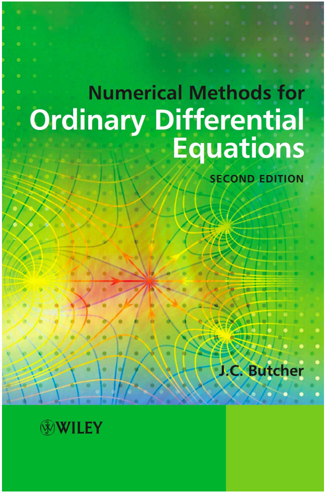
John Butcher's tutorials

Implicit Runge–Kutta methods

$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$
$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$



Since we have an order barrier, which says that order p RK methods require more than p stages if $p > 4$, we might ask how to get around this barrier.

For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

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

Since we have an order barrier, which says that order p RK methods require more than p stages if $p > 4$, we might ask how to get around this barrier.

For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

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

Since we have an order barrier, which says that order p RK methods require more than p stages if $p > 4$, we might ask how to get around this barrier.

For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

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

Since we have an order barrier, which says that order p RK methods require more than p stages if $p > 4$, we might ask how to get around this barrier.

For explicit methods, solving the order conditions becomes increasingly difficult as the order increases but everything becomes simpler for implicit methods.

For example the following method has order 5:

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

We could check the order of this method by verifying the 17 order conditions but there is an easier way.

A method has order 5 if it satisfies the B(5), C(2) and D(2) conditions.

A method satisfies B(k), C(k), D(k) and E(k,ℓ) if

$$\sum_{i=1}^{s} b_i c_i^{j-1} = \tfrac{1}{j}, \qquad j = 1,2,\dots,k, \qquad \mathrm{B}(k)$$

$$\sum_{j=1}^{s} a_{ij} c_j^{\ell-1} = \tfrac{1}{\ell} c_i^{\ell}, \qquad i = 1,2,\dots,s, \ell = 1,2,\dots,k, \quad \mathrm{C}(k)$$

$$\sum_{i=1}^{s} b_i c_i^{\ell-1} a_{ij} = \tfrac{1}{\ell} b_j (1-c_j^{\ell}), j = 1,2,\dots,s, \ell = 1,2,\dots,k, \quad \mathrm{D}(k)$$

$$\sum_{i,j=1}^{s} b_i c_i^{m-1} a_{ij} c_j^{n-1} = \tfrac{1}{(m+n)n}, \qquad m,n = 1,2,\dots,s, \qquad \mathrm{E}(k,\ell)$$

and B(5), C(2) and D(2) are easy to check for this method.

We could check the order of this method by verifying the 17 order conditions but there is an easier way.
A method has order 5 if it satisfies the B(5), C(2) and D(2) conditions.
A method satisfies B(k), C(k), D(k) and E(k,ℓ) if

$$\sum_{i=1}^{s} b_i c_i^{j-1} = \tfrac{1}{j}, \qquad j = 1, 2, \ldots, k, \qquad \mathrm{B}(k)$$

$$\sum_{j=1}^{s} a_{ij} c_j^{\ell-1} = \tfrac{1}{\ell} c_i^{\ell}, \qquad i = 1, 2, \ldots, s, \ell = 1, 2, \ldots, k, \quad \mathrm{C}(k)$$

$$\sum_{i=1}^{s} b_i c_i^{\ell-1} a_{ij} = \tfrac{1}{\ell} b_j (1 - c_j^{\ell}), j = 1, 2, \ldots, s, \ell = 1, 2, \ldots, k, \quad \mathrm{D}(k)$$

$$\sum_{i,j=1}^{s} b_i c_i^{m-1} a_{ij} c_j^{n-1} = \tfrac{1}{(m+n)n}, \qquad m, n = 1, 2, \ldots, s, \qquad \mathrm{E}(k,\ell)$$

and B(5), C(2) and D(2) are easy to check for this method.

We could check the order of this method by verifying the 17 order conditions but there is an easier way.
A method has order 5 if it satisfies the B(5), C(2) and D(2) conditions.
A method satisfies B(k), C(k), D(k) and E(k, ℓ) if

$$\sum_{i=1}^{s} b_i c_i^{j-1} = \frac{1}{j}, \qquad j = 1, 2, \dots, k, \qquad \mathrm{B}(k)$$

$$\sum_{j=1}^{s} a_{ij} c_j^{\ell-1} = \frac{1}{\ell} c_i^{\ell}, \qquad i = 1, 2, \dots, s, \ell = 1, 2, \dots, k, \qquad \mathrm{C}(k)$$

$$\sum_{i=1}^{s} b_i c_i^{\ell-1} a_{ij} = \frac{1}{\ell} b_j (1 - c_j^{\ell}), j = 1, 2, \dots, s, \ell = 1, 2, \dots, k, \qquad \mathrm{D}(k)$$

$$\sum_{i,j=1}^{s} b_i c_i^{m-1} a_{ij} c_j^{n-1} = \frac{1}{(m+n)n}, \qquad m, n = 1, 2, \dots, s, \qquad \mathrm{E}(k, \ell)$$

and B(5), C(2) and D(2) are easy to check for this method.

We could check the order of this method by verifying the 17 order conditions but there is an easier way.

A method has order 5 if it satisfies the B(5), C(2) and D(2) conditions.

A method satisfies B(k), C(k), D(k) and E(k, ℓ) if

$$\sum_{i=1}^{s} b_i c_i^{j-1} = \tfrac{1}{j}, \qquad j = 1, 2, \dots, k, \qquad \text{B}(k)$$

$$\sum_{j=1}^{s} a_{ij} c_j^{\ell-1} = \tfrac{1}{\ell} c_i^{\ell}, \qquad i = 1, 2, \dots, s, \ell = 1, 2, \dots, k, \qquad \text{C}(k)$$

$$\sum_{i=1}^{s} b_i c_i^{\ell-1} a_{ij} = \tfrac{1}{\ell} b_j (1 - c_j^{\ell}), j = 1, 2, \dots, s, \ell = 1, 2, \dots, k, \qquad \text{D}(k)$$

$$\sum_{i,j=1}^{s} b_i c_i^{m-1} a_{ij} c_j^{n-1} = \frac{1}{(m+n)n}, \qquad m, n = 1, 2, \dots, s, \qquad \text{E}(k, \ell)$$

and B(5), C(2) and D(2) are easy to check for this method.

The most important types of “fully implicit” methods (that is A can have any structure) are

- Gauss methods of order $2s$, characterized by B($2s$) and C(s). To satisfy B($2s$), the c_i must be zeros of $P_s(2x-1)=0$, where P_s is the Legendre polynomial of degree s.
- Radau IIA methods of order $2s-1$, characterized by $c_s=1$, B($2s-1$) and C(s). The c_i are zeros of $P_s(2x-1)-P_{s-1}(2x-1)=0$.

Both these families of methods are A-stable.

But both are very expensive to implement and both can suffer from order reduction.

The most important types of "fully implicit" methods (that is A can have any structure) are

- Gauss methods of order $2s$, characterized by B($2s$) and C(s). To satisfy B($2s$), the c_i must be zeros of $P_s(2x-1)=0$, where P_s is the Legendre polynomial of degree s.
- Radau IIA methods of order $2s-1$, characterized by $c_s = 1$, B($2s-1$) and C(s). The c_i are zeros of $P_s(2x-1) - P_{s-1}(2x-1) = 0$.

Both these families of methods are A-stable.

But both are very expensive to implement and both can suffer from order reduction.

The most important types of "fully implicit" methods (that is A can have any structure) are

- Gauss methods of order $2s$, characterized by $\mathrm{B}(2s)$ and $\mathrm{C}(s)$. To satisfy $\mathrm{B}(2s)$, the c_i must be zeros of $P_s(2x-1)=0$, where P_s is the Legendre polynomial of degree s.
- Radau IIA methods of order $2s-1$, characterized by $c_s=1$, $\mathrm{B}(2s-1)$ and $\mathrm{C}(s)$. The c_i are zeros of $P_s(2x-1)-P_{s-1}(2x-1)=0$.

Both these families of methods are A-stable.

But both are very expensive to implement and both can suffer from order reduction.

The most important types of "fully implicit" methods (that is A can have any structure) are

- Gauss methods of order $2s$, characterized by $\mathrm{B}(2s)$ and $\mathrm{C}(s)$. To satisfy $\mathrm{B}(2s)$, the c_i must be zeros of $P_s(2x-1)=0$, where P_s is the Legendre polynomial of degree s.
- Radau IIA methods of order $2s-1$, characterized by $c_s=1$, $\mathrm{B}(2s-1)$ and $\mathrm{C}(s)$. The c_i are zeros of $P_s(2x-1)-P_{s-1}(2x-1)=0$.

Both these families of methods are A-stable.

But both are very expensive to implement and both can suffer from order reduction.

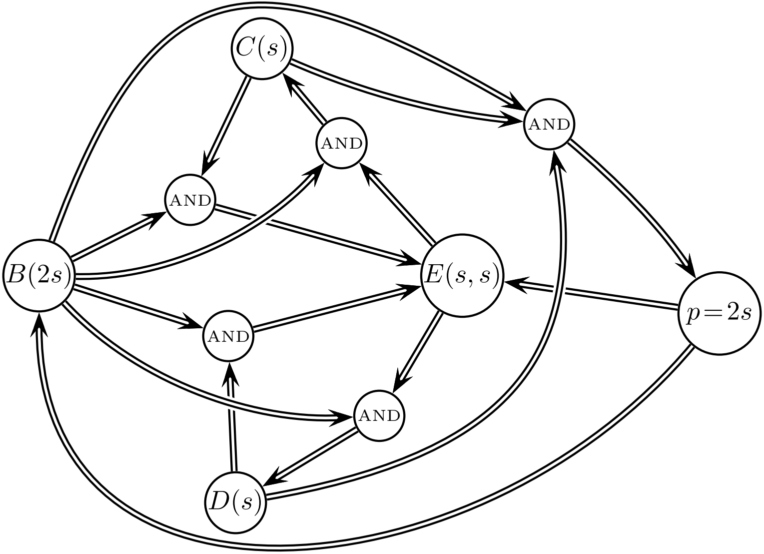
The most important types of "fully implicit" methods (that is A can have any structure) are

- Gauss methods of order $2s$, characterized by $\mathrm{B}(2s)$ and $\mathrm{C}(s)$. To satisfy $\mathrm{B}(2s)$, the c_i must be zeros of $P_s(2x-1)=0$, where P_s is the Legendre polynomial of degree s.
- Radau IIA methods of order $2s-1$, characterized by $c_s=1$, $\mathrm{B}(2s-1)$ and $\mathrm{C}(s)$. The c_i are zeros of $P_s(2x-1)-P_{s-1}(2x-1)=0$.

Both these families of methods are A-stable.

But both are very expensive to implement and both can suffer from order reduction.

Outline proof that Gauss methods have order $2s$



Examples of Gauss methods

$s = 1:$

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

$s = 2:$

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$s = 3:$

$$\begin{array}{c|ccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

Examples of Gauss methods

$$s = 1: \qquad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

$$s = 2: \qquad \begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$$s = 3: \qquad \begin{array}{c|ccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

Examples of Gauss methods

$$s = 1: \qquad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

$$s = 2: \qquad \begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$$s = 3: \qquad \begin{array}{c|ccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

Examples of Radau IIA methods

$$s = 1: \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

$$s = 2: \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

$$s = 3: \qquad \begin{array}{c|ccc} \frac{2}{5} - \frac{\sqrt{6}}{10} & \frac{11}{45} - \frac{7\sqrt{6}}{360} & \frac{37}{225} - \frac{169\sqrt{6}}{1800} & -\frac{2}{225} + \frac{\sqrt{6}}{75} \\ \frac{2}{5} + \frac{\sqrt{6}}{10} & \frac{37}{225} + \frac{169\sqrt{6}}{1800} & \frac{11}{45} + \frac{7\sqrt{6}}{360} & -\frac{2}{225} - \frac{\sqrt{6}}{75} \\ 1 & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Examples of Radau IIA methods

$s = 1:$

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

$s = 2:$

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

$s = 3:$

$$\begin{array}{c|ccc} \frac{2}{5}-\frac{\sqrt{6}}{10} & \frac{11}{45}-\frac{7\sqrt{6}}{360} & \frac{37}{225}-\frac{169\sqrt{6}}{1800} & -\frac{2}{225}+\frac{\sqrt{6}}{75} \\ \frac{2}{5}+\frac{\sqrt{6}}{10} & \frac{37}{225}+\frac{169\sqrt{6}}{1800} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & -\frac{2}{225}-\frac{\sqrt{6}}{75} \\ 1 & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Examples of Radau IIA methods

$$s = 1: \quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

$$s = 2: \quad \begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

$$s = 3: \quad \begin{array}{c|ccc} \frac{2}{5} - \frac{\sqrt{6}}{10} & \frac{11}{45} - \frac{7\sqrt{6}}{360} & \frac{37}{225} - \frac{169\sqrt{6}}{1800} & -\frac{2}{225} + \frac{\sqrt{6}}{75} \\ \frac{2}{5} + \frac{\sqrt{6}}{10} & \frac{37}{225} + \frac{169\sqrt{6}}{1800} & \frac{11}{45} + \frac{7\sqrt{6}}{360} & -\frac{2}{225} - \frac{\sqrt{6}}{75} \\ 1 & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

This idea of choosing A as a lower triangular matrix can be taken further by avoiding diagonal zeros.

If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett (referred to as semi-implicit by J.C. Butcher in 1965).

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

This idea of choosing A as a lower triangular matrix can be taken further by avoiding diagonal zeros.

If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett (referred to as semi-implicit by J.C. Butcher in 1965).

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

This idea of choosing A as a lower triangular matrix can be taken further by avoiding diagonal zeros.

If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett (referred to as semi-implicit by J.C. Butcher in 1965).

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

This idea of choosing A as a lower triangular matrix can be taken further by avoiding diagonal zeros.

If all the diagonal elements are equal, we get the Diagonally-Implicit methods of R. Alexander and the Semi-Explicit methods of S. P. Nørsett (referred to as semi-implicit by J.C. Butcher in 1965).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Suppose the matrix T transforms A to canonical form as follows

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

where

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

Suppose the matrix T transforms A to canonical form as follows

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

where

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

Suppose the matrix T transforms A to canonical form as follows

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

where

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

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

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

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

Assume the incoming approximation is y_0 and that we are attempting to evaluate

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

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

The implicit equations to be solved are

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

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

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

Assume the incoming approximation is y_0 and that we are attempting to evaluate

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

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

The implicit equations to be solved are

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

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

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

Assume the incoming approximation is y_0 and that we are attempting to evaluate

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

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

The implicit equations to be solved are

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

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

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

Assume the incoming approximation is y_0 and that we are attempting to evaluate

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

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

The implicit equations to be solved are

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

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

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

The Newton process consists of solving the linear system

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

and updating

$$Y \to Y - D$$

To benefit from the SI property, write

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

so that

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

The following table summarises the costs

	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.

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

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

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

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

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

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

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

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

Stage order s means that

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

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

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

where the vector powers are interpreted component by component.

This is equivalent to

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

Stage order s means that

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

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

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

where the vector powers are interpreted component by component.

This is equivalent to

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

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

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

Stage order s means that

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

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

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

where the vector powers are interpreted component by component.

This is equivalent to

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

Stage order s means that

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

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

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

where the vector powers are interpreted component by component.

This is equivalent to

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

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

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

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

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

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

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

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, $\ldots$, ξ_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?

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?

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

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

The question now is, how should λ be chosen?

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

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

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 $[0, 1]$.

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.

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.

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.

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:

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

It can be shown that for a SIRK method

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

Define the matrix T as follows:

$$T = \left[\begin{array}{ccccc} L_0(\xi_1) & L_1(\xi_1) & L_2(\xi_1) & \cdots & L_{s-1}(\xi_1) \\ L_0(\xi_2) & L_1(\xi_2) & L_2(\xi_2) & \cdots & L_{s-1}(\xi_2) \\ L_0(\xi_3) & L_1(\xi_3) & L_2(\xi_3) & \cdots & L_{s-1}(\xi_3) \\ \vdots & \vdots & \vdots & & \vdots \\ L_0(\xi_s) & L_1(\xi_s) & L_2(\xi_s) & \cdots & L_{s-1}(\xi_s) \end{array} \right]$$

It can be shown that for a SIRK method

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

There are two ways in which SIRK methods can be generalized

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

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

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

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

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

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

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

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

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

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

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

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

This allows us to locate the abscissae where we wish.

In “DESIRE” methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

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

A second generalization is to replace "order" by "effective order".

This allows us to locate the abscissae where we wish.

In "DESIRE" methods:

Diagonally Extended Singly Implicit Runge-Kutta methods using Effective order

these two generalizations are combined.

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