Numerical Algorithms **27:** 119–130, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

A method for global approximation of the initial value problem

F. Costabile and A. Napoli

Department of Mathematics, Università della Calabria, 87036 Rende (Cs), Italy
E-mail: {costabil;napolia}@unical.it

Received 25 June 1999; revised 2 April 2001
Communicated by C. Brezinski

For the numerical solution of the initial value problem a parallel, global integration method is derived and studied. It is a collocation method. If $f(x, y) \equiv f(x)$ the method coincides with the Filippi's modified Clenshaw–Curtis quadrature [11]. Two numerical algorithms are considered and implemented, one of which is the application of the new method to Picard iterations, so it is a waveform relaxation technique [3]. Numerical experiments are favourably compared with the ones given by the known GAM [2], GBS [14] and Sarafyan [18] methods.

Keywords: initial value problem, Chebyshev series

AMS subject classification: 65L05, 65D15

1. Introduction

In this paper we will consider the initial value problem in ordinary differential equations

$$\begin{cases} y'(x) = f\big(x, y(x)\big), & x \in [a, b], \\ y(x_0) = y_0. \end{cases} \tag{1}$$

We assume that (1) represents a single scalar equation, but nearly all of the numerical and theoretical considerations in this paper carry over systems of first order equations, where (1) could be treated in vector form.

We develop a method which produce smooth, global approximations to $y(x)$ in the form of polynomial functions.

Moreover, $f(x, y)$ should satisfy sufficient conditions to guarantee that a unique solution of (1) exists.

The known numerical methods generally fall into two categories: the continuous and the discrete. The continuous methods include collocation methods; the discrete methods include extrapolation, Runge–Kutta and linear multistep methods. Any method in group one, i.e., continuous, can be used to produce approximations at discrete points, but many discrete methods do not readily provide continuous approximations. In particular, this is the case of extrapolation and most Runge–Kutta methods. As a result, these methods can be inefficient for problems requiring globally continuous differentiable functions as approximations of $y(x)$, as it is required, for instance, to produce a graph of numerical solution of (1).

One remedy for this deficiency may be the construction of an interpolant based on a sufficient number of approximations y_n to $y(x_n)$, $x_n = x_{n-1} + h_{n-1}$, $n \geqslant 1$ and the "derivatives" $f(x_n, y_n)$. For high order Runge–Kutta methods, however, this requires information about more than two values of x_n, compromising the one-step nature of this class of methods. But in [12] by a "boat strapping" procedure the construction of high-order interpolants for Runge–Kutta methods is developed. This interpolants are globally C^1.

Dense output procedures as GBS method are not C^1.

In [18] a method which provides continuously differentiable functions as approximations of $y(x)$ is developed, which in the nodal point is a Runge–Kutta method, but it is able to work properly only on $[x_0, x_0 + h]$, with h small enough, even if variable.

An alternative way of solving differential equations, which is different are Boundary Value Methods (BVMs). GAMs (Generalized Adams Methods) are a family of BVMs, obtained from generalizations of Adams methods of order 3, 5, 7, 9 with step size control, which provide piecewise polynomial approximations.

Our approach is different: we approximate $y(x)$ on $[x_0, b]$ by a polynomial of degree n ($n > 1$), that assures sufficient condition of regularity. For this purpose we determine a polynomial of collocation on the set of nodes $x_i = \cos(\pi i/(n+1))$, $i = 1, \ldots, n+1$.

We will propose two algorithms to calculate the numerical solution of (1) in the nodal points, one of which coincides with the application of our method to Picard iterations. All that suggests an in-depth study and particularly a comparison with WR form of Runge–Kutta methods [1], which will be done in the future.

The outline of the paper is the following: in section 2 we present the method; in section 3 we study the order; in section 4 we show the analysis of convergence; in section 5 we give numerical algorithms and, finally, in section 6 we present some numerical examples in comparison with GBS [14], GAM [2] and Sarafyan [18] methods, which are considered three of the more efficient respectively dense output and continuous methods.

2. The method

We have the following:

Theorem 1 (The main theorem). Let us consider the initial value problem

$$\begin{cases} y'(x) = f\big(x, y(x)\big), & x \in [-1, 1], \\ y(-1) = y_0. \end{cases} \tag{2}$$

Setting

$$x_i = \cos \frac{\pi i}{n+1}, \quad i = 1, \dots, n, \tag{3}$$

$T_k(x)$ the Chebyshev polynomial of first kind of degree k,

$${}^n\gamma_i(x) = \frac{2}{n+1} \sin \frac{\pi i}{n+1} \sum_{k=1}^{n} \frac{T_k(x) + (-1)^{k+1}}{k} \sin \frac{\pi k i}{n+1}, \tag{4}$$

then the polynomial of degree n

$$y_n(x) = y_0 + \sum_{i=1}^{n} {}^n\gamma_i\,(x)\, f\bigl(x_i, y_n(x_i)\bigr) \tag{5}$$

satisfies the relations

$$\begin{aligned} y_n(-1) &= y_0, \\ y_n'(x_j) &= f\bigl(x_j, y_n(x_j)\bigr), \quad j = 1, \dots, n, \end{aligned} \tag{6}$$

i.e., it is a collocation polynomial for (2) [14] at nodes x_i, $i = 1, \dots, n+1$.

Proof. $\forall i, n \in \mathbb{N}$ we have

$${}^n\gamma_i(-1) = 0,$$

so that

$$y_n(-1) = y_0.$$

For the polynomial ${}^n\gamma_i(x)$ we get

$${}^n\gamma_i'(x) = \frac{2}{n+1} \sin \frac{\pi i}{n+1} \sum_{k=1}^{n} \frac{T_k'(x)}{k} \sin \frac{\pi k i}{n+1}. \tag{7}$$

But

$$T_k'(x) = k U_{k-1}(x), \quad k \geqslant 1, \tag{8}$$

where $U_{k-1}(x)$ is the Chebyshev polynomial of second kind, which satisfies

$$U_{k-1}(x) = \frac{\sin(kt)}{\sin t} \tag{9}$$

with $x = \cos t$.

Inserting (8) and (9) into (7) we have

$${}^n\gamma_i'(\cos t) = \frac{2}{n+1} \sin \frac{\pi i}{n+1} \sum_{k=1}^{n} \frac{\sin kt}{\sin t} \sin \frac{k \pi i}{n+1}$$

and for $t = t_j = \arccos x_j$, from the orthogonality of the system of functions $\sin mt$ [11, p. 353], it follows that

$$ {}^n\gamma_i'(x_j) = {}^n\gamma_i'(\cos t_j) = \delta_{ij}. \tag{10}$$

From (5) we have

$$y_n'(x) = \sum_{k=1}^{n} {}^n\gamma_k'(x) f\big(x_k, y_n(x_k)\big)$$

and from (10)

$$y_n'(x_j) = f\big(x_j, y_n(x_j)\big), \quad j = 0, \dots, n. \qquad \square$$

From the result of theorem 1 we give the following definition:

Definition 1. Using notations of theorem 1, the polynomial (5) is a global approximation method or an implicit continuous Runge–Kutta type method for the solution of (2) in $[-1, 1]$.

If $y'(t) = f(t)$, i.e., $f(t, y(t))$ does not depend on $y(t)$, for $x = 1$ the (5) coincides with the Filippi's modified Clenshaw–Curtis quadrature formula [11].

3. The order

In order to define the order of method (5) we associate with (5) the linear difference operator L defined by

$$L\big[y(t), x_0\big] = y(t) - y(x_0) - \sum_{i=1}^{n} {}^n\gamma_i(t) y'(x_i), \tag{11}$$

where $y(t)$ is the exact solution of (2), $x_0 = -1$ and x_i are defined in (3).

The following lemma holds:

Lemma 1.

$$\frac{(x_0 - x)^k}{k!} + \sum_{i=1}^{n} {}^n\gamma_i(x) \frac{(x_i - x)^{k-1}}{(k-1)!} = 0, \quad k = 1, \dots, n.$$

Proof. The result follows by applying Filippi quadrature formula [11] to

$$f(t) = \frac{(t - x)^{k-1}}{(k-1)!}, \quad k = 1, \dots, n. \qquad \square$$

Now we get:

Theorem 2. For the operator (11) we have

$$L\big[y(x), x_0\big] = c_{n+1}(x) y^{(n+1)}(x) + \cdots,$$

where $c_{n+1}(x)$ does not depend on the function $y(x)$.

Proof. From the Taylor expansion of $y(x_0)$ and of $y'(x_i)$ we have

$$L\big[y(x), x_0\big] = c_1(x) y'(x) + \cdots + c_q(x) y^{(q)}(x) + \cdots,$$

where

$$c_q(x) = -\frac{(x_0 - x)^q}{q!} - \sum_{i=1}^{n} {}^n\gamma_i(x) \frac{(x_i - x)^{q-1}}{(q-1)!}, \quad q = 1, 2, \ldots,$$

does not depend on $y(x)$.

From lemma 1 $c_1(x) = c_2(x) = \cdots = c_n(x) = 0$, so the result follows. □

From theorem 2 we say that (5) is a method of order n.

4. Convergence of the method

Let us consider the problem (2) where $f(x, y(x))$ is a real function defined and continuous on the strip $S = [-1, 1] \times \mathbb{R}$ and a constant L exists so that the inequality

$$\big|f(x, y_1) - f(x, y_2)\big| \leqslant L|y_1 - y_2|$$

holds true over the strip S.

Under these hypotheses the problem (2) has a unique solution $y(x)$.

Let us define the space

$$C_* = \big\{g \in C\big([-1, 1], \mathbb{R}\big) \colon\ g(-1) = y_0\big\}$$

and the operator $F_* : C_* \to C_*$ such that

$$(F_* g)(x) = y_0 + \int_{-1}^{x} f\big(t, g(t)\big)\, \mathrm{d}t. \tag{12}$$

So $(F_* y)(x) = y(x)$, i.e. y is a fixed point of F_*.

Let us define then the sequence of operators

$$\begin{cases} (F_1 g)(x) = y_0 + (x+1) f(-1, y_0), \\ (F_n g)(x) = y_0 + \sum_{k=1}^{n} {}^n\gamma_k(x) f\big(x_k, g(x_k)\big), \quad n > 1. \end{cases} \tag{13}$$

Proposition 1. The following statements hold:

1. There exists $\overline{n} \in \mathbb{N}$ so that $\forall n > \overline{n}$ the map F_n is contractive and has a unique fixed point $y_n \in C_*$.

2. If y is the exact solution of (2), i.e., the fixed point of F_*, then $y_n(x) \to y(x)$ in $[-1, 1]$.

3. For all fixed $x \in [-1, 1]$

$$y_n(x) - y(x) = \int_{-1}^{x} K(t, x) y^{(n)}(t)\,\mathrm{d}t,$$

where

$$K(t, x) = \frac{1}{(n-1)!}\left(\sum_{k=1}^{n} {}^{n}\gamma_k(x)(x_k - t)_+^{n-1} - \frac{(x-t)^n}{n}\right).$$

Proof.

1.

$$\begin{aligned}\left|(F_n\varpi)(x) - (F_n v)(x)\right| &\leqslant \sum_{r=0}^{n} \left|{}^{n}\gamma_r(x)\right|\left|f\left(x_r, \varpi(x_r)\right) - f\left(x_r, v(x_r)\right)\right| \\ &\leqslant \frac{LM}{n}\|\varpi - v\|\end{aligned}$$

with $0 < M < \infty$. Hence, for n large enough, $k < 1$ exists such that

$$\|F_n\varpi - F_n v\| \leqslant k\|\varpi - v\|.$$

2. Let y_n be the fixed point of F_n. Then

$$\begin{aligned}\|y - y_n\| = \|F_* y - F_n y_n\| &= \|F_* y - F_n y + F_n y - F_n y_n\| \\ &\leqslant \|F_* y - F_n y\| + \|F_n y - F_n y_n\|.\end{aligned}$$

From part 1

$$(1 - k)\|y - y_n\| \leqslant \|F_* y - F_n y\|$$

with $k < 1$. The thesis follows by observing that $(F_n y)(x)$ is equivalent to evaluate the integral in (12) by Filippi quadrature formula [11].

3. We observe that for all fixed $x \in [-1, 1]$ the linear functional

$$L_n(y, x) = \sum_{k=1}^{n} {}^{n}\gamma_k(x) y'(x_k) - \int_{-1}^{x} y'(t)\,\mathrm{d}t$$

vanishes if $y'(t)$ is a polynomial of degree less or equal to $n-1$, so the result follows from Peano's lemma [10], being

$$L_n(y, x) = y(x) - y_n(x). \qquad \square$$

5. The algorithms

In order to calculate the approximate solution of the initial value problem by (5) at $x \in [-1, 1]$ we need the values $y_n(x_k)$, $k = 1, \dots, n$. So we construct the following iterative algorithm.

Algorithm A1.

$$\begin{cases} Y^0_{n,j} = y_0 + (x_j + 1) f(-1, y_0) \\ Y^\nu_{n,j} = y_0 + \sum\limits_{k=1}^{n} a_{jk} f\big(k, Y^{\nu-1}_{n,k}\big), \quad \nu = 1, 2, \dots, \end{cases}$$

$j = 1, \dots n$, $a_{jk} = {}^n\gamma_k(x_j)$ and $Y^\nu_{n,j} = Y^\nu_{n,j}(x_j)$.

We observe that algorithm A1 can be obtained by applying the method (5) to Picard iterations

$$Y^\nu(x) = y_0 + \int_{-1}^{x} f\big(t, Y^{\nu-1}(t)\big)\, \mathrm{d}t.$$

In fact, if we approximate the function f by its truncated Chebyshev series of second kind and integrate, we obtain A1 (for details see [9]).

So the following proposition holds true.

Proposition 2.

$$\lim_{\nu \to \infty} Y^\nu_{n,j} = y_n(x_j) \quad \forall x \in [-1, 1],$$

where y_n is the solution of (5).

Being the $f(x_k, Y^{\nu-1}_{n,k})$ independent on x, we can save these values which might be needed to evaluate $y_n(x)$ at different x. This is a reasonable policy since the method might well be used to calculate approximations at distinct points.

Note 1. As A1 is equivalent to apply the (5) to Picard iterations, it is a waveform relaxation algorithm. In the case of large systems of differential equations the method (5) can be applied to more general WR techniques: Jacoby, Gauss–Seidel or SOR WR methods [3]. This interesting application will be study in the future and a comparison with WR Runge–Kutta methods [1] will be done.

Algorithm A2. An alternative way to calculate $y_n(x_r)$ is to consider the system

$$y_n(x_r) = y_0 + \sum_{k=1}^{n} {}^n\gamma_k(x_r) f\big(x_k, y_n(x_k)\big), \quad r = 1, \dots, n.$$

The system can be solved by iterative methods or Newton type methods. If we use a Jacobi type iterative method with initial value $y^{(0)}(x_i) = y_0 + (x_i + 1)f(-1, y_0)$ it coincides with the algorithm A1.

Note 2. The method (5) maybe extended to differential equations of second order both with boundary conditions and with initial values. Details are still under examination.

So far we described the method in a form which attempts to find the solution of a differential equation in the whole required range. This is not the only possible way of using it; for initial value problems it can also be used as step-by-step method with very little further work. The equation can be solved in a sub-range with one end-point at the initial point, then the solution can be used to provide starting values for the next range, and it can be repeated so much as necessary.

In the following we propose an implementation on the whole range $[x_0, x]$, where x_0 is the initial point and x the current point.

6. Some numerical examples

An automatic program to implement the algorithm A1 has been written to obtain the numerical solution of (2).

We fix the error tolerances ε_1 and ε_2, the upper bounds *it*-max and *n*-max on the number of iterations and of terms of the sum (5), respectively. In general we do not know in advance the minimum values of *it* and *n* which are necessary to achieve the desidered accuracy. So we start with an arbitrary value of n and then we increase n by 2 until $|Y_n^{\nu_1}(x) - Y_{n-2}^{\nu_2}(x)| < \varepsilon_1$ for ν_1 and ν_2 such that

$$\left|Y_n^{\nu_i}(x) - Y_n^{\nu_i}(x)\right| < \varepsilon_2, \quad i = 1, 2, \quad \text{or} \quad \left|Y_n^{\nu}(x) - y(x)\right| < \varepsilon_1$$

if the exact solution y is known, or the $(n + 1)$ term of the sum (5) is negligible. The program exits either with a value Y_n^{ν} of the solution which is allegedly correct to within the tolerance, or with a statement that the upper bounds have been achieved but not the tolerance and the "best" value of the solution determined in the process is given.

Now we will consider some numerical examples and we will make a comparison between the results obtained by algorithm A1, by GAM [2], GBS [14] and Sarafyan 4th order, 6-stage [18] algorithms.

$$\text{(i)} \quad \begin{cases} y' = x + y, \\ y(-1) = \dfrac{2}{\mathrm{e}} \end{cases} \quad \text{with solution} \quad y(x) = 2\mathrm{e}^x - x - 1;$$

(ii) $\begin{cases} y' = xy, \\ y(-1) = \sqrt{\mathrm{e}} \end{cases}$ with solution $y(x) = \mathrm{e}^{x^2/2}$;

(iii) $\begin{cases} y' = -y, \\ y(-1) = \mathrm{e} \end{cases}$ with solution $y(x) = \mathrm{e}^{-x}$;

(iv) $\begin{cases} y' = -z, & y(0) = 2, \\ z' = -3y - 2z, & z(0) = 2 \end{cases}$ with solutions $\begin{array}{l} y(x) = \mathrm{e}^{x} + \mathrm{e}^{-3x}, \\ z(x) = 3\mathrm{e}^{-3x} - \mathrm{e}^{x}. \end{array}$

All four problems are classical and discussed in literature [15,17,18]. After fixing ε_1 and ε_2, the initial value of n for algorithm A1, of the step size h_G for GAM, of h for Sarafyan algorithm (at each step h is to be halved if the tolerance is not achieved), and of predicted step size H of the last accepted step for GBS method, at some points $\overline{x}$ of the interval $[-1, 1]$ the approximate solutions obtained by algorithm A1, by GAM, GBS and Sarafyan methods have been calculated.

As we know the exact solution, the error can be calculated by computing the difference between the exact value at $\overline{x}$ and the approximated one.

In the following tables the corresponding absolute errors will be listed in a column, one below the other, in the following order: A1, GAM, GBS, Sarafyan. In the next columns the number of function evaluations (nfeval) are displayed and then the final values of n for algorithm A1 and of h for Sarafyan algorithm.

Note 3. The cost of algorithm A1 can be considerably reduced if a parallel computer is used.

We assume $\varepsilon_1 = \varepsilon_2 = \varepsilon$.

Problem (i). $\varepsilon = 10^{-11}, n = 10, h_G = H = 10^{-3}, h = 0.05.$

$\overline{x}$	Algorithm A1 GAM method Sarafyan method GBS method	nfeval	n h
1	2	3	4
-0.73	$9.09 \cdot 10^{-13}$	80	10
	$9.41 \cdot 10^{-14}$	125	
	$9.10 \cdot 10^{-13}$	234	0.0125
	$1.04 \cdot 10^{-14}$	85	
-0.29	$2.73 \cdot 10^{-12}$	110	10
	$1.47 \cdot 10^{-12}$	177	
	$8.19 \cdot 10^{-12}$	264	0.025
	$1.30 \cdot 10^{-13}$	136	

1	2	3	4
0.18	$9.09 \cdot 10^{-12}$	130	10
	$2.34 \cdot 10^{-12}$	222	
	$7.27 \cdot 10^{-12}$	432	0.025
	$3.17 \cdot 10^{-14}$	176	
0.86	$7.27 \cdot 10^{-12}$	352	12
	$4.68 \cdot 10^{-12}$	249	
	$7.28 \cdot 10^{-12}$	678	0.025
	$5.91 \cdot 10^{-12}$	280	

Problem (ii). $\varepsilon = 10^{-11}, n = 10, h_G = H = 10^{-3}, h = 0.025.$

$\overline{x}$	Algorithm A1 GAM method Sarafyan method GBS method	nfeval	n h
−0.71	$0^{(*)}$	80	10
	$6.26 \cdot 10^{-13}$	216	
	$7.27 \cdot 10^{-13}$	216	0.025
	$3.21 \cdot 10^{-13}$	85	
−0.27	$7.27 \cdot 10^{-12}$	100	10
	$9.73 \cdot 10^{-13}$	235	
	$8.19 \cdot 10^{-12}$	180	0.025
	$1.36 \cdot 10^{-13}$	136	
0.18	$5.45 \cdot 10^{-12}$	220	12
	$9.11 \cdot 10^{-13}$	307	
	$9.09 \cdot 10^{-12}$	288	0.025
	$1.64 \cdot 10^{-13}$	187	
0.73	$3.64 \cdot 10^{-12}$	360	14
	$1.18 \cdot 10^{-12}$	370	
	$1.82 \cdot 10^{-12}$	1254	0.0125
	$1.96 \cdot 10^{-13}$	374	

Problem (iii). $\varepsilon = 10^{-11}, n = 10, h_G = H = 10^{-3}, h = 0.025.$

$\overline{x}$	Algorithm A1 GAM method Sarafyan method GBS method	nfeval	n h
1	2	3	4
−0.73	$0^{(*)}$	80	10
	$1.19 \cdot 10^{-13}$	129	

1	2	3	4
−0.73	$7.27 \cdot 10^{-12}$	198	0.025
	$2.44 \cdot 10^{-14}$	85	
−0.43	$1.82 \cdot 10^{-12}$	110	10
	$3.56 \cdot 10^{-14}$	145	
	fails		
	$1.82 \cdot 10^{-14}$	136	
0.34	$2.73 \cdot 10^{-12}$	150	10
	$1.00 \cdot 10^{-14}$	234	
	$3.64 \cdot 10^{-12}$	324	0.025
	$5.81 \cdot 10^{-12}$	154	
0.95	$5.45 \cdot 10^{-12}$	374	12
	$6.53 \cdot 10^{-15}$	261	
	$4.09 \cdot 10^{-12}$	474	0.025
	$1.99 \cdot 10^{-14}$	227	

(*) If we work with 16 digits.

Problem (iv). $\varepsilon = 10^{-6}, n = 6, h_G = H = 10^{-3}, h = 0.5$.

Let $E_1 = |y_n(\overline{x}) - y(\overline{x})|$, $E_2 = |z_n(\overline{x}) - z(\overline{x})|$, where $y_n(\overline{x})$ and $z_n(\overline{x})$ are the numerical values of $y(\overline{x})$ and $z(\overline{x})$ calculated by the methods.

$\overline{x}$	Algorithm A1 GAM method Sarafyan method GBS method		nfeval	n h
	E_1	E_2		
0.24	$3.18 \cdot 10^{-7}$	$2.26 \cdot 10^{-7}$	48	6
	$1.29 \cdot 10^{-8}$	$3.68 \cdot 10^{-8}$	81	
	$5.32 \cdot 10^{-8}$	$1.61 \cdot 10^{-7}$	48	0.125
	$8.61 \cdot 10^{-8}$	$2.58 \cdot 10^{-7}$	56	
0.37	$2.90 \cdot 10^{-7}$	$1.26 \cdot 10^{-7}$	60	6
	$4.84 \cdot 10^{-8}$	$1.47 \cdot 10^{-7}$	87	
	$2.16 \cdot 10^{-8}$	$6.57 \cdot 10^{-8}$	72	0.125
	$1.74 \cdot 10^{-8}$	$5.24 \cdot 10^{-8}$	89	
0.6	$6.81 \cdot 10^{-7}$	$2.66 \cdot 10^{-7}$	72	6
	$1.10 \cdot 10^{-7}$	$3.31 \cdot 10^{-7}$	133	
	$1.03 \cdot 10^{-7}$	$3.91 \cdot 10^{-8}$	120	0.125
	$8.01 \cdot 10^{-8}$	$2.40 \cdot 10^{-7}$	108	
1.2	$3.93 \cdot 10^{-7}$	$9.52 \cdot 10^{-7}$	120	6
	$2.26 \cdot 10^{-8}$	$6.63 \cdot 10^{-8}$	181	
	$3.94 \cdot 10^{-8}$	$1.38 \cdot 10^{-8}$	216	0.125
	$3.98 \cdot 10^{-8}$	$1.19 \cdot 10^{-7}$	191	

References

[1] A. Bellen and M. Zennaro, The use of Runge–Kutta formulae in waweform relaxation methods, Appl. Numer. Math. 11 (1993) 95–114.

[2] L. Brugnano and D. Trigiante, *Solving Differential Problems by Multistep Initial Value Methods* (Gordon and Breach, Amsterdam, 1998).

[3] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations* (Clarendon Press, Oxford, 1995).

[4] J.C. Buthcher, *The Numerical Analysis of Ordinary Differential Equations: Runge–Kutta and General Linear Methods* (Wiley, Chichester, 1987).

[5] C.W. Clenshaw and A.R. Curtis, A method for numerical integration on an automatic computer, Numer. Math. 2 (1960) 197–205.

[6] C.W. Clenshaw and H.J. Norton, The solution of ordinary differential equations in Chebyshev series, Comput. J. 6 (1963) 88–92.

[7] F. Costabile, R. Luceri and M.I. Gualtieri, An operator of Bernstein type with applications to the indefinite integral and differential Cauchy problem, Suppl. Rend. Circolo Mat. Di Palermo Serie II 40 (1996).

[8] F. Costabile and A. Napoli, A method for global approximation of the Cauchy differential problem by Chebyshev polynomilas of the first kind, in: *XVI Congresso UMI*, Napoli, 1999.

[9] F. Costabile and A. Napoli, A method for global approximation of the Cauchy differential problem by Chebyshev polynomials of the second kind, LAN, Technical Report No. 37, Unical, Rende (1999).

[10] P. Davis, *Interpolation and Approximation* (Blaisdell, New York, 1975).

[11] H. Engels, *Numerical Quadrature and Cubature* (Academic Press, London, 1980).

[12] W.H. Enright et al., Interpolants for Runge–Kutta formulas, ACM Trans. Math. Software 12(3) (1986).

[13] L. Fox and I.B. Parker, *Chebyshev Polynomials in Numerical Analysis* (Oxford Univ. Press, Oxford, 1968).

[14] E. Hairer, S.P. Norsett and G. Wanner, *Solving Ordinary Differential Equations I* (Springer, Berlin, 1992).

[15] G. Hall, W.M. Enright and T.E. Hull, Detest: A program for comparing numerical methods for ordinary differential equations, Technical Report No. 60, University of Toronto (1973).

[16] P. Henrici, *Discrete-Variable Methods for Ordinary Differential Equations* (Wiley, New York, 1962).

[17] M. Nakashima, On pseudo-Runge–Kutta methods with 2 and 3 stages, RIMS, Kyoto University 18(3) (1982) 895–909.

[18] D. Sarafyan, Families of continuous approximate processes for the solution of ordinary differential equations, Comput. Math. Appl. 19 (1988) 887–894.

[19] K. Wright, Chebyshev collocation methods for ordinary differential equations, Comput. J. 6 (1964) 358–363.