Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Mathematical and Computer Modelling 40 (2004) 1157–1169
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

Construction of Explicit Runge-Kutta Pairs with Stiffness Detection

M. SOFRONIOU
Wolfram Research Inc., 100 Trade Center Drive
Champaign, IL 61820, U.S.A.
marks@wolfram.com

G. SPALETTA
Mathematics Department, Bologna University
Piazza Porta S. Donato 5, 40127 Bologna, Italy
giulia@cs.unibo.it

Abstract—Explicit Runge-Kutta schemes are the methods of choice for solving nonstiff systems of ordinary differential equations at low to medium tolerances. The construction of optimal formulae has been the subject of much research. In this article, it will be shown how to construct some low order formula pairs using tools from computer algebra. Our focus will be on methods that are equipped with local error detection (for adaptivity in the step size) and with the ability to detect stiffness. It will be demonstrated how criteria governing 'optimal' tuning of free parameters and matching of the embedded method can be accomplished by forming a constrained optimization problem. In contrast to standard numerical optimization processes our approach finds an exact (infinite precision) global minimum. Quantitative measures will be given comparing our new methods with some established formula pairs.

Keywords—Ordinary differential equations, Initial value problems, Runge-Kutta methods, Stiffness detection, Symbolic computation, Computer algebra systems, Computer generation of numerical methods.

1. INTRODUCTION

A framework for explicit Runge-Kutta methods is being implemented as part of an ongoing overhaul of MATHEMATICA's differential equation solver `NDSolve`. One of our goals was to establish a unified environment for a whole class of methods of different orders. This helps to provide a uniform basis for comparing methods, reduces the potential for programming errors, and allows the optimization of a single implementation from which all methods benefit. Furthermore, specification of additional Runge-Kutta methods can be accomplished by simply entering the appropriate coefficients.

It is well known that explicit Runge-Kutta methods are not suitable for the numerical solution of stiff differential equations. One of the features that we wanted to incorporate in our implementation was automatic stiffness detection. In this way users are provided with run-time information about when the choice of method is inappropriate. It also becomes possible to switch between stiff and nonstiff Runge-Kutta methods [1].

0895-7177/05/$ - see front matter Typeset by AMS-TEX
doi:10.1016/j.mcm.2005.01.010

Many high order schemes in the literature are capable of using the stiffness detection device that we have chosen because the necessary condition arises naturally in the simplifying assumptions that are adopted in the derivation. However, there was a lack of methods at low order with the required property. One goal of this article is to fill this void.

Many methods, especially at high order, are constructed by a numerical search to constrain free parameters. This often requires reasonable starting values and the numerical optimization process is not guaranteed to detect local minima. Another aim of this article is to use ideas that are common in algebraic geometry to derive appropriate methods. These tools are rigorous in the sense that they enable precise statements about the methods that are constructed.

This article is organized as follows. Section 2 introduces some definitions and notation. Section 3 describes the properties of explicit Runge-Kutta methods that will be considered. Details of the derivation process in MATHEMATICA are given in Section 4 and new methods are described in Section 5. The Appendix contains some formula pairs that have been used for comparison purposes.

All computations in this article were carried out using an AMD Athlon machine at 800 MHz with 1152 MB of RAM running RedHat Linux 6.2. Version 5.0 of MATHEMATICA has been used except where otherwise stated.

2. DEFINITIONS

The most common application of numerical methods for systems of ordinary differential equations of dimension $n \in \mathbb{N}$ is to *initial value problems*

$$y'(t) = f(t, y(t)), \qquad f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n, \tag{1}$$

$$y(t_0) = y_0. \tag{2}$$

Equation (1) defines a family of solutions. A specific solution is fixed by specifying the initial state, or initial conditions (2). The initial value problem (1),(2) varies continuously with time. In order to solve the problem, an approximate solution is sought at fixed output points in a finite time range.

Denote the s-stage explicit Runge-Kutta method for the approximate solution to the initial value problem (1),(2) at $t_{n+1} = t_n + h$ by

$$\begin{aligned} g_1 &= y_n, \\ k_1 &= f(t_n, g_1), \\ g_i &= y_n + h \sum_{j=1}^{i-1} a_{i,j} k_j, \qquad i = 2, \ldots, s, \\ k_i &= f(t_n + c_i h, g_i), \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{3}$$

The coefficients of the method are free parameters that are usually chosen to satisfy a Taylor series expansion through some order in the step size h. It has become customary to denote the method coefficients $c = [c_i]^\top$, $b = [b_i]^\top$, and $A = [a_{i,j}]$ using a *Butcher table*. For an explicit Runge-Kutta method the matrix A is strictly lower triangular and the Butcher table has the following form.

$$\begin{array}{c|ccccc} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

We will assume the general convention that the *row-sum conditions* hold

$$c_i = \sum_{j=1}^{i-1} a_{i,j}, \qquad i = 2, \ldots, s. \tag{4}$$

These have the effect that all points at which the function f is evaluated are first order approximations to the solution. Conditions (4) are also a useful device appearing naturally in the derivation of high order Runge-Kutta methods.

Let the local solution $u(t)$ at (t_n, y_n) be defined by

$$u'(t) = f(t, y(t)), \qquad u(t_n) = y_n.$$

For sufficiently smooth f, a Taylor expansion about (t_n, y_n) yields expressions for the local error

$$u(t_n + h) - y_{n+1} = h^{p+1} \sum_{j=1}^{r_{p+1}} T_j^{(p+1)} D_j^{(p+1)} + h^{p+2} \sum_{j=1}^{r_{p+2}} T_j^{(p+2)} D_j^{(p+2)} + O\left(h^{p+3}\right).$$

The $D_j^{(i)}$ are *elementary differentials* of order i (see [2–5]) and the $T_j^{(i)}$ are the *truncation error* coefficients. The r_i are natural numbers denoting the number of distinct elementary differentials of order i. The principal local truncation error coefficients are given by the terms of order h^{p+1}. The $T_j^{(i)}$ depend on the coefficients of the explicit Runge-Kutta method and are independent of the differential equation. An explicit Runge-Kutta method is said to be of *order* p if the Taylor series for the exact solution and the approximate solution coincide through terms in h^p

$$\|u(t_n + h) - y_{n+1}\| \leq Ch^{p+1}.$$

3. METHOD PROPERTIES

Choices of reasonable criteria for constructing explicit Runge-Kutta methods are somewhat subjective. Shampine states that [5]

> Certain of the constraints described can be translated into mathematical constraints, but most are sufficiently vague that one must explore the space of parameters in a heuristic way.

The criteria that have been chosen in the sequel will be explained and precisely formulated as mathematical constraints. Unless otherwise stated, the Euclidean norm $\|v\|$ of a vector v will be implied throughout.

Error estimation can be accomplished by considering (3) and using a linear combination of the same function values with a second set of weights $\hat{b}_i$ in place of b_i. This gives rise to embedded explicit Runge-Kutta pairs of methods, where the higher order method usually has order $p \geq 2$ and the lower order method has order $\hat{p} = p - 1$. Such methods are commonly denoted as a pair of order $p(\hat{p})$.

An estimate of the error of the formula of order $\hat{p}$ can be found by considering the difference from the formula of order p as

$$h \sum_{i=1}^{s} \left(b_i - \hat{b}_i\right) k_i.$$

A norm of this vector furnishes a scalar quantity that can be used to estimate the local error and adjust the step size in an adaptive fashion. In an implementation, the norm commonly incorporates user prescribed relative and absolute tolerances.

Denote the principal local truncation error of the higher order formula of order p as $T^{(p+1)} = [T_j^{(p+1)}]^\top$ and of the embedded formula of order $\hat{p} = p - 1$ as $\hat{T}^{(p)}$. The secondary truncation error of the embedded method will be denoted as $\hat{T}^{(p+1)}$.

Assuming that $\hat{p} = p - 1$, then quantitative measures governing a suitable choice of coefficients for the embedded method of the pair are [5]

$$B = \frac{\left\|\hat{T}^{(p+1)}\right\|}{\left\|\hat{T}^{(p)}\right\|}, \qquad C = \frac{\left\|\hat{T}^{(p+1)} - T^{(p+1)}\right\|}{\left\|\hat{T}^{(p)}\right\|}.$$

A small value of B ensures that the leading term in the expansion of the local error is likely to dominate for comparatively large step sizes. A small value of C ensures an accurate estimate of the local truncation error for large step sizes.

The following condition can be used to compare the relative accuracy of two methods μ and η of order p [6]:

$$\frac{W(\mu)}{W(\eta)} \left(\frac{\left\|T^{(p+1)}(\mu)\right\|}{\left\|T^{(p+1)}(\eta)\right\|} \right)^{1/(p+1)}. \tag{5}$$

The work W can be estimated by the number of function evaluations required by the method.

The following criteria have been chosen for the construction of explicit Runge-Kutta methods (see [7] for an example of an alternative).

- The higher order formula has order p with the minimal number of stages.
- The higher order formula is as accurate as possible, or at least close to the method which minimizes the norm of the principal truncation error, $\|T^{(p+1)}\|$. This is justified by the fact that the solution will be propagated by the higher order method (local extrapolation).
- No principal error terms in the embedded formula vanish. This ensures that the embedded scheme is never of order $r > \hat{p}$, which would result in defective error estimation for some problems.
- By construction, embedded methods have ratios B and C which are small.
- The first same as last (FSAL) device is used. This has the advantage that in (3), when a step is accepted, the last stage k_s can be reused at the next step as k_1 which saves a function evaluation. The chosen form of FSAL does not contribute to the accuracy of the higher order formula, which actually reduces to an $(s-1)$ stage method, but it allows for an embedded method to be constructed. For example, it is known that no embedded formula of order three exists for a four stage fourth-order explicit Runge-Kutta method (see for example [3, Section II.4, exercise 2]). On the other hand, an FSAL strategy enables the construction of an embedded method into a scheme with effectively four stages (see Section 5.3).
- Following [1,8–10] we construct methods with a facility for detecting stiffness by directly estimating the dominant eigenvalue λ of the Jacobian J of a problem. An alternative strategy for stiffness detection is described in [11]. Let v approximate the eigenvector corresponding to λ and consider an estimate of the form

$$|\lambda| \approx \frac{\|f(t, y+v) - f(t,y)\|}{\|v\|}.$$

By consistency (first order) and (4), a FSAL method has $c_s = 1$. Imposing the additional condition $c_{s-1} = 1$, a suitable value which approximates λ is then obtained from (3) with only some additional storage as

$$\rho = \frac{\|k_s - k_{s-1}\|}{\|g_s - g_{s-1}\|}.$$

The denominator here corresponds to several applications of the power method applied to hJ, which yields a good approximation to the eigenvector corresponding to the leading

eigenvalue λ (see [3]). As an example, let $|\partial S|$ denote the boundary of the linear stability function. Whenever $h\rho \geq |\partial S|$, then stability rather than local accuracy is restricting the choice of step size and the problem is considered to be stiff.

- The choice $c_i \neq c_j$, $c_i, c_j \in \{c_2, \ldots, c_{s-1}\}$ ensures that the method samples at distinct intermediary values. This makes it more likely that unexpected changes in the solution, such as discontinuous, will be revealed.

The following Butcher table is used to summarize the free parameters of the methods of interest here.

$$\begin{array}{c|ccccc} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 1 & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ 1 & b_1 & b_2 & \cdots & b_{s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & 0 \\ \hline & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s \end{array} \tag{6}$$

4. DERIVATION PROCESS

The solutions to the order conditions often contain free parameters. In order to obtain specific methods we need a way of imposing additional constraints. One way of eliminating the freedom of choice is to minimize the principal local truncation error [2,3,12–14].

The traditional approach for finding optimal methods uses numerical minimization routines to search for local minima on a predefined grid (e.g., [12]). The problem here is that the objective function can be highly nonlinear and there is no guarantee that a locally found minimum on a finite grid will actually be the best method. The minimization may not converge, given poor starting values for example, and if no solutions are obtained numerically that does not mean that none exist.

In this article, we will be concerned with how to derive methods that minimize the truncation error using a technique based upon *cylindrical algebraic decomposition* which effectively breaks up parameter spaces into cylindrical regions [15,16]. There are numerous advantages to this approach. In particular, it furnishes a *global minimum* for a given search region and furthermore, the minimum and parameter values are given in *exact* form.

The derivation process here is carried out in two phases. In both phases an optimization problem is formed involving the square of the Euclidean norm, since avoiding square roots is computationally more efficient. The first phase consists of solving the order conditions to construct methods of order p; an optimization problem is formed to derive the method that minimizes the objective function $\|T^{(p+1)}\|^2$. The second phase constructs another optimization problem to derive the embedded (lower order) scheme; the objective function to be minimized is $\|B\|^2 + \|C\|^2$.

The MATHEMATICA package `Butcher.m` addresses the issue of forming order conditions and related constraints for deriving Runge-Kutta methods using Butcher's rooted tree formalism [17]. Computation of the local truncation error can be accomplished using the function `ButcherPrincipalError` (see Section 4.1). The package derives equations for the free parameters of a method, but does not attempt to find solutions.

Nonlinear algebraic equations can be solved by forming *Gröbner bases* which are simpler and contain all the solutions of the original system (see, for example, [18]). It should be noted that the reduction process in forming a Gröbner basis is sensitive to the order of elimination of the variables. Furthermore, the time required to obtain solutions grows exponentially with

the number of variables and with the number of solutions of the system according to Bézout's theorem [19]. Separate cases arise for each nonlinear branch encountered. Similar comments apply to the complexity of cylindrical algebraic decomposition. For moderate sized problems, however, Gröbner bases and cylindrical algebraic decomposition are a very powerful technique.

4.1. Third-Order Optimal Method

This section illustrates a MATHEMATICA session for deriving the optimal third-order three stage explicit Runge-Kutta method with the property that one of the free parameters is used to equip the method with a stiffness detection facility.

Load a package for deriving order conditions and related quantities.

```
In[1]:= Needs["NumericalMath`Butcher` "];
```

Set the form of methods of interest.

```
In[2]:= $RungeKuttaMethod = Explicit;
```

Define a function for computing the square of the Euclidean norm of a vector.

```
In[3]:= EuclideanNormSquared[x_?VectorQ]:= Dot[x,x];
```

Choose the order p and the number of stages s of the method. Since the FSAL device is used, the higher order method can be derived with $s-1$ stages

```
In[4]:= p = 3;
In[5]:= s = 4;
```

Specify a transformation rule that embodies the criterion for stiffness detection.

```
In[6]:= stiffnesscondition = { c_{s-1} → 1};
```

Compute the order conditions for the higher order method and impose the stiffness criterion.

```
In[7]:= rkoc = RungeKuttaOrderConditions[p, s-1];
In[8]:= rkoc = Flatten[rkoc] /.  stiffnesscondition
```

$$\text{Out[8]} = \left\{b_1 + b_2 + b_3 == 1,\ b_3 + b_2\, c_2 == \frac{1}{2},\ a_{3,2} b_3 c_2 == \frac{1}{6},\ b_3 + b_2 c_2^2 == \frac{1}{3}\right\}$$

Define the variables at run time by converting the order conditions to polynomials.

```
In[9]:= vars = Variables[rkoc /.  Equal → Subtract];
```

Compute the vector of terms in the principal local error and impose the stiffness criterion.

```
In[10]:= Tp = ButcherPrincipalError[p, s-1] /.  stiffnesscondition;
```

Form the square of the Euclidean norm of the principal error terms.

```
In[11]:= objfun = EuclideanNormSquared[Tp]
```

$$\text{Out[11]} = \frac{1}{576} + \left(-\frac{1}{8} + a_{3,2} b_3 c_2\right)^2 + \frac{1}{4}\left(-\frac{1}{12} + a_{3,2} b_3 c_2^2\right)^2 + \frac{1}{36}\left(-\frac{1}{4} + b_3 + b_2 c_2^3\right)^2$$

Use Gröbner bases to reduce the previous result modulo the constraints defining the order conditions. At higher orders it may be preferable to solve the order conditions and simplify the objective function using each solution branch in place of this step.

```
In[12]:= objfun = PolynomialReduce[objfun, rkoc /.  Equal → Subtract,vars];
In[13]:= objfun = Last[objfun]
```

$$\text{Out[13]} = \frac{1}{144}\left(1 - 2b_3 + 4b_3^2 - c_2 + 6b_3 c_2^2 - 8b_3^2 c_2^2 + c_2^4 - 4b_3 c_2^4 + 4b_3^2 c_2^4\right)$$

The function `Minimize` takes an objective function, a list of equalities and/or inequalities, and a list of variables. The function returns a list of the minimum value and a list of parameters and values that yield the minimum. `Minimize` uses cylindrical algebraic decomposition to find the minimal solution for the principal error.

```
In[14]:= Timing[ Experimental`Minimize[objfun, rkoc, vars] ]
```

$$\text{Out[14]} = \left\{0.55 \text{ Second},\ \left\{\frac{1}{288}, \left\{c_2 \to \frac{1}{2},\ b_2 \to \frac{2}{3},\ a_{3,2} \to 2,\ b_3 \to \frac{1}{6},\ b_1 \to \frac{1}{6} = \right\}\right\}\right\}$$

The result is a list of the time taken, the square of the Euclidean norm representing the minimal solution, and the minimal solution.

4.2. Derivation of Embedded Methods

Following on from Section 4.1, the order conditions for a second-order embedded method are now constructed and values for the free parameters are then derived using constraints on the truncation error.

Extract the minimal solution from the last result in 4.1.

In[15]:= `minsol = Part[%,2,2]`

Construct a list of transformation rules for the lower order, embedded scheme.

In[16]:= `lowsub =` $\left\{\texttt{b}_{\texttt{i\_}} \to \hat{\texttt{b}}_{\texttt{i}},\ \texttt{c}_{\texttt{s-1}} \to 1,\ \texttt{c}_{\texttt{s}} \to 1,\ \texttt{a}_{\texttt{s,j\_}} \to \texttt{b}_{\texttt{j}}\right\}$;

Derive the order conditions for the embedded scheme.

In[17]:= `rkoclow = RungeKuttaOrderCondition[p-1,s];`

In[18]:= `rkoclow = Flatten[rkoclow] /. lowsub /. minsol;`

Determine the free parameters of the embedded scheme.

In[19]:= `lowvars = Variables[rkoclow /. Equal → Subtract];`

Solve the (linear system of) order conditions for the embedded method. A message is issued to indicate that some of the parameters are unconstrained by the order conditions.

In[20]:= `lowsol = First[ Solve[rkoclow, lowvars] ]`

Out[20] = Solve::svars: Equations may not give solutions for all solve variables

$$\left\{\hat{b}_1 \to \hat{b}_3 + \hat{b}_4,\ \hat{b}_2 \to 1 - 2\hat{b}_3 - 2\hat{b}_4\right\}$$

Construct the primary and secondary truncation error terms for the embedded method.

In[21]:= `T̂p = ButcherPrincipalError[p-1,s] /. lowsub /. lowsol /. minsol;`

In[22]:= `T̂s = ButcherPrincipalError[p,s] /. lowsub /. lowsol /. minsol;`

Simplify the primary truncation error terms for the high order method derived in Section 4.1.

In[23]:= `Tp = Tp /. minsol;`

Derive an objective function for the embedded method and simplify the expression as a rational function.

In[24]:= `BSquared = EuclideanNormSquared[T̂s]/EuclideanNormSquared[T̂p];`

In[25]:= `CSquared = EuclideanNormSquared[T̂s - Tp]/`
`EuclideanNormSquared[T̂p];`

In[26]:= `objfun = Together[BSquared + CSquared]`

Out[26] =
$$\frac{57 - 732\hat{b}_3 + 2484\hat{b}_3^2 - 396\hat{b}_4 + 2568\hat{b}_3\hat{b}_4 + 740\hat{b}_4^2}{2\left(17 - 204\hat{b}_3 + 612\hat{b}_3^2 - 108\hat{b}_4 + 648\hat{b}_3\hat{b}_4 + 180\hat{b}_4^2\right)}.$$

Find the minimal solution for the embedded scheme and also give the time taken to find the solution.

In[27]:= `Timing[ Minimize[objfun, rkoclow, lowvars] ]`

Out[27] =
$$\left\{0.35\ \text{Second},\ \left\{\frac{1}{18}\left(43 - 2\sqrt{82}\right), \left\{\hat{b}_4 \to \frac{1}{48}\left(16 - \sqrt{82}\right), \hat{b}_3 \to \frac{1}{144}\left(-4 + \sqrt{82}\right),\right.\right.\right.$$
$$\left.\left.\left.\hat{b}_2 \to \frac{1}{36}\left(14 + \sqrt{82}\right), \hat{b}_1 \to \frac{1}{72}\left(22 - \sqrt{82}\right)\right\}\right\}\right\}$$

4.3. Fourth-Order Optimal Method

Unfortunately, the high complexity of the solution process outlined in Section 4.1 currently limits its application at higher order. However, when it is possible to solve the order equations explicitly progress can still be made, although the steps involved are somewhat more intricate. As illustrated, in this section an analytic expression for the optimal fourth-order scheme with four stages is used to simplify the objective function. It is well known that there are four solution branches to the order conditions (see, for example, [3, Section II.1]). For brevity, we deal only with the main solution branch in terms of c_2 and c_3. The other solution branches can be treated similarly.

Define a function for computing the gradient of a scalar or vector valued function.

```
In[28]:= Grad[f_?VectorQ,vars_?VectorQ] := Outer[D,f,vars];
In[29]:= Grad[f_,vars_?VectorQ] := Map[D[f,#]&,vars];
```

Using a process similar to Section 4.1, it can be shown that the square of the Euclidean norm of the principal error terms is given by the following expression.

$$\begin{aligned}\text{In[30]:= num} = {} & 2187 - 16092\,c_2 + 48573\,c_2^2 - 79632\,c_2^3 + 79196\,c_2^4 \\ & - 48080\,c_2^5 + 14240\,c_2^6 - 11394\,c_3 + 81756\,c_2\,c_3 - 235872\,c_2^2\,c_3 \\ & + 358208\,c_2^3\,c_3 - 321196\,c_2^4\,c_3 + 179848\,c_2^5\,c_3 - 53600\,c_2^6\,c_3 \\ & + 23355\,c_3^2 - 164988\,c_2\,c_3^2 + 460716\,c_2^2\,c_3^2 - 653460\,c_2^3\,c_3^2 \\ & + 519332\,c_2^4\,c_3^2 - 255704\,c_2^5\,c_3^2 + 75560\,c_2^6\,c_3^2 - 22248\,c_3^3 \\ & + 157004\,c_2\,c_3^3 - 434840\,c_2^2\,c_3^3 + 598584\,c_2^3\,c_3^3 - 441736\,c_2^4\,c_3^3 \\ & + 195568\,c_2^5\,c_3^3 - 57120\,c_2^6\,c_3^3 + 8240\,c_3^4 - 58960\,c_2\,c_3^4 \\ & + 166380\,c_2^2\,c_3^4 - 234160\,c_2^3\,c_3^4 + 177680\,c_2^4\,c_3^4 - 81600\,c_2^5\,c_3^4 \\ & + 24480\,c_2^6\,c_3^4;\end{aligned}$$

```
In[31]:= den = 207360(-1 + 2c2)^2(3 - 4c2 - 4c3 + 6c2c3)^2;
In[32]:= objfun = num/den;
```

Define the free parameters.

```
In[33]:= vars = {c2,c3};
```

Find the gradient vector with respect to the free parameters.

```
In[34]:= minterms = Together[Grad[objfun, vars]];
```

Necessary conditions for a minimum can be derived by setting the components of the gradient to zero and solving the resulting equations. Here, we use a new feature of the function **Reduce** that finds only real valued solutions of a system of equations.

```
In[35]:= eqs = And[Part[minterms, 1] == 0, Part[minterms, 2] == 0,
              c2 ∈ Reals, c3 ∈ Reals];
```

This shows the time taken to find the real solutions, of which there are four. The solutions are fairly complicated so we have chosen not to display them.

```
In[36]:= Timing[Length[sols = {ToRules[Reduce[eqs, vars]]};]]
Out[36] = {25. Second, 4}
```

In order to determine which of the solutions, if any, is a minimum we need to compute the Hessian matrix and see if it is positive definite. Derive the Hessian matrix.

```
In[37]:= Hess = Together[Grad[minterms, vars]];
```

Define a function for determining if a matrix is positive definite.

```
In[38]:= PositiveDefinite[a_?MatrixQ] := Apply[And,Positive[Eigenvalues[a]]];
```

Substituting the solutions into the Hessian matrix we can see that only three of them yield a positive definite matrix. Numerical approximations to the solutions are displayed.

```
In[39]:= minsols = Select[N[sols],PositiveDefinite[Hess /.  #]&]
```

$$\begin{aligned}\text{Out[39]} = \{ & \{c_2 \to 0.357739,\ c_3 \to 0.59149\}, \\ & \{c_2 \to 0.0747369,\ c_3 \to 0.773749\}, \\ & \{c_2 \to 0.57668,\ c_3 \to 2.57771\}\}\end{aligned}$$

The first minimum yields the smallest value of the objective function.

```
In[40]:= objfun /.  minsols
Out[40] = {0.000143459, 0.000315109, 0.177558}
```

The exact value of the global minimum corresponds to roots of degree 28 polynomials.

5. NEW METHODS

New formula pairs with stiffness detection are described in this section and relevant properties are compared with some existing methods.

5.1. Methods of Type 2(1)

For methods of the form (6), the reduced order equations for a second order scheme are simply

$$b_1 + b_2 = 1, \qquad b_2 = \frac{1}{2},$$

so that the unique solution, $b_1 = b_2 = 1/2$, yields a method based on the improved Euler scheme (see [4, p. 155]). This method satisfies $\|T^{(3)}\| = \sqrt{5}/12 \approx 0.186339$. In contrast, it is a simple exercise to show that the optimal two stage order two explicit Runge-Kutta method occurs at $c_2 = 2/3$ so that $b_1 = 1/4$, $b_2 = 3/4$, and $\|T^{(3)}\| = 1/6 \approx 0.166667$. This coincides with the minimal solution of Ralston [14]. The requirement of stiffness detection has therefore slightly compromised the accuracy of the formula. Using (5) we can quantify the expected relative performance of the stiffness equipped scheme with the optimal scheme and show that the difference is less than 3.8%.

It remains to choose a suitable embedded scheme. First order requires that the consistency condition is satisfied which determines $\hat{b}_1$ in terms of $\hat{b}_2$ and $\hat{b}_3$. After simplification, the procedure outlined in Section 4.2 then yields the following objective function to be minimized:

$$B^2 + C^2 = \frac{17 - 60\hat{b}_2 + 72\hat{b}_2^2 - 84\hat{b}_3 + 144\hat{b}_2\hat{b}_3 + 144\hat{b}_3^2}{36\left(2\hat{b}_2 + 2\hat{b}_3 - 1\right)^2}.$$

It is straightforward to show that the unique solution for a minimum is attained at $\hat{b}_2 = -1/6$, $\hat{b}_3 = 1/6$ so that the resulting 2(1) method is given by

0	0	0	0
1	1	0	0
1	$\frac{1}{2}$	$\frac{1}{2}$	0
	$\frac{1}{2}$	$\frac{1}{2}$	0
	1	$-\frac{1}{6}$	$\frac{1}{6}$

There is only one principal error term for the embedded method which is nonzero.

5.2. Methods of Type 3(2)

The optimal third-order formula, which minimizes $\|T^{(4)}\|$, was derived in [20]. Some other methods, none of which have the requisite structure for stiffness detection, are listed in the Appendix.

The equations for a third-order method of the form (6) together with the square of the Euclidean norm of the weighted principal error were given in Section 4.1 along with the minimal solution. A procedure for deriving an embedded method of order two was given in Section 4.2. The resulting 3(2) method is as follows.

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
1	-1	2	0	0
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	0
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	0
	$\frac{22-\sqrt{82}}{72}$	$\frac{14+\sqrt{82}}{36}$	$\frac{-4+\sqrt{82}}{144}$	$\frac{16-\sqrt{82}}{48}$

(7)

There are two principal error terms for the embedded method and both are nonzero.

Table 1 gives error coefficients of some existing pairs and the new scheme. In comparison, the optimal third-order formula has $\|T^{(4)}\| \approx 0.0418091$ [20]. Again using (5) the expected relative performance of the stiffness equipped scheme with the optimal scheme is within 9%.

Table 1. Error coefficients for 3(2) methods.

Method	$\|T^{(4)}\|$	B	C
(9)	0.0418111	1.34919	1.37721
(10)	0.0607170	1.16428	0.928293
(7)	0.0589256	0.444795	1.08853

5.3. Methods of Type 4(3)

It is well known that a necessary condition for four stage fourth-order methods is $c_4 = 1$ (see [2,3]). Therefore, there is no loss of generality in considering the form of stiffness detection considered here for a five stage FSAL scheme.

Ralston used a numerical procedure to search for the optimum four stage fourth-order method; despite the different weighting of the principal error, the proof in Section 4.3 shows that the approximate coefficients found by Ralston correspond to the minimal method [14]. Since the optimal method involves very complicated coefficients, we decided to select the method with $c_2 = 2/5$, $c_3 = 3/5$ for which four of the fifth-order error terms vanish (a choice also found by Ralston). We then used a procedure along the lines of Section 4.2 to equip the scheme with a reasonable embedded method.

$$
\begin{array}{c|ccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{2}{5} & \frac{2}{5} & 0 & 0 & 0 & 0 \\
\frac{3}{5} & -\frac{3}{20} & \frac{3}{4} & 0 & 0 & 0 \\
1 & \frac{19}{44} & -\frac{15}{44} & \frac{10}{11} & 0 & 0 \\
1 & \frac{11}{72} & \frac{25}{72} & \frac{25}{72} & \frac{11}{72} & 0 \\
\hline
 & \frac{11}{72} & \frac{25}{72} & \frac{25}{72} & \frac{11}{72} & 0 \\
\hline
 & \frac{1251515}{8970912} & \frac{3710105}{8970912} & \frac{2519695}{8970912} & \frac{61105}{8970912} & \frac{119041}{747576}
\end{array}
\tag{8}
$$

There are four principal error terms for the embedded method and they are all nonzero.

Table 2 gives error coefficients of the new scheme and a method of Nørsett given in the Appendix. In comparison, the optimal fourth-order formula has $\|T^{(5)}\| \approx 0.0119775$.

Table 2. Error coefficients for 4(3) methods.

Method	$\|T^{(5)}\|$	B	C
(11)	0.0120655	1.03353	1.14612
(8)	0.0123216	0.830311	1.14218

6. CONCLUDING REMARKS

A number of new low order schemes have been derived which are equipped with stiffness detection and possess desirable properties when used in adaptive step mode. The derivation

process is a departure from standard numerical techniques and because of it precise statements about existence and quality of solutions can be made.

The criteria chosen here are not sufficient to derive some methods of choice at high order (see [21]). A further restriction that is often considered to reduce rounding error propagation is that the magnitude of the coefficients does not become too large. It has not been necessary to enforce such restrictions for the schemes derived here, but constraints could readily be added as inequalities in `Minimize` during the optimization process.

The derivation of explicit Runge-Kutta methods with extended stability domains has not been considered. If the method is often working near the border of the stability domain then explicit methods based on Chebyshev nodes appear to be more suitable [22–26].

Shampine suggests that it may be desirable that the linear stability regions of the higher and lower order methods are closely matched [5]. This can help to damp out oscillations when applied to mildly stiff problems. We have chosen instead to use a PI controller of Gustafsson *et al.* [27,28] which has the advantage that, since it is external to the method, it does not compromise the choice of method coefficients. A MATHEMATICA package for the analysis of stability of numerical methods for differential equations is described in [29].

Dense output has not been discussed since, for the low order schemes considered here, Hermite cubic interpolation is usually sufficient.

The construction of Runge-Kutta methods is a fairly involved and mathematically challenging topic despite the elegant formalism of Butcher [30]. We have not discussed issues related to the derivation of high order methods and in particular the use of Butcher's simplifying assumptions which reduce the nonlinearity of the algebraic equations [2,3,31]. Details relating to the implementation and use of simplifying assumptions in the package `Butcher.m` are given in [17,32].

There are several questions that can be raised for future investigation. What is the relative performance of the optimal fourth-order five stage scheme compared with the fourth-order four stage scheme given here? Can the process be extended to derive higher order optimal methods? Certainly the solution process of Section 4.2 can be used to derive useful embedded schemes given a high order solution. The derivation of explicit Runge-Kutta methods of order five and higher remains very challenging and the approach that has been explored in this article is not yet able to supplant numerical techniques.

APPENDIX

Some methods that have been used for comparison are now given. None of these methods cater for the form of stiffness detection that has been chosen here.

A popular 3(2) pair of Bogacki and Shampine is used in the Texas Instruments TI-85 pocket calculator, MATLAB, and RKSUITE [33].

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{3}{4} & 0 & \frac{3}{4} & 0 & 0 \\ 1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\ \hline & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\ \hline & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8} \end{array} \tag{9}$$

The method is based on a third-order formula of Ralston [14] who used a variant of the principal local truncation error, with a slightly different weighting, in the derivation of his methods.

A three stage 3(2) method of Sharp and Smart, which is not a FSAL pair, is given by [7].

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 0 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{1}{2} & 0 & \frac{1}{2} \end{array} \tag{10}$$

A 4(3) pair of Nørsett, which is not FSAL pair, is given by [34].

$$\begin{array}{c|cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{3}{8} & 0 & 0 & 0 & 0 \\ \frac{9}{16} & 0 & \frac{9}{16} & 0 & 0 & 0 \\ \frac{25}{32} & -\frac{125}{672} & \frac{325}{326} & 0 & 0 & 0 \\ 1 & \frac{371}{891} & -\frac{200}{297} & \frac{1120}{891} & 0 & 0 \\ \hline & \frac{25}{162} & \frac{32}{135} & \frac{256}{567} & 0 & \frac{11}{70} \\ \hline & \frac{37}{225} & \frac{44}{117} & 0 & \frac{448}{975} & 0 \end{array} \tag{11}$$

REFERENCES

1. J.C. Butcher, Order, stepsize and stiffness switching, *Computing* **44**, 209–220, (1990).
2. J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, John Wiley and Sons, New York, (1987).
3. E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd edition, Springer-Verlag, New York, (1993).
4. J.D. Lambert, *Numerical Methods for Ordinary Differential Systems. The Initial Value Problem*, John Wiley and Sons, New York, (1991).
5. L.F. Shampine, *Numerical Solution of Differential Equations*, Chapman and Hall, New York, (1994).
6. L.F. Shampine, Some practical Runge-Kutta formulae, *Math. Comp.* **46** (173), 135–150, (1986).
7. P.W. Sharp and E. Smart, Explicit Runge-Kutta pairs with one more derivative evaluation than the minimum, *SIAM J. Sci. Comp.* **14** (2), 338–348, (1993).
8. L.R. Petzold, Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations, *SIAM J. Sci. Stat. Comp.* **4**, 136–148, (1983).
9. B.C. Robertson, Detecting stiffness with explicit Runge-Kutta methods, Report 193/87, Dept. Comp. Sci. University of Toronto, (1987).
10. G. Sottas, Dynamic adaptive selection between explicit and implicit methods when solving ODEs, Report Sect. de Math., Université de Genève (1984).
11. L.F. Shampine and K.L. Hiebert, Detecting stiffness with the Fehlberg (4, 5) formulas, *Computers Math. Applic.* **3**, 41–46, (1977).
12. T.E. Hull and R.L. Johnston, Optimum Runge-Kutta methods, *Math. Comp.* **18**, 306–310, (1964).
13. T.E. Hull, A search for optimum methods for the numerical integration of ordinary differential equations, *SIAM Rev.* **9**, 647–654, (1967).
14. A. Ralston, Runge-Kutta methods with minimum error bounds, *Math. Comp.* **16**, 431–437, (1962).
15. G.E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, Springer Lecture Notes in Computer Science 33, MR 55, pp. 134–183, (1977).
16. J.H. Davenport, Y. Siret and E. Tournier, *Computer Algebra: Systems and Algorithms for Algebraic Computation*, 2nd edition, Academic Press, London, (1993).

17. M. Sofroniou, Symbolic derivation of Runge-Kutta methods, *J. Symb. Comp.* **18** (3), 265–296, (1994).
18. D.A. Cox, J.B. Little and D. O'Shea, *Ideals, Varieties and Algorithms*, 2nd edition, Springer, New York, (1997).
19. J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, (1999).
20. M. Sofroniou and G. Spaletta, Computer generation of numerical methods for ordinary differential equations, In *Recent Trends in Numerical Analysis*, (Edited by D. Trigiante), Advances in the Theory of Computational Mathematics, pp. 315–328, Nova Science Publishers, Huntington, NY, (2000).
21. J.H. Verner, Some Runge-Kutta formula pairs, *SIAM J. Numer. Anal.* **28** (2), 496–511, (1991).
22. A. Abdulle, Chebyshev methods based on orthogonal polynomials, PhD Thesis, Université de Genève (2001).
23. P.J. van der Houwen and B.P. Sommeijer, On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values, *Z. Angew. Math. Mech.* **60**, 479–485, (1980).
24. V.I. Lebedev, Explicit difference schemes with time-variable steps for solving stiff systems of differential equations, *Sov. J. Numer. Anal. Math. Modelling* **4** (2), 111–135, (1989).
25. V.I. Lebedev, How to solve stiff systems of differential equations by explicit methods, In *Numerical Methods and Applications*, (Edited by G.I. Marchuk), pp. 45–80, CRC Press, (1994).
26. A.A. Medovikov, High order explicit methods for parabolic equations, *BIT* **38**, 372–390, (1998).
27. K. Gustafsson, M. Lundh and G. Söderlind, A PI stepsize control for the numerical solution of ordinary differential equations, *BIT* **28**, 270–287, (1988).
28. K. Gustafsson, Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods, *ACM Trans. Math. Soft.* **17**, 533–554, (1991).
29. M. Sofroniou, Order stars and linear stability theory, *J. Symb. Comp.* **21** (1), 101–131, (1996).
30. K. Burrage, The work of John Butcher: An appreciation, *Annals of Numerical Mathematics* **1**, 1–24, (1994).
31. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*, 2nd edition, Springer-Verlag, New York, (1996).
32. MATHEMATICA 5 *Standard Add-On Packages*, Wolfram Media, Champaign, IL, (2004).
33. P. Bogacki and L.F. Shampine, A 3(2) pair of Runge-Kutta formulas, *Appl. Math. Lett.* **2**, 1–9, (1989).
34. W.H. Enright, K.R. Jackson, S.P. Nørsett and P.G. Thomsen, Interpolants for Runge-Kutta formulas, *ACM Trans. Math. Soft.* **12** (3), 193–218, (1986).