Runge-Kutta Methods with Rooted Trees

The Motivation of the Project

Richard Douglas

MA489 Research Seminar at Wilfrid Laurier University

November 12, 2013

Outline

A brief review of numerical methods for ODEs

Given an Ordinary Differential Equation of form

$$y'(x) = f(x, y), \forall x \in [a, b]$$

with initial condition $y(a) = y_0$,

A brief review of numerical methods for ODEs

Given an Ordinary Differential Equation of form

$$y'(x) = f(x, y), \forall x \in [a, b]$$

with initial condition $y(a) = y_0$, we can obtain a numerical approximation for $y(b)$ by first dividing the interval $[a, b]$ into N equally spaced subintervals $[x_0, x_1], [x_1, x_2], \dots, [x_{N-1}, x_N]$ where

$$h = \frac{(b-a)}{N}$$

$$x_{n+1} = x_n + h$$

Looking at individual steps

We then compute a sequence of approximations for $y(x)$

$$y_0 = y(x_0), y_1 \approx y(x_1) = y(x_0 + h), \ldots, y_N \approx y(x_N) = y(b)$$

where the last value ends up being an approximation for y when x is at the end of the interval.

Looking at individual steps

We then compute a sequence of approximations for $y(x)$

$$y_0 = y(x_0), y_1 \approx y(x_1) = y(x_0 + h), \ldots, y_N \approx y(x_N) = y(b)$$

where the last value ends up being an approximation for y when x is at the end of the interval.

We already know the value of y_0,

Looking at individual steps

We then compute a sequence of approximations for $y(x)$

$$y_0 = y(x_0), y_1 \approx y(x_1) = y(x_0 + h), \ldots, y_N \approx y(x_N) = y(b)$$

where the last value ends up being an approximation for y when x is at the end of the interval.

We already know the value of y_0, and y_{n+1} can be obtained using its preceding y approximation.

Looking at individual steps

We then compute a sequence of approximations for $y(x)$

$$y_0 = y(x_0), y_1 \approx y(x_1) = y(x_0 + h), \ldots, y_N \approx y(x_N) = y(b)$$

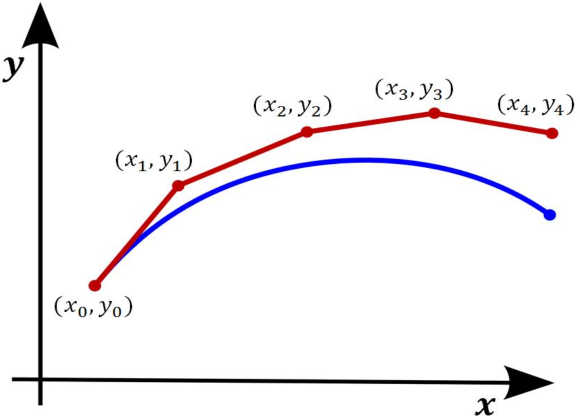
where the last value ends up being an approximation for y when x is at the end of the interval.

We already know the value of y_0, and y_{n+1} can be obtained using its preceding y approximation.

$$y(a) = y_0 \to y_1 \to \cdots \to y_N \approx y(b)$$

Example: Euler's method

$$x_{n+1} - x_n = h$$
$$y_{n+1} = y_n + hf(x_n, y_n)$$



The order of a method

As the stepsize h decreases, the numerical algorithm produces a better approximation for $y(b)$.

The order of a method

As the stepsize h decreases, the numerical algorithm produces a better approximation for $y(b)$.

That is to say

$$\lim_{h\to 0} y_N = y(b)$$

The order of a method

As the stepsize h decreases, the numerical algorithm produces a better approximation for $y(b)$.

That is to say

$$\lim_{h \to 0} y_N = y(b)$$

We say that a method of approximation is of **order p** if

$$\lim_{h \to 0} \frac{|y(b) - y_N|}{h^p} < \infty$$

The order of a method

As the stepsize h decreases, the numerical algorithm produces a better approximation for $y(b)$.

That is to say

$$\lim_{h \to 0} y_N = y(b)$$

We say that a method of approximation is of **order p** if

$$\lim_{h \to 0} \frac{|y(b) - y_N|}{h^p} < \infty$$

More compactly, we say that the method is $\mathbf{O}(\mathbf{h}^{\mathbf{p}})$.

The order of a method

As the stepsize h decreases, the numerical algorithm produces a better approximation for $y(b)$.

That is to say

$$\lim_{h \to 0} y_N = y(b)$$

We say that a method of approximation is of **order p** if

$$\lim_{h \to 0} \frac{|y(b) - y_N|}{h^p} < \infty$$

More compactly, we say that the method is $\mathbf{O(h^p)}$.

The order of a method tells us how fast it converges to the true value of what it is approximating.

Taylor methods

The usual way in which we derive a Runge-Kutta method of a given order is by comparing its terms with a Taylor method of this order.

Taylor methods

The usual way in which we derive a Runge-Kutta method of a given order is by comparing its terms with a Taylor method of this order.

Taylor methods work using the Taylor Series expansion

$$y(x) = y(x_n) + y'(x_n)(x - x_n) + \frac{y''(x_n)(x - x_n)^2}{2!} + \frac{y'''(x_n)(x - x_n)^3}{3!} + \ldots$$

Taylor methods

The usual way in which we derive a Runge-Kutta method of a given order is by comparing its terms with a Taylor method of this order.

Taylor methods work using the Taylor Series expansion

$$y(x) = y(x_n) + y'(x_n)(x - x_n) + \frac{y''(x_n)(x - x_n)^2}{2!} + \frac{y'''(x_n)(x - x_n)^3}{3!} + \ldots$$

So for an ODE with $y'(x) = f(x, y)$ and $y_0 = y(x_0)$, we will have

Taylor methods

The usual way in which we derive a Runge-Kutta method of a given order is by comparing its terms with a Taylor method of this order.

Taylor methods work using the Taylor Series expansion

$$y(x) = y(x_n) + y'(x_n)(x - x_n) + \frac{y''(x_n)(x - x_n)^2}{2!} + \frac{y'''(x_n)(x - x_n)^3}{3!} + \ldots$$

So for an ODE with $y'(x) = f(x, y)$ and $y_0 = y(x_0)$, we will have

$$x_{n+1} - x_n = h$$

Taylor methods

The usual way in which we derive a Runge-Kutta method of a given order is by comparing its terms with a Taylor method of this order.

Taylor methods work using the Taylor Series expansion

$$y(x) = y(x_n) + y'(x_n)(x - x_n) + \frac{y''(x_n)(x - x_n)^2}{2!} + \frac{y'''(x_n)(x - x_n)^3}{3!} + \ldots$$

So for an ODE with $y'(x) = f(x, y)$ and $y_0 = y(x_0)$, we will have

$$x_{n+1} - x_n = h$$

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!} + \ldots + \frac{f^{(p-1)}(x_n, y_n)h^p}{p!}$$

The order of Taylor methods

For a Taylor method with

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!} + \ldots + \frac{f^{(p-1)}(x_n, y_n)h^p}{p!}$$

The order of Taylor methods

For a Taylor method with

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!} + \ldots + \frac{f^{(p-1)}(x_n, y_n)h^p}{p!}$$

The error in making a single step is $O(h^{p+1})$ and the error made across the entire interval is $O(h^p)$.

The order of Taylor methods

For a Taylor method with

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!} + \dots + \frac{f^{(p-1)}(x_n, y_n)h^p}{p!}$$

The error in making a single step is $O(h^{p+1})$ and the error made across the entire interval is $O(h^p)$.

What's nice about this is that if we want our numerical algorithm to be of a given order, then there exists a Taylor method with that order.

The problem with Taylor methods

The main problem with Taylor methods is that we need to have the derivatives of $f(x, y)$. Assuming that they exist, differentiation grows more and more complicated as we take higher derivatives.

The problem with Taylor methods

The main problem with Taylor methods is that we need to have the derivatives of $f(x, y)$. Assuming that they exist, differentiation grows more and more complicated as we take higher derivatives.

$$f'(x, y) = f_x(x, y) + f_y(x, y)f(x, y)$$

The problem with Taylor methods

The main problem with Taylor methods is that we need to have the derivatives of $f(x, y)$. Assuming that they exist, differentiation grows more and more complicated as we take higher derivatives.

$$f'(x, y) = f_x(x, y) + f_y(x, y)f(x, y)$$

$$f''(x, y) = f_{xx} + 2f_{xy}f + f_y f_x + f_{yy}f^2 + f_y^2 f$$

The problem with Taylor methods

The main problem with Taylor methods is that we need to have the derivatives of $f(x, y)$. Assuming that they exist, differentiation grows more and more complicated as we take higher derivatives.

$$f'(x, y) = f_x(x, y) + f_y(x, y)f(x, y)$$

$$f''(x, y) = f_{xx} + 2f_{xy}f + f_y f_x + f_{yy}f^2 + f_y^2 f$$

$$f'''(x, y) = f_{xxx} + 3f_{xxy}f + 3f_{xyy}f^2 + f_{yyy}f^3 +$$
$$f_y(f_{xx} + 2f_{xy}f + f_{yy}f^2) + 3(f_x + f_y f)(f_{xy} + f_{yy}f) + f_y^2(f_x + f_y f)$$

Outline

Runge-Kutta methods

Runge-Kutta methods on the other hand are much easier to code into a computer since once you know the values of the parameters for a given Runge-Kutta method, the algorithm only needs to compute values of f during any given step.

Runge-Kutta methods

Runge-Kutta methods on the other hand are much easier to code into a computer since once you know the values of the parameters for a given Runge-Kutta method, the algorithm only needs to compute values of f during any given step.

Runge-Kutta methods work as follows:

Runge-Kutta methods

Runge-Kutta methods on the other hand are much easier to code into a computer since once you know the values of the parameters for a given Runge-Kutta method, the algorithm only needs to compute values of f during any given step.

Runge-Kutta methods work as follows:

Suppose that we have an ODE with

$$y'(x) = f(x, y)$$

where $y_0 = y(x_0)$ is known

Runge-Kutta methods

Runge-Kutta methods on the other hand are much easier to code into a computer since once you know the values of the parameters for a given Runge-Kutta method, the algorithm only needs to compute values of f during any given step.

Runge-Kutta methods work as follows:

Suppose that we have an ODE with

$$y'(x) = f(x, y)$$

$$\text{where } y_0 = y(x_0) \text{ is known}$$

As before, we will split the interval into subintervals of equal width h and compute approximations y_n.

How to do Runge-Kutta

To compute y_{n+1}, we compute approximations for y for some choices of x in the interval $[x_n, x_{n+1}]$. These approximations are called **stages**.

How to do Runge-Kutta

To compute y_{n+1}, we compute approximations for y for some choices of x in the interval $[x_n, x_{n+1}]$. These approximations are called **stages**.

The stages and their corresponding x values are then plugged into $f(x, y)$ to obtain the **stage derivatives**.

How to do Runge-Kutta

To compute y_{n+1}, we compute approximations for y for some choices of x in the interval $[x_n, x_{n+1}]$. These approximations are called **stages**.

The stages and their corresponding x values are then plugged into $f(x, y)$ to obtain the **stage derivatives**.

So when going from y_n to y_{n+1} we will have $\mathbf{Y_1}, \mathbf{Y_2}, ..., \mathbf{Y_s}$ as our stages and $\mathbf{f}(\mathbf{X_1}, \mathbf{Y_1}), \mathbf{f}(\mathbf{X_2}, \mathbf{Y_2}), ..., \mathbf{f}(\mathbf{X_s}, \mathbf{Y_s})$ as our stage derivatives.

Computing stages

The first stage values in a step are $Y_1 = y_n$ and $f(X_1, Y_1) = f(x_n, y_n)$

Computing stages

The first stage values in a step are $Y_1 = y_n$ and $f(X_1, Y_1) = f(x_n, y_n)$

For explicit Runge-Kutta methods, the value of a stage is found by using the values of the stage derivatives of its preceding stages.

Computing stages

The first stage values in a step are $Y_1 = y_n$ and $f(X_1, Y_1) = f(x_n, y_n)$

For explicit Runge-Kutta methods, the value of a stage is found by using the values of the stage derivatives of its preceding stages.

$$Y_2 = y_n + a_{21} h f(x_n, y_n)$$

$$Y_3 = y_n + a_{31} h f(X_1, Y_1) + a_{32} h f(X_2, Y_2)$$

$$\vdots$$

$$Y_s = y_n + \sum_{j=1}^{s-1} a_{sj} h f(X_j, Y_j)$$

Computing y_{n+1}

With the stages and stage derivatives now known, we obtain y_{n+1} as y_n plus a weighted sum of the stage derivatives.

Computing y_{n+1}

With the stages and stage derivatives now known, we obtain y_{n+1} as y_n plus a weighted sum of the stage derivatives.

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i f(X_i, Y_i)$$

Computing y_{n+1}

With the stages and stage derivatives now known, we obtain y_{n+1} as y_n plus a weighted sum of the stage derivatives.

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i f(X_i, Y_i)$$

$$y_{n+1} = y_n + hb_1 f(X_1, Y_1) + hb_2 f(X_2, Y_2) + \cdots + hb_s f(X_s, Y_s)$$

Computing y_{n+1}

With the stages and stage derivatives now known, we obtain y_{n+1} as y_n plus a weighted sum of the stage derivatives.

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i f(X_i, Y_i)$$

$$y_{n+1} = y_n + hb_1 f(X_1, Y_1) + hb_2 f(X_2, Y_2) + \cdots + hb_s f(X_s, Y_s)$$

The b_i terms are known as the **weights**.

X values

Recalling that

$$h = x_{n+1} - x_n$$

X values

Recalling that

$$h = x_{n+1} - x_n$$

To be in the interval $[x_n, x_{n+1}]$, the X_i values must be of form

$$X_i = x_n + c_i h$$

where the c_i are in $[0, 1]$.

X values

Recalling that

$$h = x_{n+1} - x_n$$

To be in the interval $[x_n, x_{n+1}]$, the X_i values must be of form

$$X_i = x_n + c_i h$$

where the c_i are in $[0, 1]$.

Since $X_1 = x_n$, we have $c_1 = 0$

Putting it all together

A convenient way to represent a Runge-Kutta method is by using a tableau of form

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{s(s-1)} & \\
\hline
 & b_1 & b_2 & \dots & b_s &
\end{array}
$$

Putting it all together

A convenient way to represent a Runge-Kutta method is by using a tableau of form

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{s(s-1)} & \\
\hline
 & b_1 & b_2 & \dots & b_s &
\end{array}
$$

Where

- a_{ij} represents how stage i depends on the jth stage derivative,
- b_i represents the weight of $hf(X_i, Y_i)$ in computing y_{n+1},
- c_i represents the location of X_i in the interval $[x_n, x_{n+1}]$

Outline

Deriving Runge-Kutta methods

Now suppose that we want a Runge-Kutta method that has a global error of $O(h^p)$. How do we choose the values of the a_{ij}, the b_i, and the c_i?

Deriving Runge-Kutta methods

Now suppose that we want a Runge-Kutta method that has a global error of $O(h^p)$. How do we choose the values of the a_{ij}, the b_i, and the c_i?

This is where the Taylor methods come into play.

Deriving Runge-Kutta methods

Now suppose that we want a Runge-Kutta method that has a global error of $O(h^p)$. How do we choose the values of the a_{ij}, the b_i, and the c_i?

This is where the Taylor methods come into play.

First we choose the number of stages to use in the algorithm. At least $s = p$ stages are necessary and there are advantages to using $s = p + 1$.

Deriving Runge-Kutta methods

Now suppose that we want a Runge-Kutta method that has a global error of $O(h^p)$. How do we choose the values of the a_{ij}, the b_i, and the c_i?

This is where the Taylor methods come into play.

First we choose the number of stages to use in the algorithm. At least $s = p$ stages are necessary and there are advantages to using $s = p + 1$.

Then we compute the Taylor series expansions for the Taylor method of order p and the Runge-Kutta method with s stages. Equating coefficients leads us to a system of equations which when solved, tells us the values we can use for the Runge-Kutta method's parameters.

A necessary tool for deriving Runge-Kutta methods

In order to compare the Taylor series expansions we need the following general result:

Theorem

Bivariate Taylor Expansion: *Let $f(x, y)$ be an infinitely differentiable function in some open neighbourhood around (x_0, y_0), then*

$$f(x, y) = f(x_0, y_0) + f_x(x - x_0) + f_y(y - y_0)$$

A necessary tool for deriving Runge-Kutta methods

In order to compare the Taylor series expansions we need the following general result:

Theorem

Bivariate Taylor Expansion: *Let $f(x,y)$ be an infinitely differentiable function in some open neighbourhood around (x_0, y_0), then*

$$f(x,y) = f(x_0,y_0) + f_x(x-x_0) + f_y(y-y_0)$$

$$+\frac{1}{2!}(f_{xx}(x-x_0)^2 + f_{xy}(x-x_0)(y-y_0)$$

$$+f_{yx}(y-y_0)(x-x_0) + f_{yy}(y-y_0)^2)$$

$$+\ldots$$

2nd order Runge-Kutta methods with 2 stages

These methods have X values

$$X_1 = x_n, X_2 = x_n + c_2 h,$$

2nd order Runge-Kutta methods with 2 stages

These methods have X values

$$X_1 = x_n, X_2 = x_n + c_2 h,$$

stages

$$Y_1 = y_n, Y_2 = y_n + a_{21} h f(x_n, y_n),$$

2nd order Runge-Kutta methods with 2 stages

These methods have X values

$$X_1 = x_n, X_2 = x_n + c_2 h,$$

stages

$$Y_1 = y_n, Y_2 = y_n + a_{21} h f(x_n, y_n),$$

and stage derivatives

$$f(X_1, Y_1) = f(x_n, y_n),$$

$$f(X_2, Y_2) = f(x_n + c_2 h, y_n + a_{21} h f(x_n, y_n)).$$

Obtaining the Taylor expansions

We thus have

$$y_{n+1} = y_n + hb_1 f(x_n, y_n) + hb_2 f(x_n + c_2 h, y_n + a_{21} hf(x_n, y_n))$$

Obtaining the Taylor expansions

We thus have

$$y_{n+1} = y_n + hb_1 f(x_n, y_n) + hb_2 f(x_n + c_2 h, y_n + a_{21} hf(x_n, y_n))$$

with expansion

$$y(x_{n+1}) = y_n + b_1 hf + b_2 hf + b_2 c_2 h^2 f_x + b_2 a_{21} h^2 f_y f + O(h^3)$$

Obtaining the Taylor expansions

We thus have

$$y_{n+1} = y_n + hb_1 f(x_n, y_n) + hb_2 f(x_n + c_2 h, y_n + a_{21} hf(x_n, y_n))$$

with expansion

$$y(x_{n+1}) = y_n + b_1 hf + b_2 hf + b_2 c_2 h^2 f_x + b_2 a_{21} h^2 f_y f + O(h^3)$$

The Taylor method of order 2 is

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!}$$

Obtaining the Taylor expansions

We thus have

$$y_{n+1} = y_n + hb_1 f(x_n, y_n) + hb_2 f(x_n + c_2 h, y_n + a_{21} hf(x_n, y_n))$$

with expansion

$$y(x_{n+1}) = y_n + b_1 hf + b_2 hf + b_2 c_2 h^2 f_x + b_2 a_{21} h^2 f_y f + O(h^3)$$

The Taylor method of order 2 is

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{f'(x_n, y_n)h^2}{2!}$$

with simplified expansion

$$y(x_{n+1}) = y_n + hf + \frac{h^2 f_x}{2} + \frac{h^2 f_y f}{2} + O(h^3)$$

Comparing the Taylor expansions

$$y(x_{n+1}) = y_n + b_1 hf + b_2 hf + b_2 c_2 h^2 f_x + b_2 a_{21} h^2 f_y f + O(h^3)$$

$$y(x_{n+1}) = y_n + hf + \frac{h^2 f_x}{2} + \frac{h^2 f_y f}{2} + O(h^3)$$

Comparing the Taylor expansions

$$y(x_{n+1}) = y_n + b_1hf + b_2hf + b_2c_2h^2f_x + b_2a_{21}h^2f_yf + O(h^3)$$

$$y(x_{n+1}) = y_n + hf + \frac{h^2f_x}{2} + \frac{h^2f_yf}{2} + O(h^3)$$

Equating coefficients gives us the system of equations

$$b_1 + b_2 = 1$$

$$b_2c_2 = \frac{1}{2}$$

$$b_2a_{21} = \frac{1}{2}$$

Obtaining the parameter values

Allowing b_2 to be a free parameter gives us the solution

$$b_1 = 1 - b_2$$

$$c_2 = \frac{1}{2b_2}$$

$$a_{21} = \frac{1}{2b_2}$$

Obtaining the parameter values

Allowing b_2 to be a free parameter gives us the solution

$$b_1 = 1 - b_2$$

$$c_2 = \frac{1}{2b_2}$$

$$a_{21} = \frac{1}{2b_2}$$

So all 2nd order Runge-Kutta methods with two stages are of form

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2b_2} & \frac{1}{2b_2} & \\ \hline & 1 - b_2 & b_2 \end{array}$$

Special cases of 2nd order Runge-Kutta methods

$$
\begin{array}{c|cc}
0 & & \\
\frac{1}{2b_2} & \frac{1}{2b_2} & \\
\hline
 & 1-b_2 & b_2
\end{array}
$$

Special cases of 2nd order Runge-Kutta methods

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2b_2} & \frac{1}{2b_2} & \\ \hline & 1-b_2 & b_2 \end{array}$$

Setting $b_2 = 1$ gives us the **midpoint method**

$$y_{n+1} = y_n + hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)\right)$$

Special cases of 2nd order Runge-Kutta methods

$$
\begin{array}{c|cc}
0 & & \\
\frac{1}{2b_2} & \frac{1}{2b_2} & \\
\hline
 & 1-b_2 & b_2
\end{array}
$$

Setting $b_2 = 1$ gives us the **midpoint method**

$$y_{n+1} = y_n + hf(x_n + \frac{h}{2}, y_n + \frac{h}{2}f(x_n, y_n))$$

and setting $b_2 = \frac{1}{2}$ gives us **Heun's method**.

$$y_{n+1} = y_n + \frac{h}{2}f(x_n, y_n) + \frac{h}{2}f(x_n + h, y_n + hf(x_n, y_n))$$

The RK4

One of the most popular Runge-Kutta methods used in practice is the RK4. The RK4 is a fourth order method with 4 stage values. It achieves a nice balance of being precise and not requiring a lot of stage value computations.

The RK4

One of the most popular Runge-Kutta methods used in practice is the RK4. The RK4 is a fourth order method with 4 stage values. It achieves a nice balance of being precise and not requiring a lot of stage value computations.

The RK4 is as follows

The RK4

One of the most popular Runge-Kutta methods used in practice is the RK4. The RK4 is a fourth order method with 4 stage values. It achieves a nice balance of being precise and not requiring a lot of stage value computations.

The RK4 is as follows

$$f(X_1, Y_1) = f(x_n, y_n)$$

$$f(X_2, Y_2) = f(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(X_1, Y_1))$$
$$f(X_3, Y_3) = f(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(X_2, Y_2))$$
$$f(X_4, Y_4) = f(x_n + h, y_n + hf(X_3, Y_3))$$

The RK4 (continued)

with weights

$$y_{n+1} = y_n + \frac{h}{6} f(X_1, Y_1) + \frac{h}{3} f(X_2, Y_2) + \frac{h}{3} f(X_3, Y_3) + \frac{h}{6} f(X_4, Y_4)$$

The RK4 (continued)

with weights

$$y_{n+1} = y_n + \frac{h}{6} f(X_1, Y_1) + \frac{h}{3} f(X_2, Y_2) + \frac{h}{3} f(X_3, Y_3) + \frac{h}{6} f(X_4, Y_4)$$

and tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

The problem with higher order Runge-Kutta methods

In practice, higher order Runge-Kutta methods may not be as attractive as the RK4 as they require more stage values and stage derivatives to be computed.

The problem with higher order Runge-Kutta methods

In practice, higher order Runge-Kutta methods may not be as attractive as the RK4 as they require more stage values and stage derivatives to be computed.

Another more interesting problem is that as the required order increases when we are deriving Runge-Kutta methods,

The problem with higher order Runge-Kutta methods

In practice, higher order Runge-Kutta methods may not be as attractive as the RK4 as they require more stage values and stage derivatives to be computed.

Another more interesting problem is that as the required order increases when we are deriving Runge-Kutta methods,

- we need higher derivatives of $f(x, y)$ for the coefficients of the Taylor method,

The problem with higher order Runge-Kutta methods

In practice, higher order Runge-Kutta methods may not be as attractive as the RK4 as they require more stage values and stage derivatives to be computed.

Another more interesting problem is that as the required order increases when we are deriving Runge-Kutta methods,

- we need higher derivatives of $f(x, y)$ for the coefficients of the Taylor method,
- we will need more terms from the bivariate Taylor expansion,

The problem with higher order Runge-Kutta methods

In practice, higher order Runge-Kutta methods may not be as attractive as the RK4 as they require more stage values and stage derivatives to be computed.

Another more interesting problem is that as the required order increases when we are deriving Runge-Kutta methods,

- we need higher derivatives of $f(x, y)$ for the coefficients of the Taylor method,
- we will need more terms from the bivariate Taylor expansion,
- more stage derivatives are popping up in the computation of Y_i.

System of equations for 4th order methods with four stages

$$a_{21} = c_2$$

$$a_{31} + a_{32} = c_3$$

$$a_{41} + a_{42} + a_{43} = c_4$$

$$b_1 + b_2 + b_3 + b_4 = 1$$

$$b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2}$$

$$b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3}$$

$$b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4}$$

System of equations for 4th order methods with four stages (continued)

$$b_3 a_{32} c_2 + b_4 a_{42} c_2 + b_4 a_{43} c_3 = \frac{1}{6}$$

$$b_3 c_3 c_2 a_{32} + b_4 c_4 c_2 a_{42} + b_4 c_4 c_3 a_{43} = \frac{1}{8}$$

$$b_3 c_2^2 a_{32} + b_4 c_2^2 a_{42} + b_4 c_3^2 a_{43} = \frac{1}{12}$$

$$b_4 c_2 a_{32} a_{43} = \frac{1}{24}$$

System of equations for 4th order methods with four stages (continued)

$$b_3 a_{32} c_2 + b_4 a_{42} c_2 + b_4 a_{43} c_3 = \frac{1}{6}$$

$$b_3 c_3 c_2 a_{32} + b_4 c_4 c_2 a_{42} + b_4 c_4 c_3 a_{43} = \frac{1}{8}$$

$$b_3 c_2^2 a_{32} + b_4 c_2^2 a_{42} + b_4 c_3^2 a_{43} = \frac{1}{12}$$

$$b_4 c_2 a_{32} a_{43} = \frac{1}{24}$$

From the way these equations are written, we can see some patterns.

Outline

The goal of the project

If we know the system of equations that the parameters need to satisfy, then we don't need to do any Taylor expansions or take any derivatives of f.

The goal of the project

If we know the system of equations that the parameters need to satisfy, then we don't need to do any Taylor expansions or take any derivatives of f.

Solving the system tells us how to program Runge-Kutta methods into the computer.

The goal of the project

If we know the system of equations that the parameters need to satisfy, then we don't need to do any Taylor expansions or take any derivatives of f.

Solving the system tells us how to program Runge-Kutta methods into the computer.

How else can we obtain the system of equations?

The goal of the project

If we know the system of equations that the parameters need to satisfy, then we don't need to do any Taylor expansions or take any derivatives of f.

Solving the system tells us how to program Runge-Kutta methods into the computer.

How else can we obtain the system of equations?

What is the pattern?

The pattern

The first 3 equations

$$a_{21} = c_2$$

$$a_{31} + a_{32} = c_3$$

$$a_{41} + a_{42} + a_{43} = c_4$$

Correspond to summing the rows of a's in order to get the c values

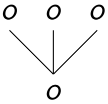
$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ c_4 & a_{41} & a_{42} & a_{43} & \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array}$$

Equations with trees

$$b_1 + b_2 + b_3 + b_4 = 1$$

$$b_1 c_1 + b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2}$$

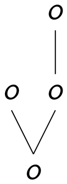
$$b_1 c_1^2 + b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3}$$



$$b_1 c_1^3 + b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4}$$

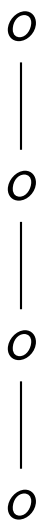
Equations with trees (continued)

$$\sum_{i,j=1}^{s} b_i a_{ij} c_j = \frac{1}{6}$$

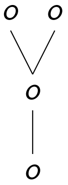


$$\sum_{i,j=1}^{s} b_i c_i a_{ij} c_j = \frac{1}{8}$$

Equations with trees (continued)



$$\sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} c_k = \frac{1}{24}$$



$$\sum_{i,j=1}^{s} b_i a_{ij} c_j^2 = \frac{1}{12}$$

Obtaining the trees

Obtaining the trees

First order

o

Obtaining the trees

First order

o

Second order

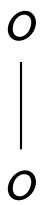
$$\begin{array}{c} o \\ | \\ o \end{array}$$

Obtaining the trees

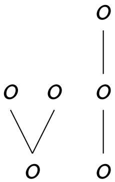
First order

o

Second order

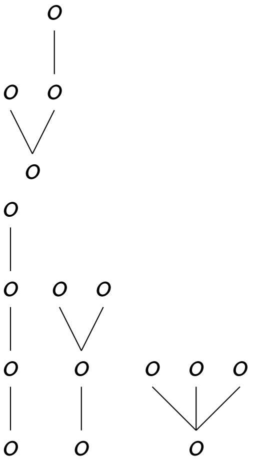


Third order



Fourth order trees

Fourth order



Conclusion

The goal of the project is to understand (with proof) how to use rooted trees to derive the system of equations for Runge-Kutta methods of a given order.

Conclusion

The goal of the project is to understand (with proof) how to use rooted trees to derive the system of equations for Runge-Kutta methods of a given order.

I may also look at the group formed by these trees (also known as the **Butcher group**).

Conclusion

The goal of the project is to understand (with proof) how to use rooted trees to derive the system of equations for Runge-Kutta methods of a given order.

I may also look at the group formed by these trees (also known as the **Butcher group**).

Thank you for viewing my presentation ☺.

References

Main reference

- Butcher, John C., *Numerical Methods for Ordinary Differential Equations* (2008), second edition

Textbook references

- Nagle et al., *Fundamentals of Differential Equations and Boundary Value Problems* (2012), sections 3.6 and 3.7, sixth edition
- Matthews, John H. and Fink, Kurtis D., *Numerical Methods using Matlab* (2004), sections 9.4 and 9.5, fourth edition

Internet references

- Kaw, Autar K., *Numerical Methods with Applications*, *http://mathforcollege.com/nm/topics/textbook_index.html*
- The Wikipedia page on Runge-Kutta methods, *http://en.wikipedia.org/wiki/Runge-Kutta*