Lecture 21 Outline

Saturday, November 9, 2019 12:13 PM

Review Session

Tuesday, November 12

3:30pm - 5pm, 805 Furnas

Lecture Shift: Monday, November 18

11:30am - 12:50pm ??

Numerical Solution to ODEs

Explicit + Implicit Methods

Multistage Methods

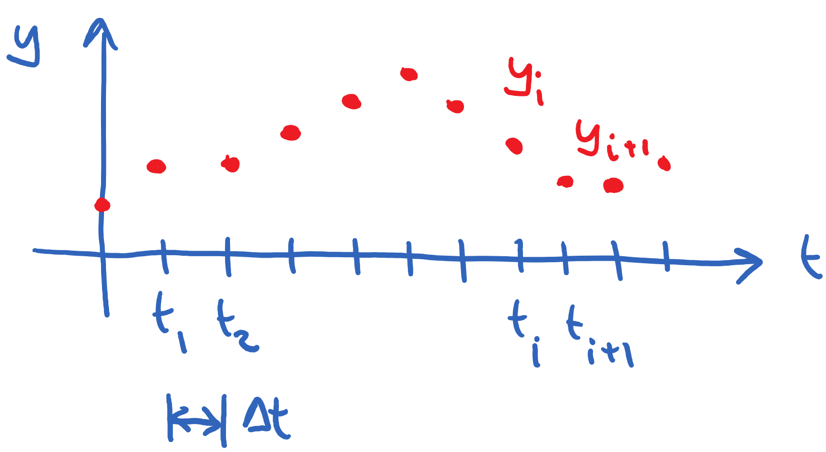
Numerical Solution to ODEs

Saturday, November 9, 2019 12:19 PM

Consider first-order ODEs of the form:

$$\frac{dy}{dt} = g(y) + f(t) \quad \text{with} \quad y(0) = y_0$$

Solve IVP numerically by using a finite difference approximation



Objective: Given y_i and Δt, find y_{i+1}

Approach: Replace derivatives in ODE with finite differences

Recall Taylor Series

$$y(t) = y(t_0) + \left.\frac{dy}{dt}\right|_{t_0}(t-t_0) + \left.\frac{d^2y}{dt^2}\right|_{t_0}\frac{(t-t_0)^2}{2}$$

$$+ \left.\frac{d^3y}{dt^3}\right|_{t_0}\frac{(t-t_0)^3}{6} + \dots + \left.\frac{d^ny}{dt^n}\right|_{t_0}\frac{(t-t_0)^n}{n!} + \dots$$

Let $t \to t_{i+1}$, $t_0 \to t_i$, $t - t_0 \to t_{i+1} - t_i = \Delta t$

Then

$$y_{i+1} = y_i + \left.\frac{dy}{dt}\right|_{t_i}\Delta t + \left.\frac{d^2y}{dt^2}\right|_{t_i}\frac{\Delta t^2}{2} + \left.\frac{d^3y}{dt^3}\right|_{t_i}\frac{\Delta t^3}{6} + \dots$$

$$\left.\frac{dy}{dt}\right|_{t_i} = \frac{1}{\Delta t}\left[y_{i+1} - y_i\right] + O(\Delta t)$$

Forward Difference

First order approximation

Substitute into ODE evaluated at time t_i

$$\frac{y_{i+1} - y_i}{\Delta t} - g(y_i) = f(t_i)$$

Forward Euler Method

Fully

Forward Euler Method

$$y_{i+1} = y_i + \Delta t\, g(y_i) + \Delta t\, f(t_i)$$ Fully explicit

y_i + all terms on RHS are known

y_{i+1} needs to be evaluated

Let $t \to t_i$, $t_0 \to t_{i+1}$ $t - t_0 \to t_i - t_{i+1} = -\Delta t$

Then

$$y_i = y_{i+1} - \left.\frac{dy}{dt}\right|_{t_{i+1}} \Delta t + \left.\frac{d^2y}{dt^2}\right|_{t_{i+1}} \frac{\Delta t^2}{2} - \left.\frac{d^3y}{dt^3}\right|_{t_{i+1}} \frac{\Delta t^3}{6} + \ldots$$

$$\left.\frac{dy}{dt}\right|_{t_{i+1}} = \frac{1}{\Delta t}\left[y_{i+1} - y_i\right] + O(\Delta t)$$

Backward Difference

First order approximation

Substitute into ODE evaluated at time t_{i+1}

$$\frac{y_{i+1} - y_i}{\Delta t} - g(y_{i+1}) = f(t_{i+1})$$

Backward Euler Method

$$y_{i+1} - \Delta t\, g(y_{i+1}) = y_i + f(t_{i+1})$$ Fully implicit

y_i + all terms on RHS are known

y_{i+1} needs to be obtained by solution

Both forward + backward Euler are $O(\Delta t)$ methods → Errors will be about the same

Why take the harder implicit approach?

Stability ⇒ Larger Δt

Example:

$$\frac{dy}{dt} + 2y = 0 \quad \text{with} \quad y(0) = 1$$

Analytical solution

$$y(t) = ae^{\lambda t} \rightarrow y' = a\lambda e^{\lambda t}$$

$$a\lambda e^{\lambda t} + 2ae^{\lambda t} = 0 \longrightarrow (\lambda + 2)ae^{\lambda t} = 0$$

$$\rightarrow \lambda = -2$$

$$\rightarrow y(t) = ae^{-2t}$$

$$y(0) = ae^{0} = a = 1 \longrightarrow y(t) = e^{-2t} \; //$$

Note $\lim_{t \to \infty} y(t) = 0$

Now, use Forward (Explicit) Euler Method to approximate $y(1)$, where exact result is

$$y(1) = 0.135335\ldots$$

Forward Euler w/ $\Delta t_1 = 0.25$

$$y_{n+1} = y_n - 2\Delta t_1 y_n \Rightarrow (1 - 2\Delta t_1) y_n$$

n	t	y_n
0	0	1

0	0	1
1	0.25	$(1-2(0.25))(1) = 0.5$
2	0.50	0.25
3	0.75	0.125
4	1.00	0.0625

Error is $e_1 = |0.135335 - 0.0625|$

$= 0.0728$

Now try $\Delta t_2 = \Delta t_1/2 = 0.125$

This gives $y(1) = 0.1001$

Error is $e_2 = |0.135335 - 0.1001|$

$= 0.035$

Then

$$\frac{e_2}{e_1} = \frac{0.035}{0.0728} \cong 0.48, \quad \frac{\Delta t_2}{\Delta t_1} = 0.5$$

$\Rightarrow$ Error is of $\mathcal{O}(\Delta t)$

as expected

Next, use Backward (Implicit) Euler Method

$$y_{n+1} = y_n - 2\Delta t\, y_{n+1} \rightarrow y_{n+1} = \frac{y_n}{1+2\Delta t}$$

For $\Delta t_1 = 0.25$, $y(1) = 0.1975$, $e_1 = 0.062$

$\Delta t_2 = 0.125$, $y(1) = 0.1677$, $e_2 = 0.0324$

$$\frac{e_2}{e_1} \cong 0.52, \quad \frac{\Delta t_2}{\Delta t_1} = 0.5 \quad \therefore \mathcal{O}(\Delta t)$$

In general, the errors for the implicit & explicit schemes of the same order will be comparable

Implicit is more expensive per time step, but has better stability characteristics

Same example, consider long time behavior, where $\lim_{t\to\infty} y(t) = 0$

Explicit Euler

Given $y_0 \rightarrow \quad y_1 = (1 - 2\Delta t)\, y_0$

$$y_2 = (1 - 2\Delta t)\, y_1$$
$$= (1 - 2\Delta t)^2\, y_0$$
$$\vdots$$
$$y_n = (1 - 2\Delta t)^n\, y_0$$

To have $\lim_{n \to \infty} y_n = 0$, then

$$|1 - 2\Delta t| < 1$$

For $\Delta t = 0.25$, $|1 - 2\Delta t| = 0.5 < 1$

$\therefore$ $\Delta t = 0.25$ is stable

For $\Delta t = 1.50$, $|1 - 2\Delta t| = 2 > 1$

$\therefore$ $\Delta t = 1.50$ is <u>not</u> stable

Implicit Euler

$$y_{n+1} = \frac{y_n}{\;} = \left(\frac{1}{1+2\Delta t}\right) y_n$$

$$y_{n+1} = \frac{y_n}{1+2\Delta t} = \left(\frac{1}{1+2\Delta t}\right) y_n$$

$$y_1 = \left(\frac{1}{1+2\Delta t}\right) y_0$$

$$y_2 = \left(\frac{1}{1+2\Delta t}\right) y_1 = \left(\frac{1}{1+2\Delta t}\right)^2 y_0$$

$$\vdots$$

$$y_n = (1+2\Delta t)^{-n} y_0$$

Need $|1+2\Delta t| > 1$, which is true for any $\Delta t > 0$ $\rightarrow$ Unconditionally stable

Summary:

Forward Euler is conditionally stable

Backward Euler is unconditionally stable

Formally, let $\frac{dy}{dt} = \lambda y$ w/ $y(0) = y_0$ + $\lambda < 0$

Forward Euler: $y_n (1+\lambda\Delta t)^n y_0$

$A = 1+\lambda\Delta t \rightarrow$ Amplification factor

Stable if $|A| = |1+\lambda\Delta t| < 1$

$$\Rightarrow \Delta t < \frac{2}{|\lambda|}$$

Backward Euler: $y_n = \dfrac{y_0}{(1-\lambda\Delta t)^n}$

$$A = \frac{1}{(1-\lambda\Delta t)}$$

$|A| < 1$ for any $\Delta t > 0$ if $\lambda < 0$

Notes:

1. Not all implicit schemes are unconditionally slable, but usually these a more stable than explicit schemes.

(2) Just because one can take a large time step does not mean that one should. Beyond stability, there is the issue of accuracy

Implicit scheme disadvantage: Cost

At best, a linear system needs to be solved (i.e., $y_{n+1} = y_n - 2\Delta t \, y_{n+1}$)

At worst, solve a non-linear equation

Example: $$\frac{dy}{dt} + \sinh(y) = 0$$

$$y_{n+1} + \Delta t \sinh(y_{n+1}) = y_n$$

Despite the cost, implicit schemes can be cheaper overall, especially for stiff problems with stringent Δt restrictions

One modification instead of full implicit schemes can be developed for non-linear systems $\rightarrow$ Semi-Implicit Scheme

Let $\frac{dy}{dt} + g(y) = 0$ with $g(y)$ is any function

If possible, split $g(y)$ into linear and non-linear parts

$$g(y) = L(y) + N(y)$$

$$\frac{dy}{dt} + L(y) + N(y) = 0$$

Then, let

$$\frac{y_{n+1} - y_n}{\Delta t} + L(y_{n+1}) + N(y_n) = 0$$

$$y_{n+1} + \Delta t\, L(y_{n+1}) = y_n - \Delta t\, N(y_n)$$

This will be less stable than fully implicit, but more stable than explicit. Also, usually much cheaper than fully implicit

Multistage Methods

Saturday, November 9, 2019 3:24 PM

Develop methods that take multiple "mini" steps between t_n and $t_n + \Delta t = t_{n+1}$ to achieve higher order schemes

Also called Predictor - Corrector Methods

Focus on explicit schemes in the Runge-Kutta family of methods

Start with scalar first order systems

$$\frac{dy}{dt} = f(t, y(t))$$

$\mathcal{O}(\Delta t)$ scheme

Let $k_1 = f(t_n, y_n)$ as the derivative $\frac{dy}{dt}$

Then $y_{n+1} = y_n + \Delta t \, k_1 \longrightarrow$ forward Euler

$\mathcal{O}(\Delta t^2)$ scheme

Let $k_1 = f(t_n, y_n)$

$$k_2 = f(t_n + c_1 \Delta t, y_n + a_1 \Delta t k_1)$$

for $c_1 \in [0,1], a_1 \in [0,1]$

Given $y_n \to k_1$ is approximation of $\frac{dy}{dt}$ at time t_n, while k_2 is the derivative at some time between t_n + t_{n+1}

Then

$$y_{n+1} = y_n + b_1 \Delta t k_1 + b_2 \Delta t k_2$$

$$= y_n + \Delta t (b_1 k_1 + b_2 k_2)$$

with a_1, b_1, b_2 + c_1 set to make scheme of order $O(\Delta t^2)$

How to do this? Use Taylor Series!

Find y_{n+1} as a series of y_n at time t_n

$$y_{n+1} = y_n + \Delta t\, y_n' + \frac{1}{2}\Delta t^2\, y_n''$$

$$y_n' = f(t_n, y_n)$$

$$y_n'' = \frac{df}{dt} = f_t + f_y\, y_n'$$

(1) $$y_{n+1} = y_n + \Delta t\, f(t_n, y_n) + \frac{1}{2}\Delta t^2 f_t(t_n, y_n) + \frac{1}{2}\Delta t^2 f_y(t_n, y_n)\, f(t_n, y_n)$$

Now expand k_2

$$k_2 = f(t_n + c_1 \Delta t,\, y_n + a_1 k_1 \Delta t) = f(t_n, y_n) + c_1 \Delta t\, f_t(t_n, y_n) + a_1 k_1 \Delta t\, f_y(t_n, y_n) + \text{H.O.T.}$$

higher order terms

Then

$$y_{n+1} = y_n + b_1 \Delta t\, k_1 + b_2 \Delta t\, k_2$$

$$(2)\quad y_{n+1} = y_n + b_1 \Delta t\, f(t_n, y_n) + b_2 \Delta t\, f(t_n, y_n)$$
$$+ b_2 c_1 \Delta t^2 f_t(t_n, y_n)$$
$$+ b_2 a_1 \Delta t^2 f(t_n, y_n) f_y(t_n, y_n)$$

Now compare (1) + (2)

$$\left.\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_1 &= \tfrac{1}{2} \\ b_2 a_1 &= \tfrac{1}{2} \end{aligned}\right\}$$

4 unknowns, but only 3 equations

$\rightarrow \infty$ solutions

$\rightarrow$ Infinite # of $\mathcal{O}(\Delta t^2)$ R-K schemes

Choose one unknown (typically b_2) + solve for others.

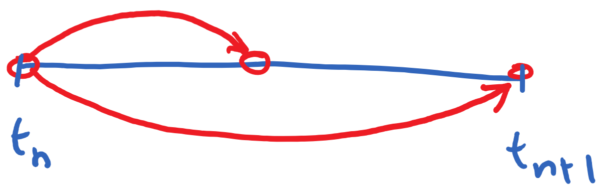
Most prominent choices:

a) Midpoint: $b_2 = 1 \rightarrow b_1 = 0,\ c_1 = a_1 = \frac{1}{2}$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{1}{2}\Delta t,\ y_n + \frac{1}{2}\Delta t\, k_1\right)$$

$$y_{n+1} = y_n + \Delta t\ k_2$$



b) Ralston's Method

$$b_2 = \frac{3}{4} \rightarrow b_1 = \frac{1}{4},\ c_1 = a_1 = \frac{2}{3}$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{2}{3}\Delta t,\ y_n + \frac{2}{3}\Delta t\ k_1\right)$$

$$y_{n+1} = y_n + \frac{1}{4}\Delta t\ k_1 + \frac{3}{4}\Delta t\ k_2$$

c) Heun's Method

$$b_2 = \frac{1}{2} \rightarrow b_1 = \frac{1}{2},\ c_1 = a_1 = 1$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \Delta t,\ y_n + \Delta t\ k_1)$$

$$y_{n+1} = y_n + \frac{1}{2}\Delta t\, k_1 + \frac{1}{2}\Delta t\, k_2$$
$$= y_n + \frac{1}{2}\Delta t\,(k_1 + k_2)$$

$\mathcal{O}(\Delta t^4)$ scheme

Using similar Taylor series analysis one can obtain 4th order schemes

The most well-known of these is simply called RK4

$$\frac{dy}{dt} = f(t, y) \quad \text{given } y_n \text{ \& } \Delta t$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{1}{2}\Delta t,\ y_n + \frac{1}{2}\Delta t\, k_1\right)$$

$$k_3 = f\left(t_n + \frac{1}{2}\Delta t,\ y_n + \frac{1}{2}\Delta t\, k_2\right)$$

$$k_4 = f\left(t_n + \Delta t,\ y_n + \Delta t\, k_3\right)$$

. .

$$y_{n+1} = y_n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

Generic RK Schemes

Saturday, November 9, 2019 4:42 PM

A compact way to write RK schemes is the Butcher Tables / Tableau

Let a generic RK scheme of order s be written

$$y_{n+1} = y_n + \Delta t \sum_{i=1}^{s} b_i k_i$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + c_2 \Delta t, y_n + \Delta t\, a_{21} k_1)$$

$$k_3 = f(t_n + c_3 \Delta t, y_n + \Delta t\, a_{31} k_1 + \Delta t\, a_{32} k_2)$$

$$\vdots$$

$$k_i = f\left(t_n + c_i \Delta t, y_n + \Delta t \sum_{j=1}^{i-1} a_{ij} k_j\right)$$

$$\vdots$$

$$k_s = f\left(t_n + c_s \Delta t, y_n + \Delta t \sum_{j=1}^{s-1} a_{sj} k_j\right)$$

which in Table form becomes

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \text{—} & a_{1s} \\
c_2 & a_{12} & a_{22} & & \\
\vdots & \vdots & \vdots & & \\
c_s & a_{1s} & & \text{——} & a_{ss} \\
\hline
 & b_1 & b_2 & \text{—} & b_s
\end{array}
$$

Examples:

Forward Euler

$$
\begin{array}{c|c}
0 & 0 \\
\hline
 & 1
\end{array}
$$

Midpoint

$$
\begin{array}{c|cc}
0 & 0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
 & 0 & 1
\end{array}
$$

Heun's Method

$$
\begin{array}{c|cc}
0 & 0 & 0 \\
1 & 1 & 0 \\
\hline
 & 1/2 & 1/2
\end{array}
$$

RK4

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Note: Each diagonal element in above tables is zero $\rightarrow$ All of these are explicit methods

Matlab + RK Schemes

Matlab has many built-in ODE solvers that require

$f(t, y)$

$[t_0, t_f]$ final

$y(t_0)$

Most used is ode45 $\rightarrow$ An $\mathcal{O}(\Delta t^5)$ scheme that uses an $\mathcal{O}(\Delta t^4)$ scheme to estimate error and then vary Δt to achieve required accuracy

Other Matlab functions: ode23, ode113

For stiff problems: ode23s, ode23t