time is printed BP = ...; the column of x values and the columns of values of the unknown functions y_k are also printed.

FF is the subroutine which is written by the user. In it are given the formulas for computing the right-hand side of f_k and also of ψ_K and φ_K. In this subroutine we must describe the general block HIF:

```
COMMON/HIF/BP,H,TAU1,XH(21),F(10),
 YØ(21,10),Y(21,10),Y1(21,10)
```
,

where BP is the moment of time t_j; H is the coordinate step; TAU1 is the reciprocal value of the time step TAU; XH is the file of x_n for n== 1,...,N; YØ is the value of the function at level t_{j-1}; and Y is the value of the function at level t_j of the previous iteration. 15

Access to the subroutine FF originates from the standard part by means of the operator CALL FF(N,NN), where N is the current index of the point x_n, NN is the collection of partition points x_n $1 \leq N \leq NN$. Correspondingly, the first operator of the subroutine FF must have the form: SUBROUTINE FF(N,NN), where N, NN are formal parameters having the above-mentioned meanings. The connection between the subroutine FF and the basic program can be effected through additional general blocks (also including unmarked ones).

On the basis of the data, we calculate the of F of dimension M, the elements of which are given by the file numerical values of expressions representing themselves or either difference approximations of the right-hand sides of the equations at the point x_n, or a difference approximation of the boundary conditions.

The variables entering into the general block HIF cannot be used in the subroutine FF as identifiers of other variables.

To economize machine time, it follows that we should avoid in the subroutine FF calculations of expressions not depending on j and n, because to calculate the Jacobian matrix at each iteration at each point x_n is the result of M+1 accesses to the subroutine FF. Furthermore, if these expressions enter the subroutine FF, then they also are computed M+1 times.

In the operators of description COMMON and DIMENSION, the files of the variables are written so that we can use the program when N ≤ 21 and M ≤ 10. When it is necessary to increase N or M, it follows that we should insert corresponding changes into these operators. In different programs the operators have the following form: /16

1) COMMON/HIF/ BP, H, TAU1, XH(N), F(M), YØ(N,M), Y(N,M), Y1(N,M)

in the subroutines HOK, ISCL, FF and OUT;

2) DIMENSION YØ(N,M) in the basic program;

3) DIMENSION YØØ(N,M) in the subroutine HOK;

4) DIMENSION BZ(2*M,3*M+1), PZ(N-1,M,M), QZ(N-2,M,M), CZ(2), F1(M)

in the subroutine ISCL.

Here the variables N and M and expressions containing them need to be replaced by appropriate numbers because the sizes of the dimensions of the files are given by integer constants without sign.

For example, if N = 51 and M = 5, the the operator DIMENSION in the subroutine ISCL needs to be replaced by the operator

DIMENSION BZ(10,16), PZ(50,5,5), QZ(49,5,5), CZ(2), F1(5) .

We need to make analogous changes in operators 1-3. After we have made these changes, we can use the program for arbitrary N ≤ 51 and M ≤ 5.

The necessity of changing the operators of description COMMON and DIMENSION for increases in N or M creates definite inconveniences in the work and is related to the limitations of the programming language Fortran, in which, in contrast to programming in Algol, we cannot use dynamic files. To eliminate this deficiency and also to increase the economy of the program, we should transfer from multidimensional files, to one-dimensional ones, as for example is done in the IBM library of standard programs {6}. The authors hope to return to this question at some time in the immediate future. /17

Example

Let there be given a system of linear differential equations of parabolic type:

$$\frac{\partial y_1}{\partial t} = \frac{\partial^2 y_1}{\partial x^2} + y_2 ,$$
$$\frac{\partial y_2}{\partial t} = \frac{\partial^2 y_2}{\partial x^2} , \tag{20}$$

where $0.5 \leq x \leq 1$, $t \geq 0$.

We put on this system of equations the following boundary and initial conditions:

$$\frac{\partial y_1(t,0{,}5)}{\partial x} = 0 , \quad y_1(t,1) = 0 ,$$
$$\frac{\partial y_2(t,0{,}5)}{\partial x} = 0 , \quad y_2(t,1) = 0 , \tag{21}$$
$$y_1(0,x) = 0 , \quad y_2(0,x) = \sin(\pi x)$$

The analytic solution to the problem formulated has the form:

$$y_1(t,x) = t \cdot e^{-\pi^2 t} \sin(\pi x) ,$$
$$y_2(t,x) = e^{-\pi^2 t} \sin(\pi x) . \tag{22}$$

We choose a uniform grid in x and t:

$$x_n = (n-1)\cdot \Delta x + 0{,}5 ; \quad n = 1, \ldots, 21 ; \quad \Delta x = 0{,}025 ;$$
$$t_j = j \cdot \Delta t ; \quad j = 0, \ldots, 100 ; \quad \Delta t = 0{,}001 .$$

The differential equations (20) may be approximated by the difference relations: /18

$$\frac{y^j_{1,n}-y^{j-1}_{1,n}}{\Delta t}=0,55\cdot\left(\frac{y^j_{1,n+1}-2y^j_{1,n}+y^j_{1,n-1}}{\Delta x^2}+y^j_{2,n}\right)+0,45\cdot\left(\frac{y^{j-1}_{1,n+1}-2y^{j-1}_{1,n}+y^{j-1}_{1,n-1}}{\Delta x^2}+y^{j-1}_{2,n}\right);\qquad(23)$$

$$\frac{y^j_{2,n}-y^{j-1}_{2,n}}{\Delta t}=0,55\,\frac{y^j_{2,n+1}-2y^j_{2,n}+y^j_{2,n-1}}{\Delta x^2}+0,45\,\frac{y^{j-1}_{2,n+1}-2y^{j-1}_{2,n}+y^{j-1}_{2,n-1}}{\Delta x^2}.$$

The difference expressions for the boundary and initial conditions (21) have the form

$$\frac{-3y^j_{1,1}+4y^j_{1,2}-y^j_{1,3}}{2\Delta x}=0,\quad y^j_{1,N}=0,$$

$$\frac{-3y^j_{2,1}+4y^j_{2,2}-y^j_{2,3}}{2\Delta x}=0,\quad y^j_{2,N}=0,\qquad(24)$$

$$y^0_{1,n}=0,\qquad y^0_{2,n}=\sin(\pi\cdot x_n),$$

where j = 0,..., 100, n = 1,..., 21.

When we accessed the standard program HOK, the following actual parameters were given:

N = 2I, M = 2, IB = 50, ITAU = 0, ITER = 7, ITB = 0,
EPS = 0,00I, TKOH = 0,I, TAU = 0,00I, XØ = 0,5, X1 = I.

In the subroutine FF, the right-hand sides of equations (23) and the boundary conditions (24) are written down. In the appendix we present the the text of the subroutine FF, where H = Δx, H2 = $(\Delta x)^2$, Y(K,N) = $y^j_{k,n}$, YØ(K,n) = $y^{j-1}_{k,n}$, N = n. N = 1 corresponds to x = xØ, and N = NN corresponds to x = x1, $1 \leq N \leq NN$.

The initial conditions are assigned in the basic program (see the text of the basic program in the appendix).

In the table we cite the approximate and exact solutions at the moment of time t = 0.1. /19

x Coordinate	Solution y_1		Solution y_2	
	Approximate	Exact	Approximate	Exact
0,50000	0,037264	0,03727I	0,37300	0,3727I
0,52500	0,037I49	0,037I56	0,37I86	0,37I56
0,55000	0,036805	0,0368I2	0,36842	0,368I2
0,57500	0,036235	0,03624I	0,3627I	0,3624I
0,60000	0,03544I	0,035447	0,35476	0,35447
0,62500	0,034428	0,034434	0,34462	0,34434
0,65000	0,033203	0,033209	0,33236	0,33209
0,67500	0,03I773	0,03I779	0,3I805	0,3I779
0,70000	0,030I48	0,030I53	0,30I78	0,30I53
0,72500	0,028396	0,02834I	0,28365	0,2834I
0,75000	0,026350	0,026355	0,26376	0,26355
0,77500	0,024202	0,024206	0,24226	0,24206
0,80000	0,02I904	0,02I907	0,2I926	0,2I907
0,82500	0,0I947I	0,0I9474	0,I9490	0,I9474
0,85000	0,0I69I8	0,0I692I	0,I6935	0,I692I
0,87500	0,0I426I	0,0I4263	0,I4275	0,I4263
0,90000	0,0II5I5	0,0II5I7	0,II527	0,II5I7
0,92500	0,008699	0,00870I	0,08708	0,0870I
0,95000	0,005829	0,00583I	0,05835	0,0583I
0,97500	0,002924	0,002924	0,02927	0,02924
I,00000	0,0	0,0	0,0	0,0

Before solving complicated problems by the standard program described here, we recommend to the user that he solve one or more simple problems, the solutions are which are well-known. /20

CONCLUSION

The method described here was shown to be effective in solving stiff systems of differential equations for modeling the chemical composition of the atmosphere of Mars.[7]. System (II) of equations of parabolic type was solved. An implicit conservative difference scheme of second-order accuracy in Δx was used. Steady-state and nonstationary solutions were found. The characteristic times of chemical decay for different components in this problem varied between the limits 10^{-8} to 10^{12} sec., while the characteristic times for the establishment of diffusion equilibrium varied depending on height within the limits 10^{2} to 10^{7} sec. In calculations with usual accuracy on electronic computers of type YeS and BESM-6, we obtain larger roundoff errors for time steps Δt of 10^{5} to 10^{7} sec. These errors can be explained by poor conditioning on the system of equations (19), which is a consequence of the great difference in characteristic times of the problem. It turns out that a steady-state solution is obtained only when calculating to double precision. In this case there are no restrictions on the time step Δt*); because the corresponding linear problem (19), which is obtained as a result of linearizing the nonlinear difference equations, is stable. We consider the steady-state solution to be obtained when t $\sim 10^{16}$ sec. (the maximal characteristic time of the problem $\sim 10^{12}$ sec.).

Using the given method to solve the described system of equations permits us to obtain a series of new results on the chemical composition of the atmosphere of Mars. /21

---

*) For an abrupt change in the solution at a given time Δt, the emergence of negative concentrations and instablity is possible. In this case, the iterations do not converge, and the program automatically changes the time step until the concentrations become positive.

The proposed method is adaptable to the solution of problems of nonstationary gas dynamics. For example, to solve two-dimensional problems of the atmospheric heat circulation of planets [8,9,10], an implicit scheme was used in which the angular derivatives were taken with preceding iterations. However, to solve the one-dimensional boundary problems obtained from it along vertical lines, a variant of the proposed method in which the Jacobian matrix was calculated analytically was used. This method is also adaptable to the calculation of thermal fields in two-dimensional structures and other problems.

In conclusion, we enumerate the basic merits and defects of the proposed method, and we give recommendations about its use. Regarding the merits of the method, we can list

1. The possiblity of using the program for solving a wide variety of physical problems, among which are those having a large range of characteristic tumes.
2. Rapid convergence of the iterations for nonlinearity, owing to the use of Newton's method.
3. The possibility of easily constructing a conservative difference scheme.
4. The automatic linearization of nonlinear difference equations by means of numerical differentiation, which significantly reduces the work of programming.

It follows that we should consider as an essential defect of the method the fact that the calculational time (the number of arithmetic operations) in solving a system of M equations is proportional to M^3. It follows that we should stress that this defect is related to the use of Gauss' method, i.e. it turns out to be closely interrelated with the merits of our method.

Owing to, on the one hand, the stability and rapidity of convergence of the iterations, and on the other hand the reduction to a minimum of the programming work of the described method, it can find wide application in scientific and engineering computations. We note that more economical methods, in which the calculational time is proportional to M^2, for example, often possess slower rates of convergence of the iterations and, as a result, can prove less economical. /22

ORIGINAL PAGE IS
OF POOR QUALITY

Recommendations for Use

1. Linear equations.

a. Stationary equations.

To solve such problems, it is necessary to solve only once a corresponding system of difference equations, i.e. we need one iteration. For that one iteration to be made, it follows that we should set ITER = 1. However, by convention, it is useful to set ITER > 1 (for example, ITER = 5) and ITB = 1. Then when there are no errors, one superfluous iteration will be made. If, however, there is an error, then with a significant probability, the iterations will not converge. The result of each iteration will be presented in print.

b. Nonstationary equations.

Because there is in the program no control over the time precision, then in order to guarantee it, it follows that we should solve the problem several times with different constant time steps $\boxed{\Delta t}$ and should equalize the results obtained.

2. Nonlinear equations.

a. Stationary equations.

If the equations are strongly nonlinear and the form of the solution is unknown beforehand, then it is necessary to solve the problem by the method of bisection. In other words, it is necessary to give an arbitrary initial approximation and time step $\boxed{\Delta t}$, which must be smaller than the minimal charact- /23
erisitic time of the problem. We need to undertake the calculation with an automatic choice of step up to a moment of time larger than the maximal characteristic time of the problem.

b. Nonstationary equations.

Here the same remarks as those in item "b" for linear equations are valid.

We can use the method to solve multidimensional problems by methods of coordinate-wise decomposition.

In the case of solving systems of equations having a large difference in characteristic times for different functions, i.e. stiff systems, we recommend the use of a program with double precision, the text of which can be found in the appendix. It follows, in view of this, that such a program will occupy twice as much space in the machine's memory. On an electronic computer of type YeS, the calculation time for such a program cannot be practically increased, but on a BESM-6, the space is increased five-fold. On the BESM-6, a number with ordinary precision has 12 decimal places, but on a computer of type YeS only 7. Therefore, for calculations on a computer of type YeS, we always recommend the use of double precision, because, on account of the restricted number of spaces, round-off errors which are too large can occur in numerical differentiation and in solving a system of algebraic equations.

To solve systems of differential equations with derivatives no higher than first order as part of the boundary conditions, we can use difference equations written on one of the boundaries. In this case, the difference analogue of the time derivative is written down in a corresponding expression for φ or ψ (see formulas (11),(12)).

If an equation not containing a time derivative enters the system, then it follows that we should multiply this equation by a sufficiently small number.

We recall that the choice of a difference scheme belongs to the user. We have only the following requirements for it: /24

a) a scheme of ordered pairs (t_{j-1},t_j) in time,

b) in the coordinate x, we establish at each partition point x_n not more than three values of each of the unknown functions $y^j_{k,n-1}$, $y^j_{k,n}$, $y^j_{k,n+1}$.

Programs with ordinary and double precision were prepared for electronic computers of type YeS and BESM-6 and can be used on them without any changes.

REFERENCES

1. Proceedings of the All-Union Symposium "Mathematical Problems in Chemistry," (Trydy vsesoyuznogo simpoziuma "matematicheskiye problemy khimii), Computational Center of the Siberian Branch of the Academy of Sciences of the USSR, Novosibirsk, 1973.

2. Proceedings of the All-Union Symposium "Mathematical Problems in Chemistry," (Trudy vsesoyuznogo simpoziuma "matematicheskiye problemy khimii), Computational Center of the Siberian Branch of the Academy of Sciences of the USSR, Novosibirsk, 1975.

3. J. T. Hasting and R. G. Roble. Planetary and Space Science, 1977, 25, 209.

4. R. D. Richtmyer and K. W. Morton. Difference Methods for Initial Value Problems, Mir Press, 1972. (In Russian translation)

5. G. E. Forsythe and C. Moler. Computer Solution of Linear Algebraic Systems, Mir Press, 1969. (In Russian translation)

6. Sbornik Nauchnykh Program Na Fortrane. Collection of Scientific Programs in Fortran, No. 2, Moscow, Statistika

7. M. N. Izakov and O. P. Krasitskiy. Kosmicheskiye Issledovaniya, 1977, 15, 455.

8. M. N. Izakov, S. K. Morozov, and E. E. Shnol'. Preprint of the Institute of Cosmic Research of the Academy of Sciences of the USSR, 1972, Pr.-115.

9. M. N. Izadov and S. K. Morozov. Kosmicheskiye Issledovaniya, 1975, 13, 404.

10. M. N. Izakov and S. K. Morozov. Kosmicheskiye Issledovaniya, 1976, 14, 476.

APPENDIX

Program with Ordinary Accuracy

Basic Program

```
      DIMENSION Y0(21,10)
      ITB=0
      ITER=7
      ITAD=0
      IB=50
      TAU=.001
      TKOH=0.1
      EPS=0.001
      X0=0.5
      X1=1.
      M=2
      N=21
      DO 2 I=1,N
      Y0(I,2)=SIN(3.14159*(X0+(X1-X0)*FLOAT(I-1)/FLOAT(N-1)))
    2 Y0(I,1)=0
      CALL HOK(N,M,IB,ITAU,ITER,ITB,EPS,TKOH,TAU,Y0,X0,X1)
C   Calculation of the solution of a control example at moment TKOH
      Y0(I,4)=EXP(-9.8696*TKOH)*SIN(3.14159*(X0+(X1-X0)*
     *FLOAT(I-1)/FLOAT(N-1)))
   11 Y0(I,3)=TKOH*Y0(I,4)
      PRINT 8
    8 FORMAT (32H  comparison of the obtained and exact
     *20H solutions for BP = TKOH)
    1 FORMAT(/)
      DO 10 I=1,N
   10 PRINT 9,(Y0(I,J),J=1,4)
    9 FORMAT (4E15.5)
      END
```

Subroutine FF

```
      SUBROUTINE FF(N,NN)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
      H2=H*H
      IF(N.NE.NN) GOTO 1
C   Boundary conditions on the right end
      F(1)=Y(NN,1)
      F(2)=Y(NN,2)
      GOTO 3
    1 IF(N.NE.1) GOTO 2
C   Boundary conditions on the left end
      F(1)=-3.*Y(1,1)+4.*Y(2,1)-Y(3,1)
      F(2)=-3.*Y(1,2)+4.*Y(2,2)-Y(3,2)
C   Right-hand sides of equations
    2 CONTINUE
      F(1)=((Y(N+1,1)-2.*Y(N,1)+Y(N-1,1))/H2+Y(N,2))
     **0.55+((Y0(N+1,1)-2.*Y0(N,1)+Y0(N-1,1))/H2+Y0(N,2))*0.45
      F(2)=(Y(N+1,2)-2.*Y(N,2)+Y(N-1,2))/H2
     **0.55+(Y0(N+1,2)-2.*Y0(N,2)+Y0(N-1,2))/H2*0.45
    3 CONTINUE
      RETURN
      END
```

Subroutine HOK

```
      SUBROUTINE HOK(N,M,IB,ITAU,ITER,ITB,EPS,TKOH,TAU,Y00,X0,X1)
      DIMENSION Y00(21,10)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
    1 FORMAT(/)
      PRINT 1
    2 FORMAT(3H N=,I3,3H M=,I3,4H IB=,I3,6H ITAU=,I3,
     16H ITER=,I3,5H ITB=,I3)
      PRINT2,N,M,IB,ITAU,ITER,ITB
      PRINT 1
    3 FORMAT(5H EPS=,E12.4,6H TKOH=,E12.4,5H TAU=,E12.4,
     14H X0=,E12.4,4H X1=,E12.4)
      PRINT3,EPS,TKOH,TAU,X0,X1
      PRINT 1
      JB=0
      BP=0
      H=(X1-X0)/(N-1)
      XH(1)=X0
      DO 24 I=2,N
   24 XH(I)=XH(I-1)+H
      IF(TAU.NE.0) TAU1=1./TAU
      IF(TAU.EQ.0) TAU1=0.
      DO 29 I=1,N
      DO 29 J=1,M
   23 Y0(I,J)=Y00(I,J)
      Y1(I,J)=Y0(I,J)
   29 Y(I,J)=Y0(I,J)
      CALL OUT(M,N)
   30 IT=-1
      BP=BP+TAU
      IF(BP.GT.TKOH*1.00001) GOTO 72
   31 IT=IT+1
      CALL ISCL(M,N,EPS/4.5
      IF(IT
C            One compulsory iteration
   32 IF(IT.EQ.ITER) GOTO 50
      DO 28 I=1,N
      DO 28 J=1,M
   28 Y(I,J)=Y1(I,J)
      IF(ITB.EQ.1) GOTO 55
      GOTO
C            Comparing the convergence of iterations
   33 DO 27 I=1,N
      DO 27 J=1,M
      IF(ABS(Y(I,J)).LE.1.E-10) GOTO 27
      IF(ABS((Y(I,J)-Y1(I,J))/Y(I,J))-EPS) 27,27,32
   27 CONTINUE
      IF(TAU.EQ.0) GOTO 57
C            Preparation for the next step
      DO 26 I=1,N
      DO 26 J=1,M
      Y0(I,J)=Y1(I,J)
   26 Y(I,J)=Y1(I,J)
      IF(ABS(BP-TKOH).LE.(TKOH*1.E-5)) GOTO 58
   25 JB=JB+1
      IF(ITAU.EQ.0) GOTO 43
      IF(IT.EQ.1.AND.TAU.LT.TKOH/10.) GOTO 41
C            Increase in step by TAU
      IF(IT
C            Decrease in step by TAU
   43 IF(IB.NE.JB) GOTO 30
C            Delivery of a result with period IB
      JB=0
      CALL OUT(M,N)
      GOTO 30
   72 TAU=TKOH-BP+TAU
      BP=TKOH
      TAU1=1./TAU
      GOTO 31
   41 TAU=1.5*TAU
      TAU1=1./TAU
      GOTO 43
   42 TAU=TAU/2.
```

```
      TAU1=1./TAU
      IF(TAU-TKON/100000.) 44,43,43
   44 PRINT 1
      [illegible]
    8 FORMAT (34H Step TAU is smaller than the permitted time =,
      [illegible]
      STOP
   50 IF(TAU.EQ.0) GOTO 52
      DO 51 I=1,N
      DO 51 J=1,M
   51 Y(I,J)=Y0(I,J)
      GOTO 42
   52 [illegible]
    9 FORMAT (39H IT = ITER Iterations do not converge with accuracy EPS)
      IT=ITER-1
      PRINT 56,ITER
      CALL OUT(M,N)
      PRINT 56,IT
      DO 53 I=1,N
      DO 53 J=1,M
   53 Y1(I,J)=Y(I,J)
      CALL OUT(M,N)
      STOP
   55 [illegible]
   56 FORMAT (10H Iteration =,     )
      CALL OUT(M,N)
      GOTO 31
   57 PRINT 56,IT
   58 CONTINUE
      CALL OUT(M,N)
      DO 59 I=1,N
      DO 59 J=1,M
   59 Y00(I,J)=Y1(I,J)
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Subroutine ISCL

```
      SUBROUTINE ISCL(MM,NN,EPS)
      DIMENSION BZ(20,31),PZ(20,10,10),QZ(19,10,10),CZ(2),F1(10)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
      MZ=MM
      NZ=NN-1
      MZ11=MZ+1
      MZ20=2*MZ
      MZ21=2*MZ+1
      MZ30=3*MZ
      MZ31=3*MZ+1
      N=NZ-1
      GOTO 103
C     Cycle in N
  102 IF(N)110,104,105
  110 DO 111 I=MZ11,MZ20
      DO 111 J=1,MZ31
  111 BZ(I,J)=0
  101 DO 112 L=1,MZ
      [illegible]
C     Search for the Pivot Element
      DO 113 I=L,MZ20
      IF(ABS(BZ(I,L)).LT.ABS(CZ(1))) GO TO 113
      K=I
      CZ(1)=BZ(I,L)
  113 CONTINUE
C     Pivot element is close to zero
      [illegible]
C     Rearrangement of the rows and elimination
      DO 114 J=L,MZ31
      BZ(K,J)=BZ(K,J)/CZ(1)
      IF(K.EQ.L) GO TO 114
      CZ(2)=BZ(K,J)
      BZ(K,J)=BZ(L,J)
      BZ(L,J)=CZ(2)
  114 CONTINUE
```

```
      DO 115 I=1,MZ20
      IF(I.EQ.L) GO TO 115
      CZ(I)=BZ(I,L)
      L1=L+1
      DO 116 J=L1,MZ31
  116 BZ(I,J)=BZ(I,J)-BZ(L,J)*CZ(I)
  115 CONTINUE
  112 CONTINUE
C     Formation of the matrices P, Q, R
      DO 118 I=1,MZ
      Y1(N+2,I)=BZ(I,MZ31)
      IF(N.EQ.-1) GOTO 118
      DO 117 J=1,MZ
      PZ(N+1,I,J)=BZ(I,J+MZ)
      IF(N.EQ.0) GOTO 117
      QZ(N,I,J)=BZ(I,J+MZ20)
  117 CONTINUE
  118 CONTINUE
      IF(N.EQ.-1) GOTO 106
      DO 120 I=1,MZ
      BZ(I,MZ31)=BZ(I+MZ,MZ31)
      DO 121 J=1,MZ20
  121 BZ(I,J)=BZ(I+MZ,J+MZ)
      DO 120 J=MZ21,MZ30
  120 BZ(I,J)=0
      N=N-1
C     End of the cycle in N
      GOTO 105
C     Receipt of the solution
  106 DO 122 J=1,MZ
      DO 122 I=1,MZ
  122 Y1(2,J)=Y1(2,J)-PZ(1,J,I)*Y1(1,I)
      DO 123 K=2,MZ
      DO 123 J=1,MZ
      DO 123 I=1,MZ
  123 Y1(K+1,J)=Y1(K+1,J)-PZ(K,J,I)*Y1(K,I)-QZ(K-1,J,I)*Y1(K-1,I)
C     Block for calculating coefficients at interior point of N
  105 CALL FF(N+1,NN)
      DO 51 K=1,MZ
   51 F1(K)=F(K)
      DO 52 I=1,MZ
      DY=(ABS(Y(N+2,I))+ABS(Y(N+3,I))+
     *ABS(Y(N+1,I))+ABS(Y0(N+2,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(N+2,I)=Y(N+2,I)+DY
      CALL FF(N+1,NN)
      Y(N+2,I)=Y(N+2,I)-DY
      DO 53 K=MZ11,MZ20
   53 BZ(K,I)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(ABS(Y(N+1,I))+ABS(Y(N+2,I))+
     *ABS(Y(N,I))+ABS(Y0(N+1,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(N+1,I)=Y(N+1,I)+DY
      CALL FF(N+1,NN)
      Y(N+1,I)=Y(N+1,I)-DY
      DO 54 K=MZ11,MZ20
   54 BZ(K,I+MZ)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(ABS(Y(N,I))+ABS(Y(N+1,I))+
     *ABS(Y(N-1,I))+ABS(Y0(N,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(N,I)=Y(N,I)+DY
      CALL FF(N+1,NN)
      Y(N,I)=Y(N,I)-DY
      DO 52 K=MZ11,MZ20
   52 BZ(K,I+MZ20)=(F1(K-MZ)-F(K-MZ))/DY
      DO 55 K=MZ11,MZ20
      BZ(K,MZ31)=F1(K-MZ)
      DO 55 I=1,MZ
   55 BZ(K,MZ31)=BZ(K,MZ31)+BZ(K,I)*Y(N+2,I)
     1+BZ(K,I+MZ)*Y(N+1,I)+BZ(K,I+MZ20)*Y(N,I)
      DO 56 K=MZ11,MZ20
      BZ(K,K)=BZ(K,K)+TAU1
   56 BZ(K,MZ31)=BZ(K,MZ31)+Y0(N+1,K-MZ)*TAU1
C     Normalization of the row
      DO 8[illegible] I=MZ11,MZ20
      CZ(1)=1
      DO 53 J=1,MZ30
      IF(ABS(BZ(I,J)).LT.ABS(CZ(1))) GOTO 5[illegible]
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      CZ(1)=BZ(I,J)
   58 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 57 J=1,MZ31
   57 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO 101
C        Block for calculating coefficients at the right end
  103 CALL FF(NN,NN)
      DO 31 K=1,MZ
   31 F1(K)=F(K)
      DO 32 I=1,MZ
      DY=(ABS(Y(NN,I))+ABS(Y(NN-1,I))+
     *ABS(Y(NN-2,I))+ABS(Y0(NN,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(NN,I)=Y(NN,I)+DY
      CALL FF(NN,NN)
      Y(NN,I)=Y(NN,I)-DY
      DO 33 K=1,MZ
   33 BZ(K,I)=(F1(K)-F(K))/DY
      DY=(ABS(Y(NN-1,I))+ABS(Y(NN,I))+
     *ABS(Y(NN-2,I))+ABS(Y0(NN-1,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(NN-1,I)=Y(NN-1,I)+DY
      CALL FF(NN,NN)
      Y(NN-1,I)=Y(NN-1,I)-DY
      DO 34 K=1,MZ
   34 BZ(K,I+MZ)=(F1(K)-F(K))/DY
      DY=(ABS(Y(NN-2,I))+ABS(Y(NN-1,I))+
     *ABS(Y(NN-3,I))+ABS(Y(NN-2,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(NN-2,I)=Y(NN-2,I)+DY
      CALL FF(NN,NN)
      Y(NN-2,I)=Y(NN-2,I)-DY
      DO 32 K=1,MZ
   32 BZ(K,I+MZ20)=(F1(K)-F(K))/DY
      DO 35 K=1,MZ
      BZ(K,MZ31)=F1(K)
      DO 35 I=1,MZ
   35 BZ(K,MZ31)=BZ(K,MZ31)+BZ(K,I)*Y(NN,I)
     1+BZ(K,I+MZ)*Y(NN-1,I)+BZ(K,I+MZ20)*Y(NN-2,I)
C        Normalization of the row
      DO 37 I=1,MZ
      CZ(1)=0
      DO 38 J=1,MZ30
      IF(ABS(BZ(I,J)).LT.ABS(CZ(1))) GOTO 38
      CZ(1)=BZ(I,J)
   38 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 37 J=1,MZ31
   37 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO 102
C        Block for calculatimg coefficients at the left end
  104 CALL FF(1,NN)
      DO 41 K=1,MZ
   41 F1(K)=F(K)
      DO 44 I=1,MZ
      DY=(ABS(Y(3,I))+ABS(Y(4,I))+
     *ABS(Y(2,I))+ABS(Y0(3,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(3,I)=Y(3,I)+DY
      CALL FF(1,NN)
      Y(3,I)=Y(3,I)-DY
      DO 43 K=MZ11,MZ20
   43 BZ(K,I)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(ABS(Y(2,I))+ABS(Y(3,I))+
     *ABS(Y(1,I))+ABS(Y0(2,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(2,I)=Y(2,I)+DY
      CALL FF(1,NN)
      Y(2,I)=Y(2,I)-DY
      DO 42 K=MZ11,MZ20
   42 BZ(K,I+MZ)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(ABS(Y(1,I))+ABS(Y(2,I))+
     *ABS(Y(3,I))+ABS(Y0(1,I)))*EPS
      IF(DY.EQ.0) DY=EPS
      Y(1,I)=Y(1,I)+DY
      CALL FF(1,NN)
      Y(1,I)=Y(1,I)-DY
      DO 44 K=MZ11,MZ20
   44 BZ(K,I+MZ20)=(F1(K-MZ)-F(K-MZ))/DY
      DO 45 K=MZ11,MZ20
      BZ(K,MZ31)=F1(K-MZ)
      DO 45 I=1,MZ
   45 BZ(K,MZ31)=BZ(K,MZ31)+BZ(K,I)*Y(3,I)+
     1BZ(K,I+MZ)*Y(2,I)+BZ(K,I+MZ20)*Y(1,I)
C      Elimination of A(0)
      DO 60 K=1,MZ
      DO 60 J=MZ11,MZ20
      IF(BZ(J,K).EQ.0) GOTO 60
      DO 61 I=1,MZ
      BZ(J,I+MZ)=BZ(J,I+MZ)-PZ(2,K,I)*BZ(J,K)
   61 BZ(J,I+MZ20)=BZ(J,I+MZ20)-QZ(1,K,I)*BZ(J,K)
      BZ(J,MZ31)=BZ(J,MZ31)-Y1(3,K)*BZ(J,K)
   60 CONTINUE
      DO 62 I=1,MZ
      DO 62 J=MZ11,MZ20
      BZ(J,I)=BZ(J,I+MZ)
      BZ(J,I+MZ)=BZ(J,I+MZ20)
   62 BZ(J,I+MZ20)=0.
C        Normalization of the row
      DO 47 I=MZ11,MZ20
      CZ(1)=0.
      DO 48 J=1,MZ20
      IF(ABS(BZ(I,J)).LT.ABS(CZ(1))) GOTO 48
      CZ(1)=BZ(I,J)
   48 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 47 J=1,MZ31
   47 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO 101
C        Failure in delivery
  100 PRINT 1,N,L,K
      DO 3 I=1,MZ20
    3 PRINT 2,(BZ(I,J),J=1,MZ31)
    1 FORMAT (/23H Matrix is singular: N =, I3,
     15H L=,I3,3H K=,I3,4H BZ=)
    2 FORMAT(10E12.4)
      STOP
      END
```

Subroutine OUT

```
      SUBROUTINE OUT(M,N)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
    1 FORMAT(//)
    2 FORMAT([illegible])
    9 FORMAT (10H          Time = ,E12,4)
      PRINT 9,BP
      PRINT 1
      DO 10 I=1,N
   10 PRINT 2,XH(I),(Y1(I,J),J=1,M)
      PRINT 1
      RETURN
      END
```

Program with Double Precision

Basic Program

```
      DOUBLE PRECISION YO,EPS,TKOH,TAU,X0,X1
      DIMENSION YO(21,10)
      ITB=0
      ITER=7
      ITAU=0
      IB=50
      TAU=.001
      TKOH=0.1
      EPS=0.001
      X0=0.5
      X1=1.
      M=2
      N=21
      DO 2 I=1,N
      YO(I,2)=DSIN(3.14159*(X0+(X1-X0)*FLOAT(I-1)/FLOAT(N-1)))
    2 YO(I,1)=0.
      CALL OKHOK([illegible]
C     Calculation of exact solution of control example at moment TKOH
      YO(I,4)=DEXP(-9.8696*TKOH)*DSIN(3.14159*(X0+(X1-X0)*
     *FLOAT(I-1)/FLOAT(N-1)))
   11 YO(I,3)=TKOH*YO(I,4)
      PRINT 8
    8 FORMAT (32H Comparison of obtained and exact
     * 20H  solutions at BP = TKOH)
    1 FORMAT(/)
      DO 10 I=1,N
   10 PRINT 9,(YO(I,J),J=1,4)
    9 FORMAT (4D15.5)
      END
```

Subroutine FF

```
      SUBROUTINE FF(N,NN)
      DOUBLE PRECISION BP,H,TAU1,XH,F,YO,Y,Y1
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),YO(21,10),Y(21,10),Y1(21,10)
      H2=H*H
C     Boundary conditions on the right end
      F(1)=Y(NN,1)
      F(2)=Y(NN,2)
      GOTO 3
    1 IF(N.NE.1) GOTO 2
C     Boundary conditions on the left end
      F(1)=-3.*Y(1,1)+4.*Y(2,1)-Y(3,1)
      F(2)=-3.*Y(1,2)+4.*Y(2,2)-Y(3,2)
      GOTO 3
C     Right-hand side of the equation
    2 CONTINUE
      F(1)=((Y(N+1,1)-2.*Y(N,1)+Y(N-1,1))/H2+Y(N,2))
     **0.55+((YO(N+1,1)-2.*YO(N,1)+YO(N-1,1))/H2+YO(N,2))*0.45
      F(2)=(Y(N+1,2)-2.*Y(N,2)+Y(N-1,2))/H2
     **0.5+(YO(N+1,2)-2.*YO(N,2)+YO(N-1,2))/H2*0.45
    3 CONTINUE
      RETURN
      END
```

Subroutine DHOK

```
      SUBROUTINE DHOK(N,M,IB,ITAU,ITER,ITB,EPS,TKOH,TAU,Y00,X0,X1)
      DOUBLE PRECISION Y00,EPS,TKOH,TAU,X0,X1
      DOUBLE PRECISION BP,H,TAU1,XH,F,Y0,Y,Y1
      DIMENSION Y00(21,10)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
    1 FORMAT(/)
      PRINT 1
    2 FORMAT(3H N=,I3,3H M=,I3,4H IB=,I3,6H ITAU=,I3,
     16H ITER=,I3,5H ITB=,I3)
      PRINT 2,N,M,IB,ITAU,ITER,ITB
      PRINT 1
    3 FORMAT(5H EPS=,D13.4,6H TKOH=,D13.4,5H TAU=,D13.4,
     14H X0=,D13.4,4H X1=,D13.4)
      PRINT 3,EPS,TKOH,TAU,X0,X1
      PRINT 1
      JB=0
      BP=0
      H=(X1-X0)/(N-1)
      XH(1)=X0
      DO 24 I=2,N
   24 XH(I)=XH(I-1)+H
      IF(TAU.NE.0) TAU1=1./TAU
      IF(TAU.EQ.0) TAU1=0.
      DO 29 I=1,N
      DO 29 J=1,M
   23 Y0(I,J)=Y00(I,J)
      Y1(I,J)=Y0(I,J)
   29 Y(I,J)=Y0(I,J)
      CALL DOUT(M,N)
   30 IT=-1
      BP=BP+TAU
      IF(BP.GT.TKOH+0.00001) GOTO 72
   31 IT=IT+1
      CALL DSCL(M,N,EPS/4.)
      IF([illegible]
C      One compulsory iteration
   32 IF(I[illegible]ER) GOTO 50
      DO 28 I=1,N
      DO 28 J=1,M
   28 Y(I,J)=Y1(I,J)
      IF(ITB.EQ.1) GOTO 55
      GOTO 31
C      Comparing the convergence of iterations
   33 DO 27 I=1,N
      DO 27 J=1,M
      IF(DABS(Y(I,J)).LE.1.D-20) GOTO 27
      IF(DABS((Y(I,J)-Y1(I,J))/Y(I,J))-EPS)27,27,32
   27 CONTINUE
      IF(TAU.EQ.0) GOTO 57
C      Preparation for the next step
      DO 26 I=1,N
      DO 26 J=1,M
      Y0(I,J)=Y1(I,J)
   26 Y(I,J)=Y1(I,J)
      IF(DABS(BP-TKOH).LE.0.00001*TKOH) GOTO 58
   25 JB=JB+1
      IF(ITAU.EQ.0) GOTO 43
      IF(IT.EQ.1.AND.TAU.LT.TKOH/[illegible]) GOTO 41
C      In crease in the step by TAU
      IF([illegible] GOTO 42
C      Decrease in the step by TAU
   43 IF(IB.NE.JB) GOTO 30
C      Delivery of the result with period IB
      JB=0
      CALL DOUT(M,N)
      GOTO 30
   72 TAU=TKOH-BP+TAU
      BP=TKOH
      TAU1=1./TAU
      GOTO 31
   41 TAU=1.5*TAU
```

```
      TAU1=1./TAU
      GOTO 43
42    TAU=TAU/2
      TAU1=1./TAU
      IF(TAU-TKON/100000.) 44,43,43
44    PRINT 8
8     FORMAT (34H Step TAU is smaller than the permitted time =,.
      PRINT 8,BP,TAU
      STOP
50    IF(TAU.EQ.0) GOTO 52
      DO 51 I=1,N
      DO 51 J=1,M
51    Y(I,J)=Y0(I,J)
      GOTO 42
52    PRINT 9
9     FORMAT (39H IT = ITER  Iterations do not converge with precision EPS)
      PRINT 56,ITER
      CALL DOUT(M,N)
      PRINT 56,IT
      DO 53 I=1,N
      DO 53 J=1,M
53    Y1(I,J)=Y(I,J)
      CALL DOUT(M,N)
      STOP
55    PRINT 56,IT
56    FORMAT (10H  iteration =, 13)
      CALL DOUT(M,N)
      GOTO 31
57    PRINT 56,IT
58    CONTINUE
      CALL DOUT(M,N)
      DO 59 I=1,N
      DO 59 J=1,M
59    Y00(I,J)=Y1(I,J)
      RETURN
      END
```

Subroutine DSCL

```
      SUBROUTINE DSCL(MM,NN,EPS)
      DOUBLE PRECISION BP,H,TAU1,XH,F,Y0,Y,Y1
      DOUBLE PRECISION BZ,PZ,QZ,CZ,F1,EPS
      DIMENSION BZ(20,31),PZ(20,10,10),QZ(19,10,10),CZ(2),F1(10)
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
      MZ=MM
      NZ=NN-1
      MZ11=MZ+1
      MZ20=2*MZ
      MZ21=2*MZ+1
      MZ30=3*MZ
      MZ31=3*MZ+1
      N=NZ-1
      GOTO 103
C     Cycle in N
102   IF(N)110,104,105
110   DO 111 I=MZ11,MZ20
      DO 111 J=1,MZ31
111   BZ(I,J)=0
101   DO 112 L=1,MZ
      CZ(1)=0
C     Search for a pivot element
      IF(DABS(BZ(I,L)).LT.DABS(CZ(1))) GOTO 113
      K=I
      CZ(1)=BZ(I,L)
113   CONTINUE
C     Pivot element is close to zero
      IF(DABS(CZ(1)).LE.1.D-50) GOTO 100
```

```
C       Rearrangement of the rows and elimination
        DO 114 J=L,MZ31
        BZ(K,J)=BZ(K,J)/CZ(1)
        IF(K.EQ.L) GO TO 114
        CZ(2)=BZ(K,J)
        BZ(K,J)=BZ(L,J)
        BZ(L,J)=CZ(2)
  114   CONTINUE
        DO 115 I=1,MZ20
        IF(I.EQ.L) GO TO 115
        CZ(1)=BZ(I,L)
        L1=L+1
        DO 116 J=L1,MZ31
  116   BZ(I,J)=BZ(I,J)-BZ(L,J)*CZ(1)
  115   CONTINUE
  112   CONTINUE
C       Formation of the matrices P, Q, R
        DO 118 I=1,MZ
        Y1(N+2,I)=BZ(I,MZ31)
        IF(N.EQ.-1) GOTO 118
        DO 117 J=1,MZ
        PZ(N+1,I,J)=BZ(I,J+MZ)
        IF(N.EQ.0) GOTO 117
        QZ(N,I,J)=BZ(I,J+MZ20)
  117   CONTINUE
  118   CONTINUE
        IF(N.EQ.-1) GOTO 106
        DO 120 I=1,MZ
        BZ(I,MZ31)=BZ(I+MZ,MZ31)
        DO 121 J=1,MZ20
  121   BZ(I,J)=BZ(I+MZ,J+MZ)
        DO 120 J=MZ21,MZ30
  120   BZ(I,J)=0
        N=N+1
C       End of the cycle in N
C       Receipt of the solution
  106   DO 122 J=1,MZ
        DO 122 I=1,MZ
  122   Y1(2,J)=Y1(2,J)-PZ(1,J,I)*Y1(1,I)
        DO 123 K=2,NZ
        DO 123 J=1,MZ
        DO 123 I=1,MZ
  123   Y1(K+1,J)=Y1(K+1,J)-PZ(K,J,I)*Y1(K,I)-QZ(K-1,J,I)*Y1(K-1,I)
C       Block for calculating coefficients at interior point of N
  105   CALL FF(N+1,NN)
        DO 51 K=1,MZ
   51   F1(K)=F(K)
        DO 52 I=1,MZ
        DY=(DABS(Y(N+2,I))+DABS(Y(N+3,I))+
       +DABS(Y(N+1,I))+DABS(Y0(N+2,I)))*EPS
        IF(DY.EQ.0) DY=EPS**2
        Y(N+2,I)=Y(N+2,I)+DY
        CALL FF(N+1,NN)
        Y(N+2,I)=Y(N+2,I)-DY
        DO 53 K=MZ11,MZ20
   53   BZ(K,I)=(F1(K-MZ)-F(K-MZ))/DY
        DY=(DABS(Y(N+1,I))+DABS(Y(N+2,I))+
       +DABS(Y(N,I))+DABS(Y0(N+1,I)))*EPS
        IF(DY.EQ.0) DY=EPS**2
        Y(N+1,I)=Y(N+1,I)+DY
        CALL FF(N+1,NN)
        Y(N+1,I)=Y(N+1,I)-DY
        DO 54 K=MZ11,MZ20
   54   BZ(K,I+MZ)=(F1(K-MZ)-F(K-MZ))/DY
        DY=(DABS(Y(N,I))+DABS(Y(N+1,I))+
       +DABS(Y(N-1,I))+DABS(Y0(N,I)))*EPS
        IF(DY.EQ.0) DY=EPS**2
        Y(N,I)=Y(N,I)+DY
        CALL FF(N+1,NN)
        Y(N,I)=Y(N,I)-DY
        DO 52 K=MZ11,MZ20
   52   BZ(K,I+MZ20)=(F1(K-MZ)-F(K-MZ))/DY
        DO 55 K=MZ11,MZ20
        BZ(K,MZ31)=F1(K-MZ)
        DO 55 I=1,MZ
   55   BZ(K,MZ31)=BZ(K,MZ31)+BZ(K,I)*Y(N+2,I)
       1+BZ(K,I+MZ)*Y(N+1,I)+BZ(K,I+MZ20)*Y(N,I)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      DO 56 K=MZ11,MZ20
      BZ(K,K)=BZ(K,K)+TAU1
   56 BZ(K,MZ31)=BZ(K,MZ31)+Y0(N+1,K-MZ)*TAU1
         Normalization of the row
      DO 57 I=MZ11,MZ20
      CZ(1)=0
      DO 58 J=1,MZ30
      IF(DABS(BZ(I,J)).LT.DABS(CZ(1))) GOTO 58
      CZ(1)=BZ(I,J)
   58 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 57 J=1,MZ31
   57 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO 101
         Block for calculating coefficients at the right end
  103 CALL FF(NN,NN)
      DO 31 K=1,MZ
   31 F1(K)=F(K)
      DO 32 I=1,MZ
      DY=(DABS(Y(NN,I))+DABS(Y(NN-1,I))+
     +DABS(Y(NN-2,I))+DABS(Y0(NN,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(NN,I)=Y(NN,I)+DY
      CALL FF(NN,NN)
      Y(NN,I)=Y(NN,I)-DY
      DO 33 K=1,MZ
   33 BZ(K,I)=(F1(K)-F(K))/DY
      DY=(DABS(Y(NN-1,I))+DABS(Y(NN,I))+
     +DABS(Y(NN-2,I))+DABS(Y0(NN-1,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(NN-1,I)=Y(NN-1,I)+DY
      CALL FF(NN,NN)
      Y(NN-1,I)=Y(NN-1,I)-DY
      DO 34 K=1,MZ
   34 BZ(K,I+MZ)=(F1(K)-F(K))/DY
      DY=(DABS(Y(NN-2,I))+DABS(Y(NN-1,I))+
     +DABS(Y(NN-3, ))+DABS(Y0(NN-2,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(NN-2,I)=Y(NN-2,I)+DY
      CALL FF(NN,NN)
      Y(NN-2,I)=Y(NN-2,I)-DY
      DO 32 K=1,MZ
   32 BZ(K,I+MZ20)=(F1(K)-F(K))/DY
      DO 35 K=1,MZ
      BZ(K,MZ31)=F1(K)
      DO 35 I=1,MZ
   35 BZ(K,MZ31)=BZ(K,MZ3 )+BZ(K,I)*Y(NN,I)
     1+BZ(K,I+MZ)*Y(NN-1,I)+BZ(K,I+MZ20)*Y(NN-2,I)
         Normalization of the row
      DO 37 I=1,MZ
      CZ(1)=0
      DO 38 J=1,MZ30
      IF(DABS(BZ(I,J)).LT.DABS(CZ(1))) GOTO 38
      CZ(1)=BZ(I,J)
   38 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 37 J=1,MZ31
   37 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO
         Block for calculating coefficients on the left end
  104 CALL FF(1,NN)
      DO 41 K=1,MZ
   41 F1(K)=F(K)
      DO 44 I=1,MZ
      DY=(DABS(Y(3,I))+DABS(Y(2,I))+
     +DABS(Y(4,I))+DABS(Y0(3,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(3,I)=Y(3,I)+DY
      CALL FF(1,NN)
      Y(3,I)=Y(3,I)-DY
      DO 43 K=MZ11,MZ20
   43 BZ(K,I)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(DABS(Y(2,I))+DABS(Y(3,I))+
     +DABS(Y(1,I))+DABS(Y0(2,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(2,I)=Y(2,I)+DY
      CALL FF(1,NN)
      Y(2,I)=Y(2,I)-DY
      DO 42 K=MZ11,MZ20
   42 BZ(K,I+MZ)=(F1(K-MZ)-F(K-MZ))/DY
      DY=(DABS(Y(1,I))+DABS(Y(2,I))+
     *DABS(Y(3,I))+DABS(Y0(1,I)))*EPS
      IF(DY.EQ.0) DY=EPS**2
      Y(1,I)=Y(1,I)+DY
      CALL FF(1,NN)
      Y(1,I)=Y(1,I)-DY
      DO 44 K=MZ11,MZ20
   44 BZ(K,I+MZ20)=(F1(K-MZ)-F(K-MZ))/DY
      DO 45 K=MZ11,MZ20
      BZ(K,MZ31)=F1(K-MZ)
      DO 45 I=1,MZ
   45 BZ(K,MZ31)=BZ(K,MZ31)+BZ(K,I)*Y(3,I)+
     1BZ(K,I+MZ)*Y(2,I)+BZ(K,I+MZ20)*Y(1,I)
C      Elimination of A(0)
      DO 60 J=MZ11,MZ20
      IF(BZ(J,K).EQ.0) GOTO 60
      DO 61 I=1,MZ
      BZ(J,I+MZ)=BZ(J,I+MZ)-PZ(2,K,I)*BZ(J,K)
   61 BZ(J,I+MZ20)=BZ(J,I+MZ20)-QZ(1,K,I)*BZ(J,K)
      BZ(J,MZ31)=BZ(J,MZ31)-Y1(3,K)*BZ(J,K)
   60 CONTINUE
      DO 62 I=1,MZ
      DO 62 J=MZ11,MZ20
      BZ(J,I)=BZ(J,I+MZ)
      BZ(J,I+MZ)=BZ(J,I+MZ20)
   62 BZ(J,[illegible]
C              Normaliztion of the row
      DO 47 I=MZ11,MZ20
      CZ(1)=0.
      DO 48 J=1,MZ20
      IF(DABS(BZ(I,J)).LT.DABS(CZ(1))) GOTO 48
      CZ(1)=BZ(I,J)
   48 CONTINUE
      IF(CZ(1).EQ.0.) GOTO 100
      DO 47 J=1,MZ31
   47 BZ(I,J)=BZ(I,J)/CZ(1)
      GOTO [illegible]
C              Failure in Delivery
  100 PRINT 1,N,K
      DO 3 I=1,MZ20
    3 [illegible]
    1 FORMAT (/23H Matrix is singular: N =,I3,
     -5H [illegible],I3,3H K=,I3,4H BZ=)
    2 FORMAT(10D12.4)
      STOP
      END
```

Subroutine DOUT

```
      SUBROUTINE DOUT(M,N)
      DOUBLE PRECISION BP,H,TAU1,XH,F,Y0,Y,Y1
      COMMON /HIF/ BP,H,TAU1,
     1XH(21),F(10),Y0(21,10),Y(21,10),Y1(21,10)
    1 FORMAT(//)
    2 FORMAT(11D11.3)
    9 FORMAT(10H     BPEMR=,D13.4)
      PRINT 9,BP
      PRINT 1
      DO 10 I=1,N
   10 PRINT 2,XH(I),(Y1(I,J),J=1,M)
      PRINT 1
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Printed at the ICR of the AS of the USSR

Sent to press 3/20/78