Numerical Solutions of Ordinary Differential Equations – Accuracy, Stability and Systems of Equations

Larry Caretto
Mechanical Engineering 501AB

Seminar in Engineering Analysis

November 20, 2017

California State University Northridge

Outline

- Review last class
- Analysis of numerical algorithms
- Stability of numerical solutions
  - Explicit *versus* implicit approaches
- Step size variation for error control
- Error control for multistep methods with constant and variable step size

California State University Northridge 2

Review Implicit Methods

- Methods discussed previously are called explicit
  - Can find y_{n+1} in terms of values at n
  - Use predictors to estimate y values between y_n and y_{n+1}
- Implicit methods use f_{n+1} in algorithm
- Usually require approximate solution
- Have better stability but require more work than explicit methods
- Trapezoid method is an example

California State University Northridge 3

Review Trapezoid Method

- Algorithm $y_{n+1} - y_n = (f_{n+1} + f_n)h/2 + O(h^3)$
- Have to handle f_{n+1} dependence of y_{n+1}
- Simple iteration $y_{n+1}^{(m+1)} = y_n + \left[f_n + f\left(x_{n+1}, y_{n+1}^{(m)}\right)\right]h/2$

$$y_{n+1}^{(m+1)} = y_{n+1}^{(m)} - \frac{y_{n+1}^{(m)} - y_n - \frac{hf_n}{2} - \frac{hf\left(x_{n+1}, y_{n+1}^{(m)}\right)}{2}}{f\left(x_{n+1}, y_{n+1}^{(m)}\right) - \frac{h}{2}\left(\frac{\partial f}{\partial y}\right)_{n+1}^{(m)}}$$

- Newton iteration
- Taylor series for f_{n+1}

$$(y_{n+1} - y_n) = \frac{hf_n + \left.\frac{\partial f}{\partial x}\right|_n \frac{h^2}{2}}{1 - \frac{h}{2}\left.\frac{\partial f}{\partial y}\right|_n}$$

California State University Northridge 4

Review Multistep Methods

- Multistep methods use information from previous steps for improved accuracy with less work than single step methods
- Need starting procedure that is a single step method
- Derivation based on interpolation polynomials which are then integrated
- Predictor-corrector process
- Derivation provides error estimate

California State University Northridge 5

Review Adams Methods

- Predictor corrector method
- Predictor equation uses four points

$$y_{n+1}^P = y_n + \frac{h}{24}\left(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}\right)$$

- Corrector equation uses four points including point n+1 with predicted y^P

$$y_{n+1}^C = y_n + \frac{h}{24}\left(9 f\left(x_{n+1}, y_{n+1}^P\right) + 19 f_n - 5 f_{n-1} + f_{n-2}\right)$$

California State University Northridge 6

Review Step Size Control

- Get estimate of truncation error, E_C, from predictor-corrector difference

$$E_C = \frac{19}{270}\left(y_{n+1}^P - y_{n+1}^C\right)$$

- If $e_{min} \le E_C \le e_{max}$, do not change h
- If $E_C < e_{min}$ double step size, h
- If $E_C > e_{max}$ half step size, h

California State University Northridge 7

Review Grid Size Changes

- Keep extra values f_{i-4} and f_{i-5} in memory to be ready for grid doubling
  - $f_{i-3,new} = f_{i-5}$; $f_{i-2,new} = f_{i-3}$; $f_{i-1,new} = f_{i-1}$; $f_{i,new} = f_{i+1}$
- Grid halving requires interpolation for missing values in old grid
  - $f_{i-2,new} = f_{i-1}$; $f_{i,new} = f_i$

$$f_{i-1,new} = \frac{1}{128}\left[-5 f_{i-4} + 28 f_{i-3} - 70 f_{i-2} + 140 f_{i-1} + 35 f_i\right]$$

$$f_{i-3,new} = \frac{1}{64}\left[3 f_{i-4} - 16 f_{i-3} + 54 f_{i-2} + 24 f_{i-1} - f_i\right]$$

California State University Northridge 8

Review Extrapolation Methods

- Use infinite series truncation error dependence on h to get better estimate from results on two values of h
- Analyze truncation error as infinite series and eliminate lowest order term
  - True value, $t = n(h) + Ah^m + Bh^{m+a}$

$$t = \frac{2^m n\left(h/2\right) - n(h)}{2^m - 1} + B\left(\frac{1}{2^a} - 1\right)h^{m+a} + \cdots$$

California State University Northridge 9

Review Midpoint Method

- Take big step from x to x + H in n steps
  - Start with results at x and define $z_0 = y(x)$
  - Compute $z_1 = z_0 + hf(x, z_0)$
  - Central difference intermediate steps
    - $z_{m+1} = z_{m-1} + 2h(x+mh, z_m)$ m = 1, 2, .. n-1
  - Final value at x + H, called y_n, is an average of the central difference value, z_n, and a backward difference value $z_{n-1} + hf(x+H, z_n)$
    - $y_n = [\, z_n + z_{n-1} + hf(x+H, z_n)\,] / 2$

California State University Northridge 10

Review Bulirsch-Stoer Method

- Three main ideas
  - Use large step size H and compute results at x + H for several values of n then extrapolate results to h = 0
  - Use midpoint method whose truncation error is $Ah^n + Bh^{n+2} + Ch^{n+4}$ … to improve accuracy of interpolation process
  - Use rational function approximation instead of simple polynomial interpolation for extrapolating to h = 0

California State University Northridge 11

Some Basic Concepts

- A finite difference equation is ***consistent*** with the corresponding differential equation if both equations give the same result as $h \to 0$
- A numerical method is ***convergent*** with the solution of the ODE if the numerical solution approaches the actual solution as $h \to 0$ (with increase in numerical precision at smaller h)
- Mainly theoretical concepts

California State University Northridge 12

More Basic Concepts

- We cannot know the ***accuracy*** of numerical solutions, but we can use error approximations to control step size
- We know the ***order*** of the ***global*** truncation error
- ***Stability*** refers to the ability of a numerical algorithm to damp any errors introduced during the solution
- Unstable solutions grow without bound

California State University Northridge 13

More on Stability

- Finite difference equations in numerical algorithms, when iterated, may numerically increase without bound
- Stability usually is obtained by keeping step size h small, sometimes smaller than the h required for accuracy
- For most ODEs stability is not a problem, but it is for stiff systems of ODEs and for partial differential equations

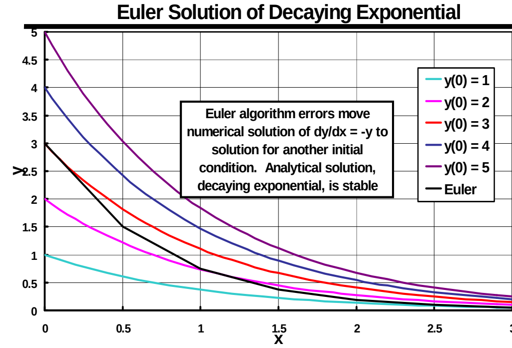
California State University Northridge 14

Stability of Exact Solutions

- Exact solutions to differential equations may be unstable
- Solutions of the form Ce^{at} with $a > 0$ are unstable because they grow without bound as $t \to \infty$
- Judge stability of a numerical method by test on an exact solution that is stable
- Test y' = -ay whose solution is $y = e^{-at}$, where a is a positive constant

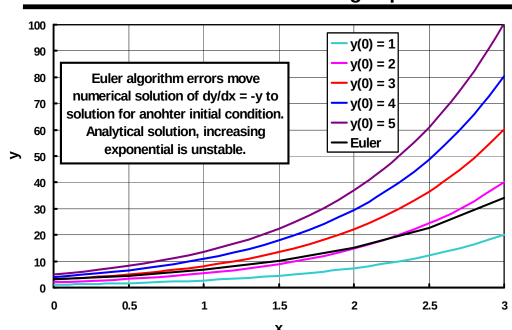
California State University Northridge 15

Euler Solution of Decaying Exponential



California State University Northridge 16

Euler Solution for Increasing Exponential



California State University Northridge 17

Examine Euler Stability

- Look at test equation with y' = f = -ay
- Exact solution is $y = y_0e^{-ax}$ so that y/y_0 is a function of ax
- Euler method: $y_{n+1} = y_n + hf_n$
- With $f_n = -ay_n$ the Euler method equation for y_{n+1}, $y_{n+1} = y_n + hf_n$, becomes $y_{n+1} = y_n + h(-ay_n) = y_n(1 – ah)$
- Compare various numerical solutions to exact solution for different values of ah in following plot of y/y_0 *versus* ax

California State University Northridge 18

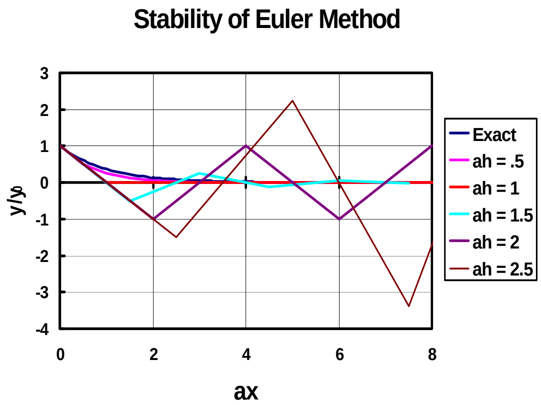


Chart Observations

- Used Euler method: $y_{n+1} = y_n + hf_n$ to solve y' = -ay
- For ah ≤ 1, method looks physically realistic if not accurate
- For 1 < ah ≤ 2, method is not physically realistic but is bounded (stable)
- Method is unstable for ah > 2
- Not shown on chart is that we usually need ah << 2 for accuracy in Euler's method

California State University Northridge 20

General Stability

- Look at trial ODE y' = f = –ay
- Define growth or amplification factor, G = y_{n+1}/y_n
- Euler method has $y_{n+1} = y_n(1 + ah)$ so G = y_{n+1}/y_n = 1 + ah
- For ah ≤ 1 (G ≤ 2), method was physically realistic if not accurate and method was unstable for ah > 2 (G > 3)
- General approach is to seek relation for h (or ah) that keeps G stable

California State University Northridge 21

General Stability II

- Use same test equation with f = -ay with positive a (negative a is unstable ODE)
- Find amplification factor G for method
- If growth is bounded for any combination of physical parameters and step size, h, method is unconditionally stable
- Conditionally stable method is stable only for some combination of step size and physical parameters giving step size limit

California State University Northridge 22

Implicit Methods

- Contrast between implicit and explicit methods discussed previously
  - Explicit methods find y_{n+1} in terms of values at x_n (may use estimated y values between x_n and x_{n+1})
- Implicit methods use f_{n+1} in algorithm
- Require iterative solution or series expansion of derivative expression for f
- Examine stability of trapezoid method for usual test problem y' = –ay

California State University Northridge 23

Implicit Stability

- Trapezoid method equation from previous class – basic equation and computation with series expansion for f_{n+1}

$$y_{n+1} = y_n + \frac{h}{2}\left(f_n + f_{n+1}\right) + O(h^3)$$

$$y_{n+1} = y_n + \frac{hf_n + \left.\frac{\partial f}{\partial x}\right|_n \frac{h^2}{2}}{1 - \frac{h}{2}\left.\frac{\partial f}{\partial y}\right|_n}$$

California State University Northridge 24

Implicit Example

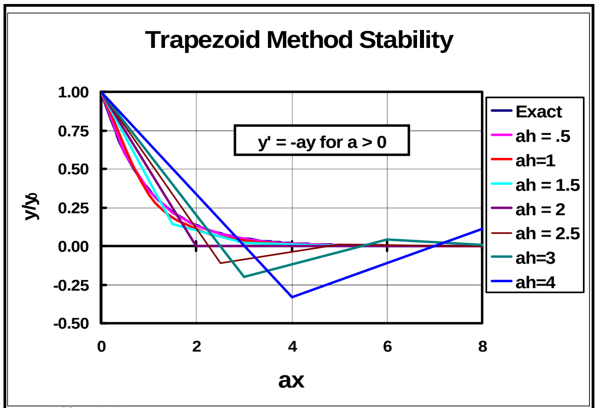
- For dy/dx = f = – ay, ∂f/∂x = 0 and ∂f/∂y = –a

$$y_{n+1} = y_n + \frac{2hf_n + \left.\frac{\partial f}{\partial x}\right|_n h^2}{2 - h\left.\frac{\partial f}{\partial y}\right|_n} = y_n + \frac{-2hay_n + 0}{2 - h(-a)}$$

$$= \frac{y_n(2+ha) - 2hay_n}{2+ha} = \frac{y_n(2-ha)}{2+ha}$$

- Here $y_{n+1} = G\,y_n$ with G = (2 – ha)/(2 + ha)
- |G| < 1 if ha > 0; stable for any h if a > 0

25



26

Chart Observations

- Trapezoid method results much closer to exact solution than Euler results
  - Expected because of $O(h^3)$ local error
- For values of ah > 2, solutions undershoot the final value of y = 0
  - Solutions remain stable, but unrealistic, giving oscillations around y = 0
- Stability is not the same as accuracy
- Must have both

27

Error Control

- How do we choose h to maintain desired accuracy?
- Want to obtain a result with some desired small global error
- Can just repeat calculations with smaller h until two results are sufficiently close
- Can use algorithms that estimate error and adjust step size during the calculation based on the error

28

Runge-Kutta Error Control

- Control error by doing integration with h and 2h along all the integration
  - Integration with 2h step requires 3 additional function evaluations per 2 steps
  - Analyze local truncation error, which is $O(h^5)$ for both steps, at even step locations

$$y_h(x+2h) = y_h + Ah^5 + Bh^6 + \cdots$$
$$y_{2h}(x+2h) = y_{2h} + A(2h)^5 + B(2h)^6 + \cdots$$
$$0 = y_{2h} - y_h + (2^5 - 1)Ah^5 + O(h^6)$$

29

Runge-Kutta Error Control II

- $y_{2h} - y_h = \Delta$ is measure of truncation error
- User specifies Δ_D, the desired error
  - Many ways to specify this, single value, relative values, relative to increments for y in one step
- Since error scales as h^5, we can adjust step size such that $h_{new} = h_{old}|\Delta_D/\Delta|^{1/5}$
- Typically use safety factor to avoid making h_{new} too large

30

Runge-Kutta-Fehlberg

- Uses two equations to compute y_{n+1}, one has $O(h^5)$, the other $O(h^6)$ error
- Requires six derivative evaluations per step (same evaluations used for both equations)
- The error estimate can be used for step size control based on an overall 5th order error
- Cask-Karp version and Runge-Kutta-Verner use same idea

Northridge 31

Runge-Kutta-Fehlberg II

- One algorithm on following slides
- Typical formula components below
- $y_{n+1} = y_n + (16k_1/135 + 6656k_2/12825 \ldots$
- $k_3 = hf(x_n + 3h/8,\ y_n + 3k_1/32 + 9k_2/32)$
- $\text{Error} = k_1/360 - 128k_3/4275 \ldots$
- $h_{new} = h_{old}|E_{Desired}/\text{Error}|^{1/4}$
- $E_{Desired}$ is set by user
- RKF45 code by Watts and Shampine

Northridge 32

RKF45 k Equations

$$k_1 = hf(x_n, y_n) \qquad k_2 = hf\left(x_n + \frac{h}{4}, y_n + \frac{k_1}{4}\right)$$

$$k_3 = hf\left(x_n + \frac{3h}{8}, y_n + \frac{3k_1}{32} + \frac{9k_2}{32}\right)$$

$$k_4 = hf\left(x_n + \frac{12h}{13}, y_n + \frac{1932k_1}{2197} - \frac{7200k_2}{2197} + \frac{7296k_3}{2197}\right)$$

$$k_5 = hf\left(x_n + h, y_n + \frac{439k_1}{216} - 8k_2 + \frac{3680k_3}{513} - \frac{845k_4}{4104}\right)$$

$$k_6 = hf\left(x_n + \frac{h}{2}, y_n - \frac{8k_1}{27} + 2k_2 - \frac{3544k_3}{2565} + \frac{1859k_4}{4104} - \frac{11k_5}{40}\right)$$

Northridge 33

RKF45 Equations for y_{n+1} / h_{new}

$$y_{n+1} = y_n + \frac{16k_1}{135} + \frac{6656k_3}{12825} + \frac{23561k_4}{56430} - \frac{9k_5}{50} + \frac{2k_6}{55} + O(h^5)$$

$$y_{n+1}^* = y_n + \frac{25k_1}{216} + \frac{1408k_3}{2565} + \frac{2197k_4}{4104} - \frac{k_5}{5} + O(h^4)$$

- Difference between y_{n+1} and y^*_{n+1} used for error estimate to adjust step size
  - R_{max} is user-specified maximum error per step

$$h_{new} = 0.84h_{old}\left(\frac{hR_{max}}{|y_{n+1} - y_{n+1}^*|}\right)^{1/4}$$

Northridge 34

Solving Simultaneous ODEs

- Apply same algorithms used for single ODEs
  - Must apply each part of each algorithm step to all equations in system before going on to next step
  - Key is having consistent x and **y** values in determination of $f_i(x,\mathbf{y})$
  - All y_i values in **y** must be available at the same x point when computing the f_i
  - E.g., in Runge-Kutta we must evaluate k_1 for all equations before finding k_2

Northridge 35

Runge-Kutta for ODE System

- $\mathbf{y}_{(n)}$ is vector of dependent variables at $x = x_n$
- $\mathbf{k}_{(1)}$, $\mathbf{k}_{(2)}$, $\mathbf{k}_{(3)}$, and $\mathbf{k}_{(4)}$, are vectors containing intermediate Runge-Kutta results
- $\mathbf{f}$ is a vector containing the derivatives
- $\mathbf{k}_{(1)} = h\mathbf{f} = h\mathbf{f}(x_n, \mathbf{y}_{(n)})$
- $\mathbf{k}_{(2)} = h\mathbf{f}(x_n + h/2, \mathbf{y}_{(n)} + \mathbf{k}_{(1)}/2)$
- $\mathbf{k}_{(3)} = h\mathbf{f}(x_n + h/2, \mathbf{y}_{(n)} + \mathbf{k}_{(2)}/2)$
- $\mathbf{k}_{(4)} = h\mathbf{f}(x_n + h, \mathbf{y}_{(n)} + \mathbf{k}_{(3)})$
- $\mathbf{y}_{(n+1)} = (\mathbf{k}_{(1)} + 2\mathbf{k}_{(2)} + 2\mathbf{k}_{(3)} + \mathbf{k}_{(4)})/6$

Northridge 36

ODE System by RK4

- $dy/dx = -y + z$ and $dz/dx = y - z$ with $y(0) = 1$ and $z(0) = -1$ with $h = .1$
- Details of first step from y_0 to y_1
- $k_{(1)y} = h[-y + z] = 0.1[-1 + (-1)] = -.2$
- $k_{(1)z} = h[y - z] = 0.1[1 - (-1)] = .2$
- $k_{(2)y} = h[-(y+ k_{(1)y}/2) + z + k_{(1)z}/2] = 0.1[-(1 + -0.2/2) + (-1 + .2/2)] = -.18$
- $k_{(2)z} = h[(y+ k_{(1)y}/2) -(z + k_{(1)z}/2)] = 0.1[(1 + -0.2)/2 - (-1 + .2/2)] = .18$

Northridge 37

ODE System by RK4 II

- $k_{(3)y} = h[-(y+ k_{(2)y}/2) + z + k_{(2)z}/2] = 0.1[-(1 + -0.18/2) + (-1 + .18/2)] = -.182$
- $k_{(3)z} = h[(y+ k_{(2)y}/2) -(z + k_{(2)z}/2)] = 0.1[(1 + -0.18)/2 - (-1 + .18/2)] = .182$
- $k_{(4)y} = h[-(y+ k_{(3)y}) + z + k_{(3)z}] = 0.1[-(1 + -0.182) + (-1 + .182)] = -.1636$
- $k_{(4)z} = h[(y+ k_{(3)y}) -(z + k_{(3)z})] = 0.1[(1 + -0.182) - (-1 + .182)] = .1636$

Northridge 38

ODE System by RK4 III

- $y_{i+1} = y_i + (k_{(1)y} + 2k_{(2)y} + 2k_{(3)y} + k_{(4)y})/6 = 1 + [(-.2) + 2(-.18) + 2(-.182) + (-.1636)]/6 = .8187$ (here $i = 0$)
- $z_{i+1} = z_i + (k_{(1)z} + 2k_{(2)z} + 2k_{(3)z} + k_{(4)z})/6 = -1 + [(.2) + 2(.18) + 2(.182) + (.1636)]/6 = -.8187$
- Continue in this fashion until desired final x value is reached
  - Note all k_m computed before any k_{m+1}
- No x dependence for f in this example

Northridge 39