MATHEMATICA – REVUE D'ANALYSE NUMÉRIQUE
ET DE THÉORIE DE L'APPROXIMATION
L'ANALYSE NUMÉRIQUE ET LA THÉORIE DE L'APPROXIMATION,
Tome 19, N° 1, 1990, pp. 71–76

AN ALGORITHM FOR SOLVING THE MULTI-POINT BOUNDARY VALUE PROBLEMS FOR ORDINARY DIFFERENTIAL EQUATIONS

JÁN PEKÁR
(Bratislava)

1. Introduction. In this paper we construct and test an algorithm for solving the multi-point boundary value problem, which is from the class of such algorithms as were described in [4]. The algorithm for solving two-point boundary problems was constructed by Tewarson and Huslak [5], the algorithm for solving three-point boundary value problems was constructed by Pekár [3]. The exact formulation of the problem and the used notation are given in §2. A generalisation of the above-mentioned algorithms for multi-point boundary problems is described in detail in §3. The numerical aspects of the method are considered in §4. Here are also discussed some results obtained from computationa experiments.

2. The formulation of the problem. We consider the numerical solution of the multi-point boundary value problem for the system of s non-linear differential equations

$$y'(x) - f(x, y(x)) = 0 \tag{1}$$

with linear boundary value conditions

$$\sum_{i=1}^{N} A_i\, y(x_i) = b \tag{2}$$

where y and f are functions with values in R^s, $x_1 < x_2 < \ldots < x_N$, $A_i (i = 1, 2, \ldots, N)$ are given $N \times N$ matrices, and b is a given $N \times 1$ vector. We assume that problem (1), (2) has an isolated solution and the function f is sufficiently smooth.

Let us subdivide each x range $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, N - 1$, into m_i equal parts of the length $h_i = (x_{i+1} - x_i)/m_i$. Thus the x range $[x_1, x_N]$ is subdivided into $m = m_1 + m_2 + \cdots + m_{N-1}$ subintervals $[z_j, z_{j+1}]$, and we get a mesh containing $m + 1$ points.

Let us denote the exact value of the solution of the problem (1), (2) at point z_j as $y(z_j)$, the approximation of $y(z_j)$ as y_j, and $f_j = f(z_j, y_j)$. Now y is a vector with $m + 1$ components y_j. Every component y_j is a vector containing s components.

If we integrate equation (1) in the interval $[z_j, z_{j+1}]$, $j = 1, 2, \ldots, m$, then we have

$$(3) \qquad y(z_{j+1}) - y(z_j) - \int_{z_j}^{z_{j+1}} f(t, y(z))\, dt = 0.$$

Let us evaluate the integral in (3) by a numerical rule H_j. We obtain the following integration formulae

$$(4) \qquad y_{j+1} - y_j - h_i \cdot H_j = 0,$$

$j = 1, 2, \ldots, m$, $i = 1, 2, \ldots, N - 1$, where i depends on j. The integral may be evaluated by different quadrature rules. If the used rule has the local order of accuracy σ, then (4) will be written in the form

$$(5) \qquad y_{j+1} - y_j - h_i \cdot H_j^{\sigma} = 0.$$

3. The description of the method. The evaluation of the integral in (3) by (4) leads to a system of $s.m$ equations in $s.(m + 1)$ variables. From (2) we get

$$(6) \qquad A_1 y_1 + A_2 y_{m_1+1} + A_3 y_{m_2+1} + \ldots + A_n y_{m+1} = b$$

which is now a system of s equations in $s \cdot N$ variables. Because (6) is a linear system, we can express s variables from (6) in terms of the other $s.(N - 1)$ variables and then utilize them to eliminate s variables from the system obtained by (4). The resulting system of $s.m$ equations in $s.m$ variables can be written as

$$(7) \qquad F(y) = 0.$$

To solve (7), we select a starting vector $y^{(0)}$ and proceed iteratively, obtaining refined approximations $y^{(k)}$ to the solution vector.

An iteration consists of three steps which we describe now in greater detail.

Let ε be a given local error tolerance and $y^{(k-1)}$ an approximate solution vector for (1), (2). Let us specify the minimum order of accuracy of the used numerical rule in (4) and denote it as S. First we calculate the values of H_j^{CS} and H_j^{S+2}, where H_j^{CS} denotes the composite numerical rule of the order of accuracy S (the rule H_j^S is used twice: on the subinterval $[z_j, z_{j+1/2}]$ and on the subinterval $[z_{j+1/2}, z_{j+1}]$) and H_j^{S+2} denotes a numerical rule of the order of accuracy $S + 2$ which is from the same class of numerical rules as the rule H_j^S is. We calculate these values for all j. If the difference

$$(8) \qquad |H_j^{CS}(y^{(k-1)}) - H_j^{S+2}(y^{(k-1)})|$$

which is an estimate of the local error and is of the order S, is less than ε for any j, then we use $H_j^{S+2}(y^{(k-1)})$ in (7); otherwise the estimate of the order $S + 2$

$$(9) \qquad |H_j^{C(S)+2}(y^{(k-1)}) - H_j^{S+4}(y^{(k-1)})|$$

is calculated and compared with ε. If it is less than ε, then the value of $H_j^{(S+4)}(y^{(k-1)})$ is used as the value of H_j in (7); otherwise a repeated bisection of the interval under consideration $[z_j, z_{j+1}]$ until the local error tolerance is met, is necessary.

Thus we can write the calculation of the value H_j (actually the step a of the iteration) in the form

$$H_i = \begin{cases} H_j^{S+2}(y^{(k-1)}) & \text{if } |H_j^{CS}(y^{(k-1)}) - H_j^{S+2}(y^{(k-1)})| < \varepsilon. \\ H_j^{S+4}(y^{(k-1)}) & \text{if } |H_j^{C(S+2)}(y^{(k-1)}) - H_j^{S+1}(y^{(k-1)})| < \varepsilon. \\ \text{see step } b & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \ldots, m$.

If the estimate (9) is greater than or equal to ε, then we start the step b of the iteration, else we go to the step c.

If the use of step b is necessary, we add point $z_{j+1/2}$ to the original mesh and instead of the interval $[z_j, z_{j+1/2}]$ we consider the subintervals $[z_j, z_{j+1/2}]$ and $[z_{j+1/2}, z_{j+1}]$. We apply the step a of the iteration on both subintervals. At this stage $0(h^{S+3})$ values of the approximation of y at points inside the mesh-points are required. This problem is touched in § 4.

We repeatedly bisect all such intervals which do not satisfy the local error tolerance until the above-mentioned tolerance is met or too many bisections take place. In the first case we give the sum of the partial values of H obtained by repeated bisection at the place of H_j in (7)

$$(11) \qquad H_j = \sum_l H_{j_l}(y^{(k-1)})$$

for all such l, that $j \leqslant j_l < j + 1$ and the local error tolerance is met on $[z_{j_l}, t_{j_{l+1}}]$. In the latter case the use of our proposed method does not lead to the successful finish.

The above-described steps of the current iteration give the local error control of computations.

The final step of the iteration, step c, is the Newton step to update the current approximation $y^{(k)}$:

$$(12) \qquad y^{(k)} = y^{(k-1)} - [J(F(y^{(k-1)}))]^{-1} F(y^{(l-1)}),$$

where $J(F(y^{(k-1)}))$ denotes the Jacobian of F with respect to y evaluated at $y^{(k-1)}$.

The Jacobian in (12) can be computed by the consistent discrete approximation [2]:

$$(13) \qquad J(F(y^{(k-1)}))\, e_j(F(y^{(k-1)}) + \delta\, e_j) - F(y^{(k-1)}))/\delta,$$

where e_j is the j-th column of the identity matrix of the order sm and δ is a suitable small number. By the Discrete Newton Theorem [2] the convergence of (12) is not affected by the use of (13).

4. Numerical aspects of proposed method and numerical experiments.

In this part of the paper we touch the question "How to choose a suitable interpolation formula for the step b of the iteration" and discusse results obtained from computational experiments.

The algorithm was tested on following two multi-point boundary value problems [1]:

$$y_1' = y_2,$$
$$y_2' = y_3,$$
$$y_3' = y_4,$$
$$y_4' = (t^4 + 14t^3 + 49\,t^2 + 32\,t - 12)\,e^t,$$
$$y_1(0) = 0,\ y_2(0) = 0,\ y_1(1) = 0,\ y_2(1) = 0,$$

with exact solution

$$y_1(t) = t^2(1-t)^2\,e^t,$$
$$y_2(t) = (t^4 + 2t^3 - 5t^2 + 2t)\,e^t,$$
$$y_3(t) = (t^4 + 6t^3 + t^2 - 3t + 2)\,e^t,$$
$$y_4(t) = (t^4 + 10t^3 + 19t^2 - t - 1)\,e^t,$$

resp.

$$y_1' = y_2,$$
$$y_2' = \beta(y_1 - y_3),$$
$$y_3' = y_4,$$
$$y_4' = \alpha(y_3 - y_1),$$
$$y_1(0) = 0,\quad y_4(0) = 0,\quad y_2(s) = 0,\quad y_4(s) = C,$$

with $s = 10,\ C = 10^{-3},\quad \alpha = \beta = 2.5,$
which exact solution us

$$y_1 = \frac{\beta C}{r^2}\,(\gamma/r + t - \gamma\cos h(rt)\,/ + \frac{\beta}{\alpha}\,\sin h(rt)/r).$$

$$y_2 = \frac{\beta C}{r^2}\,(1 - \gamma\sin h(rt) + \frac{\beta}{\alpha}\,\cos h(rt)),$$

$$y_3 = \frac{C}{r^2}\,(\beta\gamma/r + \beta t + \alpha\gamma\cos h(rt)/r - \beta\sin h(rt)/r),$$

$$y_4 = \frac{C}{r^2}\,(\beta + \alpha\gamma\sin h(rt) - \beta\cos h(rt)),$$

where

$$r = (\alpha + \beta)^{1/2}$$

$$= \left(\frac{\beta}{\alpha}\cos h(rs) + 1\right)/\sin h\,(rs).$$

All computations were performed in double precision arithmetics on a SM 52/11 computer using FORTRAN 77. In the step *a* of every iteration the following quadrature rules were used:

$$H_j^3 = (f_j + f_{j+1})\,/2,$$
$$H_j^{C3} = (f_j + 2f_{j+1/2} + f_{j+1})/4,$$
$$H_j^5 = (f_j + 4f_{j+1/2} + f_{j+1})/6,$$
$$H_j^{C5} = (f_j + 4f_{j+1/4} + 2f_{j+1/2} + 4f_{j+3/4} + f_{j+1})/12,$$

and

$$H_j^7 = (f_j + 32f_{j+1/4} + 12f_{j+1/2} + 32f_{j+3/4} + f_{j+1})/90,$$

All these formulae are well known. H_j^3 denotes the trapezoidal rule, H_j^5 denotes the Simpson rule, H_j^7 denotes the Newton-Côtes rule, and H_j^{C3} and H_j^{C5} denote the appropriate composite rules.

The last problem we yet touch is the computation of the approximate values of the solution inside the mesh-points and the appropriate values of *f*. Various interpolation formulae are available to satisfy the error criterion. The used formula must be at least of the order $S + 4$, because the use of lower order formula does not lead to significantly better results. For all needed interpolations the Newton forward rule and the Newton backward rule of the 7-th order were used.

In Table 1 we compare the maximum of absolute errors for all components of the solution for both tested problems. The computations started with 1001 mesh-points (i.e. $m = 1000$).

Table 1

	maximum of absolute error in			
	y_1	y_2	y_3	y_4
the first problem	1.10^{-13}	7.10^{-14}	1.10^{-13}	5.10^{-13}
the second problem	3.10^{-11}	4.10^{-11}	7.10^{-12}	7.10^{-12}

In Table 2 you can find the number of iterations and the number of added points which was needed to achieve the requested accuracy for the second boundary value problem which is strong non-linear.

Table 2

Absolute accuracy	10^{-6}	10^{-8}	10^{-10}
number of iterations	13	17	25
number of added mesh-points	5	19	43

Finally, we can say that the proposed method requests a lot of computations, but it leads to the successful finish.

REFERENCES

1. L e n t i n i , M. P e r e y r a , V., *A variable order finite difference method for nonlinear multipoint boundary value problem.* Math. Comp. 28(1974), No. 128, 981—1003.
2. O r t e g a , J. R h e i n b o l d t , W. C., *Iterative solution of non-linear equations in several variables.* Academic Press, New York, 1970.
3. P e k á r , J., *An algorithm for solving the three-point boundary value problem for O.D.E.* Acta Math. (in press).
4. P e k á r J., *O jednej triede algoritmov pre riešenie okrajových úloh pre nelineárne obyčajné diferenciálne rovnice.* Sisy-Větev, 19(1985), 108—109.
5. T e w a r s o n , R. P., H u s l a k , N. S., *An adaptive implementation of interpolation methods for boundary value ordinary differential equations.* BIT 23 (1983), 382—387.

Received 22.XII.1988

Dept. of Numerical and Optimizing Methods
Comenius University
842 15 Bratislava
Czechoslovakia