Runge–Kutta methods for ordinary differential equations

John Butcher

The University of Auckland
New Zealand

COE Workshop on Numerical Analysis
Kyushu University
May 2005

Contents

Contents

Contents

Contents

- Introduction to Runge–Kutta methods
- Formulation of method
- Taylor expansion of exact solution
- Taylor expansion for numerical approximation

Contents

- Introduction to Runge–Kutta methods
- Formulation of method
- Taylor expansion of exact solution
- Taylor expansion for numerical approximation
- Order conditions

Contents

Contents

- Introduction to Runge–Kutta methods
- Formulation of method
- Taylor expansion of exact solution
- Taylor expansion for numerical approximation
- Order conditions
- Construction of low order explicit methods
- Order barriers

Contents

Contents

Contents

Contents

Introduction to Runge–Kutta methods

It will be convenient to consider only autonomous initial value problems

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0, \quad f : \mathbb{R}^N \to \mathbb{R}^N.$$

Introduction to Runge–Kutta methods

It will be convenient to consider only autonomous initial value problems

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0, \quad f : \mathbb{R}^N \to \mathbb{R}^N.$$

The simple Euler method:

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

can be made more accurate by using the mid-point quadrature formula:

$$y_n = y_{n-1} + hf\Big(y_{n-1} + \tfrac{1}{2}hf(y_{n-1})\Big).$$

Introduction to Runge–Kutta methods

It will be convenient to consider only autonomous initial value problems

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0, \quad f : \mathbb{R}^N \to \mathbb{R}^N.$$

The simple Euler method:

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

can be made more accurate by using either the mid-point or the trapezoidal rule quadrature formula:

$$y_n = y_{n-1} + hf\big(y_{n-1} + \tfrac{1}{2}hf(y_{n-1})\big).$$

$$y_n = y_{n-1} + \tfrac{1}{2}hf(y_{n-1}) + \tfrac{1}{2}hf\big(y_{n-1} + hf(y_{n-1})\big).$$

These methods from Runge's 1895 paper are "second order" because the error in a single step behaves like $O(h^3)$.

These methods from Runge's 1895 paper are "second order" because the error in a single step behaves like $O(h^3)$.

A few years later, Heun gave a full explanation of order 3 methods and Kutta gave a detailed analysis of order 4 methods.

These methods from Runge's 1895 paper are "second order" because the error in a single step behaves like $O(h^3)$.

A few years later, Heun gave a full explanation of order 3 methods and Kutta gave a detailed analysis of order 4 methods.

In the early days of Runge–Kutta methods the aim seemed to be to find explicit methods of higher and higher order.

These methods from Runge's 1895 paper are "second order" because the error in a single step behaves like $O(h^3)$.

A few years later, Heun gave a full explanation of order 3 methods and Kutta gave a detailed analysis of order 4 methods.

In the early days of Runge–Kutta methods the aim seemed to be to find explicit methods of higher and higher order.

Later the aim shifted to finding methods that seemed to be optimal in terms of local truncation error and to finding built-in error estimators.

With the emergence of stiff problems as an important application area, attention moved to implicit methods.

With the emergence of stiff problems as an important application area, attention moved to implicit methods.

Methods have been found based on Gaussian quadrature.

With the emergence of stiff problems as an important application area, attention moved to implicit methods.

Methods have been found based on Gaussian quadrature.

Later this extended to methods related to Radau and Lobatto quadrature.

With the emergence of stiff problems as an important application area, attention moved to implicit methods.

Methods have been found based on Gaussian quadrature.

Later this extended to methods related to Radau and Lobatto quadrature.

A-stable methods exist in these classes.

With the emergence of stiff problems as an important application area, attention moved to implicit methods.

Methods have been found based on Gaussian quadrature.

Later this extended to methods related to Radau and Lobatto quadrature.

A-stable methods exist in these classes.

Because of the high cost of these methods, attention moved to diagonally and singly implicit methods.

Formulation of method

In carrying out a step we evaluate s stage values

$$Y_1, \quad Y_2, \quad \dots, \quad Y_s$$

and s stage derivatives

$$F_1, \quad F_2, \quad \dots, \quad F_s,$$

using the formula $F_i = f(Y_i)$.

Formulation of method

In carrying out a step we evaluate s stage values

$$Y_1, \quad Y_2, \quad \dots, \quad Y_s$$

and s stage derivatives

$$F_1, \quad F_2, \quad \dots, \quad F_s,$$

using the formula $F_i = f(Y_i)$.

Each Y_i is found as a linear combination of the F_j added on to y_0:

$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} F_j$$

Formulation of method

In carrying out a step we evaluate s stage values

$$Y_1, \quad Y_2, \quad \dots, \quad Y_s$$

and s stage derivatives

$$F_1, \quad F_2, \quad \dots, \quad F_s,$$

using the formula $F_i = f(Y_i)$.

Each Y_i is found as a linear combination of the F_j added on to y_0:

$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} F_j$$

and the approximation at $x_1 = x_0 + h$ is found from

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i F_i$$

We represent the method by a tableau:

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}
$$

We represent the method by a tableau:

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}
$$

or, if the method is explicit, by the simplified tableau

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

Examples:

$$y_1 = y_0 + \mathbf{0} hf(y_0) + \mathbf{1} hf\left(y_0 + \mathbf{\tfrac{1}{2}} hf(y_0)\right)$$

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

Examples:

$$y_1 = y_0 + \mathbf{0}hf\underbrace{(y_0)}_{Y_1} + \mathbf{1}hf\Big(\underbrace{y_0 + \tfrac{\mathbf{1}}{\mathbf{2}}hf(y_0)}_{Y_2}\Big)$$

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

Examples:

$$y_1 = y_0 + \mathbf{0}hf(y_0) + \mathbf{1}hf\big(y_0 + \tfrac{\mathbf{1}}{\mathbf{2}}hf(y_0)\big)$$

Y_1 Y_2

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

$$y_1 = y_0 + \tfrac{\mathbf{1}}{\mathbf{2}}hf(y_0) + \tfrac{\mathbf{1}}{\mathbf{2}}hf\big(y_0 + \mathbf{1}hf(y_0)\big)$$

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

Examples:

$$y_1 = y_0 + \mathbf{0}hf\underbrace{(y_0)}_{Y_1} + \mathbf{1}hf\underbrace{\left(y_0 + \tfrac{\mathbf{1}}{\mathbf{2}}hf(y_0)\right)}_{Y_2}$$

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

$$y_1 = y_0 + \tfrac{\mathbf{1}}{\mathbf{2}}hf\underbrace{(y_0)}_{Y_1} + \tfrac{\mathbf{1}}{\mathbf{2}}hf\underbrace{\left(y_0 + \mathbf{1}hf(y_0)\right)}_{Y_2}$$

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

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

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

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

$$
\begin{aligned}
y'(x) &= f(y(x)) \\
y''(x) &= f'(y(x))y'(x) \\
&= f'(y(x))f(y(x))
\end{aligned}
$$

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

$$
\begin{aligned}
y'(x) &= f(y(x)) \\
y''(x) &= f'(y(x))y'(x) \\
&= f'(y(x))f(y(x)) \\
y'''(x) &= f''(y(x))(f(y(x)), y'(x)) + f'(y(x))f'(y(x))y'(x) \\
&= f''(y(x))(f(y(x)), f(y(x))) + f'(y(x))f'(y(x))f(y(x))
\end{aligned}
$$

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

$$
\begin{aligned}
y'(x) &= f(y(x)) \\
y''(x) &= f'(y(x))y'(x) \\
&= f'(y(x))f(y(x)) \\
y'''(x) &= f''(y(x))(f(y(x)), y'(x)) + f'(y(x))f'(y(x))y'(x) \\
&= f''(y(x))(f(y(x)), f(y(x))) + f'(y(x))f'(y(x))f(y(x))
\end{aligned}
$$

This will become increasingly complicated as we evaluate higher derivatives.

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

$$
\begin{aligned}
y'(x) &= f(y(x)) \\
y''(x) &= f'(y(x))y'(x) \\
&= f'(y(x))f(y(x))) \\
y'''(x) &= f''(y(x))(f(y(x)), y'(x)) + f'(y(x))f'(y(x))y'(x) \\
&= f''(y(x))(f(y(x)), f(y(x))) + f'(y(x))f'(y(x))f(y(x))
\end{aligned}
$$

This will become increasingly complicated as we evaluate higher derivatives.

Hence we look for a systematic pattern.

Taylor expansion of exact solution

We need formulae for the second, third, . . . , derivatives.

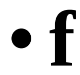
$$\begin{aligned} y'(x) &= f(y(x)) \\ y''(x) &= f'(y(x))y'(x) \\ &= f'(y(x))f(y(x))) \\ y'''(x) &= f''(y(x))(f(y(x)), y'(x)) + f'(y(x))f'(y(x))y'(x) \\ &= f''(y(x))(f(y(x)), f(y(x))) + f'(y(x))f'(y(x))f(y(x)) \end{aligned}$$

This will become increasingly complicated as we evaluate higher derivatives.

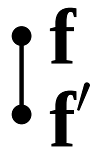
Hence we look for a systematic pattern.

Write $\mathbf{f} = f(y(x))$, $\mathbf{f}' = f'(y(x))$, $\mathbf{f}'' = f''(y(x))$, ...

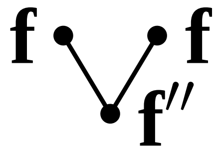
$$y'(x) = \mathbf{f}$$



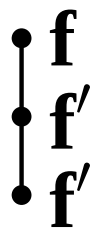
$$y''(x) = \mathbf{f}'\mathbf{f}$$



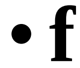
$$y'''(x) = \mathbf{f}''(\mathbf{f},\mathbf{f})$$



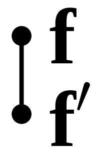
$$+\,\mathbf{f}'\mathbf{f}'\mathbf{f}$$



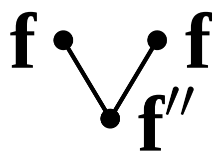
$$y'(x) = \mathbf{f}$$



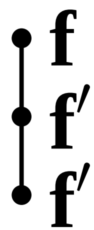
$$y''(x) = \mathbf{f}'\mathbf{f}$$



$$y'''(x) = \mathbf{f}''(\mathbf{f}, \mathbf{f})$$

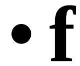


$$+ \mathbf{f}'\mathbf{f}'\mathbf{f}$$

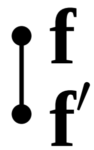


The various terms have a structure related to rooted-trees.

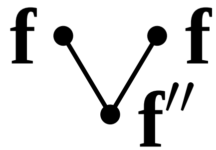
$$y'(x) = \mathbf{f}$$



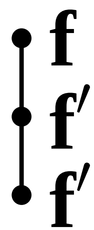
$$y''(x) = \mathbf{f}'\mathbf{f}$$



$$y'''(x) = \mathbf{f}''(\mathbf{f}, \mathbf{f})$$



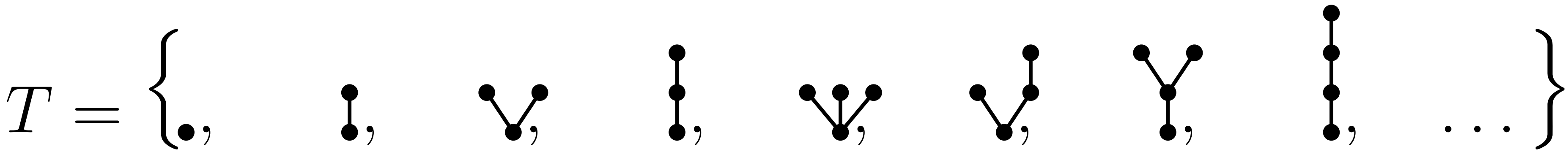
$$+\,\mathbf{f}'\mathbf{f}'\mathbf{f}$$



The various terms have a structure related to rooted-trees.

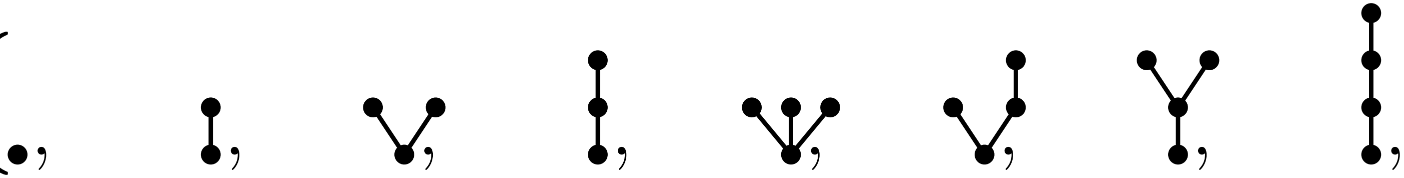
Hence, we introduce the set of all rooted trees and some functions on this set.

Let T denote the set of rooted trees:



We identify the following functions on T.

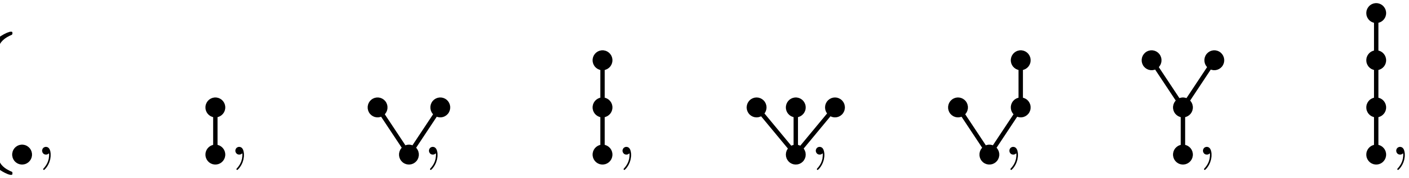
Let T denote the set of rooted trees:

$T = \Big\{$  $\ldots \Big\}$

We identify the following functions on T.

In this table, t will denote a typical tree

Let T denote the set of rooted trees:

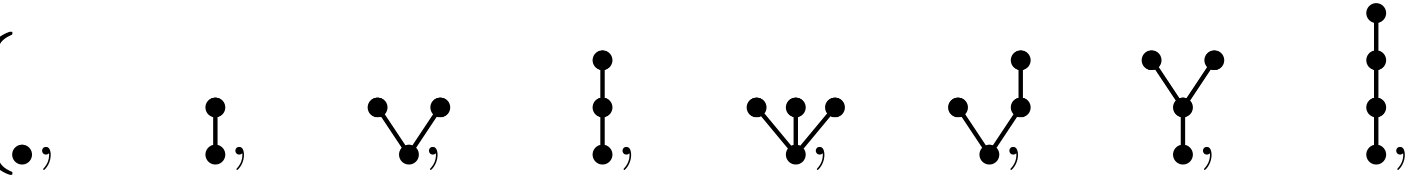
$T = \Bigg\{$  $\ldots \Bigg\}$

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices

Let T denote the set of rooted trees:

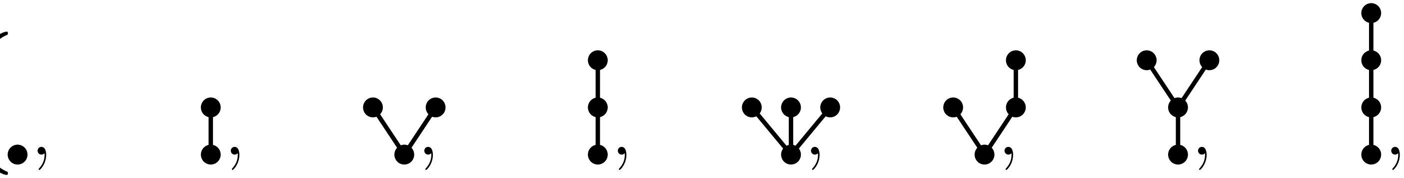
$$T = \Big\{$$  $$\ldots \Big\}$$

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group

Let T denote the set of rooted trees:

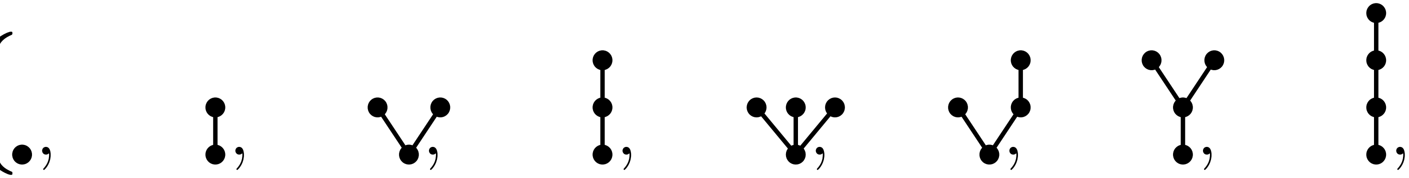
$T = \Big\{$  $\ldots \Big\}$

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group
$\gamma(t)$	density of t

Let T denote the set of rooted trees:

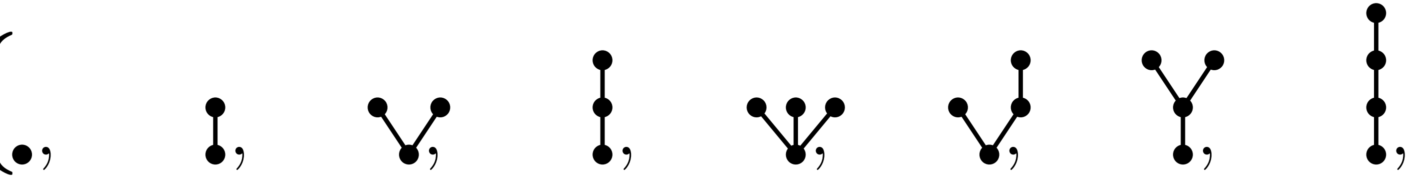
$T = \Big\{$  $\ldots \Big\}$

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group
$\gamma(t)$	density of t
$\alpha(t)$	number of ways of labelling with an ordered set

Let T denote the set of rooted trees:

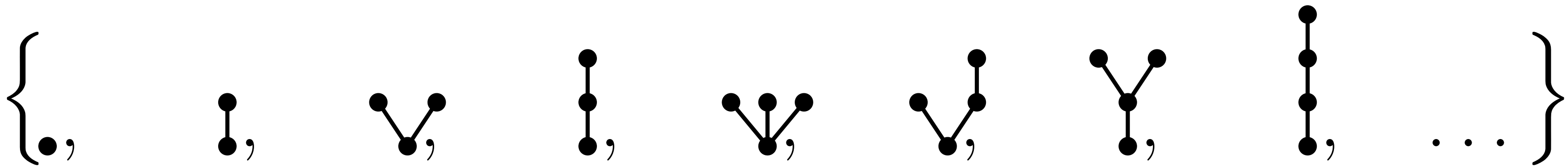
$T = \{$  $\ldots \}$

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group
$\gamma(t)$	density of t
$\alpha(t)$	number of ways of labelling with an ordered set
$\beta(t)$	number of ways of labelling with an unordered set

Let T denote the set of rooted trees:

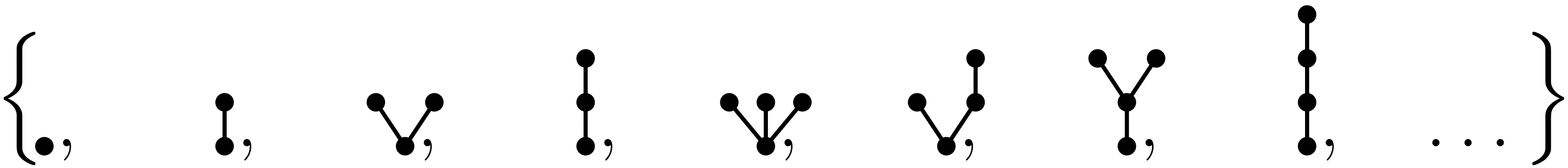
$T =$ 

We identify the following functions on T.

In this table, t will denote a typical tree

$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group
$\gamma(t)$	density of t
$\alpha(t)$	number of ways of labelling with an ordered set
$\beta(t)$	number of ways of labelling with an unordered set
$F(t)(y_0)$	elementary differential

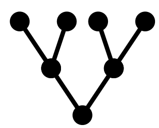
Let T denote the set of rooted trees:

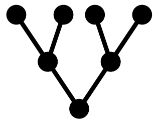
$T =$ 

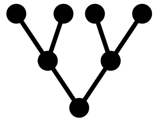
We identify the following functions on T.

In this table, t will denote a typical tree

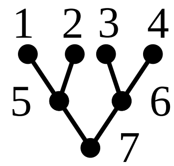
$r(t)$	order of t = number of vertices
$\sigma(t)$	symmetry of t = order of automorphism group
$\gamma(t)$	density of t
$\alpha(t)$	number of ways of labelling with an ordered set
$\beta(t)$	number of ways of labelling with an unordered set
$F(t)(y_0)$	elementary differential

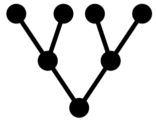
We will give examples of these functions based on $t =$ 

$t =$ 

$t =$ 

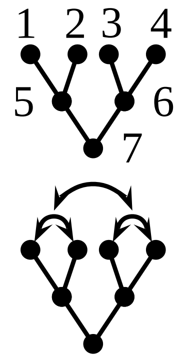
$r(t) = 7$

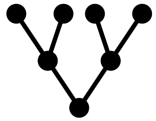


$$t =$$ 

$$r(t) = 7$$

$$\sigma(t) = 8$$

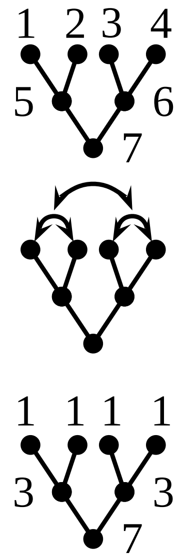


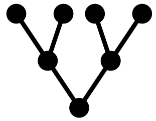
$t =$ 

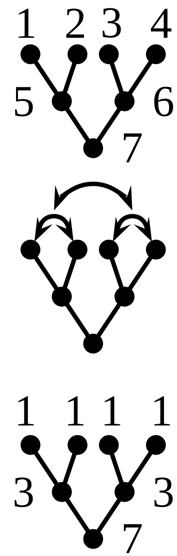
$$r(t) = 7$$

$$\sigma(t) = 8$$

$$\gamma(t) = 63$$



$t =$ 

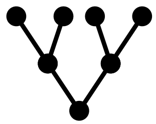


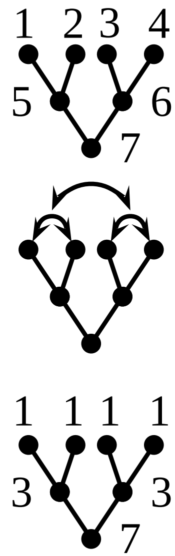
$$r(t) = 7$$

$$\sigma(t) = 8$$

$$\gamma(t) = 63$$

$$\alpha(t) = \frac{r(t)!}{\sigma(t)\gamma(t)} = 10$$

$$t = $$ 



$$r(t) = 7$$

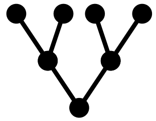
$$\sigma(t) = 8$$

$$\gamma(t) = 63$$

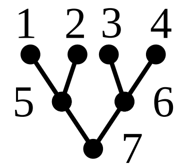
$$\alpha(t) = \frac{r(t)!}{\sigma(t)\gamma(t)} = 10$$

$$\beta(t) = \frac{r(t)!}{\sigma(t)} = 630$$

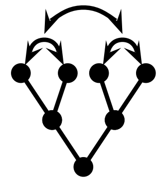
$$t =$$



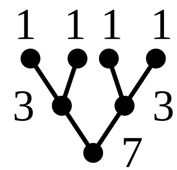
$$r(t) = 7$$



$$\sigma(t) = 8$$



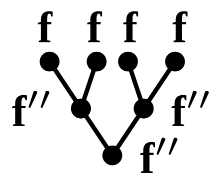
$$\gamma(t) = 63$$



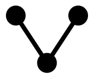
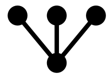
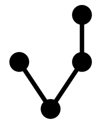
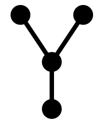
$$\alpha(t) = \frac{r(t)!}{\sigma(t)\gamma(t)} = 10$$

$$\beta(t) = \frac{r(t)!}{\sigma(t)} = 630$$

$$F(t) = \mathbf{f}''\big(\mathbf{f}''(\mathbf{f},\mathbf{f}),\mathbf{f}''(\mathbf{f},\mathbf{f})\big)$$



These functions are easy to compute up to order 4 trees:

t								
$r(t)$	1	2	3	3	4	4	4	4
$\sigma(t)$	1	1	2	1	6	1	2	1
$\gamma(t)$	1	2	3	6	4	8	12	24
$\alpha(t)$	1	1	1	1	1	3	1	1
$\beta(t)$	1	2	3	6	4	24	12	24
$F(t)$	$\mathbf{f}$	$\mathbf{f}'\mathbf{f}$	$\mathbf{f}''(\mathbf{f},\mathbf{f})$	$\mathbf{f}'\mathbf{f}'\mathbf{f}$	$\mathbf{f}^{(3)}(\mathbf{f},\mathbf{f},\mathbf{f})$	$\mathbf{f}''(\mathbf{f},\mathbf{f}'\mathbf{f})$	$\mathbf{f}'\mathbf{f}''(\mathbf{f},\mathbf{f})$	$\mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}$

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{\alpha(t) h^{r(t)}}{r(t)!} F(t)(y_0)$$

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{\alpha(t) h^{r(t)}}{r(t)!} F(t)(y_0)$$

Using the known formula for $\alpha(t)$, we can write this as

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{\alpha(t) h^{r(t)}}{r(t)!} F(t)(y_0)$$

Using the known formula for $\alpha(t)$, we can write this as

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

Our aim will now be to find a corresponding formula for the result computed by one step of a Runge-Kutta method.

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{\alpha(t) h^{r(t)}}{r(t)!} F(t)(y_0)$$

Using the known formula for $\alpha(t)$, we can write this as

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

Our aim will now be to find a corresponding formula for the result computed by one step of a Runge-Kutta method.

By comparing these formulae term by term, we will obtain conditions for a specific order of accuracy.

Taylor expansion for numerical approximation

We need to evaluate various expressions which depend on the tableau for a particular method.

Taylor expansion for numerical approximation

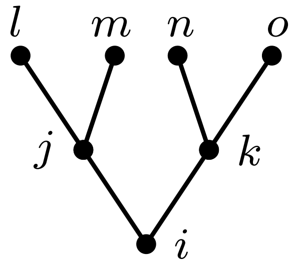
We need to evaluate various expressions which depend on the tableau for a particular method.
These are known as “elementary weights”.

Taylor expansion for numerical approximation

We need to evaluate various expressions which depend on the tableau for a particular method.

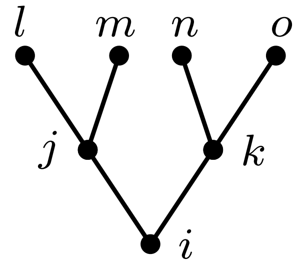
These are known as "elementary weights".

We use the example tree we have already considered to illustrate the construction of the elementary weight $\Phi(t)$.

$t =$ 

Taylor expansion for numerical approximation

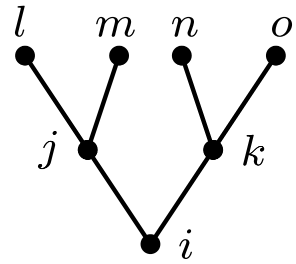
We need to evaluate various expressions which depend on the tableau for a particular method.
These are known as "elementary weights".
We use the example tree we have already considered to illustrate the construction of the elementary weight $\Phi(t)$.

$t =$ 

$$\Phi(t) = \sum_{i,j,k,l,m,n,o=1}^{s} b_i a_{ij} a_{ik} a_{jl} a_{jm} a_{kn} a_{ko}$$

Taylor expansion for numerical approximation

We need to evaluate various expressions which depend on the tableau for a particular method.
These are known as “elementary weights”.
We use the example tree we have already considered to illustrate the construction of the elementary weight $\Phi(t)$.

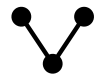
$t =$ 

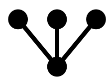
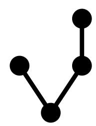
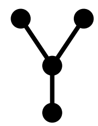
$$\Phi(t) = \sum_{i,j,k,l,m,n,o=1}^{s} b_i a_{ij} a_{ik} a_{jl} a_{jm} a_{kn} a_{ko}$$

Simplify by summing over l, m, n, o:

$$\Phi(t) = \sum_{i,j,k=1}^{s} b_i a_{ij} c_j^2 a_{ik} c_k^2$$

Now add $\Phi(t)$ to the table of functions:

t				
$r(t)$	1	2	3	3
$\alpha(t)$	1	1	1	1
$\beta(t)$	1	2	3	6
$\Phi(t)$	$\sum b_i$	$\sum b_i c_i$	$\sum b_i c_i^2$	$\sum b_i a_{ij} c_j$

t				
$r(t)$	4	4	4	4
$\alpha(t)$	1	3	1	1
$\beta(t)$	4	24	12	24
$\Phi(t)$	$\sum b_i c_i^3$	$\sum b_i c_i a_{ij} c_j$	$\sum b_i a_{ij} c_j^2$	$\sum b_i a_{ij} a_{jk} c_k$

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y_1 = y_0 + \sum_{t \in T} \frac{\beta(t) h^{r(t)}}{r(t)!} \Phi(t) F(t)(y_0)$$

The formal Taylor expansion of the solution at $x_0 + h$ is

$$y_1 = y_0 + \sum_{t \in T} \frac{\beta(t) h^{r(t)}}{r(t)!} \Phi(t) F(t)(y_0)$$

Using the known formula for $\beta(t)$, we can write this as

$$y_1 = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)} \Phi(t) F(t)(y_0)$$

Order conditions

To match the Taylor series

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

$$y_1 = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)} \Phi(t) F(t)(y_0)$$

up to h^p terms we need to ensure that

$$\Phi(t) = \frac{1}{\gamma(t)},$$

Order conditions

To match the Taylor series

$$y(x_0+h) = y_0 + \sum_{t\in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

$$y_1 = y_0 + \sum_{t\in T} \frac{h^{r(t)}}{\sigma(t)} \Phi(t) F(t)(y_0)$$

up to h^p terms we need to ensure that

$$\Phi(t) = \frac{1}{\gamma(t)},$$

for all trees such that

$$r(t) \le p.$$

Order conditions

To match the Taylor series

$$y(x_0 + h) = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)\gamma(t)} F(t)(y_0)$$

$$y_1 = y_0 + \sum_{t \in T} \frac{h^{r(t)}}{\sigma(t)} \Phi(t) F(t)(y_0)$$

up to h^p terms we need to ensure that

$$\Phi(t) = \frac{1}{\gamma(t)},$$

for all trees such that

$$r(t) \leq p.$$

These are the “order conditions”.

Construction of low order explicit methods

We will attempt to construct methods of order $p = s$ with s stages for $s = 1, 2, \ldots$.

Construction of low order explicit methods

We will attempt to construct methods of order $p = s$ with s stages for $s = 1, 2, \ldots$.
We will find that this is possible up to order 4 but not for $p \geq 5$.

Construction of low order explicit methods

We will attempt to construct methods of order $p = s$ with s stages for $s = 1, 2, \ldots$.

We will find that this is possible up to order 4 but not for $p \geq 5$.

The usual approach will be to first choose $c_2, c_3, \ldots, c_s$ and then solve for $b_1, b_2, \ldots, b_s$.

Construction of low order explicit methods

We will attempt to construct methods of order $p = s$ with s stages for $s = 1, 2, \ldots$.

We will find that this is possible up to order 4 but not for $p \geq 5$.

The usual approach will be to first choose $c_2, c_3, \ldots, c_s$ and then solve for $b_1, b_2, \ldots, b_s$.

After this solve for those of the a_{ij} which can be found as solutions to linear equations.

Construction of low order explicit methods

We will attempt to construct methods of order $p = s$ with s stages for $s = 1, 2, \ldots$.
We will find that this is possible up to order 4 but not for $p \geq 5$.
The usual approach will be to first choose $c_2, c_3, \ldots, c_s$ and then solve for $b_1, b_2, \ldots, b_s$.
After this solve for those of the a_{ij} which can be found as solutions to linear equations.

Order 2:

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_2 &= \tfrac{1}{2} \end{aligned}$$

$$\begin{array}{c|cc} 0 & & \\ c_2 & c_2 & \\ \hline & 1 - \frac{1}{2c_2} & \frac{1}{2c_2} \end{array} \qquad \begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \qquad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Order 3:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_2 c_2 + b_3 c_3 &= \tfrac{1}{2} \\ b_2 c_2^2 + b_3 c_3^2 &= \tfrac{1}{3} \\ b_3 a_{32} c_2 &= \tfrac{1}{6} \end{aligned}$$

Order 3:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_2c_2 + b_3c_3 &= \tfrac{1}{2} \\ b_2c_2^2 + b_3c_3^2 &= \tfrac{1}{3} \\ b_3a_{32}c_2 &= \tfrac{1}{6} \end{aligned}$$

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \qquad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{array} \qquad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ 0 & -1 & 1 & \\ \hline & 0 & \frac{3}{4} & \frac{1}{4} \end{array}$$

Order 4:

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{1}$$

$$b_2c_2 + b_3c_3 + b_4c_4 = \frac{1}{2} \tag{2}$$

$$b_2c_2^2 + b_3c_3^2 + b_4c_4^2 = \frac{1}{3} \tag{3}$$

$$b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 = \frac{1}{6} \tag{4}$$

$$b_2c_2^3 + b_3c_3^3 + b_4c_4^3 = \frac{1}{4} \tag{5}$$

$$b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 = \frac{1}{8} \tag{6}$$

$$b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 = \frac{1}{12} \tag{7}$$

$$b_4a_{43}a_{32}c_2 = \frac{1}{24} \tag{8}$$

Order 4:

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{1}$$

$$b_2c_2 + b_3c_3 + b_4c_4 = \tfrac{1}{2} \tag{2}$$

$$b_2c_2^2 + b_3c_3^2 + b_4c_4^2 = \tfrac{1}{3} \tag{3}$$

$$b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 = \tfrac{1}{6} \tag{4}$$

$$b_2c_2^3 + b_3c_3^3 + b_4c_4^3 = \tfrac{1}{4} \tag{5}$$

$$b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 = \tfrac{1}{8} \tag{6}$$

$$b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 = \tfrac{1}{12} \tag{7}$$

$$b_4a_{43}a_{32}c_2 = \tfrac{1}{24} \tag{8}$$

To solve these equations, treat c_2, c_3, c_4 as parameters, and solve for b_1, b_2, b_3, b_4 from (1), (2), (3), (5).

Order 4:

$$
\begin{aligned}
b_1 + b_2 + b_3 + b_4 &= 1 && (1)\\
b_2c_2 + b_3c_3 + b_4c_4 &= \tfrac{1}{2} && (2)\\
b_2c_2^2 + b_3c_3^2 + b_4c_4^2 &= \tfrac{1}{3} && (3)\\
b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 &= \tfrac{1}{6} && (4)\\
b_2c_2^3 + b_3c_3^3 + b_4c_4^3 &= \tfrac{1}{4} && (5)\\
b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 &= \tfrac{1}{8} && (6)\\
b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 &= \tfrac{1}{12} && (7)\\
b_4a_{43}a_{32}c_2 &= \tfrac{1}{24} && (8)
\end{aligned}
$$

To solve these equations, treat c_2, c_3, c_4 as parameters, and solve for b_1, b_2, b_3, b_4 from (1), (2), (3), (5).

Now solve for a_{32}, a_{42}, a_{43} from (4). (6), (7).

Order 4:

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{1}$$
$$b_2c_2 + b_3c_3 + b_4c_4 = \frac{1}{2} \tag{2}$$
$$b_2c_2^2 + b_3c_3^2 + b_4c_4^2 = \frac{1}{3} \tag{3}$$
$$b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 = \frac{1}{6} \tag{4}$$
$$b_2c_2^3 + b_3c_3^3 + b_4c_4^3 = \frac{1}{4} \tag{5}$$
$$b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 = \frac{1}{8} \tag{6}$$
$$b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 = \frac{1}{12} \tag{7}$$
$$b_4a_{43}a_{32}c_2 = \frac{1}{24} \tag{8}$$

To solve these equations, treat c_2, c_3, c_4 as parameters, and solve for b_1, b_2, b_3, b_4 from (1), (2), (3), (5).

Now solve for a_{32}, a_{42}, a_{43} from (4). (6), (7).

Use (8) to obtain consistency condition on c_2, c_3, c_4.

Order 4:

$$b_1 + b_2 + b_3 + b_4 = 1 \qquad (1)$$

$$b_2c_2 + b_3c_3 + b_4c_4 = \frac{1}{2} \qquad (2)$$

$$b_2c_2^2 + b_3c_3^2 + b_4c_4^2 = \frac{1}{3} \qquad (3)$$

$$b_3a_{32}c_2 + b_4a_{42}c_2 + b_4a_{43}c_3 = \frac{1}{6} \qquad (4)$$

$$b_2c_2^3 + b_3c_3^3 + b_4c_4^3 = \frac{1}{4} \qquad (5)$$

$$b_3c_3a_{32}c_2 + b_4c_4a_{42}c_2 + b_4c_4a_{43}c_3 = \frac{1}{8} \qquad (6)$$

$$b_3a_{32}c_2^2 + b_4a_{42}c_2^2 + b_4a_{43}c_3^2 = \frac{1}{12} \qquad (7)$$

$$b_4a_{43}a_{32}c_2 = \frac{1}{24} \qquad (8)$$

To solve these equations, treat c_2, c_3, c_4 as parameters, and solve for b_1, b_2, b_3, b_4 from (1), (2), (3), (5).

Now solve for a_{32}, a_{42}, a_{43} from (4). (6), (7).

Use (8) to obtain consistency condition on c_2, c_3, c_4.

Result: $c_4 = 1$.

We will prove a stronger result in another way.

We will prove a stronger result in another way.

Lemma 1 *Let U and V be 3×3 matrices such that*

$$UV = \begin{bmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \textit{ where } \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \textit{ is non-singular}$$

then either the last row of U is zero or the last column of V is zero.

We will prove a stronger result in another way.

Lemma 1 *Let U and V be 3×3 matrices such that*

$$UV = \begin{bmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \textit{ where } \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \textit{ is non-singular}$$

then either the last row of U is zero or the last column of V is zero.

Proof Let $W = UV$. Either U or V is singular. If U is singular, let x be a non-zero vector such that $x^T U = 0$. Therefore $x^T W = 0$. Therefore the first two components of x are zero. Hence, the last row of U is zero. The second case follows similarly if V is singular. ■

We will prove a stronger result in another way.

Lemma 1 *Let U and V be 3×3 matrices such that*

$$UV = \begin{bmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \textit{ where } \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \textit{ is non-singular}$$

then either the last row of U is zero or the last column of V is zero.

Proof Let $W = UV$. Either U or V is singular. If U is singular, let x be a non-zero vector such that $x^T U = 0$. Therefore $x^T W = 0$. Therefore the first two components of x are zero. Hence, the last row of U is zero. The second case follows similarly if V is singular. ■

We will apply this result with a specific choice of U and V.

Let

$$U = \begin{bmatrix} b_2 & b_3 & b_4 \\ b_2 c_2 & b_3 c_3 & b_4 c_4 \\ \sum_i b_i a_{i2} & \sum_i b_i a_{i3} & \sum_i b_i a_{i4} \\ -b_2(1-c_2) & -b_3(1-c_3) & -b_4(1-c_4) \end{bmatrix}$$

Let

$$U = \begin{bmatrix} b_2 & b_3 & b_4 \\ b_2 c_2 & b_3 c_3 & b_4 c_4 \\ \sum_i b_i a_{i2} & \sum_i b_i a_{i3} & \sum_i b_i a_{i4} \\ \quad -b_2(1-c_2) & \quad -b_3(1-c_3) & \quad -b_4(1-c_4) \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j} c_j - \frac{1}{2} c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j} c_j - \frac{1}{2} c_3^2 \\ c_4 & c_4^2 & \sum_j a_{4j} c_j - \frac{1}{2} c_4^2 \end{bmatrix}$$

Let

$$U = \begin{bmatrix} b_2 & b_3 & b_4 \\ b_2 c_2 & b_3 c_3 & b_4 c_4 \\ \sum_i b_i a_{i2} & \sum_i b_i a_{i3} & \sum_i b_i a_{i4} \\ -b_2(1-c_2) & -b_3(1-c_3) & -b_4(1-c_4) \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j}c_j - \frac{1}{2}c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j}c_j - \frac{1}{2}c_3^2 \\ c_4 & c_4^2 & \sum_j a_{4j}c_j - \frac{1}{2}c_4^2 \end{bmatrix}$$

then

$$UV = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.
The first two options are impossible because
$b_4 a_{43} a_{32} c_2 = \frac{1}{24}$.

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.
The first two options are impossible because
$b_4 a_{43} a_{32} c_2 = \frac{1}{24}$.
Hence, $c_4 = 1$ and the last row of U is zero.

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.
The first two options are impossible because
$b_4 a_{43} a_{32} c_2 = \frac{1}{24}$.
Hence, $c_4 = 1$ and the last row of U is zero.
The construction of fourth order Runge–Kutta methods now becomes straightforward.

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.
The first two options are impossible because
$b_4 a_{43} a_{32} c_2 = \frac{1}{24}$.
Hence, $c_4 = 1$ and the last row of U is zero.
The construction of fourth order Runge–Kutta methods now becomes straightforward.
Kutta classified all solutions to the fourth order conditions.

It follows that $b_4 = 0$, $c_2 = 0$ or $c_4 = 1$.
The first two options are impossible because
$b_4 a_{43} a_{32} c_2 = \frac{1}{24}$.
Hence, $c_4 = 1$ and the last row of U is zero.
The construction of fourth order Runge–Kutta methods now becomes straightforward.
Kutta classified all solutions to the fourth order conditions.
In particular we have his famous method:

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

Order barriers

We will review what is achievable up to order 8.

Order barriers

We will review what is achievable up to order 8.
In the table below, N_p is the number of order conditions to achieve this order.

Order barriers

We will review what is achievable up to order 8.
In the table below, N_p is the number of order conditions to achieve this order.
$M_s = s(s+1)/2$ is the number of free parameters to satisfy the order conditions for the required s stages.

Order barriers

We will review what is achievable up to order 8.
In the table below, N_p is the number of order conditions to achieve this order.
$M_s = s(s+1)/2$ is the number of free parameters to satisfy the order conditions for the required s stages.

p	N_p	s	M_s
1	1	1	1
2	2	2	3
3	4	3	6
4	8	4	10
5	17	6	21
6	37	7	28
7	115	9	45
8	200	11	66

We will now prove that $s = p = 5$ is impossible.

We will now prove that $s = p = 5$ is impossible.

Let $\widehat{b}_j = \sum_{i=1}^{5} b_i a_{ij}$, $j = 1, 2, 3, 4$ and let

$$U = \begin{bmatrix} \widehat{b}_2 & \widehat{b}_3 & \widehat{b}_4 \\ \widehat{b}_2 c_2 & \widehat{b}_3 c_3 & \widehat{b}_4 c_4 \\ \sum_i \widehat{b}_i a_{i2} & \sum_i \widehat{b}_i a_{i3} & \sum_i \widehat{b}_i a_{i4} \\ -\frac{1}{2}\widehat{b}_2(1-c_2) & -\frac{1}{2}\widehat{b}_3(1-c_3) & -\frac{1}{2}\widehat{b}_4(1-c_4) \end{bmatrix}$$

We will now prove that $s = p = 5$ is impossible.

Let $\widehat{b}_j = \sum_{i=1}^{5} b_i a_{ij}$, $j = 1, 2, 3, 4$ and let

$$U = \begin{bmatrix} \widehat{b}_2 & \widehat{b}_3 & \widehat{b}_4 \\ \widehat{b}_2 c_2 & \widehat{b}_3 c_3 & \widehat{b}_4 c_4 \\ \sum_i \widehat{b}_i a_{i2} & \sum_i \widehat{b}_i a_{i3} & \sum_i \widehat{b}_i a_{i4} \\ -\frac{1}{2}\widehat{b}_2(1-c_2) & -\frac{1}{2}\widehat{b}_3(1-c_3) & -\frac{1}{2}\widehat{b}_4(1-c_4) \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j}c_j - \frac{1}{2}c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j}c_j - \frac{1}{2}c_3^2 \\ c_4 & c_4^2 & \sum_j a_{4j}c_j - \frac{1}{2}c_4^2 \end{bmatrix}$$

We will now prove that $s = p = 5$ is impossible.

Let $\widehat{b}_j = \sum_{i=1}^{5} b_i a_{ij}$, $j = 1, 2, 3, 4$ and let

$$U = \begin{bmatrix} \widehat{b}_2 & \widehat{b}_3 & \widehat{b}_4 \\ \widehat{b}_2 c_2 & \widehat{b}_3 c_3 & \widehat{b}_4 c_4 \\ \sum_i \widehat{b}_i a_{i2} & \sum_i \widehat{b}_i a_{i3} & \sum_i \widehat{b}_i a_{i4} \\ -\frac{1}{2}\widehat{b}_2(1-c_2) & -\frac{1}{2}\widehat{b}_3(1-c_3) & -\frac{1}{2}\widehat{b}_4(1-c_4) \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j}c_j - \frac{1}{2}c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j}c_j - \frac{1}{2}c_3^2 \\ c_4 & c_4^2 & \sum_j a_{4j}c_j - \frac{1}{2}c_4^2 \end{bmatrix}$$

then

$$UV = \begin{bmatrix} \frac{1}{6} & \frac{1}{12} & 0 \\ \frac{1}{12} & \frac{1}{20} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Using Lemma 1, we deduce that $c_4 = 1$.

Using Lemma 1, we deduce that $c_4 = 1$. Now use the lemma again with

$$U = \begin{bmatrix} b_2(1-c_2) & b_3(1-c_3) & b_5(1-c_5) \\ b_2c_2(1-c_2) & b_3c_3(1-c_3) & b_5c_5(1-c_5) \\ \begin{array}{c}\sum_i b_ia_{i2}(1-c_2)\\ -b_2(1-c_2)^2\end{array} & \begin{array}{c}\sum_i b_ia_{i3}(1-c_3)\\ -b_3(1-c_3)^2\end{array} & \begin{array}{c}\sum_i b_ia_{i5}(1-c_5)\\ -b_5(1-c_5)^2\end{array} \end{bmatrix}$$

Using Lemma 1, we deduce that $c_4 = 1$. Now use the lemma again with

$$U = \begin{bmatrix} b_2(1-c_2) & b_3(1-c_3) & b_5(1-c_5) \\ b_2c_2(1-c_2) & b_3c_3(1-c_3) & b_5c_5(1-c_5) \\ \sum_i b_i a_{i2}(1-c_2) & \sum_i b_i a_{i3}(1-c_3) & \sum_i b_i a_{i5}(1-c_5) \\ -b_2(1-c_2)^2 & -b_3(1-c_3)^2 & -b_5(1-c_5)^2 \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j}c_j - \frac{1}{2}c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j}c_j - \frac{1}{2}c_3^2 \\ c_5 & c_5^2 & \sum_j a_{5j}c_j - \frac{1}{2}c_5^2 \end{bmatrix}$$

Using Lemma 1, we deduce that $c_4 = 1$. Now use the lemma again with

$$U = \begin{bmatrix} b_2(1-c_2) & b_3(1-c_3) & b_5(1-c_5) \\ b_2c_2(1-c_2) & b_3c_3(1-c_3) & b_5c_5(1-c_5) \\ \sum_i b_i a_{i2}(1-c_2) & \sum_i b_i a_{i3}(1-c_3) & \sum_i b_i a_{i5}(1-c_5) \\ -b_2(1-c_2)^2 & -b_3(1-c_3)^2 & -b_5(1-c_5)^2 \end{bmatrix}$$

$$V = \begin{bmatrix} c_2 & c_2^2 & \sum_j a_{2j}c_j - \frac{1}{2}c_2^2 \\ c_3 & c_3^2 & \sum_j a_{3j}c_j - \frac{1}{2}c_3^2 \\ c_5 & c_5^2 & \sum_j a_{5j}c_j - \frac{1}{2}c_5^2 \end{bmatrix}$$

then

$$UV = \begin{bmatrix} \frac{1}{6} & \frac{1}{12} & 0 \\ \frac{1}{12} & \frac{1}{20} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

It follows that $c_5 = 1$.

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$\sum b_i(1 - c_i)a_{ij}a_{jk}c_k$$

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$\sum b_i(1 - c_i)a_{ij}a_{jk}c_k = \tfrac{1}{120}$$

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$0 = \sum b_i(1 - c_i)a_{ij}a_{jk}c_k = \tfrac{1}{120}$$

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$0 = \sum b_i(1 - c_i)a_{ij}a_{jk}c_k = \tfrac{1}{120}$$

By modifying the details slightly, we can prove that $s = p > 5$ is never possible.

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$0 = \sum b_i(1 - c_i)a_{ij}a_{jk}c_k = \tfrac{1}{120}$$

By modifying the details slightly, we can prove that $s = p > 5$ is never possible.

The proof that $s = p + 1$ is impossible when $p \geq 7$ is more complicated.

It follows that $c_5 = 1$.

Since we already know that $c_4 = 1$, we obtain a contradiction from

$$0 = \sum b_i(1 - c_i)a_{ij}a_{jk}c_k = \tfrac{1}{120}$$

By modifying the details slightly, we can prove that $s = p > 5$ is never possible.

The proof that $s = p + 1$ is impossible when $p \geq 7$ is more complicated.

The proof that $s = p + 2$ is impossible when $p \geq 8$ is much more complicated.

Algebraic interpretation

We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

Algebraic interpretation

We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

This centres on the meaning of order for Runge-Kutta methods and leads to a possible generalisation to "effective order".

Algebraic interpretation

We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

This centres on the meaning of order for Runge-Kutta methods and leads to a possible generalisation to “effective order”.

Denote by G the group consisting of mappings of (rooted) trees to real numbers where the group operation is defined in the usual way

Algebraic interpretation

We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

This centres on the meaning of order for Runge-Kutta methods and leads to a possible generalisation to "effective order".

Denote by G the group consisting of mappings of (rooted) trees to real numbers where the group operation is defined in the usual way, according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

Algebraic interpretation

We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

This centres on the meaning of order for Runge-Kutta methods and leads to a possible generalisation to "effective order".

Denote by G the group consisting of mappings of (rooted) trees to real numbers where the group operation is defined in the usual way, according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

We will illustrate this operation in a table

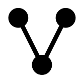
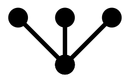
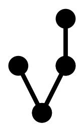
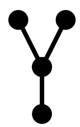
Algebraic interpretation

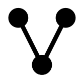
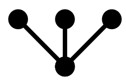
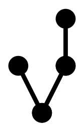
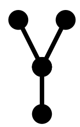
We will introduce an algebraic system which represents individual Runge-Kutta methods and also compositions of methods.

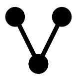
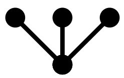
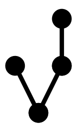
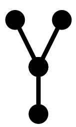
This centres on the meaning of order for Runge-Kutta methods and leads to a possible generalisation to "effective order".

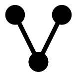
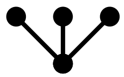
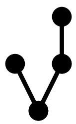
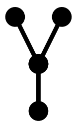
Denote by G the group consisting of mappings of (rooted) trees to real numbers where the group operation is defined in the usual way, according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

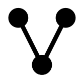
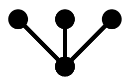
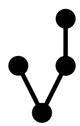
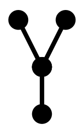
We will illustrate this operation in a table, where we also introduce the special member $E \in G$.

i	t_i
1	
2	
3	
4	
5	
6	
7	
8	

$r(t_i)$	i	t_i
1	1	
2	2	
3	3	
3	4	
4	5	
4	6	
4	7	
4	8	

$r(t_i)$	i	t_i	$\alpha(t_i)$	$\beta(t_i)$
1	1		α_1	β_1
2	2		α_2	β_2
3	3		α_3	β_3
3	4		α_4	β_4
4	5		α_5	β_5
4	6		α_6	β_6
4	7		α_7	β_7
4	8		α_8	β_8

$r(t_i)$	i	t_i	$\alpha(t_i)$	$\beta(t_i)$	$(\alpha\beta)(t_i)$
1	1		α_1	β_1	$\alpha_1 + \beta_1$
2	2		α_2	β_2	$\alpha_2 + \alpha_1\beta_1 + \beta_2$
3	3		α_3	β_3	$\alpha_3 + \alpha_1^2\beta_1 + 2\alpha_1\beta_2 + \beta_3$
3	4		α_4	β_4	$\alpha_4 + \alpha_2\beta_1 + \alpha_1\beta_2 + \beta_4$
4	5		α_5	β_5	$\alpha_5 + \alpha_1^3\beta_1 + 3\alpha_1^2\beta_2 + 3\alpha_1\beta_3 + \beta_5$
4	6		α_6	β_6	$\alpha_6 + \alpha_1\alpha_2\beta_1 + (\alpha_1^2 + \alpha_2)\beta_2$ $+ \alpha_1(\beta_3 + \beta_4) + \beta_6$
4	7		α_7	β_7	$\alpha_7 + \alpha_3\beta_1 + \alpha_1^2\beta_2 + 2\alpha_1\beta_4 + \beta_7$
4	8		α_8	β_8	$\alpha_8 + \alpha_4\beta_1 + \alpha_2\beta_2 + \alpha_1\beta_4 + \beta_8$

$r(t_i)$	i	t_i	$\alpha(t_i)$	$\beta(t_i)$	$(\alpha\beta)(t_i)$	$E(t_i)$
1	1		α_1	β_1	$\alpha_1 + \beta_1$	1
2	2		α_2	β_2	$\alpha_2 + \alpha_1\beta_1 + \beta_2$	$\frac{1}{2}$
3	3		α_3	β_3	$\alpha_3 + \alpha_1^2\beta_1 + 2\alpha_1\beta_2 + \beta_3$	$\frac{1}{3}$
3	4		α_4	β_4	$\alpha_4 + \alpha_2\beta_1 + \alpha_1\beta_2 + \beta_4$	$\frac{1}{6}$
4	5		α_5	β_5	$\alpha_5 + \alpha_1^3\beta_1 + 3\alpha_1^2\beta_2 + 3\alpha_1\beta_3 + \beta_5$	$\frac{1}{4}$
4	6		α_6	β_6	$\alpha_6 + \alpha_1\alpha_2\beta_1 + (\alpha_1^2 + \alpha_2)\beta_2$ $+ \alpha_1(\beta_3 + \beta_4) + \beta_6$	$\frac{1}{8}$
4	7		α_7	β_7	$\alpha_7 + \alpha_3\beta_1 + \alpha_1^2\beta_2 + 2\alpha_1\beta_4 + \beta_7$	$\frac{1}{12}$
4	8		α_8	β_8	$\alpha_8 + \alpha_4\beta_1 + \alpha_2\beta_2 + \alpha_1\beta_4 + \beta_8$	$\frac{1}{24}$

G_p will denote the normal subgroup defined by $t \mapsto 0$ for $r(t) \leq p$.

G_p will denote the normal subgroup defined by $t \mapsto 0$ for $r(t) \leq p$.

If $\alpha \in G$ then this maps canonically to $\alpha G_p \in G/G_p$.

G_p will denote the normal subgroup defined by $t \mapsto 0$ for $r(t) \leq p$.

If $\alpha \in G$ then this maps canonically to $\alpha G_p \in G/G_p$.

If α is defined from the elementary weights for a Runge-Kutta method then order p can be written as

$$\alpha G_p = E G_p.$$

G_p will denote the normal subgroup defined by $t \mapsto 0$ for $r(t) \leq p$.

If $\alpha \in G$ then this maps canonically to $\alpha G_p \in G/G_p$.

If α is defined from the elementary weights for a Runge-Kutta method then order p can be written as

$$\alpha G_p = E G_p.$$

Effective order p is defined by the existence of β such that

$$\beta \alpha G_p = E \beta G_p.$$

The computational interpretation of this idea is that we carry out a starting step corresponding to β

The computational interpretation of this idea is that we carry out a starting step corresponding to β and a finishing step corresponding to β^{-1}

The computational interpretation of this idea is that we carry out a starting step corresponding to β and a finishing step corresponding to β^{-1}, with many steps in between corresponding to α.

The computational interpretation of this idea is that we carry out a starting step corresponding to β and a finishing step corresponding to β^{-1}, with many steps in between corresponding to α.

This is equivalent to many steps all corresponding to $\beta\alpha\beta^{-1}$.

The computational interpretation of this idea is that we carry out a starting step corresponding to β and a finishing step corresponding to β^{-1}, with many steps in between corresponding to α.

This is equivalent to many steps all corresponding to $\beta\alpha\beta^{-1}$.

Thus, the benefits of high order can be enjoyed by high effective order.

We analyse the conditions for effective order 4.

Without loss of generality assume $\beta(t_1) = 0$.

i	$(\beta\alpha)(t_i)$	$(E\beta)(t_i)$
1	α_1	1
2	$\beta_2 + \alpha_2$	$\frac{1}{2} + \beta_2$
3	$\beta_3 + \alpha_3$	$\frac{1}{3} + 2\beta_2 + \beta_3$
4	$\beta_4 + \beta_2\alpha_1 + \alpha_4$	$\frac{1}{6} + \beta_2 + \beta_4$
5	$\beta_5 + \alpha_5$	$\frac{1}{4} + 3\beta_2 + 3\beta_3 + \beta_5$
6	$\beta_6 + \beta_2\alpha_2 + \alpha_6$	$\frac{1}{8} + \frac{3}{2}\beta_2 + \beta_3 + \beta_4 + \beta_6$
7	$\beta_7 + \beta_3\alpha_1 + \alpha_7$	$\frac{1}{12} + \beta_2 + 2\beta_4 + \beta_7$
8	$\beta_8 + \beta_4\alpha_1 + \beta_2\alpha_2 + \alpha_8$	$\frac{1}{24} + \frac{1}{2}\beta_2 + \beta_4 + \beta_8$

Of these 8 conditions, only 5 are conditions on α.

Of these 8 conditions, only 5 are conditions on α.

Once α is known, there remain 3 conditions on β.

Of these 8 conditions, only 5 are conditions on α.

Once α is known, there remain 3 conditions on β.

The 5 order conditions, written in terms of the Runge-Kutta tableau, are

$$\begin{aligned}
\sum b_i &= 1 \\
\sum b_i c_i &= \tfrac{1}{2} \\
\sum b_i a_{ij} c_j &= \tfrac{1}{6} \\
\sum b_i a_{ij} a_{jk} c_k &= \tfrac{1}{24} \\
\sum b_i c_i^2 (1 - c_i) + \sum b_i a_{ij} c_j (2c_i - c_j) &= \tfrac{1}{4}
\end{aligned}$$

Implicit Runge–Kutta methods

Since we have the order barriers, we might ask how to get around them.

Implicit Runge–Kutta methods

Since we have the order barriers, we might ask how to get around them. For explicit methods, solving the order conditions becomes increasingly difficult as the order increases

Implicit Runge–Kutta methods

Since we have the order barriers, we might ask how to get around them. For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

Implicit Runge–Kutta methods

Since we have the order barriers, we might ask how to get around them. For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{4} & \frac{1}{8} & \frac{1}{8} & & \\
\frac{7}{10} & -\frac{1}{100} & \frac{14}{25} & \frac{3}{20} & \\
1 & \frac{2}{7} & 0 & \frac{5}{7} & \\
\hline
 & \frac{1}{14} & \frac{32}{81} & \frac{250}{567} & \frac{5}{54}
\end{array}
$$

This idea can be taken further by introducing a full lower triangular A matrix.

This idea can be taken further by introducing a full lower triangular A matrix.
If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett.

This idea can be taken further by introducing a full lower triangular A matrix.

If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett.

The following third order L-stable method illustrates what is possible for DIRK methods

This idea can be taken further by introducing a full lower triangular A matrix.
If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett.
The following third order L-stable method illustrates what is possible for DIRK methods

$$\begin{array}{c|ccc}
\lambda & \lambda & & \\
\frac{1}{2}(1+\lambda) & \frac{1}{2}(1-\lambda) & \lambda & \\
1 & \frac{1}{4}(-6\lambda^2+16\lambda-1) & \frac{1}{4}(6\lambda^2-20\lambda+5) & \lambda \\
\hline
 & \frac{1}{4}(-6\lambda^2+16\lambda-1) & \frac{1}{4}(6\lambda^2-20\lambda+5) & \lambda
\end{array}$$

where $\lambda \approx 0.4358665215$ satisfies $\frac{1}{6}-\frac{3}{2}\lambda+3\lambda^2-\lambda^3 = 0$.

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$.

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$. That is A has a one-point spectrum.

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$. That is A has a one-point spectrum.

For DIRK methods the stages can be computed independently and sequentially from equations of the form

$$Y_i - h\lambda f(Y_i) = \text{a known quantity}$$

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$. That is A has a one-point spectrum.

For DIRK methods the stages can be computed independently and sequentially from equations of the form

$$Y_i - h\lambda f(Y_i) = \text{a known quantity}$$

Each stage requires the same factorised matrix $I - h\lambda\mathcal{J}$ to permit solution by a modified Newton iteration process (where $\mathcal{J} \approx \partial f/\partial y$).

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$. That is A has a one-point spectrum.

For DIRK methods the stages can be computed independently and sequentially from equations of the form

$$Y_i - h\lambda f(Y_i) = \text{a known quantity}$$

Each stage requires the same factorised matrix $I - h\lambda\mathcal{J}$ to permit solution by a modified Newton iteration process (where $\mathcal{J} \approx \partial f/\partial y$).

How then is it possible to implement SIRK methods in a similarly efficient manner?

Singly-implicit Runge-Kutta methods

A SIRK method is characterised by the equation $\sigma(A) = \{\lambda\}$. That is A has a one-point spectrum.

For DIRK methods the stages can be computed independently and sequentially from equations of the form

$$Y_i - h\lambda f(Y_i) = \text{a known quantity}$$

Each stage requires the same factorised matrix $I - h\lambda\mathcal{J}$ to permit solution by a modified Newton iteration process (where $\mathcal{J} \approx \partial f / \partial y$).

How then is it possible to implement SIRK methods in a similarly efficient manner?

The answer lies in the inclusion of a transformation to Jordan canonical form into the computation.

Suppose the matrix T transforms A to canonical form as follows

$$T^{-1}AT = \overline{A}$$

Suppose the matrix T transforms A to canonical form as follows

$$T^{-1}AT = \overline{A}$$

where

$$\overline{A} = \lambda(I - J)$$

Suppose the matrix T transforms A to canonical form as follows

$$T^{-1}AT = \overline{A}$$

where

$$\overline{A} = \lambda(I - J) = \lambda \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian $\mathcal{J}$ for each stage.

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian $\mathcal{J}$ for each stage. Assume the incoming approximation is y_0 and that we are attempting to evaluate

$$y_1 = y_0 + h(b^T \otimes I)F$$

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian $\mathcal{J}$ for each stage. Assume the incoming approximation is y_0 and that we are attempting to evaluate

$$y_1 = y_0 + h(b^T \otimes I)F$$

where F is made up from the s subvectors $F_i = f(Y_i)$, $i = 1, 2, \ldots, s$.

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian $\mathcal{J}$ for each stage. Assume the incoming approximation is y_0 and that we are attempting to evaluate

$$y_1 = y_0 + h(b^T \otimes I)F$$

where F is made up from the s subvectors $F_i = f(Y_i)$, $i = 1, 2, \ldots, s$.
The implicit equations to be solved are

$$Y = e \otimes y_0 + h(A \otimes I)F$$

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian $\mathcal{J}$ for each stage. Assume the incoming approximation is y_0 and that we are attempting to evaluate

$$y_1 = y_0 + h(b^T \otimes I)F$$

where F is made up from the s subvectors $F_i = f(Y_i)$, $i = 1, 2, \dots, s$.

The implicit equations to be solved are

$$Y = e \otimes y_0 + h(A \otimes I)F$$

where e is the vector in $\mathbb{R}^n$ with every component equal to 1 and Y has subvectors Y_i, $i = 1, 2, \dots, s$

The Newton process consists of solving the linear system

$$(I_s \otimes I - hA \otimes \mathcal{J})D = Y - e \otimes y_0 - h(A \otimes I)F$$

The Newton process consists of solving the linear system

$$(I_s \otimes I - hA \otimes \mathcal{J})D = Y - e \otimes y_0 - h(A \otimes I)F$$

and updating

$$Y \to Y - D$$

The Newton process consists of solving the linear system

$$(I_s \otimes I - hA \otimes \mathcal{J})D = Y - e \otimes y_0 - h(A \otimes I)F$$

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

$$\overline{Y} = (T^{-1} \otimes I)Y, \quad \overline{F} = (T^{-1} \otimes I)F, \quad \overline{D} = (T^{-1} \otimes I)D,$$

The Newton process consists of solving the linear system

$$(I_s \otimes I - hA \otimes \mathcal{J})D = Y - e \otimes y_0 - h(A \otimes I)F$$

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

$$\overline{Y} = (T^{-1} \otimes I)Y, \quad \overline{F} = (T^{-1} \otimes I)F, \quad \overline{D} = (T^{-1} \otimes I)D,$$

so that

$$(I_s \otimes I - h\overline{A} \otimes \mathcal{J})\overline{D} = \overline{Y} - \overline{e} \otimes y_0 - h(\overline{A} \otimes I)\overline{F}$$

The Newton process consists of solving the linear system

$$(I_s \otimes I - hA \otimes \mathcal{J})D = Y - e \otimes y_0 - h(A \otimes I)F$$

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

$$\overline{Y} = (T^{-1} \otimes I)Y, \quad \overline{F} = (T^{-1} \otimes I)F, \quad \overline{D} = (T^{-1} \otimes I)D,$$

so that

$$(I_s \otimes I - h\overline{A} \otimes \mathcal{J})\overline{D} = \overline{Y} - \overline{e} \otimes y_0 - h(\overline{A} \otimes I)\overline{F}$$

The following table summarises the costs

LU factorisation	s^3N^3	
Backsolves	s^2N^2	

	without transformation	
LU factorisation	s^3N^3	
Backsolves	s^2N^2	

	without transformation	with transformation
LU factorisation	s^3N^3	
Backsolves	s^2N^2	

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Backsolves	s^2N^2	sN^2

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation Backsolves Transformation	s^2N^2	s^2N sN^2 s^2N

In summary, we reduce the very high LU factorisation cost

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation Backsolves Transformation	s^2N^2	s^2N sN^2 s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost to the same work per stage as for DIRK or BDF

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost to the same work per stage as for DIRK or BDF methods.

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost to the same work per stage as for DIRK or BDF methods.

By comparison, the additional transformation costs are insignificant for large problems

	without transformation	with transformation
LU factorisation	s^3N^3	N^3
Transformation		s^2N
Backsolves	s^2N^2	sN^2
Transformation		s^2N

In summary, we reduce the very high LU factorisation cost to a level comparable to BDF methods.

Also we reduce the back substitution cost to the same work per stage as for DIRK or BDF methods.

By comparison, the additional transformation costs are insignificant for large problems.

Stage order s means that

$$\sum_{j=1}^{s} a_{ij}\phi(c_i) = \int_0^{c_i} \phi(t)dt,$$

Stage order s means that

$$\sum_{j=1}^{s} a_{ij}\phi(c_i) = \int_0^{c_i} \phi(t)dt,$$

for ϕ any polynomial of degree $s-1$.

Stage order s means that

$$\sum_{j=1}^{s} a_{ij}\phi(c_i) = \int_0^{c_i} \phi(t)dt,$$

for ϕ any polynomial of degree $s-1$. This implies that

$$Ac^{k-1} = \tfrac{1}{k}c^k, \qquad k = 1, 2, \ldots, s,$$

Stage order s means that

$$\sum_{j=1}^{s} a_{ij}\phi(c_i) = \int_0^{c_i} \phi(t)dt,$$

for ϕ any polynomial of degree $s-1$. This implies that

$$Ac^{k-1} = \tfrac{1}{k}c^k, \qquad k = 1, 2, \ldots, s,$$

where the vector powers are interpreted component by component.

Stage order s means that

$$\sum_{j=1}^{s} a_{ij}\phi(c_i) = \int_0^{c_i} \phi(t)dt,$$

for ϕ any polynomial of degree $s-1$. This implies that

$$Ac^{k-1} = \tfrac{1}{k}c^k, \qquad k = 1, 2, \ldots, s,$$

where the vector powers are interpreted component by component.
This is equivalent to

$$A^k c^0 = \frac{1}{k!}c^k, \qquad k = 1, 2, \ldots, s \qquad\qquad (*)$$

From the Cayley-Hamilton theorem

$$(A - \lambda I)^s c^0 = 0$$

From the Cayley-Hamilton theorem

$$(A - \lambda I)^s c^0 = 0$$

and hence

$$\sum_{i=0}^{s} \binom{s}{i} (-\lambda)^{s-i} A^i c^0 = 0.$$

From the Cayley-Hamilton theorem

$$(A - \lambda I)^s c^0 = 0$$

and hence

$$\sum_{i=0}^{s} \binom{s}{i} (-\lambda)^{s-i} A^i c^0 = 0.$$

Substitute from $(*)$ and it is found that

$$\sum_{i=0}^{s} \frac{1}{i!} \binom{s}{i} (-\lambda)^{s-i} c^i = 0.$$

Hence each component of c satisfies

$$\sum_{i=0}^{s} \frac{1}{i!} \binom{s}{i} \left(-\frac{x}{\lambda}\right)^{i} = 0$$

Hence each component of c satisfies

$$\sum_{i=0}^{s} \frac{1}{i!} \binom{s}{i} \left(-\frac{x}{\lambda}\right)^i = 0$$

That is

$$L_s\left(\frac{x}{\lambda}\right) = 0$$

where L_S denotes the Laguerre polynomial of degree s.

Hence each component of c satisfies

$$\sum_{i=0}^{s} \frac{1}{i!} \binom{s}{i} \left(-\frac{x}{\lambda}\right)^{i} = 0$$

That is

$$L_s\left(\frac{x}{\lambda}\right) = 0$$

where L_S denotes the Laguerre polynomial of degree s.

Let $\xi_1, \xi_2, \ldots, \xi_s$ denote the zeros of L_s so that

$$c_i = \lambda \xi_i, \qquad i = 1, 2, \ldots, s$$

Hence each component of c satisfies

$$\sum_{i=0}^{s} \frac{1}{i!}\binom{s}{i}\left(-\frac{x}{\lambda}\right)^{i} = 0$$

That is

$$L_s\left(\frac{x}{\lambda}\right) = 0$$

where L_S denotes the Laguerre polynomial of degree s.

Let $\xi_1, \xi_2, \ldots, \xi_s$ denote the zeros of L_s so that

$$c_i = \lambda \xi_i, \qquad i = 1, 2, \ldots, s$$

The question now is, how should λ be chosen?

Unfortunately, to obtain A-stability, at least for orders $p > 2$, λ has to be chosen so that some of the c_i are outside the interval $[0, 1]$.

Unfortunately, to obtain A-stability, at least for orders $p > 2$, λ has to be chosen so that some of the c_i are outside the interval $[0, 1]$.

This effect becomes more severe for increasingly high orders and can be seen as a major disadvantage of these methods.

Unfortunately, to obtain A-stability, at least for orders $p > 2$, λ has to be chosen so that some of the c_i are outside the interval $[0, 1]$.

This effect becomes more severe for increasingly high orders and can be seen as a major disadvantage of these methods.

We will look at two approaches for overcoming this disadvantage.

Unfortunately, to obtain A-stability, at least for orders $p > 2$, λ has to be chosen so that some of the c_i are outside the interval $[0, 1]$.

This effect becomes more severe for increasingly high orders and can be seen as a major disadvantage of these methods.

We will look at two approaches for overcoming this disadvantage.

However, we first look at the transformation matrix T for efficient implementation.

Define the matrix T as follows:

$$T = \begin{bmatrix} L_0(\xi_1) & L_1(\xi_1) & L_2(\xi_1) & \cdots & L_{s-1}(\xi_1) \\ L_0(\xi_2) & L_1(\xi_2) & L_2(\xi_2) & \cdots & L_{s-1}(\xi_2) \\ L_0(\xi_3) & L_1(\xi_3) & L_2(\xi_3) & \cdots & L_{s-1}(\xi_3) \\ \vdots & \vdots & \vdots & & \vdots \\ L_0(\xi_s) & L_1(\xi_s) & L_2(\xi_s) & \cdots & L_{s-1}(\xi_s) \end{bmatrix}$$

Define the matrix T as follows:

$$T = \begin{bmatrix} L_0(\xi_1) & L_1(\xi_1) & L_2(\xi_1) & \cdots & L_{s-1}(\xi_1) \\ L_0(\xi_2) & L_1(\xi_2) & L_2(\xi_2) & \cdots & L_{s-1}(\xi_2) \\ L_0(\xi_3) & L_1(\xi_3) & L_2(\xi_3) & \cdots & L_{s-1}(\xi_3) \\ \vdots & \vdots & \vdots & & \vdots \\ L_0(\xi_s) & L_1(\xi_s) & L_2(\xi_s) & \cdots & L_{s-1}(\xi_s) \end{bmatrix}$$

It can be shown that for a SIRK method

$$T^{-1}AT = \lambda(I - J)$$

There are two ways in which SIRK methods can be generalized
In the first of these we add extra diagonally implicit stages so that the coefficient matrix looks like this:

$$\begin{bmatrix} \widehat{A} & 0 \\ W & \lambda I \end{bmatrix},$$

where the spectrum of the $p \times p$ submatrix $\widehat{A}$ is

$$\sigma(\widehat{A}) = \{\lambda\}$$

For $s - p = 1, 2, 3, \ldots$ we get improvements to the behaviour of the methods

A second generalization is to replace “order” by “effective order”.

A second generalization is to replace “order” by “effective order”.

This allows us to locate the abscissae where we wish.

A second generalization is to replace “order” by “effective order”.

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

A second generalization is to replace “order” by “effective order”.

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

This seems to be as far as we can go in constructing efficient and accurate singly-implicit Runge-Kutta methods.