Numerical Solutions to Differential Equations

Lecture Notes #4 — Stability Regions Revisited & Runge-Kutta Methods

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2015

Outline

Last Lecture: Quick Review

Euler's Method:

- Analysis: Local Truncation Error (LTE), Consistency, Accuracy, Stability (Region of Stability), Convergence

Improvements:

- Higher order Taylor Series Methods
- Multi-Point Methods
  - Heun's Method
  - Euler's "Midpoint Method"

Stability Regions Revisited

Recall: Euler's Method

$$y_{n+1} = y_n + hf(t_n, y_n), \quad y(t_0) = y_0$$

applied to

$$y'(t) = \lambda y(t)$$

gives

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n = (1 + h\lambda)^{n+1} y_0.$$

The stability criterion is (non-exponential growth):

$$|1 + h\lambda| \leq 1$$

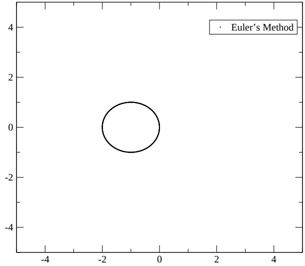
Finding the Stability Region

How do we find the stability region from the expression

$$|1 + h\lambda| \leq 1.$$

The boundary of the region is given by

$$1 + h\lambda = e^{i\theta} \quad \Leftrightarrow \quad h\lambda = e^{i\theta} - 1, \quad \theta \in [0, 2\pi)$$



Stability Regions for Higher Order Taylor Series Methods

Consider

$$y(t_{i+1}) = \sum_{k=0}^{n} \frac{h^k}{k!} y^{(k)}(t_i) + \frac{h^{n+1}}{(n+1)!} y^{(n+1)}(\xi_i), \quad \xi_i \in [t_i, t_{i+1}]$$

Now, with $y'(t) = \lambda y(t)$ we have

$$y''(t) = \lambda y'(t) = \lambda^2 y(t)$$

So

$$y^{(n)}(t) = \lambda^n y(t)$$

And it follows that

$$y(t_{i+1}) = \sum_{k=0}^{n} \frac{(h\lambda)^k}{k!} y(t_i) + \frac{(h\lambda)^{n+1}}{(n+1)!} y^{(n+1)}(\xi_i), \quad \xi_i \in [t_i, t_{i+1}]$$

Stability Regions for Higher Order Taylor Series Methods

The stability criterion is given by the relation

$$y(t_{i+1}) = y(t_0) \left[\sum_{k=0}^{n} \frac{(h\lambda)^k}{k!} \right]^{i+1}$$

i.e.

$$\left| \sum_{k=0}^{n} \frac{(h\lambda)^k}{k!} \right| \leq 1$$

Again, the boundary is given by

$$\sum_{k=0}^{n} \frac{(h\lambda)^k}{k!} = e^{i\theta}$$

Plotting the Boundary of the Stability Region

For $n = 4$ we have

$$\frac{(h\lambda)^4}{24} + \frac{(h\lambda)^3}{6} + \frac{(h\lambda)^2}{2} + (h\lambda) + 1 - e^{i\theta} = 0$$

matlab⟩⟩ `z=roots([1/24 1/6 1/2 1 1-exp(i*θ)])`

Now, vary θ in the interval $[0, 2\pi)$, collect all the roots, and plot in the complex plane (x =real(z), y =imag(z)) —

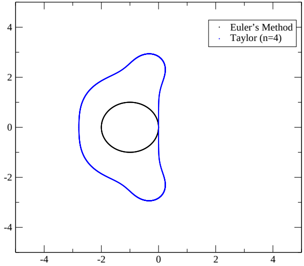


Figure: The circle corresponding to Euler's Method is included for comparison.

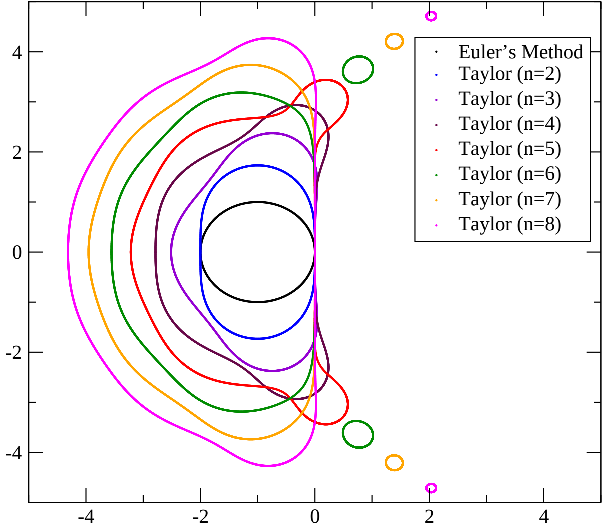
Some Comments on Higher Order Taylor Series Methods

In the cases where the derivative(s) $f^{(k)}(t, y)$ can be computed, higher order ($n > 1$) Taylor series method are superior to Euler's method (Taylor order 1) for two reasons:

1. The local truncation error is smaller $\sim \mathcal{O}(h^n)$
2. The region of stability is larger(!), allowing for (slightly) larger step-sizes h.

Next slide shows the stability regions for Taylor's Method of orders 1 through 8.

Regions of Stability for Taylor's Method ($n = 1, 2, \ldots, 8$)



Improving Euler's Method: Alternatives

When the derivative(s) of $f(t, y)$ cannot be computed — f may be a result of measurements — and/or is too expensive to compute/evaluate, we need alternative approaches to improve on Euler's Method.

We are going to explore the following approaches:

- Runge-Kutta Methods
- Linear Multistep Methods
- Predictor-Corrector Methods

There is significant overlap between these different approaches, hence we will "re-discover" some methods in several contexts.

Runge Kutta Methods

Runge-Kutta (RK) methods

- One-step methods — moving from time t_n to time t_{n+1}: Still easy to build adaptive methods if/when necessary (step-length changes on-the-fly are "easy.")
- Breaks/complicates linearity — the structure of the local error becomes more complicated.

Catch-22: Easy to change step-size since it is a one-step method, but hard(er) to tell when it is needed (local error complicated).

Linear Multistep Methods: Reverse Catch-22

When we look at Linear Multistep Methods (which use multiple points y_n, y_{n-1}, ..., y_{n-k} in order to compute y_{n+1}), we will see that they have the reverse problem: —

- for this class of methods it is easy to estimate the local error ($\Rightarrow$ easy to know when a change in step-size is necessary to maintain a certain level of local accuracy),
- but the multistep structure makes it hard to change the step-size...

A General s-stage RK method

A general s-stage RK method for the problem

$$y'(t) = f(t, y), \quad y(t_0) = y_0$$

is defined by

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

where the k_is are multiple estimates of the right-hand-side $f(t, y)$

$$k_i = f\left(t_n + c_i h,\ y_n + h \sum_{j=1}^{s} a_{i,j} k_j\right), \quad i = 1, 2, \ldots, s$$

with the following row-sum condition

$$c_i = \sum_{j=1}^{s} a_{i,j} \quad i = 1, 2, \ldots, s$$

Example: Heun's Method is an RK 2-stage Method

$$
\begin{aligned}
k_1 &= f(t_n, y_n) & &\Rightarrow c_1 = 0,\ a_{1,j} = 0 \\
k_2 &= f(t_n + h, y_n + hk_1) & &\Rightarrow c_2 = 1,\ a_{2,1} = 1,\ a_{2,2} = 0 \\
y_{n+1} &= y_n + h\left[\frac{k_1 + k_2}{2}\right] & &\Rightarrow b_1 = b_2 = \frac{1}{2}
\end{aligned}
$$

The **Butcher Array** describing Heun's Method

$$
\begin{array}{c|cc}
c_1 & a_{1,1} & a_{1,2} \\
c_2 & a_{2,1} & a_{2,2} \\
\hline
 & b_1 & b_2
\end{array}
=
\begin{array}{c|cc}
0 & 0 & 0 \\
1 & 1 & 0 \\
\hline
 & 1/2 & 1/2
\end{array}
$$

Example: Euler's Midpoint Method

$$\begin{aligned} k_1 &= f(t_n, y_n) & &\Rightarrow c_1 = 0,\ a_{1,j} = 0 \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{k_1\, h}{2}\right) & &\Rightarrow c_2 = \tfrac{1}{2},\ a_{2,1} = \tfrac{1}{2},\ a_{2,2} = 0 \\ y_{n+1} &= y_n + hk_2 & &\Rightarrow b_1 = 0,\ b_2 = 1 \end{aligned}$$

The **Butcher Array** describing Euler's Midpoint Method

$$\begin{array}{c|cc} c_1 & a_{1,1} & a_{1,2} \\ c_2 & a_{2,1} & a_{2,2} \\ \hline & b_1 & b_2 \end{array} = \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$$

The Butcher Array

The Butcher array for a general s-stage RK method is

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad = \quad \begin{array}{c|c} \tilde{\mathbf{c}} & A \\ \hline & \tilde{\mathbf{b}}^T \end{array}$$

We define the s-dimensional vectors $\tilde{\mathbf{b}}$, $\tilde{\mathbf{c}}$ and the $s \times s$-matrix A:

$$\tilde{\mathbf{b}} = [b_1, b_2, \ldots, b_s]^T, \quad \tilde{\mathbf{c}} = [c_1, c_2, \ldots, c_s]^T, \quad A = [a_{i,j}]_{i,j=1}^{s}$$

3 Types of RK-methods, I/III

- **Explicit** (*e.g.* Heun's and Midpoint):
  - If each k_j only depends on previously computed k_i $(i < j)$, then the method is **explicit**, and the matrix A is **strictly lower triangular** (*i.e.* the elements on and above the diagonal are zero).

3 Types of RK-methods, II/III

- **Semi-implicit***
  - If A is lower-triangular with non-zero entries on the diagonal, then each k_i is defined by a non-linear system:

$$k_i = f\left(t_n + c_i h, y_n + \sum_{j=1}^{i} a_{i,j} k_j \right), \quad i = 1, 2, \ldots, s$$

We have to solve s non-linear (but uncoupled) systems of equations in each iteration...

* Butcher (1965) calls these methods "Semi-implicit," Norsett (1974) "Semi-explicit," and Alexander (1977) "Diagonally Implicit RK" or DIRK methods.

3 Types of RK-methods, III/III

- **Implicit**:
  - If A is a general matrix (non-zeros above the diagonal) then each k_i is defined by a non-linear system:

$$k_i = f\left(t_n + c_i h, y_n + \sum_{j=1}^{s} a_{i,j} k_j\right), \quad i = 1, 2, \ldots, s$$

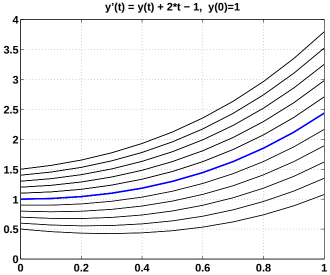
We have to solve s non-linear **coupled** systems of equations in each iteration... This can be a daunting computational task *(see Math 693a)*; most of the time we will try to avoid going this route!

A Remark on RK-methods — One Point of View I/II

RK-methods constitute a sensible idea. The unique solution to a well posed initial value ODE problem

$$y'(t) = f(t, y), \quad y(t_0) = y_0$$

is a single curve in (t, y)-space. Solutions to the same ODE with (slightly) different initial conditions form a family of solutions:



A Remark on RK-methods — One Point of View II/II

Due to numerical errors — truncation, and roundoff errors — any numerical solution "wanders off" the exact solution curve. The numerical solution is affected by neighboring solutions.

RK-methods gather information information about this "family" of solution curves.

An explicit RK-method sends out "feelers" into solution space, gathering samples of the derivative, and then decides in what direction to take the final Euler-like step.

Paraphrased from J.D. Lambert, *"Numerical Solutions for Ordinary Differential Systems: the Initial Value Problem."*

Deriving Explicit 1-stage RK-methods

The Butcher array for an 1-stage RK method has the form:

$$\begin{array}{c|c} c_1 & a_{1,1} \\ \hline & b_1 \end{array}$$

If we want an explicit scheme, then $a_{1,1} = 0$, and since $c_1 = \sum_{j=1}^{1} a_{1,s}$, we have $c_1 = 0$. Further **consistency*** requires that $\sum b_j = 1$, so $b_1 = 1$. We are left with

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

or

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ y_{n+1} &= y_n + hk_1 = y_n + hf(t_n, y_n), \quad \textbf{Euler's Method!} \end{aligned}$$

* We have yet to prove this condition.

Deriving Explicit 2-stage RK-methods, I/III

The Butcher array for a 2-stage explicit RK method has the form:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & a_{2,1} & 0 \\ \hline & b_1 & b_2 \end{array} \quad \sim \quad \begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & c_2 & 0 \\ \hline & b_1 & 1-b_1 \end{array}$$

Hence,

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + c_2 h, y_n + c_2 h k_1) \\ y_{n+1} = y_n + h\left[b_1 k_1 + (1-b_1) k_2\right] \end{cases}$$

describes all possible explicit 2-stage RK-methods.

How do we choose the parameters c_2 and b_1???
— Taylor Expansion, of course!

Deriving Explicit 2-stage RK-methods, II/III

With the following Taylor expansions:

$$\begin{aligned} y_{n+1} &= y_n + hf_n + \frac{h^2}{2} f_n' + \mathcal{O}(h^3) \\ k_1 &= f_n \\ k_2 &= f(t_n + c_2 h, y_n + c_2 h k_1) \\ &= f_n + (c_2 h) \frac{\partial}{\partial t} f(t_n, y_n) + (c_2 h f_n) \frac{\partial}{\partial y} f(t_n, y_n) + \mathcal{O}(h^2) \end{aligned}$$

We can define the Local Truncation Error

$$\begin{aligned} \mathrm{LTE}(h) &= \frac{y_{n+1} - y_n}{h} - b_1 k_1 - (1 - b_1) k_2 \\ &= \left[f_n + \frac{h}{2} f_n' + \mathcal{O}(h^2) \right] - \\ &\quad - \left[b_1 f_n - (1 - b_1) \left(f_n + (c_2 h) \left[\frac{\partial}{\partial t} f_n + \frac{\partial}{\partial y} f_n \cdot f_n \right] \right) \right] \\ &= \frac{h}{2} \left[\frac{\partial}{\partial t} f_n + \frac{\partial}{\partial y} f_n \cdot f_n \right] - \mathbf{b_2} c_2 h \left[\frac{\partial}{\partial t} f_n + \frac{\partial}{\partial y} f_n \cdot f_n \right] + \mathcal{O}(h^2) \end{aligned}$$

Deriving Explicit 2-stage RK-methods, III/III

We have

$$\text{LTE}(h) = \frac{h}{2}\left[\frac{\partial}{\partial t} f_n + \frac{\partial}{\partial y} f_n \cdot f_n\right] - \mathbf{b_2} c_2 h \left[\frac{\partial}{\partial t} f_n + \frac{\partial}{\partial y} f_n \cdot f_n\right] + \mathcal{O}(h^2)$$

Now, if

$$\frac{h}{2} - b_2 c_2 h = 0 \quad \Leftrightarrow 2 b_2 c_2 = 1$$

we get $\text{LTE}(h) \sim \mathcal{O}(h^2)$, *i.e.* our 2-stage RK-method is **second order**.

The corresponding family of Butcher arrays is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & c_2 & 0 \\ \hline & 1 - 1/(2c_2) & 1/(2c_2) \end{array}$$

Sanity check: $c_2 = 1/2$ gives Euler's Midpoint Method, and $c_2 = 1$ gives Heun's Method.

Deriving Explicit Higher Order RK-methods

We can use the same approach — Taylor expansion and parameter matching, to find higher order explicit RK-methods.

Natural question: Is this the best way of deriving the RK-methods?

Answer: There are more elegant methods for deriving the RK-methods. Most of the work was done by Butcher starting in the mid-1960s. The methods depend on defining the **Frechet derivative** and also requires some basic understanding of graph theory ("rooted trees.")

Butcher, J.C. (1987), *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, Wiley, Chichester.

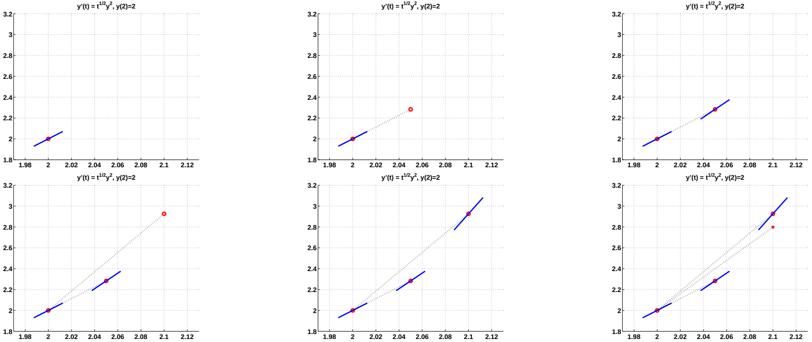
Example: 3-stage RK-method

$$\begin{array}{c|ccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline & b_1 & b_2 & b_3 \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

$$\begin{aligned} k_1 &= f\left(t_n, y_n\right) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right) \\ k_3 &= f\left(t_n + h, y_n - hk_1 + 2hk_2\right) \\ y_{n+1} &= y_n + \frac{h}{6}\left(k_1 + 4k_2 + k_3\right) \end{aligned}$$

(See next slide for visualization.)

Example: 3-stage RK-method — One Step



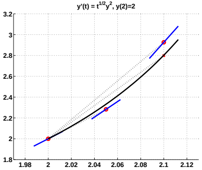


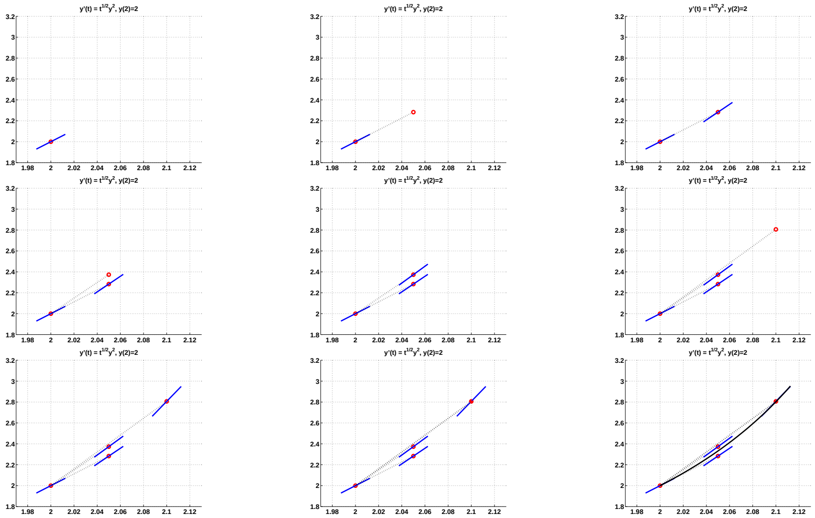
Figure: Different "stages" of one step of the 3-stage RK method.

Example: 4-stage RK-method (Attributed to Runge)

$$
\begin{array}{c|cccc}
c_1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
c_2 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
c_3 & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
c_4 & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\
\hline
 & b_1 & b_2 & b_3 & b_4
\end{array}
=
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
$$

$$
\begin{aligned}
k_1 &= f\left(t_n, y_n\right) \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{hk_2}{2}\right) \\
k_4 &= f\left(t_n + h, y_n + hk_3\right) \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Example: Runge's 4-stage method — One Step

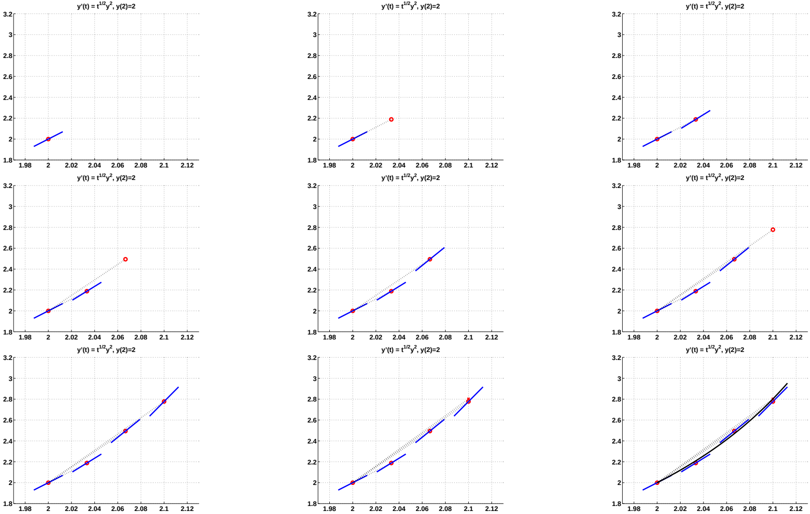


Example: 4-stage RK-method (Attributed to Kutta)

$$
\begin{array}{c|cccc}
c_1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
c_2 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
c_3 & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
c_4 & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\
\hline
 & b_1 & b_2 & b_3 & b_4
\end{array}
=
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 0 \\
2/3 & -1/3 & 1 & 0 & 0 \\
1 & 1 & -1 & 1 & 0 \\
\hline
 & 1/8 & 3/8 & 3/8 & 1/8
\end{array}
$$

$$
\begin{aligned}
k_1 &= f\left(t_n, y_n\right) \\
k_2 &= f\left(t_n + \frac{h}{3}, y_n + \frac{hk_1}{3}\right) \\
k_3 &= f\left(t_n + \frac{2h}{3}, y_n - \frac{hk_1}{3} + hk_2\right) \\
k_4 &= f(t_n + h, y_n + hk_1 - hk_2 + hk_3) \\
y_{n+1} &= y_n + \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4)
\end{aligned}
$$

Example: Kutta's 4-stage method — One Step



Next Lecture — Residual Issues

We have some important residual issues related to Runge-Kutta methods to clear up next time:

- **Consistency** condition: The requirement $\sum b_i = 1$.
- **Error estimation** (using Richardson's Extrapolation).
- **Stability Analysis**.
- Some more examples of RK-methods in action.

Future topic:

- Deriving RK-methods using rooted trees.