Scientific Computation and Differential Equations

John Butcher

The University of Auckland
New Zealand

Annual Fundamental Sciences Seminar
June 2006

Institut Kajian Sains Fundamental Ibnu Sina

Overview

This year marks the fiftieth anniversary of the first computer in the Southern Hemisphere – the SILLIAC computer at the University of Sydney.

Overview

This year marks the fiftieth anniversary of the first computer in the Southern Hemisphere – the SILLIAC computer at the University of Sydney.

I had the privilege of being present when this computer did its first calculation and my own first program ran on this computer soon afterwards.

Overview

This year marks the fiftieth anniversary of the first computer in the Southern Hemisphere – the SILLIAC computer at the University of Sydney.

I had the privilege of being present when this computer did its first calculation and my own first program ran on this computer soon afterwards.

This computer was built for one reason only – to solve scientific problems.

Overview

This year marks the fiftieth anniversary of the first computer in the Southern Hemisphere – the SILLIAC computer at the University of Sydney.

I had the privilege of being present when this computer did its first calculation and my own first program ran on this computer soon afterwards.

This computer was built for one reason only – to solve scientific problems.

Scientific problems have always been the driving force in the development of faster and faster computers.

Overview

This year marks the fiftieth anniversary of the first computer in the Southern Hemisphere – the SILLIAC computer at the University of Sydney.

I had the privilege of being present when this computer did its first calculation and my own first program ran on this computer soon afterwards.

This computer was built for one reason only – to solve scientific problems.

Scientific problems have always been the driving force in the development of faster and faster computers.

Within Scientific Computation, the approximate solution of differential equations has always been an area of special challenge.

Differential equations can usually not be solved analytically and numerical methods are necessary.

Differential equations can usually not be solved analytically and numerical methods are necessary.

Two main types of numerical methods exist: linear multistep methods and Runge–Kutta methods.

Differential equations can usually not be solved analytically and numerical methods are necessary.

Two main types of numerical methods exist: linear multistep methods and Runge–Kutta methods.

These traditional methods are special cases within the larger class of "General Linear Methods".

Differential equations can usually not be solved analytically and numerical methods are necessary.

Two main types of numerical methods exist: linear multistep methods and Runge–Kutta methods.

These traditional methods are special cases within the larger class of "General Linear Methods".

Today we will look briefly at the history of numerical methods for differential equations.

Differential equations can usually not be solved analytically and numerical methods are necessary.

Two main types of numerical methods exist: linear multistep methods and Runge–Kutta methods.

These traditional methods are special cases within the larger class of “General Linear Methods”.

Today we will look briefly at the history of numerical methods for differential equations.

We will then look at some particular questions concerning the theory of general linear methods.

Differential equations can usually not be solved analytically and numerical methods are necessary.

Two main types of numerical methods exist: linear multistep methods and Runge–Kutta methods.

These traditional methods are special cases within the larger class of “General Linear Methods”.

Today we will look briefly at the history of numerical methods for differential equations.

We will then look at some particular questions concerning the theory of general linear methods.

We will also look at some aspects of their practical implementation.

Contents

- A short history of numerical differential equations

Contents

- A short history of numerical differential equations
- Linear multistep methods

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods
- General linear methods

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods
- General linear methods
- Examples of general linear methods

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods
- General linear methods
- Examples of general linear methods
- Order of GLMs

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods
- General linear methods
- Examples of general linear methods
- Order of GLMs
- Methods with the IRK stability property

Contents

- A short history of numerical differential equations
- Linear multistep methods
- Runge–Kutta methods
- General linear methods
- Examples of general linear methods
- Order of GLMs
- Methods with the IRK stability property
- Implementation questions for IRKS methods

A short history of numerical ODEs

We will make use of three standard types of initial value problems

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}, \tag{1}$$

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N, \tag{2}$$

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N. \tag{3}$$

A short history of numerical ODEs

We will make use of three standard types of initial value problems

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}, \tag{1}$$

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N, \tag{2}$$

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N. \tag{3}$$

Problem (1) is used in traditional descriptions of numerical methods but in applications we need to use either (2) or (3).

A short history of numerical ODEs

We will make use of three standard types of initial value problems

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}, \tag{1}$$

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N, \tag{2}$$

$$y'(x) = f(y(x)), \qquad y(x_0) = y_0 \in \mathbb{R}^N. \tag{3}$$

Problem (1) is used in traditional descriptions of numerical methods but in applications we need to use either (2) or (3).

These are actually equivalent and we will often use (3) instead of (2) because of its simplicity.

The Euler method

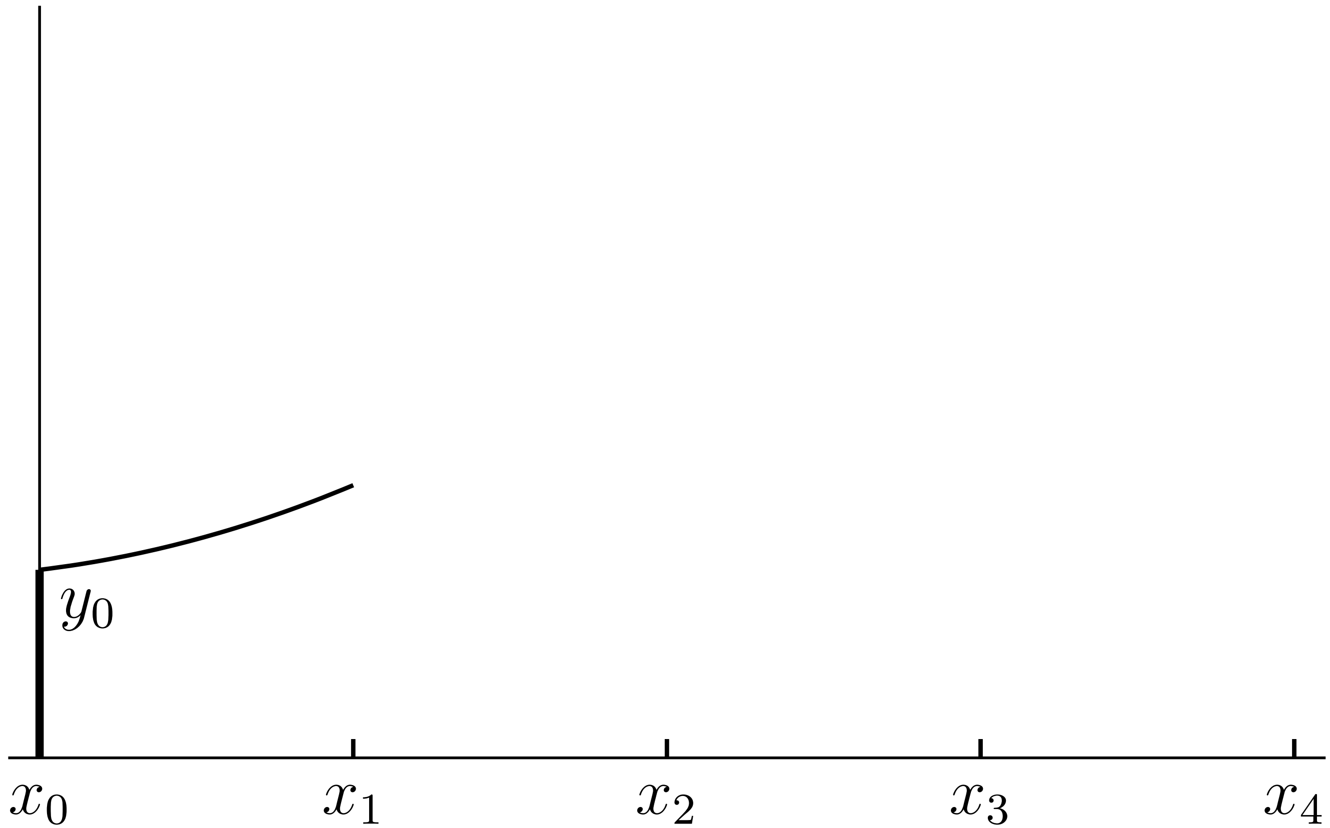
Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations.

The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.

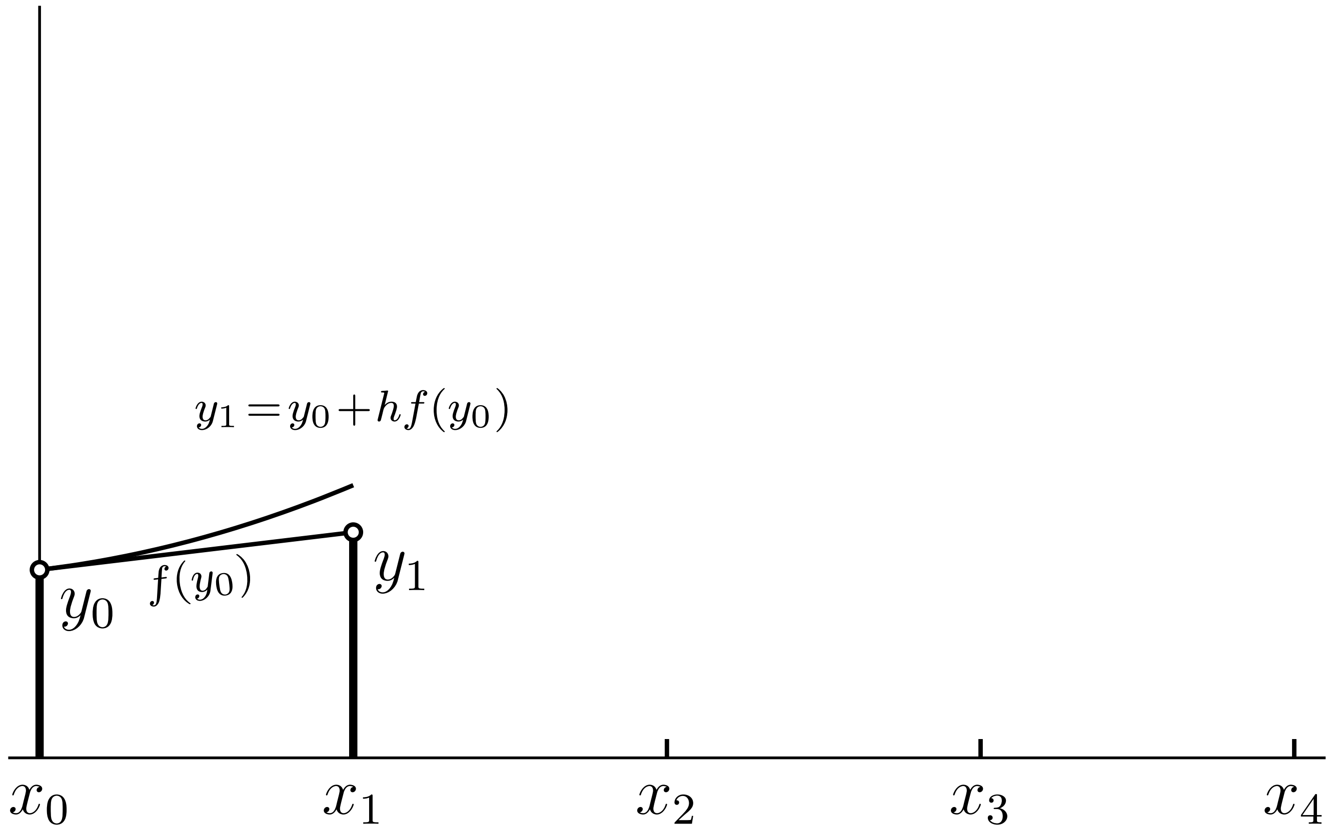
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



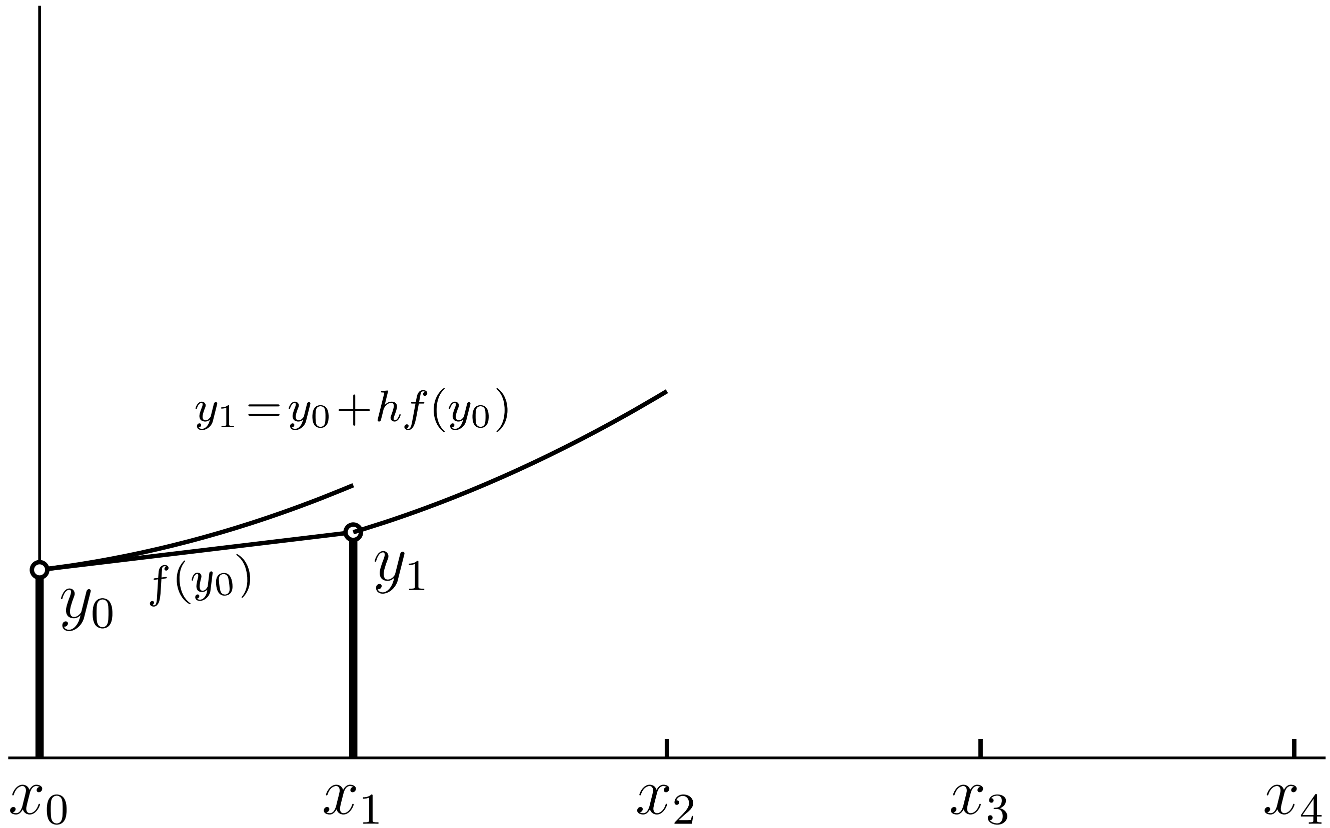
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



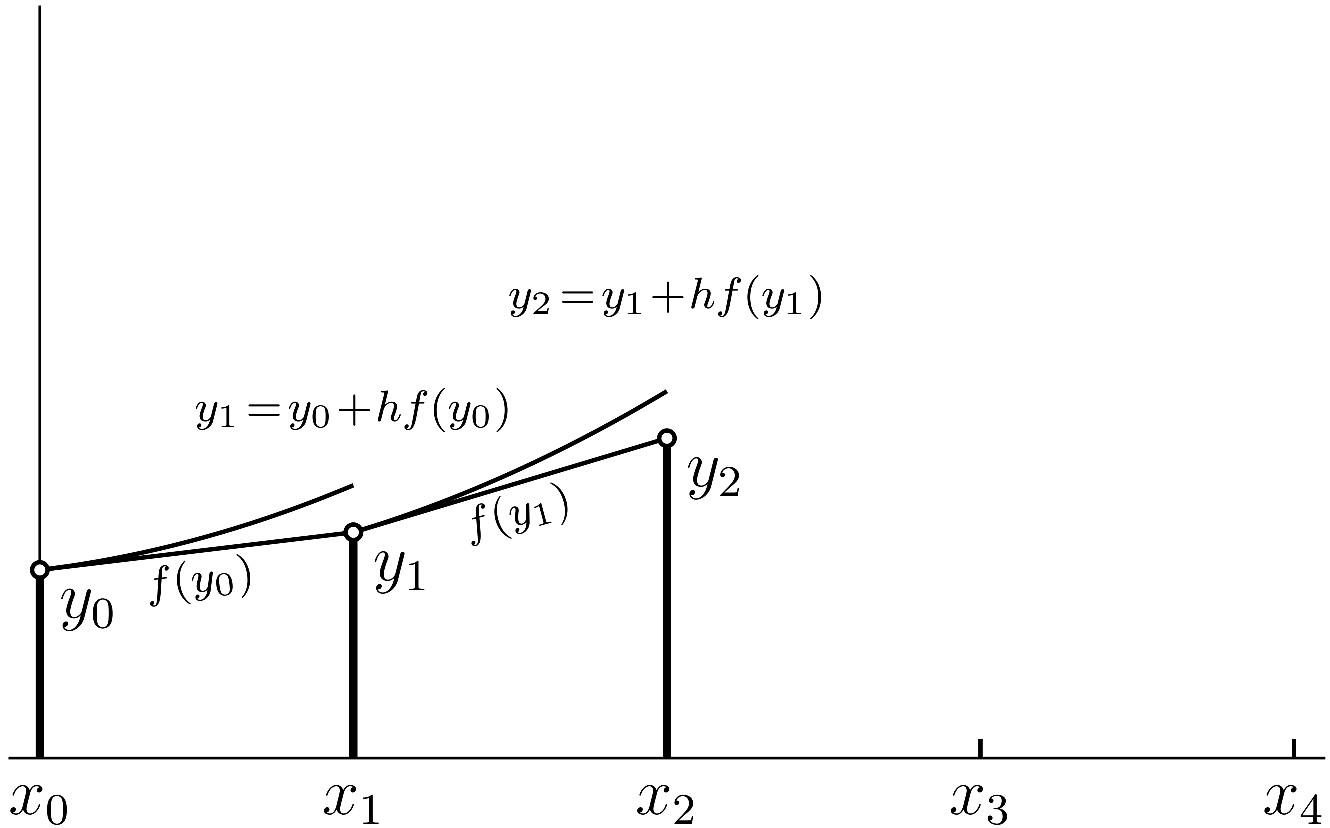
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



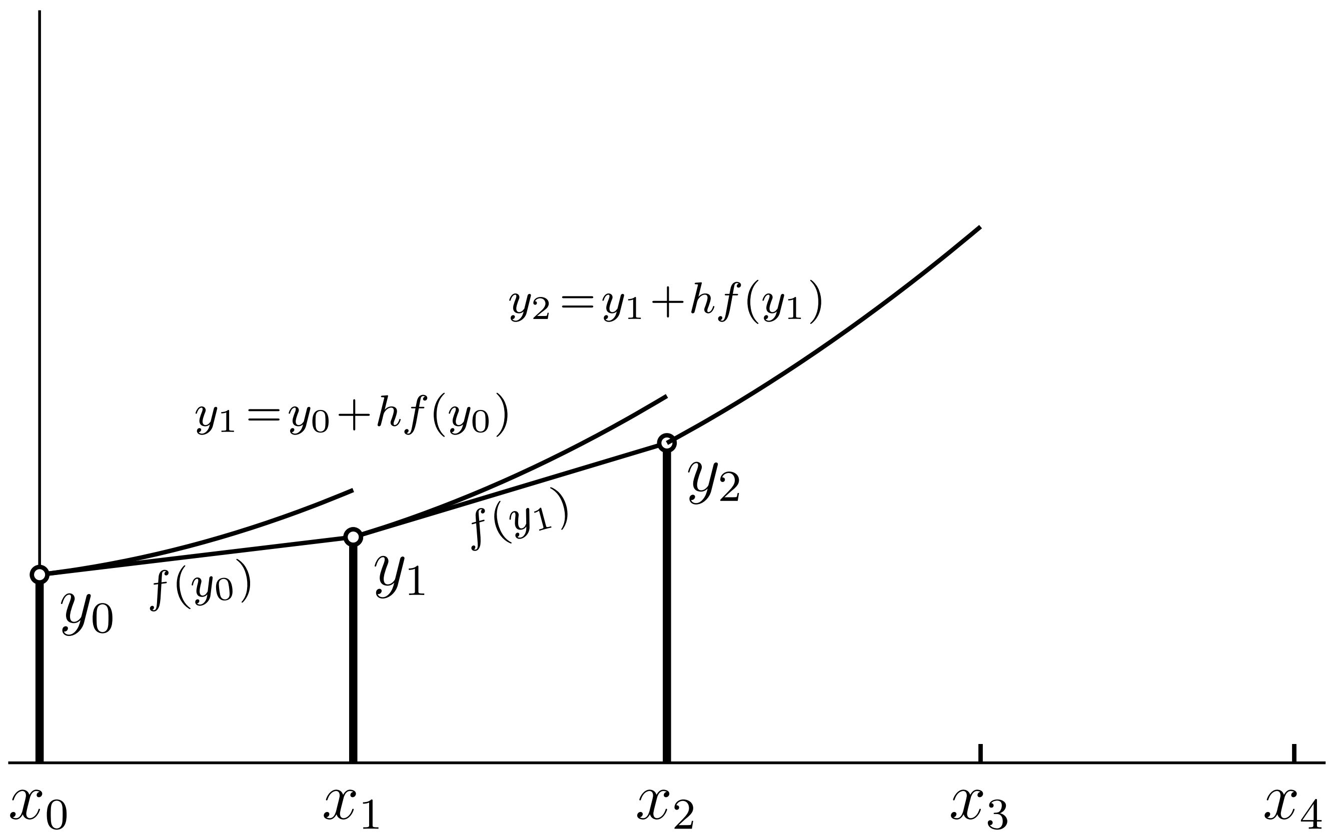
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



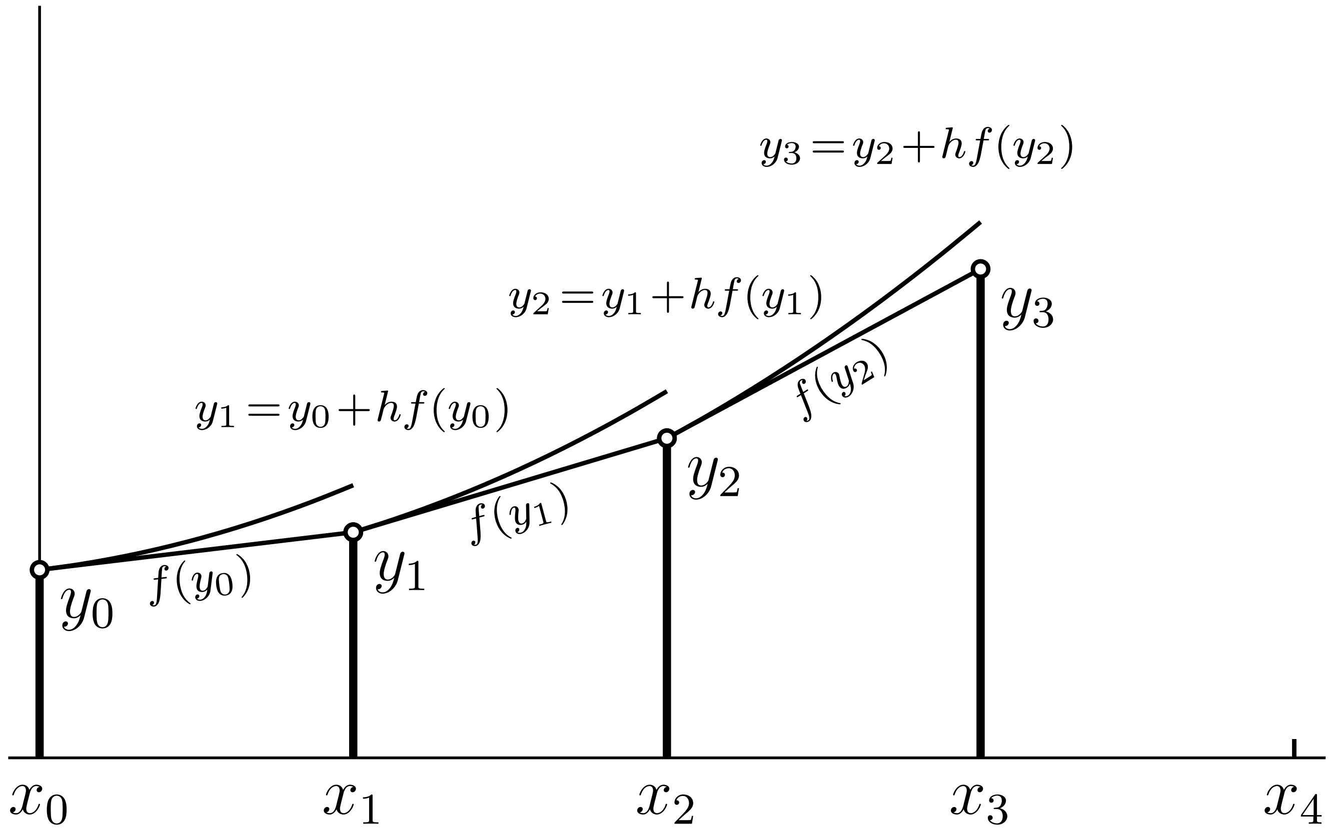
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



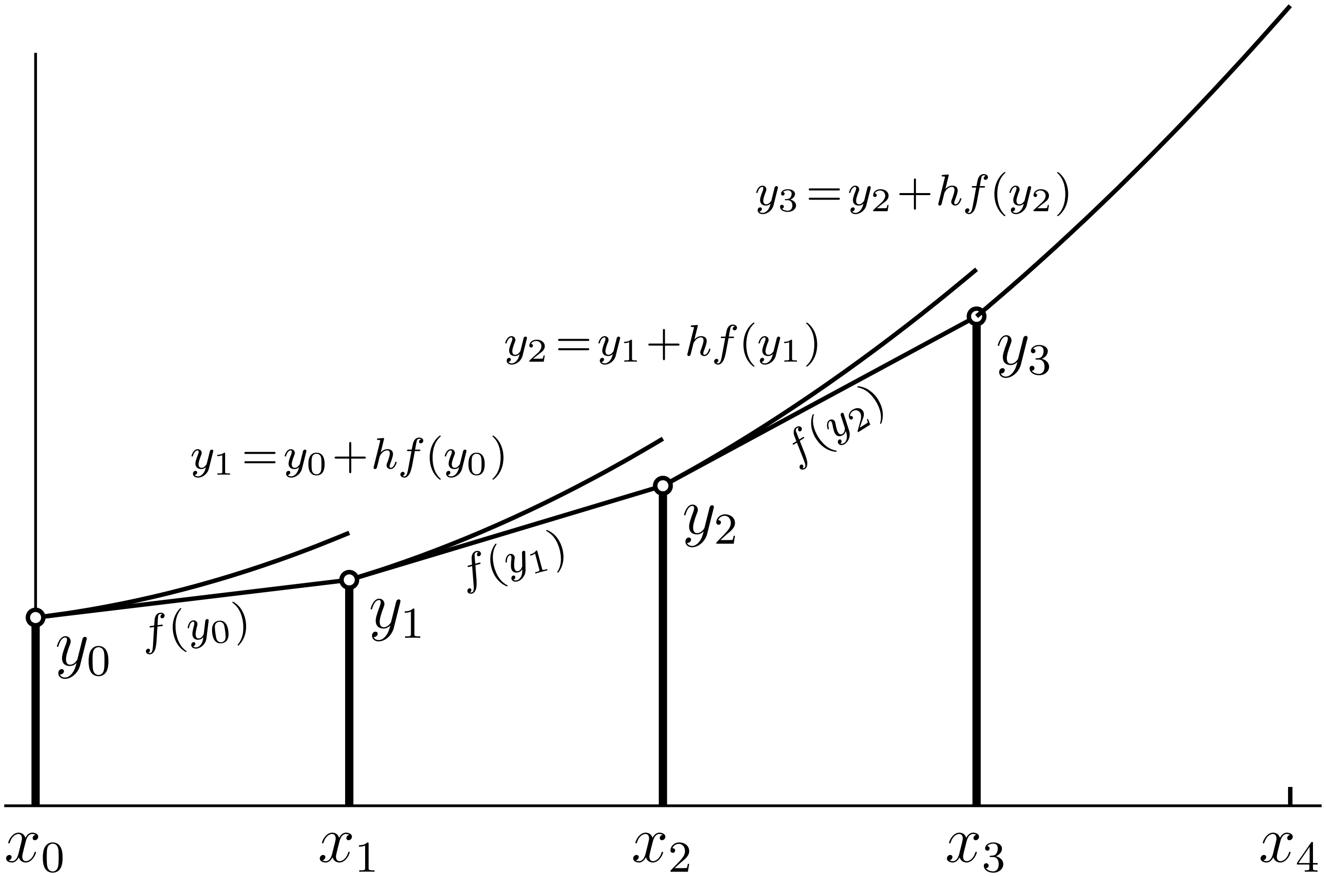
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



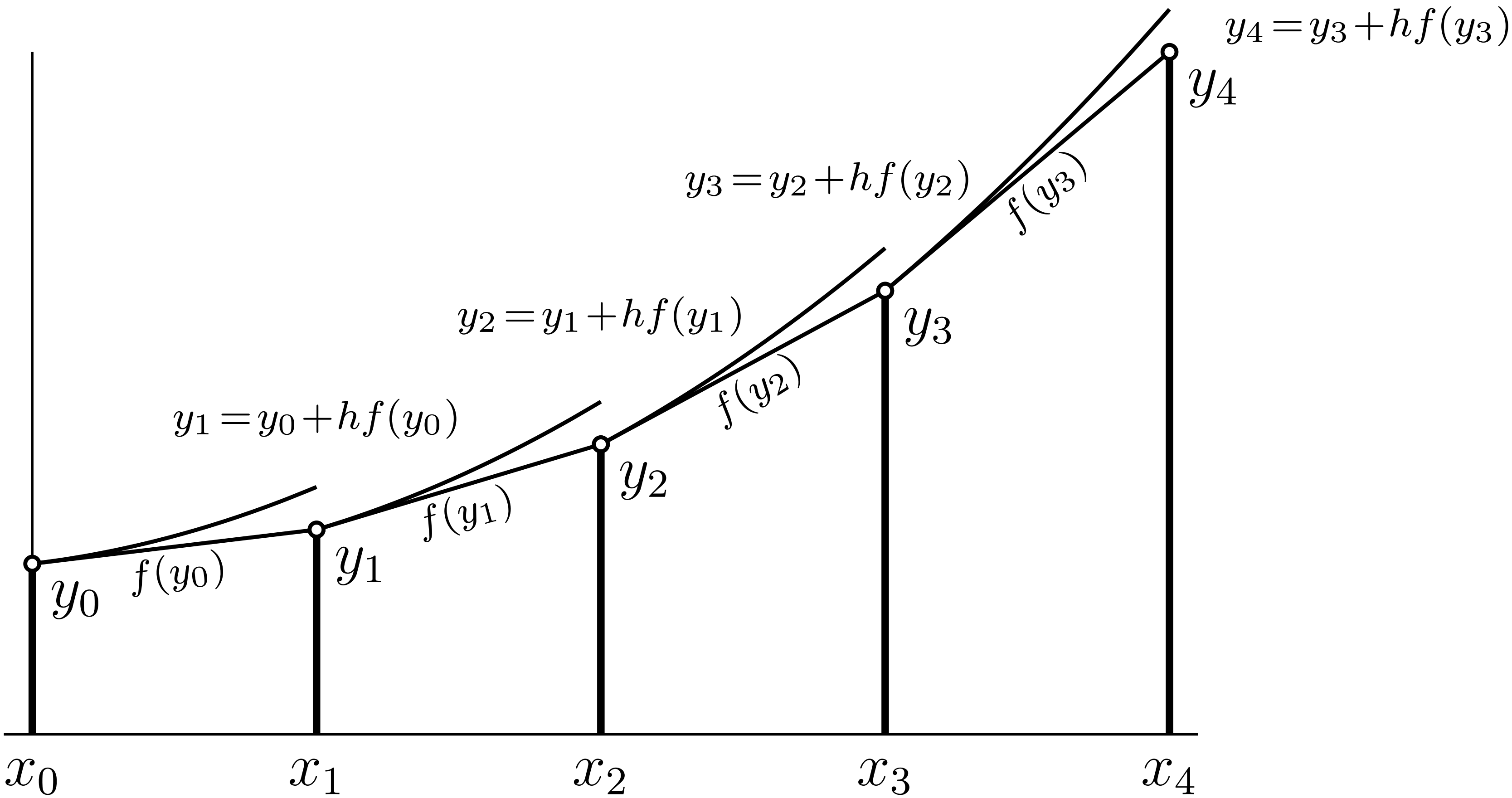
The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



The Euler method

Euler proposed a simple numerical scheme in approximately 1770; this can be used for a system of first order equations. The idea is to treat the solution as though it had constant derivative in each time step.



More modern methods attempt to improve on the Euler method by

More modern methods attempt to improve on the Euler method by

1. Using more past history

More modern methods attempt to improve on the Euler method by

1. Using more past history
   – *Linear multistep methods*

More modern methods attempt to improve on the Euler method by

1. Using more past history
   - *Linear multistep methods*
2. Doing more complicated calculations in each step

More modern methods attempt to improve on the Euler method by

1. Using more past history
   – *Linear multistep methods*
2. Doing more complicated calculations in each step
   – *Runge–Kutta methods*

More modern methods attempt to improve on the Euler method by

1. Using more past history

   – *Linear multistep methods*

2. Doing more complicated calculations in each step

   – *Runge–Kutta methods*

3. Doing both of these

More modern methods attempt to improve on the Euler method by

1. Using more past history
   – *Linear multistep methods*
2. Doing more complicated calculations in each step
   – *Runge–Kutta methods*
3. Doing both of these
   – *General linear methods*

Some important dates

1883	Adams & Bashforth	Linear multistep methods
1895	Runge	Runge-Kutta method
1901	Kutta	
1925	Nyström	Special methods for second order
1926	Moulton	Adams-Moulton method
1952	Curtiss & Hirschfelder	Stiff problems

Linear multistep methods

We will write the differential equation in autonomous form

$$y'(x) = f(y(x)), \quad y(x_0) = y_0,$$

Linear multistep methods

We will write the differential equation in autonomous form

$$y'(x) = f(y(x)), \quad y(x_0) = y_0,$$

and the aim, for the moment, will be to calculate approximations to $y(x_i)$, where

$$x_i = x_0 + hi, \quad i = 1, 2, 3, \dots,$$

and h is the "stepsize".

Linear multistep methods

We will write the differential equation in autonomous form

$$y'(x) = f(y(x)), \quad y(x_0) = y_0,$$

and the aim, for the moment, will be to calculate approximations to $y(x_i)$, where

$$x_i = x_0 + hi, \quad i = 1, 2, 3, \ldots,$$

and h is the “stepsize”.

Linear multistep methods base the approximation to $y(x_n)$ on a linear combination of approximations to $y(x_{n-i})$ and approximations to $y'(x_{n-i})$, $i = 1, 2, \ldots, k$.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

This is a 1-stage $2k$-value method.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

This is a 1-stage $2k$-value method.

1 stage? One evaluation of f per step.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

This is a 1-stage $2k$-value method.

1 stage? One evaluation of f per step.

$2k$ value? This many quantities are passed between steps.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

This is a 1-stage $2k$-value method.

1 stage? One evaluation of f per step.

$2k$ value? This many quantities are passed between steps.

$\beta_0 = 0$: explicit.

Write y_i as the approximation to $y(x_i)$ and f_i as the approximation to $y'(x_i) = f(y(x_i))$.

A linear multistep method can be written as

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i}$$

This is a 1-stage $2k$-value method.

1 stage? One evaluation of f per step.

$2k$ value? This many quantities are passed between steps.

$\beta_0 = 0$: explicit. $\beta_0 \neq 0$: implicit.

Runge–Kutta methods

A Runge–Kutta method computes y_n in terms of a single input y_{n-1} and s stages $Y_1, Y_2, \ldots, Y_s$,

Runge–Kutta methods

A Runge–Kutta method computes y_n in terms of a single input y_{n-1} and s stages Y_1, Y_2, $\ldots$, Y_s, where

$$Y_i = y_{n-1} + h \sum_{j=1}^{s} a_{ij} f(Y_j), \qquad i = 1, 2, \ldots, s,$$

Runge–Kutta methods

A Runge–Kutta method computes y_n in terms of a single input y_{n-1} and s stages Y_1, Y_2, $\ldots$, Y_s, where

$$Y_i = y_{n-1} + h \sum_{j=1}^{s} a_{ij} f(Y_j), \qquad i = 1, 2, \ldots, s,$$

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

Runge–Kutta methods

A Runge–Kutta method computes y_n in terms of a single input y_{n-1} and s stages $Y_1, Y_2, \ldots, Y_s$, where

$$Y_i = y_{n-1} + h \sum_{j=1}^{s} a_{ij} f(Y_j), \qquad i = 1, 2, \ldots, s,$$

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

This is an s-stage 1-value method.

Runge–Kutta methods

A Runge–Kutta method computes y_n in terms of a single input y_{n-1} and s stages $Y_1, Y_2, \dots, Y_s$, where

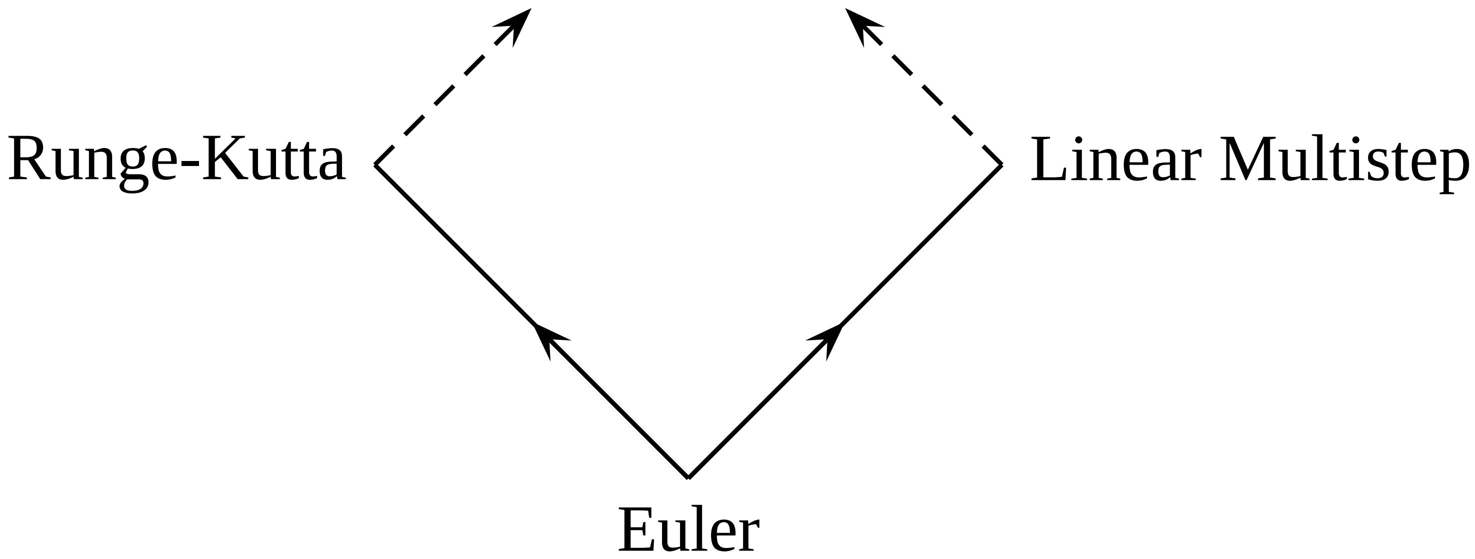
$$Y_i = y_{n-1} + h \sum_{j=1}^{s} a_{ij} f(Y_j), \qquad i = 1, 2, \dots, s,$$

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

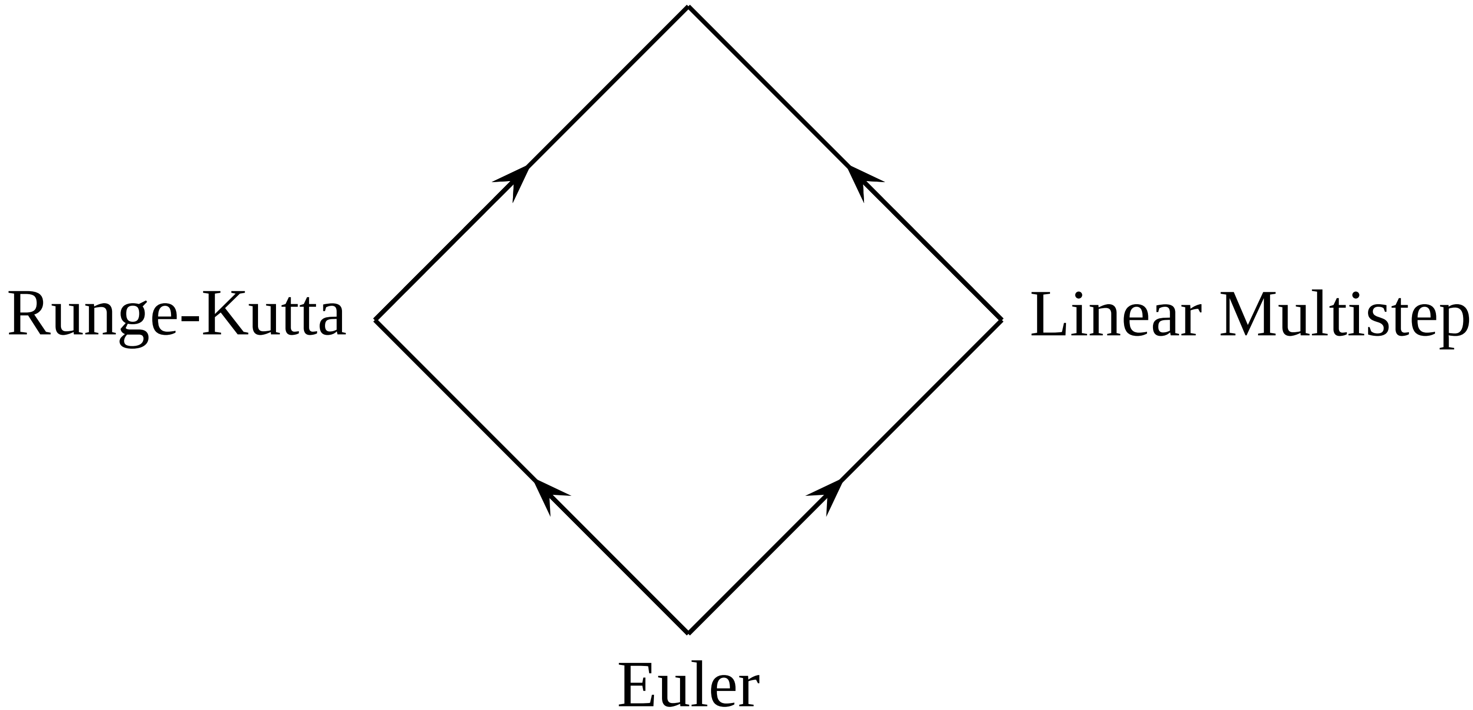
This is an s-stage 1-value method.

It is natural to ask if there are useful methods which are multistage (as for Runge–Kutta methods) and multivalue (as for linear multistep methods).

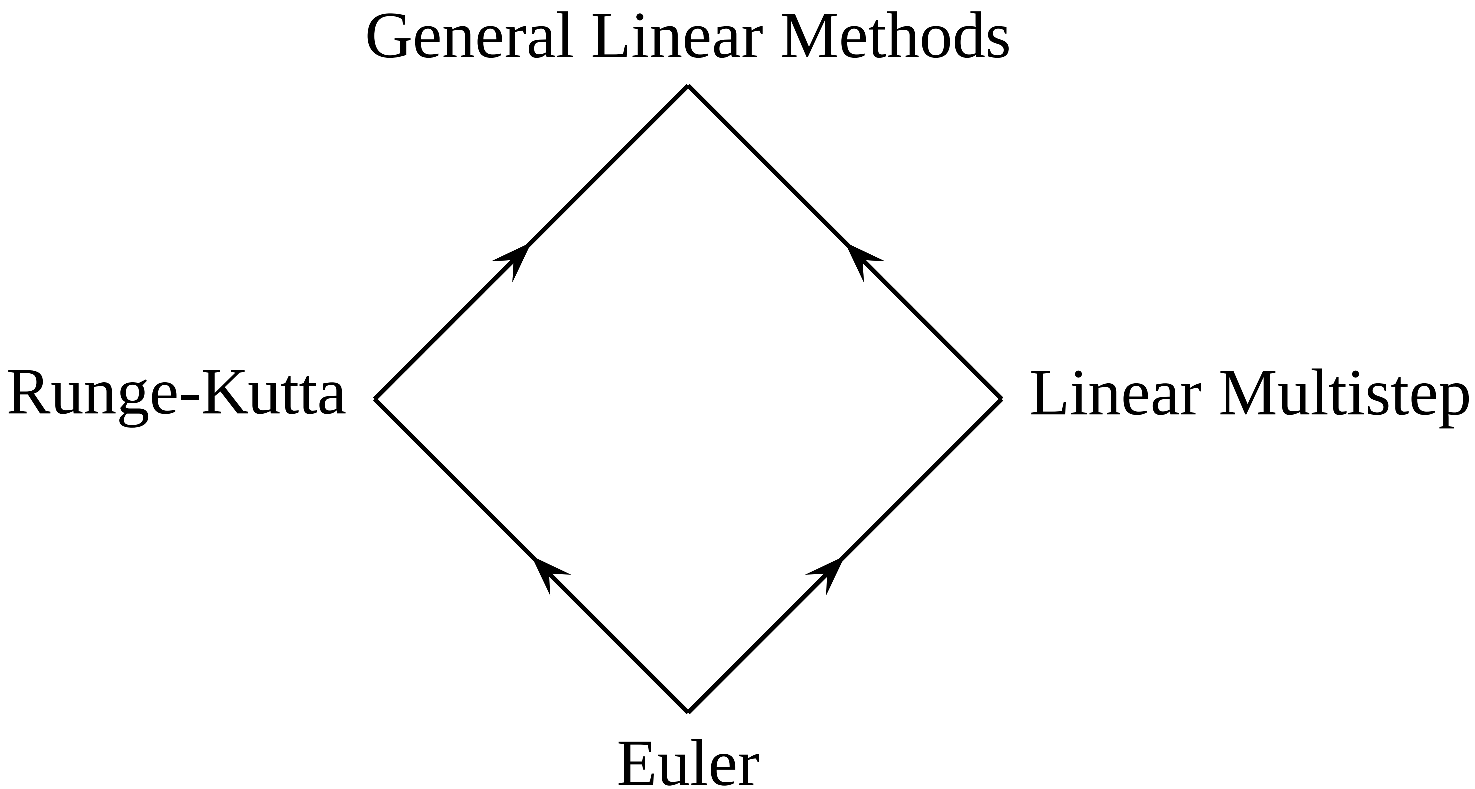
In other words, we ask if there is any value in completing this diagram:



In other words, we ask if there is any value in completing this diagram:



In other words, we ask if there is any value in completing this diagram:



General linear methods

We will consider methods characterised by an $(s+r)\times(s+r)$ partitioned matrix of the form

$$\begin{array}{cc} & \begin{array}{cc} \overset{s}{\longleftrightarrow} & \overset{r}{\longleftrightarrow} \end{array} \\ \begin{array}{c} s \updownarrow \\ r \updownarrow \end{array} & \left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] \end{array}.$$

General linear methods

We will consider methods characterised by an $(s+r)\times(s+r)$ partitioned matrix of the form

$$\begin{array}{c} \\ s \\ r \end{array}\!\!\begin{array}{c} \;\;s \quad\;\; r \\ \left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] \end{array}.$$

The r values input to step $n-1$ will be denoted by $y_i^{[n-1]}$, $i=1,2,\dots,r$ with corresponding output values $y_i^{[n]}$ and the stage values by Y_i, $i=1,2,\dots,s$.

General linear methods

We will consider methods characterised by an $(s+r)\times(s+r)$ partitioned matrix of the form

$$\begin{array}{c} \\ s \\ r \end{array}\begin{array}{c} \begin{array}{cc} \;\; s & \;\; r \end{array} \\ \left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] \end{array}.$$

The r values input to step $n-1$ will be denoted by $y_i^{[n-1]}$, $i=1,2,\dots,r$ with corresponding output values $y_i^{[n]}$ and the stage values by Y_i, $i=1,2,\dots,s$.

The stage derivatives will be denoted by $F_i=f(Y_i)$.

The formula for computing the stages (and simultaneously the stage derivatives) are:

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

for $i = 1, 2, \ldots, s$.

The formula for computing the stages (and simultaneously the stage derivatives) are:

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

for $i = 1, 2, \ldots, s$.

To compute the output values, use the formula

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

For convenience, write

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

For convenience, write

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

so that we can write the calculations in a step more simply as

$$\begin{bmatrix} Y \\ y^{[n]} \end{bmatrix} = \begin{bmatrix} A & U \\ B & V \end{bmatrix} \begin{bmatrix} hF \\ y^{[n-1]} \end{bmatrix}.$$

Examples of general linear methods

We will look at five examples

Examples of general linear methods

We will look at five examples

- A Runge–Kutta method

Examples of general linear methods

We will look at five examples

- A Runge–Kutta method
- A "re-use" method

Examples of general linear methods

We will look at five examples

- A Runge–Kutta method
- A “re-use” method
- An Almost Runge–Kutta method

Examples of general linear methods

We will look at five examples

- A Runge–Kutta method
- A “re-use” method
- An Almost Runge–Kutta method
- An Adams-Bashforth/Adams-Moulton method

Examples of general linear methods

We will look at five examples

- A Runge–Kutta method
- A “re-use” method
- An Almost Runge–Kutta method
- An Adams-Bashforth/Adams-Moulton method
- A modified linear multistep method

A Runge–Kutta method

One of the famous families of fourth order methods of Kutta, written as a general linear method, is

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \theta & 0 & 0 & 0 & 1 \\ \frac{1}{2} - \frac{1}{8\theta} & \frac{1}{8\theta} & 0 & 0 & 1 \\ \frac{1}{2\theta} - 1 & -\frac{1}{2\theta} & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

A Runge–Kutta method

One of the famous families of fourth order methods of Kutta, written as a general linear method, is

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \theta & 0 & 0 & 0 & 1 \\ \frac{1}{2}-\frac{1}{8\theta} & \frac{1}{8\theta} & 0 & 0 & 1 \\ \frac{1}{2\theta}-1 & -\frac{1}{2\theta} & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

In a step from x_{n-1} to $x_n = x_{n-1} + h$, the stages give approximations at

$$x_{n-1}, \quad x_{n-1} + \theta h, \quad x_{n-1} + \tfrac{1}{2}h \quad \text{and} \quad x_{n-1} + h.$$

A Runge–Kutta method

One of the famous families of fourth order methods of Kutta, written as a general linear method, is

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \theta & 0 & 0 & 0 & 1 \\ \frac{1}{2} - \frac{1}{8\theta} & \frac{1}{8\theta} & 0 & 0 & 1 \\ \frac{1}{2\theta} - 1 & -\frac{1}{2\theta} & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

In a step from x_{n-1} to $x_n = x_{n-1} + h$, the stages give approximations at

$$x_{n-1}, \quad x_{n-1} + \theta h, \quad x_{n-1} + \tfrac{1}{2}h \quad \text{and} \quad x_{n-1} + h.$$

We will look at the special case $\theta = -\frac{1}{2}$.

In the special $\theta = -\frac{1}{2}$ case

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 1 \\ -2 & 1 & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

In the special $\theta = -\frac{1}{2}$ case

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 1 \\ -2 & 1 & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

Because the derivative at

$$x_{n-1} + \theta h = x_{n-1} - \tfrac{1}{2}h = x_{n-2} + \tfrac{1}{2}h,$$

was evaluated in the previous step, we can try re-using this value.

In the special $\theta = -\frac{1}{2}$ case

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 1 \\ -2 & 1 & 2 & 0 & 1 \\ \hline \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 1 \end{array}\right]$$

Because the derivative at

$$x_{n-1} + \theta h = x_{n-1} - \frac{1}{2}h = x_{n-2} + \frac{1}{2}h,$$

was evaluated in the previous step, we can try re-using this value.

This will save one function evaluation.

A ‘re-use’ method

This gives the re-use method

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ \frac{3}{4} & 0 & 0 & 1 & -\frac{1}{4} \\ -2 & 2 & 0 & 1 & 1 \\ \hline \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

A ‘re-use’ method

This gives the re-use method

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ \frac{3}{4} & 0 & 0 & 1 & -\frac{1}{4} \\ -2 & 2 & 0 & 1 & 1 \\ \hline \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

Why should this method not be preferred to a standard Runge–Kutta method?

A 're-use' method

This gives the re-use method

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ \frac{3}{4} & 0 & 0 & 1 & -\frac{1}{4} \\ -2 & 2 & 0 & 1 & 1 \\ \hline \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

Why should this method not be preferred to a standard Runge–Kutta method?

There are at least two reasons

- Stepsize change is complicated and difficult

A ‘re-use’ method

This gives the re-use method

$$\left[\begin{array}{cc} A & U \\ B & V \end{array}\right] = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ \frac{3}{4} & 0 & 0 & 1 & -\frac{1}{4} \\ -2 & 2 & 0 & 1 & 1 \\ \hline \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

Why should this method not be preferred to a standard Runge–Kutta method?

There are at least two reasons

- Stepsize change is complicated and difficult
- The stability region is smaller

To overcome these difficulties, we can do several things:

To overcome these difficulties, we can do several things:

- Restore the missing stage,

To overcome these difficulties, we can do several things:

- Restore the missing stage,
- Move the first derivative calculation to the end of the previous step,

To overcome these difficulties, we can do several things:

- Restore the missing stage,
- Move the first derivative calculation to the end of the previous step,
- Use a linear combination of the derivatives computed in the previous step (instead of just one of these),

To overcome these difficulties, we can do several things:

- Restore the missing stage,
- Move the first derivative calculation to the end of the previous step,
- Use a linear combination of the derivatives computed in the previous step (instead of just one of these),
- Re-organize the data passed between steps.

To overcome these difficulties, we can do several things:

- Restore the missing stage,
- Move the first derivative calculation to the end of the previous step,
- Use a linear combination of the derivatives computed in the previous step (instead of just one of these),
- Re-organize the data passed between steps.

We then get methods like the following:

An ARK method

$$\left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 1 & 1 & \frac{1}{2} \\ \frac{1}{16} & 0 & 0 & 0 & 1 & \frac{7}{16} & \frac{1}{16} \\ -\frac{1}{4} & 2 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{1}{4} \\ 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ \hline 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 0 & -\frac{2}{3} & 2 & 0 & -1 & 0 \end{array}\right],$$

An ARK method

$$
\left[\begin{array}{cccc|ccc}
0 & 0 & 0 & 0 & 1 & 1 & \frac{1}{2} \\
\frac{1}{16} & 0 & 0 & 0 & 1 & \frac{7}{16} & \frac{1}{16} \\
-\frac{1}{4} & 2 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{1}{4} \\
0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\
\hline
0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-\frac{1}{3} & 0 & -\frac{2}{3} & 2 & 0 & -1 & 0
\end{array}\right],
$$

where

$$
y_1^{[n]} \approx y(x_n), \quad y_2^{[n]} \approx hy'(x_n), \quad y_3^{[n]} \approx h^2 y''(x_n),
$$

An ARK method

$$\left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 1 & 1 & \frac{1}{2} \\ \frac{1}{16} & 0 & 0 & 0 & 1 & \frac{7}{16} & \frac{1}{16} \\ -\frac{1}{4} & 2 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{1}{4} \\ 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ \hline 0 & \frac{2}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 0 & -\frac{2}{3} & 2 & 0 & -1 & 0 \end{array}\right],$$

where

$$y_1^{[n]} \approx y(x_n), \quad y_2^{[n]} \approx hy'(x_n), \quad y_3^{[n]} \approx h^2 y''(x_n),$$

with

$$Y_1 \approx Y_3 \approx Y_4 \approx y(x_n), \qquad Y_2 \approx y(x_{n-1} + \tfrac{1}{2}h).$$

The good things about this “Almost Runge–Kutta method” are:

The good things about this “Almost Runge–Kutta method” are:

- It has the same stability region as for a genuine Runge–Kutta method

The good things about this “Almost Runge–Kutta method” are:

- It has the same stability region as for a genuine Runge–Kutta method
- Unlike standard Runge–Kutta methods, the stage order is 2.

The good things about this “Almost Runge–Kutta method” are:

- It has the same stability region as for a genuine Runge–Kutta method
- Unlike standard Runge–Kutta methods, the stage order is 2.
  This means that the stage values are computed to the same accuracy as an order 2 *Runge-Kutta method.*

The good things about this "Almost Runge–Kutta method" are:

- It has the same stability region as for a genuine Runge–Kutta method
- Unlike standard Runge–Kutta methods, the stage order is 2.
  This means that the stage values are computed to the same accuracy as an order 2 *Runge-Kutta method.*
- Although it is a multi-value method, both starting the method and changing stepsize are essentially cost-free operations.

An Adams-Bashforth/Adams-Moulton method

It is usual practice to combine Adams–Bashforth and Adams–Moulton methods as a predictor corrector pair.

An Adams-Bashforth/Adams-Moulton method

It is usual practice to combine Adams–Bashforth and Adams–Moulton methods as a predictor corrector pair.

For example, the ‘PECE’ method of order 3 computes a predictor y_n^* and a corrector y_n by the formulae

$$y_n^* = y_{n-1} + h\Big(\tfrac{23}{12}f(y_{n-1}) - \tfrac{4}{3}f(y_{n-2}) + \tfrac{5}{12}f(y_{n-3})\Big),$$

An Adams-Bashforth/Adams-Moulton method

It is usual practice to combine Adams–Bashforth and Adams–Moulton methods as a predictor corrector pair.

For example, the ‘PECE’ method of order 3 computes a predictor y_n^* and a corrector y_n by the formulae

$$y_n^* = y_{n-1} + h\Big(\tfrac{23}{12}f(y_{n-1}) - \tfrac{4}{3}f(y_{n-2}) + \tfrac{5}{12}f(y_{n-3})\Big),$$
$$y_n = y_{n-1} + h\Big(\tfrac{5}{12}f(y_n^*) + \tfrac{2}{3}f(y_{n-1}) - \tfrac{1}{12}f(y_{n-2})\Big).$$

An Adams-Bashforth/Adams-Moulton method

It is usual practice to combine Adams–Bashforth and Adams–Moulton methods as a predictor corrector pair.

For example, the 'PECE' method of order 3 computes a predictor y_n^* and a corrector y_n by the formulae

$$y_n^* = y_{n-1} + h\left(\tfrac{23}{12} f(y_{n-1}) - \tfrac{4}{3} f(y_{n-2}) + \tfrac{5}{12} f(y_{n-3})\right),$$

$$y_n = y_{n-1} + h\left(\tfrac{5}{12} f(y_n^*) + \tfrac{2}{3} f(y_{n-1}) - \tfrac{1}{12} f(y_{n-2})\right).$$

It might be asked: Is it possible to obtain improved order by using values of y_{n-2}, y_{n-3} in the formulae?

An Adams-Bashforth/Adams-Moulton method

It is usual practice to combine Adams–Bashforth and Adams–Moulton methods as a predictor corrector pair.

For example, the 'PECE' method of order 3 computes a predictor y_n^* and a corrector y_n by the formulae

$$y_n^* = y_{n-1} + h\Big(\tfrac{23}{12} f(y_{n-1}) - \tfrac{4}{3} f(y_{n-2}) + \tfrac{5}{12} f(y_{n-3})\Big),$$
$$y_n = y_{n-1} + h\Big(\tfrac{5}{12} f(y_n^*) + \tfrac{2}{3} f(y_{n-1}) - \tfrac{1}{12} f(y_{n-2})\Big).$$

It might be asked: Is it possible to obtain improved order by using values of y_{n-2}, y_{n-3} in the formulae?

The answer is that not much can be gained because we are limited by the famous 'Dahlquist barrier'.

A modified linear multistep method

But what if we allow off-step points?

A modified linear multistep method

But what if we allow off-step points?

We can get order 5 if we allow for two predictors, the first giving an approximation to $y(x_n - \frac{1}{2}h)$.

A modified linear multistep method

But what if we allow off-step points?

We can get order 5 if we allow for two predictors, the first giving an approximation to $y(x_n - \frac{1}{2}h)$.

This new method, with predictors at the off-step point and also at the end of the step, is

$$y^*_{n-\frac{1}{2}} = y_{n-2} + h\left(\frac{9}{8}f(y_{n-1}) + \frac{3}{8}f(y_{n-2})\right),$$

A modified linear multistep method

But what if we allow off-step points?

We can get order 5 if we allow for two predictors, the first giving an approximation to $y(x_n - \frac{1}{2}h)$.

This new method, with predictors at the off-step point and also at the end of the step, is

$$y^*_{n-\frac{1}{2}} = y_{n-2} + h\Big(\tfrac{9}{8}f(y_{n-1}) + \tfrac{3}{8}f(y_{n-2})\Big),$$

$$\begin{aligned} y^*_n &= \tfrac{28}{5}y_{n-1} - \tfrac{23}{5}y_{n-2} \\ &\quad + h\Big(\tfrac{32}{15}f(y^*_{n-\frac{1}{2}}) - 4f(y_{n-1}) - \tfrac{26}{15}f(y_{n-2})\Big), \end{aligned}$$

A modified linear multistep method

But what if we allow off-step points?

We can get order 5 if we allow for two predictors, the first giving an approximation to $y(x_n - \frac{1}{2}h)$.

This new method, with predictors at the off-step point and also at the end of the step, is

$$y^*_{n-\frac{1}{2}} = y_{n-2} + h\Big(\tfrac{9}{8}f(y_{n-1}) + \tfrac{3}{8}f(y_{n-2})\Big),$$

$$\begin{aligned} y^*_n &= \tfrac{28}{5}y_{n-1} - \tfrac{23}{5}y_{n-2} \\ &\quad + h\Big(\tfrac{32}{15}f(y^*_{n-\frac{1}{2}}) - 4f(y_{n-1}) - \tfrac{26}{15}f(y_{n-2})\Big), \end{aligned}$$

$$\begin{aligned} y_n &= \tfrac{32}{31}y_{n-1} - \tfrac{1}{31}y_{n-2} \\ &\quad + h\Big(\tfrac{64}{93}f(y^*_{n-\frac{1}{2}}) + \tfrac{5}{31}f(y^*_n) + \tfrac{4}{31}f(y_{n-1}) - \tfrac{1}{93}f(y_{n-2})\Big). \end{aligned}$$

Order of general linear methods

Classical methods are all built on a plan where we know in advance what we are trying to approximate.

Order of general linear methods

Classical methods are all built on a plan where we know in advance what we are trying to approximate.

For an abstract general linear method, the interpretation of the input and output quantities is quite general.

Order of general linear methods

Classical methods are all built on a plan where we know in advance what we are trying to approximate.

For an abstract general linear method, the interpretation of the input and output quantities is quite general.

We want to understand order in a similar general way.

Order of general linear methods

Classical methods are all built on a plan where we know in advance what we are trying to approximate.

For an abstract general linear method, the interpretation of the input and output quantities is quite general.

We want to understand order in a similar general way.

The key ideas are

- Use a general starting method to represent the input to a step.

Order of general linear methods

Classical methods are all built on a plan where we know in advance what we are trying to approximate.

For an abstract general linear method, the interpretation of the input and output quantities is quite general.

We want to understand order in a similar general way.

The key ideas are

- Use a general starting method to represent the input to a step.
- Require the output to be similarly related to the starting method applied one time-step later.

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

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.

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 is supposed to approximate, then the “local truncation error” is defined as the error in the output after a single step.

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 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.

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 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 $\mathcal{S}$ denote the “starting method”

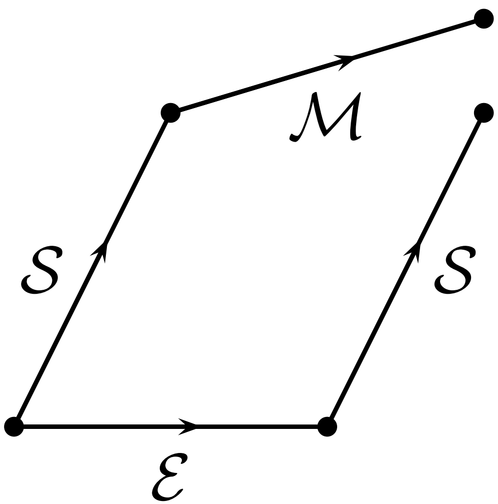
Let $\mathcal{S}$ denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$

Let $\mathcal{S}$ denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$, and let $\mathcal{F} : \mathbb{R}^{rN} \longrightarrow \mathbb{R}^N$ denote a corresponding finishing method

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

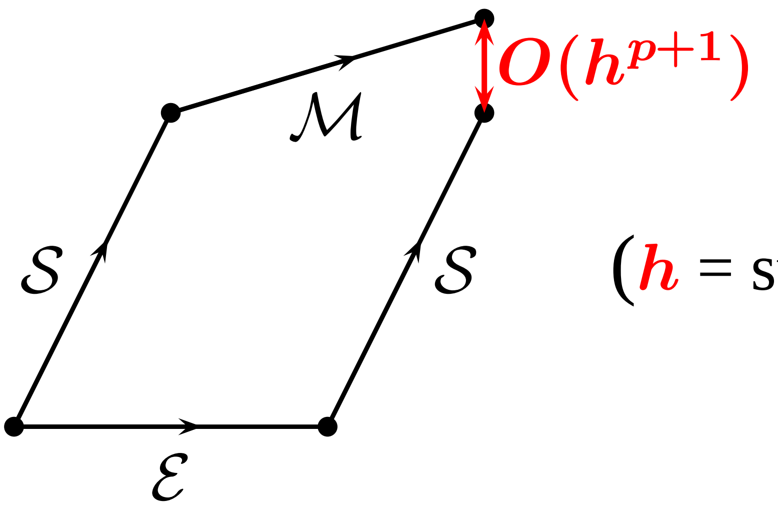
Let $\mathcal{S}$ denote the "starting method", that is a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{rN}$, and let $\mathcal{F} : \mathbb{R}^{rN} \to \mathbb{R}^N$ denote a corresponding finishing method, such that $\mathcal{F} \circ \mathcal{S} = \mathrm{id}$.

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



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

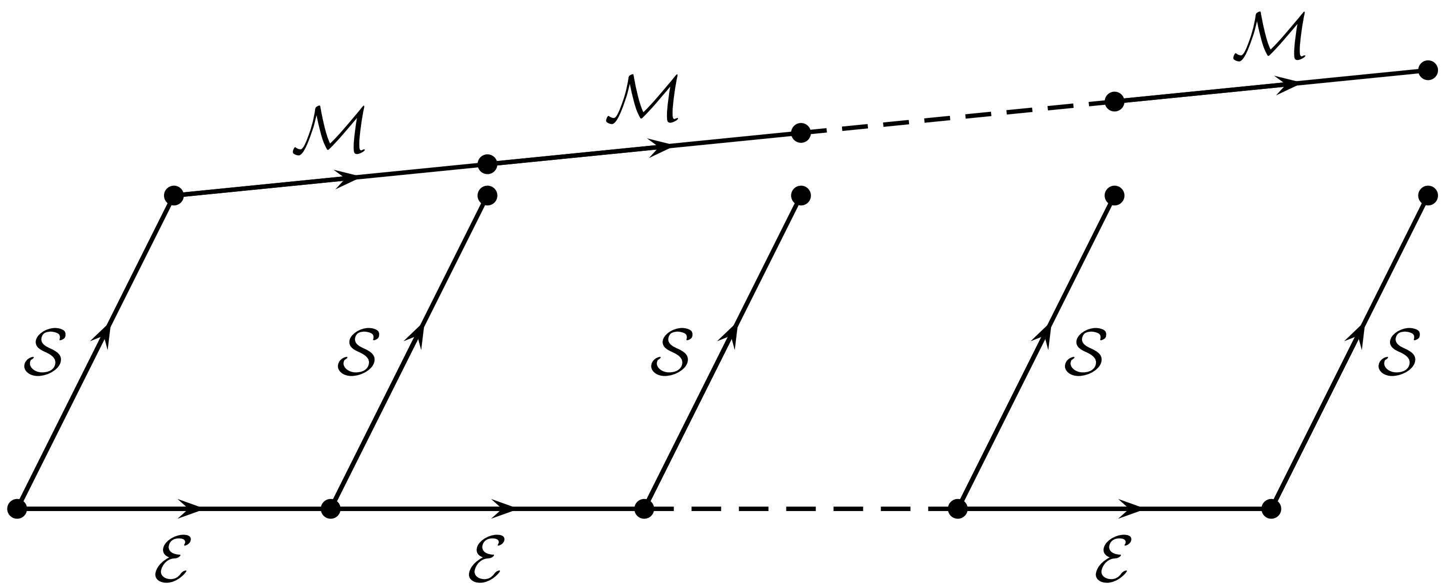
The order of accuracy of a multivalue method is defined in terms of the diagram



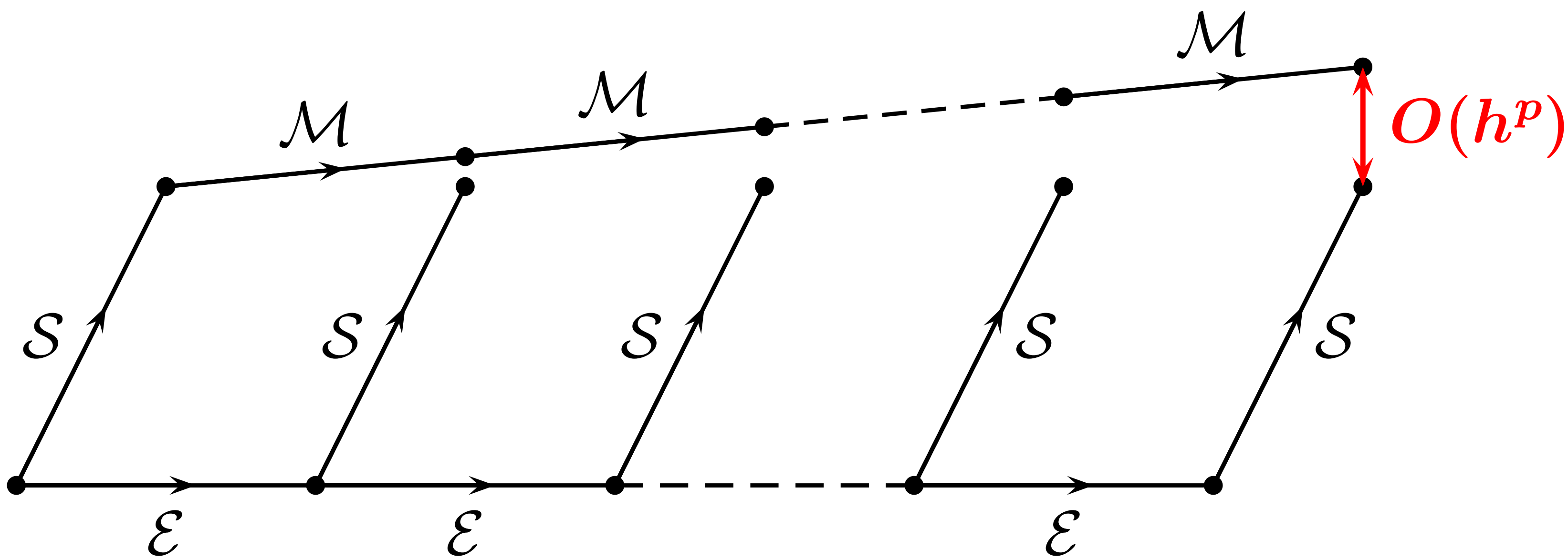
(h = stepsize)

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

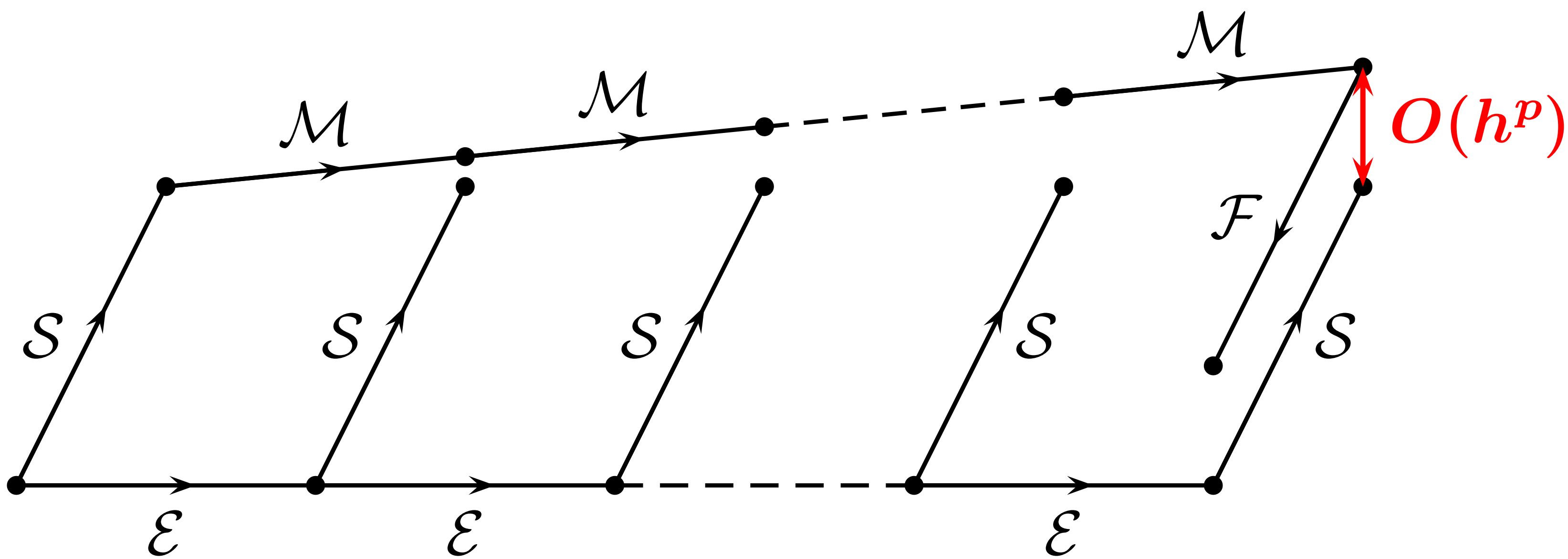
By duplicating this diagram over many steps, global error estimates can be found.



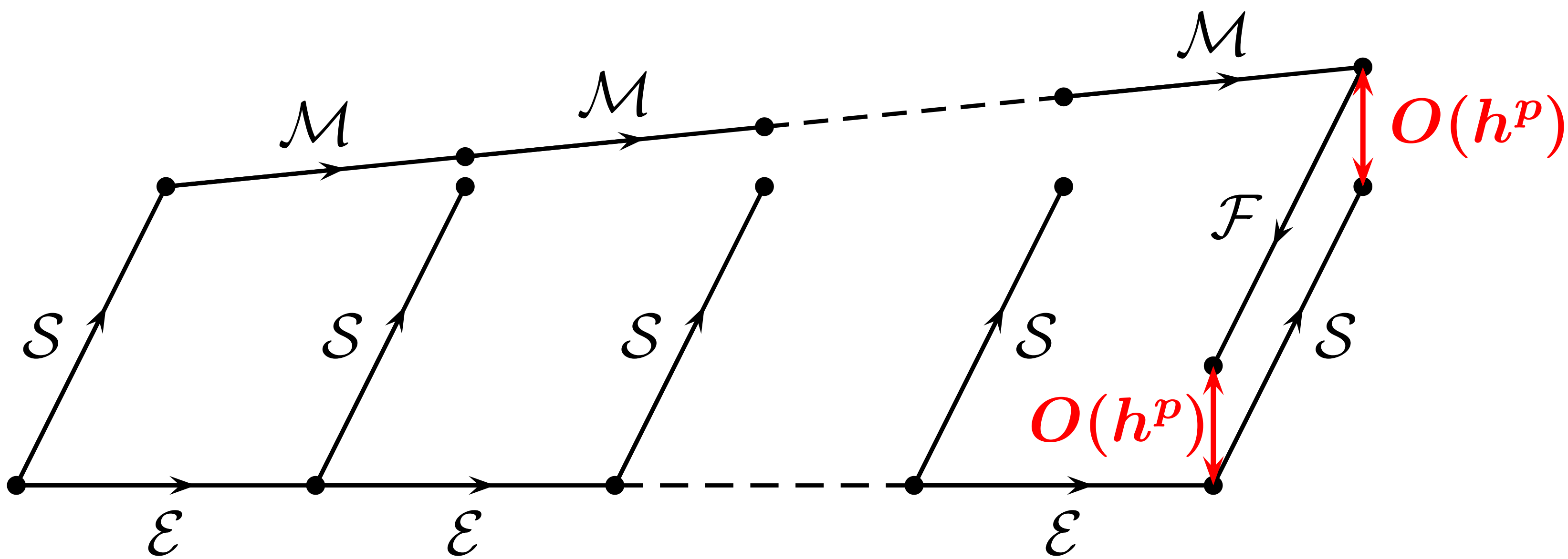
By duplicating this diagram over many steps, global error estimates can be found.



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



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



Methods with the IRK Stability property

An important attribute of a numerical method is its "stability matrix" $M(z)$ defined by

$$M(z) = V + zB(I - zA)^{-1}U.$$

Methods with the IRK Stability property

An important attribute of a numerical method is its "stability matrix" $M(z)$ defined by

$$M(z) = V + zB(I - zA)^{-1}U.$$

This represents the behaviour of the method in the case of *linear* problems.

Methods with the IRK Stability property

An important attribute of a numerical method is its "stability matrix" $M(z)$ defined by

$$M(z) = V + zB(I - zA)^{-1}U.$$

This represents the behaviour of the method in the case of *linear* problems.

That is, for the problem $y'(x) = qy(x)$, we have

$$y^{[n]} = M(z)y^{[n-1]} \qquad \text{where } z = hq$$

Methods with the IRK Stability property

An important attribute of a numerical method is its "stability matrix" $M(z)$ defined by

$$M(z) = V + zB(I - zA)^{-1}U.$$

This represents the behaviour of the method in the case of *linear* problems.

That is, for the problem $y'(x) = qy(x)$, we have

$$y^{[n]} = M(z)y^{[n-1]} \qquad \text{where } z = hq$$

In the special case of a Runge–Kutta method, $M(z)$ is a scalar $R(z)$.

To solve “stiff” problems, we want to use A-stable methods or, even better L-stable methods.

To solve “stiff” problems, we want to use A-stable methods or, even better L-stable methods.

In the case of Runge–Kutta methods the meanings of these are

To solve “stiff” problems, we want to use A-stable methods or, even better L-stable methods.

In the case of Runge–Kutta methods the meanings of these are

- For an A-stable method,

$$|R(z)| \le 1, \qquad \text{if } \mathrm{Re}\, z \le 0.$$

To solve “stiff” problems, we want to use A-stable methods or, even better L-stable methods.

In the case of Runge–Kutta methods the meanings of these are

- For an A-stable method,

$$|R(z)| \leq 1, \qquad \text{if } \mathrm{Re}\, z \leq 0.$$

- An L-stable method is A-stable and, in addition,

$$R(\infty) = 0.$$

A general linear method is said to have

"Runge–Kutta stability"

if the stability matrix for the method $M(z)$ has characteristic polynomial of the form

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

A general linear method is said to have

"Runge–Kutta stability"

if the stability matrix for the method $M(z)$ has characteristic polynomial of the form

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

This means that the method has exactly the same stability region as a Runge–Kutta method whose stability function is $R(z)$.

Do methods with RK stability exist?

Do methods with RK stability exist?

Yes, it is even possible to construct them with rational operations by imposing a condition known as “Inherent RK stability”.

Do methods with RK stability exist?

Yes, it is even possible to construct them with rational operations by imposing a condition known as “Inherent RK stability”.

Methods exist for both stiff and non-stiff problems for arbitrary orders and the only question is how to select the best methods from the large families that are available.

Do methods with RK stability exist?

Yes, it is even possible to construct them with rational operations by imposing a condition known as “Inherent RK stability”.

Methods exist for both stiff and non-stiff problems for arbitrary orders and the only question is how to select the best methods from the large families that are available.

We will give just two examples.

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

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

Implementation questions for IRKS methods

Many implementation questions are similar to those for traditional methods but there are some new challenges.

Implementation questions for IRKS methods

Many implementation questions are similar to those for traditional methods but there are some new challenges.

We want variable order and stepsize and it is even a realistic aim to change between stiff and non-stiff methods automatically.

Implementation questions for IRKS methods

Many implementation questions are similar to those for traditional methods but there are some new challenges.

We want variable order and stepsize and it is even a realistic aim to change between stiff and non-stiff methods automatically.

Because of the variable order and stepsize aims, we wish to be able to do the following:

- Estimate the local truncation error of the current step

Implementation questions for IRKS methods

Many implementation questions are similar to those for traditional methods but there are some new challenges.

We want variable order and stepsize and it is even a realistic aim to change between stiff and non-stiff methods automatically.

Because of the variable order and stepsize aims, we wish to be able to do the following:

- Estimate the local truncation error of the current step
- Estimate the local truncation error of an alternative method of higher order

Implementation questions for IRKS methods

Many implementation questions are similar to those for traditional methods but there are some new challenges.

We want variable order and stepsize and it is even a realistic aim to change between stiff and non-stiff methods automatically.

Because of the variable order and stepsize aims, we wish to be able to do the following:

- Estimate the local truncation error of the current step
- Estimate the local truncation error of an alternative method of higher order
- Change the stepsize with little cost and with little impact on stability

We believe we have solutions to all these problems and that we can construct methods of quite high orders which will work well and competitively.

We believe we have solutions to all these problems and that we can construct methods of quite high orders which will work well and competitively.

I would like to name, with gratitude and appreciation, my principal collaborators in this project:

We believe we have solutions to all these problems and that we can construct methods of quite high orders which will work well and competitively.

I would like to name, with gratitude and appreciation, my principal collaborators in this project:

Zdzisław Jackiewicz	Arizona State University, Phoenix AZ
Helmut Podhaisky	Martin Luther Universität, Halle
Will Wright	La Trobe University, Melbourne

We believe we have solutions to all these problems and that we can construct methods of quite high orders which will work well and competitively.

I would like to name, with gratitude and appreciation, my principal collaborators in this project:

Zdzisław Jackiewicz	Arizona State University, Phoenix AZ
Helmut Podhaisky	Martin Luther Universität, Halle
Will Wright	La Trobe University, Melbourne

I also express my thanks to other colleagues who are closely associated with this project, especially:

Robert Chan, Allison Heard, Shirley Huang, Nicolette Rattenbury, Gustaf Söderlind, Angela Tsai.