Symbolic Derivation of Order Conditions for Hybrid Numerov-type methods solving $y'' = f(x, y)$

I. Th. Famelis [1]

TEI of Athens, Department of Mathematics, GR 122 10 Egaleo, Greece

Ch. Tsitouras [2]

TEI of Chalkis, Department of Applied Sciences,GR 34400 Psahna, Greece

Abstract

Numerov-type ODE Solvers are widely used for the numerical treatment of second order initial value problems. In this work we present a powerful and efficient symbolic code in Mathematica for the derivation of their order conditions and principal truncation error terms. The relative tree theory for such order conditions is presented along with the elements of combinatorial mathematics, partitions of integer numbers and computer algebra which are the basis of the implementation of the symbolic code.

Key words: Numerov-type Methods, Order Conditions, Rooted trees, integer partitions, truncation error, MATHEMATICA.

1 Introduction

Second order Ordinary Differential Equations (ODEs) that do not involve y',

$$y'' = f(t, y),\ y(t_0) = y^{[0]},\ y'(t_0) = y'^{[0]}, \tag{1}$$

where $f : \Re^N \longrightarrow \Re^N$ and $y^{[0]},\ y'^{[0]} \in \Re^N$, are widely used to model physical problems. Thus, methods for the numerical treatment of such ODEs are of great importance. There exist various classes of methods for the numerical

[1] E-Mail: `ifamelis@teiath.gr`, URL address: `http://math.teiath.gr/ifamelis/`

[2] E-Mail: `tsitoura@teihal.gr`, URL address: `http://users.ntua.gr/tsitoura/`

solution of such problems. For instance, problem (1) can be treated using a Runge-Kutta Nyström method, or if we transform it in a system of first order ODEs, it can be solved by a Runge-Kutta method [13]. One of the most widely used methods for solving (1) is the Numerov which attains fourth algebraic and sixth phase-lag order. This method is implicit and its implementation involve computations of Jacobians and solutions of non-linear systems of equations, [8]. So, many authors proposed explicit modifications of the Numerov method which are usually called hybrid or two-step Numerov-type methods. The construction of such methods require the derivation and the solution of equations called "order conditions". Such a procedure is a tedious task since the number of order conditions to be derived and then solved increases as the order of a method increases. The order conditions are nonlinear expressions which involve the methods coefficients . So, constructing a specific method requires the solution of a system of nonlinear equations. For high order methods, we usually use symbolic computations to solve some of the equations resulting in exact expressions for coefficients that involve other coefficients. Then, the numerical treatment of the remaining system, usually by powerful minimization algorithms, yields solutions that fail to satisfy the order conditions with an acceptable accuracy. Nevertheless, the outcome can be useful as a set of initial values for the next phase of our work. Computer Algebra systems, such as Mathematica, provide the capability to apply numerical methods asking the results to satisfy a lot more than sixteen digits of accuracy. So, using the numerical results as good guesses we hope and usually manage to specify coefficients that satisfy the order conditions with very high acceptable accuracy. Therefor, for both the derivation and the solution of the order conditions the use of a Computer Algebra systems is needed.

In the literature, computer codes for generating Runge–Kutta trees, order conditions and truncation order coefficients can be found. Keipers [17] program, written in Mathematica language, was probably the first but it was limited in deriving low order conditions. Hoseas [16] presented a code named RKTEC, written in ANSI C, which is available from `Netlib`. This was based on a recurrence procedure due to Albrecht [1] that generates order conditions. A new perspective was introduced by Harisson [15] and Papakostas [23] as it was acknowledged in [28]. They suggested the use of tensor notation which resulted in very interesting symbolic codes. That early package due to Papakostas had been a powerful tool for the research work of our group ([26,35,25,31]) when truncation error calculations were needed. It can be asked by e-mail from the present authors. Sofroniou [28] as well, has published an integrated package for deriving Runge–Kutta order conditions. Then Papakostas, in his Ph.D. Thesis [24], proposed that in such codes the derivation of trees should be avoided. Following his suggestions, we have presented [10] a very efficient code for the derivation of Runge–Kutta order conditions. Finally, a relative Matlab code is due to Cameron [6].

For the class of Runge–Kutta–Nyström methods a first code to generate RKN trees is due to Okunbor [20]. This specific code, which was based on the Keipers program philosophy fails at high orders. Following the same lines of our previous work our team presented a powerful and efficient symbolic package for the derivation of Runge–Kutta–Nyström [**?**] order conditions and principal truncation error terms. Here, we present the first symbolic package for the derivation of order conditions and principal local truncation error terms for two-step Numerov-type methods.

In the following section we outline the theory of construction of two-step Numerov-type methods. Then we present the elements of Combinatorial Mathematics and Tree theory which have been used to approach the construction of our symbolic program. In our approach, constructing the trees as matrix products results in a very fast, neat and cheap in memory usage code.

2 Hybrid Numerov Methods

Two-step Numerov-type methods proceed to the evaluation of $y^{[k+1]}$ as an estimation of $y\left(t_{k+1}\right) = y\left(t_k + h\right)$, according to the following formulae:

$$
\begin{aligned}
Y^{[1]} &= (1-c_1)y^{[k]} + c_1 y^{[k-1]} + h^2 \sum_{j=1}^{s} a_{1j}\ f_{k-c_1} \\
f_{k-c_1} &= f\left(t_k - c_1 h, Y^{[1]}\right) \\
Y^{[2]} &= (1-c_2)y^{[k]} + c_2 y^{[k-1]} + h^2 \sum_{j=1}^{s} a_{2j}\ f_{k-c_2} \\
f_{k-c_2} &= f\left(t_k - c_2 h, Y^{[2]}\right) \\
\cdots & \quad \cdots \\
Y^{[s]} &= (1-c_s)y^{[k]} + c_s y^{[k-1]} + h^2 \sum_{j=1}^{s} a_{sj}\ f_{k-c_s} \\
f_{k+c_s} &= f\left(t_k - c_s h, Y^{[s]}\right) \\
y^{[k+1]} &= 2y^{[k]} - y^{[k-1]} + h^2 \sum_{j=1}^{s} b_j\ f_{k-c_s},
\end{aligned}
\tag{2}
$$

where $h = t_{k+1} - t_k = t_k - t_{k-1} = \cdots = t_1 - t_0$ and the vectors $y^{[k]}$ and $y^{[k-1]}$ are previous step approximations of $y(t_k)$ and $y(t_k - h)$ respectively.

In vector notation, for an autonomous system $y'' = f(y)$, an s-stage Numerov-type method takes the form

$$\begin{aligned} y^{[k+1]} &= 2y^{[k]} - y^{[k-1]} + h^2 \cdot (b \otimes I_s) \cdot f(Y) \\ Y &= (e-c) \otimes y^{[k]} + c \otimes y^{[k-1]} + h^2 \cdot (A \otimes I_s) \cdot f(Y) \end{aligned} \tag{3}$$

with $I_s \in \Re^{s\times s}$ the identity matrix, $A = [a_{ij}] \in \Re^{s\times s}, b^T = [b_i] \in \Re^s, c = [c_i] \in \Re^s$ the coefficient matrices of the method and

$$e = [1\ 1\ \cdots 1]^T \in \Re^s.$$

For this case the independent variable t can be considered as an extra component of y, setting

$$y''_{N+1} = 0,\ y^{[0]}_{N+1} = t_0,\ y'^{[0]}_{N+1} = 1.$$

Using the Butcher tableau notation [2,3] the coefficients of such a method can be presented by the table,

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}.$$

When the matrix A is strictly lower triangular the method is explicit and can be applied directly. Otherwise, the method is implicit and a system of nonlinear equations has to be solved in each step.

Local Truncation Error (LTE) measures the methods approximation error $\| y^{[k+1]} - y(t_{k+1}) \|$ assuming the previous two step values are exact. Taking the Taylor expansions of (3)and its its theoretical correspondence and subtracting, LTE is derived. The resulted quantity is a series of the form

$$h^2 T_{11} F_{11} + h^3 T_{21} F_{21} + h^4 \cdot (T_{31} F_{31} + T_{32} F_{32}) + h^5 \cdot (T_{41} F_{41} + T_{42} F_{42} + T_{43} F_{43}) + \cdots$$

where T_{ij}'s are the truncation error coefficients depending exclusively on the method coefficients A, b, c. Moreover, F_{ij}'s are elementary differentials with respect to y', f and $f^{(k)} = \frac{\partial^k f}{\partial t^k}$, $k = 1, 2, ...$ [11] are problem depended.

For an order p method the coefficients of $h^2, h^3, ..., h^{p+1}$ have to be zero. So, for a fourth order method

$$T_{11} = T_{21} = T_{31} = T_{32} = T_{41} = T_{42} = T_{43} = 0,$$

have to be satisfied. These equations are called order conditions (o.c.). The number of o.c. needed to achieve a desired order is presented in Table 1. Hybrid Numerov and Runge Kutta Nyström methods of the same order need equal number of conditions to be fulfilled.

Table 1
Number of order conditions (o.c.) to achieve order p.

Order p	1	2	3	4	5	6	7	8	9	10	11
number of o.c.	1	1	2	3	6	10	20	36	72	137	275
cumulative number of o.c.	1	2	4	7	13	23	43	79	151	288	563

Table 2
Terms of truncation error coefficients of 1−st to 5−th order.

order	equations	
1	$T_{11} = be - 1$	
2	$T_{21} = b \cdot c$	
3	$T_{31} = \frac{1}{2} b \cdot c + b \cdot A \cdot e - \frac{1}{12}$,	$T_{32} = \frac{1}{2} b \cdot c^2 - \frac{1}{12}$
4	$T_{41} = b \cdot A \cdot c + \frac{1}{6} b \cdot c$,	$T_{42} = \frac{1}{2} b \cdot c^2 + b \cdot (c * A \cdot e)$,
	$T_{43} = \frac{1}{6} b \cdot c^3$	
5	$T_{51} = \frac{1}{24} b \cdot c + \frac{1}{2} b \cdot A \cdot c + b \cdot A^2 \cdot e - \frac{1}{360}$,	
	$T_{52} = \frac{1}{24} b \cdot c + \frac{1}{2} b \cdot A \cdot c^2 - \frac{1}{360}$,	
	$T_{53} = \frac{1}{6} b \cdot c^2 + b \cdot (c * A \cdot c) - \frac{1}{90}$,	
	$T_{54} = \frac{1}{8} b \cdot c^2 + \frac{1}{2} b \cdot (c * A \cdot e) + \frac{1}{2} b \cdot (A \cdot e)^2 - \frac{1}{120}$,	
	$T_{55} = \frac{1}{4} b \cdot c^3 + \frac{1}{2} b \cdot (c^2 * A \cdot e) - \frac{1}{60}$	$T_{56} = \frac{1}{24} b \cdot c^4 - \frac{1}{360}$

For instance, in order to construct a method of order five the thirteen order conditions $T_{ij} = 0$ presented in Table-2 should be considered. In this table we set

$$c^i = [c_1^i,\ c_2^i, \cdots, c_s^i]^T,$$

while the operation "*" may be understood as component-wise multiplication:

$$[u_1\ \ u_2 \cdots u_n]^T * [v_1\ \ v_2 \cdots v_n]^T = [u_1 v_1\ \ u_2 v_2 \cdots u_n v_n]^T.$$

This operation has the less priority. Parentheses, powers and dot products are always evaluated before "*".

Observe that for a p−th order method the principal local truncation error term is multiplied by h^{p+2}. So, the method has truncation error of $O(h^{p+2})$ and not $O(h^{p+1})$. This is due to accuracy reduction from the non-existence of y' in the formulas (3), see Hairer et. al. [13, p. 468].

It must be noticed that the presentation in Table-2 can be simplified assuming that lower order conditions are satisfied. For an order p method $(1 \leq p \leq 5)$, when all the lower order conditions are fulfilled, the simplified expressions are

Table 3
Simplified terms of truncation error coefficients of 1−st to 5−th order.

order	equations		
1	$T_{11} = b \cdot e - 1$		
2	$T_{21} = b \cdot c$		
3	$T_{31} = b \cdot A \cdot e - \frac{1}{12}$,	$T_{32} = \frac{b \cdot c^2}{2} - \frac{1}{12}$	
4	$T_{41} = b \cdot A \cdot c$,	$T_{42} = b \cdot (c * A \cdot e) + \frac{1}{12}$,	$T_{43} = \frac{b \cdot c^3}{6}$
5	$T_{51} = b \cdot A^2 \cdot e - \frac{1}{360}$,	$T_{52} = \frac{1}{2} b \cdot A \cdot c^2 - \frac{1}{360}$,	$T_{53} = b \cdot (c * A \cdot c) + \frac{1}{60}$
	$T_{54} = \frac{1}{2} b \cdot (A \cdot e)^2 - \frac{7}{240}$,	$T_{55} = \frac{1}{2} b \cdot (c^2 * A \cdot e) - \frac{1}{60}$,	$T_{56} = \frac{b \cdot c^4}{24} - \frac{1}{360}$

now listed in Table 3. For example, for a fourth order method

$$T_{21} = b \cdot c = 0, \text{ and } T_{41} = b \cdot A \cdot c = 0$$

hold, so we conclude that

$$T_{51} = \frac{1}{24} b \cdot c + \frac{1}{2} b \cdot A \cdot c + b \cdot A^2 \cdot e - \frac{1}{360} = b \cdot A^2 \cdot e - \frac{1}{360}. \tag{4}$$

Whereas, in the case of studying a third order method, which implies that $T_{11} = T_{21} = T_{31} = T_{32} = 0$ and T_{41} generally not zero, (4) would not correspond to the truncation error coefficient term T_{51} of h^6. In such a case

$$T_{51} = \frac{1}{2} b \cdot A \cdot c + b \cdot A^2 \cdot e - \frac{1}{360}$$

should be considered.

Another interesting issue is to keep the magnitude of the principal truncation error term Euclidean norm small. Therefor, for a fourth order method it is important to have the value

$$\|T^{(5)}\|_2 = \sqrt{T_{5,1}^2 + T_{5,2}^2 + \cdots + T_{5,6}^2},$$

as small as possible. The set $T^{(5)}$ collects all the fifth order truncation error coefficients. Similarly $T^{(1)} = \{T_{11}\}$, while

$$T^{(2)} = \{T_{21}\},\ T^{(3)} = \{T_{31}, T_{32}\}, \cdots .$$

The number of equations that should be fulfilled is reduced assuming that one or more of the following simplifying assumptions hold

$$\begin{aligned} A\cdot e &= \tfrac{1}{2}\,(c^2-c) \\ A\cdot c &= \tfrac{1}{6}\,(c^3-c) \\ A\cdot c^2 &= \tfrac{1}{12}\,(c^4-c) \\ A\cdot c^3 &= \tfrac{1}{20}\,(c^5-c) \\ &\ldots\ldots \end{aligned} \tag{5}$$

When adopting the first simplifying assumptions, the order conditions containing expression $A^i\cdot e,\ i=1,2,3..$, coincide with others and vanish. For example, it can be easily seen that only one of the elements from $T^{(3)}$ is needed since

$$T_{31}=b\cdot A\cdot e-\frac{1}{12}=b\cdot\frac{1}{2}\cdot(c^2-c)-\frac{1}{12}=\frac{1}{2}\,b\cdot c^2-\frac{1}{2}\,b\cdot c-\frac{1}{12}=\frac{1}{2}\,b\cdot c^2-\frac{1}{12}=T_{32}$$

In a similar way after using the second row of (5) we may discard all equations containing $A^i\cdot c,\ i>0$.

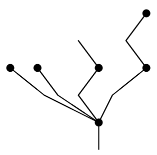
When conditions (5) are applied in an implicit method its nodes are interpolatory points of the proper order [32]. On the other hand for explicit methods, assumptions (5) correspond to the concept of stage order of Runge-Kutta methods, see [33]. The early Numerov–type methods were constructed without taking consideration conditions (5)resulting interior nodes of an specific algebraic order, say $p-2$. Then a method of order p was derived by using an interpolatory approach. This useless procedure was our motive for introducing methods of the form (2) in [29].

3 Tree Theory for Order Conditions

In the 60's, J. C. Butcher [4] established a theory, based in trees, to derive the order conditions of a Runge–Kutta method. The extension of the tree theory for the case of Runge–Kutta–Nyström methods can be found in [13] where the SN-trees (Special Nyström trees) were defined. For the derivation of order conditions for two-step methods, J. P. Coleman [9] chooses a slightly different family of trees called $\top_2$. These are the SN-Trees grafted onto a meagre root.

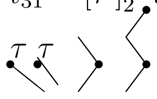
$\top_2$ rooted trees have two kind of vertices, meagre vertices which are drawn as a point and fat vertices which are drawn as a larger dot. The root of such a tree is a meagre vertex which is connected to a single fat vertex. The branches of a $\top_2$ are connected to that fat vertex. Let $\emptyset$ be the empty tree, τ' the single meagre vertex tree and τ the tree . By using these three elements and recursion we can generate the whole set $\top_2$ of trees.

A tree $t \in \top_2$ can be written as $t = [t_1, t_2, \cdots, t_m]_2$ where $t_1, t_2, \ldots, t_m \in \top_2$ are its branches. So, t is obtained by connecting the roots of $t_1, t_2, \ldots, t_m$ to a new fat vertex, and then connecting that vertex to a new meagre root. For example the tree



can be written as $t = [\tau, \tau, t_{31}, t_{41}]_2 = [\tau^2, t_{31}, t_{41}]_2$ where

$t_{31} = [\tau']_2$, $t_{41} = [\tau]_2$



Now, the following functions can be defined on $t = [t_1, t_2, \ldots, t_m]_2 \in \top_2$:

• Order $r(t)$:

$$r(t) = 2 + r(t_1) + \ldots + r(t_m)$$

with $r(\emptyset) = 0$,$r(\tau') = 1$ and $r(\tau) = 2$

• Symmetry $\sigma(t)$:

$$\sigma\left([t_1^{n_1}, t_2^{n_2}, \cdots, t_k^{n_k}]_2\right) = n_1! \cdots n_k! \sigma(t_1)^{n_1} \cdots \sigma(t_k)^{n_k}$$

with $\sigma(\emptyset) = 1$,$\sigma(\tau') = 1$ and $\sigma(\tau) = 1$

•$\Psi(t)$:

$$\Psi(t) = c + A\Psi''(t)$$

if $r(t) \geq 2$ with $\psi(\tau') = e$ and $\psi(\emptyset) = c$ where $e = [1, 1, \cdots, 1]^T \in \Re^s$.

•$\Psi''(t)$:

$$\Psi''(t) = r(t)r(t-1)\prod_{i=1}^{m} \Psi(t_i)$$

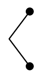
The following theorem relates the $\top_2$ trees to the order conditions that must hold so a method to attain order p. Its proof is based on B-series theory and can be found in [9].

Theorem 1 *A two-step Numerov-type method is of order p if and only if*

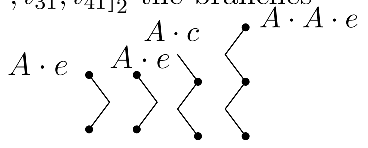
$$b^T \Psi''(t) = 1 + (-1)^{r(t)}$$

for every $t \in \top_2$ with $r(t) \leq p + 1$.

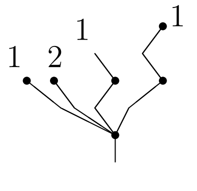
Each order condition involves expressions which are linear in the components of b and nonlinear in the components of A and c. There exists a one-to-one relationship between the set of order p conditions and the $\top_2$ rooted trees with $p+1$ nodes. Each tree can be correlated with a specific expression in its corresponding order condition from which all the other expressions can be produced. All these correspondences are presented in Table-4.

By following simple rules it is not hard to form the corresponding order condition. The root is b, each terminating meagre vertex is c and the tree formation  corresponds to A. Branches that are grafted together have their matrix representations multiplied component-wise. For each long branch consisting of other tree elements in a row, their matrix representations are multiplied by using the usual matrix multiplication. Finally, for branches that have terminating meagre nodes their expressions are multiplied by e.

So, in our example $t = [\tau^2, t_{31}, t_{41}]_2$ the branches



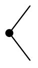
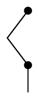
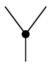
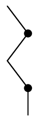
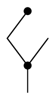
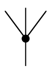
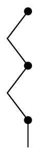
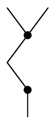
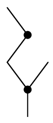
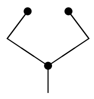
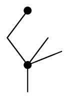
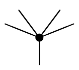
are grafted together in a fat vertex so their expressions are multiplied component-wise. Then they are connected to the root. So the corresponding matrix expression $b \cdot ((A \cdot e)^2 * (A \cdot c) * (A^2 \cdot e))$. This expression is multiplied by one over the symmetry number of the tree. The symmetry of a tree can be easily calculated if we label all the terminating branches with a number. Groups are formed containing the identical branches grafted to the fat vertex connected to the root. Then the elements of each group are numbered. The corresponding symmetry value is the product of all these numbers that label the branches.



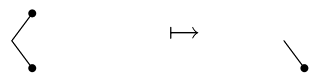
For our example the symmetry value is $\frac{1}{1 \times 1 \times 2 \times 1} = \frac{1}{2}$.

So far, given a tree, the way to form its corresponding expression has been

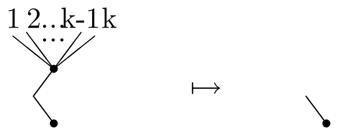
Table 4
Correspondence of $\mathbb{T}_2$ trees to 1−st to 5−th order conditions.

order	Order Condition $\leftrightarrow$ Tree	Tree $\leftrightarrow$ Corresponding Expression	
1	$T_{11} = b \cdot e - 1 \leftrightarrow t_{21} = \tau$		$\leftrightarrow b \cdot e$
2	$T_{21} = b \cdot c \leftrightarrow t_{31} = [\tau']_2$		$\leftrightarrow b \cdot c$
3	$T_{31} = \frac{1}{2} b \cdot c + b \cdot A \cdot e - \frac{1}{12} \leftrightarrow t_{41} = [\tau]_2\,,$		$\leftrightarrow b \cdot A \cdot e$
	$T_{32} = \frac{1}{2} b \cdot c^2 - \frac{1}{12} \leftrightarrow t_{42} = [\tau', \tau']_2$		$\leftrightarrow b \cdot c^2$
4	$T_{41} = b \cdot A \cdot c + \frac{1}{6} b \cdot c \leftrightarrow t_{51} = [t_{31}]_2$		$\leftrightarrow b \cdot A \cdot c$
	$T_{42} = \frac{1}{2} b \cdot c^2 + b \cdot (c * A \cdot e) \leftrightarrow t_{52} = [\tau', \tau]_2$		$\leftrightarrow b \cdot (c * A \cdot e)$
	$T_{43} = \frac{1}{6} b \cdot c^3 = \leftrightarrow t_{53} = [\tau', \tau', \tau']_2$		$\leftrightarrow b \cdot c^3$
5	$T_{51} = \frac{1}{24} b \cdot c + \frac{1}{2} b \cdot A \cdot c + b \cdot A^2 \cdot e - \frac{1}{360} \leftrightarrow$ $t_{61} = [t_{41}]_2,$		$\leftrightarrow b \cdot A^2 \cdot e$
	$T_{52} = \frac{1}{24} b \cdot c + \frac{1}{2} b \cdot A \cdot c^2 - \frac{1}{360} \leftrightarrow t_{62} = [t_{42}]_2$		$\leftrightarrow b \cdot A \cdot c^2$
	$T_{53} = \frac{1}{6} b \cdot c^2 + b \cdot (c * A \cdot c) - \frac{1}{90},$		$\leftrightarrow b \cdot (c * A \cdot c)$
	$T_{54} = \frac{1}{8} b \cdot c^2 + \frac{1}{2} b \cdot (c * A \cdot e) + \frac{1}{2} b \cdot (A \cdot e)^2 -$ $\frac{1}{120} \leftrightarrow t_{64} = [\tau, \tau]_2$		$\leftrightarrow b(A \cdot e)^2$
	$T_{55} = \frac{1}{4} b \cdot c^3 + \frac{1}{2} b \cdot (c^2 * A \cdot e) - \frac{1}{60} \leftrightarrow t_{65} =$ $[\tau', \tau', \tau]_2$		$\leftrightarrow b \cdot (c^2 * A \cdot e)$
	$T_{56} = \frac{1}{24} b \cdot c^4 - \frac{1}{360} \leftrightarrow t_{66} = [\tau', \tau', \tau', \tau']_2$		$\leftrightarrow b \cdot c^4$

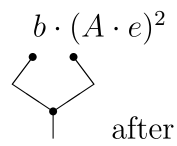
considered. After performing a kind of "trimming", the same tree can give the rest of the lower order trees contributing expressions involved in its order condition. Two kinds of trimming are considered. Every branch, taking the form on the left is trimmed to take the form of the branch on the right



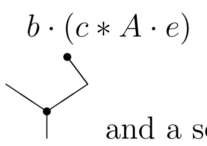
and



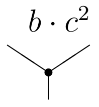
By taking all possible combinations each trimming produces a new tree. For example the corresponding tree in $T_{54} = \frac{1}{8}b\cdot c^2 + \frac{1}{2}b\cdot(c*A\cdot e) + \frac{1}{2}b\cdot(A\cdot e)^2 - \frac{1}{120}$ is



after a first trimming we get

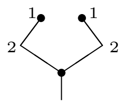


and a second



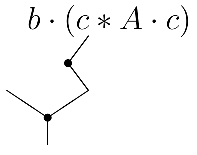
.

In order to find the coefficient of the trimmed tree expression, each vertex of the branch trimmed is characterized with the order of the tree produced assuming that this particular vertex is a root of a tree containing the remaining vertexes. Let *prod* be the product of these orders, then the expression coefficient is one over the symmetry value of the initial tree multiplied by the number of the trimmings resulting the specific tree and by one over *prod* . For example, for T_{54}

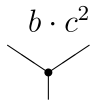


we find that the coefficient for the first trimming is $\frac{1}{2}\times 2\times\frac{1}{1\times 2} = \frac{1}{2}$ (note that we can get $b\cdot(c*Ae)$ after two different trimmings) and for the second it is $\frac{1}{2}\times\frac{1}{1\times2\times1\times2} = \frac{1}{8}$.

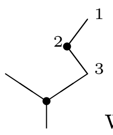
Whereas in $T_{53} = \frac{1}{6}b\cdot c^2 + b\cdot(c*A\cdot c) - \frac{1}{90}$ the tree is



and after the trimming we get



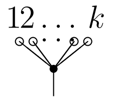
.

Now for T_{53} 

we find that the coefficient for the trimming is $1 \times \frac{1}{1\times 2\times 3}$

4 The Symbolic Code

As we have mentioned in the previous section, a tree with $p+1$ nodes (of order $p+1$) can be constructed by taking trees with cumulative order $p-1$ obtained and connecting their roots to a new fat vertex and then connecting that vertex to a new meagre root. In other words, the set of trees with $p+1$ nodes can be formed by taking combinations with repetition of k trees with cumulative order $p-1$.



If we set $\mathcal{T}^i = \{t_{i\#} | where\ t_{i\#}\ a\ \top_2\ rooted\ tree\ of\ order\ i\}$ we have to form combination of objects with repetition to produce the products $t^{i_1}_{\pi_1\#} t^{i_2}_{\pi_2\#} \cdots t^{i_k}_{\pi_k\#}$ where $t_{\pi_j\#} \in \mathcal{T}^{\pi_j}$ and $i_1\pi_1 + i_2\pi_2 + \cdots + i_k\pi_k = p-1$, $k = 1, 2, \ldots, p-1$.

This connects our problem with the set of unrestricted partitions of an integer. For example, an unrestricted partition of 5 is $1, 1, 1, 2$. This is usually written as $1^3 2$. So, an unrestricted partition of p has the form $\pi_1^{i_1}\pi_2^{i_2}\cdots\pi_k^{i_k}$ where $i_1\pi_1 + i_2\pi_2 + \cdots + i_k\pi_k = p$. This is a notation similar to the one used for the trees.

In conclusion, in order to construct all the rooted trees of order $p+1$ we have to find all the unrestricted partitions of $p-1$ and for each of them to form all the corresponding combinations with repetition $t^{i_1}_{\pi_1\#} t^{i_2}_{\pi_2\#} \cdots t^{i_k}_{\pi_k\#}$ selecting $t_{\pi_j\#}$ from T^{π_j}. In a programming point of view the best way is to work by forming the matrix notation products of the expressions involving the method coefficients instead of constructing the corresponding trees.

Hence, in our code the t_{ij} are not the trees but the corresponding matrix multiplication expressions $\Psi(t_{ij})$. Moreover the outer products formed are based on pointwise multiplication.

The proposed code provides two functions. The function **BCO** which gives the order conditions in a simplified form as far as the numeric coefficients of the expressions is concerned and the function **TCO** which returns the full form of the terms of the principal local truncation error coefficient.

In the following example the outcome is the lists with elements order 1 to 5 order conditions.

```
In[1]:=<<numer

In[2]:=BCO[a, b, c, e, 1]
Out[2]:={-1+b.e}

In[3]:=BCO[a, b, c, e, 2]
Out[3]:={b.c}

In[4]:=BCO[a, b, c, e, 3]
Out[4]:={-(1/6)+b.c+2*b.a.e,-(1/12)+b.c^2/2}

In[5]:=BCO[a, b, c, e, 4]
Out[5]:={b.c+6*b.a.c,b.c^2+2*b.(c*a.e),b.c^3/6}

In[6]:=BCO[a, b, c, e, 5]
Out[6]={-(1/15)+b.c+12*b.a.c+24*b.a.a.e,
-(1/30)+(1/2)*(b.c+12*b.a.c^2),-(1/15)+b.c^2+6*b.(c*a.c),
-(1/30)+(1/2)*(b.c^2+4*b.(c*a.e)+4*b.(a.e)^2),
-(1/30)+(1/2)*(b.c^3+2*b.(c^2*a.e)),-(1/360)+b.c^4/24}
```

The simplified order 5 conditions, assuming that the lower order conditions are fulfilled, can be produced by using the function `BCOSIM` which we define as follows.

```
In[7]:=BCOSIM[a_, b_, c_, e_, 1] := {b.e - 1};

In[8]:=BCOSIM[a_, b_, c_, e_, orderr_] :=
       Expand[TCO[a, b, c, e, orderr] /.
                 Flatten[Table[Flatten[Map[Solve,
                          Map[# == 0 &, BCOSIM[a, b, c, e, i]]]],
                          {i, 1,orderr - 1}
                               ]
                          ]
             ];
```

Now for order 5 we have:

```
In[9]:=BCOSIM[a,b,c,e,5]
Out[9]:={-(1/360)+b.a.a.e,-(1/360)+(1/2)*b.a.c^2,1/60+b.(c*a.c),
-(7/240)+(1/2)*b.(a.e)^2,-(1/60)+(1/2)*b.(c^2*a.e),
-(1/360)+b.c^4/24}
```

For a method of order 4 the coefficient terms of the principal local truncation order coefficient $T^{(5)}$ can be given by:

Table 5
Number of trees and times of evaluation

	Two-step			RK	
order	number of equations	number of trees	Time (sec)	number of equations	Time (sec)
11	275	1551	0.21	1842	0.14
12	541	3520	0.47	4766	0.34
13	1098	8262	1.09	12486	0.69
14	2208	19114	2.51	32973	1.64
15	4521	45049	6.05	87811	4.25
16	9420	105671	14.7	235381	11.1
17	19084	250376	34.5	634847	30.0
18	39451	593033	81.4	1721159	82.9

```
In[10]:=TCO[a,b,c,e,5]
Out[10]:={-(1/360)+b.c/24+b.a.c/2+b.a.a.e,
-(1/360)+b.c/24+(1/2)*b.a.c^2,-(1/90)+b.c^2/6+b.(c*a.c),
-(1/120)+b.c^2/8+(1/2)*b.(c*a.e)+(1/2)*b.(a.e)^2,
-(1/60)+b.c^3/4+(1/2)*b.(c^2*a.e),-(1/360)+b.c^4/24}
```

In all the calls of `BCO` and `TCO`, a, b, c, e can be Mathematica symbols or matrices with numeric entries.

The algorithm presented here is competitive to the one given in [10] for RK methods. In Table 5 we present computation times for our algorithms for two-step Numerov-type and RK methods and the corresponding number of trees and order conditions for various orders. In particular, we have used the RK function `RKTrunc` implemented in [10] and `BOC` function of our new package. As we have mentioned order conditions for the two step methods involve more than one tree expressions whereas for the RK methods the number of produced trees and the number of order conditions coincide.

The runs were performed in the Mathematica 5.2 environment on a Pentium 3.2 MHz system having 1 GByte RAM memory which was running Windows XP–SP2 Operating System.

5 Conclusions

In this paper we have presented, for the first time, a very efficient symbolic code for the derivation of two-step Numerov-type method order conditions and principal local truncation error coefficients. The code is fast and economical in computer memory. Finally, another remarkable fact is that the source code of the proposed package covers less than two journal pages and this helps in the direction of better and easier understanding.

Appendix

The Mathematica package implements the one and a half A4 pages code.

```
BeginPackage[ "NUMER‘", {"DiscreteMath‘Combinatorica‘"}];
Clear["NUMER‘*" ]
BCO::usage = "  BCO[a,b,c,e,order] returns order
conditions of Numerov type methods. "
TCO::usage = "
TCO[a,b,c,e,order] returns trunc error terms of Numerov type
methods. " Begin["‘Private‘"]; Clear[ "NUMER‘Private‘*" ];

BCO[a_,b_,c_,e_,1]:={b.e-1}
BCO[a_,b_,c_,e_,order_]:=1/S[order+1]*Map[g,Map[Distribute[#] &,
Map[Expand[#] &, TO[a,b,c,e,order+1], order+1],
order+1]]-1/S[order+1]*(1+(-1)^(order+1))/((order+1)*order)
TCO[a_,b_,c_,e_,1]:={b.e-1};
TCO[a_,b_,c_,e_,order_]:=Module[{aa,bb,cc,ee,tr},
    Off[First::normal];
    tr=Expand[1/Append[Delete[Map[First, Map[Last, SSON[aa,bb,cc, ee,order]]],
    -1], 1] BCO[a, b, c, e, order]];
    On[First::normal];
    Return[tr]];
(*--------------------------------------------------------------------------*)
RunLengthEncode[x_List] := (Through[{First, Length}[#1]] &) /@
Split[x];

Combinations[list_, num_] :=
 Module[{i},
        Table[Map[Prepend[#, list[[i]]]&,
                  Flatten[Combinations[list, num - 1]
                            [[Array[Identity, Length[list] - i + 1, i]]], 1
                         ], {1}
                ],
              {i, 1, Length[list]}
```

```
             ]]/;  (num > 1) ;
Combinations[list_, 1] := Compinations[list, 1] = Map[{{#}}&,
list];

Combinations2[list_, num_] :=
 Apply[Times, Flatten[Combinations[list, num], 1], {1} ]/;  (num > 1);
Combinations2[list_, 1] := list;
(*-----------------------------------------------------------------------*)
f[x_] := Apply[Times, Extract[
      x, Map[Append[#, 1] &, Position[x,
        Times[_Integer, _]]]]]*ReplacePart[x, 1,
          Map[Append[#, 1] &, Position[x, Times[_Integer, _]]]];
 g[y_] := Map[f, y, 1];
(*-----------------------------------------------------------------------*)
T[a_,c_,e_,0] = {e}; T[a_,c_,e_,1] = {c}; T[a_,c_,e_,2] = {2* a.e+
c}; T[a_,c_,e_,order_] := T[a,c,e,order] =
 Module[{temp},
        temp = Map[Combinations2[T[a,c,e,#[[1]] ], #[[2]]]&,
        Map[RunLengthEncode[#] &, Partitions[order-2], {1}], {2}];
        temp = Map[CoverList[#]&, temp, {3}];
        temp = Apply[MyOuter, temp, {1}];
        temp = Flatten[temp, 1];
        temp = temp /. CoverList[every_] -> every;
        temp = (order-1)*order *Map[ (a . # )&, temp, {1}];
        temp = Map[(# + c)&, temp, {1}];
        temp=temp ];
MyOuter[lists__] := Flatten[Outer[Times, lists], Length[{lists}] -
1]; TO[a_,b_,c_,e_,1] = {b.e-1};    TO[a_,b_,c_,e_,2] = {b.c};
TO[a_,b_,c_,e_,order_] := TO[a,b,c,e,order] =
 Module[{temp},
        temp = Map[Combinations2[T[a,c,e,#[[1]] ], #[[2]]]&,
        Map[RunLengthEncode[#] &, Partitions[order-2], {1}], {2}];
        temp = Map[CoverList[#]&, temp, {3}];
        temp = Apply[MyOuter, temp, {1}];
        temp = Flatten[temp, 1];
        temp = temp /. CoverList[every_] -> every;
        temp = Map[(b . #)&, temp, {1}]]
(*-----------------------------------------------------------------------*)
S[1] = {1}; S[2]={1}; S[order_] := S[order] =
 Module[{temp},
    temp = Map[Combinations2[MapIndexed[ff, S[#[[1]]]], #[[2]]] &,
    Map[RunLengthEncode[#] &, Partitions[order-2], {1}], {2}];
     temp=temp /. {ff[a_, b_]^p_ -> Factorial[p]*a^p, ff[a_, b_] -> a};
    temp = Apply[MyOuter, temp, {1}];
    temp = Flatten[temp, 1]];
(*-----------------------------------------------------------------------*)
SSON[a_,b_,c_,e_,order_]:=Map[g,Map[Distribute[#] &, Map[Expand[#]
&, TO[a,b,c,e,order+1], order+1], order+1]] End[]; EndPackage[];
```

A brief description of the above code :

`MyOuter`: Performs outer products of lists elements.

`Combinations`: Produces the non ordered combinations without repetition of n objects taken from the elements of a list.

`Combinations2`: Returns the products of the elements taken from Combinations.

`f, g`: These two functions work with the numerical coefficients of the tree expressions performing needed expansions.

`RunLengthEncode`: Gives a list of pairs (x, y) which corresponds to element x of length y in a list.

`T`: This function applies the main ideas of our approach. Using Combinations2 and recursion, produces a list with all the possible matrix expressions which correspond to trees with cumulative order $p-1$. The function CoverList is a dummy function which is used to protect the recursively produced elements of the lists (in levelspec 3) from the outer product. These elements which are expressions that correspond to the branches of each tree are multiplied using MyOuter and the list is flattened to produce the full list needed. Then the CoverList protection is taken out and the expressions are multiplied by a to meet the fact that a new node is added to each tree.

`TO`: Taking the results of T gives a list with the expressions for y which corresponds to the grafting of the trees with cumulative order $p-1$ into a new fat vertex and connecting that vertex into a new meagre root.

`S`: Using Combinations2 and recursion produces a list with elements all the values of symmetry function $\sigma(t)$ which correspond to all possible trees with cumulative order $p-1$.

`SSON`: This function is needed internally to produce the correct form of the `TCO` and it is similar to `BCO`. order $p+1$.

The code can be downloaded from http://math.teiath.gr/ifamelis/ipapers.html.

Acknowledgements

This research is co-funded by the European Social Fund (75%) and National Resources (25%) under the framework of the Program - Pythagoras II of the National Technical University of Athens.

References

[1] P. Albrecht, *Numerical treatment of O.D.E.s: the theory of A–methods*, Numer. Math., 47 (1985) pp. 59–87.

[2] J. C. Butcher, *Implicit Runge–Kutta processes*, Math. Comput., 18 (1964) pp. 50–64.

[3] J. C. Butcher, *On Runge–Kutta processes of high order*, J. Austral. Math. Soc., 4 (1964) pp. 179–194.

[4] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, Wiley, Chichester, 1987.

[5] M. P. Calvo and J. M. Sanz–Serna, *Order conditions for canonical Runge–Kutta–Nyström methods*, BIT, 32 (1992), pp. 131–142.

[6] F. Cameron , *A Matlab Package for Automatically Generationg Runge–Kutta Trees, Order Conditions, and Truncatiom Error Coefficients*, ACM Trans. on Math. Soft., 32 (2006), pp. 274–298.

[7] M. P. Calvo and J. M. Sanz–Serna, *High order symplectic Runge–Kutta–Nyström methods*, SIAM J. Sci. Comput., 14 (1993), pp. 1237–1252.

[8] M. M. Chawla and P. S. Rao, *Numerov type method with minimal phase lag for the integration of second order periodic initial value problems II. Explicit method*, J. Comput. Appl. Math., 15 (1986) pp. 329–337.

[9] J. P. Coleman, *Order conditions for a class of two–step methods for* $y'' = f(x, y)$, IMA J. Numer. Anal., 23 (2003), pp. 197–220.

[10] I. Th. Famelis, S. N. Papakostas and Ch. Tsitouras, *Symbolic derivation of Runge–Kutta order conditions*, J. Symb. Comput., 37 (2004), pp. 311–327.

[11] E. Fehlberg, NASA TR R381, Marsal Space Flight Center, Ala 35812 (1972).

[12] E. Hairer, C. Lubich and G. Wanner, *Geometric Numerical Integration*, Springer–Verlag, Berlin, 2002.

[13] E. Hairer, S. Nørsett and G. Wanner, *Solving Ordinary differential equations I: Nonstiff problems, Second Revised Edition*, Springer–Verlag, Berlin, 1993.

[14] E. Hairer and G. Wanner, *A theory of Nyström methods*, Numer. Math., 25 (1976), pp. 383–400.

[15] A. J. Harrison, *Runge–Kutta Order Conditions Package*, Aveliable from *http://library.wolfram.com/infocenter/MathSource/1524/*

[16] M. E. Hosea, *A new recurrence for computing Runge–Kutta truncation error coefficients* SIAM J. Numer. Anal., 32 (1997), pp. 1989–2001.

[17] J. Keiper, *NumericalMath‘Butcher‘.m, Version 1.2*, Wolfram Research Inc., 1989.

[18] J. D. Lambert, *Numerical methods for ordinary differential systems*, Wiley, Chichester, 1991.

[19] C. L. Liu, *Introduction to Combinatorial Theory*, Mac Grow–Hill, 1968.

[20] D. I. Okunbor, Canonical integration methods for Hamiltonian dynamical systems, (1992) *Report UIUCDS-R-92-1885, Univ. Illinois, Urbana*

[21] G. Papageorgiou and Ch. Tsitouras, *Runge–Kutta pairs for scalar autonomous Initial Value Problems*, Int. J. Comput. Math., 80 (2003), pp. 201–209.

[22] A. Papaioannou, *Enumeration of Graphs (in Greek)*, NTUA, Athens, 2000.

[23] S. N. Papakostas, *Unpublished software*, 1992–1993.

[24] S. N. Papakostas, *Ph.D. Thesis (in Greek)*, Athens, 1996.

[25] S. N. Papakostas and Ch. Tsitouras, *High algebraic order, high phase–lag order Runge–Kutta and Nyström pairs*, SIAM J Sci. Comput., 21 (1999), pp. 747–763.

[26] S. N. Papakostas, Ch. Tsitouras and G. Papageorgiou, *A general family of explicit Runge–Kutta pairs of orders 6(5)*, SIAM J Numer. Anal., 33 (1996), pp. 917–936.

[27] J. Riordan, *An Introduction to Combinatorial Analysis*, Wiley, N. York, 1958.

[28] M. Sofroniou, *Symbolic Derivation of Runge–Kutta methods* J. Symbol. Comput., 18 (1994), pp. 265–296.

[29] Ch. Tsitouras, *Explicit Numerov–type methods with reduced number of stages*, Proc. 1st Inter. Symp. Nonlinear Problems, ed. N. Stavrakakis, NTUAthens, January 2000, pp 429–438.

[30] Ch. Tsitouras, *A parameter study of a Runge–Kutta pair of orders 6(5)*, Appl. Math. Lett., 11 (1998), pp 65–69.

[31] Ch. Tsitouras, *Optimal Runge–Kutta pairs of orders 9(8)*, Appl. Numer. Math., 38 (2001), pp. 123–134.

[32] Ch. Tsitouras, *Stage reduction on P-stable Numerov type methods of eighth order*, J. Comput. Appl. Math., 191 (2006), pp 297–305

[33] Ch. Tsitouras, *Explicit eighth order two-step methods with nine stages for integrating oscillatory problems*, Int. J. Mod. Phys. C, 191 (2006), pp 297–305

[34] Ch. Tsitouras and I. Th. Famelis, *Symbolic Derivation of Runge–Kutta–Nyström Order Conditions.*, J. Comput. Appl. Math., to appear.

[35] Ch. Tsitouras and S. N. Papakostas, *Cheap error estimation for Runge–Kutta pairs*, SIAM J Sci., Comput. 20 (1999), pp. 2067–2088.

[36] S. Wolfram, *The Mathematica Book, 5th ed.*, Wolfram Med., 2003.