General linear methods

John Butcher

The University of Auckland

New Zealand

SANUM 2005, Stellenbosch

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

Runge-Kutta methods are examples of these methods.

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

Runge-Kutta methods are examples of these methods.

Linear multistep methods are further examples.

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

Runge-Kutta methods are examples of these methods.

Linear multistep methods are further examples.

Our aim is to understand the general class of GLMs and to search for useful methods which do not exist within the standard special cases.

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

Runge-Kutta methods are examples of these methods.

Linear multistep methods are further examples.

Our aim is to understand the general class of GLMs and to search for useful methods which do not exist within the standard special cases.

Our starting point will be the classical methods and some mild generalizations.

General linear methods

The name “general linear methods” applies to a large family of numerical methods for ordinary differential equations.

Runge-Kutta methods are examples of these methods.

Linear multistep methods are further examples.

Our aim is to understand the general class of GLMs and to search for useful methods which do not exist within the standard special cases.

Our starting point will be the classical methods and some mild generalizations.

Our finishing point will be some completely new methods.

Contents

Contents

- Generalizations of Linear Multistep Methods
  - Hybrid methods
  - Cyclic composite methods

Contents

- Generalizations of Linear Multistep Methods
  - Hybrid methods
  - Cyclic composite methods
- Generalizations of Runge-Kutta Methods

Contents

Contents

- Generalizations of Linear Multistep Methods
  - Hybrid methods
  - Cyclic composite methods
- Generalizations of Runge-Kutta Methods
  - Reuse of past values
  - Pseudo Runge-Kutta methods
  - ARK methods
  - Effective Order
- General Linear Methods

Contents

Contents

Contents

Generalizations of Linear Multistep Methods

- Linear multistep methods are inexpensive because they involve only a single function evaluation per step.

Generalizations of Linear Multistep Methods

- Linear multistep methods are inexpensive because they involve only a single function evaluation per step.
- Variable stepsize and variable order are complicated.

Generalizations of Linear Multistep Methods

- Linear multistep methods are inexpensive because they involve only a single function evaluation per step.
- Variable stepsize and variable order are complicated.
- Their performance is limited by the Dahlquist barrier.

Generalizations of Linear Multistep Methods

- Linear multistep methods are inexpensive because they involve only a single function evaluation per step.
- Variable stepsize and variable order are complicated.
- Their performance is limited by the Dahlquist barrier.
- For stiff problems where A-stability is desirable, order is limited to 2.

Generalizations of Linear Multistep Methods

- Linear multistep methods are inexpensive because they involve only a single function evaluation per step.
- Variable stepsize and variable order are complicated.
- Their performance is limited by the Dahlquist barrier.
- For stiff problems where A-stability is desirable, order is limited to 2.
- We will look at two possible generalizations which retain the general nature of linear multistep methods but overcome some of the handicaps.

Rather than methods like Adams-Bashforth

$$y_n^* = y_{n-1} + \tfrac{3}{2}hf_{n-1} - \tfrac{1}{2}hf_{n-2}$$

Hybrid methods

Rather than methods like Adams-Bashforth - Adams-Moulton

$$y_n^* = y_{n-1} + \tfrac{3}{2}hf_{n-1} - \tfrac{1}{2}hf_{n-2}$$
$$y_n = y_{n-1} + \tfrac{1}{2}hf_n^* + \tfrac{1}{2}hf_{n-1}$$

Rather than methods like Adams-Bashforth - Adams-Moulton predictor-corrector pairs:

$$y_n^* = y_{n-1} + \tfrac{3}{2}hf_{n-1} - \tfrac{1}{2}hf_{n-2}$$
$$y_n = y_{n-1} + \tfrac{1}{2}hf_n^* + \tfrac{1}{2}hf_{n-1}$$

Hybrid methods

Rather than methods like Adams-Bashforth - Adams-Moulton predictor-corrector pairs:

$$y_n^* = y_{n-1} + \tfrac{3}{2}hf_{n-1} - \tfrac{1}{2}hf_{n-2}$$

$$y_n = y_{n-1} + \tfrac{1}{2}hf_n^* + \tfrac{1}{2}hf_{n-1}$$

we can include an “off-step point” as an additional predictor:

Rather than methods like Adams-Bashforth - Adams-Moulton predictor-corrector pairs:

$$y_n^* = y_{n-1} + \tfrac{3}{2}hf_{n-1} - \tfrac{1}{2}hf_{n-2}$$
$$y_n = y_{n-1} + \tfrac{1}{2}hf_n^* + \tfrac{1}{2}hf_{n-1}$$

we can include an “off-step point” as an additional predictor:

$$y_{n-\frac{1}{2}}^* = y_{n-2} + \tfrac{9}{8}hf_{n-1} + \tfrac{3}{8}hf_{n-2}$$
$$y_n^* = \tfrac{28}{5}y_{n-1} - \tfrac{23}{5}y_{n-2} + \tfrac{32}{15}hf_{n-\frac{1}{2}}^* - 4hf_{n-1} - \tfrac{26}{15}hf_{n-2}$$
$$y_n = \tfrac{32}{31}y_{n-1} - \tfrac{1}{31}y_{n-2} + \tfrac{5}{31}hf_n^* + \tfrac{64}{93}hf_{n-\frac{1}{2}}^* + \tfrac{4}{31}hf_{n-1} - \tfrac{1}{93}hf_{n-2}$$

This particular method overcomes the (first) Dahlquist barrier and has order 5.

This particular method overcomes the (first) Dahlquist barrier and has order 5.

k-step methods like it exist up to $k = 7$ with order $2k + 1$.

This particular method overcomes the (first) Dahlquist barrier and has order 5.

k-step methods like it exist up to $k = 7$ with order $2k + 1$.

Below is a selected bibliography

Butcher J. C. (1965) A modified multistep method for the numerical integration of ordinary differential equations, *J. Assoc. Comput. Mach.*, 12: 124–135.

Gear C. W. (1965) Hybrid methods for initial value problems in ordinary differential equations, *SIAM J. Numer. Anal.*, 2: 69–86.

Gragg W. B. and Stetter H. J. (1964) Generalized multistep predictor–corrector methods, *J. Assoc. Comput. Mach.* 11: 188–209.

Cyclic composite methods

Given m linear multistep methods

$$y_n = \sum_{i=1}^{k} \alpha_i^{[j]} y_{n-i} + \sum_{i=0}^{k} \beta_i^{[j]} h f_{n-i}, \quad j = 1, \dots, m$$

apply them cyclically.

Given m linear multistep methods

$$y_n = \sum_{i=1}^{k} \alpha_i^{[j]} y_{n-i} + \sum_{i=0}^{k} \beta_i^{[j]} h f_{n-i}, \quad j = 1, \ldots, m$$

apply them cyclically.

By careful choice of the m constituent methods, many limitations of single methods can be overcome.

As a trivial example, consider the following two methods based on (open) Newton-Cotes formulae:

$$y_n = y_{n-2} + 2hf_{n-1} \qquad (*)$$

(**)

As a trivial example, consider the following two methods based on (open) Newton-Cotes formulae:

$$y_n = y_{n-2} + 2hf_{n-1} \qquad (*)$$

$$y_n = y_{n-3} + \tfrac{3}{2}hf_{n-1} + \tfrac{3}{2}hf_{n-2} \qquad (**)$$

As a trivial example, consider the following two methods based on (open) Newton-Cotes formulae:

$$y_n = y_{n-2} + 2hf_{n-1} \qquad (*)$$

$$y_n = y_{n-3} + \tfrac{3}{2}hf_{n-1} + \tfrac{3}{2}hf_{n-2} \qquad (**)$$

By itself each of these methods is weakly stable

As a trivial example, consider the following two methods based on (open) Newton-Cotes formulae:

$$y_n = y_{n-2} + 2hf_{n-1} \qquad (*)$$

$$y_n = y_{n-3} + \tfrac{3}{2}hf_{n-1} + \tfrac{3}{2}hf_{n-2} \qquad (**)$$

By itself each of these methods is weakly stable but this handicap is overcome if the pair of methods is used in alternation.

As a trivial example, consider the following two methods based on (open) Newton-Cotes formulae:

$$y_n = y_{n-2} + 2hf_{n-1} \qquad (*)$$

$$y_n = y_{n-3} + \tfrac{3}{2}hf_{n-1} + \tfrac{3}{2}hf_{n-2} \qquad (**)$$

By itself each of these methods is weakly stable but this handicap is overcome if the pair of methods is used in alternation.

That is, if n is odd then (*) is used and if n is even then (**) is used.

Cycles of explicit methods can be constructed which overcome the first Dahlquist barrier.

Cycles of explicit methods can be constructed which overcome the first Dahlquist barrier.

For example:

$$y_n = -\frac{8}{11}y_{n-1} + \frac{19}{11}y_{n-2} + \frac{10}{11}hf_n + \frac{19}{11}hf_{n-1} + \frac{8}{11}hf_{n-2} - \frac{1}{33}hf_{n-3}$$

$$y_n = \frac{449}{240}y_{n-1} + \frac{19}{30}y_{n-2} - \frac{361}{240}y_{n-3} + \frac{251}{720}hf_n + \frac{19}{30}hf_{n-1} - \frac{449}{240}hf_{n-2} - \frac{35}{72}hf_{n-3}$$

Cycles of explicit methods can be constructed which overcome the first Dahlquist barrier.

For example:

$$
\begin{aligned}
y_n &= -\tfrac{8}{11}y_{n-1} + \tfrac{19}{11}y_{n-2} \\
&\qquad + \tfrac{10}{11}hf_n + \tfrac{19}{11}hf_{n-1} + \tfrac{8}{11}hf_{n-2} - \tfrac{1}{33}hf_{n-3} \\
y_n &= \tfrac{449}{240}y_{n-1} + \tfrac{19}{30}y_{n-2} - \tfrac{361}{240}y_{n-3} \\
&\qquad + \tfrac{251}{720}hf_n + \tfrac{19}{30}hf_{n-1} - \tfrac{449}{240}hf_{n-2} - \tfrac{35}{72}hf_{n-3}
\end{aligned}
$$

Each of these methods has order 5 and each is unstable.

Cycles of explicit methods can be constructed which overcome the first Dahlquist barrier.

For example:

$$\begin{aligned} y_n &= -\frac{8}{11}y_{n-1} + \frac{19}{11}y_{n-2} \\ &\quad + \frac{10}{11}hf_n + \frac{19}{11}hf_{n-1} + \frac{8}{11}hf_{n-2} - \frac{1}{33}hf_{n-3} \\ y_n &= \frac{449}{240}y_{n-1} + \frac{19}{30}y_{n-2} - \frac{361}{240}y_{n-3} \\ &\quad + \frac{251}{720}hf_n + \frac{19}{30}hf_{n-1} - \frac{449}{240}hf_{n-2} - \frac{35}{72}hf_{n-3} \end{aligned}$$

Each of these methods has order 5 and each is unstable.

The corresponding cyclic method has perfect stability.

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11}y_{n-1} + \frac{19}{11}y_{n-2} \qquad \text{(*)}$$

$$y_n = \frac{449}{240}y_{n-1} + \frac{19}{30}y_{n-2} - \frac{361}{240}y_{n-3} \qquad \text{(**)}$$

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11} y_{n-1} + \frac{19}{11} y_{n-2} \qquad (*)$$

$$y_n = \frac{449}{240} y_{n-1} + \frac{19}{30} y_{n-2} - \frac{361}{240} y_{n-3} \qquad (**)$$

The difference equation for $y_n - y_{n-1}$ is

$$\begin{bmatrix} y_n - y_{n-1} \\ y_{n-1} - y_{n-2} \end{bmatrix} = X \begin{bmatrix} y_{n-1} - y_{n-2} \\ y_{n-2} - y_{n-3} \end{bmatrix}$$

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11} y_{n-1} + \frac{19}{11} y_{n-2} \qquad (*)$$

$$y_n = \frac{449}{240} y_{n-1} + \frac{19}{30} y_{n-2} - \frac{361}{240} y_{n-3} \qquad (**)$$

The difference equation for $y_n - y_{n-1}$ is

$$\begin{bmatrix} y_n - y_{n-1} \\ y_{n-1} - y_{n-2} \end{bmatrix} = X \begin{bmatrix} y_{n-1} - y_{n-2} \\ y_{n-2} - y_{n-3} \end{bmatrix}$$

where X is $\begin{bmatrix} -\frac{19}{11} & 0 \\ 1 & 0 \end{bmatrix}$ for (*)

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11} y_{n-1} + \frac{19}{11} y_{n-2} \qquad (*)$$

$$y_n = \frac{449}{240} y_{n-1} + \frac{19}{30} y_{n-2} - \frac{361}{240} y_{n-3} \qquad (**)$$

The difference equation for $y_n - y_{n-1}$ is

$$\begin{bmatrix} y_n - y_{n-1} \\ y_{n-1} - y_{n-2} \end{bmatrix} = X \begin{bmatrix} y_{n-1} - y_{n-2} \\ y_{n-2} - y_{n-3} \end{bmatrix}$$

where X is $\begin{bmatrix} -\frac{19}{11} & 0 \\ 1 & 0 \end{bmatrix}$ for (*) or $\begin{bmatrix} \frac{209}{240} & \frac{361}{240} \\ 1 & 0 \end{bmatrix}$ for (**).

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11} y_{n-1} + \frac{19}{11} y_{n-2} \qquad (*)$$

$$y_n = \frac{449}{240} y_{n-1} + \frac{19}{30} y_{n-2} - \frac{361}{240} y_{n-3} \qquad (**)$$

The difference equation for $y_n - y_{n-1}$ is

$$\begin{bmatrix} y_n - y_{n-1} \\ y_{n-1} - y_{n-2} \end{bmatrix} = X \begin{bmatrix} y_{n-1} - y_{n-2} \\ y_{n-2} - y_{n-3} \end{bmatrix}$$

where X is $\begin{bmatrix} -\frac{19}{11} & 0 \\ 1 & 0 \end{bmatrix}$ for (*) or $\begin{bmatrix} \frac{209}{240} & \frac{361}{240} \\ 1 & 0 \end{bmatrix}$ for (**).

Neither matrix is power-bounded

To verify these remarks, analyse stability using $y' = 0$

$$y_n = -\frac{8}{11}y_{n-1} + \frac{19}{11}y_{n-2} \qquad (*)$$

$$y_n = \frac{449}{240}y_{n-1} + \frac{19}{30}y_{n-2} - \frac{361}{240}y_{n-3} \qquad (**)$$

The difference equation for $y_n - y_{n-1}$ is

$$\begin{bmatrix} y_n - y_{n-1} \\ y_{n-1} - y_{n-2} \end{bmatrix} = X \begin{bmatrix} y_{n-1} - y_{n-2} \\ y_{n-2} - y_{n-3} \end{bmatrix}$$

where X is $\begin{bmatrix} -\frac{19}{11} & 0 \\ 1 & 0 \end{bmatrix}$ for (*) or $\begin{bmatrix} \frac{209}{240} & \frac{361}{240} \\ 1 & 0 \end{bmatrix}$ for (**).

Neither matrix is power-bounded but their product is nilpotent.

Furthermore A-stable methods of orders greater than 2 (thus breaking the second barrier), can be found.

Furthermore A-stable methods of orders greater than 2 (thus breaking the second barrier), can be found.

Below is a selected bibliography

J. Donelson, and E. Hansen (1971) ‘Cyclic composite multistep predictor-corrector methods’. *SIAM J. Numer. Anal.* **8** 137–157.

T. A. Bickart and Z. Picel (1973) ‘High order stiffly stable composite multistep methods for numerical integration of stiff differential equations’, *BIT* **13** 272–286.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.
- An implicit method can have order $p = 2s$.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.
- An implicit method can have order $p = 2s$.
- Although such methods are A-stable, they have many disadvantages.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.
- An implicit method can have order $p = 2s$.
- Although such methods are A-stable, they have many disadvantages.
- For example, they have low stage-order.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.
- An implicit method can have order $p = 2s$.
- Although such methods are A-stable, they have many disadvantages.
- For example, they have low stage-order.
- And they are very expensive to implement.

Generalizations of Runge-Kutta Methods

- Runge-Kutta methods have always been regarded as expensive because of their multistage (multiple function calls in each timestep) structure.
- For low values of the order p the number of stages s can equal p but this is impossible if $p > 4$.
- An implicit method can have order $p = 2s$.
- Although such methods are A-stable, they have many disadvantages.
- For example, they have low stage-order.
- And they are very expensive to implement.
- For both explicit and implicit RK methods, it is very difficult to estimate errors for variable h and p.

From one of Kutta's fourth order families:

$$
\begin{array}{c|cccc}
0 & & & & \\
c_2 & c_2 & & & \\
\frac{1}{2} & \frac{1}{2}-\frac{1}{8c_2} & \frac{1}{8c_2} & & \\
1 & \frac{1}{2c_2}-1 & -\frac{1}{2c_2} & 2 & \\
\hline
 & \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6}
\end{array}
$$

From one of Kutta's fourth order families:

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & c_2 & & & \\ \frac{1}{2} & \frac{1}{2}-\frac{1}{8c_2} & \frac{1}{8c_2} & & \\ 1 & \frac{1}{2c_2}-1 & -\frac{1}{2c_2} & 2 & \\ \hline & \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} \end{array}$$

If $c_2 = -1$:

$$\begin{array}{c|cccc} 0 & & & & \\ -1 & -1 & & & \\ \frac{1}{2} & \frac{5}{8} & -\frac{1}{8} & & \\ 1 & -\frac{3}{2} & \frac{1}{2} & 2 & \\ \hline & \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} \end{array}$$

We can interpret the abscissa at -1 as reuse of the derivative found as the beginning of the previous step.

We can interpret the abscissa at -1 as reuse of the derivative found as the beginning of the previous step.

We then have the method

$$Y_1 = y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), \qquad F_1 = f(Y_1)$$

$$Y_2 = y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, \quad F_2 = f(Y_2)$$

$$y_n = y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2$$

We can interpret the abscissa at -1 as reuse of the derivative found as the beginning of the previous step.

We then have the method

$$Y_1 = y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), \qquad F_1 = f(Y_1)$$

$$Y_2 = y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, \quad F_2 = f(Y_2)$$

$$y_n = y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2$$

Like the Runge-Kutta method, this retains order 4.

We can interpret the abscissa at -1 as reuse of the derivative found as the beginning of the previous step.

We then have the method

$$Y_1 = y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), \qquad F_1 = f(Y_1)$$

$$Y_2 = y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, \quad F_2 = f(Y_2)$$

$$y_n = y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2$$

Like the Runge-Kutta method, this retains order 4.

This evaluates f only 3 times per timestep compared with 4 for the original method.

We can interpret the abscissa at -1 as reuse of the derivative found as the beginning of the previous step.

We then have the method

$$Y_1 = y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), \qquad F_1 = f(Y_1)$$

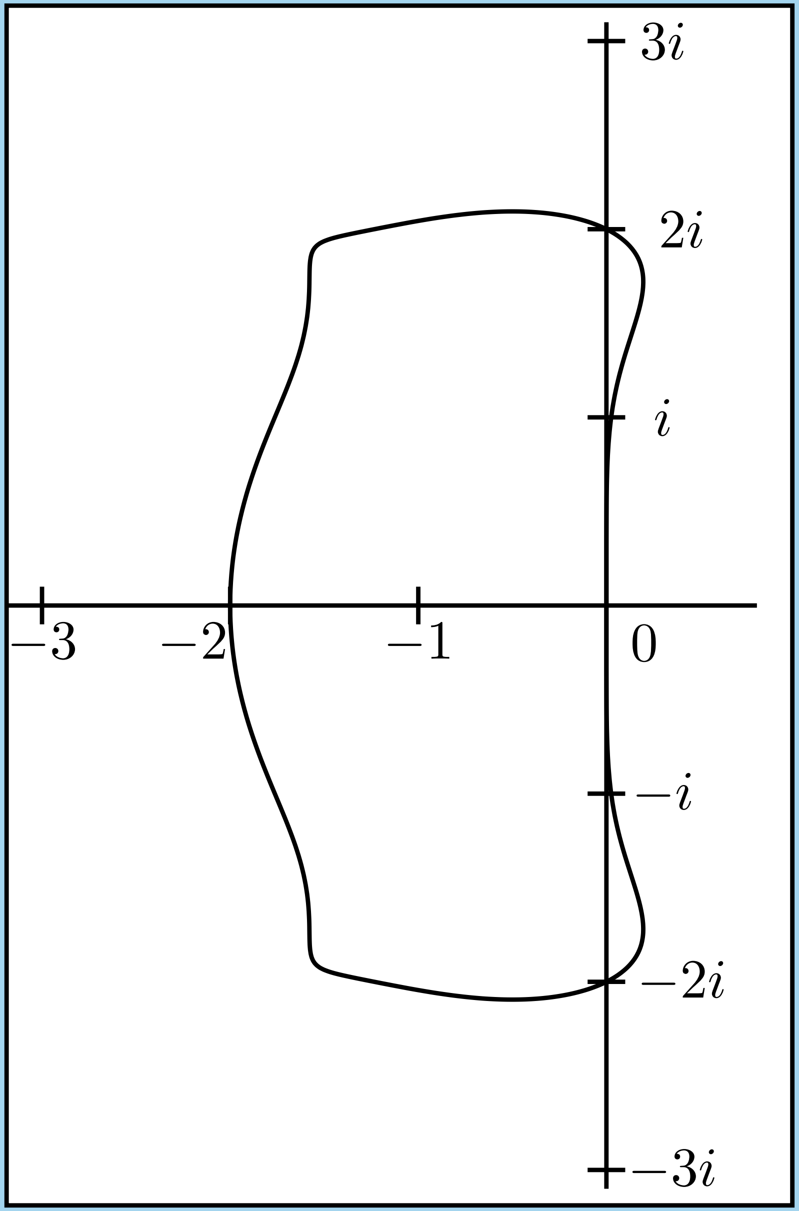
$$Y_2 = y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, \quad F_2 = f(Y_2)$$

$$y_n = y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2$$

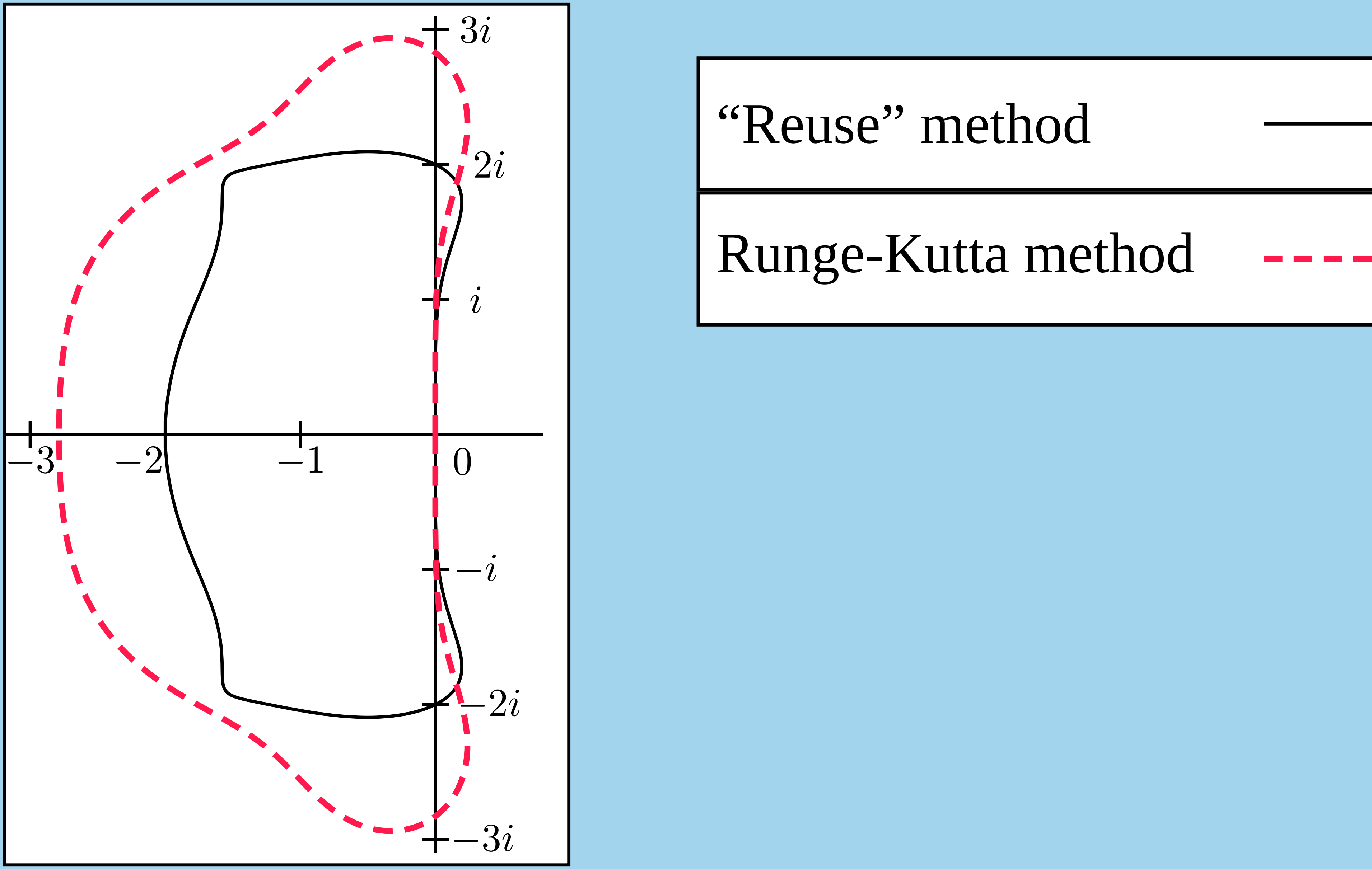
Like the Runge-Kutta method, this retains order 4.

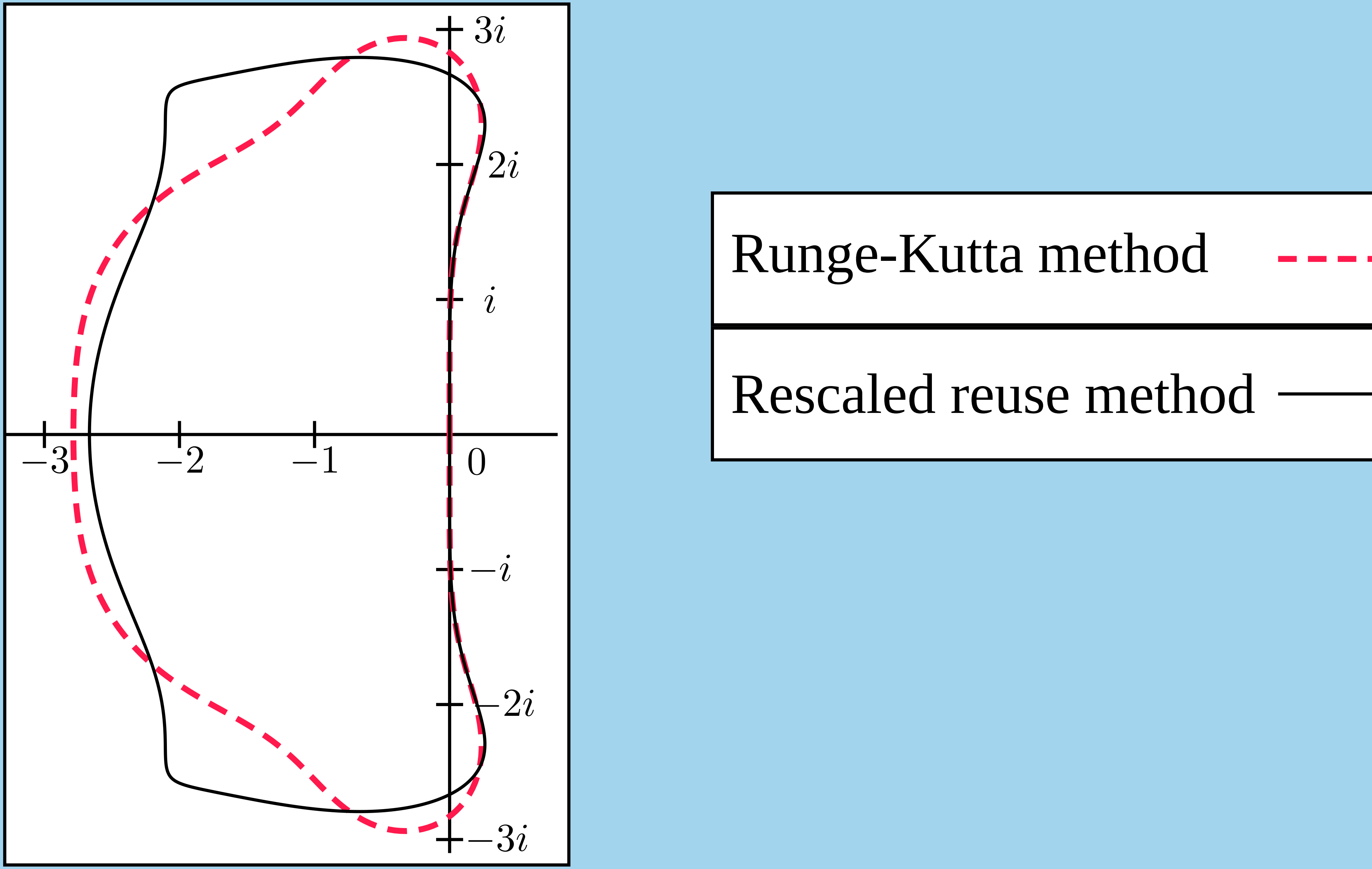
This evaluates f only 3 times per timestep compared with 4 for the original method.

We can understand something about the behaviour of the new method by plotting its stability region.



"Reuse" method


3i
2i
i
−i
−2i
−3i
−3
−2
−1
0
“Reuse” method
Runge-Kutta method

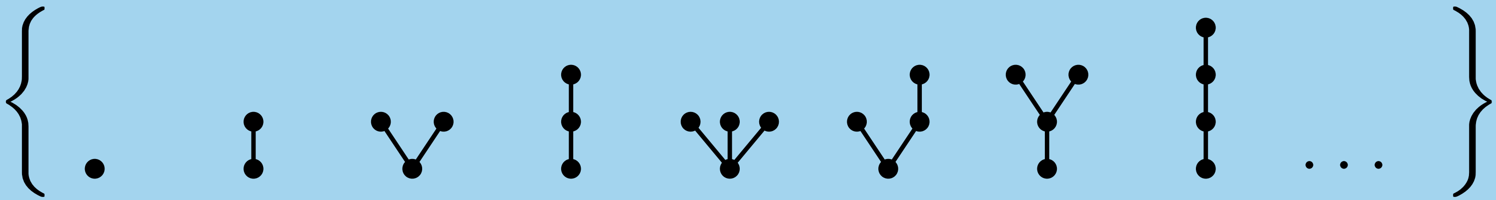

3i
2i
i
−i
−2i
−3i
−3
−2
−1
0
Runge-Kutta method
Rescaled reuse method

Pseudo RK methods

Recall the conditions for a Runge-Kutta method to have order p.

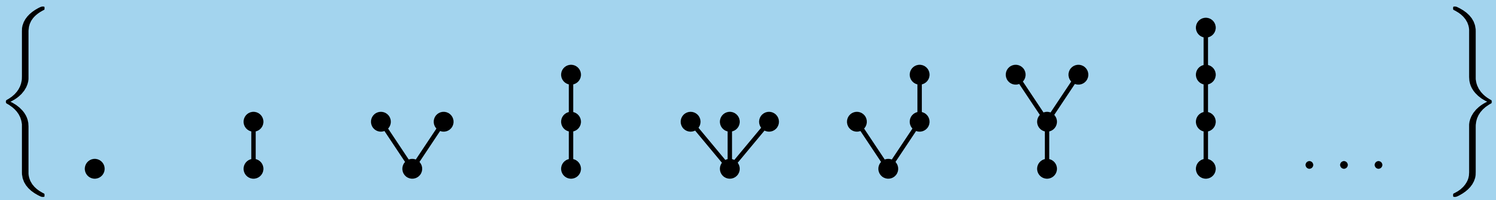
Recall the conditions for a Runge-Kutta method to have order p.

Let T denote the set of rooted trees:



Pseudo RK methods

Recall the conditions for a Runge-Kutta method to have order p.
Let T denote the set of rooted trees:



Associated with each $t \in T$ is an equation

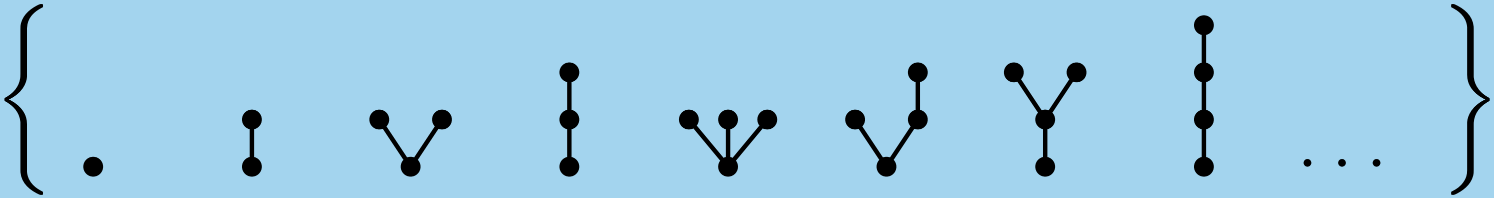
$$\Phi(t) = \frac{1}{\gamma(t)}$$

where the “elementary weight” is a function of the coefficients of the method.

Generalizations of Runge-Kutta Methods

Pseudo RK methods

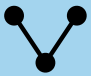
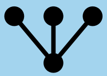
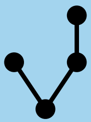
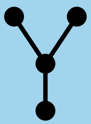
Recall the conditions for a Runge-Kutta method to have order p.
Let T denote the set of rooted trees:

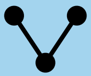
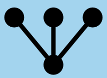
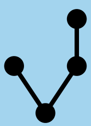
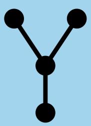


Associated with each $t \in T$ is an equation

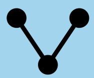
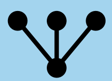
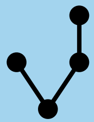
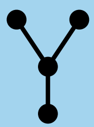
$$\Phi(t) = \frac{1}{\gamma(t)}$$

where the “elementary weight” is a function of the coefficients of the method.
Expressions for Φ and γ are given on the next slide.

t	$\Phi(t)$	$\gamma(t)$
	$\sum b_i$	1
	$\sum b_i c_i$	2
	$\sum b_i c_i^2$	3
	$\sum b_i a_{ij} c_j$	6
	$\sum b_i c_i^3$	4
	$\sum b_i c_i a_{ij} c_j$	8
	$\sum b_i a_{ij} c_j^2$	12
	$\sum b_i a_{ij} a_{jk} c_k$	24

t	$\Phi(t)$	$\gamma(t)$
	$\sum b_i$	1
	$\sum b_i c_i$	2
	$\sum b_i c_i^2$	3
	$\sum b_i a_{ij} c_j$	6
	$\sum b_i c_i^3$	4
	$\sum b_i c_i a_{ij} c_j$	8
	$\sum b_i a_{ij} c_j^2$	12
	$\sum b_i a_{ij} a_{jk} c_k$	24

We will now introduce an additional column $\widehat{\Phi}(t)$

t	$\Phi(t)$	$\gamma(t)$	$\widehat{\Phi}(t)$
	$\sum b_i$	1	$\sum \widehat{b}_i$
	$\sum b_i c_i$	2	$\sum \widehat{b}_i (c_i - 1)$
	$\sum b_i c_i^2$	3	$\sum \widehat{b}_i (c_i - 1)^2$
	$\sum b_i a_{ij} c_j$	6	$\sum \widehat{b}_i (a_{ij} c_j - c_i + \frac{1}{2})$
	$\sum b_i c_i^3$	4	$\sum \widehat{b}_i (c_i - 1)^3$
	$\sum b_i c_i a_{ij} c_j$	8	$\sum \widehat{b}_i (c_i - 1)(a_{ij} c_j - c_i + \frac{1}{2})$
	$\sum b_i a_{ij} c_j^2$	12	$\sum \widehat{b}_i (a_{ij}(c_j^2 - 2c_j) + c_i - \frac{1}{3})$
	$\sum b_i a_{ij} a_{jk} c_k$	24	$\sum \widehat{b}_i (a_{ij}(a_{jk} c_k - c_j) + \frac{1}{2} c_i - \frac{1}{6})$

The expression $\widehat{\Phi}$ would be used in modified order conditions in which stage derivatives are used from the *previous* step.

The expression $\widehat{\Phi}$ would be used in modified order conditions in which stage derivatives are used from the *previous* step.

In a pseudo-Runge-Kutta method stage derivatives are used from both the previous and the current step.

The expression $\widehat{\Phi}$ would be used in modified order conditions in which stage derivatives are used from the *previous* step.

In a pseudo-Runge-Kutta method stage derivatives are used from both the previous and the current step.

The order conditions thus become

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

The expression $\widehat{\Phi}$ would be used in modified order conditions in which stage derivatives are used from the *previous* step.
In a pseudo-Runge-Kutta method stage derivatives are used from both the previous and the current step.
The order conditions thus become

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

A third order method can be constructed with two stages:

$$F_1^{[n]} = f(y_{n-1})$$

$$F_2^{[n]} = f(y_{n-1} + hF_1^{[n]})$$

$$y_n = y_{n-1} - \tfrac{1}{12}hF_1^{[n-1]} - \tfrac{5}{12}hF_2^{[n-1]} + \tfrac{13}{12}hF_1^{[n]} + \tfrac{5}{12}hF_2^{[n]}$$

The idea of using information from a previous step can be taken much further.

The idea of using information from a previous step can be taken much further.

One possible generalization is known as “Two Step Runge-Kutta” methods in which all quantities computed in one step are available for the evaluation of the stages and the output value in the following step.

The idea of using information from a previous step can be taken much further.

One possible generalization is known as “Two Step Runge-Kutta” methods in which all quantities computed in one step are available for the evaluation of the stages and the output value in the following step.

Basic references on pseudo RK methods are given below

G. D. Byrne and R. J. Lambert (1966) ‘Pseudo-Runge-Kutta methods involving two points’, *J. Assoc. Comput. Mach* **13** 114–123.

R. Caira, C. Costabile and F. Costabile (1990) ‘A class of pseudo Runge-Kutta methods’, *BIT* **30** 642–649.

ARK methods

The idea of reuse of stage derivatives can be taken further to produce "Almost Runge-Kutta" methods (ARK methods).

ARK methods

The idea of reuse of stage derivatives can be taken further to produce "Almost Runge-Kutta" methods (ARK methods).
To introduce this generalization we reformulate the reuse method

The idea of reuse of stage derivatives can be taken further to produce “Almost Runge-Kutta” methods (ARK methods).
To introduce this generalization we reformulate the reuse method

$$
\begin{aligned}
Y_1 &= y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), & F_1 &= hf(Y_1)\\
Y_2 &= y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, & F_2 &= f(Y_2)\\
y_n &= y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2
\end{aligned}
$$

The idea of reuse of stage derivatives can be taken further to produce “Almost Runge-Kutta” methods (ARK methods).
To introduce this generalization we reformulate the reuse method

$$Y_1 = y_{n-1} + \tfrac{5}{8}hf(y_{n-1}) - \tfrac{1}{8}hf(y_{n-2}), \qquad F_1 = hf(Y_1)$$
$$Y_2 = y_{n-1} - \tfrac{3}{2}hf(y_{n-1}) + \tfrac{1}{2}hf(y_{n-2}) + 2hF_1, \quad F_2 = hf(Y_2)$$
$$y_n = y_{n-1} + \tfrac{1}{6}hf(y_{n-1}) + \tfrac{2}{3}hF_1 + \tfrac{1}{6}hF_2$$

$$\boldsymbol{y_n \longrightarrow y_1^{[n]}}, \qquad \boldsymbol{hf(y_n) \longrightarrow y_2^{[n]}}$$

ARK methods

The idea of reuse of stage derivatives can be taken further to produce “Almost Runge-Kutta” methods (ARK methods).
To introduce this generalization we reformulate the reuse method

$$
\begin{aligned}
Y_1 &= y_1^{[n-1]} + \tfrac{1}{2} y_2^{[n-1]} + \tfrac{1}{8}(y_2^{[n-1]} - y_2^{[n-2]}), \qquad F_1 = f(Y_1) \\
Y_2 &= y_1^{[n-1]} - y_2^{[n-1]} - \tfrac{1}{2}(y_2^{[n-1]} - y_2^{[n-2]}) + 2hF_1, \quad F_2 = f(Y_2) \\
y_1^{[n]} &= y_1^{[n-1]} + \tfrac{1}{6} y_2^{[n-1]} + \tfrac{2}{3} hF_1 + \tfrac{1}{6} hF_2 \\
y_2^{[n]} &= hf(y_1^{[n]})
\end{aligned}
$$

ARK methods

The idea of reuse of stage derivatives can be taken further to produce "Almost Runge-Kutta" methods (ARK methods).
To introduce this generalization we reformulate the reuse method

$$
\begin{aligned}
Y_1 &= y_1^{[n-1]} + \tfrac{1}{2} y_2^{[n-1]} + \tfrac{1}{8}(y_2^{[n-1]} - y_2^{[n-2]}), & F_1 &= f(Y_1) \\
Y_2 &= y_1^{[n-1]} - y_2^{[n-1]} - \tfrac{1}{2}(y_2^{[n-1]} - y_2^{[n-2]}) + 2hF_1, & F_2 &= f(Y_2) \\
y_1^{[n]} &= y_1^{[n-1]} + \tfrac{1}{6} y_2^{[n-1]} + \tfrac{2}{3} hF_1 + \tfrac{1}{6} hF_2 \\
y_2^{[n]} &= hf(y_1^{[n]})
\end{aligned}
$$

$$
\boldsymbol{y_2^{[n]} - y_2^{[n-1]} \longrightarrow y_3^{[n]}}
$$

ARK methods

The idea of reuse of stage derivatives can be taken further to produce “Almost Runge-Kutta” methods (ARK methods).
To introduce this generalization we reformulate the reuse method

$$
\begin{aligned}
Y_1 &= y_1^{[n-1]} + \tfrac{1}{2} y_2^{[n-1]} + \tfrac{1}{8} y_3^{[n]}, & F_1 &= f(Y_1)\\
Y_2 &= y_1^{[n-1]} - y_2^{[n-1]} - \tfrac{1}{2} y_3^{[n]} + 2hF_1, & F_2 &= f(Y_2)\\
y_1^{[n]} &= y_1^{[n-1]} + \tfrac{1}{6} y_2^{[n-1]} + \tfrac{2}{3} hF_1 + \tfrac{1}{6} hF_2\\
y_2^{[n]} &= hf(y_1^{[n]})\\
y_3^{[n]} &= y_2^{[n]} - y_2^{[n-1]}
\end{aligned}
$$

Note that in this formulation there are three quantities passed from step to step and three derivative computations within each step.
The three input and output quantities approximate scaled derivatives as follows

$$
\begin{aligned}
y_1^{[n-1]} &\approx y(x_{n-1}) & \qquad y_1^{[n]} &\approx y(x_n)\\
y_2^{[n-1]} &\approx hy'(x_{n-1}) & \qquad y_2^{[n]} &\approx hy'(x_n)\\
y_3^{[n-1]} &\approx h^2y''(x_{n-1}) & \qquad y_3^{[n]} &\approx h^2y''(x_n)
\end{aligned}
$$

Even though the method has order 4, the third output quantity is accurate only to order 2.

We now extend this idea by restoring a fourth stage and making $y_3^{[n]}$ depend on quantities computed in the step.

We now extend this idea by restoring a fourth stage and making $y_3^{[n]}$ depend on quantities computed in the step. For example

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ \hline y_1^{[n]} \\ y_2^{[n]} \\ y_3^{[n]} \end{bmatrix} = \left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 1 & 1 & \frac{1}{2} \\ \frac{1}{16} & 0 & 0 & 0 & 1 & \frac{7}{16} & \frac{1}{16} \\ -\frac{4}{3} & 2 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{1}{4} \\ 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ \hline 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 0 & -\frac{2}{3} & 2 & 0 & -1 & 0 \end{array}\right] \begin{bmatrix} hF_1 \\ hF_2 \\ hF_3 \\ hF_4 \\ \hline y_1^{[n-1]} \\ y_2^{[n-1]} \\ y_3^{[n-1]} \end{bmatrix}$$

- The abscissae for this method are $[1\ \ \frac{1}{2}\ \ 1\ \ 1]$.

- The abscissae for this method are $[1 \ \ \frac{1}{2} \ \ 1 \ \ 1]$.
- It has exactly the same stability region as for a classical fourth order Runge-Kutta method.

- The abscissae for this method are $[1 \;\; \frac{1}{2} \;\; 1 \;\; 1]$.
- It has exactly the same stability region as for a classical fourth order Runge-Kutta method.
- The stage-order is 2 rather than 1 as for a classical method.

- The abscissae for this method are $[1 \;\; \frac{1}{2} \;\; 1 \;\; 1]$.
- It has exactly the same stability region as for a classical fourth order Runge-Kutta method.
- The stage-order is 2 rather than 1 as for a classical method.
- A possible starting method is

$$y_1^{[0]} = y_0, \quad y_2^{[0]} = hf(y_1^{[0]}), \quad y_3^{[0]} = hf(y_0 + y_2^{[0]}) - y_2^{[0]}$$

- The abscissae for this method are $[1 \;\; \frac{1}{2} \;\; 1 \;\; 1]$.
- It has exactly the same stability region as for a classical fourth order Runge-Kutta method.
- The stage-order is 2 rather than 1 as for a classical method.
- A possible starting method is

$$y_1^{[0]} = y_0, \quad y_2^{[0]} = hf(y_1^{[0]}), \quad y_3^{[0]} = hf(y_0 + y_2^{[0]}) - y_2^{[0]}$$

- Stepsize change $h \to rh$ can be achieved without loss of order by

$$y_1^{[n]} \to y_1^{[n]}, \quad y_2^{[n]} \to r y_2^{[n]}, \quad y_3^{[n]} \to r^2 y_3^{[n]}$$

- The abscissae for this method are $[1 \;\; \frac{1}{2} \;\; 1 \;\; 1]$.
- It has exactly the same stability region as for a classical fourth order Runge-Kutta method.
- The stage-order is 2 rather than 1 as for a classical method.
- A possible starting method is

$$y_1^{[0]} = y_0, \quad y_2^{[0]} = hf(y_1^{[0]}), \quad y_3^{[0]} = hf(y_0 + y_2^{[0]}) - y_2^{[0]}$$

- Stepsize change $h \to rh$ can be achieved without loss of order by

$$y_1^{[n]} \to y_1^{[n]}, \quad y_2^{[n]} \to r y_2^{[n]}, \quad y_3^{[n]} \to r^2 y_3^{[n]}$$

- A method like this is an “Almost Runge-Kutta method” (ARK method).

Basic references on ARK methods are given below

J. C. Butcher (1997b) ‘An introduction to “Almost Runge–Kutta" methods’, *Appl. Numer. Math.* **24** 331–342.

J. C. Butcher (1998) ‘ARK methods up to order five’, *Numer. Algorithms*, **17** 193–221.

J. C. Butcher and N. Moir (2003) ‘Experiments with a new fifth order method’, *Numer. Algorithms*, **33** 137–151.

N. Moir (2005) ‘ARK methods: some recent developments’, *J. Comput. Appl. Math.*, **175** 101–111.

Effective Order

Developing the ideas on Runge-Kutta and pseudo Runge-Kutta methods, we introduce a group G whose elements are mappings on the set of trees to real numbers

Effective Order

Developing the ideas on Runge-Kutta and pseudo Runge-Kutta methods, we introduce a group G whose elements are mappings on the set of trees to real numbers and where the group operation is defined according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

Effective Order

Developing the ideas on Runge-Kutta and pseudo Runge-Kutta methods, we introduce a group G whose elements are mappings on the set of trees to real numbers and where the group operation is defined according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.
A Runge-Kutta method is represented by its sequence of elementary weights

Effective Order

Developing the ideas on Runge-Kutta and pseudo Runge-Kutta methods, we introduce a group G whose elements are mappings on the set of trees to real numbers and where the group operation is defined according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

A Runge-Kutta method is represented by its sequence of elementary weights and the flow of a vector field (that is, the exact solution), which we will denote by E, is represented by $t \mapsto \gamma(t)^{-1}$.

Effective Order

Developing the ideas on Runge-Kutta and pseudo Runge-Kutta methods, we introduce a group G whose elements are mappings on the set of trees to real numbers and where the group operation is defined according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

A Runge-Kutta method is represented by its sequence of elementary weights and the flow of a vector field (that is, the exact solution), which we will denote by E, is represented by $t \mapsto \gamma(t)^{-1}$.

We will write H_p as the normal subgroup whose members are characterized by $t \mapsto 0$ if t has less than or equal to p vertices.

For a Runge-Kutta method to have order p, its corresponding group element, α say, is in the same coset αH_p as E.

For a Runge-Kutta method to have order p, its corresponding group element, α say, is in the same coset αH_p as E. That is

$$\alpha H_p = E H_p$$

For a Runge-Kutta method to have order p, its corresponding group element, α say, is in the same coset αH_p as E. That is

$$\alpha H_p = E H_p$$

A method has *effective order* p if there exists $\beta \in G$ such that

$$\beta\alpha H_p = E\beta H_p$$

For a Runge-Kutta method to have order p, its corresponding group element, α say, is in the same coset αH_p as E. That is

$$\alpha H_p = E H_p$$

A method has *effective order* p if there exists $\beta \in G$ such that

$$\beta\alpha H_p = E\beta H_p$$

We will illustrate the group operation in a table

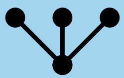
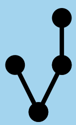
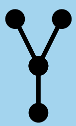
For a Runge-Kutta method to have order p, its corresponding group element, α say, is in the same coset αH_p as E. That is

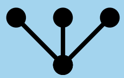
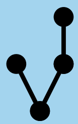
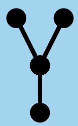
$$\alpha H_p = E H_p$$

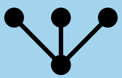
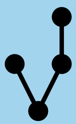
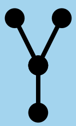
A method has *effective order* p if there exists $\beta \in G$ such that

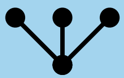
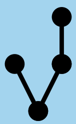
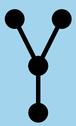
$$\beta \alpha H_p = E \beta H_p$$

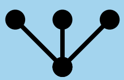
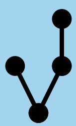
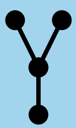
We will illustrate the group operation in a table where we also give values of E.

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	$\alpha_8+\alpha_4\beta_1+\alpha_2\beta_2+\alpha_1\beta_4+\beta_8$	$\frac{1}{24}$

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

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

General linear methods

All the generalizations we have considered possess several components in common.

General linear methods

All the generalizations we have considered possess several components in common.

1. A number of quantities are imported at the start of any particular step.

General linear methods

All the generalizations we have considered possess several components in common.

1. A number of quantities are imported at the start of any particular step.
2. A number of stage values together with the corresponding stage derivatives are computed.

General linear methods

All the generalizations we have considered possess several components in common.

1. A number of quantities are imported at the start of any particular step.
2. A number of stage values together with the corresponding stage derivatives are computed.
3. Each of the stage values is a linear combination of the stage derivatives and the input quantities.

General linear methods

All the generalizations we have considered possess several components in common.

1. A number of quantities are imported at the start of any particular step.
2. A number of stage values together with the corresponding stage derivatives are computed.
3. Each of the stage values is a linear combination of the stage derivatives and the input quantities.
4. Output quantities are computed corresponding to the input quantities in step 1.

General linear methods

All the generalizations we have considered possess several components in common.

1. A number of quantities are imported at the start of any particular step.
2. A number of stage values together with the corresponding stage derivatives are computed.
3. Each of the stage values is a linear combination of the stage derivatives and the input quantities.
4. Output quantities are computed corresponding to the input quantities in step 1.
5. These output quantities are also linear combinations of the stage derivatives and the input quantities.

We have a range of possibilities from 1 input quantity, as in Runge-Kutta methods, to a large number as in multistep methods.

We have a range of possibilities from 1 input quantity, as in Runge-Kutta methods, to a large number as in multistep methods.

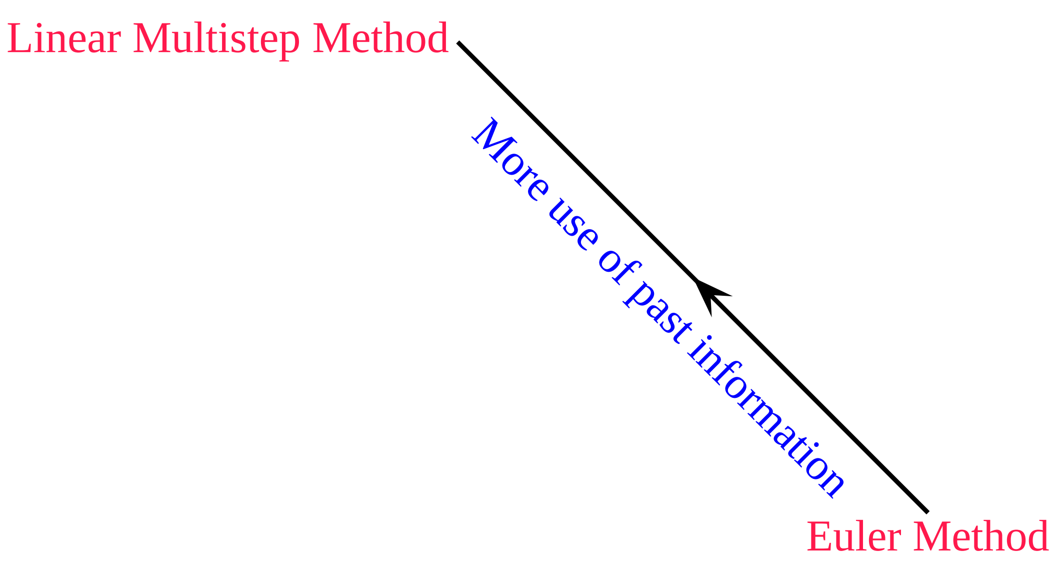
We also have a range of possibilities for the number of stages from 1, as in linear multistep method, to a large number as in Runge-Kutta methods and their generalizations.

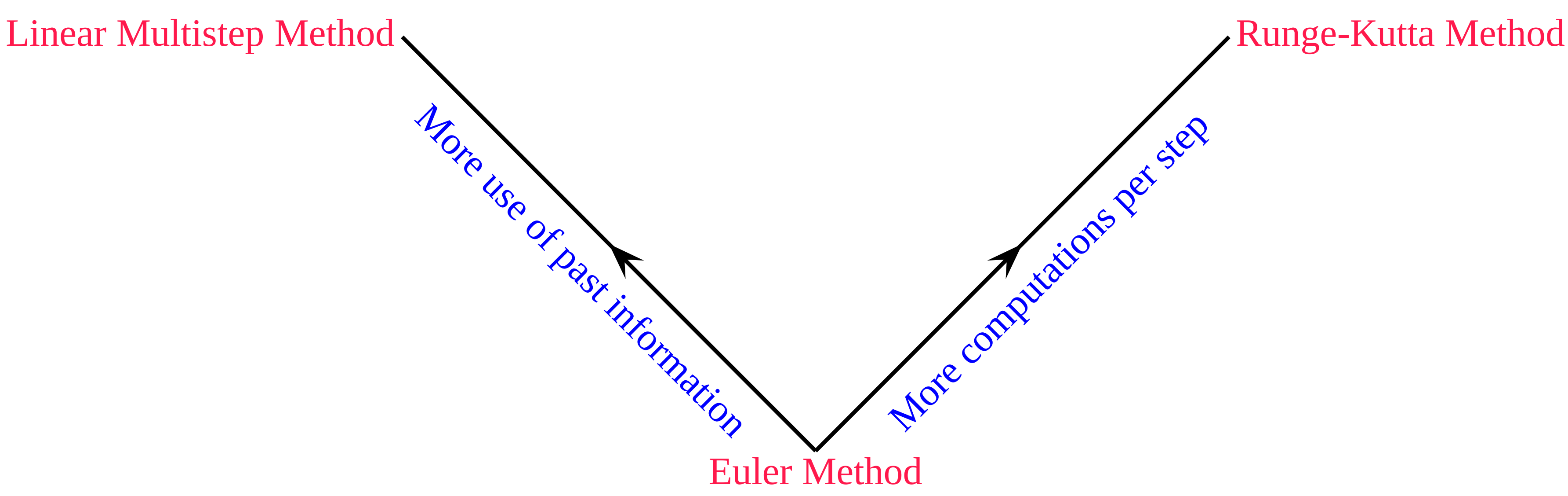
We have a range of possibilities from 1 input quantity, as in Runge-Kutta methods, to a large number as in multistep methods.

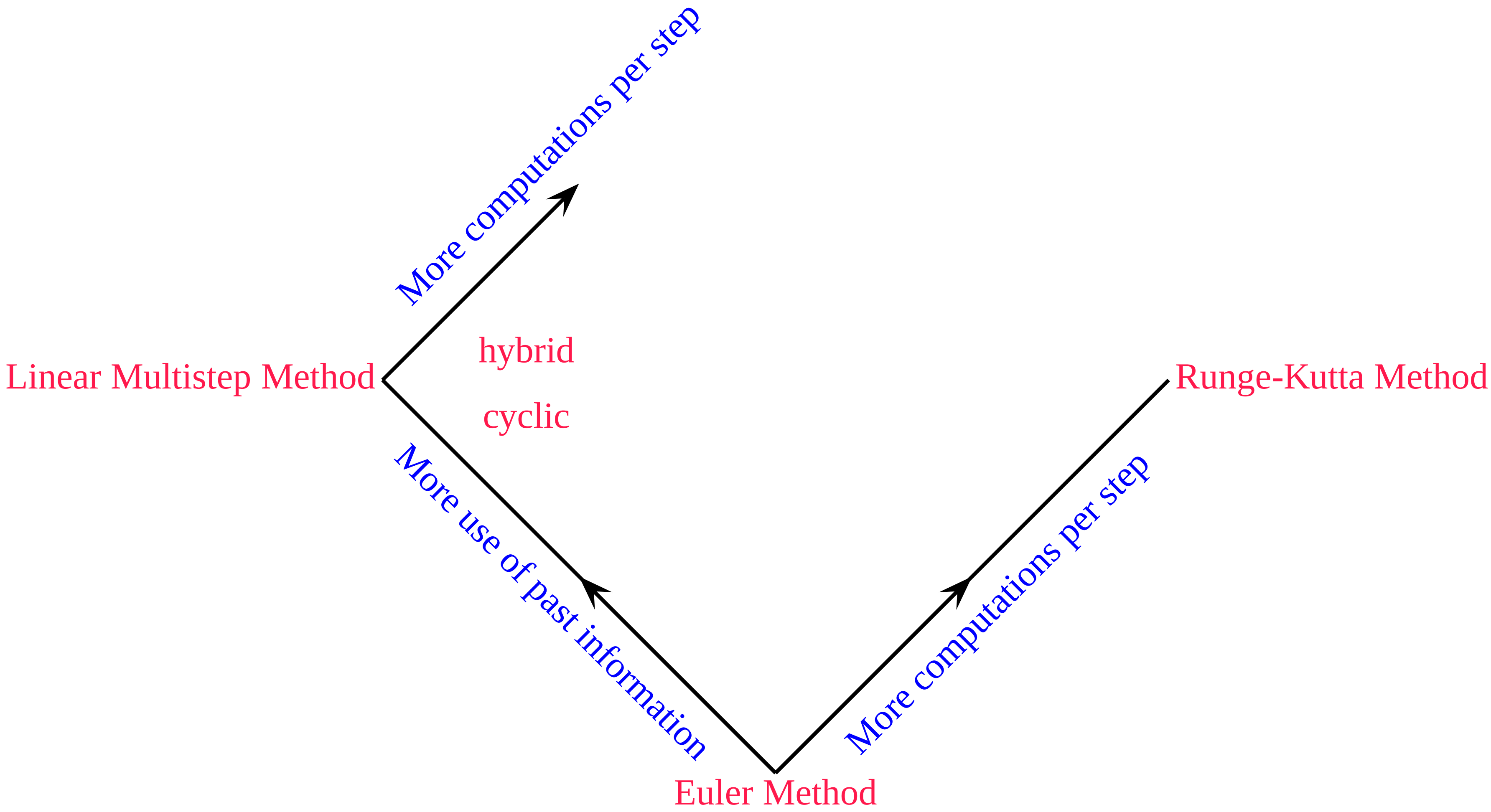
We also have a range of possibilities for the number of stages from 1, as in linear multistep method, to a large number as in Runge-Kutta methods and their generalizations.

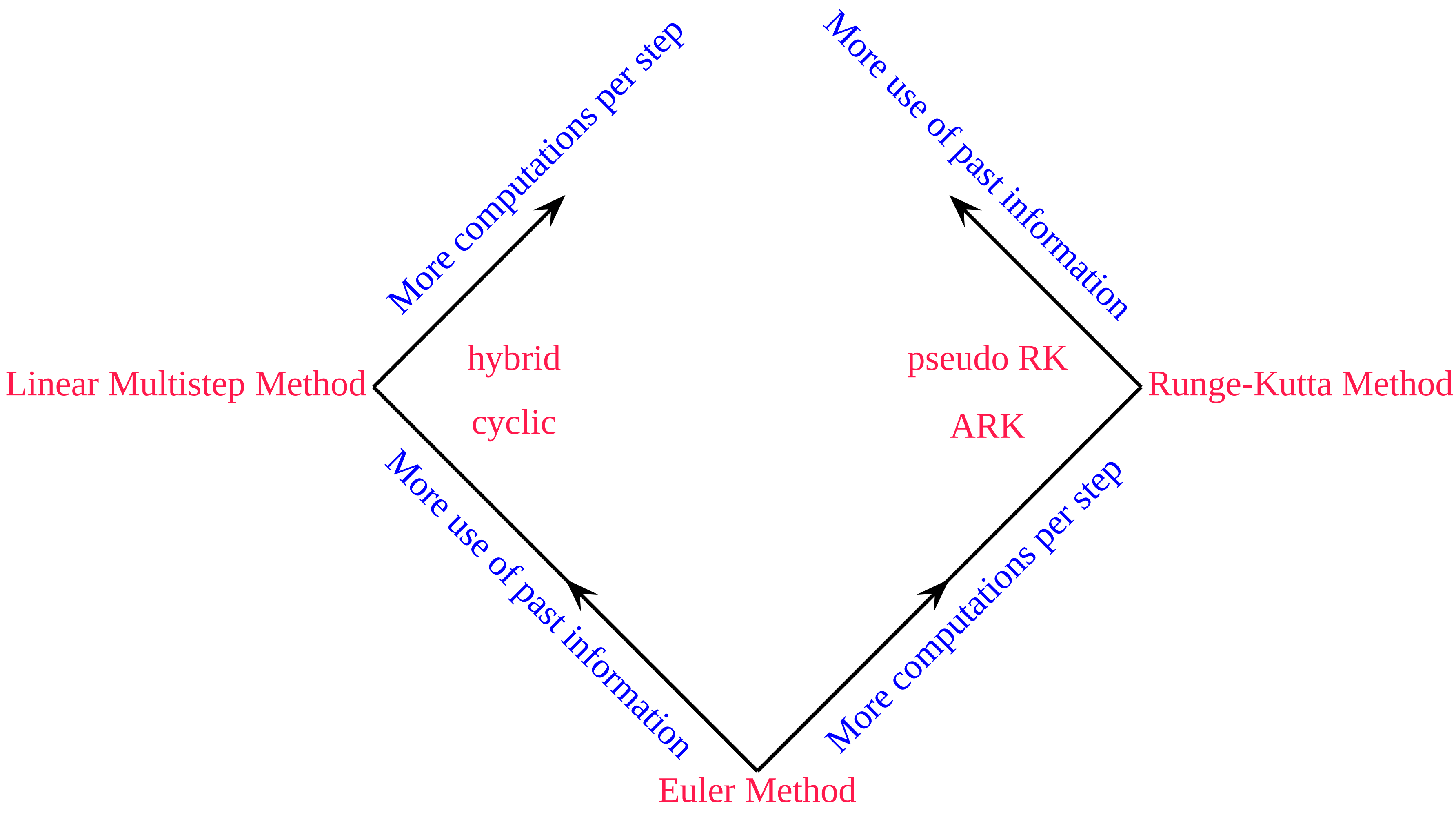
We will summarize this in the diagram on the next slide.

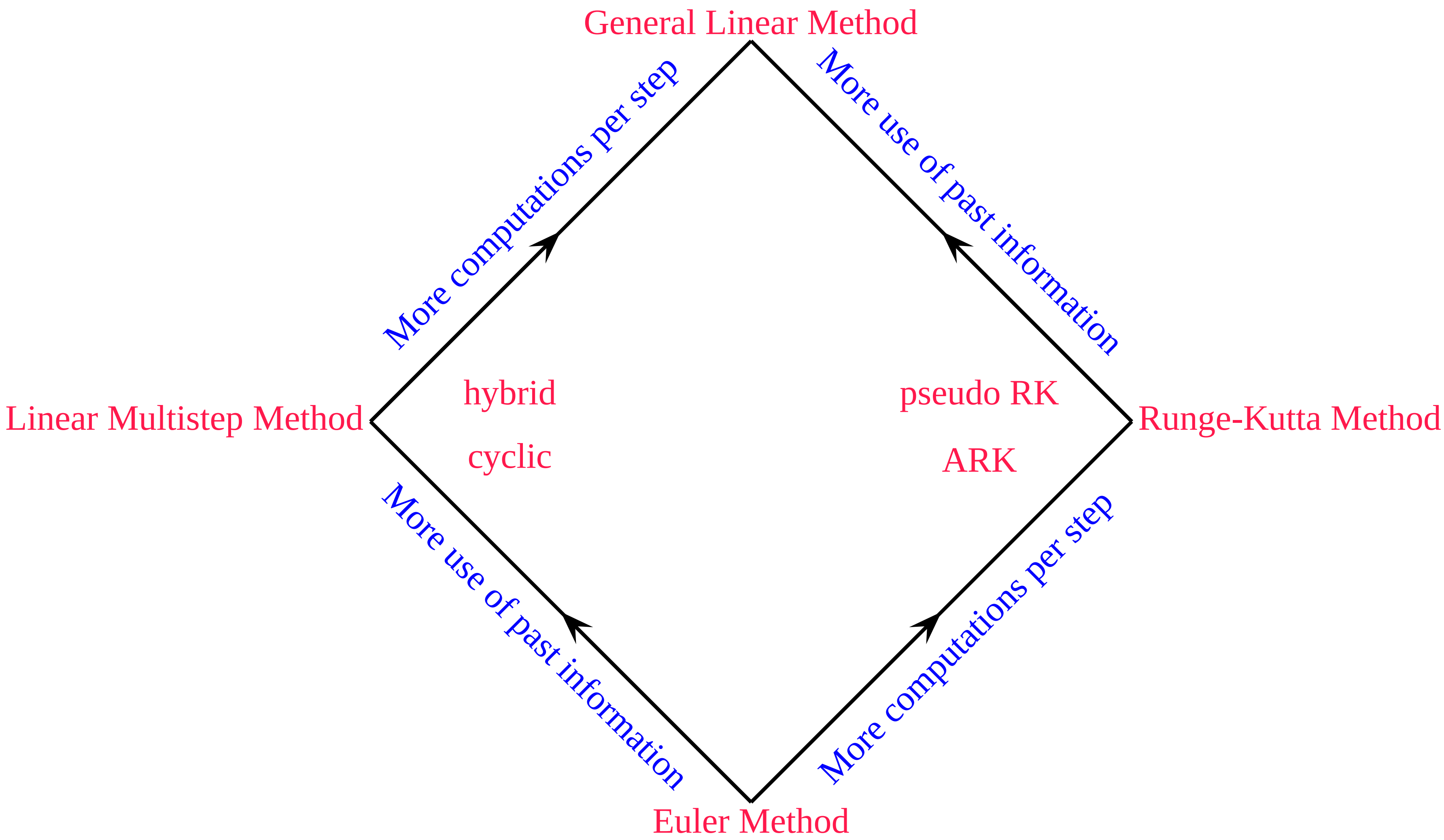
Euler Method


Linear Multistep Method
More use of past information
Euler Method


Linear Multistep Method
Runge-Kutta Method
More use of past information
More computations per step
Euler Method


More computations per step
hybrid
cyclic
Linear Multistep Method
Runge-Kutta Method
More use of past information
More computations per step
Euler Method


More computations per step
More use of past information
Linear Multistep Method
hybrid
cyclic
pseudo RK
ARK
Runge-Kutta Method
More use of past information
More computations per step
Euler Method


General Linear Method
More computations per step
More use of past information
Linear Multistep Method
hybrid
cyclic
pseudo RK
ARK
Runge-Kutta Method
More use of past information
More computations per step
Euler Method

The number of input quantities will be denoted by r and the number of stages by s.

The number of input quantities will be denoted by r and the number of stages by s.

The quantities input at the start of step n will be denoted by $y_i^{[n-1]}$, $i = 1, 2, \dots, r$.

The number of input quantities will be denoted by r and the number of stages by s.

The quantities input at the start of step n will be denoted by $y_i^{[n-1]}$, $i = 1, 2, \ldots, r$.

The stage values computed in step n will be denoted by Y_i, $i = 1, 2, \ldots, s$.

Formulation

The number of input quantities will be denoted by r and the number of stages by s.

The quantities input at the start of step n will be denoted by $y_i^{[n-1]}$, $i = 1, 2, \ldots, r$.

The stage values computed in step n will be denoted by Y_i, $i = 1, 2, \ldots, s$.

The stage derivatives computed in step n will be denoted by F_i, $i = 1, 2, \ldots, s$.

Formulation

The number of input quantities will be denoted by r and the number of stages by s.

The quantities input at the start of step n will be denoted by $y_i^{[n-1]}$, $i = 1, 2, \dots, r$.

The stage values computed in step n will be denoted by Y_i, $i = 1, 2, \dots, s$.

The stage derivatives computed in step n will be denoted by F_i, $i = 1, 2, \dots, s$.

The quantities exported at the end of step n will be denoted by $y_i^{[n]}$, $i = 1, 2, \dots, r$.

For convenience we will write:

$$y^{[n-1]} = \begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_r^{[n-1]} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}$$

For convenience we will write:

$$y^{[n-1]} = \begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_r^{[n-1]} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}$$

and we note that $F_i = f(Y_i)$, $i = 1, 2, \ldots, s$, for a non-stiff or stiff problem

For convenience we will write:

$$y^{[n-1]} = \begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_r^{[n-1]} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}$$

and we note that $F_i = f(Y_i)$, $i = 1, 2, \ldots, s$, for a non-stiff or stiff problem , with a more complicated relationship between these vectors for a DAE.

For convenience we will write:

$$y^{[n-1]} = \begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_r^{[n-1]} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}$$

and we note that $F_i = f(Y_i)$, $i = 1, 2, \dots, s$, for a non-stiff or stiff problem , with a more complicated relationship between these vectors for a DAE.

We now go through the process of carrying out a step in terms of this notation.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{[n-1]}$.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{[n-1]}$.
4. The r subvectors comprising $y^{[n]}$ are computed corresponding to the $y^{[n-1]}$ subvectors.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{[n-1]}$.
4. The r subvectors comprising $y^{[n]}$ are computed corresponding to the $y^{[n-1]}$ subvectors.
5. The $y_i^{[n]}$ are linear combinations of the hF_j and the $y_j^{[n-1]}$.

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{[n-1]}$.
4. The r subvectors comprising $y^{[n]}$ are computed corresponding to the $y^{[n-1]}$ subvectors.
5. The $y_i^{[n]}$ are linear combinations of the hF_j and the $y_j^{[n-1]}$.

The matrices of coefficients in step 3 are A and U

1. The r subvectors comprising $y^{[n-1]}$ are imported at the start of step n.
2. The subvectors in Y and F are computed.
3. Each of the Y_i is a linear combination of the hF_j and the $y_j^{[n-1]}$.
4. The r subvectors comprising $y^{[n]}$ are computed corresponding to the $y^{[n-1]}$ subvectors.
5. The $y_i^{[n]}$ are linear combinations of the hF_j and the $y_j^{[n-1]}$.

The matrices of coefficients in step 3 are A and U and those in step 5 are B and V.

The formulae for the various steps are

$$Y_i = \sum_{j=1}^{s} a_{ij} h F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \qquad i = 1, 2, \ldots s$$

The formulae for the various steps are

$$Y_i = \sum_{j=1}^{s} a_{ij} h F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \quad F_i = f(Y_i), \quad i = 1, 2, \ldots s$$

The formulae for the various steps are

$$Y_i = \sum_{j=1}^{s} a_{ij} h F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \quad F_i = f(Y_i), \quad i = 1, 2, \ldots s$$

$$y_i^{[n]} = \sum_{j=1}^{s} b_{ij} h F_j + \sum_{j=1}^{r} v_{ij} y_j^{[n-1]}, \qquad i = 1, 2, \ldots r$$

The formulae for the various steps are

$$Y_i = \sum_{j=1}^{s} a_{ij} h F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \quad F_i = f(Y_i), \quad i = 1, 2, \dots s$$

$$y_i^{[n]} = \sum_{j=1}^{s} b_{ij} h F_j + \sum_{j=1}^{r} v_{ij} y_j^{[n-1]}, \qquad i = 1, 2, \dots r$$

or, using a compact notation,

$$Y = (A \otimes I) hF + (U \otimes I) y^{[n-1]}$$
$$y^{[n]} = (B \otimes I) hF + (V \otimes I) y^{[n-1]}$$

Consistency, Stability, Convergence

Just as for linear multistep methods, the concept of convergence expresses the ability of a numerical method to generate arbitrarily accurate approximations to the solution at a specific time value for sufficiently small stepsize.

Consistency, Stability, Convergence

Just as for linear multistep methods, the concept of convergence expresses the ability of a numerical method to generate arbitrarily accurate approximations to the solution at a specific time value for sufficiently small stepsize.

The basic theorem is

$$\left.\begin{array}{r}\text{Consistency}\\ \text{Stability}\end{array}\right\} \Longleftrightarrow \text{Convergence}$$

Just as for linear multistep methods, the concept of convergence expresses the ability of a numerical method to generate arbitrarily accurate approximations to the solution at a specific time value for sufficiently small stepsize.

The basic theorem is

$$\left.\begin{array}{r}\text{Consistency}\\ \text{Stability}\end{array}\right\} \Longleftrightarrow \quad \text{Convergence}$$

and we will discuss the meaning of this result in the next few slides.

Introduce two vectors $u, v \in \mathbb{R}^r$, known as the pre-consistency and consistency vectors respectively.

Introduce two vectors $u, v \in \mathbb{R}^r$, known as the pre-consistency and consistency vectors respectively.

We will require that the GL method with inputs

$$y_i^{[n-1]} = u_i y(x_{n-1}) + v_i h y'(x_{n-1}) + O(h^2), \quad i = 1, 2, \ldots, r$$

Introduce two vectors $u, v \in \mathbb{R}^r$, known as the pre-consistency and consistency vectors respectively.

We will require that the GL method with inputs

$$y_i^{[n-1]} = u_i y(x_{n-1}) + v_i h y'(x_{n-1}) + O(h^2), \quad i = 1, 2, \ldots, r$$

will yield stage values

$$Y_i = y(x_{n-1}) + O(h), \quad i = 1, 2, \ldots, s$$

Introduce two vectors $u, v \in \mathbb{R}^r$, known as the pre-consistency and consistency vectors respectively.

We will require that the GL method with inputs

$$y_i^{[n-1]} = u_i y(x_{n-1}) + v_i h y'(x_{n-1}) + O(h^2), \quad i = 1, 2, \ldots, r$$

will yield stage values

$$Y_i = y(x_{n-1}) + O(h), \quad i = 1, 2, \ldots, s$$

and outputs

$$y_i^{[n]} = u_i y(x_n) + v_i h y'(x_n) + O(h^2), \quad i = 1, 2, \ldots, s$$

By Taylor's theory these requirements can be written

$$Uu = \mathbf{1}$$

$$Vu = u \qquad \textbf{(*)}$$

$$B\mathbf{1} + Vv = u + v \qquad \textbf{(**)}$$

By Taylor's theory these requirements can be written

$$\begin{aligned} Uu &= \mathbf{1} \\ Vu &= u \qquad \text{(*)} \\ B\mathbf{1} + Vv &= u + v \qquad \text{(**)} \end{aligned}$$

Note that (*) and (**) are related to the ability of the numerical method to solve the problem

$$y'(x) = 1$$

exactly, for an arbitrary initial value.

Stability refers to the ability of a method to generate a convergent sequence of approximations to the problem

$$y'(x) = 0, \qquad y(0) = 0,$$

under appropriate conditions on the values of the initial values $y^{[0]}$.

Stability refers to the ability of a method to generate a convergent sequence of approximations to the problem

$$y'(x) = 0, \qquad y(0) = 0,$$

under appropriate conditions on the values of the initial values $y^{[0]}$.

This is equivalent to the requirement that V should be power-bounded.

Stability refers to the ability of a method to generate a convergent sequence of approximations to the problem

$$y'(x) = 0, \qquad y(0) = 0,$$

under appropriate conditions on the values of the initial values $y^{[0]}$.

This is equivalent to the requirement that V should be power-bounded.

This in turn is equivalent to the requirement that the minimal polynomial of V has all its zeros in the closed unit disc with only simple zeros on the boundary.

Order

The input to a step is an approximation to some vector of quantities related to the exact solution at x_{n-1}.

Order

The input to a step is an approximation to some vector of quantities related to the exact solution at x_{n-1}.

When the step has been completed, the vectors comprising the output are approximations to the same quantities, but now related to x_n.

Order

The input to a step is an approximation to some vector of quantities related to the exact solution at x_{n-1}.

When the step has been completed, the vectors comprising the output are approximations to the same quantities, but now related to x_n.

If the input is exactly what it it is supposed to approximate, then the "local truncation error" is defined as the error in the output after a single step.

Order

The input to a step is an approximation to some vector of quantities related to the exact solution at x_{n-1}.

When the step has been completed, the vectors comprising the output are approximations to the same quantities, but now related to x_n.

If the input is exactly what it it is supposed to approximate, then the "local truncation error" is defined as the error in the output after a single step.

If this can be estimated in terms of h^{p+1}, then the method has order p.

Order

The input to a step is an approximation to some vector of quantities related to the exact solution at x_{n-1}.

When the step has been completed, the vectors comprising the output are approximations to the same quantities, but now related to x_n.

If the input is exactly what it it is supposed to approximate, then the "local truncation error" is defined as the error in the output after a single step.

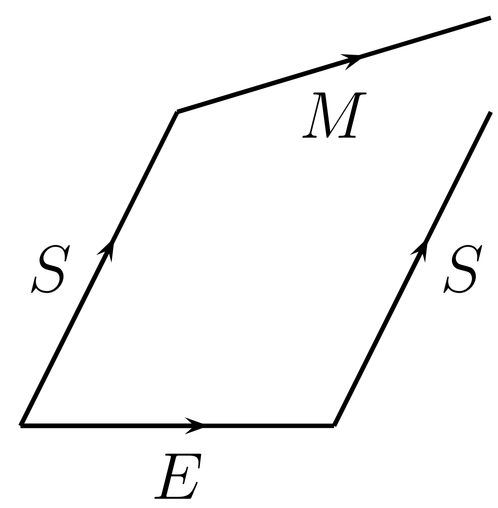
If this can be estimated in terms of h^{p+1}, then the method has order p.

We will refer to the calculation which produces $y^{[n-1]}$ from $y(x_{n-1})$ as a "starting method".

Let S denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$ and a corresponding finishing method $F : \mathbb{R}^{rN} \to \mathbb{R}^N$ such that $F \circ S = \mathrm{id}$.

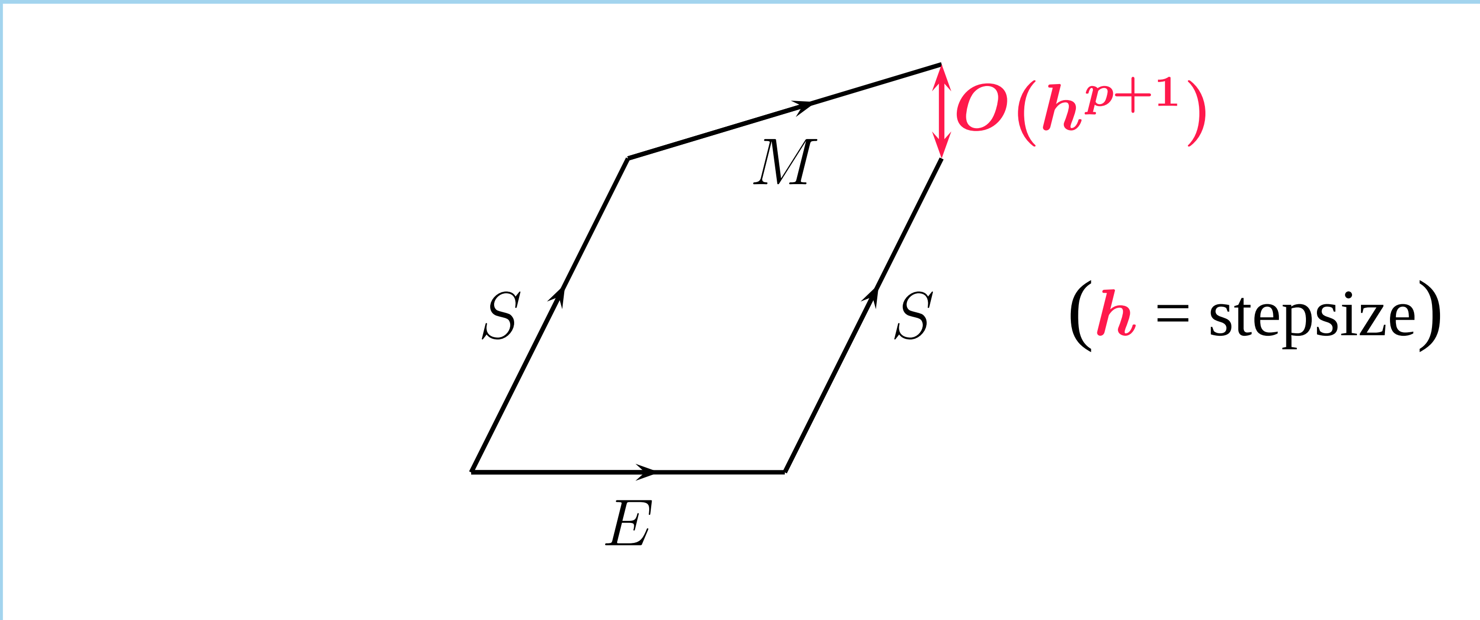
Let S denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$ and a corresponding finishing method $F : \mathbb{R}^{rN} \to \mathbb{R}^N$ such that $F \circ S = \text{id}$.

The order of accuracy of a multivalue method is defined in terms of the diagram



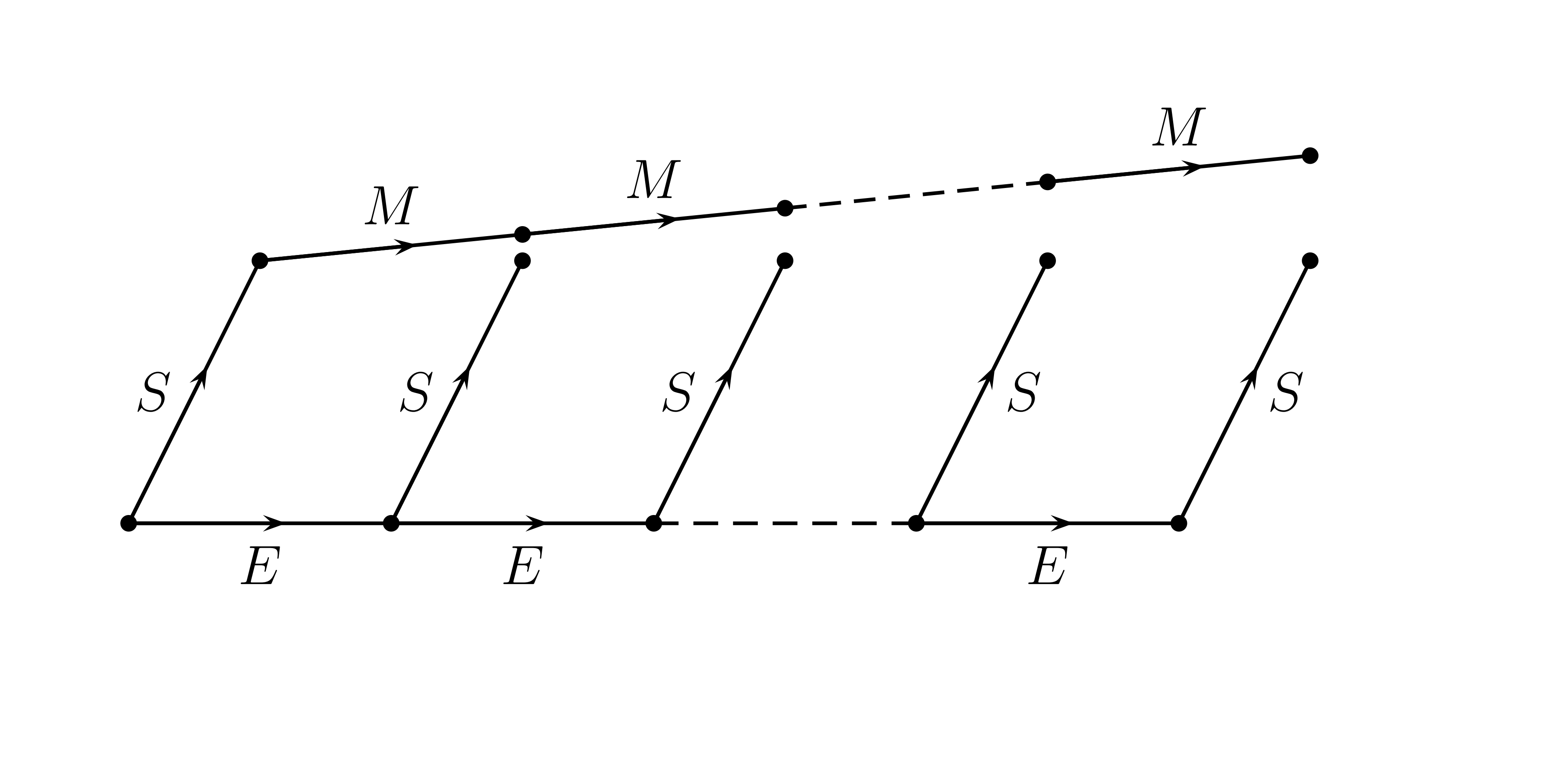
Let S denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$ and a corresponding finishing method $F : \mathbb{R}^{rN} \to \mathbb{R}^N$ such that $F \circ S = \text{id}$.

The order of accuracy of a multivalue method is defined in terms of the diagram

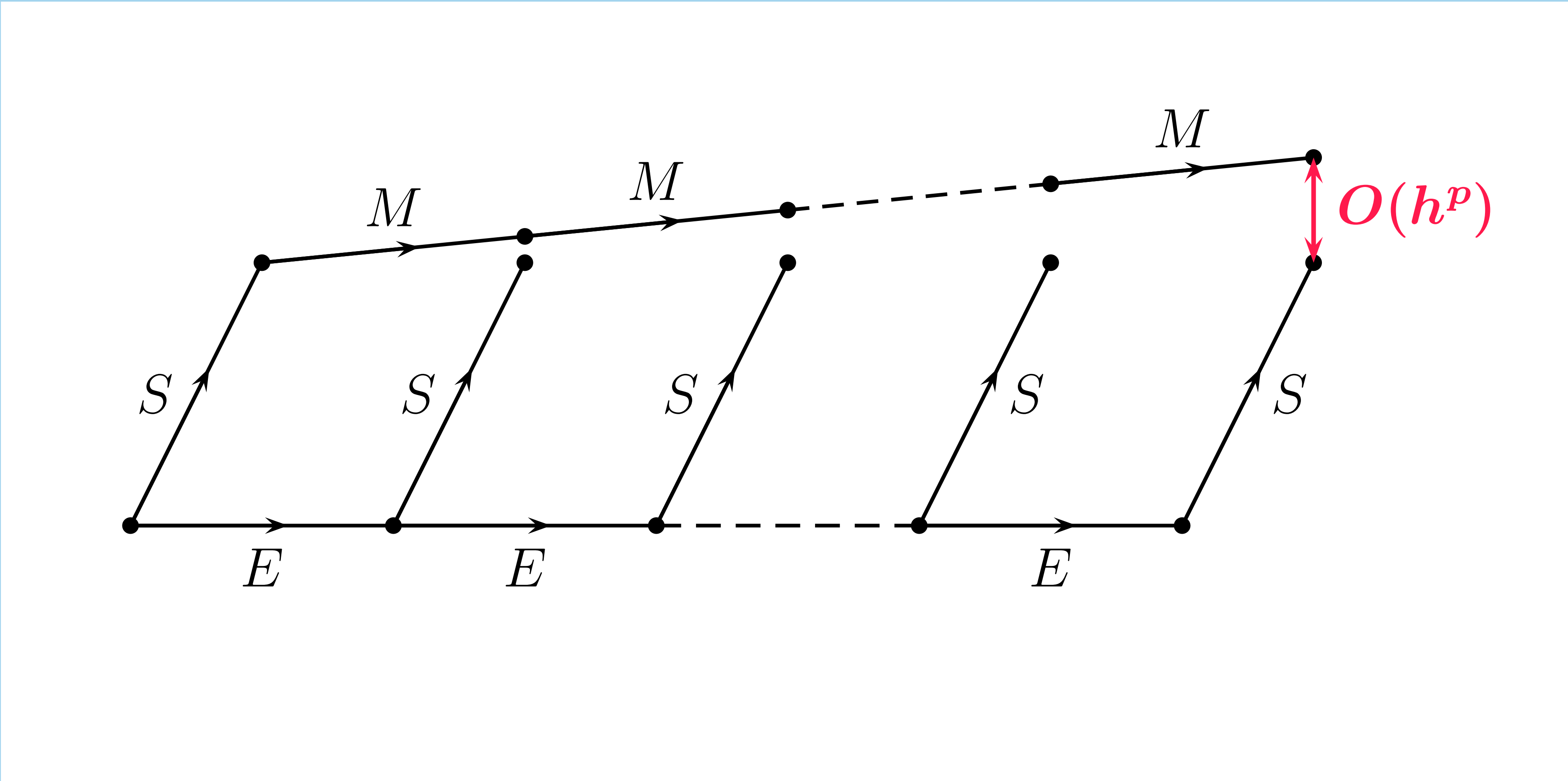


By duplicating this diagram over many steps, global error estimates may be found.

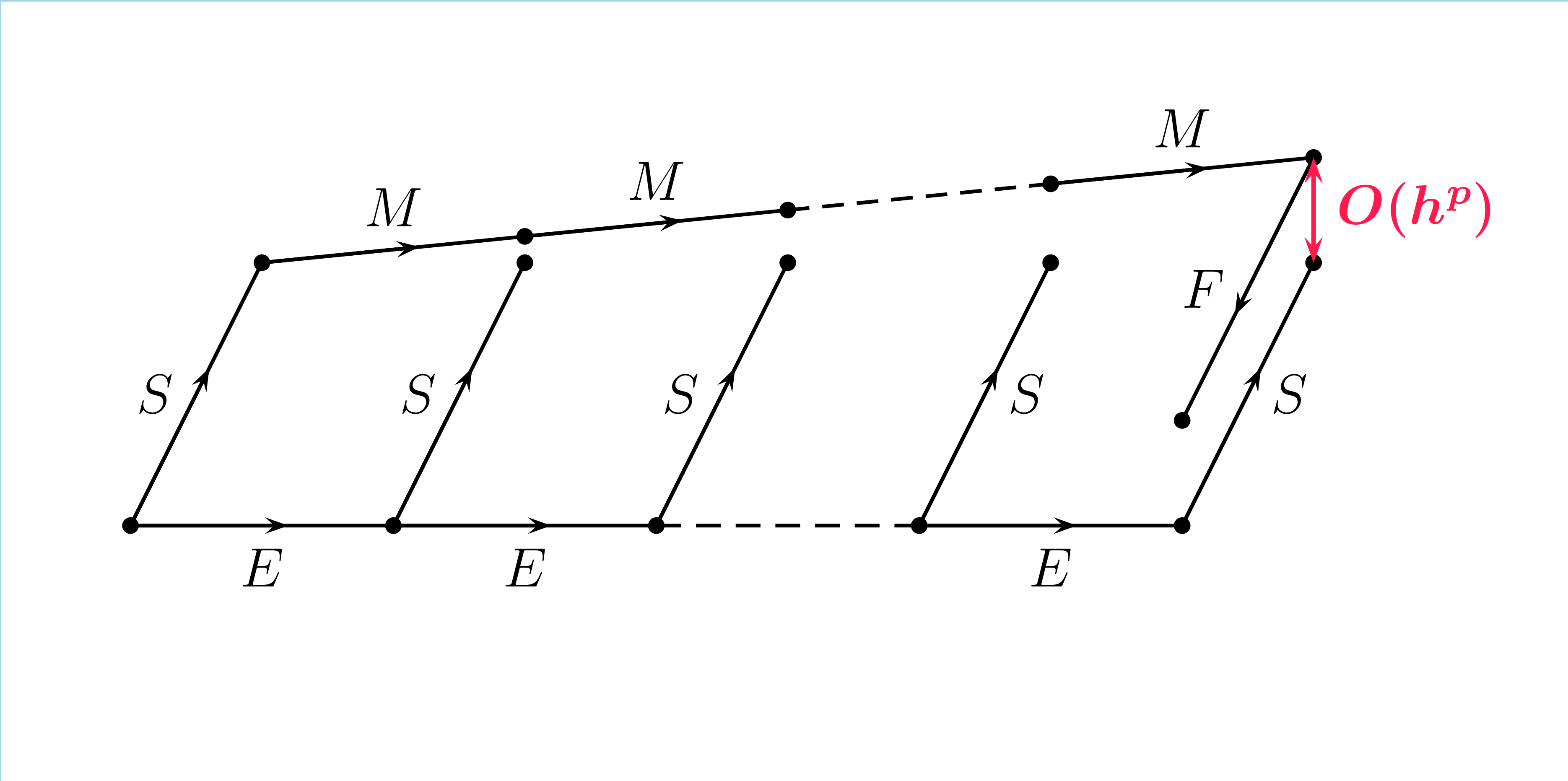
By duplicating this diagram over many steps, global error estimates may be found.



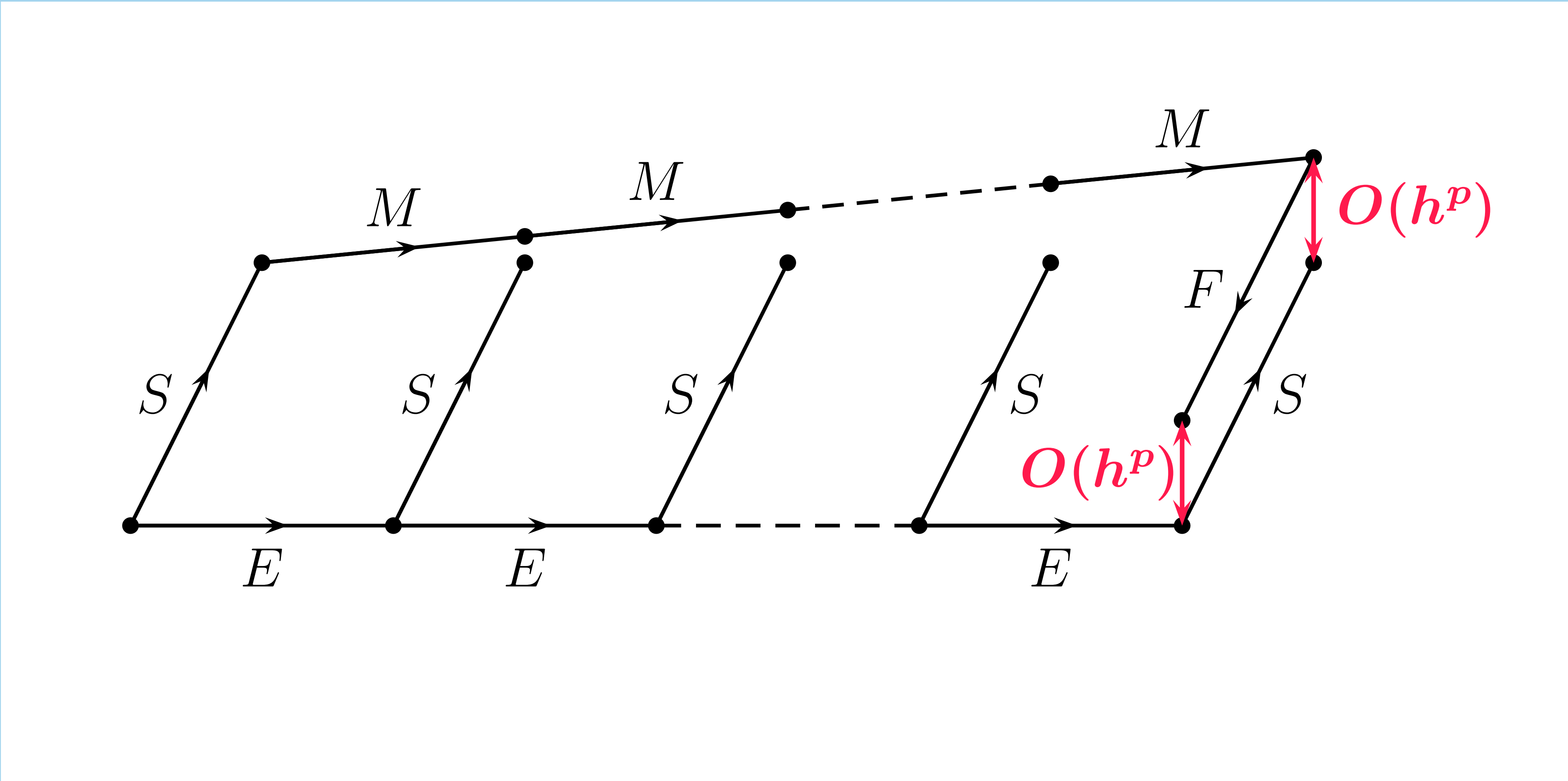
By duplicating this diagram over many steps, global error estimates may be found.



By duplicating this diagram over many steps, global error estimates may be found.



By duplicating this diagram over many steps, global error estimates may be found.



To represent S and turn the definition of order into a practical algorithm for analysing a specific method, operations on the set of mappings $T \to \mathbb{R}$ can be used.

To represent S and turn the definition of order into a practical algorithm for analysing a specific method, operations on the set of mappings $T \rightarrow \mathbb{R}$ can be used. Without considering the details, the conditions are

$$\xi = A\xi D + U\eta$$

$$E\eta = B\xi D + V\eta$$

To represent S and turn the definition of order into a practical algorithm for analysing a specific method, operations on the set of mappings $T \to \mathbb{R}$ can be used. Without considering the details, the conditions are

$$\begin{aligned} \xi &= A\xi D + U\eta \\ E\eta &= B\xi D + V\eta \end{aligned}$$

In the case of Runge–Kutta methods, this definition has the same meaning as “effective order”.

To represent S and turn the definition of order into a practical algorithm for analysing a specific method, operations on the set of mappings $T \to \mathbb{R}$ can be used. Without considering the details, the conditions are

$$\xi = A\xi D + U\eta$$
$$E\eta = B\xi D + V\eta$$

In the case of Runge–Kutta methods, this definition has the same meaning as "effective order".
It is possible for a Runge–Kutta method with 5 stages to have effective order 5.

If we want not only order p but also “stage-order” p (or possibly $p-1$), things become simpler.

If we want not only order p but also “stage-order” p (or possibly $p-1$), things become simpler.

$$\exp(cz) = zA\exp(cz) + U\phi(z) + O(z^{p+1})$$

If we want not only order p but also "stage-order" p (or possibly $p-1$), things become simpler.

$$\exp(cz) = zA\exp(cz) + U\phi(z) + O(z^{p+1})$$

$$\exp(z)\phi(z) = zB\exp(cz) + V\phi(z) + O(z^{p+1})$$

If we want not only order p but also “stage-order” p (or possibly $p-1$), things become simpler.

$$\exp(cz) = zA\exp(cz) + U\phi(z) + O(z^{p+1})$$
$$\exp(z)\phi(z) = zB\exp(cz) + V\phi(z) + O(z^{p+1})$$

where it is assumed the input is

$$y_i^{[n-1]} = \alpha_{i1}y(x_{n-1}) + \alpha_{i2}hy'(x_{n-1}) + \cdots + \alpha_{i,p+1}h^p y^{(p)}(x_{n-1})$$

If we want not only order p but also "stage-order" p (or possibly $p-1$), things become simpler.

$$\exp(cz) = zA\exp(cz) + U\phi(z) + O(z^{p+1})$$
$$\exp(z)\phi(z) = zB\exp(cz) + V\phi(z) + O(z^{p+1})$$

where it is assumed the input is

$$y_i^{[n-1]} = \alpha_{i1}y(x_{n-1}) + \alpha_{i2}hy'(x_{n-1}) + \cdots + \alpha_{i,p+1}h^p y^{(p)}(x_{n-1})$$

and where

$$\phi_i(z) = \alpha_{i1} + \alpha_{i2}z + \cdots + \alpha_{i,p+1}z^p$$

Methods with inherent Runge-Kutta stability

By "Runge-Kutta stability" we mean the property a method might have in which the characteristic polynomial of its stability matrix has all except one of its zeros equal to zero.

Methods with inherent Runge-Kutta stability

By "Runge-Kutta stability" we mean the property a method might have in which the characteristic polynomial of its stability matrix has all except one of its zeros equal to zero.

$$\det(wI - M(z)) = w^{r-1}(w - R(z))$$

Methods with inherent Runge-Kutta stability

By “Runge-Kutta stability” we mean the property a method might have in which the characteristic polynomial of its stability matrix has all except one of its zeros equal to zero.

$$\det(wI - M(z)) = w^{r-1}(w - R(z))$$

Although methods exist with this property with $r = s = p = q$, it is difficult to construct them.

Methods with inherent Runge-Kutta stability

By “Runge-Kutta stability” we mean the property a method might have in which the characteristic polynomial of its stability matrix has all except one of its zeros equal to zero.

$$\det(wI - M(z)) = w^{r-1}(w - R(z))$$

Although methods exist with this property with $r = s = p = q$, it is difficult to construct them.

If $s \geq r = p + 1$, it is possible to construct the methods in a systematic way by imposing a condition known as “Inherent Runge-Kutta Stability”.

Doubly companion matrices

Matrices like the following are “companion matrices” for the polynomial

$$z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n$$

$$\begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{n-1} & -\alpha_n \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

Doubly companion matrices

Matrices like the following are "companion matrices" for the polynomial

$$z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n$$

or

$$z^n + \beta_1 z^{n-1} + \cdots + \beta_n,$$

respectively:

$$\begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{n-1} & -\alpha_n \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -\beta_n \\ 1 & 0 & 0 & \cdots & 0 & -\beta_{n-1} \\ 0 & 1 & 0 & \cdots & 0 & -\beta_{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -\beta_2 \\ 0 & 0 & 0 & \cdots & 1 & -\beta_1 \end{bmatrix}$$

Their characteristic polynomials can be found from $\det(I - zA) = \alpha(z)$ or $\beta(z)$, respectively, where,

$$\alpha(z) = 1+\alpha_1 z+\cdots+\alpha_n z^n, \quad \beta(z) = 1+\beta_1 z+\cdots+\beta_n z^n.$$

Their characteristic polynomials can be found from $\det(I - zA) = \alpha(z)$ or $\beta(z)$, respectively, where,
$\alpha(z) = 1+\alpha_1 z+\cdots+\alpha_n z^n, \quad \beta(z) = 1+\beta_1 z+\cdots+\beta_n z^n.$
A matrix with both α and β terms:

$$X = \begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{n-1} & -\alpha_n - \beta_n \\ 1 & 0 & 0 & \cdots & 0 & -\beta_{n-1} \\ 0 & 1 & 0 & \cdots & 0 & -\beta_{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -\beta_2 \\ 0 & 0 & 0 & \cdots & 1 & -\beta_1 \end{bmatrix},$$

is known as a “doubly companion matrix”

Their characteristic polynomials can be found from $\det(I - zA) = \alpha(z)$ or $\beta(z)$, respectively, where, $\alpha(z) = 1+\alpha_1 z+\cdots+\alpha_n z^n, \quad \beta(z) = 1+\beta_1 z+\cdots+\beta_n z^n.$
A matrix with both α and β terms:

$$X = \begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{n-1} & -\alpha_n - \beta_n \\ 1 & 0 & 0 & \cdots & 0 & -\beta_{n-1} \\ 0 & 1 & 0 & \cdots & 0 & -\beta_{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -\beta_2 \\ 0 & 0 & 0 & \cdots & 1 & -\beta_1 \end{bmatrix},$$

is known as a “doubly companion matrix” and has characteristic polynomial defined by

$$\det(I - zX) = \alpha(z)\beta(z) + O(z^{n+1})$$

Matrices Ψ^{-1} and Ψ transforming X to Jordan canonical form are known.

Matrices Ψ^{-1} and Ψ transforming X to Jordan canonical form are known.

In the special case of a single Jordan block with n-fold eigenvalue λ, we have

$$\Psi^{-1} = \begin{bmatrix} 1 & \lambda+\alpha_1 & \lambda^2+\alpha_1\lambda+\alpha_2 & \cdots \\ 0 & 1 & 2\lambda+\alpha_1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Matrices Ψ^{-1} and Ψ transforming X to Jordan canonical form are known.

In the special case of a single Jordan block with n-fold eigenvalue λ, we have

$$\Psi^{-1} = \begin{bmatrix} 1 & \lambda + \alpha_1 & \lambda^2 + \alpha_1\lambda + \alpha_2 & \cdots \\ 0 & 1 & 2\lambda + \alpha_1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where row number $i+1$ is formed from row number i by differentiating with respect to λ and dividing by i.

Matrices Ψ^{-1} and Ψ transforming X to Jordan canonical form are known.

In the special case of a single Jordan block with n-fold eigenvalue λ, we have

$$\Psi^{-1} = \begin{bmatrix} 1 & \lambda+\alpha_1 & \lambda^2+\alpha_1\lambda+\alpha_2 & \cdots \\ 0 & 1 & 2\lambda+\alpha_1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where row number $i+1$ is formed from row number i by differentiating with respect to λ and dividing by i.

We have a similar expression for Ψ:

$$\Psi = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots \\ \cdots & 1 & 2\lambda + \beta_1 & \lambda^2 + \beta_1\lambda + \beta_2 \\ \cdots & 0 & 1 & \lambda + \beta_1 \\ \cdots & 0 & 0 & 1 \end{bmatrix}$$

$$\Psi = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots \\ \cdots & 1 & 2\lambda+\beta_1 & \lambda^2+\beta_1\lambda+\beta_2 \\ \cdots & 0 & 1 & \lambda+\beta_1 \\ \cdots & 0 & 0 & 1 \end{bmatrix}$$

The Jordan form is $\Psi^{-1}X\Psi = J + \lambda I$, where $J_{ij} = \delta_{i,j+1}$.

$$\Psi = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots \\ \cdots & 1 & 2\lambda + \beta_1 & \lambda^2 + \beta_1\lambda + \beta_2 \\ \cdots & 0 & 1 & \lambda + \beta_1 \\ \cdots & 0 & 0 & 1 \end{bmatrix}$$

The Jordan form is $\Psi^{-1}X\Psi = J + \lambda I$, where $J_{ij} = \delta_{i,j+1}$. That is

$$\Psi^{-1}X\Psi = \begin{bmatrix} \lambda & 0 & \cdots & 0 & 0 \\ 1 & \lambda & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda & 0 \\ 0 & 0 & \cdots & 1 & \lambda \end{bmatrix}$$

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations.

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess “Inherent Runge–Kutta Stability”.

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess “Inherent Runge–Kutta Stability”. Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess "Inherent Runge–Kutta Stability". Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

- λ single eigenvalue of A

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess "Inherent Runge–Kutta Stability". Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

- λ single eigenvalue of A
- $c_1, c_2, \ldots, c_s$ stage abscissae

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess "Inherent Runge–Kutta Stability". Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

- λ single eigenvalue of A
- $c_1, c_2, \ldots, c_s$ stage abscissae
- Error constant

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess “Inherent Runge–Kutta Stability”. Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

- λ single eigenvalue of A
- $c_1, c_2, \ldots, c_s$ stage abscissae
- Error constant
- $\beta_1, \beta_2, \ldots, \beta_p$ elements in last column of $s \times s$ doubly companion matrix X

Inherent Runge-Kutta stability

Using doubly companion matrices, it is possible to construct GL methods possessing RK stability with rational operations. The methods constructed in this way are said to possess “Inherent Runge–Kutta Stability”. Apart from exceptional cases, (in which certain matrices are singular), we characterize the method with $r = s = p + 1 = q + 1$ by the parameters

- λ single eigenvalue of A
- $c_1, c_2, \dots, c_s$ stage abscissae
- Error constant
- $\beta_1, \beta_2, \dots, \beta_p$ elements in last column of $s \times s$ doubly companion matrix X
- Information on the structure of V

Consider only methods for which the step n outputs approximate the "Nordsieck vector"

Consider only methods for which the step n outputs approximate the "Nordsieck vector":

$$\begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ y_3^{[n]} \\ \vdots \\ y_{p+1}^{[n]} \end{bmatrix} \approx \begin{bmatrix} y(x_n) \\ hy'(x_n) \\ h^2y''(x_n) \\ \vdots \\ h^py^{(p)}(x_n) \end{bmatrix}$$

Consider only methods for which the step n outputs approximate the "Nordsieck vector":

$$\begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ y_3^{[n]} \\ \vdots \\ y_{p+1}^{[n]} \end{bmatrix} \approx \begin{bmatrix} y(x_n) \\ hy'(x_n) \\ h^2y''(x_n) \\ \vdots \\ h^py^{(p)}(x_n) \end{bmatrix}$$

For such methods, V has the form

$$V = \begin{bmatrix} 1 & v^T \\ 0 & \dot{V} \end{bmatrix}$$

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB,$$

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T,$$

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T, \quad \rho(\dot{V}) = 0$$

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T, \quad \rho(\dot{V}) = 0$$

It can be shown that, for such methods, the stability matrix satisfies

$$M(z) \sim V + ze_1\xi^T(I - zX)^{-1}$$

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T, \quad \rho(\dot{V}) = 0$$

It can be shown that, for such methods, the stability matrix satisfies

$$M(z) \sim V + ze_1\xi^T(I - zX)^{-1}$$

which has all except one of its eigenvalues zero.

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T, \quad \rho(\dot{V}) = 0$$

It can be shown that, for such methods, the stability matrix satisfies

$$M(z) \sim V + ze_1\xi^T(I - zX)^{-1}$$

which has all except one of its eigenvalues zero. The non-zero eigenvalue has the role of stability function

Such a method has the IRKS property if a doubly companion matrix X exists so that for some vector ξ,

$$BA = XB, \quad BU = XV - VX + e_1\xi^T, \quad \rho(\dot{V}) = 0$$

It can be shown that, for such methods, the stability matrix satisfies

$$M(z) \sim V + ze_1\xi^T(I - zX)^{-1}$$

which has all except one of its eigenvalues zero. The non-zero eigenvalue has the role of stability function

$$R(z) = \frac{N(z)}{(1-\lambda z)^s}$$

The following third order method is explicit and suitable for the solution of non-stiff problems

$$
\begin{bmatrix} AU \\ BV \end{bmatrix} = \left[\begin{array}{cccc|cccc}
0 & 0 & 0 & 0 & 1 & \frac{1}{4} & \frac{1}{32} & \frac{1}{384} \\
-\frac{176}{1885} & 0 & 0 & 0 & 1 & \frac{2237}{3770} & \frac{2237}{15080} & \frac{2149}{90480} \\
-\frac{335624}{311025} & \frac{29}{55} & 0 & 0 & 1 & \frac{1619591}{1244100} & \frac{260027}{904800} & \frac{1517801}{39811200} \\
-\frac{67843}{6435} & \frac{395}{33} & -5 & 0 & 1 & \frac{29428}{6435} & \frac{527}{585} & \frac{41819}{102960} \\
\hline
-\frac{67843}{6435} & \frac{395}{33} & -5 & 0 & 1 & \frac{29428}{6435} & \frac{527}{585} & \frac{41819}{102960} \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\frac{82}{33} & -\frac{274}{11} & \frac{170}{9} & -\frac{4}{3} & 0 & \frac{482}{99} & 0 & -\frac{161}{264} \\
-8 & -12 & \frac{40}{3} & -2 & 0 & \frac{26}{3} & 0 & 0
\end{array}\right]
$$

The following fourth order method is implicit, L-stable, and suitable for the solution of stiff problems

$$
\left[\begin{array}{ccccc|ccccc}
\frac{1}{4} & 0 & 0 & 0 & 0 & 1 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\
-\frac{513}{54272} & \frac{1}{4} & 0 & 0 & 0 & 1 & \frac{27649}{54272} & \frac{5601}{27136} & \frac{1539}{54272} & -\frac{459}{6784} \\
\frac{3706119}{69088256} & -\frac{488}{3819} & \frac{1}{4} & 0 & 0 & 1 & \frac{15366379}{207264768} & \frac{756057}{34544128} & \frac{1620299}{69088256} & -\frac{4854}{454528} \\
\frac{32161061}{197549232} & -\frac{111814}{232959} & \frac{134}{183} & \frac{1}{4} & 0 & 1 & -\frac{32609017}{197549232} & \frac{929753}{32924872} & \frac{4008881}{32924872} & \frac{174981}{3465776} \\
-\frac{135425}{2948496} & -\frac{641}{10431} & \frac{73}{183} & \frac{1}{2} & \frac{1}{4} & 1 & -\frac{367313}{8845488} & -\frac{22727}{1474248} & \frac{40979}{982832} & \frac{323}{25864} \\
\hline
-\frac{135425}{2948496} & -\frac{641}{10431} & \frac{73}{183} & \frac{1}{2} & \frac{1}{4} & 1 & -\frac{367313}{8845488} & -\frac{22727}{1474248} & \frac{40979}{982832} & \frac{323}{25864} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\frac{2255}{2318} & -\frac{47125}{20862} & \frac{447}{122} & -\frac{11}{4} & \frac{4}{3} & 0 & -\frac{28745}{20862} & -\frac{1937}{13908} & \frac{351}{18544} & \frac{65}{976} \\
\frac{12620}{10431} & -\frac{96388}{31293} & \frac{3364}{549} & -\frac{10}{3} & \frac{4}{3} & 0 & -\frac{70634}{31293} & -\frac{2050}{10431} & -\frac{187}{2318} & \frac{113}{366} \\
\frac{414}{1159} & -\frac{29954}{31293} & \frac{130}{61} & -1 & \frac{1}{3} & 0 & -\frac{27052}{31293} & -\frac{113}{10431} & -\frac{491}{4636} & \frac{161}{732}
\end{array}\right]
$$

References

Selected references on general linear methods

J. C. Butcher (1966) ‘On the convergence of numerical solutions of ordinary differential equations’, *Math. Comp.* **20** 1–10.

J. C. Butcher (1973) ‘The order of numerical methods for ordinary differential equations’, *Math. Comp.* **27** 793–806.

J. C. Butcher and Z. Jackiewicz (2002) ‘Error estimation for Nordsieck methods’, *Numer. Algorithms*, **31** 75–85.

J. C. Butcher and W. M. Wright (2003) ‘The construction of practical general linear methods’, *BIT* **43** 695–721.

W. M. Wright (2002) ‘Explicit general linear methods with inherent Runge–Kutta stability’, *Numer. Algorithms* **31** 381–399.

URL for this talk:

`http://www.math.auckland.ac.nz/`
`~butcher/conferences.html`