Math 6630: Numerical Solutions of Partial Differential Equations
Solvers for initial value problems, Part IV

See Ascher and Petzold 1998, Chapters 1-5

Akil Narayan[1]

[1]Department of Mathematics, and Scientific Computing and Imaging (SCI) Institute
University of Utah

February 8, 2023

THE UNIVERSITY OF UTAH

www.sci.utah.edu

Initial value problems

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

$$\boldsymbol{u}_{n+1} \approx \boldsymbol{u}_n + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t, \boldsymbol{u}(t))\mathrm{d}t$$

We have previously discussed

- Simple schemes: forward/backward Euler, Crank-Nicolson
- Consistency and LTE
- 0-stability and scheme convergence
- absolute/A-stability and consequences
- multi-stage (Runge-Kutta) methods

Finally, we'll discuss multi-step schemes.

*
multi-stage

Preliminaries: polynomial interpolation

To begin we review some basic concepts about (univariate) polynomial interpolation:

Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function, and let $x_0, \ldots, x_n$ be any distinct points on $\mathbb{R}$.

Theorem

There is a unique polynomial $p(x)$ *of degree* n *such that* $f(x_j) = p(x_j)$ *for all* $j = 0, \ldots, n$.

Preliminaries: polynomial interpolation

To begin we review some basic concepts about (univariate) polynomial interpolation:

Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function, and let $x_0, \ldots, x_n$ be any distinct points on $\mathbb{R}$.

Theorem

There is a unique polynomial $p(x)$ of degree n such that $f(x_j) = p(x_j)$ for all $j = 0, \ldots, n$.

One way to construct this polynomial is via divided differences. Define

$$f[x_j] = f(x_j), \quad f[x_j, \ldots, x_{j+\ell}] = \frac{f[x_{j+1}, \ldots, x_{j+\ell}] - f[x_j, \ldots, x_{j+\ell-1}]}{x_{j+\ell} - x_j},$$

which are approximations to ℓth derivatives. Then,

$$p(x) = \sum_{\ell=0}^{n} f[x_0, \ldots, x_j] \prod_{j=0}^{\ell-1} (x - x_j).$$

$$\prod_{j=0}^{-1} a_j = 1$$

This is the Newton form of the interpolating polynomial.

Preliminaries: polynomial interpolation

To begin we review some basic concepts about (univariate) polynomial interpolation:

Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function, and let $x_0, \ldots, x_n$ be any distinct points on $\mathbb{R}$.

Theorem

There is a unique polynomial $p(x)$ of degree n such that $f(x_j) = p(x_j)$ for all $j = 0, \ldots, n$.

One way to construct this polynomial is via divided differences. Define

$$f[x_j] = f(x_j), \quad f[x_j, \ldots, x_{j+\ell}] = \frac{f[x_{j+1}, \ldots, x_{j+\ell}] - f[x_j, \ldots, x_{j+\ell-1}]}{x_{j+\ell} - x_j},$$

which are approximations to ℓth derivatives. Then,

$$p(x) = \sum_{\ell=0}^{n} f[x_0, \ldots, x_j] \prod_{j=0}^{\ell-1} (x - x_j).$$

This is the Newton form of the interpolating polynomial.

If $x_j = x_0 + jk$ for some $k > 0$, then expressions simplify considerably and more explicit formulas can be derived.

Preliminaries: difference equations

Simple theory for linear difference equations parallels linear differential equations:

$$u^{(s)}(t) + \sum_{j=1}^{s} \alpha_j u^{(s-j)}(t) = 0, \qquad u^{(j)}(0) = u_0^j, \qquad j = 0, \ldots, s-1.$$

Solve for a function $u(t)$, $t > 0$. The order is $s > 0$.

$$u_n + \sum_{j=1}^{s} \alpha_j u_{n-j} = 0, \qquad u_{n-j} = u_{n-j,0}, \qquad j = 1, \ldots, s.$$

Solve for a sequence u_ℓ, $\ell \geqslant 0$. The order is $s > 0$.

Preliminaries: difference equations

Simple theory for linear difference equations parallels linear differential equations:

$$u^{(s)}(t) + \sum_{j=1}^{s} \alpha_j u^{(s-j)}(t) = 0, \qquad u^{(j)}(0) = u_0^j, \qquad j = 0, \ldots, s-1.$$

Solve for a function $u(t)$, $t > 0$. The order is $s > 0$.

$$\text{Ansatz } u(t) = e^{zt} \implies p(z) := \sum_{j=0}^{s} \alpha_j z^{s-j} = 0, \quad (\alpha_0 = 1)$$

Solutions take the form $u(t) \sim e^{z_j t}$, where $z_1, \ldots, z_s$ are the roots of p.

$$u_n + \sum_{j=1}^{s} \alpha_j u_{n-j} = 0, \qquad u_{n-j} = u_{n-j,0}, \qquad j = 1, \ldots, s.$$

Solve for a sequence u_ℓ, $\ell \geqslant 0$. The order is $s > 0$.

$$\text{Ansatz } u_n = z^n \implies p(z) := \sum_{j=0}^{s} \alpha_j z^{s-j} = 0, \quad (\alpha_0 = 1)$$

Solutions take the form $u_n \sim z_j^n$, where $z_1, \ldots, z_s$ are the roots of p.

Preliminaries: difference equations

Simple theory for linear difference equations parallels linear differential equations:

$$u^{(s)}(t) + \sum_{j=1}^{s} \alpha_j u^{(s-j)}(t) = 0, \qquad u^{(j)}(0) = u_0^j, \qquad j = 0, \ldots, s-1.$$

Solve for a function $u(t)$, $t > 0$. The order is $s > 0$.

$$\text{Ansatz } u(t) = e^{zt} \implies p(z) := \sum_{j=0}^{s} \alpha_j z^{s-j} = 0, \quad (\alpha_0 = 1)$$

Solutions take the form $u(t) \sim e^{z_j t}$, where $z_1, \ldots, z_s$ are the roots of p.

Solutions $u(t)$ are stable if $\Re z_j \leqslant 0$. (Asymptotically stable if $\Re z_j < 0$.)

$$u_n + \sum_{j=1}^{s} \alpha_j u_{n-j} = 0, \qquad u_{n-j} = u_{n-j,0}, \qquad j = 1, \ldots, s.$$

Solve for a sequence u_ℓ, $\ell \geqslant 0$. The order is $s > 0$.

$$\text{Ansatz } u_n = z^n \implies p(z) := \sum_{j=0}^{s} \alpha_j z^{s-j} = 0, \quad (\alpha_0 = 1)$$

Solutions take the form $u_n \sim z_j^n$, where $z_1, \ldots, z_s$ are the roots of p.

Solutions u_n are stable if $|z_j| \leqslant 1$. (Asymptotically stable if $|z_j| < 1$.)

Multi-step methods, I

$f(z) = \frac{az+b}{cz+d}$, $ad - bc \neq 0$

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t;\boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad \alpha_j, \beta_j \in \mathbb{R}$$

Multi-step methods, I

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad \alpha_j, \beta_j \in \mathbb{R}$$

Comments:

- $s = 1$ corresponds to a general single-step (and single-stage) method
- $s > 1$: we need time history, e.g., $\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-3}, \ldots$
- We assume $\alpha_0 \neq 0$.
- We can rescale the equation by a constant without changing anything: we fix this freedom by setting $\alpha_0 = 1$.
- To avoid some minor pathologies, we typically assume that either $\alpha_j \neq 0$ or $\beta_j \neq 0$ for every j.
- $\beta_0 \neq 0$ corresponds to an implicit method. $\beta_0 = 0$ is an explicit method.

Multi-step methods, I

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad \alpha_j, \beta_j \in \mathbb{R}$$

Comments:

- $s = 1$ corresponds to a general single-step (and single-stage) method
- $s > 1$: we need time history, e.g., $\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-3}, \ldots$
- We assume $\alpha_0 \neq 0$.
- We can rescale the equation by a constant without changing anything: we fix this freedom by setting $\alpha_0 = 1$.
- To avoid some minor pathologies, we typically assume that either $\alpha_j \neq 0$ or $\beta_j \neq 0$ for every j.
- $\beta_0 \neq 0$ corresponds to an implicit method. $\beta_0 = 0$ is an explicit method.

Multi-step methods, I

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad \alpha_j, \beta_j \in \mathbb{R}$$

Comments:

- $s = 1$ corresponds to a general single-step (and single-stage) method
- $s > 1$: we need time history, e.g., $\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-3}, \ldots$
- We assume $\alpha_0 \neq 0$.
- We can rescale the equation by a constant without changing anything: we fix this freedom by setting $\alpha_0 = 1$.
- To avoid some minor pathologies, we typically assume that either $\alpha_j \neq 0$ or $\beta_j \neq 0$ for every j.
- $\beta_0 \neq 0$ corresponds to an implicit method. $\beta_0 = 0$ is an explicit method.

Multi-step methods, I

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad \alpha_j, \beta_j \in \mathbb{R}$$

Comments:

- $s = 1$ corresponds to a general single-step (and single-stage) method
- $s > 1$: we need time history, e.g., $\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-3}, \ldots$
- We assume $\alpha_0 \neq 0$.
- We can rescale the equation by a constant without changing anything: we fix this freedom by setting $\alpha_0 = 1$.
- To avoid some minor pathologies, we typically assume that either $\alpha_j \neq 0$ or $\beta_j \neq 0$ for every j.
- $\beta_0 \neq 0$ corresponds to an implicit method. $\beta_0 = 0$ is an explicit method.

Multi-step methods, I

For the IVP,

$$\boldsymbol{u}'(t) = \boldsymbol{f}(t; \boldsymbol{u}), \qquad\qquad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

$$\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$$

a general s-step multi-step scheme with timestep k has the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}), \qquad\qquad \alpha_j, \beta_j \in \mathbb{R}$$

$= \beta_0 f_{n+1} + \beta_1 f_n + \beta_2 f_{n-1} + \cdots$

Comments:

- $s = 1$ corresponds to a general single-step (and single-stage) method
- $s > 1$: we need time history, e.g., $\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-3}, \ldots$
- We assume $\alpha_0 \neq 0$.
- We can rescale the equation by a constant without changing anything: we fix this freedom by setting $\alpha_0 = 1$.
- To avoid some minor pathologies, we typically assume that either $\alpha_j \neq 0$ or $\beta_j \neq 0$ for every j.
- $\beta_0 \neq 0$ corresponds to an implicit method. $\beta_0 = 0$ is an explicit method.

Multi-step methods, II

To simplify notation, we will assume the ODE is autonomous ($\boldsymbol{f}(t, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{u})$), and will abbreviate $\boldsymbol{f}(\boldsymbol{u}_j)$ as $\boldsymbol{f}_j$. Then the multi-step method takes the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}$$

Multi-step methods, II

To simplify notation, we will assume the ODE is autonomous ($\boldsymbol{f}(t, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{u})$), and will abbreviate $\boldsymbol{f}(\boldsymbol{u}_j)$ as $\boldsymbol{f}_j$. Then the multi-step method takes the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}$$

Generally speaking, the constants are chosen so that:

- The α_j approximate $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(t_n)$
- The β_j approximate $\frac{1}{k}\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r$

Multi-step methods, II

To simplify notation, we will assume the ODE is autonomous ($\boldsymbol{f}(t, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{u})$), and will abbreviate $\boldsymbol{f}(\boldsymbol{u}_j)$ as $\boldsymbol{f}_j$. Then the multi-step method takes the form,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}$$

Generally speaking, the constants are chosen so that:

- The α_j approximate $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(t_n)$
- The β_j approximate $\frac{1}{k}\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r$

There are some miscellaneous issues we'll answer later, e.g.,

- If $s \geqslant 2$, how is $\boldsymbol{u}_1$ computed from $\boldsymbol{u}_0$?
- Must we fix the time-step k?

A warmup: single-step specializations

Specializing to single-step methods ($s = 1$) yields a transparent family of methods:

$$\boldsymbol{u}_{n+1} + \alpha_1 \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right).$$

(Recall $\alpha_0 = 1$)

A warmup: single-step specializations

Specializing to single-step methods ($s = 1$) yields a transparent family of methods:

$$\boldsymbol{u}_{n+1} + \alpha_1 \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right).$$

(Recall $\alpha_0 = 1$)

For any reasonable notion of consistency (to approximate $\boldsymbol{u}'(t_n)$), we should take $\alpha_1 = -1$.

A warmup: single-step specializations

Specializing to single-step methods ($s = 1$) yields a transparent family of methods:

$$\boldsymbol{u}_{n+1} + \alpha_1 \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right).$$

(Recall $\alpha_0 = 1$)

For any reasonable notion of consistency (to approximate $\boldsymbol{u}'(t_n)$), we should take $\alpha_1 = -1$.

With this restriction, then we have

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right),$$

and hence the right hand side should approximate $\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r)) \mathrm{d}r$, requiring $\beta_0 + \beta_1 = 1$ for consistency.

A warmup: single-step specializations

Specializing to single-step methods ($s = 1$) yields a transparent family of methods:

$$\boldsymbol{u}_{n+1} + \alpha_1 \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right).$$

(Recall $\alpha_0 = 1$)

For any reasonable notion of consistency (to approximate $\boldsymbol{u}'(t_n)$), we should take $\alpha_1 = -1$.

With this restriction, then we have

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \left(\beta_0 \boldsymbol{f}_{n+1} + \beta_1 \boldsymbol{f}_n \right),$$

and hence the right hand side should approximate $\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r)) \mathrm{d}r$, requiring $\beta_0 + \beta_1 = 1$ for consistency.

Then our general family of methods is

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + k \left(\beta \boldsymbol{f}_{n+1} + (1 - \beta) \boldsymbol{f}_n \right),$$

specializing to,

- $\beta = 0$: Forward Euler
- $\beta = 1$: Backward Euler
- $\beta = 1/2$: Crank-Nicolson

The Adams Family

There are two major classes of most popular multi-step methods. The first is the family of *Adams* methods.

For these methods we start with,

$$\boldsymbol{u}(t_{n+1}) = \boldsymbol{u}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r,$$

suggesting that we should take $\alpha_0 = 1$, $\alpha_1 = -1$.

The Adams Family

There are two major classes of most popular multi-step methods. The first is the family of *Adams* methods.

For these methods we start with,

$$\boldsymbol{u}(t_{n+1}) = \boldsymbol{u}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r,$$

suggesting that we should take $\alpha_0 = 1$, $\alpha_1 = -1$.

The β_j are chosen as a quadrature rule to approximate the integral:

$$\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r \approx k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}$$

Note that we are using points *outside* the interval of intergration (if $s > 1$).

The Adams Family

There are two major classes of most popular multi-step methods. The first is the family of *Adams* methods.

For these methods we start with,

$$\boldsymbol{u}(t_{n+1}) = \boldsymbol{u}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r,$$

suggesting that we should take $\alpha_0 = 1$, $\alpha_1 = -1$.

The β_j are chosen as a quadrature rule to approximate the integral:

$$\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{u}(r))\mathrm{d}r \approx k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}$$

Note that we are using points *outside* the interval of intergration (if $s > 1$).
Again, the particular type of scheme depends on whether we want an implicit or an explicit method:

- $\beta_0 = 0$ yields explicit methods (one fewer parameter to invest in LTE reduction)
- $\beta_0 \neq 0$ yields implicit methods

Adams-Bashforth Methods

The choice of explicit path yields the family of Adams-Bashforth methods.

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \sum_{j=1}^{s} \beta_j \boldsymbol{f}_{n+1-j}.$$

The β_j coefficients are used to ensure high-order LTE. E.g., two equivalent strategies:

- Expand in Taylor series, match terms by setting β_j
- Interpolate a degree-$(s-1)$ polynomial on data at $t_{n+1-s}, \ldots, t_n$, integrate the polynomial. The resulting coefficients multiplying the data are the β_j.

Adams-Bashforth Methods

The choice of explicit path yields the family of Adams-Bashforth methods.

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \sum_{j=1}^{s} \beta_j \boldsymbol{f}_{n+1-j}.$$

The β_j coefficients are used to ensure high-order LTE. E.g., two equivalent strategies:

- Expand in Taylor series, match terms by setting β_j
- Interpolate a degree-$(s-1)$ polynomial on data at $t_{n+1-s}, \ldots, t_n$, integrate the polynomial. The resulting coefficients multiplying the data are the β_j.

Coefficients for the Adams-Bashforth methods with order=steps:

	β_1	β_2	β_3	β_4	β_5	β_6
$p = s = 1$	1					
$p = s = 2$	$\frac{3}{2}$	-1				
$p = s = 3$	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$			
$p = s = 4$	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$		
$p = s = 5$	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$	
$p = s = 6$	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{9982}{1440}$	$-\frac{7298}{1440}$	$\frac{2877}{1440}$	$-\frac{475}{1440}$

Adams-Moulton Methods

The choice of implicit path yields the family of Adams-Moulton methods.

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}.$$

The β_j coefficients are used to ensure high-order LTE.
The same strategies as before are usable.

Note that technically we can take $s = 0$ here, which yields backward Euler.
(Though you'd still call this a 1-step method.)

Adams-Moulton Methods

The choice of implicit path yields the family of Adams-Moulton methods.

$$\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}_{n+1-j}.$$

The β_j coefficients are used to ensure high-order LTE.
The same strategies as before are usable.

Note that technically we can take $s = 0$ here, which yields backward Euler. (Though you'd still call this a 1-step method.) Coefficients for the Adams-Moulton methods with order=steps+1:

	β_0	β_1	β_2	β_3	β_4	β_5
$p-1=s=1$	$\frac{1}{2}$	$\frac{1}{2}$				
$p-1=s=2$	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$			
$p-1=s=3$	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		
$p-1=s=4$	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$	
$p-1=s=5$	$\frac{475}{1440}$	$\frac{1427}{1440}$	$-\frac{798}{1440}$	$\frac{482}{1440}$	$-\frac{173}{1440}$	$\frac{27}{1440}$

Backward Differentiation formulas

The Adams family of methods is not particularly robust for stiff problems.

As an alternative, consider the general form:

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

and now instead let us focus effort on setting $\beta_j = 0$ for $j > 0$, and choosing α_j to approximate $y'(t_n)$ to high order:

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k\beta_0 \boldsymbol{f}_{n+1}.$$

This is the family of (implicit) backward differentiation formulas (BDF) methods.

Backward Differentiation formulas

The Adams family of methods is not particularly robust for stiff problems.

As an alternative, consider the general form:

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

and now instead let us focus effort on setting $\beta_j = 0$ for $j > 0$, and choosing α_j to approximate $y'(t_n)$ to high order:

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k\beta_0 \boldsymbol{f}_{n+1}.$$

This is the family of (implicit) backward differentiation formulas (BDF) methods. Again, the BDF coefficients are explicitly computable:

	β_0	α_0	α_1	α_2	α_3	α_4	α_5	α_6
$p = s = 1$	1	1	-1					
$p = s = 2$	$\frac{2}{3}$	1	$-\frac{4}{3}$	$\frac{1}{3}$				
$p = s = 3$	$\frac{6}{11}$	1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$			
$p = s = 4$	$\frac{12}{25}$	1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$		
$p = s = 5$	$\frac{60}{137}$	1	$-\frac{300}{137}$	$\frac{300}{137}$	$-\frac{200}{137}$	$\frac{75}{137}$	$-\frac{12}{137}$	
$p = s = 6$	$\frac{60}{147}$	1	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$

Consistency and order of approximation

It's much easier to compute order conditions for multi-step methods (compared to multi-stage ones).

In particular, to compute the LTE for the scheme,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

we need to compute the residual for the expression

$$\frac{1}{k} \sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}(t_{n+1-j})).$$

Consistency and order of approximation

It's much easier to compute order conditions for multi-step methods (compared to multi-stage ones).

In particular, to compute the LTE for the scheme,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

we need to compute the residual for the expression

$$\frac{1}{k} \sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}(t_{n+1-j})).$$

Noting that $\boldsymbol{u}'(t) = \boldsymbol{f}(t, \boldsymbol{u}(t))$, the above expression is equivalent to,

$$\frac{1}{k} \sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{u}'(t_{n+1-j}),$$

and hence we can compute order conditions simply by computing Taylor expansions of $\boldsymbol{u}$ and $\boldsymbol{u}'$.

Consistency of multi-step methods, I

$$\frac{1}{k}\sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{u}'(t_{n+1-j}),$$

The $\mathcal{O}(1/k)$ terms from the above come from Taylor expansions of the α_j terms, implying that we require,

$$\sum_{j=0}^{s} \alpha_j = 0.$$

($s=1 \Rightarrow \alpha_0 = 1$, $1 + \alpha_1 = 0$)

Consistency of multi-step methods, I

$$\frac{1}{k}\sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{u}'(t_{n+1-j}),$$

The $\mathcal{O}(1/k)$ terms from the above come from Taylor expansions of the α_j terms, implying that we require,

$$\sum_{j=0}^{s} \alpha_j = 0.$$

For consistency (LTE vanishing as $k \downarrow 0$), we likewise require the $\mathcal{O}(1)$ terms to vanish, i.e.,

$$\sum_{j=0}^{s} (s-j)\alpha_j - \sum_{j=0}^{s} \beta_j = 0.$$

Consistency of multi-step methods, I

$$\frac{1}{k}\sum_{j=0}^{s}\alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s}\beta_j \boldsymbol{u}'(t_{n+1-j}),$$

The $\mathcal{O}(1/k)$ terms from the above come from Taylor expansions of the α_j terms, implying that we require,

$$\sum_{j=0}^{s}\alpha_j = 0.$$

For consistency (LTE vanishing as $k \downarrow 0$), we likewise require the $\mathcal{O}(1)$ terms to vanish, i.e.,

$$\sum_{j=0}^{s}(s-j)\alpha_j - \sum_{j=0}^{s}\beta_j = 0.$$

These two expressions are evaluations of certain *characteristic* polynomials:

$$\left.\begin{array}{l}\rho(w) = \sum_{j=0}^{s}\alpha_j w^{s-j} \\ \sigma(w) = \sum_{j=0}^{s}\beta_j w^{s-j}\end{array}\right\} \Longrightarrow \begin{array}{l}\rho(1) = 0 \\ \rho'(1) = \sigma(1)\end{array}$$

Consistency of multi-step methods, II

$$\text{LTE} = \frac{1}{k}\sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{u}'(t_{n+1-j}),$$

$$\rho(w) = \sum_{j=0}^{s} \alpha_j w^{s-j}$$

$$\sigma(w) = \sum_{j=0}^{s} \beta w^{s-j}$$

We have shown the following:

Theorem

A multi-step method is consistent if and only if $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$.

Consistency of multi-step methods, II

$$\text{LTE} = \frac{1}{k}\sum_{j=0}^{s} \alpha_j \boldsymbol{u}(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j \boldsymbol{u}'(t_{n+1-j}),$$

$$\rho(w) = \sum_{j=0}^{s} \alpha_j w^{s-j}$$

$$\sigma(w) = \sum_{j=0}^{s} \beta w^{s-j}$$

We have shown the following:

Theorem

A multi-step method is consistent if and only if $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$.

Of course, to attain more than first-order accuracy, we require more conditions.

0-stability: $k \downarrow 0 \Longrightarrow \sum_{j=0}^{s} \alpha_j u_{n+1-j} = 0$

Let's derive a multistep method.

$s=2$, explicit

$$u_{n+1} + \alpha_1 u_n + \alpha_2 u_{n-1} = \overset{k}{\vee}\beta_1 f_n + \overset{k}{\vee}\beta_2 f_{n-1}$$

$$u_{n+1} \approx u_{n-1} + 2k u'_{n-1} + \frac{4k^2}{2} u''_{n-1} + \frac{8k^3}{6} u'''_{n-1} + \dots$$

$$u_n \approx u_{n-1} + k u'_{n-1} + \frac{k^2}{2} u''_{n-1} + \frac{k^3}{6} u'''_{n-1} + \dots$$

$$f_n = u'_n \approx u'_{n-1} + k u''_{n-1} + \frac{k^2}{2} u'''_{n-1} + \dots$$

u_{n-1}: $1 + \alpha_1 + \alpha_2 = 0$ $\quad (\rho(1) = 0)$

u'_{n-1}: $2k + \alpha_1 k = \beta_1 k + \beta_2 k$ $\quad (\rho'(1) = \sigma(1))$

u''_{n-1}: $2k^2 + \alpha_1 k^2/2 = \beta_1 k \rightarrow 2 + \alpha_1/2 = \beta_1$

u'''_{n-1}: $\frac{4}{3}k^3 + \alpha_1 k^3/6 = \frac{k^3}{2}\beta_1 \rightarrow 4/3 + \alpha_1/6 = \beta_1/2$

$\Rightarrow -\frac{2}{3} + \frac{1}{6}\alpha_1 = 0$

$\alpha_1 = 4$

$\alpha_2 = -5$

$\beta_1 = 4$

$\beta_2 = 2$

$$u_{n+1} + 4u_n - 5u_{n-1} = 4f_n + 2f_{n-1}$$

2-step explicit method, LTE: k^3. $(p=3)$

0-Stability of multi-step methods

The characteristic polynomials are also integral in determining 0-stability:

Theorem

An s-step linear multi-step method is 0-stable if and only if the roots $w_1, \ldots, w_s$ of $\rho(w)$ all satisfy $|w_i| \leqslant 1$, and any roots satisfying $|w_i| = 1$ are simple.

This gives a fairly computable condition to identify 0-stability.

0-Stability of multi-step methods

The characteristic polynomials are also integral in determining 0-stability:

Theorem

An s-step linear multi-step method is 0-stable if and only if the roots $w_1, \ldots, w_s$ of $\rho(w)$ all satisfy $|w_i| \leqslant 1$, and any roots satisfying $|w_i| = 1$ are simple.

This gives a fairly computable condition to identify 0-stability.

Example: any one-step ($s = 1$) method is 0-stable, since $\rho(w) = w - 1$.

0-Stability of multi-step methods

The characteristic polynomials are also integral in determining 0-stability:

Theorem

An s-step linear multi-step method is 0-stable if and only if the roots $w_1, \ldots, w_s$ of $\rho(w)$ all satisfy $|w_i| \leqslant 1$, and any roots satisfying $|w_i| = 1$ are simple.

This gives a fairly computable condition to identify 0-stability.

Example: any one-step ($s = 1$) method is 0-stable, since $\rho(w) = w - 1$.

Example: All Adams- methods are 0-stable, since $\rho(w) = w^s - w^{s-1}$.

0-Stability of multi-step methods

The characteristic polynomials are also integral in determining 0-stability:

Theorem

An s-step linear multi-step method is 0-stable if and only if the roots $w_1, \ldots, w_s$ of $\rho(w)$ all satisfy $|w_i| \leqslant 1$, and any roots satisfying $|w_i| = 1$ are simple.

This gives a fairly computable condition to identify 0-stability.

Example: any one-step $(s = 1)$ method is 0-stable, since $\rho(w) = w - 1$.

Example: All Adams- methods are 0-stable, since $\rho(w) = w^s - w^{s-1}$.

Example: All BDF methods for $s \leqslant 6$ are 0-stable. Any BDF method with $s > 6$ is unstable.

0-Stability of multi-step methods

The characteristic polynomials are also integral in determining 0-stability:

Theorem

An s-step linear multi-step method is 0-stable if and only if the roots $w_1, \ldots, w_s$ of $\rho(w)$ all satisfy $|w_i| \leqslant 1$, and any roots satisfying $|w_i| = 1$ are simple.

This gives a fairly computable condition to identify 0-stability.

Example: any one-step ($s = 1$) method is 0-stable, since $\rho(w) = w - 1$.

Example: All Adams- methods are 0-stable, since $\rho(w) = w^s - w^{s-1}$.

Example: All BDF methods for $s \leqslant 6$ are 0-stable. Any BDF method with $s > 6$ is unstable.

There are reasonable-looking methods that violate 0-stability:

$$\boldsymbol{u}_{n+1} + 4\boldsymbol{u}_n - 5\boldsymbol{u}_{n-1} = k\left(4\boldsymbol{f}_n + 2\boldsymbol{f}_{n-1}\right),$$

and these methods are actually quite unstable.

$w = -5$ ☹

$(w+5)(w-1)$

$\rho(w) = w^2 + 4w - 5$

Absolute stability

We have a similar notion of absolute stability for multi-step methods: We require that the iteration,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

produces solutions $\boldsymbol{u}_n$ that do not grow exponentially in n for the test equation $u' = \lambda u$.

Absolute stability

We have a similar notion of absolute stability for multi-step methods: We require that the iteration,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

produces solutions $\boldsymbol{u}_n$ that do not grow exponentially in n for the test equation $u' = \lambda u$.

This results in the difference equation,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k\lambda \sum_{j=0}^{s} \beta_j \boldsymbol{u}_{n+1-j},$$

whose characteristic equation is,

$$\rho(w) = k\lambda\sigma(w) \stackrel{z=\lambda k}{=} z\sigma(w).$$

Absolute stability

We have a similar notion of absolute stability for multi-step methods: We require that the iteration,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

produces solutions $\boldsymbol{u}_n$ that do not grow exponentially in n for the test equation $u' = \lambda u$.

This results in the difference equation,

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k\lambda \sum_{j=0}^{s} \beta_j \boldsymbol{u}_{n+1-j},$$

whose characteristic equation is,

$$\rho(w) = k\lambda\sigma(w) \stackrel{z=\lambda k}{=} z\sigma(w).$$

Thus, we say that the region of (absolute) stability for the scheme is the set of z values such that $\rho(w) - z\sigma(w)$ has roots $w_1, \ldots, w_s$ all satisfying $|w_j| \leqslant 1$.

Forward Euler: $\alpha_0 u_{n+1} + \alpha_1 u_n = \beta_1 f_n$

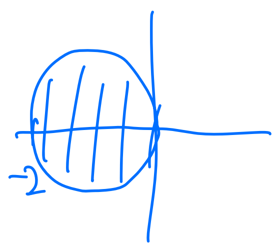
$\alpha_0 = \frac{1}{k} \quad \alpha_1 = -\frac{1}{k}, \quad \beta_1 = 1$

$\rho(w) = w - 1 \quad \sigma(w) = 1$

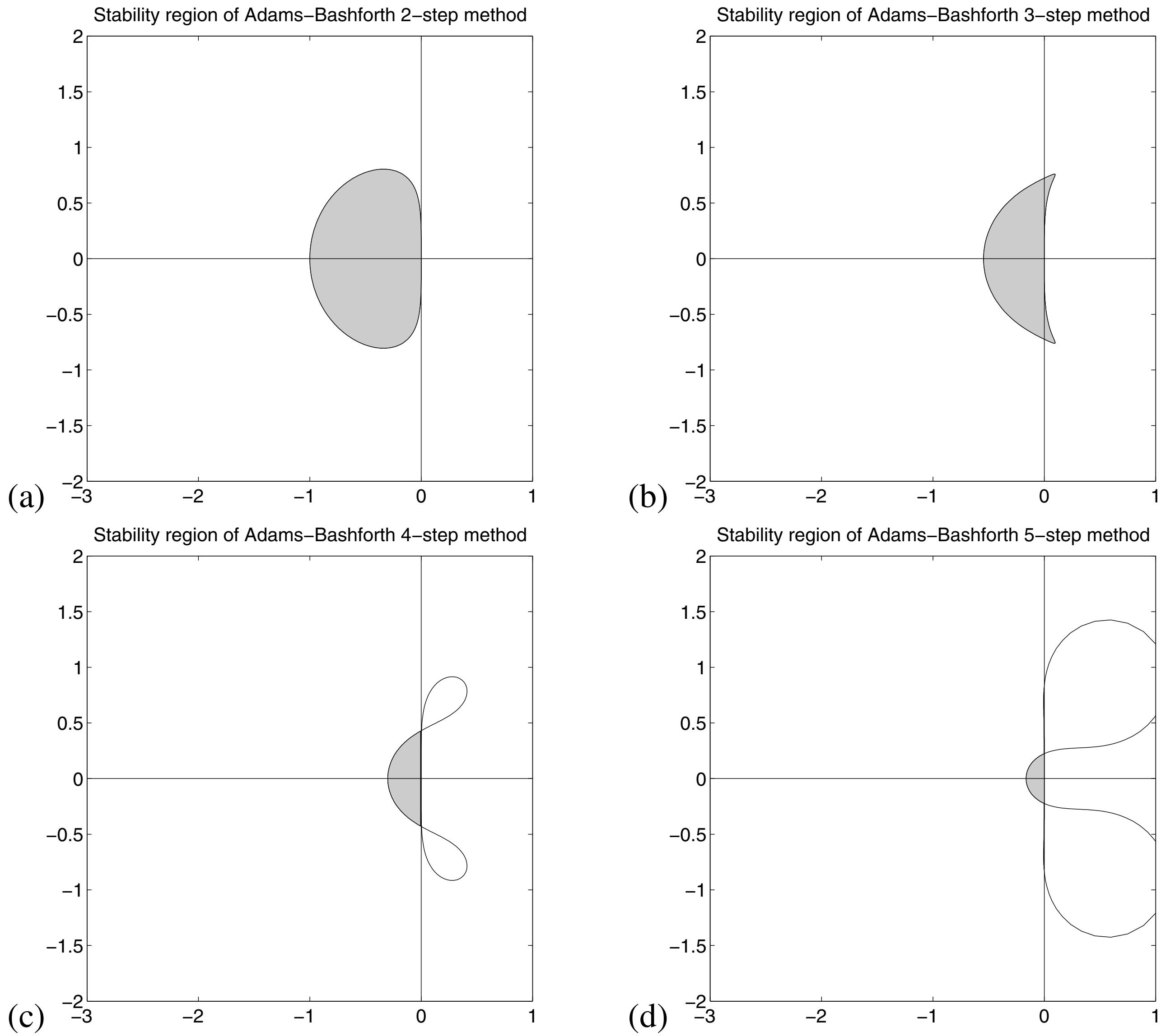
roots of $\rho(w) - z\sigma(w)$

$w - 1 - z \Rightarrow w = z + 1$

when is $|z+1| \le 1$?

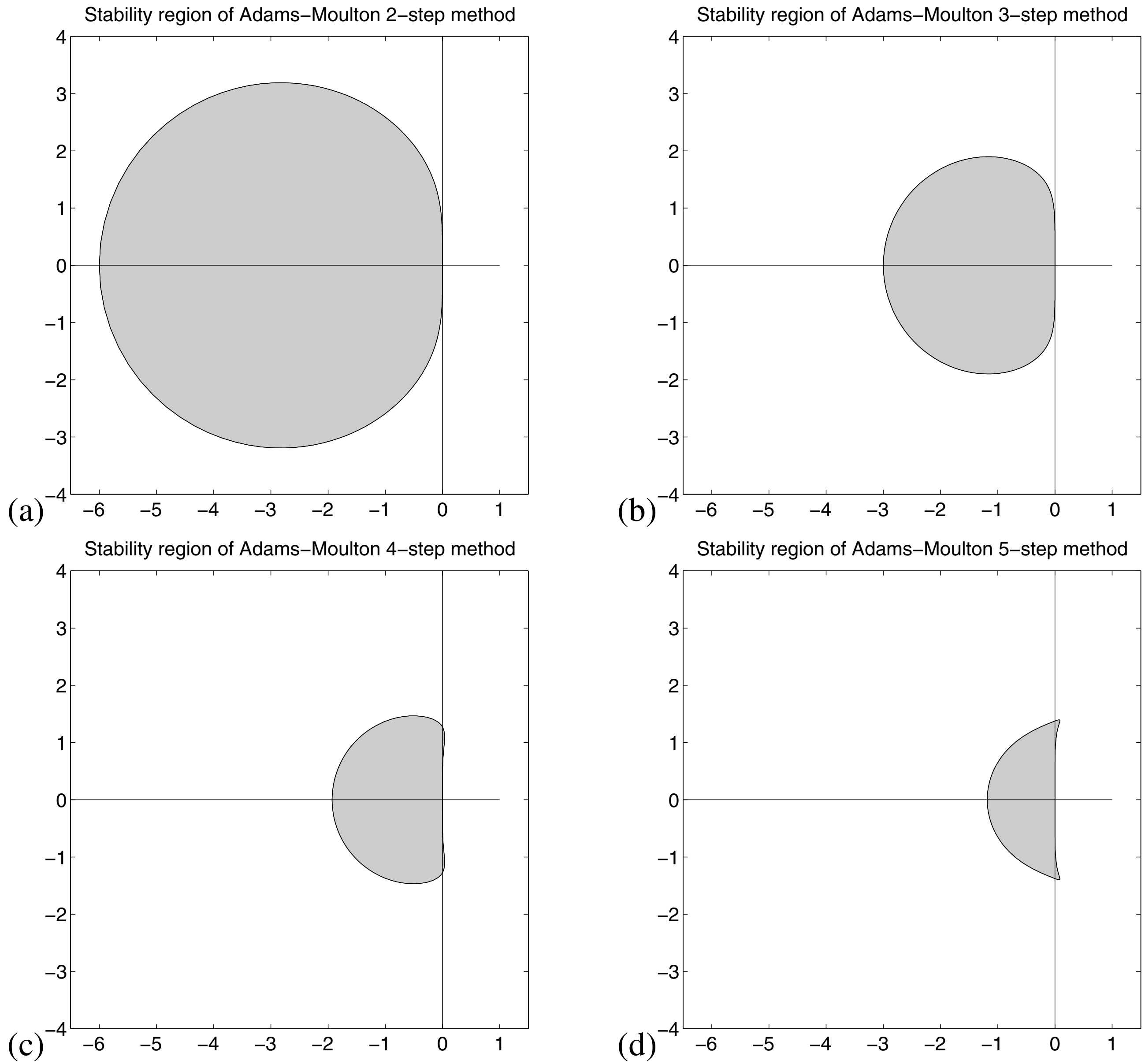


Absolute stability: Adams-Bashforth



LeVeque 2007, Figure 7.2

Absolute stability: Adams-Moulton



LeVeque 2007, Figure 7.3

Odds and ends for multi-step methods

Startup

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

How to start from $n = 0$ if $s > 1$?

Usually accomplished with Runge-Kutta methods of similar order.

Odds and ends for multi-step methods

Startup

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

How to start from $n = 0$ if $s > 1$?

Usually accomplished with Runge-Kutta methods of similar order.

Odds and ends for multi-step methods

Startup

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

How to start from $n = 0$ if $s > 1$?

Usually accomplished with Runge-Kutta methods of similar order.

Predictor-corrector methods

Explicit and implicit methods are frequently used in *predictor-corrector* frameworks, e.g.,:

- An explicit approximation to $\boldsymbol{u}_{n+1}$ is computed with an Adams-Bashforth method.
- This approximation is used as an emulator for the unknown $\boldsymbol{u}(t_{n+1})$ on the right-hand side of an Adams-Moulton method.

Odds and ends for multi-step methods

Startup

$$\sum_{j=0}^{s} \alpha_j \boldsymbol{u}_{n+1-j} = k \sum_{j=0}^{s} \beta_j \boldsymbol{f}(t_{n+1-j}, \boldsymbol{u}_{n+1-j}),$$

How to start from $n = 0$ if $s > 1$?

Usually accomplished with Runge-Kutta methods of similar order.

Predictor-corrector methods

Explicit and implicit methods are frequently used in *predictor-corrector* frameworks, e.g.,:

- An explicit approximation to $\boldsymbol{u}_{n+1}$ is computed with an Adams-Bashforth method.
- This approximation is used as an emulator for the unknown $\boldsymbol{u}(t_{n+1})$ on the right-hand side of an Adams-Moulton method.

Predictor-corrector methods are an example from a more general class of methods called *general linear methods*, which encompass both multi-stage and multi-step methods.

References I

Ascher, Uri M. and Linda R. Petzold (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM. ISBN: 978-1-61197-139-2.

Butcher, J. C. (2006). "General Linear Methods". In: *Acta Numerica* 15, pp. 157–256. ISSN: 1474-0508, 0962-4929. DOI: 10.1017/S0962492906220014.

LeVeque, Randall J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM. ISBN: 978-0-89871-783-9.