Parallel two-step Runge-Kutta methods

Helmut Podhaisky, podhaisky@mathematik.uni-halle.de

Martin–Luther–University Halle–Wittenberg, Germany

20 July 2006
Outline

Runge–Kutta and multi-step methods

Explicit parallel peer methods

Implicit parallel peer methods

Summary
Explicit Runge–Kutta methods for $y' = f(y)$

classical Runge–Kutta method (1904)

\[ Y_{m1} = y_{m-1} \]
\[ Y_{m2} = y_{m-1} + \frac{1}{2} hf(Y_{m1}) \]
\[ Y_{m3} = y_{m-1} + \frac{1}{2} hf(Y_{m2}) \]
\[ Y_{m4} = y_{m-1} + hf(Y_{m3}) \]
\[ y_m = y_{m-1} + \frac{h}{6} f(Y_{m1}) + \frac{h}{3} f(Y_{m2}) + \frac{h}{3} f(Y_{m3}) + \frac{h}{6} f(Y_{m4}) \]
Explicit Runge–Kutta methods for $y' = f(y)$

classical Runge–Kutta method (1904)

\[
\begin{align*}
Y_{m1} &= y_{m-1} \\
Y_{m2} &= y_{m-1} + \frac{1}{2}hf(Y_{m1}) \\
Y_{m3} &= y_{m-1} + \frac{1}{2}hf(Y_{m2}) \\
Y_{m4} &= y_{m-1} + hf(Y_{m3}) \\
y_m &= y_{m-1} + \frac{h}{6}f(Y_{m1}) + \frac{h}{3}f(Y_{m2}) + \frac{h}{3}f(Y_{m3}) + \frac{h}{6}f(Y_{m4})
\end{align*}
\]

Butcher scheme

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
1 & 0 & 0 & 1 \\
\frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
\]

- the computation is sequential
Explicit Runge–Kutta methods for $y' = f(y)$

classical Runge–Kutta method (1904)

\[ Y_{m1} = y_{m-1} \]
\[ Y_{m2} = y_{m-1} + \frac{1}{2} hf(Y_{m1}) \]
\[ Y_{m3} = y_{m-1} + \frac{1}{2} hf(Y_{m2}) \]
\[ Y_{m4} = y_{m-1} + hf(Y_{m3}) \]
\[ y_m = y_{m-1} + \frac{h}{6} f(Y_{m1}) + \frac{h}{3} f(Y_{m2}) + \frac{h}{3} f(Y_{m3}) + \frac{h}{6} f(Y_{m4}) \]

Butcher scheme

\[
\begin{array}{cccccc}
0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \\
\frac{1}{2} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
1 & 0 & 0 & 0 & 1
\end{array}
\]

- the computation is sequential
- $Y_{mk}$ is not an accurate approximation to $y(t_{m-1} + c_k h)$ leading to complex order conditions
- linear stability can be studied easily
- stepsize changes are trivial
Explicit Runge–Kutta methods, Dormand/Prince (1980)

order 5, much better accuracy than RK4

<table>
<thead>
<tr>
<th>0</th>
<th>1/5</th>
<th>1/5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/10</td>
<td>3/40</td>
<td>9/40</td>
</tr>
<tr>
<td>4/5</td>
<td>44/45</td>
<td>-56/15</td>
</tr>
<tr>
<td>9/8</td>
<td>19372/6561</td>
<td>-25360/2187</td>
</tr>
<tr>
<td>1</td>
<td>9017/3168</td>
<td>-355/33</td>
</tr>
<tr>
<td>1</td>
<td>35/384</td>
<td>0</td>
</tr>
<tr>
<td>35/384</td>
<td>0</td>
<td>500/1113</td>
</tr>
</tbody>
</table>

- widely used, Matlab: ode45
- not parallelizable
Bashforth and Adams (1883)

explicit multistep methods, e.g. order 4:

\[ y^{[m]} = y^{[m-1]} + \frac{h}{24} (55f_{m-1} - 59f_{m-2} + 37f_{m-3} - 9f_{m-4}) \]

- the computation in sequential
- the order conditions are rather simple, just interpolation
- linear stability is more difficult to analyze
- implementing a robust variable stepsize–variable order code is quite a challenge, Matlab ode113
Implicit methods

Runge–Kutta methods

- low stage order $\Rightarrow$ order reduction
- high stage order $\Rightarrow$ coupled iterations (and a good chance for some parallelization, bottleneck: LU-decomposition of the Jacobian)

Questions:

- What limitations can be overcome with more general methods? In particular: what about parallelism?
- Is it possible to be competitive in applications?

Multistep methods

- second Dahlquist barrier: an A-stable method cannot have order $p > 2$
- not parallelizable
Explicit parallel peer methods

Idea: re-use stages from the last step

\[
Y_m = \begin{bmatrix}
Y_{m1} \\
Y_{m2} \\
\vdots \\
Y_{ms}
\end{bmatrix}, \quad Y_{mk} = y(t_m + c_k h) + O(h^{p+1})
\]

Scheme: A and B full \( s \times s \) matrices

\[
Y_m = hAf(Y_{m-1}) + BY_{m-1}
\]

- the computation is completely parallel
- the order conditions are simple as long as \( p \leq s - 1 \) (interpolation, \( B = B(A) \))
- we want zero-stability with no parasitic roots, \( \sigma(B) = \{1, 0\} \)
- difficult: optimizing stability and variable stepsize behaviour
order conditions for $Y_m = hAf(Y_{m-1}) + BY_{m-1}$

Taylor series expansion

With $z = h\frac{d}{dt}$, we have $Y_m = \exp(cz)y(t_m)$ and

$$\exp(cz) = Az \exp((c - 1)z) + B \exp((c - 1)z) + O(z^{p+1})$$

which can be satisfied for $B$ easily if $p = s - 1$. 
order conditions for \( Y_m = hAf(Y_{m-1}) + BY_{m-1} \)

Taylor series expansion

With \( z = h\frac{d}{dt} \), we have \( Y_m = \exp(cz)y(t_m) \) and

\[
\exp(cz) = Az \exp((c-1)z) + B \exp((c-1)z) + O(z^{p+1})
\]

which can be satisfied for \( B \) easily if \( p = s - 1 \).

ordered by powers of \( z \)

\[
\exp(cz) \sim \begin{bmatrix}
1 & c_1 & \frac{c_1^2}{2} & \ldots & \frac{c_1^{s-1}}{s-1!} \\
1 & c_2 & \frac{c_2^2}{2} & \ldots & \frac{c_2^{s-1}}{s-1!} \\
\vdots & \vdots & \vdots & \ldots & \vdots \\
1 & c_s & \frac{c_s^2}{2} & \ldots & \frac{c_s^{s-1}}{s-1!}
\end{bmatrix}
\]
order conditions for $Y_m = hAf(Y_{m-1}) + BY_{m-1}$

Taylor series expansion
With $z = h\frac{d}{dt}$, we have $Y_m = \exp(cz)y(t_m)$ and

$$\exp(cz) = Az \exp((c-1)z) + B \exp((c-1)z) + O(z^{p+1})$$

which can be satisfied for $B$ easily if $p = s - 1$.

ordered by powers of $z$

$$\exp((c-1)z) \sim \begin{bmatrix} 1 & c_1 - 1 & \frac{(c-1)^2}{2} & \ldots & \frac{(c_1-1)^{s-1}}{s-1!} \\ 1 & c_2 - 1 & \frac{(c-1)^2}{2} & \ldots & \frac{(c_2-1)^{s-1}}{s-1!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & c_s - 1 & \frac{(c-1)^2}{2} & \ldots & \frac{(c_s-1)^{s-1}}{s-1!} \end{bmatrix}$$
order conditions for $Y_m = hAf(Y_{m-1}) + BY_{m-1}$

Taylor series expansion

With $z = h \frac{d}{dt}$, we have $Y_m = \exp(cz)y(t_m)$ and

$$\exp(cz) = Az \exp((c - 1)z) + B \exp((c - 1)z) + O(z^{p+1})$$

which can be satisfied for $B$ easily if $p = s - 1$.

ordered by powers of $z$

$$z \exp((c - 1)z) \sim \begin{bmatrix}
0 & 1 & c_1 - 1 & \ldots & \frac{(c_1-1)^{s-2}}{s-2!} \\
0 & 1 & c_2 - 1 & \ldots & \frac{(c_2-1)^{s-2}}{s-2!} \\
\vdots & \vdots & \vdots & \ldots & \vdots \\
0 & 1 & c_s - 1 & \ldots & \frac{(c_s-1)^{s-2}}{s-2!}
\end{bmatrix}$$
Explicit parallel peer methods, linear stability

Let \( y' = \lambda y \) and \( z = h\lambda \).
We obtain

\[ Y_m = M(z)Y_{m-1} = (B + zA)Y_{m-1} \]

- \( M(z) \) is called stability or amplification matrix.
- zero stability: \( B = M(0) \) must be power-bounded
- stability domain: \( \{ z : M(z) \text{ is power-bounded} \} \)
Explicit parallel peer methods, linear stability

Let \( y' = \lambda y \) and \( z = h\lambda \).

We obtain

\[
Y_m = M(z)Y_{m-1} = (B + zA)Y_{m-1}
\]

- \( M(z) \) is called stability or amplification matrix.
- zero stability: \( B = M(0) \) must be power-bounded
- stability domain: \( \{ z : M(z) \text{ is power-bounded} \} \)
- example \( s = 6 \)

compute boundary:

\[
M(z)v = \exp(\phi i)v
\]

leads to

generalized EV problem:

\[
zA = (\exp(\phi i)I - B)v
\]
Explicit parallel peer methods, optimization

inner loop: linear conditions

- order $p \leq s - 1$
- zero-parasitic eigenvalues: Schur-form $Q^T \hat{B} Q$ ist strictly upper triangular,
  $Q = (I + S)(I - S)^{-1}$ Cayley-transform, $S$ – free skew matrix
- linear least squares, e.g. $\|A\|_F$ and $\|B\|_F$

outer loop: nonlinear conditions

- $\rho(M(z))$, shape of the stability domain
- higher order errors $p \geq s$, super-convergence conditions

After all, the method has to be reformulated for variable stepsizes.
Explicit parallel peer methods, optimization

inner loop: linear conditions

- order $p \leq s - 1$
- zero-parasitic eigenvalues: Schur-form $Q^T \hat{B} Q$ ist strictly upper triangular,
  $Q = (I + S)(I - S)^{-1}$ Cayley-transform, $S$ – free skew matrix
- linear least squares, e.g. $\|A\|_F$ and $\|B\|_F$

outer loop: nonlinear conditions

- $\rho(M(z))$, shape of the stability domain
- higher order errors $p \geq s$, super-convergence conditions

After all, the method has to be reformulated for variable stepsizes.
$\Rightarrow$ work in progress
implicit parallel peer methods
Implicit parallel peer methods

Parallel+implicit

$Y_{mi}$ depends on $hf(Y_{mi})$

$$Y_m = Ghf(Y_m) + hAf(Y_{m-1}) + BY_{m-1}, \quad G = \text{diag}(g_1, \ldots, g_s)$$

Stiff accuracy

$M(z) = (I - zG)^{-1}(zA + B)$ vanishes for $z \to -\infty$ iff $A = 0$

Example: method PP3, 3 stages, order 2

L-stable, stiffly accurate

$c = (-0.1, 0.7, 1), \quad G = \text{diag}(1.7, 0.2670, 0.4454)$
Implicit parallel peer methods

parallel+implicit

$Y_{mi}$ depends on $hf(Y_{mi})$

$$Y_m = Ghf(Y_m) + hAf(Y_{m-1}) + BY_{m-1}, \quad G = \text{diag}(g_1, \ldots, g_s)$$

stiff accuracy

$M(z) = (I - zG)^{-1}(zA + B)$ vanishes for $z \to -\infty$ iff $A = 0$

example: method PP3, 3 stages, order 2

L-stable, stiffly accurate

$c = (-0.1, 0.7, 1), \quad G = \text{diag}(1.7, 0.2670, 0.4454)$

parasitic roots: $B = \begin{bmatrix} -0.21 & 6.79 & -5.57 \\ 0.27 & -2.46 & 3.18 \\ 0.31 & -2.99 & 3.68 \end{bmatrix}$, $\sigma(B) = \{1, -0.0005 \pm 0.2202i\}$ (for constant stepsize)
Implementation

- one Newton iteration per stage, linearly implicit method
- Krylov subspace $\mathcal{K} := \text{span}\{b, Tb, \ldots, T^{\kappa-1}\}$, with $T = f_y$
  Find $\bar{x}$ for $(I - \gamma h T)x = b$ with $\bar{x} \in \mathcal{K}$ and $(A\bar{x} - b) \perp \mathcal{K}$.
- Fortran 90 using OpenMP for loop parallelization
- experiments on a SunFire (shared memory)
- spatial discretizations with finite differences and finite volumes, respectively
Example 1: Diffu, \( u_t = u_{xx} + u_{yy} + f(t, x, y) \)

with \( 0 \leq t, x, y \leq 1 \) and exact solution
\[
\sin(\pi x) \sin(\pi y)(1 + 4xy \sin t)
\]
Example 1: Diffu, \( u_t = u_{xx} + u_{yy} + f(t, x, y) \)

with \( 0 \leq t, x, y \leq 1 \) and exact solution

\[
\sin(\pi x) \sin(\pi y)(1 + 4xy \sin t)
\]
Example 1: Diffu, $u_t = u_{xx} + u_{yy} + f(t, x, y)$

with $0 \leq t, x, y \leq 1$ and exact solution

$$\sin(\pi x) \sin(\pi y)(1 + 4xy \sin t)$$
Example 1: Diffu, \( u_t = u_{xx} + u_{yy} + f(t, x, y) \)

with \( 0 \leq t, x, y \leq 1 \) and exact solution
\[
\sin(\pi x) \sin(\pi y)(1 + 4xy \sin t)
\]
Example 2: Radiation-Diffusion


\[ E_t = \nabla \cdot (D_r \nabla E) + \sigma (T^4 - E), \quad D_r = (3\sigma + (1/E)|\partial E/\partial x|)^{-1}, \]
\[ T_t = \nabla \cdot (D_t \nabla T) - \sigma (T^4 - E), \quad D_t = kT^{5/2}, \quad k = 0 \text{ or } k = 0.1 \]

strongly nonlinear, conservative second-order discretization
Example 2: Radiation-Diffusion


\[ E_t = \nabla \cdot (D_r \nabla E) + \sigma (T^4 - E), \quad D_r = (3\sigma + (1/E)|\partial E/\partial x|)^{-1}, \]

\[ T_t = \nabla \cdot (D_t \nabla T) - \sigma (T^4 - E), \quad D_t = kT^{5/2}, \quad k = 0 \text{ or } k = 0.1 \]

strongly nonlinear, conservative second-order discretization
Example 2: Radiation-Diffusion


\[ E_t = \nabla \cdot (D_r \nabla E) + \sigma (T^4 - E), \quad D_r = (3\sigma + (1/E)\|\partial E/\partial x\|)^{-1}, \]

\[ T_t = \nabla \cdot (D_t \nabla T) - \sigma (T^4 - E), \quad D_t = kT^{5/2}, \quad k = 0 \text{ or } k = 0.1 \]

strongly nonlinear, conservative second-order discretization
Example 2: Radiation-Diffusion


\[ E_t = \nabla \cdot (D_r \nabla E) + \sigma (T^4 - E), \quad D_r = (3\sigma + (1/E)|\partial E/\partial x|)^{-1}, \]

\[ T_t = \nabla \cdot (D_t \nabla T) - \sigma (T^4 - E), \quad D_t = kT^{5/2}, \quad k = 0 \text{ or } k = 0.1 \]

strongly nonlinear, conservative second-order discretization
Summary

- classical methods cannot be parallelized, we need multi-stage, multi-step methods
- methods with parallel stages scale can be implemented easily on shared memory computers (with, say, 2 to 8 CPUs/cores)
- order conditions can be satisfied by interpolation, however, super-convergence is more difficult to achieve
- the main difficulty lies in finding robust methods with small coefficients and moderate error constants
- sequential general linear methods are worth to be studied, too

  John Butcher (NZ), Adrian Hill (UK), Caren Tischendorf (Matheon/Colone), group in Halle, . . .