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 *k-step* method uses k previous approximations $x_{n-i} \approx x(t_{n-i}), \quad i = 1:k,$

$$x_n = \alpha_{k-1}x_{n-1} + \cdots + \alpha_0 x_{n-k} + h_n(\beta_k f_n + \cdots + \beta_0 f_{n-k})$$
$$h_i = t_i - t_{i-1}, \qquad f_i = f(t_i, x_i)$$

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):

type	max order
implicit	$k+1$
explicit	k

For **Nonstiff** problems:

- $\mathbf{E}_k$: *Explicit k-step*, order $q = k$, e.g. Adams–Bashforth
- $\mathbf{I}_k^+$: *Implicit k-step*, order $q = k + 1$, e.g. Adams–Moulton

For **Stiff** problems:

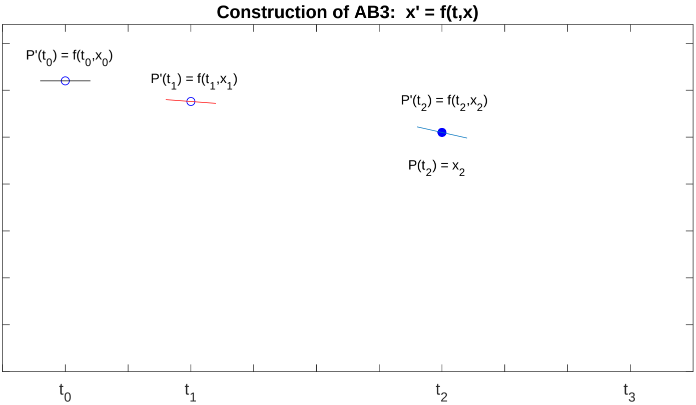
- $\mathbf{I}_k$: *Implicit k-step*, order $q = k$, e.g. BDF methods

Method polynomial for an E_k type method

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

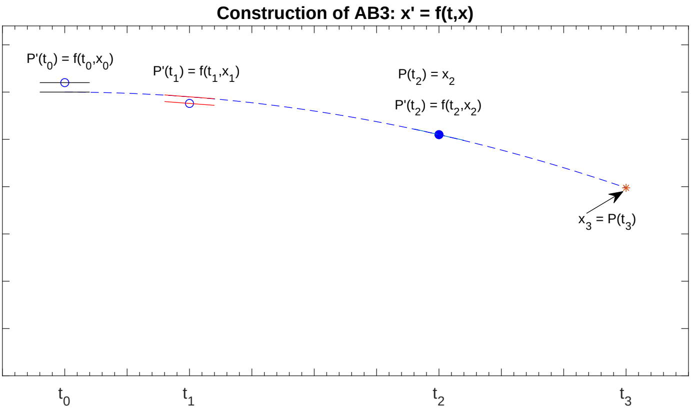
Method polynomial for an E_k type method

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 $q = 3$, $P_n \in \mathcal{P}_3$

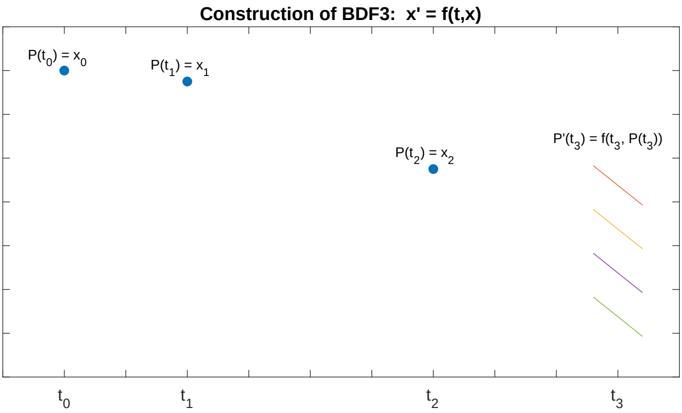


Method polynomial for an I_k type method

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

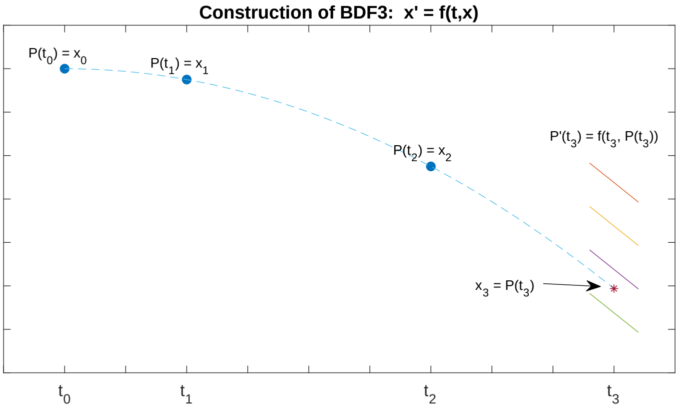
Method polynomial for an I_k type method

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



Method polynomial for an I_k type method

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



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

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

But there are infinitely many explicit k-step, order k methods!

P_n for classical k-step formulas

ABk

$$\begin{aligned} P_n(t_{n-1}) &= x_{n-1} \\ \dot{P}_n(t_{n-i}) &= x'_{n-i}, \quad i = 1, \dots, k \end{aligned}$$

P_n for classical k-step formulas

ABk

$$\begin{aligned} P_n(t_{n-1}) &= x_{n-1} \\ \dot{P}_n(t_{n-i}) &= x'_{n-i}, \quad i = 1, \ldots, k \end{aligned}$$

AMk

$$\begin{aligned} P_n(t_{n-1}) &= x_{n-1} \\ \dot{P}_n(t_{n-i}) &= x'_{n-i}, \quad i = 1, \ldots, k \\ \dot{P}_n(t_n) &= f(t_n, P_n(t_n)) \end{aligned}$$

P_n for classical k-step formulas

ABk

$$\begin{aligned} P_n(t_{n-1}) &= x_{n-1} \\ \dot{P}_n(t_{n-i}) &= x'_{n-i}, \quad i = 1, \ldots, k \end{aligned}$$

AMk

$$\begin{aligned} P_n(t_{n-1}) &= x_{n-1} \\ \dot{P}_n(t_{n-i}) &= x'_{n-i}, \quad i = 1, \ldots, k \\ \dot{P}_n(t_n) &= f(t_n, P_n(t_n)) \end{aligned}$$

BDFk

$$\begin{aligned} P_n(t_{n-i}) &= x_{n-i}, \quad i = 1, \ldots, k \\ \dot{P}_n(t_n) &= f(t_n, P_n(t_n)) \end{aligned}$$

Solution: slack conditions

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

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

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:

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

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

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

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

$$\mathbf{I}_k \quad \begin{cases} \dot{P}_n(t_n) = f(t_n, P_n(t_n)) & \text{(implicit collocation condition)} \\ \cos\theta_0\, s_{n-1} + \sin\theta_0\, h_{n-1}\, s'_{n-1} = 0 & \text{(slack balance condition)} \\ \quad \theta_0 \in (-\pi/2, \pi/2] \end{cases}$$

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

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 θ_i uniquely define a method.

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 θ_i uniquely define a method.

The parameter set $\{\theta_i\}$ is grid independent.

Theorem: *Each linear multistep method of type* $\mathbf{E}_k$, $\mathbf{I}_k$, or $\mathbf{I}_k^+$ *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* $\mathbf{E}_k$, $\mathbf{I}_k$, or $\mathbf{I}_k^+$ *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* $\mathbf{E}_k$, $\mathbf{I}_k$, or $\mathbf{I}_k^+$ *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

Method	**Order**	I_k **method parameters** $\tan(\theta_j)$, $j = 0 : k-1$				
BDFk	$k \leq 6$	$\{0\}_{0:k-1}$				
Kregel	3	154/543	-11/78	0		
Rockswold	3	73/350	71/200	∞		

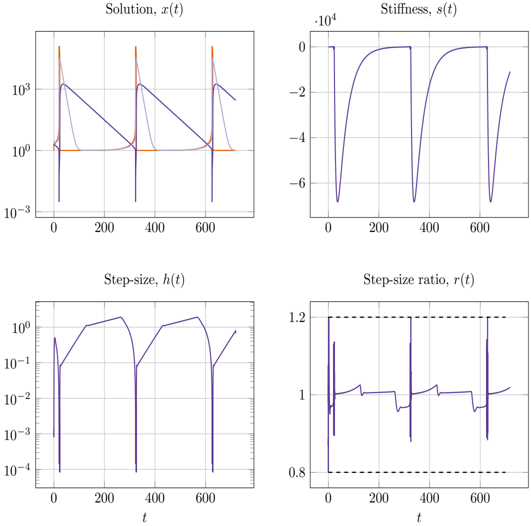
Method	**Order**	I_k^+ **method parameters** $\cot(\theta_j)$, $j = 1 : k-1$				
AMk	$k+1$	$\{0\}_{1:k-1}$				
dcBDFk	$k+1$	$\{(k+1)/(j+1)\}_{1:k-1}$				
Milne2	4	3				
Milne4	5	15/4	0	0		
IDC23	4	6/7	0			
IDC24	5	15/26	0	0		
IDC34	5	5/4	20/33	0		
IDC45	6	45/28	10/11	15/32	0	
IDC56	7	84/43	7/6	21/29	21/55	0

Method	**Order**	E_k **method parameters** $\cot(\theta_j)$, $j = 1 : k-1$				
ABk	k	$\{0\}_{1:k-1}$				
EDFk	k	$\{1/(j+1)\}_{1:k-1}$				
Explicit Euler	1					
Midpoint	2	∞				
Nyström3	3	-3/2	0			
Nyström4	4	-3/5	0	0		
Nyström5	5	-45/133	0	0	0	
EDC22	3	3/14	0			
EDC23	4	6/49	0	0		
EDC33	4	2/7	4/39	0		
EDC24	5	90/1121	0	0	0	
EDC34	5	10/53	10/219	0	0	
EDC45	6	45/193	10/121	15/692	0	0

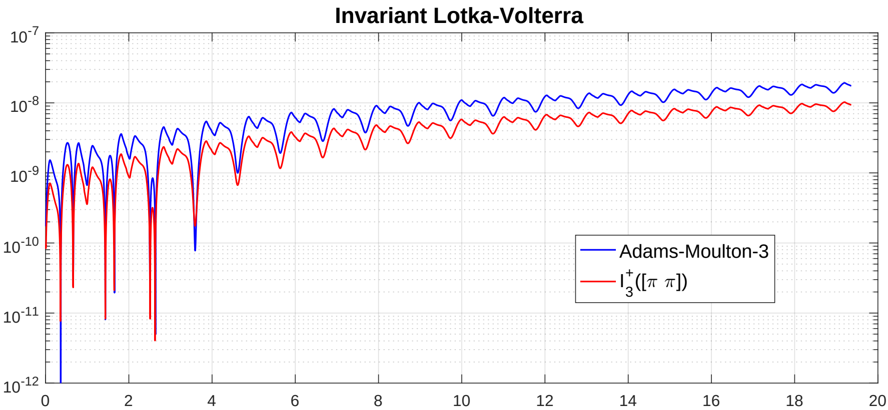
Experimental software platform, **MODES** offers:

- 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

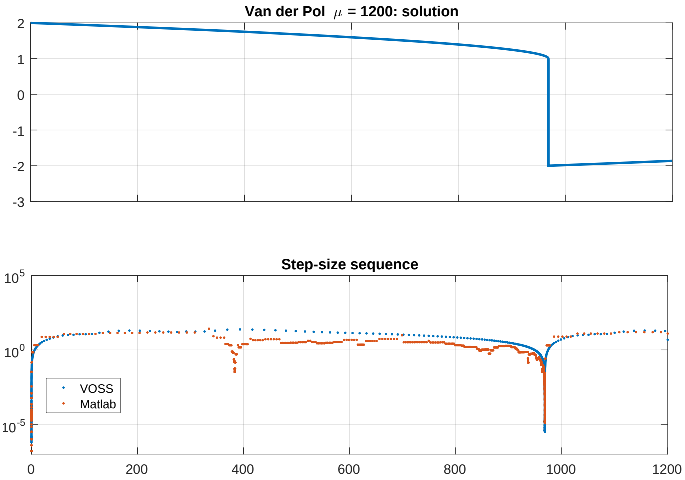
Results for the Oregonator problem solved with MODES-BDF5



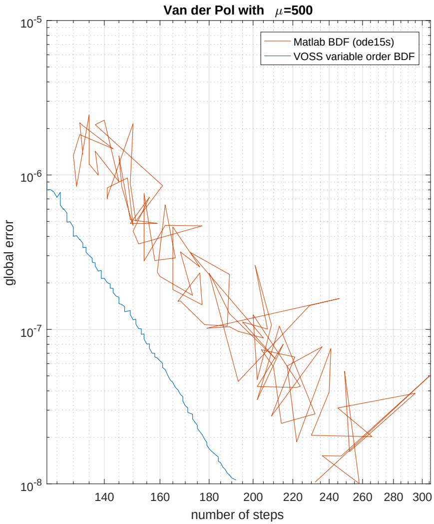
Compare different methods under exact same conditions: AM3 vs. another I_3^+ method. The second method is twice as accurate.



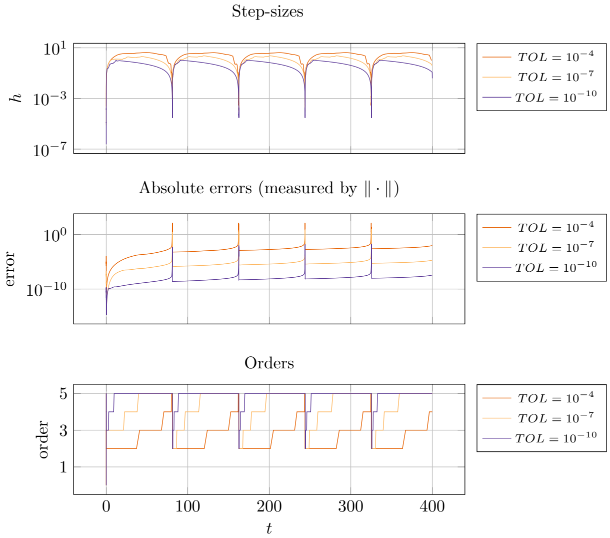
State-of-the-art error control: Matlab BDF1-5 vs. MODES BDF1-5 implementation



Accuracy/Work proportionality for MODES and Matlab's ode15s.

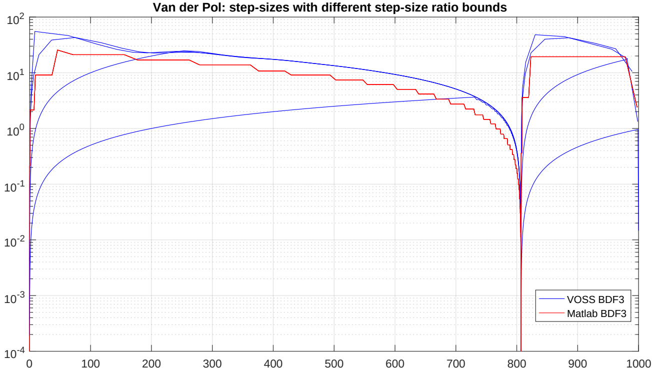


Van der Pol with variable order BDF



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

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

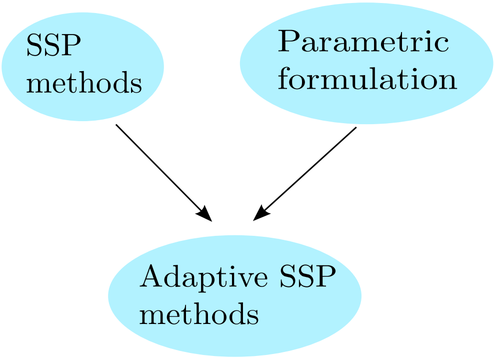
SSP if $\|y_n\| \leq \max\{\|y_{n-1}\|, \ldots, \|y_{n-k}\|\}$ for $0 < h \leq Ch^*$

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 $\tau_i = \beta_i/\alpha_i$

Example of optimal explicit 8-step SSP method of order 5

Its nonzero coefficients:

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

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

$$\begin{aligned} s_{n-1} &= 0 \\ s'_{n-1} &= 0 \\ s_{n-4} + h_{n-4}\tau_4 s'_{n-4} &= 0 \\ s_{n-5} + h_{n-5}\tau_5 s'_{n-5} &= 0 \\ s_{n-8} &= 0 \\ s'_{n-8} &= 0 \end{aligned}$$

$$\tau_4 = \beta_4/\alpha_4, \quad \tau_5 = \beta_5/\alpha_5$$

An application to differential-algebraic systems

DAEs are differential equations coupled with algebraic constraints

$$
\begin{aligned}
\dot{x} &= f(x, \lambda) \\
0 &= g(x, \lambda)
\end{aligned}
$$

λ is called the algebraic variable

An application to differential-algebraic systems

DAEs are differential equations coupled with algebraic constraints

$$\begin{aligned} \dot{x} &= f(x, \lambda) \\ 0 &= g(x, \lambda) \end{aligned}$$

λ is called the algebraic variable

The *index* characterizes the difficulty of a DAE.

Index 2 Euler-Lagrange DAE in multibody dynamics:

$$\begin{aligned} \dot{x} &= f(x) - G(x)^{\mathrm{T}}\lambda \\ 0 &= g(x) \end{aligned}$$

with $G(x) = \partial g/\partial x$, $G(x)G(x)^{\mathrm{T}}$ invertible

Generating polynomials of a multistep method: ρ, σ

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

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: ρ, σ

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

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: ρ, σ

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

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 σ not satisfying the strict root condition?

Instability in algebraic variables

Consequence of σ 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 σ 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: β-blocking

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

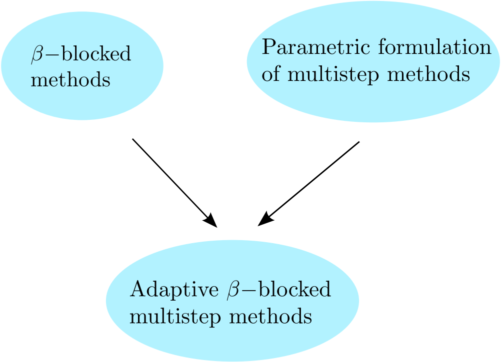
Construction of β-blocker operator

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

Order:

$k + 1$ for differential variable, k for algebraic variable

β-blocking published 1997–2000 as fixed step-size technique



Standard formulation of index-2 Euler-Lagrange DAE

$P_n \in \mathcal{P}_{k+1}$

$$
\begin{aligned}
&P_n'(t_n) = f(P_n(t_n)) - G(P_n(t_n))^{\mathrm{T}}\lambda_n \\
&P_n(t_{n-1}) = x_{n-1} \\
&P_n'(t_{n-1}) = f(x_{n-1}) - G(x_{n-1})^{\mathrm{T}}\lambda_{n-1} \\
&\cos\theta_{j-1}s_{n-j} + h_{n-j}\sin\theta_{j-1}(s_{n-j}' + G(x_{n-j})^{\mathrm{T}}\lambda_{n-j}) = 0 \\
&g(P_n(t_n)) = 0
\end{aligned}
$$

for $j = 2 : k$.

Then $x_n := P_n(t_n)$.

β-blocked formulation of index-2 Euler-Lagrange DAE

$P_n \in \mathcal{P}_{k+1}$ and $Q_n \in \mathcal{P}_k$

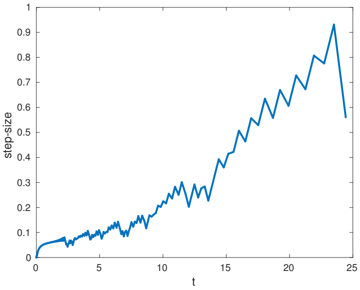
$$
\begin{aligned}
&P_n'(t_n) = f(P_n(t_n)) - G(P_n(t_n))^{\mathrm{T}}(Q_n(t_n) + \hat{c}\; h_{n-1}^k Q_n^{(k)}(t_n)) \\
&P_n(t_{n-1}) = x_{n-1} \\
&P_n'(t_{n-1}) = f(x_{n-1}) - G(x_{n-1})^{\mathrm{T}} Q_n(t_{n-1}) \\
&Q_n(t_{n-j}) = \lambda_{n-j} \\
&\cos\theta_{j-1} s_{n-j} + h_{n-j} \sin\theta_{j-1}(s_{n-j}') + G(x_{n-j})^{\mathrm{T}}\lambda_{n-j}) = 0 \\
&g(P_n(t_n)) = 0
\end{aligned}
$$

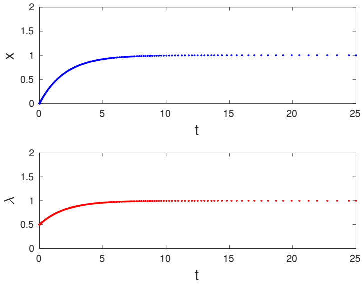
for $j = 2 : k, \quad \hat{c} = c\; \beta_k^{-1}.$

Set $x_n := P_n(t_n)$ and $\lambda_n := Q_n(t_n)$.

$\beta-$blocked AM2

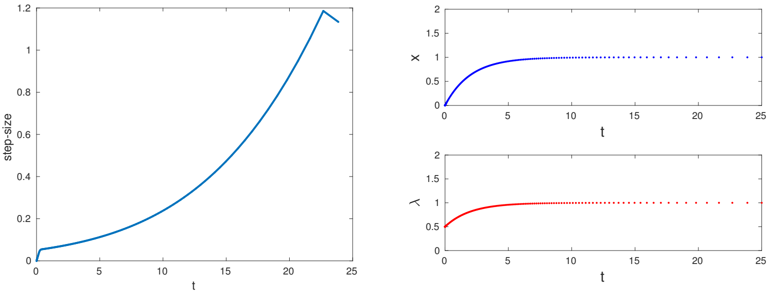
Effect of standard PI controller





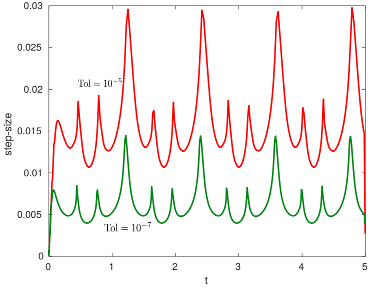
$\beta-$blocked AM2

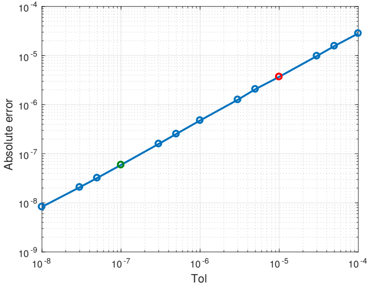
Effect of low-pass digital filter controller



4−step β−blocked Adams-Moulton for nonlinear pendulum

Low-pass filter controller: error-tolerance 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 β-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 β-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