Math 405: 6b: More on Initial Value Problems

Runge-Kutta methods

We've seen several of these (forward Euler, improved Euler, RK4). They use temporary intermediate "stage values" to advance from U^n to U^{n+1}.

Matlab's code "ode45" uses Runge-Kutta methods.

Linear-Multistep Methods

Uses previous step values (e.g., U^{n-1} and U^n) to advance to U^{n+1}.

In general: $\sum_{i=0}^{r} \alpha_i U^{n+1-i} = k \sum_{i=0}^{r} \beta_i f(U^{n+1-i})$.

Example 1: [demo_06_consistent_but_unstab.m]

$U^{n+1} = 3U^n - 2U^{n-1} - kf(t-k, U^{n-1})$

Consistency: yes, via our usual Taylor series analysis.

Zero-stability: apply method to $u' = 0$, get a *difference equation*. How to solve? Guess an "ansatz", $U^n = \xi^n$ (latter is a power).

We see zero-stability does not follow automatically from consistency (as it does for one-step methods). An important difference here is that the linear multistep methods are not "self starting".

Example 2: Adams-Bashforth-2 method, explicit

$U^{n+1} = U^n + k\left(3/2f(t, U^n) - 1/2f(t-k, U^{n-1})\right)$

Consistency: yes, $O(k^2)$.

Example 3: BDF-2 method (backward-differentation-formula)

$U^{n+1} - 4/3U^n + 1/3U^{n-1} = 2/3kf(t+k, U^{n+1})$

Consistency: $O(k^2)$

Look at absolute stability analysis—it is A-stable and L-stable.

- Second Dahlquist barrier: there are no explicit A-stable linear multistep methods. Implicit A-stable linear multistep methods have order at most 2.
- Matlab's code "ode15s" uses implicit linear multistep methods with variable order.
- A version of our "fundamental thm": Dalquist Equivalence Thm: consistency + zero-stability of linear multistep methods implies convergence.

Implicit time-stepping methods

(see earlier in notes for Backward Euler and Trapezoidal Rule methods).

$u^{n+1} = u^n + kf(u^{n+1})$

u^{n+1} on RHS makes this more *expensive* than forward Euler.

Advantages? A-stability. Some implicit methods (including BE, TR and BDF-2) have no time-step restriction for (absolute) stability.

Implicit methods are often useful for *stiff problems*.

Stiffness?

The classical zero-stability/consistency/convergence theory for ODEs was established by Dahlquist in 1956. A few years later it began to be widely appreciated that something was missing from this theory. Key paper: [Dahlquist, 1963]

(Chemists Curtiss & Hirschfelder [1952], used the term "stiff", which may actually have originated with the statistician John Tukey (who also invented "FFT" and "bit").

Example

ODE $u' = -\sin(t)$ with IC $u(0) = 1$ has solution $u(t) = \cos(t)$.

Change ODE to

$$u' = -100(u(t) - \cos(t)) - \sin(t),$$

then this *still* has solution $u(t) = \cos(t)$.

But the numerics are much different: [demo_06_stiff.m], a convergence study showing forward Euler/backward Euler convergence on these problems.

Absolute stability is right tool in theory (to understand) and in practice (to deal with) stiffness.

Analysis (for this example): linearize around the soln: let $u(t) = \cos(t) + w(t)$ and we get an ODE for $w(t)$ of $w' = -100w$ which does indeed have a very different *time scale* than $\cos(t)$.

Definition of stiffness

- A stiff ODE is one with widely varying time scales.
- More precisely, an ODE with solution of interest $u(t)$ is stiff when there are time scales present in the equation that are much shorter than that of $u(t)$ itself.

Stiffness ratio: smallest eigenvalue to largest eigenvalue. (Continuous diffusion is infinitely stiff.)

Neither above "definitions" are ideal.

- My favourite: a stiff *problem* is one were implicit methods work better. (I learned this from Raymond Spiteri but probably due to Gear.) C.W. Gear, 1982:

Non-linearity and implicit methods

For nonlinear problems (or nonlinear discretizations of linear problems), implicit methods require solving nonlinear equations at each time-step. Similarly, for nonlinear steady-state problems.

For this, see Newton's method, covered earlier in the course.

IMEX methods

Implicit/Explicit methods. Best of both worlds? Treat some part of equation (often linear diffusion or hyperdiffusion) implicitly to avoid time-step restrictions. But treat the nonlinear terms explicitly (to avoid nonlinear system solves).

Example: [demo_06_kuramoto_sivashinsky.m]

$$u_t = -u_{xx} - u_{xxxx} - (u^2/2)_x$$