Runge–Kutta Methods

Again we consider the initial value problem (IVP)
\[ x' = f(t, x(t)) \]
\[ x(t_0) = x_0 \]
where \( t \) is usually time, and \( x(t) \) is a vector valued function.

To achieve the approximation \( x_n = x(t_n) \) where \( t_n = t_0 + nh \). Each step of an \( s \)-stage Runge–Kutta method is

\[ k_i = f(t_n + c_i h, x_n + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 1, \ldots, s \]
\[ x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \]

Each stage is dependent upon the previous stages.

A compact notation for this is the Butcher array (named after John Butcher who retired for University of Auckland in New Zealand). For a four stage method the Butcher array looks like so.

\[
\begin{array}{|cccc|}
\hline
 c_1 & 0 & 0 & 0 & 0 \\
 c_2 & a_{21} & 0 & 0 & 0 \\
 c_3 & a_{31} & a_{32} & 0 & 0 \\
 c_4 & a_{41} & a_{42} & a_{43} & 0 \\
 \hline
 b_1 & b_2 & b_3 & b_4 \\
\end{array}
\]

For instance the four stage, fourth order method

\[ k_1 = f(t_n, x_n) \quad (1) \]
\[ k_2 = f(t_n + h/2, x_n + h k_1/2) \quad (2) \]
\[ k_3 = f(t_n + h/2, x_n + h k_2/2) \quad (3) \]
\[ k_4 = f(t_n + h, x_n + h k_3) \quad (4) \]
\[ x_{n+1} = x_n + \frac{h}{6}(k_1 + 2(k_2 + k_3) + k_4) \quad (5) \]
has the Butcher array
\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
c_4 & 0 & 0 & 1 & 0 \\
\end{array}
\]
\[
\begin{array}{cccc}
1/6 & 1/3 & 1/3 & 1/6 \\
\end{array}
\]

Order, Local Error, and Global Error

Local error is the truncation error from one step. It is modeled by pretending that the IVP is started at the last step. So take the solution to

\[
\begin{align*}
z'_n(t) &= f(t, z_n(t)) \\
z_n(t_n) &= x_n
\end{align*}
\]

Note that \( z_n(t) \) is a different function for every \( n \).

For Euler’s method,

\[
x_{n+1} = x_n + hf(t_n, x_n) = x_n + hz'_n(t).
\]

Since

\[
z_n(t_{n+1}) = z_n(t_n + h) = x_n + hz'_n(t) + \frac{h^2}{2}z''_n(t) + O(h^3).
\]

Thus

\[
z_n(t_{n+1}) = x_{n+1} + \frac{h^2}{2}z''_n(t) + O(h^3).
\]

This local error is \( O(h^2) \).

However, this is an order 1 method. That is because of global error. For each \( n \) the solution \( x(t) \) to (1)–(1) satisfies

\[
x(t_n) = x_n + \beta_1 h + O(h^2).
\]

Likewise, the Runge–Kutta method (1)–(5) is fourth order. The local error has the form

\[
z_n(t_{n+1}) - x_{n+1} = \alpha_5 h^5 + O(h^6).
\]

The global error has the form

\[
x(t_{n+1}) - x_{n+1} = \beta_4 h^4 + O(h^5).
\]
When we choose the step size, the choice is based upon local error. It may that a good $s$ stage method is of order $s$. Unfortunately, to get a $p$ order method for $p > 4$ requires more than $p$ stages. In the table below, $p$ is the desired order, $s$ is minimum number of stages necessary.

<table>
<thead>
<tr>
<th>$p$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

After order 8, we do not know how to fill in this table. In practice, we do not do much with Runge–Kutta methods of much greater than 5.

To estimate local error we introduce the idea of embedding due to Fehlberg. The Runge–Kutta methods that we have introduced so far are not suitable for embedding.

For that, we introduce a five stage, fourth order method of the form

\[
\begin{align*}
    k_1 &= f(t_n, x_n) \\
    k_2 &= f(t_n + c_1 h, x_n + ha_{21}k_1) \\
    k_3 &= f(t_n + c_3 h, x_n + h(a_{31}k_1 + a_{32}k_2)) \\
    k_4 &= f(t_n + c_4 h, x_n + h(a_{41}k_1 + a_{42}k_2 + a_{43}k_3)) \\
    k_5 &= f(t_n + c_5 h, x_n + h(a_{51}k_1 + a_{52}k_2 + a_{53}k_3 + a_{54}k_4)) \\
    x_{n+1} &= x_n + h \sum_{i=1}^{5} b_i k_i
\end{align*}
\]

It has the Butcher array (unfilled spaces are zeros)

\[
\begin{array}{cccccc}
  0 & & & & & \\
  1/4 & 1/4 & & & & \\
  12/13 & 1932/2197 & -7200/2197 & 7296/2197 & & \\
  1 & 439/216 & -8 & 3680/513 & -845/4104 & \\
  & 25/216 & 0 & 1408/2565 & 2197/4104 & -1/5 \\
\end{array}
\]

It seems as though this method has no advantages over the four stage fourth order method we talked about earlier. However, if we add one more stage

\[
k_6 = f(t_n + h/2, x_n + h \sum_{i=1}^{5} a_{6j}k_i).
\]
where $a_{6j}, j = 1, 2, 3, 4, 5$ are given by

\[-8/27, 2, −3544/2565, 1859/4104, −11/40,\]

then we can construct a fifth order method. It has the form

\[\hat{x}_{n+1} = x_n + h \sum_{i=1}^{6} \hat{b}_i k_i.\]

where $\hat{b}_i, i = 1, 2, 3, 4, 5, 6$ is given by

\[16/135, 0, 6656/12825, 28561/56430, −9/50, 2/55.\]

The first method is said to be embedded in the second. The first method has local truncation error of the form

\[\|z_n(t_{n+1}) − x_{n+1}\| = \alpha_5 h^5 + O(h^6)\]

while the second has local truncation error of the form

\[\|z_n(t_{n+1}) − \hat{x}_{n+1}\| = \alpha_6 h^6 + O(h^7).\]

Thus

\[\|\hat{x}_{n+1} − x_{n+1}\| = \alpha_5 h^5 ± \alpha_6 h^6 + O(h^6) = \alpha_5 h^5 + O(h^6).\]

Thereby,

\[\|\hat{x}_{n+1} − x_{n+1}\| = \|z_n(t_{n+1}) − x_{n+1}\| + O(h^6).\]

We have an estimate of the highest order term of the local error. The only extra effort is that of computing an extra stage.

We represent this method in the modified Butcher array like so.

\[
\begin{array}{c|ccc}
0 & 1/4 & 3/8 & 12/13 \\
1/4 & 3/32 & 9/32 & 1932/2197 \\
3/8 & 439/216 & −8 & −7200/2197 \\
12/13 & −8/27 & 2 & 7296/2197 \\
1 & 1408/2565 & −845/4104 & −3544/2565 \\
1/2 & 1859/4104 & −11/40 & 1859/4104 \\
1 & 2197/4104 & −1/5 & 2/55 \\
16/135 & 28561/56430 & −9/50 & (7)
\end{array}
\]
With an error estimate, we choose the step size $h$ to achieve a certain local error. For instance, for a single equation

$$Err = \|\hat{x}_{n+1} - x_{n+1}\| = \alpha_5 h^5 + O(h^6).$$

We want to choose $\hat{h}$ so that for some prescribed tolerance $tol$,

$$tol = \|\hat{x}_{n+1} - x_{n+1}\| = \alpha_5 \hat{h}^5 + O(h^6).$$

Ignoring the higher order terms, we have

$$tol/Err = \hat{h}^5/h^5.$$ 

Thus a good choice for the step size is

$$\hat{h} = h(tol/Err)^{1/5}.$$ 

To be conservative, we usually use a "fudge" factor, say, about 0.8 and instead choose

$$\hat{h} = 0.8h(tol/Err)^{1/5}.$$ 

Usually, the step size is not changed if $Err$ is within a factor of 2 of $tol$. Otherwise, we do this adjustment.