Mathematical Modelling and Analysis
Volume 4, 1999, pages 51–57

DEGENERATE MATRIX METHOD WITH CHEBYSHEV NODES FOR SOLVING NONLINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS

T. CĪRULIS, O. LIETUVIETIS

Institute of Mathematics of Latvian Academy of Sciences and University of Latvia

Akadēmijas laukums 1, Rīga LV–1524, Latvia

E-mail: cirulis@lanet.lv, ojarsl@lanet.lv

Received September 30, 1999

ABSTRACT

One of the simplest schemes of the degenerate matrix method with nodes as zeroes of Chebyshev polynomials of the second kind is considered. Performance of simple iterations and some modifications of Newton method for the discrete problem is compared.

1. INTRODUCTION

Initial value problem of nonlinear systems of differential equations has many important applications to different practical tasks. In practice majority of these problems can be solved only numerically. Therefore, numerical methods are considered in the literature very often, for example, volumes [1; 2]. They contain bibliography of at least 800 titles. In [3] DM-method (Degenerate matrix method) as the method for numerical solving of the above-mentioned problems is considered. DM-method is based on the choice of nodes in the standardized interval $[-1, 1]$ and on the use of two types of matrices for the discretization of the differential problem: matrices for derivatives which are always degenerated and their pseudo-inverses. This paper is devoted to DM-method with nodes as zeroes of Chebyshev polynomials of the second kind and with specially selected constants in the pseudo-inverse matrix for a derivative. This discretization for problems of differential equations leads to very simple

computing schemes and can be realized by computers easy. In addition, these computing scheme can be generalized applying the Newton method instead of simple iterations for the discretized equation. As example an application of the method to van der Pol's equation is considered

2. SIMPLEST DM-METHOD WITH CHEBYSHEV POLYNOMIALS OF THE SECOND KIND

The problem

$$\frac{dy}{dt} = f(t,y), y(a) = \alpha, \quad t \in [a,b], \tag{2.1}$$

where $y \in \mathbf{R}^m$, $\alpha \in \mathbf{R}^m, f = (f_1, f_2, \ldots, f_m)$, can be solved using the multistep procedure on subintervals $[t_i, t_{i+1}], a = t_0 < t_1 < \ldots < t_s = b$. Each of problems on the subinterval can be represented by the substitution

$$t = t_i + 0.5h_i(x+1), \quad h_i = t_{i+1} - t_i \tag{2.2}$$

as a problem on the standardized interval $[-1,1]$:

$$\frac{dy}{dx} = \frac{h}{2}F(x,y), \; y(-1) = \beta, \; x \in [-1,1]. \tag{2.3}$$

Here $h = h_i; \beta = \beta_i = y(t_i)$ and F are vectors obtained by the substitution (2.2) in (2.1).

We choose nodes x_k on the interval $[-1,1]$ as the zeroes of Chebyshev polynomial $U_N(x)$ of the second kind:

$$x_k = -\cos\frac{k\pi}{N+1}, \quad k = 1,2,\ldots,N. \tag{2.4}$$

This system of nodes can be extended by the endpoints ± 1 simply putting $k = 0$ and $k = N+1$ in (2.4). In [3] one of pseudo-inverse matrices with nodes (2.4) is given, which is suitable for non-stiff problems. For stiff problems DM-method is useful only theoretically since it requires very long time for calculations. Now we want to present one of DM-methods with nodes (2.4) which have very simple pseudo-inverse matrix. It is possibly the simplest one among all of them for DM-method.

This scheme for solving (2.3) with such matrix is following:

$$\mathbf{Y}[N,m] = \mathbf{Y}_0[N,m] + \frac{h}{2}\mathbf{B}_N\mathbf{F}[N,m], \tag{2.5}$$

$$y(1) = \beta + \frac{h}{2}\sum_{k=1}^{N} b_{N+1,k}F(x_k, y(x_k)). \tag{2.6}$$

Here $\mathbf{B}_N$ is the $N \times N$ matrix with elements

$$b_{jk} = \frac{4}{N+1} \sin \frac{\pi k}{N+1} \sum_{n=1}^{N} \frac{1}{n} \sin \frac{\pi k n}{N+1} \sin^2 \frac{\pi j n}{2N+2}, \; j,k = 1,2,\ldots,N. \tag{2.7}$$

(2.7) is correct for $j = 0$ and $j = N+1$, too.

$\mathbf{Y}$, $\mathbf{Y}_0$ and $\mathbf{F}$ are matrices with elements: $y_{kj} = y_j(x_k), y_{kj}^{(0)} = y_j(-1), F_{kj} = F_j(x_k, y(x_k)), k = 1,2,\ldots,N; j = 1,2,\ldots,m.$

The norm

$$\|\mathbf{B}_N\| := \max_j \sum_{k=1}^{N} |b_{jk}| < 2. \tag{2.8}$$

The inequality (2.8) is correct for any N.

Proof. We are starting from DM-method with nodes $x_k = -\cos \frac{k\pi}{N+1}$, $k = 0,1,\ldots,N$ and pseudo-inverse matrix $\tilde{\mathbf{B}}_{N+1}$ in the general form [3]. Then elements $\tilde{b}_{jk}$ have the representation

$$\tilde{b}_{jk} = \frac{1}{p'_{N+1}(x_k)} \left[\int_{-1}^{x_j} \frac{p_{N+1}(\tau)}{\tau - x_k} d\tau + c_j \right], \; p_{N+1}(x) = (1+x)U_N(x), \tag{2.9}$$

where c_j are constants, $\mathbf{U}_N$ are Chebyshev polynomials of the second kind. We choose ones according to the conditions: $\tilde{b}_{0k} = 0$, $k = 0,1,\ldots,N$; and $\tilde{b}_{j0} = 0$, $j = 1,2,\ldots,N$. Then we obtain $c_0 = 0$, and $c_j = -\int_{-1}^{x_j} U_N(\tau)d\tau$. Denoting elements of $\mathbf{B}_{N+1}$ with such constants by b_{jk} leads to

$$b_{jk} = \frac{1}{U'_N(x_k)} \int_{-1}^{x_j} \frac{U_N(\tau)}{\tau - x_k} d\tau, \; j,k = 0,1,\ldots,N. \tag{2.10}$$

Let $\mathbf{B}_N$ be the matrix with $b_{jk}, j,k = 1,2,\ldots,N$. Computing the integral by means of the classical Christoffel-Darboux formula leads to (2.7). Now we define $b_{N+1,k} = b_{jk}$ if $j = N+1$. So we have obtained (2.6). The equation (2.5) follows from the scheme of DM-method. The inequality (2.8) follows from properties: $b_{N+1,k} > 0$ if $k = 1,2,\ldots,N$. ■

Remark. Instead of (2.5) and (2.6) only one equation

$$\mathbf{Y}[N+2,m] = \mathbf{Y}_0[N+2,m] + \frac{h}{2}\tilde{\mathbf{B}}_{N+2}\mathbf{F}[N+2,m] \tag{2.11}$$

can be used, where the elements of the matrix $\tilde{\mathbf{B}}_{N+2}$ can be computed by (2.7) with $j,k = 0,1,\ldots,N+1$. The equation (2.11) can be solved by iterations, too. In this case $\|\tilde{\mathbf{B}}_{N+2}\| = 2$ for any N.

Method using (2.5) and (2.6) is a special case of Runge-Kutta method with very simple nodes and matrix $\mathbf{B}_N$. The N as a number of stages can be large on each step, too. A choice of nodes as zeroes of orthogonal polynomials gives a nonsaturated approximation for functions. Therefore, the use of DM-method with large N is very reasonable because such N allows us to decrease computing errors of the method rapidly.

Using these matrices with different values of N in the same subinterval of (2.3) we can control a total precision of the method simply and directly.

3. APPLICATION OF NEWTON METHOD

Simple iterations for solving (2.5) in the case of small enough h guarantee a high precision. But these iterations with a small h and large numbers of stages N require a long time for calculations by the multi-step procedure. Numerical experiments show that $\|\mathbf{B}_N^k\|$ for each fixed N decreases rapidly if k tends to infinity. Therefore, it is appropriate to use Newton method or its modifications instead of simple iterations for solving of (2.5) . In order to formulate these methods we will consider some operations with matrices. We will denote matrices by $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$, but matrix vectors by $\vec{\mathbf{A}}, \vec{\mathbf{B}}, \vec{\mathbf{C}}, \ldots$.

DEFINITION 3.1. Let $\mathbf{A}$ be an $N \times n$ matrix. We define the operation $\bigotimes$ between the matrix vector $\vec{\mathbf{B}}$ having N matrices of type $n \times k$ as components and a matrix $\mathbf{A}$ in the following way:

$$\vec{\mathbf{B}} \bigotimes \mathbf{A} = \mathbf{C}, \tag{3.1}$$

if $\mathbf{C}$ is the $N \times k$ matrix the j-th column of which can be obtained multiplying the matrix $\mathbf{B}_j$ as the component of $\vec{\mathbf{B}}$ with j-th column of $\mathbf{A}$ as with $N \times 1$ matrix.

Analogously the multiplying of a matrix $\mathbf{A}$ with a matrix vector having k matrices of the type $n \times N$ as components can be defined:
$\mathbf{A} \bigotimes \vec{\mathbf{B}} = \mathbf{C}$ if $\mathbf{C}$ is the $n \times k$ matrix each row of which can be obtained multiplying the j-th row of $\mathbf{A}$ as $1 \times k$ matrix with $\mathbf{B}_j$ as the component of $\vec{\mathbf{B}}$.

DEFINITION 3.2. Let $\vec{\mathbf{A}}$ and $\vec{\mathbf{B}}$ be matrix vectors, having the same number n of components. Then we define

$$\vec{\mathbf{A}} * \vec{\mathbf{B}} = \vec{\mathbf{C}} := (\mathbf{A_1 B_1}, \mathbf{A_2 B_2}, \ldots, \mathbf{A_n B_n}), \tag{3.2}$$

where $\mathbf{A}_i \mathbf{B}_i$ are matrix products.

We will use also notations

$$\vec{\mathbf{A}}^T = (\mathbf{A}_1^T, \ldots, \mathbf{A}_n^T), \vec{\mathbf{E}} = (\mathbf{E}, \ldots, \mathbf{E}); \vec{\mathbf{A}}^{-1} = (\mathbf{A}_1^{-1}, \ldots, \mathbf{A}_n^{-1}),$$

if $\mathbf{A}_j^{-1}$ exist for each $j = 1, 2, \ldots, n$. It is obviously that $\vec{\mathbf{A}} * \vec{\mathbf{A}}^{-1} = \vec{\mathbf{A}}^{-1} * \vec{\mathbf{A}} = \vec{\mathbf{E}}$, $(\vec{\mathbf{A}} * \vec{\mathbf{B}})^T = \vec{\mathbf{B}}^T * \vec{\mathbf{A}}^T$, $(\mathbf{A}, \mathbf{A}, \ldots, \mathbf{A}) \bigotimes \mathbf{B} = \mathbf{AB}$,
$\mathbf{B} \bigotimes (\mathbf{A}, \mathbf{A}, \ldots, \mathbf{A}) = \mathbf{BA}$, $\mathbf{A} \bigotimes \vec{\mathbf{E}} = \vec{\mathbf{E}} \bigotimes \mathbf{A} = \mathbf{A}$, etc.

We formulate still one property which will be necessary below.

Proposition 3.1. *Let us consider* $*, \bigotimes, .$ *as the operations of taking a product of: 1) two matrix vectors; 2) a matrix with a matrix vector or a matrix vector with a matrix; 3) two matrices, respectively. We assert that these operations considered as one generalized multiplication are associative. For example,*

$$\vec{\mathbf{A}} \bigotimes (\vec{\mathbf{B}} \bigotimes \mathbf{C}) = (\vec{\mathbf{A}} * \vec{\mathbf{B}}) \bigotimes \mathbf{C},$$

$$\mathbf{A}.(\vec{\mathbf{B}} \bigotimes \mathbf{C}) = (\mathbf{A} \bigotimes \vec{\mathbf{B}}).\mathbf{C},$$

$$\mathbf{A}.(\mathbf{C} \bigotimes \vec{\mathbf{B}}) = (\mathbf{AC}) \bigotimes \vec{\mathbf{B}}, \text{ etc.}$$

A proof of the proposition can be obtained by simple verifying of all possible cases.

Now we will use Newton method for solving the equation (2.5). Denoting for simplicity $\mathbf{Y}[N, m] = \mathbf{Y}$ and $\mathbf{F}[N, m] = \mathbf{F}(\mathbf{Y})$, we have instead of (2.5)

$$\mathbf{\Phi}(\mathbf{Y}) \equiv \mathbf{Y} - \mathbf{Y}_0 - \frac{h}{2}\mathbf{B}_N\mathbf{F}(\mathbf{Y}) = 0. \tag{3.3}$$

The Frechet differential of $\mathbf{\Phi}$ is

$$d\mathbf{\Phi}(\mathbf{Y}) = \Delta\mathbf{Y} - \frac{h}{2}\mathbf{B}_N.\Delta\mathbf{Y} \bigotimes \vec{\mathbf{F}}'(\mathbf{Y}), \tag{3.4}$$

where $\vec{\mathbf{F}}'(\mathbf{Y})$ is a matrix vector with components, which are transposed Jacoby matrices

$$\frac{D(F_1, F_2, \ldots, F_m)}{D(y_1, y_2, \ldots, y_m)} \text{ at } x = x_k, k = 1, 2, \ldots, N. \tag{3.5}$$

Accordingly to the well known Newton method iterations for equation $\mathbf{\Phi}(\mathbf{Y}) = 0$ are $d\mathbf{\Phi}(\mathbf{Y^{(n)}}) = -\mathbf{\Phi}(\mathbf{Y^{(n)}})$, $n = 0, 1\ldots$. Using (3.4) and (3.5) we have

$$\mathbf{Y}^{(n+1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B_N}\mathbf{F}(\mathbf{Y^{(n)}}) + \frac{h}{2}\mathbf{B}_N[\mathbf{Y}^{(n+1)} - \mathbf{Y}^{(n)}] \bigotimes \vec{\mathbf{F}}'(\mathbf{Y^{(n)}}). \tag{3.6}$$

Unfortunately, a direct application of (3.6) requires solving of the linear algebraic system of kind

$$\mathbf{Y}^{(n+1)} - \frac{h}{2}\mathbf{B_N}\mathbf{Y}^{(n+1)} \bigotimes \vec{\mathbf{F}}'(\mathbf{Y^{(n)}}) = \mathbf{\Psi}(\mathbf{Y^{(n)}}) \tag{3.7}$$

for each $n = 0, 1, \ldots$. It is difficult in practice. We suggest three rough methods for solving (3.7).

Algorithm 3.1. Replacing $\mathbf{Y}^{(n+1)}$ in the right side of (3.6) with

$$\mathbf{Z}^{(n+1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B}_N\mathbf{F}(\mathbf{Y}^{(n)}) \approx \mathbf{Y}^{(n+1)} \tag{3.8}$$

we obtain the following computing scheme for iterations:

$$\mathbf{Z}^{(n+1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B}_N\mathbf{F}(\mathbf{Y}^{(n)}),$$

$$\mathbf{Y}^{(n+1)} = \mathbf{Z}^{(n+1)} + \frac{h}{2}\mathbf{B}_N[\mathbf{Z}^{(n+1)} - \mathbf{Y}^{(n)}] \bigotimes \vec{\mathbf{F}}'(\mathbf{Y}^{(n)}), \tag{3.9}$$

Algorithm 3.2. Replacing $\Delta\mathbf{Y}^{(n)} = \mathbf{Y}^{(n+1)} - \mathbf{Y}^{(n)}$ roughly with $\Delta\mathbf{Y}^{(n-1)} = \mathbf{Y}^{(n)} - \mathbf{Y}^{(n-1)}$ we obtain the following computing scheme:

$$\mathbf{Y}^{(0)} = \mathbf{Y}_0; \quad \mathbf{Y}^{(1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B_N}\mathbf{F}(\mathbf{Y}^{(\mathbf{0})}),$$

$$\mathbf{Y}^{(n+1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B_N}\mathbf{F}(\mathbf{Y}^{(\mathbf{n})}) + \frac{h}{2}\mathbf{B_N}(\mathbf{Y}^{(\mathbf{n})} - \mathbf{Y}^{(\mathbf{n-1})}) \bigotimes \vec{\mathbf{F}}'(\mathbf{Y}^{(\mathbf{n})}), \tag{3.10}$$

$$n = 1, 2, \ldots$$

Algorithm 3.3. In this case we replace $\vec{\mathbf{F}}'(\mathbf{Y}^{(n+1)})$ in (3.6) with $\vec{\mathbf{F}}'(\mathbf{Y}_0)$ and obtain $\mathbf{Y}^{(n+1)}$ directly from the system of algebraic equations

$$\mathbf{Y}^{(n+1)} = \mathbf{Y}_0 + \frac{h}{2}\mathbf{B_N}\mathbf{F}(\mathbf{Y}^{(\mathbf{n})}) + \frac{h}{2}\mathbf{B}_N[\mathbf{Y}^{(n+1)} - \mathbf{Y}^{(n)}] \bigotimes \vec{\mathbf{F}}'(\mathbf{Y_0}). \tag{3.11}$$

4. NUMERICAL EXAMPLES

The above mentioned algorithms were applied to the van der Pol's equation written as a system

$$y_1' = y_2, \quad y_2' = \epsilon[(1 - y_1^2)y_2 - y_1] \tag{4.1}$$

with initial conditions

$$y_1(0) = 2, \quad y_2(0) = -0.66$$

and $\epsilon = 10$.

Calculations were carried out with different step sizes h and numbers N of nodes on the standardized interval [-1,1]. The obtained numerical results

were compared with ones in the case of simple iterations for (2.5). We can make the following conclusions.

The Algorithm 3.1 converges approximately two times faster than the simple iterations of (2.5) and gives the same results. However, the computing time of all calculations is not reduced very much. It is because each iteration requires two multiplications with the pseudo-inverse matrix in (3.9) instead of one in (2.5). The observed efficiency decreases for small step sizes and large numbers of nodes.

The Algorithm 3.2 has a rate of the convergence lower than simple iterations in (2.5) approximately $1.5 \div 2$ times, and it requires a bit smaller step size. Obviously, this method can not be recommended instead of simple iterations.

The Algorithm 3.3 is the standard modification of Newton method. This allows greater step size and converges faster than simple iterations do. However, the precision is lower than in the case of simple iterations and it can not be increased choosing a larger number of nodes. This method can be recommended in cases when the step size for simple iterations is too large to achieve the convergence of simple iterations.

REFERENCES

[1] E.Haier, P.Norsett and G.Wagner. *Solving Ordinary Differential Equations, Vol.1: Nonstiff Problems.* Springer Verlag, Berlin, 1993.

[2] E. Haier and G. Wanner. *Solving Ordinary Differential Equations, Vol.2: Stiff and Differential-Algebraic Problems.* Springer Verlag, Berlin, 1996.

[3] T. Cīrulis and O. Lietuvietis. Degenerate matrix method for solving nonlinear system of differential equations. *Journal of Mathematical Modelling and Analysis*, **3**, 1998, 45 – 56.

IŠSIGIMUSIŲ MATRICŲ METODAS NETIESINIŲ DIFERENCIALINIŲ LYGČIŲ SPRENDIMUI

T. CIRULIS, O. LIETUVIETIS

Šiame darbe toliau nagrinėjamas išsigimusių matricų metodas netiesinių diferencialinių lygčių pradinio uždavinio sprendimui. Sudarant metodo algoritmą naudojami antrojo tipo Čebyševo polinomų nuliai bei specialiai parenkamos konstantos aproksimuojant išvestinės kvaziatvirkštinę matricą. Metodas pritaikomas dviejų tipų iteraciniams algoritmams: paprastųjų iteracijų ir Niutono metodams. Pateikiami skaitinio eksperimento rezultatai ir naujasis metodas palygintas su populiariaisiais skaitiniais algoritmais.