Multistep methods

The other major class of nonstiff solvers use previous values of y_k

The best-behaved are the **Adams methods** (Adams, 1883)

Go back to the quadrature formula:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau))\, d\tau$$

Basic idea:

- use previous f values $f_k \equiv f(t_k, y_k), k = n, n-1, \cdots$ to construct a polynomial interpolant
- integrate the interpolant (extrapolation) to get y_{n+1}

Adams-Bashforth methods

A family of **explicit** multistep methods

Example

Use f_n only $\rightarrow$ Euler's Method = AB1

our previous derivation using LH rectangle rule

Example

Use f_n, f_{n-1} $\rightarrow$ linear interpolant $\rightarrow$ 2nd order 2-step method AB2

$$y_{n+1} = y_n + h[\frac{3}{2}f_n - \frac{1}{2}f_{n-1}]$$

Note: we need 2 starting values y_0, y_1 to start this

Multistep methods are not self-starting

Need to use another method to start them off.

Why bother?

Same idea $\rightarrow$ AB3:

$$y_{n+1} = y_n + \frac{h}{12}[23f_n - 16f_{n-1} + 5f_{n-2}]$$

AB4 etc.
Why bother?

AB methods need only 1 function evaluation per step

$\Longrightarrow$ very efficient compared to RK methods!

Properties of AB methods

stability analysis more complicated for multistep methods

i.e. harder to guarantee 0-stability

But Adams methods are well-behaved ...

All Adams methods are 0-stable
$\implies$ **All Adams methods are convergent**

For Adams-Bashforth methods, regions of absolute stability shrink rapidly with increasing order (Diagram)

To improve linear stability behavior look at **implicit methods**.

Adams-Moulton methods

Back to derivation but now include f_{n+1} in the interpolant

$\rightarrow$ y_{n+1} appears on both sides of the equation, so must solve a nonlinear equation (or system) at each time step

$\rightarrow$ a family of implicit multistep methods — **Adams-Moulton methods** (Adams 1883)

Backward Euler method

Example

0th degree interpolant thru f_{n+1}: use RH rectangle rule

$$y_{n+1} = y_n + hf_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Backward Euler method = AM0 (1st order implicit method)

RAS: $\{h\lambda \in \mathbb{C} : |1 - h\lambda| > 1\}$

Backward Euler method is 1st order convergent and A-stable!

Trapezoid method

Example

put linear interpolant thru f_n, f_{n+1} and integrate over $[t_n, t_{n+1}]$ (trapezoid rule) $\rightarrow$ **implicit trapezoid rule** $=$ AM1

$$y_{n+1} = y_n + \frac{h}{2}[f_n + f_{n+1}] = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$$

Error of trapezoid rule $\sim h^3 \rightarrow$ 2nd order method
RAS: $\{h\lambda \in \mathbb{C} : Re(\lambda) < 0\}$

Trapezoid method is 2nd order convergent and A-stable!
Implemented in variable-step manner in MATLAB's `ode23t`

Predictor-corrector methods

In practice, we don't solve AM methods to convergence
instead just do **1 iteration of the fixed point iteration**

Example

Solve trapezoid method

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \equiv g(y_{n+1})$$

using the fixed point iteration

$$y_{n+1}^{(k+1)} = g(y_{n+1}^{(k)}) = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{(k)})]$$

but need initial guess for y_{n+1} !

use corresponding AB method (explicit) of same order to get initial guess $y_{n+1}^{(0)}$
$\rightarrow$ **Predictor-Corrector pair** (Moulton, Milne 1926)

Adams-Bashforth-Moulton

Use AB (explicit) to get initial guess $\hat{y}_{n+1}$ — the **predictor** step
then use AM method (1 iteration of fixed point iteration) to get y_{n+1} — the **corrector** step

Example

AB2-AM1 (both 2nd order)

$\hat{y}_{n+1} = y_n + h[\frac{3}{2} f_n - \frac{1}{2} f_{n-1}]$ **P**redict

$\hat{f}_{n+1} = f(t_{n+1}, \hat{y}_{n+1})$ **E**valuate

$y_{n+1} = y_n + \frac{h}{2}[f_n + \hat{f}_{n+1}]$ **C**orrect

$f_{n+1} = f(t_{n+1}, y_{n+1})$ **E**valuate (ready for next step)

PECE

Overall $\rightarrow$ an explicit method with 2 function evaluations per step but much bigger stability regions (than AB)

Modern Adams codes

In practice, modern Adams codes are variable-step (messy) and variable order — not for amateurs

Example

Mathematica's `NDSolve` (500,000 lines of C)

Example

MATLAB's `ode113` — Adams methods of orders 1–13
(700 lines of MATLAB)

Summary of nonstiff solvers

1-step (RK)	Multistep (Adams)
expensive for high order p	easy to get high order
$\geq p$ Fevals per step	2 Fevals per step (Pred-Corr)
stability region grows with order	stability region shrinks with order
0-stable $\implies$ convergent	0-stable $\implies$ convergent
variable-step $\rightarrow$ simple code	variable step/order $\rightarrow$ complex code
self-starting	start with low order or RK
general purpose solver	best if problem very smooth
	or tight tolerances
	or f **very** expensive

Stiff solvers

The methods we have seen so far (except implicit trapezoid method = AM1) have had finite regions of absolute stability.

$\implies$ if $Re(hJ) \ll 0$, the stepsize will need to be tiny for accurate numerical solution

$\implies$ h **determined by stability, not accuracy**

What happens if you try a nonstiff variable-step solver (e.g. ode23) on a stiff problem?

Demo

A nonstiff solver on a stiff problem

If h is outside stability region, the 2 estimates will vary a lot (due to error growth)
$\rightarrow$ the solver thinks h is too big (for accuracy reasons) so cuts h down
$\Longrightarrow$ solver keeps going happily, just with very small steps, so as to keep h inside the stability region

Moral: a nonstiff solver will solve a stiff problem — just takes a long time to do it!

The Dahlquist Barrier

To avoid this problem need methods with unbounded stability domains, such as A-stable methods

The Backward Euler (implicit Trapezoid) methods are A-stable, but only 1st (2nd) order resp.

Can we do better? NO

Theorem

No A-stable multistep method has order > 2 (Dahlquist 1963)

To get around the Dahlquist barrier, we weaken our requirement from A-stability.

Problems with sharp transients typically have λ near the negative real axis, not near the imaginary axis (**oscillatory ODEs**)

$\implies$ A-stability is too strong for most stiff problems

$A(\alpha)$-stability

$\implies$ **weaken A-stability** to require

$$|y_{n+1}| < |y_n|$$

for λh in a wedge about negative real axis i.e. doesn't include region near imaginary axis $\rightarrow$ $A(\alpha)$**-stability**
then the Dahlquist barrier doesn't apply
$\rightarrow$ higher order methods?

BDF methods

The first common class of methods to have these properties were the **BDF methods** (Curtiss & Hirschfelder 1953, Gear 1971), which we get by approximating y' not $\int_{t_n}^{t_{n+1}} f \, d\tau$. Start from

$$y'(t_{n+1}) = f(t_{n+1}, y_{n+1})$$

$\Longrightarrow$ we'll have only 1 f i.e. f_{n+1} $\rightarrow$ an implicit method

Now approximate the LHS by the derivative of the polynomial interpolant thru $t_{n+1}, t_n, t_{n-1} \cdots$

Example

Use $t_{n+1}, t_n \rightarrow$ **B**ackward **D**ifference **F**ormula

$y' \approx \frac{y_{n+1} - y_n}{h}$

$\rightarrow$ Backward Euler method =AM0= BDF1

BDF1-5

Example

Use $t_{n+1}, t_n, t_{n-1} \to$

$$y_{n+1} - \frac{4}{3}y_n + \frac{1}{3}y_{n-1} = \frac{2}{3}hf_{n+1}$$

$=$ BDF2 — a 2nd order A-stable, L-stable method

Proceeding in the same way $\to$ BDF3, BDF4, BDF5 with stability regions showing $A(\alpha)$ stability.

BDF methods have traded in A-stability near the imaginary axis for L-stability $\to$ first general purpose stiff solvers

Solving implicit methods

These stability regions only hold if we solve nonlinear equations to convergence – how to do this?

Example

use Backward Euler to illustrate

Can we use fixed point iteration?

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \equiv g(y_{n+1})$$

Fixed point iteration only converges if $\mid g'(x^*) \mid < 1$ (a **contraction mapping**)

In our case $g' = h\frac{\partial f}{\partial y} = hJ$ so the restriction $\mid g'(x^*) \mid < 1 \implies$

$$\mid hJ \mid < 1$$

which then puts the same restrictions on stepsize that we must lift to solve stiff problems

Moral: cannot use fixed-point iteration to solve implicit equations for stiff problems

Modern stiff solvers

Variable-step variable-order codes based on BDF or similar methods

Example

Mathematica's `NDSolve` switches between Adams methods and BDF methods when it thinks DE has become stiff.

Example

Matlab's `ode15s` uses similar methods and has an option to use BDF

All MATLAB's IVP solvers have the same calling sequence
$\Longrightarrow$ just replace `ode23` by `ode113` or `ode15s` to use a different solver!

Is that all?

NO

Implicit RK solvers (Butcher,1964):

- are all convergent
- can be A-stable and high-order
- can be L-stable and high-order
- can be proved stable for nonlinear nonautonomous problems

BUT take a lot more work! :-(

Example

MATLAB `ode23tb`
2nd order A-stable, AND L-stable

End of Week 11