CHAPTER 5: *Linear Multistep Methods*

Multistep: use information from many steps

$\rightarrow$ Higher order possible with fewer function evaluations than with RK.

$\rightarrow$ Convenient error estimates.

$\rightarrow$ Changing stepsize or order is more difficult!

Recall $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \qquad t \geq t_0.$

Denote

$$\begin{aligned} \mathbf{f}_l &= \mathbf{f}(t_l, \mathbf{y}_l), \\ \mathbf{y}_l &\approx \mathbf{y}(t_l). \end{aligned}$$

A k-step linear multistep method is

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = \Delta t \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}.$$

α_j: the method's coefficients

β_j: the method's coefficients

To avoid degeneracy, assume

$$\alpha_0 \neq 0, \qquad |\alpha_k| + |\beta_k| \neq 0.$$

To eliminate scaling: $\alpha_0 = 1$.

LMM is explicit if $\beta_0 = 0$,
implicit otherwise.

Note 1. *Past k integration steps are equally spaced.*

Assume $\mathbf{f}$ has many bounded derivatives as needed.

Note 2. *LMM is called linear*
because it is linear in $\mathbf{f}$,
unlike RK methods. (review!)

Note 3. *$\mathbf{f}$ itself is not linear!*

The most popular LMMs are based on polynomial interpolation (even those that are not still use interpolation to change step-size).

Review : Polynomial interpolation and divided differences

We interpolate $f(x)$ at k distinct points $t_1, t_2, \cdots, t_k$ by the unique polynomial $\phi(t)$ of degree $< k$.
$\phi(t) = \phi_0 + \phi_1 t + \phi_2 t^2 + \cdots + \phi_{k-1} t^{k-1}$, k unknowns;

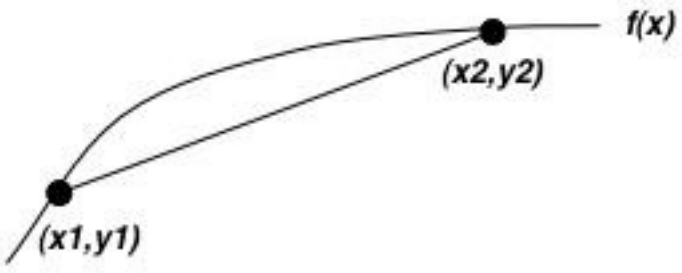
i.e., $\phi(t_l) = f(t_l)$, $l = 1, 2, \cdots, k$ k equations.

$\rightarrow$ Can set up and solve linear system for ϕ_i.

$\rightarrow$ Can write down Lagrange interpolating polynomial.

$\rightarrow$ Can use Newton form (divided differences).

Note 4. *These are simply different descriptions of the same polynomial!*



$$y = (\tfrac{y_2-y_1}{x_1-x_2})x + (\tfrac{y_1x_2-y_2x_1}{x_2-x_1})$$
$$y = y_1(\tfrac{x-x_2}{x_1-x_2}) + y_2(\tfrac{x-x_1}{x_2-x_1})$$
$$y = y_1 + (\tfrac{y_2-y_1}{x_2-x_1})(x-x_1)$$

Newton's form

$$\begin{aligned}\phi(t) &= f[t_1] + f[t_1,t_2](t-t_1) + \cdots \\ &\quad + f[t_1,t_2,\cdots,t_k](t-t_1)(t-t_2)\cdots(t-t_{k-1}),\end{aligned}$$

where the divided differences are defined by

$$\begin{aligned} f[t_l] &= f(t_l), \\ f[t_l,t_m] &= \tfrac{f[t_l]-f[t_m]}{t_l-t_m}, \\ &\vdots \\ f[t_l,\cdots,t_{l+i}] &= \tfrac{f[t_{l+1},\cdots,t_{l+i}]-f[t_l,\cdots,t_{l+i-1}]}{t_{l+i}-t_l}. \end{aligned}$$

Interpolation error at any point t:

$$\begin{aligned} f(t)-\phi(t) &= f[t_1,\cdots,t_k,t]\prod_{i=1}^{k}(t-t_i) \\ &= O((\Delta t)^k) \text{ if all } t_i \text{ are within } \mathcal{O}(\Delta t). \end{aligned}$$

Note 5. $f[t_1,t_2,\cdots,t_k,t] \approx \frac{f^{(k)}(t)}{k!}$ *if Δt is small.*

5.1.1 Adams methods

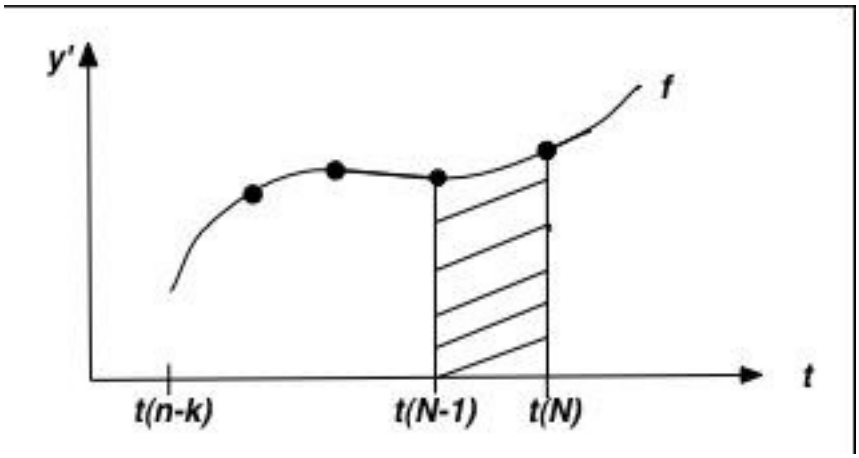
- Most popular non-stiff solvers

$$\begin{aligned} \dot{y} &= f(t,y) \\ \Rightarrow \qquad y(t_n) &= y(t_{n-1}) + \int_{t_{n-1}}^{t_n} f(t,y(t))dt. \end{aligned}$$

Approximate $f(t,y(t))$ with interpolating polynomial.

Comparing with $\sum_{j=0}^{k} \alpha_j y_{n-j} = \Delta t \sum_{j=0}^{k} \beta_j f_{n-j}$, we see $\alpha_0 = 1,\ \alpha_1 = 1,\ \alpha_j = 0,\ j = 2,3,\cdots,k.$

For k-step explicit Adams (Adams–Bashforth) interpolate f through k previous points $t_{n-1}, t_{n-2}, \cdots, t_{n-k}$.



It can be shown that

$$y_n = y_{n-1} + \Delta t \sum_{j=1}^{k} \beta_j f_{n-j},$$

where

$$\beta_j = (-1)^{j-1} \sum_{i=j-1}^{k-1} \binom{i}{j-1} \gamma_i,$$

$$\gamma_i = (-1)^i \int_0^1 \binom{-s}{i} ds.$$

Note 6. $\binom{s}{i} = \dfrac{\overbrace{s(s-1)(s-2)\cdots(s-i+1)}^{i \text{ factors}}}{i!},$

$$\binom{s}{0} = 1.$$

k-step methods $\leftrightarrow$ use information at k points

$$t_{n-1}, t_{n-2}, \cdots, t_{n-k}.$$

Also called a $(k+1)$-value method:

$k+1 \leftrightarrow$ total memory requirement.

- Local truncation error
  $C_{p+1}(\Delta t)^p y^{(p+1)}(t_n) + \mathcal{O}((\Delta t)^{p+1})$ with $p = k$.

Note 7. *Only 1 new function evaluation per step.*

Example 1. *First-order AB:*

$$y_n = y_{n-1} + \Delta t \beta_1 f_{n-1},$$

where $\beta_1 = (-1)^0 \sum_{i=0}^{1} \binom{i}{0} \gamma_0 = \gamma_0,$ *(verify)*

and $\gamma_0 = (-1)^0 \int_0^1 \binom{-s}{0} ds = 1.$ *(verify)*

$$y_n = y_{n-1} + \Delta t f_{n-1}.$$

Forward Euler!

$$k = 2: \qquad y_n = y_{n-1} + \Delta t \beta_1 f_{n-1} + \Delta t \beta_2 f_{n-2}.$$

$$\begin{array}{lllllll} \beta_1 & = & (-1)^0 \sum_{i=0}^{1} \binom{i}{0} \gamma_i & = & \gamma_0 + \gamma_1 & = & \frac{3}{2} \\ \beta_2 & = & (-1)^1 \sum_{i=1}^{0} \binom{i}{1} \gamma_i & = & -\gamma_1 & = & -\frac{1}{2} \\ \gamma_0 & = & (-1)^0 \int_0^1 \binom{-s}{0} ds & = & 1 & & \\ \gamma_1 & = & (-1)^1 \int_0^1 \binom{-s}{1} ds & = & \int_0^1 s ds & = & \frac{1}{2} \end{array}$$

AB2: $y_n = y_{n-1} + \Delta t \left[\frac{3}{2} f_{n-1} - \frac{1}{2} f_{n-2} \right].$

Verify $\beta_1, \beta_2, \gamma_0, \gamma_1$, and AB2.

- In general, AB methods have small regions of absolute stability!

$\rightarrow$ We look for implicit methods:

Implicit Adams methods

$\updownarrow$

Adams–Moulton.

Derivation is similar to AB,
but interpolating polynomial has degree $\leq k$.

$\updownarrow$

Use information at t_n also.

$$y_n = y_{n-1} + \Delta t \sum_{j=0}^{k} \beta_j f_{n-j}.$$

AM methods have order $p = k + 1$.

There are 2 AM1 methods

— one of order 1 and one of order 2.

$y_n = y_{n-1} + \Delta t f_n$ backward Euler

$y_n = y_{n-1} + \frac{\Delta t}{2}[f_n + f_{n-1}]$ Trapezoidal method

AM2: $y_n = y_{n-1} + \frac{\Delta t}{12}[5f_n + 8f_{n-1} - f_{n-2}]$.

Note 8. • *AM have smaller error constants than AB (of the same order).*

- *They have one less data point for a given order.*
- *AM have much larger stability regions than AB.*
- *AB-AM often used as predictor-corrector pair.*
- *AM derived by straightforward polynomial interpolation.*

5.1.2 BDF (Backward Differentiation Formulas)

- Most popular multistep method for stiff problems.
- Defining feature: only $\beta_0 \neq 0$;
  i.e., only use f_n.
- Motivation: obtain a formula with stiff decay.

Apply LMM to $\dot{y} = \lambda(y - g(t)),\ \Delta t \mathcal{R}e(\lambda) \to -\infty$

$$\Rightarrow \qquad \sum_{j=0}^{k} \beta_j (y_{n-j} - g(t_{n-j})) \to 0.$$

So to have $y_n - g(t_n) \to 0$ for arbitrary $g(t)$,
we must have $\beta_0 \neq 0$ and $\beta_1 = \beta_2 = \cdots = \beta_k = 0$.

Recall, Adams methods fit a polynomial to past values of f and integrate it.

In contrast, BDF methods fit a polynomial to past values of y and set the derivative of the polynomial at t_n equal to f_n:

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = \Delta t \beta_0 f(t_n, y_n).$$

Note 9. • *BDF methods are implicit*
$\rightarrow$ *Usually implemented with modified Newton (more later).*

- *Only the first 6 BDF methods are stable!*
  (order 6 is the bound on BDF order)

- *BDF1 is backward Euler.*

5.1.3 Initial values

With one-step methods, $y_0 = y(t_0)$ is all we need!

With a k-step LMM, we need $k-1$ additional starting values.
$\rightarrow$ These values must be $\mathcal{O}((\Delta t)^p)$ accurate!
($\leftrightarrow$ or within error tolerance).

Possible solutions:

- Start with a RK method of order p.

- (More elegant?)
  Recursively use a j-step method,
  building $j = 1, 2, \cdots, k-1$.
  $\rightarrow$ Gradually build up to k-step method.

  - Nice conceptually because you can use only methods from one family (this is what codes do in practice).
  - But order is less with lower k!
    $\rightarrow$ Need error control
    ($\rightarrow$ and order control!)

Example 2.

$$\dot{y} = -5ty^2 + \frac{5}{t} - \frac{1}{t^2}, \quad y(1) = 1.$$

step h	$k=1$ error	rate	$k=2$ error	rate	$k=4$ error	rate
0.2	.40e-2		*		*	
0.1	.65e-6	12.59	.32e-2		*	
0.05	.32e-6	1.00	.16e-8	20.9	.16e-1	
0.02	.13e-6	1.00	.26e-9	2.00	.35e-14	31.8
0.01	.65e-7	1.00	.65e-10	2.00	.16e-14	1.17
0.005	.32e-7	1.00	.16e-10	2.00	.11e-14	0.54
0.002	.13e-7	1.00	.26e-11	2.00	.47e-14	-1.61

Table 5.4: Example 5.3: Errors and calculated convergence rates for Adams-Bashforth methods.

step h	$k=1,\ p=1$ error	rate	$k=1,\ p=2$ error	rate	$k=3\ p=4$ error	rate
0.2	.13e-5		.52e-8		.22e-11	
0.1	.65e-6	1.01	.13e-8	2.00	.14e-12	4.03
0.05	.32e-6	1.00	.33e-9	2.00	.87e-14	3.96
0.02	.13e-6	1.00	.52e-10	2.00	.50e-15	3.12
0.01	.65e-7	1.00	.13e-10	2.00	.17e-14	-1.82
0.005	.32e-7	1.00	.33e-11	2.00	.11e-14	0.73
0.002	.13e-7	1.00	.52e-12	2.00	.47e-14	-1.62

Table 5.5: Example 5.3: Errors and calculated convergence rates for Adams-Moulton methods.

5.2 Order, 0-stability, Convergence

It is still true that

Consistency + 0-stability = Convergence

(order at least 1)

For RK, 0-stability was easy,

high order was difficult!

For LMM, high order is straightforward:

just use enough past values.

But now 0-stability is not trivial.

5.2.1 Order

$\rightarrow$ Simple but general derivation.

$\rightarrow$ Not only order, but also leading coefficient of truncation error.

$\rightarrow$ Very useful in designing LMMs!

Define $\mathcal{L}_h y(t) = \sum_{j=0}^{k} \alpha_j y(t-j\Delta t) - \Delta t \beta_j \dot{y}(t-j\Delta t)$.

$\rightarrow$ Truncation error

$$d_n = \frac{1}{\Delta t} \mathcal{L}_h y(t_n).$$

Because exact solution satisfies $\dot{y}(t) = f(t, y(t))$,

$$\mathcal{L}_h y(t) = \sum_{j=0}^{k} \alpha_j \underbrace{y(t-j\Delta t)}_{\text{expand}} - \Delta t \beta_j \underbrace{f(t-j\Delta t, y(t-j\Delta t))}_{\text{about } t}$$

$$\mathcal{L}_h y(t) = C_0 y(t) + C_1 \Delta t \dot{y}(t) + \cdots + C_q (\Delta t)^q y^{(q)}(t) + \cdots,$$

where

$$C_0 = \sum_{j=0}^{k} \alpha_j$$

$$C_i = (-1)^i \left[\frac{1}{i} \sum_{j=1}^{k} j^i \alpha_j + \frac{1}{(i-1)!} \sum_{j=0}^{k} j^{i-1} \beta_j \right]$$

$$i = 1, 2, \cdots .$$

For a method of order p,

$$C_0 = C_1 = \cdots = C_p = 0, \quad C_{p+1} \neq 0.$$

C_{p+1} is called the error constant.

The first few equations:

C_0: $0 = \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_k$

C_1: $0 = (\alpha_1 + 2\alpha_2 + \cdots + k\alpha_k) + (\beta_0 + \beta_1 + \cdots + \beta_k)$

C_2: $0 = \frac{1}{2!}(\alpha_1 + 2^2\alpha_2 + \cdots + k^2\alpha_k)$
$+(\beta_1 + 2\beta_2 + \cdots + k\beta_k)$

$\vdots$

Example 3. • *Forward Euler:* $\alpha_1 = -1, \beta_1 = 1$
$(\alpha_0 = 1, \beta_0 = 0)$

$C_0 : \alpha_0 + \alpha_1 = 1 - 1 = 0$
$C_1 : -(\alpha_1 + \beta_0 + \beta_1) = -1 + 0 + 1 = 0$
$C_2 : \frac{1}{2}\alpha_1 + \beta_1 = -\frac{1}{2} + 1 = \frac{1}{2} \neq 0 \Rightarrow$ *first order.*

- *AB2:* $y_n = y_{n-1} + \Delta t\left(\frac{3}{2}f_{n-1} - \frac{1}{2}f_{n-2}\right)$

$\Rightarrow \alpha_0 = 1, \alpha_1 = -1, \beta_0 = 0, \beta_1 = \frac{3}{2}, \beta_2 = -\frac{1}{2}$
$(\alpha_2 = 0)$

$C_0 : \alpha_0 + \alpha_1 + \alpha_2 = 1 - 1 + 0 = 0$

$C_1 : -(\alpha_1 + 2\alpha_1 + \beta_0 + \beta_1 + \beta_2)$
$= -(-1 + 0 + 0 + \frac{3}{2} - \frac{1}{2}) = 0$

$C_2 : \frac{1}{2!}[\alpha_1 + 2^2\alpha_2] + [\beta_1 + 2\beta_2] = \frac{1}{2}(-1) + \frac{3}{2} - 1 = 0$

$C_3 : -\left(\frac{1}{3!}[\alpha_1 + 2^3\alpha_2] + \frac{1}{2!}[\beta_1 + 2^2\beta_2]\right)$
$= -\left(-\frac{1}{6} + \frac{1}{2!}\left(\frac{3}{2} - 2\right)\right) = \frac{5}{12} \neq 0$

$\Rightarrow$ *second order*

Example 4. *You can derive/design LMMs similarly; e.g., derive the 2-step BDF:*

$$y_n + \alpha_1 y_{n-1} + \alpha_2 y_{n-2} = \Delta t \beta_0 f_n.$$

$$\left.\begin{array}{rrcl} C_0: & 1 + \alpha_1 + \alpha_2 & = & 0 \\ C_1: & \alpha_1 + 2\alpha_2 + \beta_0 & = & 0 \\ C_2: & \frac{1}{2!}(\alpha_1 + 4\alpha_2) & = & 0 \end{array}\right\} \Rightarrow \begin{array}{rcl} \alpha_1 & = & -\frac{4}{3} \\ \alpha_2 & = & \frac{1}{3} \\ \beta_0 & = & \frac{2}{3} \end{array}$$

(verify!)

Also $C_3 = -\frac{2}{9}$. *(verify!)*

→ You can design "optimal" methods,
e.g., highest order for given k; smallest C_{p+1}.
But beware of stability! Do not take it for granted!

Recall: Consistency $\leftrightarrow$ Order $p \geq 1$.

For a LMM, this means $\sum_{j=0}^{k} \alpha_j = 0$,
$\sum_{j=1}^{k} j\alpha_j + \sum_{j=0}^{k} \beta_j = 0$.

Define characteristic polynomials

$$\begin{aligned}\rho(\xi) &= \sum_{j=0}^{k} \alpha_j \xi^{k-j}, \\ \sigma(\xi) &= \sum_{j=0}^{k} \beta_j \xi^{k-j}.\end{aligned}$$

In these terms, a LMM is consistent iff

$$\rho(1) = 0 \quad \text{and} \quad \rho'(1) = \sigma(1).$$

5.2.2 Root Condition

We now derive (simple) conditions on the roots of $\rho(\xi)$ to guarantee 0-stability.

(Then adding consistency, we get convergence.)

We already know that for consistency $\rho(1) = 0$.

$\therefore$ $\xi = 1$ must be a root of $\rho(\xi)$.

(This is called the principal root.)

Remaining analysis shows 0-stability requires all other roots of $\rho(\xi)$ have magnitude strictly less than 1 or if their magnitude is 1, then their multiplicity cannot exceed 1 (they cannot be repeated).

The combination of these two requirements is known as the root condition.

Formal statement to follow shortly.

5.2.3 0-Stability and Convergence

Recall: 0-stability has a complicated definition.

Essentially, we are looking at the stability of the difference equation as $\Delta t \to 0$.

It turns out to be sufficient to consider $\dot{y} = 0$.

Then we obtain

$$\alpha_0 y_n + \alpha_1 y_{n-1} + \cdots + \alpha_k y_{n-k} = 0.$$

This must be stable for the LMM to be stable!
This is a linear, constant-coefficient difference equation.
We guess a solution of the form

$$y_n = \xi^n.$$

Then we obtain

$$\begin{aligned} \alpha_0\xi^n + \alpha_1\xi^{n-1} + \cdots + \alpha_k\xi^{n-k} &= 0, \\ \text{or} \quad \xi^{n-k}\underbrace{[\alpha_0\xi^k + \alpha_1\xi^{k-1} + \cdots + \alpha_k]}_{\rho(\xi)} &= 0. \end{aligned}$$

For $y_n = \xi^n$ to be stable,

$$\begin{aligned} |y_n| &\leq |y_{n-1}| \\ |\xi^n| &\leq |\xi^{n-1}| \\ \Rightarrow \quad |\xi| &\leq 1 \end{aligned}$$

Theorem 1. *The LMM is 0-stable iff all roots ξ_i of $\rho(\xi)$ satisfy*

$$|\xi_i| \leq 1,$$

and if $|\xi_i| = 1$, then ξ_i is a simple root.

This is called the root condition.

If

- *the root condition is satisfied,*
- *the method has order p,*
- *the starting values are order p,*

then the method is convergent with order p.

Exercise: Use this to show one-step methods are (trivially) 0-stable.

Example 5. *Consider*

$$y_n = -4y_{n-1} + 5y_{n-2} + 4\Delta t f_{n-1} + 2\Delta t f_{n-2}.$$

This turns out to be the most accurate explicit, 2-step method (in terms of LTE - Local Truncation Error).

However,

$$\rho(\xi) = \xi^2 + 4\xi - 5 = \underbrace{(\xi - 1)}_{\text{always if method is consistent}} (\xi + 5).$$

$\rightarrow$ *We see* $\quad \xi_2 = -5.$
$\rightarrow |\xi_2| > 1 \quad \Rightarrow \quad$ *method is unstable!*

Consider solving $\quad \dot{y} = 0$ with $\quad y_0 = 0, \quad y_1 = \epsilon.$

Then

$$\begin{array}{lcccc} y_2 & = & -4y_1 & = & -4\epsilon \\ y_3 & = & -4y_2 + 5y_1 & = & 21\epsilon \\ y_4 & = & -4y_3 + 5y_2 & = & -104\epsilon \\ & \vdots & & & \end{array}$$

It blows up!

Consider now the test equation $\quad \dot{y} = \lambda y.$
Applying a LMM:

$$\sum_{j=0}^{k} (\alpha_j - \Delta t \lambda \beta_j) y_{n-j} = 0 \qquad \text{(verify)}$$

$\rightarrow$ Linear, constant-coefficient difference equation solutions of the form $\quad \xi_i^n \quad$ where $\quad \xi_i \quad$ is a root of

$$\rho(\xi) - \Delta t \lambda \rho(\xi) = 0.$$

For $\mathcal{R}e(\lambda) < 0$, we want $|y_n| < |y_{n-1}|$.
This is not possible if there are extraneous roots of $\rho(\xi)$ with magnitude 1.

Recall, for consistency, we need $\rho(1) = 0$.
This is called the principal root.
All others are called "extraneous roots".

This leads us to the following definitions:

Definition 1. *A LMM is*

- *strongly stable if all extraneous roots satisfy*

$$|\xi_i| < 1;$$

- *weakly stable if at least one extraneous root satisfies*

$$|\xi_i| = 1,$$

and the method is 0-stable $(|\xi_i| \not> 1)$.

Example 6. *Weak stability is dangerous!*
Consider Milne's method

$$y_n = y_{n-2} + \frac{\Delta t}{3}(f_n + 4f_{n-1} + f_{n-2})$$

applied to $\dot{y} = \lambda y$; *the error* e_n *satisfies*

$$e_n = e_{n-2} + \tfrac{\Delta t\lambda}{3}(e_n + 4e_{n-1} + e_{n-2}). \qquad \textit{(verify)}$$

Guessing $e_n = \xi^n$, *we have*

$$(1 - \tfrac{1}{3}\Delta t\lambda)\xi^2 - \tfrac{4}{3}\Delta t\lambda\xi - (1 + \tfrac{1}{3}\Delta t\lambda) = 0. \qquad \textit{(verify)}$$

The roots are

$$\xi = \frac{\frac{2}{3}\Delta t\lambda \pm \sqrt{1 + \frac{1}{3}(\Delta t\lambda)^2}}{1 - \frac{1}{3}\Delta t\lambda}.$$

Expand to yield

$$\begin{aligned} \xi_1 &= e^{\Delta t\lambda} + \mathcal{O}(\Delta t\lambda)^5 && \leftarrow \textit{principal} \\ \xi_2 &= -e^{-(\frac{\Delta t\lambda}{3})} + \mathcal{O}((\Delta t)^3) && \leftarrow \textit{extraneous} \end{aligned}$$

If $\mathcal{R}e(\lambda) < 0, \quad |\xi_2| \to \infty$*! Solution is unstable!*

$\Longrightarrow$ Any useful LMM must be strongly stable.

Consequently, Dahlquist proved that

A strongly stable k-step LMM has order at most $k+1$.

Example 7. *All Adams methods have*

$$\rho(\xi) = \xi^k - \xi^{k-1} = \xi^{k-1}(\xi - 1).$$

$\to$ *Extraneous roots all zero! (*"optimally" *strongly stable)*

$\to$ *AM have optimal order.*

Example 8. *BDF methods are not 0-stable for* $k > 6$*!*

5.3 Absolute Stability

Recall: Absolute stability is the property

$$|y_n| \leq |y_{n-1}|$$

when the numerical scheme is applied to $\dot{y} = \lambda y$ with $\mathcal{R}e(\lambda) \leq 0$.

A method is A-stable if its region of absolute stability contains the left half-plane $\Delta t \mathcal{R}e(\lambda) < 0$.

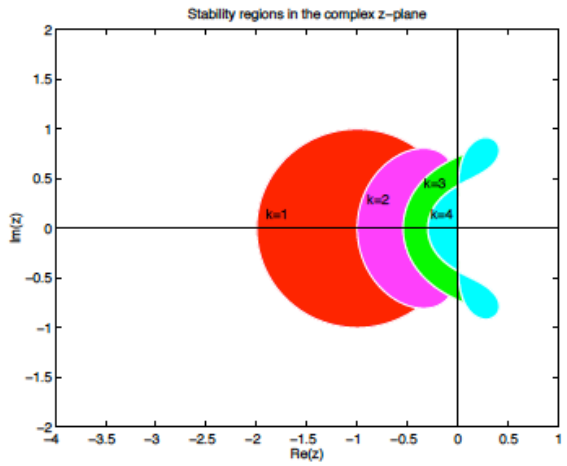
A-stability is very difficult for LMMs to attain!

We have the following results from Dahlquist (1960s):

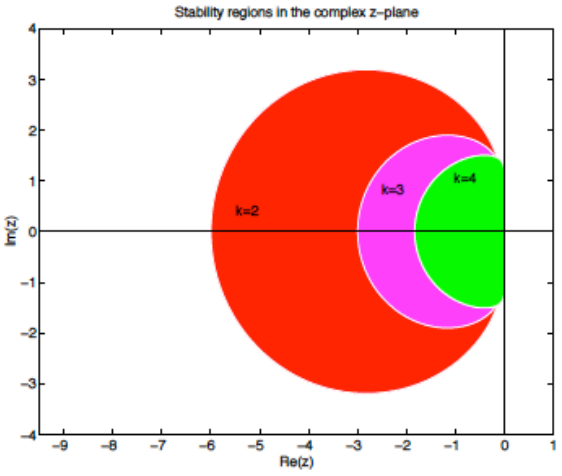
- An explicit LMM cannot be A-stable.
- An A-stable LMM cannot have $p > 2$.
- The trapezoidal method is the "best" second-order, A-stable LMM in terms of error constant $(C_3 = \frac{1}{12})$.

For stiff problems, A-stability may not be crucial.
$\rightarrow$ Stiff decay may be more useful!

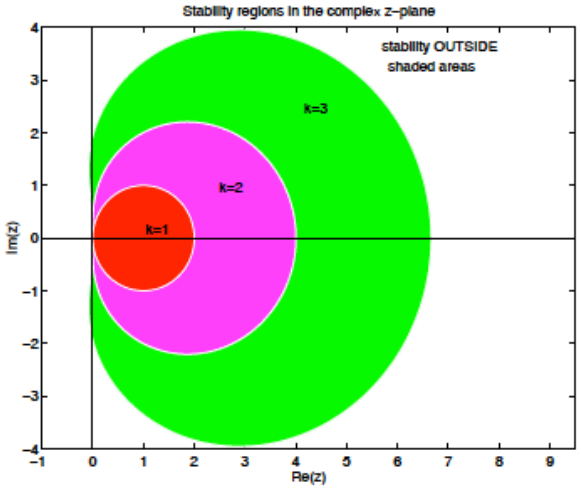
BDF methods sacrifice A-stability for stiff decay.



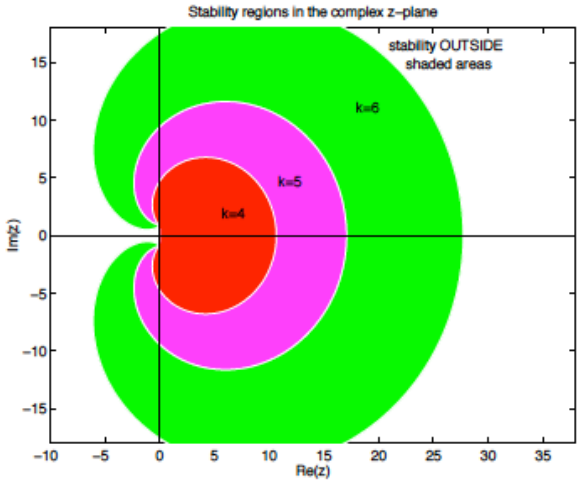
(a) Adams-Bashforth $k = 1, 2, 3, 4$



(b) Adams-Moulton $k = 2, 3, 4$



(a) BDF $k = 1, 2, 3$



(b) BDF $k = 4, 5, 6$

5.4 Implementation of LMMs

Recall the implicit k-step linear multistep method

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = \Delta t \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}, \qquad \beta_0 \neq 0.$$

$\rightarrow$ Solve a system of m nonlinear equations each step.

Use functional iteration (for non-stiff systems) or (modified) Newton iteration for (stiff systems).

The initial guess can be from using an interpolant of past $\mathbf{y}$ or $\mathbf{f}$ values or via an explicit LMM.

5.4.1 Implicit LMMs

- Functional iteration

$$\mathbf{y}_n^{(\nu+1)} = \Delta t \beta_0 \mathbf{f}(t_n, \mathbf{y}_n^{(\nu)}) - \sum_{j=1}^{k} \alpha_j \mathbf{y}_{n-j} + \Delta t \sum_{j=1}^{k} \beta_j \mathbf{f}_{n-j},$$

$$\nu = 0, 1, \cdots .$$

Note 10.

- *Only appropriate for nonstiff problems*
- *Iterate to "convergence" (as described below).*
- *If no convergence in 2-3 iterations, or rate of convergence too slow, reject current step and retry with smaller step size.*

5.4.2 Predictor-Corrector Methods

Often in codes for non-stiff problems, nonlinear equations are not solved "exactly", i.e., down to a small multiple of unit roundoff.
Instead, only a fixed number of iterations is used.

- First use an explicit LMM to predict $\mathbf{y}_n^{(0)}$.
  (This is "better" than predicting $\mathbf{y}_n^{(0)} = \mathbf{y}_{n-1}$.)
  e.g., k-step AB of order k

$$P: \quad \mathbf{y}_n^{(0)} = -\hat{\alpha}_1 \mathbf{y}_{n-1} - \cdots - \hat{\alpha}_k \mathbf{y}_{n-k} + \Delta t[\hat{\beta}_1 \mathbf{f}_{n-1} + \cdots \hat{\beta}_k \mathbf{f}_{n-k}].$$

- Evaluate right-hand side

$$E: \quad \mathbf{f}_n^{(0)} = \mathbf{f}(t_n, \mathbf{y}_n^{(0)}).$$

- Correct using implicit LMM,
  e.g., k-step AM of order $k+1$

$$C: \quad \mathbf{y}_n^{(1)} = -\alpha_1 \mathbf{y}_{n-1} - \cdots - \alpha_k \mathbf{y}_{n-k} + \Delta t[\beta_0 \mathbf{f}_n^{(0)} + \cdots + \beta_k \mathbf{f}_{n-k}].$$

We can stop here, (PEC method)
or the steps (EC) can be iterated ν times:
$\rightarrow$ a P(EC)$^\nu$ method.

It is advantageous (and natural!) to end with E step.

$\rightarrow$ Use best guess for $\mathbf{y}_n$ in $\mathbf{f}_{n-1}$ for the next step.

This turns out to significantly enhance the region of absolute stability over the P(EC)$^\nu$ method.

The most common scheme is PECE ($\nu = 1$).

Note 11. • *This is an explicit method.*

- *Because corrector is not iterated to convergence, the order, error, and stability properties are not generally the same as those of the corrector.*

- *Should only be used for non-stiff problems.*

Example 9. *2-step AB predictor + 1-step (order 2) AM corrector*

Given $\mathbf{y}_{n-1}, \quad \mathbf{f}_{n-1}, \quad \mathbf{f}_{n-2},$

$$
\begin{aligned}
&1.\ P: \quad \mathbf{y}_n^{(0)} &&= \mathbf{y}_{n-1} + \frac{\Delta t}{2}[3\mathbf{f}_{n-1} - \mathbf{f}_{n-2}] \\
&2.\ E: \quad \mathbf{f}_n^{(0)} &&= \mathbf{f}(t_n, \mathbf{y}_n^{(0)}) \\
&3.\ C: \quad \mathbf{y}_n &&= \mathbf{y}_{n-1} + \frac{\Delta t}{2}[\mathbf{f}_n^{(0)} + \mathbf{f}_{n-1}] \\
&4.\ E: \quad \mathbf{f}_n &&= \mathbf{f}(t_n, \mathbf{y}_n)
\end{aligned}
$$

$\rightarrow$ *Explicit, 2nd-order method with LTE*

$$\mathbf{d}_n = -\tfrac{1}{12}(\Delta t)^2 \dddot{\mathbf{y}}(t_n) + \mathcal{O}((\Delta t)^3)$$

(same as $\mathbf{d}_n$ *for corrector)*

$\downarrow$

This is always the case.

(roughly because $\mathbf{y}_n^{(0)}$ *is also order* $k+1$ *and enters the corrector formula multiplied by* Δt*.)*

5.4.3 Modified Newton Iteration

For stiff problems, you need some kind of Newton iteration to solve the nonlinear equations.

$$\mathbf{y}_n - \Delta t \beta_0 \mathbf{f}(t_n, \mathbf{y}_n) = \underbrace{-\sum_{j=1}^{k} [\alpha_j \mathbf{y}_{n-j} + \Delta t \beta_j \mathbf{f}_{n-j}]}_{\text{known!}}$$

Newton's iteration:

$$\mathbf{y}_n^{(\nu+1)} = \mathbf{y}_{n-1}^{(\nu)} - \left[I - \Delta t \beta_0 \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{y}}}_{\text{at } \mathbf{y}=\mathbf{y}_n^{(\nu)}} \right]^{-1}$$

$$(\alpha_0 \mathbf{y}_{n-1}^{(\nu)} + \sum_{j=1}^{k} \alpha_j \mathbf{y}_{n-j} + \Delta t \beta_j \mathbf{f}_{n-j}). \quad \text{(verify)}$$

Initial guess $\mathbf{y}_n^{(0)}$ from interpolant through past values.
$\rightarrow$ Not the cheapest possible!

Idea: Update $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ and LU factors of $\left[I - \Delta t \beta_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]$ only when necessary!
e.g.,

- Iteration does not converge.
- Stepsize changed significantly.
- Order changed.
- Some number of steps have passed.

Iteration has converged when

$$\frac{\rho}{1-\rho}|\mathbf{y}_n^{(\nu+1)} - \mathbf{y}_n^{(\nu)}| < NTOL,$$

$NTOL$: Newton tolerance $\approx \frac{1}{3} ETOL$

where ρ is a measure of the convergence rate

$$\rho = \left[\frac{|\mathbf{y}_n^{(\nu+1)} - \mathbf{y}_n^{(\nu)}|}{|\mathbf{y}_n^{(1)} - \mathbf{y}_n^{(0)}|}\right]^{\frac{1}{\nu}}, \qquad \nu = 1, 2, \cdots .$$

5.5 Software Design Issues

- Error estimation and control

  $\updownarrow$

  varying step size varying order

- Solving nonlinear algebraic equations (done)

Less straightforward than for one-step methods!

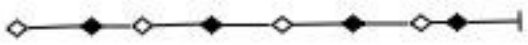
5.5.1 Variable Stepsize Formulas

$\rightarrow$ A crucial part of a practical code!

Recall, $\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = \Delta t \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}$
assumes k equal steps!

If we change the stepsize to $\Delta t_n \quad (\neq \Delta t)$,
then we need $(k-1)$ new values at points
$t_{n-1} - \Delta t_n, \quad t_{n-1} - 2\Delta t_n, \quad \cdots, \quad t_{n-1} - (k-1)\Delta t_n.$

We had
$t_{n-1} - \Delta t, \quad t_{n-1} - 2\Delta t, \quad \cdots, \quad t_{n-1} - (k-1)\Delta t.$



There are 3 main strategies to obtain the missing values.

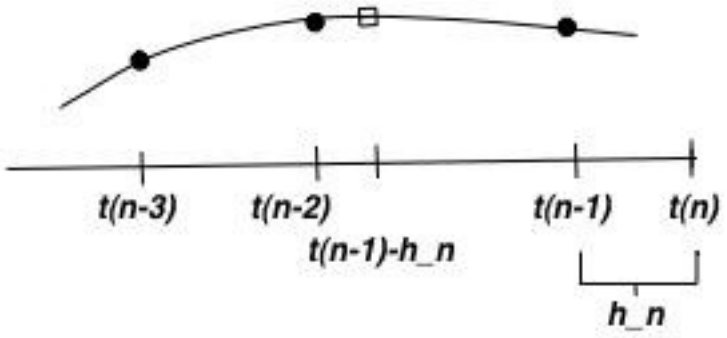
We now illustrate in terms of BDF2.
This requires interpolation in $\mathbf{y}$.

For Adams methods, the interpolation is in terms of past values of $\mathbf{f}$.

- Fixed-coefficient strategy: In this case, we want to use the formula with fixed coefficients $\leftrightarrow$ constant Δt, so we generate the missing solution values at evenly spaced points by interpolation.

  BDF2 $\quad \frac{3}{2}(\mathbf{y}_n - \frac{4}{3}\mathbf{y}_{n-1} + \frac{1}{3}\mathbf{y}_{n-2}) = \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n)$

  $\rightarrow$ Suppose we want to take a step Δt_n having just taken steps $\Delta t_{n-1}, \Delta t_{n-2}$.
  Do quadratic (polynomial) interpolation to get $\mathbf{y}$ at $t_{n-1} + \Delta t_n$.



Advantage: simple!
Method coefficients can be pre-computed.
Work is proportional to number of equations m.

Disadvantage: interpolation error
Interpolation must be performed every time Δt changes.
Theoretical and empirical evidence shows them to be less stable than other methods.
e.g., Gear, Tu, and Watanabe show that it is not hard to come up with examples that are unstable because of this strategy.

$\rightarrow$ Important when Δt_n needs to be changed often or by a large amount.

- Variable-coefficient strategy: coeffs depend on Δt.

  This strategy does not require that past values be evenly spaced.

  It turns out that stability is better when formulas are derived based on unequally spaced data.

  This allows codes to change step size and order more frequently (and drastically), but it is less efficient than the alternatives.

  In fact, this makes the variable-coefficient strategies the method of choice for variable step size codes applied to non-stiff problems.

  Recall, BDF was derived by interpolating past $\mathbf{y}$ values, then forcing the derivative of the interpolant to agree with $\mathbf{f}$ at $t = t_n$.

  Now do the same thing but without constant step! Use Newton interpolant form: (BDF2)

$$\begin{aligned} \boldsymbol{\phi}(t) &= \mathbf{y}_n + [\mathbf{y}_n, \mathbf{y}_{n-1}](t - t_n) \\ &\quad + [\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{y}_{n-2}](t - t_n)(t - t_{n-1}). \end{aligned}$$

Then,

$$\dot{\phi}(t) = [\mathbf{y}_n, \mathbf{y}_{n-1}] + [\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{y}_{n-2}](t_n - t_{n-1}).$$

(verify)

So the collocation condition $\dot{\phi}(t_n) = \mathbf{f}(t_n, \mathbf{y}_n)$ translates to

$$\mathbf{f}(t_n, \mathbf{y}_n) = [\mathbf{y}_n, \mathbf{y}_{n-1}] + \Delta t_n [\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{y}_{n-2}]. \quad (*)$$

Writing out the divided difference in (*),

$$\begin{aligned} \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n) &= \mathbf{y}_n - \mathbf{y}_{n-1} \\ &+ \frac{(\Delta t_n)^2}{\Delta t_n + \Delta t_{n-1}} \left[\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{\Delta t_n} - \frac{\mathbf{y}_{n-1} - \mathbf{y}_{n-2}}{\Delta t_{n-1}} \right]. \end{aligned}$$

(verify)

$\rightarrow$ The Newton iteration matrix is thus

$$\left[\left(1 + \frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}} \right) \mathbf{I} - \Delta t_n \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right].$$

(verify)

In general, dependence on previous $k-1$ stepsizes!

$\rightarrow$ Not possible in practice to effectively freeze the Newton iteration matrix.

- Fixed leading-coefficient strategy:

  $\rightarrow$ An "optimal" tradeoff between the efficiency of fixed coefficients and the stability of variable ones.

  Demonstrate on k-step BDF.
  Construct *predictor* polynomial that interpolates the (unequally spaced) $\quad \mathbf{y}_{n-i}, \quad i = 1, 2, \cdots, k+1,$

$$\boldsymbol{\phi}(t_{n-i}) = \mathbf{y}_{n-i}.$$

  Now require a second interpolating polynomial to match the predictor polynomial on a uniform mesh and satisfy the ODE at $t = t_n$:

$$\begin{aligned} \boldsymbol{\psi}(t_n - i\Delta t_n) &= \boldsymbol{\phi}(t_n - i\Delta t_n), \quad i = 1, 2, \cdots, k, \\ \boldsymbol{\psi}'(t_n) &= \mathbf{f}(t_n, \boldsymbol{\psi}(\boldsymbol{t_n})). \end{aligned}$$

  Then take $\qquad \mathbf{y}_n = \boldsymbol{\psi}(\boldsymbol{t_n}).$

- Stability properties are intermediate to other two strategies, but efficiency matches fixed-coefficient.

5.5.2 Estimating and controlling local error

As usual, local error is easier to control than global error. (why?)

Recall, $\Delta t_n(\|\mathbf{d}_n\| + \mathcal{O}((\Delta t)^{p+1})) = \|\mathbf{l}_n\|(1 + \mathcal{O}(\Delta t_n))$.

$\rightarrow$ Codes try to estimate and control $\Delta t_n \|\mathbf{d}_n\|$.

To make life easier, assume "starting values" exact. For predictor-corrector methods, error estimate can be expressed in term of P-C difference:

Predictor: $\hat{\mathbf{d}}_n = \hat{C}_{p+1}(\Delta t)^p \mathbf{y}^{(p+1)}(t_n) + \mathcal{O}((\Delta t)^{p+1})$.

Now consider

$\mathbf{y}_n - \mathbf{y}_n^{(0)} = (C_{p+1} - \hat{C}_{p+1})(\Delta t)^p \mathbf{y}^{(p+1)}(t_n) + \mathcal{O}((\Delta t)^{p+1})$.

$\mathbf{y}_n^{(0)}$	:	predicted
$\mathbf{y}_n$	:	corrected
$(\Delta t)^p \mathbf{y}^{(p+1)}$	:	"solve" for $\mathbf{d}_n$ below

$\therefore$ Corrector error

$$\begin{aligned}\mathbf{d}_n &= C_{p+1}\overbrace{(\Delta t)^p\mathbf{y}^{(p+1)}}(t_n) + \mathcal{O}((\Delta t)^{p+1}) \\ &= \frac{C_{p+1}}{C_{p+1} - \hat{C}_{p+1}}(\mathbf{y}_n - \mathbf{y}_n^{(0)}) + \mathcal{O}((\Delta t)^{p+1}).\end{aligned}$$

Milne's estimate

Note 12. *Use difference of approximations of same order (unlike embedded RK).*

e.g., k-step AB + $(k-1)$-step AM
$\Rightarrow$ $\mathcal{O}((\Delta t)^k)$ predictor-corrector with two function evaluations per step.

Note 13.

- *If predictor is order $k-1$, then advancing with corrected value of order k is local extrapolation.*

- *For general LMMs, $\mathbf{d}_n$ estimated directly using divided differences; e.g., for BDF2, if $\phi(t)$ is the quadratic through $\mathbf{y}_n$, $\mathbf{y}_{n-1}$, and $\mathbf{y}_{n-2}$, then*

$$\mathbf{f}_n = \dot{\phi}(t_n) = [\mathbf{y}_n, \mathbf{y}_{n-1}] + \Delta t_n[\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{y}_{n-2}] + \mathbf{r}_n,$$

where

$$\mathbf{r}_n = \Delta t_n(\Delta t_n + \Delta t_{n-1})[\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{y}_{n-2}, \mathbf{y}_{n-3}].$$

Then

$$\mathbf{d}_n \approx \beta_0 \mathbf{r}_n.$$

- *Step accepted if*

$$\Delta t \qquad \underbrace{\|\mathbf{d}_n\|}_{\text{estimated truncation error}} \qquad \leq ETOL.$$

Choosing stepsize and order for next step.

Begin by forming estimates of error by methods of order $p-2, p-1, \underbrace{p}_{\text{current order}}, p+1$

Then ...

- Choose next order to maximize Δt.
- Raise or lower the order according to whether

  $\|(\Delta t)^{p-1}\mathbf{y}^{(p-1)}\|, \|(\Delta t)^{p}\mathbf{y}^{(p)}\|, \|(\Delta t)^{p+1}\mathbf{y}^{(p+1)}\|, \|(\Delta t)^{p+2}\mathbf{y}^{(p+2)}\|$

  is increasing or decreasing.

If size of terms decreases ...
Taylor series is behaving (increase order);
Else lower order.

Now given order $\hat{p}$ for next step,
choose $$\Delta t_{n+1} = \alpha \Delta t_n,$$

where $$\|\alpha^{\hat{p}+1} \underbrace{(\Delta t_n)^{\hat{p}+1} C_{\hat{p}+1}\mathbf{y}^{(\hat{p}+1)}}_{EST}\| = \underbrace{frac}_{0.9} ETOL$$

$$\Rightarrow \qquad \alpha = \left(\frac{fracETOL}{EST}\right)^{\frac{1}{\hat{p}+1}}.$$

5.5.3 Off-step points

What if you need the solution at a non-mesh point ?

$\rightarrow$ Easy and cheap to construct polynomial interpolant. (Harder for one-step methods!)

But note that the natural interpolant for BDF is continuous (but not differentiable), and the natural interpolant for Adams methods is not continuous (but its derivative "is").

Interpolants with higher orders of smoothness are known.