An intrinsically adaptive formulation of multistep methods

Carmen Arévalo
with G. Söderlind, F. Mohammadi, C. Führer

Lund University

Farkas Miklos Seminar of the Technical University Budapest
September 20, 2018
To approximate the solution of an IVP

\[ \frac{dx}{dt} = f(t, x), \quad x(t_0) = x_0, \quad t \in [t_0, t_f] \]

a \textit{k-step} method uses \( k \) previous approximations

\[ x_{n-i} \approx x(t_{n-i}), \quad i = 1 : k, \]

\[ x_n = \alpha_{k-1}x_{n-1} + \cdots + \alpha_0x_{n-k} + h_n(\beta_k f_n + \cdots + \beta_0 f_{n-k}) \]

\[ h_i = t_i - t_{i-1}, \quad f_i = f(t_i, x_i) \]

Adaptivity: choose \( h_n \) to attain prescribed accuracy.
Adaptivity

Why do we need adaptivity?

- accuracy
- efficiency
- stability
Adaptivity

Why do we need adaptivity?

▶ accuracy
▶ efficiency
▶ stability

Adaptivity aims to control the error at each integration step

▶ estimate the local error
▶ choose $h_n$ to keep the error at an assigned value (tolerance)
Order of a multistep method

Approximation: \( x_n \approx x(t_n) \) and \( x'_n \approx f(t_n, x(t_n)) \)

When is a method said to be of order \( q \)?
Order of a multistep method

Approximation: \( x_n \approx x(t_n) \) and \( x'_n \approx f(t_n, x(t_n)) \)

When is a method said to be of order \( q \)?

For any ODE whose solution \( x \) is a polynomial of degree \( q \), the method recovers the exact solution:

\[ x_n = x(t_n) \]
Order of a multistep method

Approximation: \( x_n \approx x(t_n) \) and \( x'_n \approx f(t_n, x(t_n)) \)

When is a method said to be of order \( q \)?

For any ODE whose solution \( x \) is a polynomial of degree \( q \), the method recovers the exact solution:

\[ x_n = x(t_n) \]

Lower order methods \( \rightarrow \) larger stability regions
Higher order methods \( \rightarrow \) allow larger step-sizes
Polynomial of a method with \( k \) steps and order \( q \)

To advance a step of an order \( q \) method we construct a method polynomial \( P_n \in \mathcal{P}_q \) using previously calculated values and define

\[
x_n = P_n(t_n)
\]

The polynomial of a \( k \)-step method will depend on the last \( k \) approximated solutions and their derivatives.
Maximal order multistep methods

Three types of high order $k$-step methods (Dahlquist 1st barrier):

<table>
<thead>
<tr>
<th>type</th>
<th>max order</th>
</tr>
</thead>
<tbody>
<tr>
<td>implicit</td>
<td>$k + 1$</td>
</tr>
<tr>
<td>explicit</td>
<td>$k$</td>
</tr>
</tbody>
</table>

For **Nonstiff** problems:

- $E_k$: *Explicit $k$-step*, order $q = k$, e.g. Adams–Bashforth

- $I_k^+$: *Implicit $k$-step*, order $q = k + 1$, e.g. Adams–Moulton

For **Stiff** problems:

- $I_k$: *Implicit $k$-step*, order $q = k$, e.g. BDF methods
Method polynomial for an $E_k$ type method

Adams-Bashforth-3: explicit, $k = 3$, order $q = 3$, $P_n \in \mathcal{P}_3$
Method polynomial for an $E_k$ type method

Adams-Bashforth-3: explicit, $k = 3$, order $q = 3$, $P_n \in \mathcal{P}_3$
Method polynomial for an $E_k$ type method

Adams-Bashforth-3: explicit, $k = 3$, order $q = 3$, $P_n \in \mathcal{P}_3$

Construction of AB3: $x' = f(t,x)$

$P'(t_0) = f(t_0,x_0)$
$P'(t_1) = f(t_1,x_1)$
$P'(t_2) = f(t_2,x_2)$
$P(t_2) = x_2$
$x_3 = P(t_3)$
Method polynomial for an $I_k$ type method

BDF-3: implicit, $k = 3$, order $q = 3$, $P_n \in \mathcal{P}_3$
Method polynomial for an $I_k$ type method

BDF-3: implicit, $k = 3$, order $q = 3$, $P_n \in \mathcal{P}_3$

Construction of BDF3: $x' = f(t, x)$

- $P(t_0) = x_0$
- $P(t_1) = x_1$
- $P(t_2) = x_2$
- $P'(t_3) = f(t_3, P(t_3))$
Method polynomial for an $I_k$ type method

BDF-3: implicit, $k = 3$, order $q = 3$, $P_n \in P_3$

Construction of BDF3: $x' = f(t, x)$

- $P(t_0) = x_0$
- $P(t_1) = x_1$
- $P(t_2) = x_2$
- $x_3 = P(t_3)$
- $P'(t_3) = f(t_3, P(t_3))$
Construction of $E_k$ method of order $q = k$

Interpolation conditions:

$$P(t_{n-i}) = x_{n-i}, \quad i = 1, \ldots, k$$

Collocation conditions:

$$\dot{P}(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k$$

$2k$ possible conditions to define $P \in \mathcal{P}_k$: choose $k + 1$ conditions

$$\binom{2k}{k+1} = \frac{(2k)!}{(k-1)!(k+1)!} \quad \text{e.g.} \quad k = 3 \quad \Rightarrow \quad 15 \text{ possible methods}$$
Construction of $E_k$ method of order $q = k$

Interpolation conditions:

$$P(t_{n-i}) = x_{n-i}, \quad i = 1, \ldots, k$$

Collocation conditions:

$$\dot{P}(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k$$

$2k$ possible conditions to define $P \in \mathcal{P}_k$: choose $k + 1$ conditions

$$\binom{2k}{k+1} = \frac{(2k)!}{(k-1)!(k+1)!} \quad e.g. \ k = 3 \quad \Rightarrow \quad 15 \text{ possible methods}$$

But there are infinitely many explicit $k$-step, order $k$ methods!
$P_n$ for classical $k$-step formulas

\begin{align*}
\text{AB}k & \quad P_n(t_{n-1}) = x_{n-1} \\
\dot{P}_n(t_{n-i}) & = x'_{n-i}, \quad i = 1, \ldots, k
\end{align*}
$P_n$ for classical $k$-step formulas

AB$k$

\[
P_n(t_{n-1}) = x_{n-1}
\]
\[
\dot{P}_n(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k
\]

AM$k$

\[
P_n(t_{n-1}) = x_{n-1}
\]
\[
\dot{P}_n(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k
\]
\[
\dot{P}_n(t_n) = f(t_n, P_n(t_n))
\]
$P_n$ for classical $k$-step formulas

**AB$k$**

\[
P_n(t_{n-1}) = x_{n-1}
\]
\[
\dot{P}_n(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k
\]

**AM$k$**

\[
P_n(t_{n-1}) = x_{n-1}
\]
\[
\dot{P}_n(t_{n-i}) = x'_{n-i}, \quad i = 1, \ldots, k
\]
\[
\dot{P}_n(t_n) = f(t_n, P_n(t_n))
\]

**BDF$k$**

\[
P_n(t_{n-i}) = x_{n-i}, \quad i = 1, \ldots, k
\]
\[
\dot{P}_n(t_n) = f(t_n, P_n(t_n))
\]
Solution: slack conditions

Do not insist on interpolation/collocation conditions; allow a slack:

\[ s_{n-i} = P_n(t_{n-i}) - x_{n-i} \quad \text{(state slack)} \]
\[ s'_{n-i} = \dot{P}_n(t_{n-i}) - x'_{n-i} \quad \text{(derivative slack)} \]

and combine each slack pair into a slack balance condition:

\[ a s_{n-i} + b h_{n-i} s'_{n-i} = 0 \]
Solution: slack conditions

Do not insist on interpolation/collocation conditions; allow a slack:

\[
\begin{align*}
    s_{n-i} &= P_n(t_{n-i}) - x_{n-i} \quad \text{(state slack)} \\
    s'_{n-i} &= \dot{P}_n(t_{n-i}) - x'_{n-i} \quad \text{(derivative slack)}
\end{align*}
\]

and combine each slack pair into a slack balance condition:

\[
a s_{n-i} + b h_{n-i} s'_{n-i} = 0
\]

simplified to

\[
\cos \theta \ s_{n-i} + \sin \theta \ h_{n-i} s'_{n-i} = 0, \quad \theta \in (-\pi/2, \pi/2]
\]
Each method type is characterized by its **structural conditions**

\[
\mathbf{E}_k \quad \begin{cases} 
  s_{n-1} = 0 & \text{(interpolation condition)} \\
  s'_{n-1} = 0 & \text{(explicit collocation condition)}
\end{cases}
\]

\[
\mathbf{I}^+_k \quad \begin{cases} 
  \dot{P}_n(t_n) = f(t_n, P_n(t_n)) & \text{(implicit collocation condition)} \\
  s_{n-1} = 0 & \text{(interpolation condition)} \\
  s'_{n-1} = 0 & \text{(explicit collocation condition)}
\end{cases}
\]

\[
\mathbf{I}_k \quad \begin{cases} 
  \dot{P}_n(t_n) = f(t_n, P_n(t_n)) & \text{(implicit collocation condition)} \\
  \cos \theta_0 s_{n-1} + \sin \theta_0 h_{n-1} s'_{n-1} = 0 & \text{(slack balance condition)} \\
  \theta_0 \in (-\pi/2, \pi/2]
\end{cases}
\]
To complete the number of required conditions we impose $k - 1$ additional *slack balance conditions*

$$\cos \theta_i s_{n-i-1} + \sin \theta_i h_{n-i-1}s'_{n-i-1} = 0$$

with $\theta_i \in (-\pi/2, \pi/2]$, $i = 1 : k - 1$
To complete the number of required conditions we impose $k - 1$ additional *slack balance conditions*

$$\cos \theta_i \ s_{n-i-1} + \sin \theta_i \ h_{n-i-1}s'_{n-i-1} = 0$$

with $\theta_i \in (-\pi/2, \pi/2]$, \quad $i = 1 : k - 1$

Method parameters $\theta_i$ uniquely define a method.
To complete the number of required conditions we impose \( k - 1 \) additional \textit{slack balance conditions}

\[
\cos \theta_i s_{n-i-1} + \sin \theta_i h_{n-i-1}s'_{n-i-1} = 0
\]

with \( \theta_i \in (-\pi/2, \pi/2], \quad i = 1 : k - 1 \)

Method parameters \( \theta_i \) uniquely define a method.

The parameter set \( \{\theta_i\} \) is \textit{grid independent}. 
Theorem: Each linear multistep method of type $E_k$, $I_k$, or $I_k^+$ can be represented by a single polynomial in $[t_{n-1}, t_n]$, with $k - 1$, $k$, and $k - 1$ parameters, respectively.
**Theorem:** Each linear multistep method of type $E_k$, $I_k$, or $I_k^+$ can be represented by a single polynomial in $[t_{n-1}, t_n]$, with $k - 1$, $k$, and $k - 1$ parameters, respectively.

We can implement every maximal order method by constructing its method polynomial, $P_n$, advancing the integration at each step: $x_n = P_n(t_n)$. 
**Theorem**: Each linear multistep method of type $E_k$, $I_k$, or $I_k^+$ can be represented by a single polynomial in $[t_{n-1}, t_n]$, with $k - 1$, $k$, and $k - 1$ parameters, respectively.

We can implement every maximal order method by constructing its *method polynomial*, $P_n$, advancing the integration at each step: $x_n = P_n(t_n)$.

The solver (MODES) includes every possible multistep method of maximal order, stiff and non-stiff, implicit and explicit.
### Parametric formulation of 0-stable multistep methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Order</th>
<th>$I_k$ method parameters</th>
<th>$\tan(\theta_j), \ j = 0 : k - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDF$k$</td>
<td>$k \leq 6$</td>
<td>${0}_{0:k-1}$</td>
<td></td>
</tr>
<tr>
<td>Kregel</td>
<td>3</td>
<td>154/543</td>
<td>-11/78</td>
</tr>
<tr>
<td>Rockswold</td>
<td>3</td>
<td>73/350</td>
<td>71/200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Order</th>
<th>$I_k^+$ method parameters</th>
<th>$\cot(\theta_j), \ j = 1 : k - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM$k$</td>
<td>$k + 1$</td>
<td>${0}_{1:k-1}$</td>
<td></td>
</tr>
<tr>
<td>dcBDF$k$</td>
<td>$k + 1$</td>
<td>${(k + 1)/(j + 1)}_{1:k-1}$</td>
<td></td>
</tr>
<tr>
<td>Milne2</td>
<td>4</td>
<td>$3$</td>
<td></td>
</tr>
<tr>
<td>Milne4</td>
<td>5</td>
<td>$15/4$</td>
<td>$0$</td>
</tr>
<tr>
<td>IDC23</td>
<td>4</td>
<td>$6/7$</td>
<td>$0$</td>
</tr>
<tr>
<td>IDC24</td>
<td>5</td>
<td>$15/26$</td>
<td>$0$</td>
</tr>
<tr>
<td>IDC34</td>
<td>5</td>
<td>$5/4$</td>
<td>$20/33$</td>
</tr>
<tr>
<td>IDC45</td>
<td>6</td>
<td>$45/28$</td>
<td>$10/11$</td>
</tr>
<tr>
<td>IDC56</td>
<td>7</td>
<td>$84/43$</td>
<td>$7/6$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Order</th>
<th>$E_k$ method parameters</th>
<th>$\cot(\theta_j), \ j = 1 : k - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB$k$</td>
<td>$k$</td>
<td>${0}_{1:k-1}$</td>
<td></td>
</tr>
<tr>
<td>EDF$k$</td>
<td>$k$</td>
<td>${1/(j + 1)}_{1:k-1}$</td>
<td></td>
</tr>
<tr>
<td>Explicit Euler</td>
<td>1</td>
<td>$\infty$</td>
<td></td>
</tr>
<tr>
<td>Midpoint</td>
<td>2</td>
<td>$-3/2$</td>
<td>$0$</td>
</tr>
<tr>
<td>Nyström3</td>
<td>3</td>
<td>$-3/5$</td>
<td>$0$</td>
</tr>
<tr>
<td>Nyström4</td>
<td>4</td>
<td>$-45/133$</td>
<td>$0$</td>
</tr>
<tr>
<td>Nyström5</td>
<td>5</td>
<td>$3/14$</td>
<td>$0$</td>
</tr>
<tr>
<td>EDC22</td>
<td>3</td>
<td>$6/49$</td>
<td>$0$</td>
</tr>
<tr>
<td>EDC23</td>
<td>4</td>
<td>$2/7$</td>
<td>$4/39$</td>
</tr>
<tr>
<td>EDC33</td>
<td>4</td>
<td>$10/53$</td>
<td>$10/219$</td>
</tr>
<tr>
<td>EDC24</td>
<td>5</td>
<td>$90/1121$</td>
<td>$0$</td>
</tr>
<tr>
<td>EDC34</td>
<td>5</td>
<td>$10/53$</td>
<td>$10/219$</td>
</tr>
<tr>
<td>EDC45</td>
<td>6</td>
<td>$45/193$</td>
<td>$10/121$</td>
</tr>
</tbody>
</table>
Experimental software platform, **MODES** offers:

- Any multistep method of type $E_k$, $I_k$ or $I_k^+$
- Constant or variable step-sizes
- Constant or variable order
- Initial step-size and starters (classical Gear, Runge-Kutta)
- Error per step or error per unit step
- Step-size controllers (several PI and low pass digital filters)
- Upper and lower bounds for step-size ratios
Results for the Oregonator problem solved with MODES-BDF5
Compare different methods under exact same conditions: AM3 vs. another $I_3^+$ method. The second method is twice as accurate.
State-of-the-art error control: Matlab BDF1-5 vs. MODES
BDF1-5 implementation

Van der Pol $\mu = 1200$: solution

Step-size sequence
Accuracy/Work proportionality for **MODES** and **Matlab’s ode15s**.
Van der Pol with variable order BDF

Step-sizes

Absolute errors (measured by $\| \cdot \|$)

Orders
Various step-size ratio bounds: none, 160%, 10% and 0.05%.

The controller in **MODES** keeps the step-size ratios in check to maintain stability.
An application to SSP multistep methods

Strong stability preserving methods avoid instabilities when solving ODEs arising from the semi-discretization of hyperbolic PDEs.

\[ \dot{y} = F(y), \quad y(t_0) = y_0, \quad t \in [t_0, t_f] \]

with the property

\[ \|y + hF(y)\| \leq \|y\| \quad \text{for all } y \text{ and } h \leq h^* \]

solved by explicit multistep method

\[ y_n = \sum_{i=1}^{k} (\alpha_i y_{n-i} + h\beta_i F(y_{n-i})) , \quad \text{with } \alpha_i, \beta_i \geq 0 \]

SSP if \( \|y_n\| \leq \max\{\|y_{n-1}\|, \ldots, \|y_{n-k}\|\} \) for \( 0 < h \leq Ch^* \)
Explicit SSP multistep methods

- $\alpha_i, \beta_i \geq 0$
- $0 < h \leq Ch^*$
- SSP constant: $C = \min_i \{\alpha_i / \beta_i\}$
- $q < k$
- Zero coefficients of fixed step-size formula should be preserved by variable step-size extension

Adaptive explicit SSP multistep methods: formulation for lower order methods that preserves pattern of zero coefficients
Hadjimichael et al. (2016): first variable step-size optimal SSP methods of orders 2 and 3
Formulation of optimal SSP methods

Procedure to construct optimal SSP\((k, q)\) method

- Take \(s_{n-1} = 0\) and \(s'_{n-1} = 0\)
- Take \(s_{n-i} + h_{n-i} \frac{\beta_i}{\alpha_i} s'_{n-i} = 0\) whenever \(\alpha_i \neq 0\), \(1 < i < k\)
- Take \(s_{n-k} = 0\)
- If \(q\) is odd, also add \(s'_{n-k} = 0\)

The method parameters are \(\tau_i = \frac{\beta_i}{\alpha_i}\)
Example of optimal explicit 8-step SSP method of order 5

Its nonzero coefficients:

\[
\alpha_1 = \frac{1360}{4363}, \quad \alpha_4 = \frac{233}{2112}, \quad \alpha_5 = \frac{2323}{10831}, \quad \alpha_8 = \frac{896}{2465}
\]

\[
\beta_1 = \frac{275}{128}, \quad \beta_4 = \frac{1044}{1373}, \quad \beta_5 = \frac{6661}{4506}, \quad \beta_8 = \frac{1781}{5144}
\]

\[
s_{n-1} = 0 \\
s'_{n-1} = 0 \\
s_{n-4} + h_{n-4}\tau_4s'_{n-4} = 0 \\
s_{n-5} + h_{n-5}\tau_5s'_{n-5} = 0 \\
s_{n-8} = 0 \\
s'_{n-8} = 0
\]

\[
\tau_4 = \frac{\beta_4}{\alpha_4}, \quad \tau_5 = \frac{\beta_5}{\alpha_5}
\]
An application to differential-algebraic systems

DAEs are differential equations coupled with algebraic constraints

\[ \dot{x} = f(x, \lambda) \]
\[ 0 = g(x, \lambda) \]

\( \lambda \) is called the algebraic variable
An application to differential-algebraic systems

DAEs are differential equations coupled with algebraic constraints

\[
\dot{x} = f(x, \lambda) \\
0 = g(x, \lambda)
\]

\(\lambda\) is called the algebraic variable

The \textit{index} characterizes the difficulty of a DAE.

Index 2 Euler-Lagrange DAE in multibody dynamics:

\[
\dot{x} = f(x) - G(x)^T \lambda \\
0 = g(x)
\]

with \(G(x) = \partial g / \partial x\), \(G(x)G(x)^T\) invertible
Generating polynomials of a multistep method: $\rho$, $\sigma$

$$
\sum_{j=0}^{k} \alpha_{k-j} x_{n-j} = h \sum_{j=0}^{k} \beta_{k-j} f_{n-j}
$$

As difference operators

$$
\rho x_n = \sum_{i=0}^{k} \alpha_{k-i} x_{n-i}, \quad \sigma f_n = \sum_{i=0}^{k} \beta_{k-i} f_{n-i}
$$

As generating polynomials:

$$
\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j
$$
Generating polynomials of a multistep method: \( \rho, \sigma \)

\[
\sum_{j=0}^{k} \alpha_{k-j} x_{n-j} = h \sum_{j=0}^{k} \beta_{k-j} f_{n-j}
\]

As difference operators

\[
\rho x_n = \sum_{i=0}^{k} \alpha_{k-i} x_{n-i}, \quad \sigma f_n = \sum_{i=0}^{k} \beta_{k-i} f_{n-i}
\]

As generating polynomials:

\[
\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j
\]

Requirement for convergence:

- ODE methods: roots of \( \rho(\zeta) \) on or within the unit circle; those on the unit circle are simple
Generating polynomials of a multistep method: $\rho$, $\sigma$

$$ \sum_{j=0}^{k} \alpha_{k-j} x_{n-j} = h \sum_{j=0}^{k} \beta_{k-j} f_{n-j} $$

As difference operators

$$ \rho x_n = \sum_{i=0}^{k} \alpha_{k-i} x_{n-i}, \quad \sigma f_n = \sum_{i=0}^{k} \beta_{k-i} f_{n-i} $$

As generating polynomials:

$$ \rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j $$

Requirement for convergence:

- ODE methods: roots of $\rho(\zeta)$ on or within the unit circle; those on the unit circle are simple
- methods for index 2 DAEs: roots of $\sigma(\zeta)$ inside the unit circle
Consequence of $\sigma$ not satisfying the strict root condition?

Instability in algebraic variables
Consequence of $\sigma$ not satisfying the strict root condition?

Instability in algebraic variables

Implicit methods with roots of $\sigma(\zeta)$ inside the unit circle

- e.g. BDF ($k$-step, order $k$)
- no implicit $k$-step method of order $k + 1$ (e.g. Adams-Moulton)
Consequence of $\sigma$ not satisfying the strict root condition?

Instability in algebraic variables

Implicit methods with roots of $\sigma(\zeta)$ inside the unit circle

- e.g. BDF ($k$-step, order $k$)
- no implicit $k$-step method of order $k + 1$ (e.g. Adams-Moulton)

Solution: $\beta$-blocking

- Modify $\sigma$ to move roots into unit circle
- New operator $\sigma + \tau$, with roots inside unit circle, should only affect algebraic variables
- $\tau$ should disturb order as little as possible
Construction of $\beta$-blocker operator

- Reduce order as little as possible by taking $\tau = c \nabla^k$
- Keep $\sigma$ for differential variables
- $\sigma + \tau$ for algebraic variables

Order:
$k + 1$ for differential variable, $k$ for algebraic variable
\( \beta \)-blocking published 1997–2000 as fixed step-size technique

- \( \beta \)-blocked methods
- Parametric formulation of multistep methods
- Adaptive \( \beta \)-blocked multistep methods
Standard formulation of index-2 Euler-Lagrange DAE

\[ P_n \in \mathcal{P}_{k+1} \]

\[ P'_n(t_n) = f(P_n(t_n)) - G(P_n(t_n))^T \lambda_n \]
\[ P_n(t_{n-1}) = x_{n-1} \]
\[ P'_n(t_{n-1}) = f(x_{n-1}) - G(x_{n-1})^T \lambda_{n-1} \]

\[ \cos \theta_{j-1} s_{n-j} + h_{n-j} \sin \theta_{j-1}(s'_{n-j} + G(x_{n-j})^T \lambda_{n-j}) = 0 \]
\[ g(P_n(t_n)) = 0 \]

for \( j = 2 : k \).

Then \( x_n := P_n(t_n) \).
\[ P_n \in \mathcal{P}_{k+1} \text{ and } Q_n \in \mathcal{P}_k \]

\[ P'_n(t_n) = f(P_n(t_n)) - G(P_n(t_n))^T(Q_n(t_n) + \hat{c} h_{n-1}^k Q_n^{(k)}(t_n)) \]

\[ P_n(t_{n-1}) = x_{n-1} \]

\[ P'_n(t_{n-1}) = f(x_{n-1}) - G(x_{n-1})^T Q_n(t_{n-1}) \]

\[ Q_n(t_{n-j}) = \lambda_{n-j} \]

\[ \cos \theta_{j-1} s_{n-j} + h_{n-j} \sin \theta_{j-1} (s'_{n-j}) + G(x_{n-j})^T \lambda_{n-j} = 0 \]

\[ g(P_n(t_n)) = 0 \]

For \( j = 2 : k \), \( \hat{c} = c \beta_k^{-1} \).

Set \( x_n := P_n(t_n) \) and \( \lambda_n := Q_n(t_n) \).
\(\beta\)-blocked AM2

Effect of standard PI controller

![Graph showing the effect of a standard PI controller.](image)

- The graph on the left shows the step-size over time. The step-size increases rapidly at first and then plateaus.
- The graphs on the right display the behavior of variables \(x\) and \(\lambda\) over time. Both variables show a slow, steady increase, with \(x\) reaching a higher value than \(\lambda\).
Effect of low-pass digital filter controller
4–step $\beta$–blocked Adams–Moulton for nonlinear pendulum

Low-pass filter controller: error-tolerance proportionality

![Graph showing step-size vs. time and absolute error vs. tolerance]

- Step-size vs. time: The graphs show the step-size for different values of time $t$.
- Absolute error vs. tolerance: The graph illustrates the relationship between absolute error and tolerance, showing a linear proportionality.
Results

A unified variable step-size formulation for all explicit and implicit (stiff and nonstiff) methods of maximal order
Results

A unified variable step-size formulation for all explicit and implicit (stiff and nonstiff) methods of maximal order

A unified variable step-size formulation for explicit SSP multistep methods In particular, a straightforward procedure for the construction of optimal SSP methods
Results

A unified variable step-size formulation for all explicit and implicit (stiff and nonstiff) methods of maximal order

A unified variable step-size formulation for explicit SSP multistep methods  In particular, a straightforward procedure for the construction of optimal SSP methods

A unified variable step-size formulation of $\beta$-blocked methods for index 2 Euler-Lagrange DAEs
Results

A unified variable step-size formulation for all explicit and implicit (stiff and nonstiff) methods of maximal order

A unified variable step-size formulation for explicit SSP multistep methods In particular, a straightforward procedure for the construction of optimal SSP methods

A unified variable step-size formulation of $\beta$-blocked methods for index 2 Euler-Lagrange DAEs

MODES: a comprehensive multistep solver that uses these new formulations; this allows experimentation with adaptive multistep methods.
Köszönöm