SIRK$\longrightarrow$SERK$\longrightarrow$DIRK$\longrightarrow$SDIRK$\longrightarrow$SIRK

John Butcher

The University of Auckland
New Zealand

Contents

- SIRK (Semi-Implicit Runge-Kutta methods)

Contents

- SIRK (Semi-Implicit Runge-Kutta methods)
- SERK (Semi-Explicit Runge-Kutta methods)

Contents

- SIRK (Semi-Implicit Runge-Kutta methods)
- SERK (Semi-Explicit Runge-Kutta methods)
- DIRK (Diagonally-Implicit Runge-Kutta methods)

Contents

- SIRK (Semi-Implicit Runge-Kutta methods)
- SERK (Semi-Explicit Runge-Kutta methods)
- DIRK (Diagonally-Implicit Runge-Kutta methods)
- SDIRK (Singly-Diagonally-Implicit RK methods)

Contents

- SIRK (Semi-Implicit Runge-Kutta methods)
- SERK (Semi-Explicit Runge-Kutta methods)
- DIRK (Diagonally-Implicit Runge-Kutta methods)
- SDIRK (Singly-Diagonally-Implicit RK methods)
- SIRK (Singly-Implicit Runge-Kutta methods)

Optimism and Pessimism

If a glass contains 50% of a pleasant liquid and 50% of space, do we say

"The glass is half empty"?

Optimism and Pessimism

If a glass contains 50% of a pleasant liquid and 50% of space, do we say

“The glass is half empty”

or

“The glass is half full”?

Optimism and Pessimism

If a glass contains 50% of a pleasant liquid and 50% of space, do we say

"The glass is half empty"

or

"The glass is half full"?

This test to distinguish pessimism from optimism has a counterpart in solving ordinary differential equations.

Optimism and Pessimism

If a glass contains 50% of a pleasant liquid and 50% of space, do we say

“The glass is half empty”

or

“The glass is half full”?

This test to distinguish pessimism from optimism has a counterpart in solving ordinary differential equations.

If a numerical method is midway between being fully implicit and fully explicit do we say

“The method is semi-implicit”?

Optimism and Pessimism

If a glass contains 50% of a pleasant liquid and 50% of space, do we say

"The glass is half empty"

or

"The glass is half full"?

This test to distinguish pessimism from optimism has a counterpart in solving ordinary differential equations.

If a numerical method is midway between being fully implicit and fully explicit do we say

"The method is semi-implicit"

or

"The method is semi-explicit"?

This test between optimism and pessimism was failed by me

This test between optimism and pessimism was failed by me but passed by Syvert Nørsett

This test between optimism and pessimism was failed by me but passed by Syvert Nørsett when we first started looking at Runge-Kutta methods with the structure

$$
\begin{array}{c|ccccc}
c_1 & a_{11} & 0 & 0 & \cdots & 0 \\
c_2 & a_{21} & a_{22} & 0 & \cdots & 0 \\
c_3 & a_{31} & a_{32} & a_{33} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_s
\end{array}
$$

This test between optimism and pessimism was failed by me but passed by Syvert Nørsett when we first started looking at Runge-Kutta methods with the structure

$$\begin{array}{c|ccccc} c_1 & a_{11} & 0 & 0 & \cdots & 0 \\ c_2 & a_{21} & a_{22} & 0 & \cdots & 0 \\ c_3 & a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & b_3 & \cdots & b_s \end{array}$$

I called these methods “Semi-Implicit”

This test between optimism and pessimism was failed by me but passed by Syvert Nørsett when we first started looking at Runge-Kutta methods with the structure

$$\begin{array}{c|ccccc} c_1 & a_{11} & 0 & 0 & \cdots & 0 \\ c_2 & a_{21} & a_{22} & 0 & \cdots & 0 \\ c_3 & a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & b_3 & \cdots & b_s \end{array}$$

I called these methods “Semi-Implicit” and Syvert called them “Semi-Explicit”

SIRK

My first interest in Implicit Runge-Kutta methods came from the desire to solve the order conditions.

SIRK

My first interest in Implicit Runge-Kutta methods came from the desire to solve the order conditions. For explicit methods, this becomes increasingly more difficult as the order increases

SIRK

My first interest in Implicit Runge-Kutta methods came from the desire to solve the order conditions. For explicit methods, this becomes increasingly more difficult as the order increases but everything becomes simpler for implicit methods.

SIRK

My first interest in Implicit Runge-Kutta methods came from the desire to solve the order conditions. For explicit methods, this becomes increasingly more difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

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

SERK

One of the topics in the multi-faceted thesis of Syvert Nørsett was the study of methods which are similar to SIRK methods

SERK

One of the topics in the multi-faceted thesis of Syvert Nørsett was the study of methods which are similar to SIRK methods, but with the additional assumption

$$a_{11} = a_{22} = \cdots = a_{ss}$$

SERK

One of the topics in the multi-faceted thesis of Syvert Nørsett was the study of methods which are similar to SIRK methods, but with the additional assumption

$$a_{11} = a_{22} = \cdots = a_{ss}$$

These are the famous SERK methods and, when they were introduced, led to a resurgence of interest in the so-called restricted Padé approximations:

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

SERK

One of the topics in the multi-faceted thesis of Syvert Nørsett was the study of methods which are similar to SIRK methods, but with the additional assumption

$$a_{11} = a_{22} = \cdots = a_{ss}$$

These are the famous SERK methods and, when they were introduced, led to a resurgence of interest in the so-called restricted Padé approximations:

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

Knowing which cases lead to A-stable methods is of crucial importance in the solution of stiff problems.

A brief list of papers illustrates the debt we owe to the work of S. P. Nørsett on SE and other RK methods

A brief list of papers illustrates the debt we owe to the work of S. P. Nørsett on SE and other RK methods

- S. P. Nørsett: C-polynomials for rational approximation to the exponential function, *Numer. Math.* **25** (1975/1976), 39–56.

A brief list of papers illustrates the debt we owe to the work of S. P. Nørsett on SE and other RK methods

- S. P. Nørsett: C-polynomials for rational approximation to the exponential function, *Numer. Math.* **25** (1975/1976), 39–56.
- S. P. Nørsett: Runge-Kutta methods with a multiple real eigenvalue only, *BIT* **16** (1976), 388–393.

A brief list of papers illustrates the debt we owe to the work of S. P. Nørsett on SE and other RK methods

- S. P. Nørsett: C-polynomials for rational approximation to the exponential function, *Numer. Math.* **25** (1975/1976), 39–56.
- S. P. Nørsett: Runge-Kutta methods with a multiple real eigenvalue only, *BIT* **16** (1976), 388–393.
- S. P. Nørsett, A. Wolfbrandt: Attainable order of rational approximation to the exponential function with only real poles, *BIT* **17** (1977), 200–208.

A brief list of papers illustrates the debt we owe to the work of S. P. Nørsett on SE and other RK methods

- S. P. Nørsett: C-polynomials for rational approximation to the exponential function, *Numer. Math.* **25** (1975/1976), 39–56.
- S. P. Nørsett: Runge-Kutta methods with a multiple real eigenvalue only, *BIT* **16** (1976), 388–393.
- S. P. Nørsett, A. Wolfbrandt: Attainable order of rational approximation to the exponential function with only real poles, *BIT* **17** (1977), 200–208.
- S. P. Nørsett: Restricted Padé approximations to the exponential function, *SIAM J. Numer. Anal.* **15** (1978), 1008–1092.

DIRK

These methods with the acronym denoting *Diagonally-Implicit Runge-Kutta methods* were studied by Roger Alexander.

DIRK

These methods with the acronym denoting *Diagonally-Implicit Runge-Kutta methods* were studied by Roger Alexander. The methods are just like SERK methods and were developed independently.

DIRK

These methods with the acronym denoting *Diagonally-Implicit Runge-Kutta methods* were studied by Roger Alexander. The methods are just like SERK methods and were developed independently.
The following third order L-stable method illustrates what is possible for DIRK methods

DIRK

These methods with the acronym denoting *Diagonally-Implicit Runge-Kutta methods* were studied by Roger Alexander. The methods are just like SERK methods and were developed independently.
The following third order L-stable method illustrates what is possible for DIRK methods

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

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

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system.

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system. As we shall see in the next few slides, singly-implicit methods *without the DIRK structure* are also possible.

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system. As we shall see in the next few slides, singly-implicit methods *without the DIRK structure* are also possible. At least two papers by the indomitable team of S. P. Nørsett and P. G. Thomsen have used the SDIRK name.

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system. As we shall see in the next few slides, singly-implicit methods *without the DIRK structure* are also possible. At least two papers by the indomitable team of S. P. Nørsett and P. G. Thomsen have used the SDIRK name.

- Embedded SDIRK methods of basic order three, *BIT* **24** (1984), 364-646.

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system. As we shall see in the next few slides, singly-implicit methods *without the DIRK structure* are also possible. At least two papers by the indomitable team of S. P. Nørsett and P. G. Thomsen have used the SDIRK name.

- Embedded SDIRK methods of basic order three, *BIT* **24** (1984), 364-646.
- Local error control in SDIRK methods, *BIT* **26** (1986), 100-113.

SDIRK

The change of name emphasised the singly-implicit nature of SDIRK methods and seems to have been part of an attempt at a systematic naming system. As we shall see in the next few slides, singly-implicit methods *without the DIRK structure* are also possible. At least two papers by the indomitable team of S. P. Nørsett and P. G. Thomsen have used the SDIRK name.

- Embedded SDIRK methods of basic order three, *BIT* **24** (1984), 364-646.
- Local error control in SDIRK methods, *BIT* **26** (1986), 100-113.

This work is part of a practical project to obtain efficient stiff solvers of moderate order.

SIRK

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

SIRK

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

SIRK

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

For DIRK methods the stages can be computed independently and sequentially

SIRK

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

For DIRK methods the stages can be computed independently and sequentially and each requires approximately the same factorised matrix $I - h\lambda J$ to permit solution by a modified Newton iteration process.

SIRK

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

For DIRK methods the stages can be computed independently and sequentially and each requires approximately the same factorised matrix $I - h\lambda J$ to permit solution by a modified Newton iteration process.

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

SIRK

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

For DIRK methods the stages can be computed independently and sequentially and each requires approximately the same factorised matrix $I - h\lambda J$ to permit solution by a modified Newton iteration process.

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

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

Suppose the matrix T transforms A to canonical form as follows

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

Suppose the matrix T transforms A to canonical form as follows

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

where

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

Suppose the matrix T transforms A to canonical form as follows

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

where

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

Consider a single Newton iteration, simplified by the use of the same approximate Jacobian J for each stage.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The Newton process consists of solving the linear system

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

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

The Newton process consists of solving the linear system

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

and updating

$$Y \rightarrow Y - D$$

To benefit from the SI property, write

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

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

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

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

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

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

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

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

In summary, we reduce the very high LU factorisation cost

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

In summary, we reduce the very high LU factorisation cost

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

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

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

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

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

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

Also we reduce the back substitution cost

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

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

Also we reduce the back substitution cost

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SIRK methods and stage order

Stage order s means that

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

SIRK methods and stage order

Stage order s means that

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

for ϕ any polynomial of degree $s-1$.

SIRK methods and stage order

Stage order s means that

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

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

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

SIRK methods and stage order

Stage order s means that

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

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

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

where the vector powers are interpreted component by component.

SIRK methods and stage order

Stage order s means that

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

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

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

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

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

From the Cayley-Hamilton theorem

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

From the Cayley-Hamilton theorem

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

and hence

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

From the Cayley-Hamilton theorem

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

and hence

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

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

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

Hence each component of c satisfies

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

Hence each component of c satisfies

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

That is

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

where L_S denotes the Laguerre polynomial of degree s.

Hence each component of c satisfies

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

That is

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

where L_S denotes the Laguerre polynomial of degree s.

Let ξ_1, ξ_2, . . . , ξ_s denote the zeros of L_s so that

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

Hence each component of c satisfies

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

That is

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

where L_S denotes the Laguerre polynomial of degree s.

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

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

The question now is, how should λ be chosen?

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

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

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

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

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

We will look at two approaches for overcoming this disadvantage.

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

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

We will look at two approaches for overcoming this disadvantage.

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

Define the matrix T as follows:

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

Define the matrix T as follows:

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

It can be shown that for a SIRK method

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

Improving SIRK methods

There are two ways in which SIRK methods can be generalized

In the first of these we add extra diagonally implicit stages so that the coefficient matrix looks like this:

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

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

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

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

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

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

This allows us to locate the abscissae where we wish.

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

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

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

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

We will examine effective order in more detail.

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

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

Effective order

We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

Effective order

We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

First we look at the meaning of order for Runge-Kutta methods and to its generalisation to effective order.

Effective order

We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

First we look at the meaning of order for Runge-Kutta methods and to its generalisation to effective order.

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

Effective order

We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

First we look at the meaning of order for Runge-Kutta methods and to its generalisation to effective order.

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

Effective order

We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

First we look at the meaning of order for Runge-Kutta methods and to its generalisation to effective order.

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

We will illustrate this operation in a table

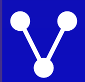
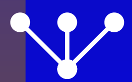
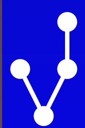
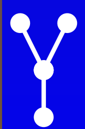
Effective order

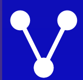
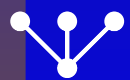
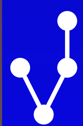
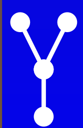
We will consider how to use the properties of doubly-companion matrices to derive SIRK methods with effective order s.

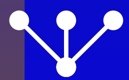
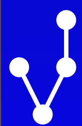
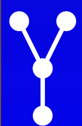
First we look at the meaning of order for Runge-Kutta methods and to its generalisation to effective order.

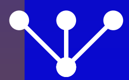
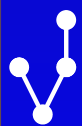
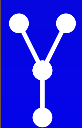
Denote by G the group consisting of mappings of (rooted) trees to real numbers where the group operation is defined in the usual way, according to the algebraic theory of Runge-Kutta methods or to the (equivalent) theory of B-series.

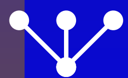
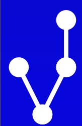
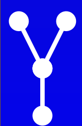
We will illustrate this operation in a table, where we also introduce the special member $E \in G$.

i	t_i
1	
2	
3	
4	
5	
6	
7	
8	

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

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

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

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

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

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

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

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

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

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

$$\alpha G_p = E G_p.$$

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

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

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

$$\alpha G_p = E G_p.$$

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

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

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

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

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

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

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

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

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

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

We analyse the conditions for effective order 4.

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

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

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

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

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

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

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

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

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

Effective order of SIRK methods

If the stage order is equal to the order, then this analysis can be simplified.

Effective order of SIRK methods

If the stage order is equal to the order, then this analysis can be simplified.
We can assume the input to step n is an approximation to

$$y(x_{n-1}) + \alpha_1 h y'(x_{n-1}) + \cdots + \alpha_s h^s y^{(s)}(x_{n-1})$$

Effective order of SIRK methods

If the stage order is equal to the order, then this analysis can be simplified.
We can assume the input to step n is an approximation to

$$y(x_{n-1}) + \alpha_1 h y'(x_{n-1}) + \cdots + \alpha_s h^s y^{(s)}(x_{n-1})$$

and we want the output to be an approximation to

$$y(x_n) + \alpha_1 h y'(x_n) + \cdots + \alpha_s h^s y^{(s)}(x_n)$$

Effective order of SIRK methods

If the stage order is equal to the order, then this analysis can be simplified.
We can assume the input to step n is an approximation to

$$y(x_{n-1}) + \alpha_1 h y'(x_{n-1}) + \cdots + \alpha_s h^s y^{(s)}(x_{n-1})$$

and we want the output to be an approximation to

$$y(x_n) + \alpha_1 h y'(x_n) + \cdots + \alpha_s h^s y^{(s)}(x_n)$$

To construct a SIRK method with effective order s, and with a specific choice of the abscissa vector c and a specific value of λ, use the properties of doubly companion matrices.

Construction of methods

- Choose λ and abscissa vector c.

Construction of methods

- Choose λ and abscissa vector c.

- Define $\beta_1, \beta_2, \ldots, \beta_s$ so that the zeros of the polynomial

$$\frac{1}{s!}x^s + \frac{\beta_1}{(s-1)!}x^{s-1} + \cdots + \beta_s$$

  are $c_1, c_2, \ldots\ c_s$.

Construction of methods

- Choose λ and abscissa vector c.
- Define β_1, β_2, $\ldots$, β_s so that the zeros of the polynomial

$$\frac{1}{s!}x^s + \frac{\beta_1}{(s-1)!}x^{s-1} + \cdots + \beta_s$$

  are c_1, c_2, $\ldots$ c_s.

- Define α_1, α_2, $\ldots$, α_s so that $\alpha(z)\beta(z) = (1-\lambda z)^s + O(z^{s+1})$

Construction of methods

- Choose λ and abscissa vector c.
- Define $\beta_1, \beta_2, \ldots, \beta_s$ so that the zeros of the polynomial

$$\frac{1}{s!}x^s + \frac{\beta_1}{(s-1)!}x^{s-1} + \cdots + \beta_s$$

  are $c_1, c_2, \ldots\ c_s$.
- Define $\alpha_1, \alpha_2, \ldots, \alpha_s$ so that $\alpha(z)\beta(z) = (1-\lambda z)^s + O(z^{s+1})$
- Construct the corresponding doubly companion matrix X

- Construct $A = CXC^{-1}$

- Construct $A = CXC^{-1}$, where

$$C = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(s-1)!}c_1^{s-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(s-1)!}c_2^{s-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_s & \frac{1}{2!}c_s^2 & \cdots & \frac{1}{(s-1)!}c_s^{s-1} \end{bmatrix}$$

- Construct $A = CXC^{-1}$, where

$$C = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(s-1)!}c_1^{s-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(s-1)!}c_2^{s-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_s & \frac{1}{2!}c_s^2 & \cdots & \frac{1}{(s-1)!}c_s^{s-1} \end{bmatrix}$$

- Construct $b^T = \widehat{b}^T C^{-1}$

- Construct $A = CXC^{-1}$, where

$$C = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(s-1)!}c_1^{s-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(s-1)!}c_2^{s-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_s & \frac{1}{2!}c_s^2 & \cdots & \frac{1}{(s-1)!}c_s^{s-1} \end{bmatrix}$$

- Construct $b^T = \widehat{b}^T C^{-1}$, where

$$\widehat{b}^T = \left[\frac{1}{1!},\ \frac{\alpha_1}{1!} + \frac{1}{2!},\ \frac{\alpha_2}{1!} + \frac{\alpha_1}{2!} + \frac{1}{3!},\ \cdots\ ,\ \frac{\alpha_{s-1}}{1!} + \cdots + \frac{1}{s!}\right]$$

Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:

  accuracy, stability, low cost

Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:
  accuracy, stability, low cost
- A good place to look for these methods is amongst SERK methods and their generalisations

Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:
  accuracy, stability, low cost
- A good place to look for these methods is amongst SERK methods and their generalisations
- Further generalisations are also possible

Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:
  accuracy,   stability,   low cost
- A good place to look for these methods is amongst SERK methods and their generalisations
- Further generalisations are also possible
- The development of stiff methods is inextricably linked with the pioneering work of S. P. Nørsett

Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:
  accuracy, stability, low cost
- A good place to look for these methods is amongst SERK methods and their generalisations
- Further generalisations are also possible
- The development of stiff methods is inextricably linked with the pioneering work of S. P. Nørsett
- I am greatly honoured to be present at this celebration of his outstanding contributions

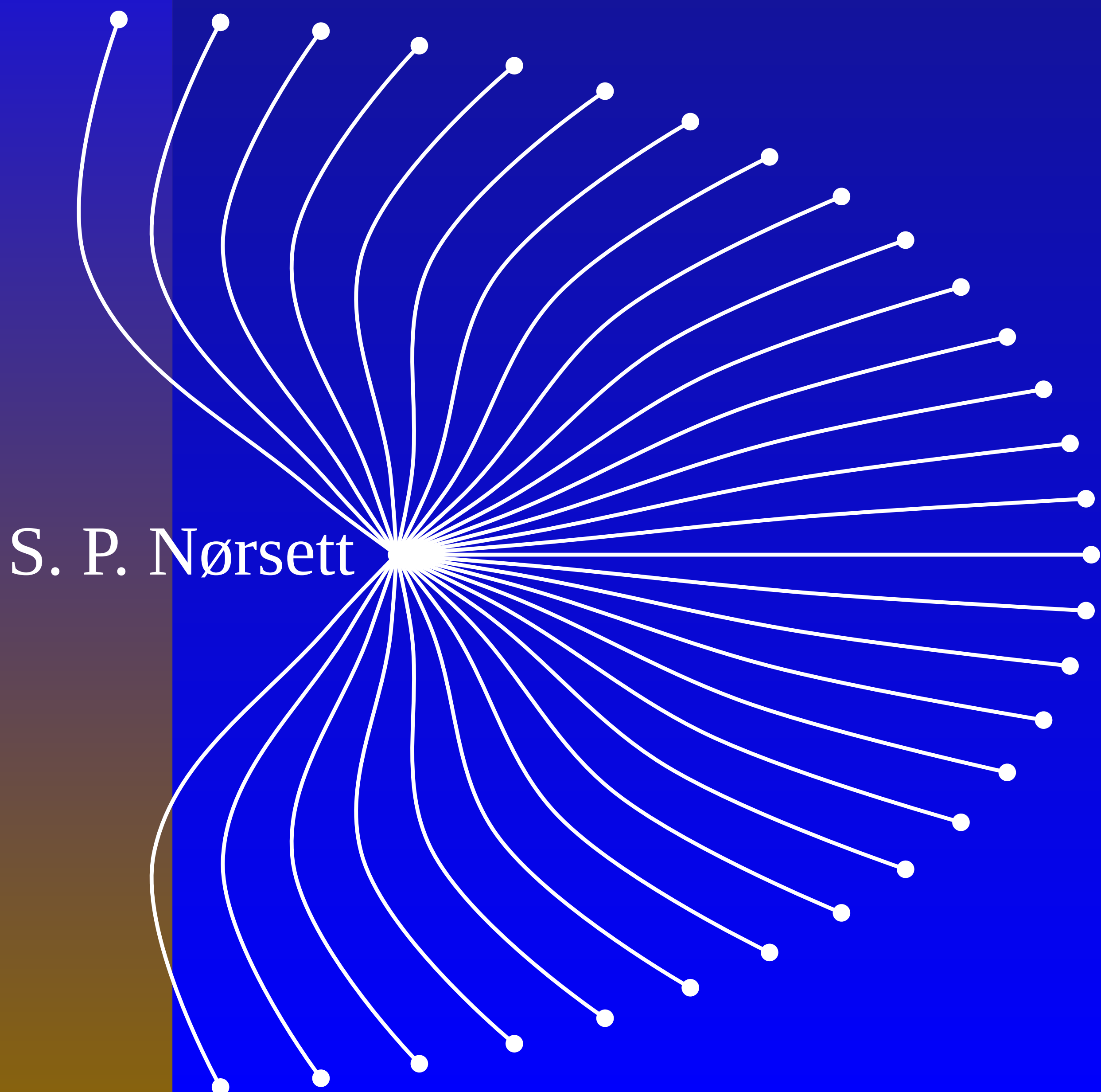
Final comments

- The search for efficient methods to solve stiff problems requires a balance between three aims:
  accuracy, stability, low cost
- A good place to look for these methods is amongst SERK methods and their generalisations
- Further generalisations are also possible
- The development of stiff methods is inextricably linked with the pioneering work of S. P. Nørsett
- I am greatly honoured to be present at this celebration of his outstanding contributions
- Lastly: I will report on my efforts to document links between the two of us

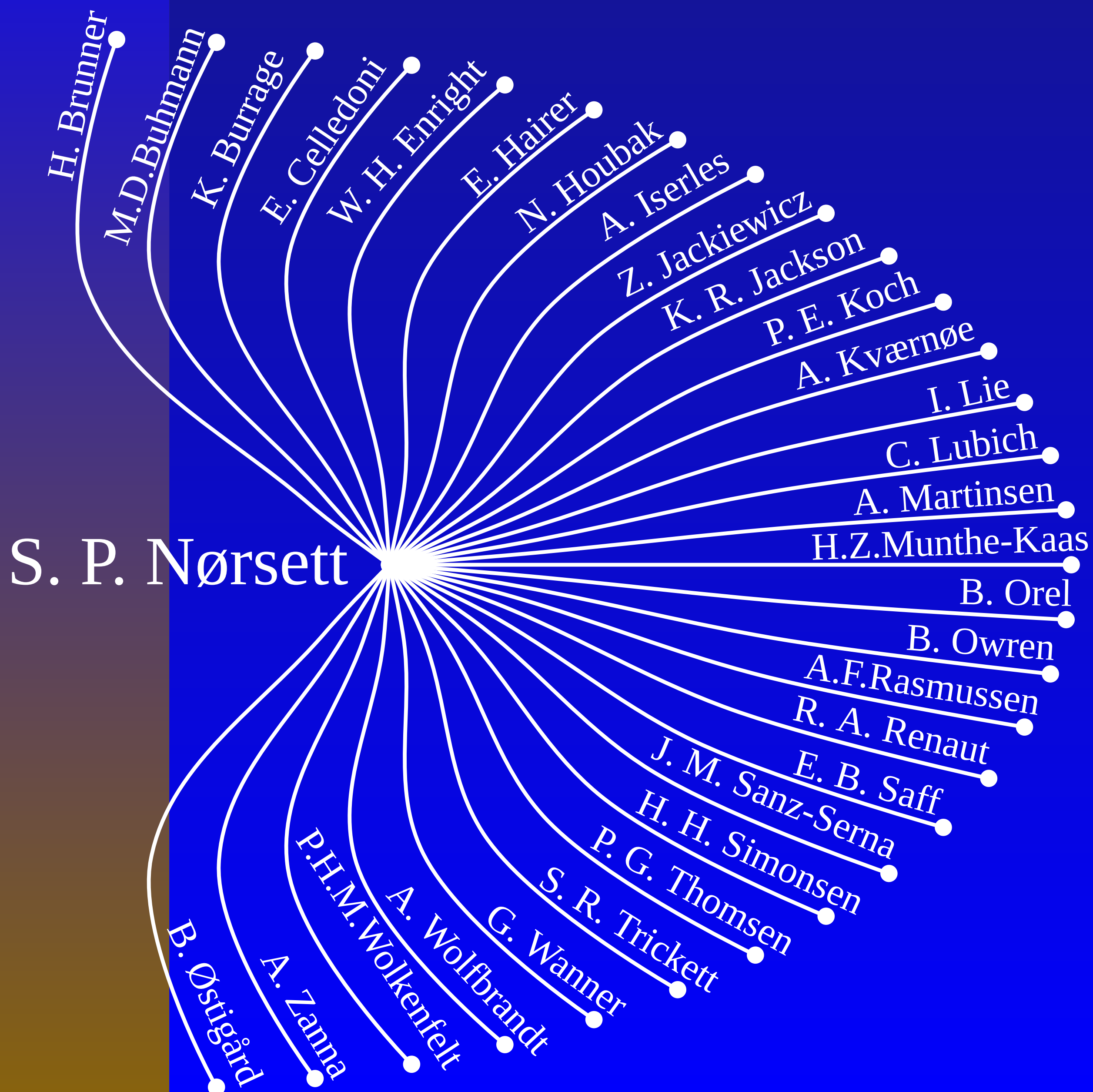
Proof that my Nørsett number is 2

- S. P. Nørsett

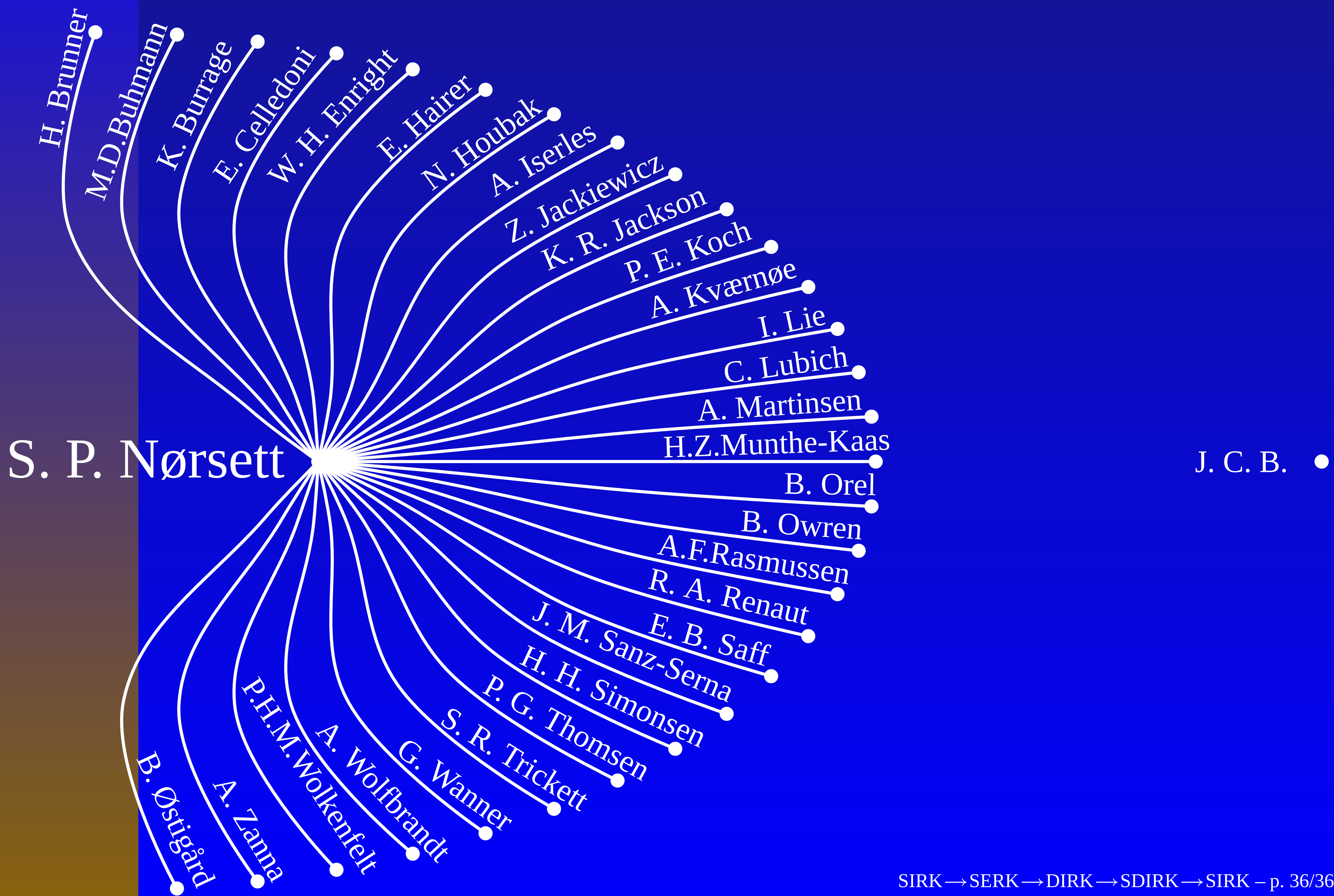
Proof that my Nørsett number is 2



Proof that my Nørsett number is 2



Proof that my Nørsett number is 2



Proof that my Nørsett number is 2