CWI Quarterly Volume 11 (1) 1998, pp. 75 – 82

An Explicit Runge-Kutta Method of Order Twenty-five

B.P. Sommeijer
CWI, P.O. Box 94079,
1090 GB Amsterdam, The Netherlands

1. Prologue

This note was written on the occasion of the twenty-fifth anniversary of Prof. dr. P.J. van der Houwen's stay at our institute. It was written, merely to contribute to the festive joy of that celebration, without having any other pretensions.

On this jubilee we have concentrated on the number 25 and asked ourselves: is there anything special about 25? Mathematically speaking, this number is not very spectacular; it is not even a prime. On the other hand, it is a square itself and at the same time the sum of 2 squares. Furthermore, it is the only square that is two less than a cube (a result due to Fermat, cf. [8]), and moreover, 25 is the only integer of the form n^k which can be written in the form $(n-1)!+1$ (proved by Liouville, cf. [8]).

Leaving the field of number theory and taking into account the celebrator's major interest, one is automatically led into the direction of Runge-Kutta (RK) methods. What have Runge-Kutta methods in common with the number 25? Maybe the fact that Kutta died on the 25th of December, or possibly that Nyström came up in 1925 with an extension to RK methods? Do there exist RK methods with 25 stages? Or, even more extravagant, has ever before someone wasted his time by constructing an (explicit) RK method of order 25? Since I am not aware of the existence of such a method, I will construct one. However, it should be noted that the recreational character of such a product cannot be overemphasized.

2. Introduction

There is, and now I am serious again, a market for high-order methods to integrate the initial-value problem for the (nonstiff) ordinary differential equation (ODE)

$$\frac{dyt}{dt} = f(y(t)), \quad t \geq t_0. \tag{1}$$

order of the RK method	conditions to be satisfied
1	1
2	2
4	8
7	85
10	1205
15	141,083
20	20,247,374
25	3,231,706,871

TABLE 1.

For example, problems in the field of astronomy usually have to be solved with high accuracy demands. This requirement is, in general, more efficiently fulfilled by using a method of high order. Concentrating on the class of explicit RK methods (cf. [1, p.152]), the difficulties in constructing such a method grow exponentially with the order. This is because the number of nonlinear algebraic equations to satisfy the order conditions is a rapidly increasing function of the order and, to solve this system is far from trivial. In Table 1, we give an idea of the dimension of this system. The next question is: how many parameters do we need in order to satisfy this huge amount of conditions; or, in other words, how many stages are at least required? This problem has now been solved up to and including order 8 (see BUTCHER [1, p.194]). For higher orders it is still an open question. For example, starting with an s-stage explicit RK method providing $s(s+1)/2$ free parameters, HAIRER [4] succeeded in satisfying the 1205 conditions for order 10 using only 17 stages, i.e., with 153 parameters. Within the class of explicit RK methods, this is the highest order obtained. This 10th-order method earned Hairer an entry in the Guinness Book of Records.

Here we will present an explicit RK method of order 25 using 313 stages. This is certainly not the least number of stages to obtain this order. Although the minimal number is unknown, upperbounds provided by COOPER & VERNER [2] and by GRAGG [3] state that such a scheme can be constructed with at most 170 stages.

Starting with 313 stages, we obtain 49141 free parameters which appears to be sufficient to satisfy the (approximately) $3 \cdot 10^9$ order conditions. Moreover, 45228 of these parameters are set to zero. Among the remaining 3913 parameters many equal numbers occur. Summarizing, the scheme is completely determined by specifying only 182 parameters.

3. Construction

The construction of the method of order 25 does not follow the classical approach (since the number of order conditions is unmanageable, of course) but is obtained by iterating an *implicit* RK method. The construction is completely described in [5] and will be reproduced here in condensed form (the description is given for a scalar differential equation; however, the extension to a system of ODEs is straightforward).

Let y_n denote an approximation to the exact solution $y(t)$ of the ODE at $t = t_n$. Then, one step of the s-stage, implicit Runge-Kutta method

$$\begin{aligned} k_i &= f(y_n + h\sum_{j=1}^{s} a_{ij}k_j), \quad i = 1, 2, \ldots, s, \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j k_j \end{aligned} \tag{2}$$

advances the solution from t_n to $t_n + h$, h being the stepsize. The parameters a_{ij} and b_j completely determine the RK method. Therefore, such methods are usually represented by their so-called Butcher tableau [1, p.163]

$$\begin{array}{c|c} & A = (a_{ij}) \\ \hline & \mathbf{b}^T = (b_1, \ldots, b_s) \end{array} \quad . \tag{3}$$

It is well known that the order of accuracy of the method (2) is maximal (relative to the number of stages s) when the RK method is of the so-called *Gauss-Legendre* type; in that case, the order is equal to $2s$. Therefore, our starting point for the construction of an RK method of order 25 will be a Gauss-Legendre method (with 13 stages). In the following, A and $\mathbf{b}$, both of dimension 13, will correspond to this method.

Introducing the vector $\mathbf{k} := (k_1, \ldots, k_{13})^T$, the RK method (2) can be written compactly as

$$\begin{aligned} \mathbf{k} &= f(y_n\mathbf{e} + hA\mathbf{k}), \\ y_{n+1} &= y_n + h\mathbf{b}^T\mathbf{k}, \end{aligned} \tag{2'}$$

where $\mathbf{e}$ denotes the unit vector of dimension 13. Furthermore, we make the convention that for any vector $\mathbf{v} = (v_j)$, $f(\mathbf{v})$ denotes the vector with entries $f(v_j)$. From (2'), we observe that $\mathbf{k}$ is implicitly defined. To solve for this vector, the following iteration process is proposed

$$\begin{aligned} \mathbf{k}^{(0)} &= f(y_n)\mathbf{e}, \\ \mathbf{k}^{(j)} &= f(y_n\mathbf{e} + hA\mathbf{k}^{(j-1)}), \quad j = 1, 2, \ldots, 24\ . \end{aligned} \tag{4a}$$

Together with

$$y_{n+1} = y_n + h\mathbf{b}^T\mathbf{k}^{(24)}, \tag{4b}$$

this defines a new method for the integration of the ODE (1) over one step. Notice that the method {(4a),(4b)} is an RK method of *explicit* type, since $\mathbf{k}^{(j)}$ is explicitly expressed in terms of $\mathbf{k}^{(j-1)}$. Since in each 'iteration' we gain one order (see also [7]), it is easy to show that this scheme is of order 25 indeed.

The Butcher tableau of the method {(4a),(4b)} has the form

$$\begin{array}{c|ccccccc}
O & & & & & & \\
A & O & & & & & \\
O & A & O & & & & \\
 & O & A & O & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
O & \cdots & & & \mathrm{O} & A & O \\
\hline
\mathbf{0}^T & \cdots & & & & \mathbf{0}^T & \mathbf{b}^T
\end{array} \tag{4'}$$

where O and $\mathbf{0}^T$ respectively denote a matrix and a vector (both of dimension 13) with zero entries. In this method we have $25 \cdot 13 = 325$ stages. However, the first 13 stages (represented by the O-matrix in the first row) are identical and thus require only one f-evaluation. Hence, this scheme counts for $24 \cdot 13 + 1 = 313$ f-evaluations. The generating Butcher tableau (3) of the underlying Gauss-Legendre method completely determines the scheme {(4a),(4b)} and is given in the Appendix.

Finally, we make two observations:

(i) in each 'iteration' in (4a) we need the evaluation of $f(\mathbf{v})$ with $\mathbf{v} = y_n\mathbf{e} + hA\mathbf{k}^{(j-1)}$, which requires 13 calls to the derivative function f. However, it should be observed that the components of $\mathbf{v}$ are known prior to all these calls. Consequently, these derivatives can be calculated *concurrently*. Thus, on a computer architecture possessing (at least) 13 parallel processors, one iteration in (4a) requires *effectively* the time needed for the calculation of one single f-evaluation. In this way the method {(4a),(4b)} counts for 25 *effective* f-evaluations to yield order 25.

(ii) another observation is that, apart from the final result y_{n+1} calculated in (4b), we can easily construct an *'embedded' approximation* by calculating $y^{(m)} = y_n + h\mathbf{b}^T\mathbf{k}^{(m)}$ for some $m < 24$. This does not require additional f-evaluations since $\mathbf{k}^{(m)}$ has already been computed in order to continue the iteration process. This embedded reference solution can be used to equip the method with an error control strategy (cf. [5] for details).

RK$_4$			**Hairer$_{10}$**			**iterated Gauss$_{25}$**			
h	CD	Σf	h	CD	Σf	h	CD	Σf_{seq}	Σf_{paral}
1/200	9.6	48000	1/4	10.1	4080	3	9.1	6260	500
1/400	10.8	96000	1/8	12.8	8160	5/2	10.7	7512	600
1/800	12.0	192000	1/16	15.9	16320	2	12.8	9390	750
1/3200	14.4	768000	1/20	16.8	20400	1	19.9	18780	1500
1/12800	16.8	3072000							

TABLE 2.

4. SOME TEST RESULTS

As a first example, we consider Euler's equation of motion [6]

$$\begin{aligned} y_1' &= y_2 y_3, & y_1(0) &= 0, \\ y_2' &= -y_1 y_3, & y_2(0) &= 1, \quad 0 \le t \le 60. \\ y_3' &= -0.51 y_1 y_2, & y_3(0) &= 1. \end{aligned}$$

We will apply the 'classical' 4th-order RK method [1, p.181] to this problem as well as the 10th-order method of HAIRER [4] and the iterated Gauss-Legendre method {(4a),(4b)} of order 25.

In Table 2, CD denotes the number of correct digits in the solution at the end point of the integration interval, i.e., $CD := -\log_{10}(||\text{ global error }||_\infty)$ and Σf denotes the total number of f-evaluations required by the various methods.

The second example is provided by the orbit equation [6]

$$\begin{aligned} y_1' &= y_3, & y_1(0) &= 1-\varepsilon, \\ y_2' &= y_4, & y_2(0) &= 0, \\ & & & \qquad 0 \le t \le 20, \quad \varepsilon = 0.3. \\ y_3' &= -\frac{y_1}{(y_1^2+y_2^2)^{3/2}}, & y_3(0) &= 0, \\ y_4' &= -\frac{y_2}{(y_1^2+y_2^2)^{3/2}}, & y_4(0) &= [(1+\varepsilon)/(1-\varepsilon)]^{1/2}. \end{aligned}$$

The results obtained by the three methods are given in Table 3.

RK_4			$Hairer_{10}$			iterated $Gauss_{25}$			
h	CD	Σf	h	CD	Σf	h	CD	Σf_{seq}	Σf_{paral}
1/32	5.2	2560	1/2	2.0	680	4	2.8	1565	125
1/128	7.8	10240	1/4	5.6	1360	2	6.9	3130	250
1/512	10.2	40960	1/8	9.4	2720	1	13.4	6260	500
1/2048	12.6	163840	1/16	12.8	5440	1/2	19.3	12520	1000
1/8192	15.0	655360	1/32	15.6	10880				

TABLE 3.

5. CONCLUSIONS

From these tables it is clear that the high-order schemes are much more efficient than the 4th-order method in case the problem has to be solved with high accuracy demands. Comparing the 10th-order method of Hairer with the iterated Gauss-Legendre method of order 25, run on a sequential computer, we see that Hairer's method is more efficient, unless a global error is required smaller than approximately 10^{-14}. However, matters are different when parallel computers are used. In that case, the number of f-evaluations required by the iterated Gauss-Legendre method (i.e., Σf_{paral}) is only a fraction of the number required by Hairer's method, especially in the high-accuracy range.

6. EPILOGUE

We have shown that it is very simple to construct an explicit Runge-Kutta method of order 25. In fact, following this approach, it is straightforward to obtain methods of arbitrarily high order (hence, no matter how many years a jubilee is celebrating, a suitable present is now available).

It is of course questionable whether as high an order as 25 is realistic within the field of the numerical solution of ODEs. However, since this idea of constructing parallel methods can also be exploited to solve the special higher-order differential equation $d^k y(t)/dt^k = f(y(t)), (k \geq 2)$, it may prove to be useful to obtain efficient parallel methods of a realistic order (10, say) for this type of problems.

REFERENCES

1. J.C. BUTCHER (1987). *The numerical analysis of ordinary differential equations, Runge-Kutta and general linear methods*, Wiley, Chichester-New York-Brisbane-Toronto-Singapore.

2. G.J. Cooper and J.H. Verner (1972). *Some explicit methods of high order*, SIAM J. Numer. Anal. 15, pp. 643-661.
3. W.B. Gragg (1965). *On extrapolation algorithms for ordinary initial value problems*, SIAM J. Numer. Anal. 2, pp. 384-403.
4. E. Hairer (1978). *A Runge-Kutta method of order 10*, J. Inst. Maths. Applics. 21, pp. 47–59.
5. P.J. van der Houwen and B.P. Sommeijer (1990). *Parallel iteration of high-order Runge-Kutta methods with stepsize control*, J. Comput. Appl. Math. 29, pp. 111–127.
6. T.E. Hull, W.H. Enright, B.M. Fellen and A.E. Sedgwick (1972). Comparing numerical methods for ordinary differential equations, SIAM J. Numer. Anal. 9, pp. 603-637.
7. K.R. Jackson and S.P. Nørsett (1988). *Parallel Runge-Kutta methods*, to appear.
8. D. Wells (1986). *The Penguin dictionary of curious and interesting numbers*, Penguin Books.

APPENDIX. The Butcher tableau (3) of the 13-point Gauss-Legendre RK method

.010121001191328969880	-.003773071585325731176	.002785615756062805405	-.002218270384046127274
.001803064691152919247	-.001464286183975972063	.001172194345233766127	-.000913582925476946914
.000682707775014527071	-.000477846825328057045	.000299962625259445461	-.000152576299212045336
.000043560460018371881			
.021900128317529511652	.023030374959432111978	-.006043319825448234459	.004023676755673410248
-.003042494726226345487	.002382034987767637450	-.001867246400811313992	.001436474648283859267
-.001064422446011306992	.000740852231417186149	-.000463345539787940887	.000235139088196600369
-.000067051661504157900			
.019429947719646208013	.050069575543263592342	.034718377554946809615	-.007863032067308685706
.004852746158553562344	-.003484003336607684475	.002609062429679834287	-.001953720768410971637
.001423295087533456274	-.000979733532987650877	.000608355059029741647	-.000307359452503456644
.000087444238510288418			
.020746108444520132986	.043980056680602746238	.075566373412184083483	.044536495190486434570
-.009220916842601834711	.005354650885687748358	-.003655226799851316632	.002604429002348943755
-.001842009894761704927	.001244608122107674162	-.000763750235756611259	.000383103892798337037
-.000108595157793474338			
.019890752898251060546	.047409438726828386754	.066193897297749457289	.096977429397685786696
.051954011884222125578	-.010063307033250376007	.005534424914749518585	-.003585220082857202774
.002408946275359569017	-.001579024638206943695	.000951119931434866800	-.000471841291130799782
.000132996200941124552			
.020503975759472767006	.045091006451920914165	.071574942192159462540	.084857433870937933433
.113150057093106559270	.056570795065724309603	-.010351456092249522975	.005394003899694171611
-.003292627699078813801	.002050389067403360880	-.001199028848645799803	.000584799095209111448
-.000163447833221850413			
.020037939581102676277	.046800429085137372298	.067878693454495903597	.091873074724079340319
.098962271761227679498	.123214025135056194245	.058137888307718477548	-.010072435003607575039
.004945752007216571657	-.002800084343106471179	.001558061655397715634	-.000739679166273148341
.000204062801555263482			
.020405450215879790173	.045475950823655112508	.070635783958539419035	.087022601313569508259
.107200651467523064957	.107747586231754447594	.126627232707686478072	.056570795065724309603
-.009242033324662308114	.004215556510034935706	-.002138187082265843308	.000969743466943309791
-.000261973376814827246			
.020109006181716815207	.046532591209995023739	.068485635178458752431	.090652015019179812835
.101499077493084682138	.116726810214305821980	.110741351700687436512	.123204897164698995213
.051954011884222125578	-.007904439016712917556	.003242857812144161942	-.001348688807964162797
.000351249484406879213			
.020350593960592683143	.045677646026065886919	.070200505345650230491	.087828382258865194977
.105750033663205956083	.110537161129099675450	.119931003415288271729	.107786939245760870847
.113128940611046085867	.044536495190486434570	-.006129618302290464252	.002080693238261477718
-.000504106061862193226			
.020154558144147651341	.046368109371367680601	.068828400050863877584	.090052723913960520017
.102484728680910794882	.115095310899859590843	.113666714185757120810	.116625593468056303682
.099055277609890688811	.096936022448281554846	.034718377554946809615	-.004008825624399368385
.000812054663011731746			
.020309054044162097660	.045825610830667623587	.069900100649681560119	.088332138149555682990
.104972446214455558149	.111705115483164759938	.118143023016248269090	.110759555143680981755
.106950518494670596643	.085049313625299458891	.075480074935341853691	.023030374959432111978
-.001658125934871571892			
.020198441922639567878	.046213326218076269293	.069136792484634173770	.089550837206300926185
.103225315993429724084	.114055173056925566120	.115103582270203188969	.114605876315424591270
.102104959077291331908	.091291260765018996414	.066651139353830813826	.049833821504189955133
.010121001191328969880			

.020242002382657939760	.046060749918864223957	.069436755109893619231	.089072990380972869140
.103908023768444251156	.113141590131448619206	.116275776615436955097	.113141590131448619206
.103908023768444251156	.089072990380972869140	.069436755109893619231	.046060749918864223957
.020242002382657939760			