High Order Explicit RK Schemes

Richard Fears and Jason Gregersen

Purpose

To explore the construction of high order explicit RK schemes

Introduction

The modern organization of RK schemes presented by Butcher ([1] pg 94,95) utilizes a Butcher tableau in the following form. $\begin{pmatrix} c & A \\ 0 & b \end{pmatrix}$ This leads to the following algorithm for an RK scheme. $y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$ where: $k_i = f\left(\mathrm{yn} + h \sum_{j=1}^{s} a_{\mathrm{ij}} k_j\right)$

Thus to determine an RK scheme one just needs to construct the proper entries into the table to be used in the algorithm. The number of rows of A is the number of "stages" of the scheme. The order of the scheme is related to the order of accuracy for the scheme. In general we need at least as many stages as the order, but as the desired order increases the number of stages required becomes larger than the desired order. Coefficients that lead to schemes of a given order are found by satisfying a set of equations knows as order conditions.

Order Conditions

To establish the correct order conditions we can compare the Taylor series expansion of the exact solution and the approximate solution. It is through this process that theory supporting the appropriate order conditions was established. As a method to actually calculating the appropriate order conditions we can look at the "Butcher trees". Each order is associated with a family of trees. The tree has as many vertices as the order. Each tree leads to an equation. Thus to form the order conditions for a third order RK scheme one has to look at the family of trees with one vertex and form the appropriate order condition, then move on to the family of trees with two vertices and the family of trees with three vertices. The process of creating an order condition from a specific tree is detailed at the following website; http://www.123mathvids.com/vids/rkordereqn/rko-rdereqn.html As an example, the order conditions for a third order scheme with "s" stages is shown below:

```
<< NumericalDifferentialEquationAnalysis`
Eqs = RungeKuttaOrderConditions[3, s];
```

```
Eqs
```

$$\left\{\left\{\sum_{K[1]=1}^{s} \dot{b}_{K[1]} == 1\right\}, \left\{\sum_{K[1]=1}^{s} \dot{b}_{K[1]}\, \dot{c}_{K[1]} == \frac{1}{2}\right\},\right.$$

$$\left.\left\{\sum_{K[1]=1}^{s} \dot{b}_{K[1]} \sum_{K[2]=1}^{s} \dot{c}_{K[2]}\, \dot{a}_{K[1],K[2]} == \frac{1}{6}, \sum_{K[1]=1}^{s} \dot{b}_{K[1]}\, \dot{c}_{K[1]}^{2} == \frac{1}{3}\right\}\right\}$$

Here we can see that only one condition was formed for the first and second order, as there exists only one tree in each of those groups. Using this process we can construct a set of order equations, but since this becomes a large set of nonlinear equations quickly (for a 12th order method there are 7813 order conditions [3]) it becomes difficult to solve this system and then construct the associated high order RK schemes. Additional conditions are generally added, but an optimal strategy for all cases is not defined.

- **Project Outline**

1. We will construct and solve the order equations needed to form a third order scheme.

2. We will discuss reasons for choosing choosing parameters to form specific schemes and compare our choice to those of others.

3. We will perform the same analysis for a 5th order 6 stage scheme.

4. We will discuss trying to solve larger systems by forming a minimization problem.

Code

- **Function**
- **Initial Values**
- **Tableau's from the previous homework**
- **General ODE solver**
- **Step Size Adjuster**
- **Drivers**
- **Misc Code**

Small Scheme Creation and Special Properties

- **Creating Small Schemes**

In order to create an RK scheme we first need to establish and then solve the necessary order equations as derived earlier. We started with a straight forward third order three stage scheme. Adding in the standard Butcher Simplifications (column sum, row sum) and specifying that the scheme be explicit results in a nonlinear system of 8 equations and 9 variables. Given the small size of this system we are able to solve it directly using the “Solve” command in mathematica. The result is:

$$\left\{\dot{b}_1 \to \frac{-1+3\,\dot{c}_2}{6\,\dot{c}_2},\ \dot{c}_1 \to 0,\ \dot{a}_{2,1} \to \dot{c}_2,\ \dot{a}_{3,1} \to \frac{1-3\,\dot{c}_2+3\,\dot{c}_2^2}{\dot{c}_2\,(-2+3\,\dot{c}_2)},\right.$$
$$\left.\dot{a}_{3,2} \to \frac{-1+\dot{c}_2}{\dot{c}_2\,(-2+3\,\dot{c}_2)},\ \dot{b}_3 \to \frac{-2+3\,\dot{c}_2}{6\,(-1+\dot{c}_2)},\ \dot{b}_2 \to -\frac{1}{6\,(-1+\dot{c}_2)\,\dot{c}_2},\ \dot{c}_3 \to 1\right\}$$

- **Choosing Parameters, error vs stability**

Thus any choice of $\dot{c}_2$ should yield a third order RK scheme. We are now going to explore some options in choosing a value of $\dot{c}_2$. It is standard practice to calculate the Principal Error terms for this scheme. That is, to determine the accuracy of the scheme we look at the difference between one step of the scheme to the taylor series expansion at that point. Since our values satisfy the order equations for a third order scheme all of the terms to that power will be eliminated and the values in the next highest power (the first group not eliminated) are the principal error terms. When we take this error and replace the values with our solution we will get the principal err as a function of our parameter.

■

$$\left\{-\frac{1}{24}, \frac{1}{2}\left(-\frac{1}{12}+\frac{\dot{c}_2}{6}\right), \frac{1}{24}, \frac{1}{6}\left(-\frac{1}{4}-\frac{\dot{c}_2^2}{6\,(-1+\dot{c}_2)}+\frac{-2+3\,\dot{c}_2}{6\,(-1+\dot{c}_2)}\right)\right\}$$

■ **Error minimization**

One method for choosing a value for $\dot{c}_2$ is to choose it so that it minimizes the error term. Typically this is done with the L2 Norm, but we would like choose a few others and compare the results. Using the L1,L2,L3, and L∞ we find the results for the value of the optimal $\dot{c}_2$ in each case. Since the difference between the L1, L2, and L3 norm are minimal. We will only compare the results using the L2 Norm, L∞ norm, and a case where we choose a value at random. We then put these optimal values back into our solution and form the appropriate Tableaus. Lastly we compare the results. By comparing our answer to the analytic solution to the Keplar two body problem for varying stepsizes. The results are show below.

■

$$\left\{\left\{\dot{c}_2 \to \frac{1}{2}\right\}, \left\{\dot{c}_2 \to \frac{1}{2}\right\}, \left\{\dot{c}_2 \to \frac{1}{2}\right\}, \{\dot{c}_2 \to 0.039709928902375415166\}\right\}$$

■ **Tableau Formation Code**

To use these parameter values to form the Talbeaus we first form the solution by replacing the parameter with the appropriate value. Next, we can automate the process of putting our solution into a Tableau form that will be useable by our odesolvers. Note that here we have entered a second row of "b's". This is so our talbeaus have the correct form to be implemented by ODESolver. The choice of values in the second row only effects the error estimate, so as long as we only used a fixed step driver to test the tableau we should be fine.

■

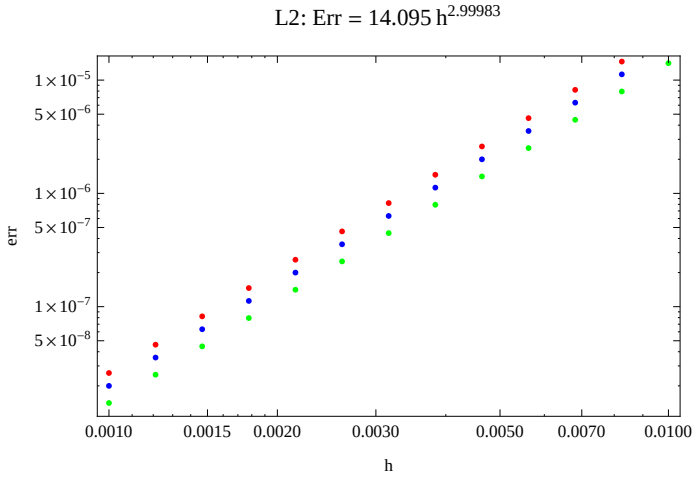
■

$$\left\{\left\{\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ -1 & 2 & 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 \end{pmatrix}\right\}, \left\{\begin{pmatrix} 0 & 0 & 0 \\ 0.039709928902375415166 & 0 & 0 \\ -11.857143770799938969 & 12.857143770799938969 & 0 \end{pmatrix}, \begin{pmatrix} -3.6971031244202707261 & 4.3706617934963389651 & 0.32644133092393176097 5 \\ 0 & 0 & 0 \end{pmatrix}\right\},$$

$$\left\{\begin{pmatrix} 0 & 0 & 0 \\ 0.660084 & 0 & 0 \\ -25.0778 & 26.0778 & 0 \end{pmatrix}, \begin{pmatrix} 0.247507 & 0.742811 & 0.00968228 \\ 0 & 0 & 0 \end{pmatrix}\right\}\right\}$$

■ **Results for GF33....**

The graph below show the convergence of the three cases. Here the green points are for the parameter chosen for optimality in the L2 norm,, the blue is for the Linf norm and the red is the parameter chosen randomly from the interval [0,2].



- **Stability for GF33**

Another option for choosing $\dot{c}_2$ would be to choose a value that maximized the stability region? To do this we will form the talbeau maintaining the parameter. The resulting talbeau is:

$$\left\{\begin{pmatrix} 0 & 0 & 0 \\ \dot{c}_2 & 0 & 0 \\ \frac{1-3\dot{c}_2+3\dot{c}_2^2}{\dot{c}_2\left(-2+3\dot{c}_2\right)} & \frac{-1+\dot{c}_2}{\dot{c}_2\left(-2+3\dot{c}_2\right)} & 0 \end{pmatrix}, \begin{pmatrix} \frac{-1+3\dot{c}_2}{6\dot{c}_2} & -\frac{1}{6\left(-1+\dot{c}_2\right)\dot{c}_2} & \frac{-2+3\dot{c}_2}{6\left(-1+\dot{c}_2\right)} \\ 0 & 0 & 0 \end{pmatrix}\right\}$$

By doing the standard stability analysis, we generate a function R(z) which is equal to the third order polynomial shown below.

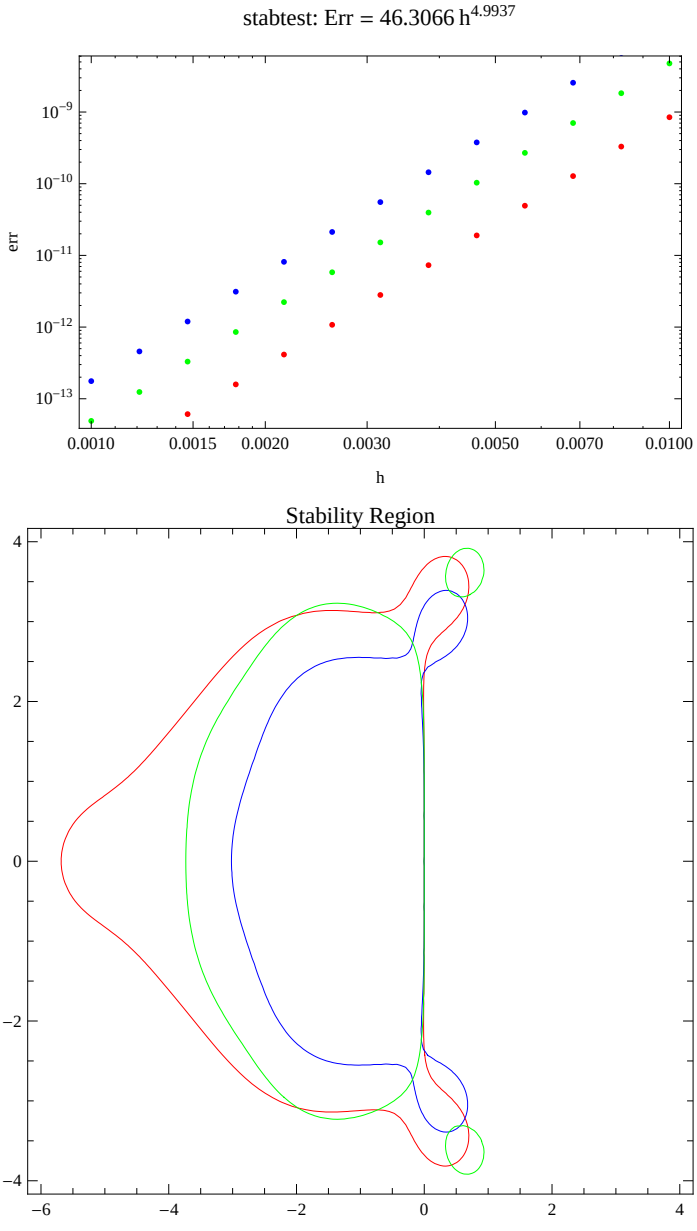
$$\frac{1}{6}\left(6+6\,z\$+3\,z\$^2+z\$^3\right)$$

The important fact here is that this polynomial is independent of $\dot{c}_2$, thus it doesn't provide any insight into choosing $\dot{c}_2$. This phenomena is explained in [1, pg 101], and holds whenever the order and stage number of the scheme are the same. Since schemes like that are restriced to having an order four or less, we will look at a larger and more interesting case. Looking at a fifth order six stage scheme, we find that mathematica already has trouble finding an analytic solution using the Solve command. To compensate, we will choose the values of "c" to be evenly spaced values between 0 and 1. Solve then yields a solution of one parameter. The resulting stability function maintains that parameter which implies that different values will yield different stability regions for the resulting 5th order scheme. By trial and error we determine that by setting the parameter $\dot{a}_{4,3}$ equal to -.165, we get an improved stability region to other comparable schemes. The resulting tableau which we will call GF56 is shown below alongside the accuracy and stability comparision of GF56 (Red) to Fehlberg (blue) and CashKarp (Green).

- **Tableau creation for GF56**

- **GF56 Results...**

$$\left\{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{8} & 0 & 0 & 0 & 0 \\ -0.0825 & 0.914167 & -0.165 & 0 & 0 & 0 \\ -0.252546 & 0.965972 & 0.328241 & -\frac{5}{24} & 0 & 0 \\ -1.04781 & 2.13094 & -0.420625 & \frac{27}{16} & -\frac{27}{20} & 0 \end{pmatrix}, \begin{pmatrix} \frac{41}{300} & 0 & \frac{19}{15} & -\frac{27}{20} & \frac{27}{25} & -\frac{2}{15} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right\}$$



Higher Order Schemes

As we attempt to create higher order schemes, the number of equations quickly becomes large and difficult to solve. For a 6,8 scheme, even choosing all of the c values still doesn't allow us to solve the system using the Solve command. To account for this problem we will focus our attention on one specific order equations from each order family[4]. Specifically looking at the equation $\sum_i^8 b_i c_i = \frac{1}{\gamma}$, [1, pg 94,95] if we choose the c values we can solve for 6 of the 8 b's. At this point we can once again use the Solve command to find the nice analytic solution to the system. In this case after making all of our assumptions, our solution is still a function of 4 parameters. To choose these parameters we will once again minimize the principal error terms via the L2 norm. After doing this we arrive at the following scheme called "GF68". Once created we would like to test the scheme. To do this we will compare its accuracy and stability to Dormand Prince which is the most

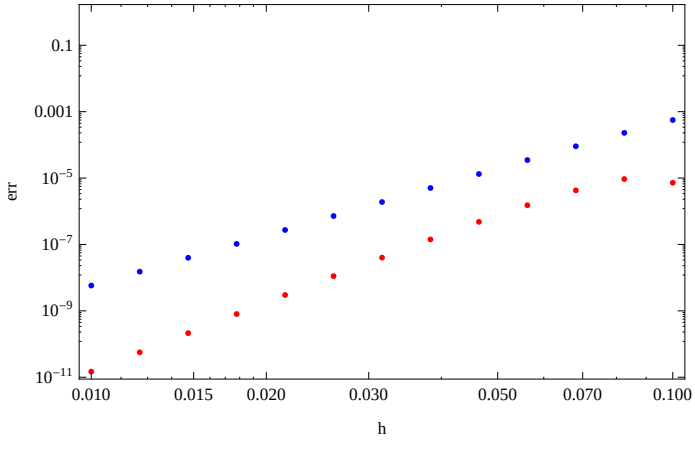
commonly used 6,8 scheme. The results shown below indicate that our scheme converged slightly faster but did have a smaller stability region.

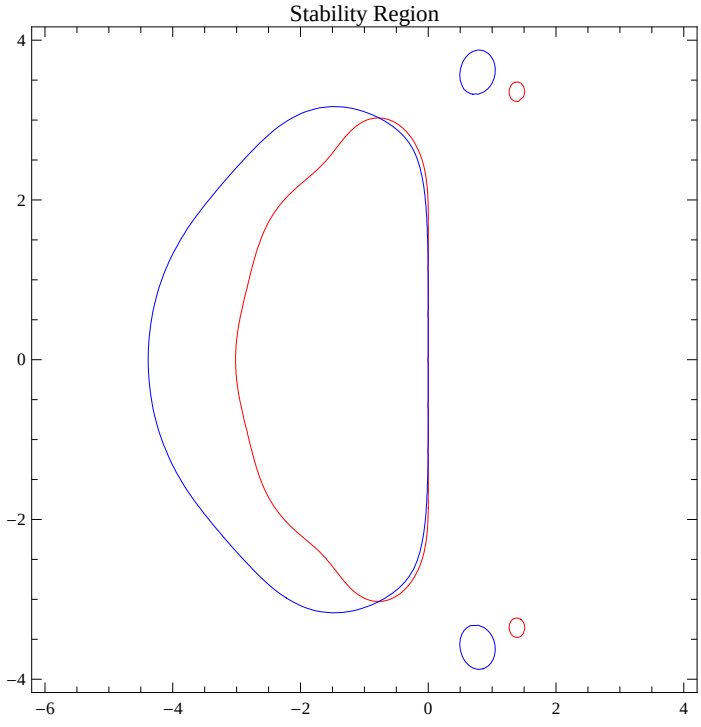
- **6th order 8 stage scheme using Solve**
- **Forming the equations**
- **Solving the b's and c's...**
- **Minimizing the Principal Error with four parameters**
- **GF68 Results...**

$$\left\{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.462764 & -0.888292 & 0.925528 & 0 & 0 & 0 & 0 & 0 \\ 0.170743 & 0.118602 & 0.0173256 & 0.31833 & 0 & 0 & 0 & 0 \\ 0.0691201 & 0.381658 & -0.392506 & 0.794438 & -0.10271 & 0 & 0 & 0 \\ 0.00690324 & 0.69925 & -1.45594 & 1.7083 & 0.300282 & -0.383802 & 0 & 0 \\ 0.042488 & 0.652721 & -1.79478 & 2.22843 & -0.410223 & 0.753244 & -0.471884 & 0 \end{pmatrix},\right.$$

$$\left.\begin{pmatrix} 0.104414 & 8.88178\times10^{-16} & 0.64173 & 0.393511 & -1.83851 & 2.48024 & -1.06296 & 0.281573 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right\}$$

GF68: Err = $41.5356\, h^{6.11728}$





3rd order 3 stage scheme using the minimization technique

Larger/ small Systems

Having run out of options for extending our ability to use the Solve command, we will look at solving the system numerically by re-casting it as a minimization problem. To do this we will subtract the γ(t) [1, pg 97] portion of the order equations to the other side giving us a large nonlinear system of equations all set to be equal zero. We will then minimize $\frac{1}{2} \left| \phi(x) \right|^2$. The result might be a solution to $\phi = 0$ and thus a solution to our system. As an example, we look at a small 3rd order 3stage problem.

Minimization Equations Code

First we set up the equations the same way we did as before. Next we will create a function using all the information from order equations. We will then perform the minimization as described above. We will do this twice, once with the default tolerances “minsolnonacc” and once with a higher working precision “minsolacc”. To test to see if the minimization does in fact lead to the solution to our system, we will replace the values in the equations with those found in the minimization process. The result should be zero. As shown below, the actual results will depend on the minimization precision.

- **...**

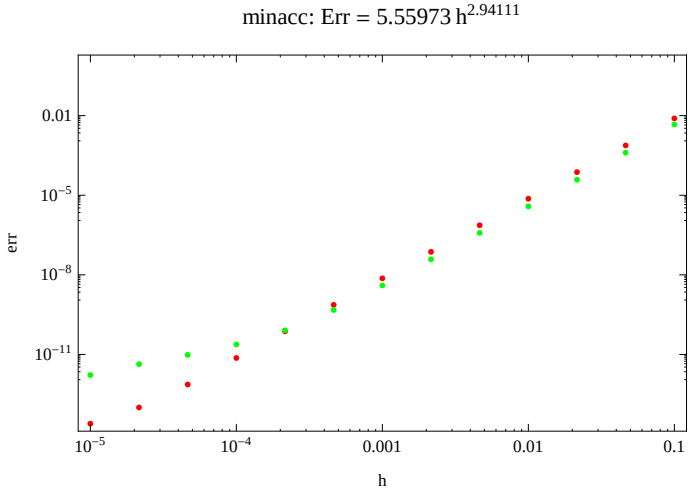
```
mineqs /. minsolacc[[2]]
mineqs /. minsolnonacc[[2]]
```

$$\{-4.8525765671028032241\times 10^{-21},\ -7.4017002283016604621 1\times 10^{-20},\ 8.9780371250951725010 0\times 10^{-20},\ -2.3130503665483967032\times 10^{-20},\ -1.5708237081990297976 75\times 10^{-19},\ 4.9999999999999929549188965346718123568 72\times 10^{-21},\ -5.4217156645910206041\times 10^{-21},\ -5.6238293564265088 93\times 10^{-21}\}$$

$$\{-1.15412\times 10^{-8},\ -8.28432\times 10^{-8},\ 1.24369\times 10^{-7},\ -4.39345\times 10^{-8},\ -1.8592\times 10^{-7},\ -6.82078\times 10^{-9},\ -5.36671\times 10^{-9},\ -1.98593\times 10^{-9}\}$$

- **Form talbeaus**
- **Testing Programs...**

  After forming the Tableau's using the minimization process above, we check that the schemes really give us the correct convergence. As we can see in the graph below, both schemes converge like 3rd order, but the scheme formed from the less accurate minimization (green) begins to lose it convergence rate sooner that the scheme formed from the more accurate minimization.



- **Large Systems Using minimization techniques**

  The next step is to take this process and try to find higher order scheme. We attempt to find an 8th order 15 stage scheme. We tried several variations of Nminimize, FindMinimum, and FindInstance, and even some combination such as using a low precision NMinimize to form initial conditions for a FindMinimum. In all cases either the command didn't finish, we recieved precision errors, or we formed schemes that did not achieve the desired accuracy. In summary we still need either more assumptions or a better technique to handle the large nonlinear systems formed by these order equations.

- **Setting up the equations**
- **Results**
- **....**

References

[1] Butcher J. C. (2008). Numerical Methods for Ordinary Differential Equations. Wiley
[2] Butcher J.C. (). The non-existence of ten stage eighth order explicit Runge-Kutta methods.

BIT Numerical Mathematics, 25 (3), 521-540
[3] http://reference.wolfram.com/mathematica/NumericalDifferentialEquationAnalysis/tutorial/NumericalDifferentialEquationAnalysis.html
[4] Butcher J.C. (2008). http://www.math.auckland.ac.nz/~butcher/ODE-book-2008/Tutorials/IRK.pdf