Today

Adaptive Runga Kutta

Stiff ODEs

Implicit Methods

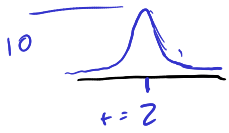
Announcements

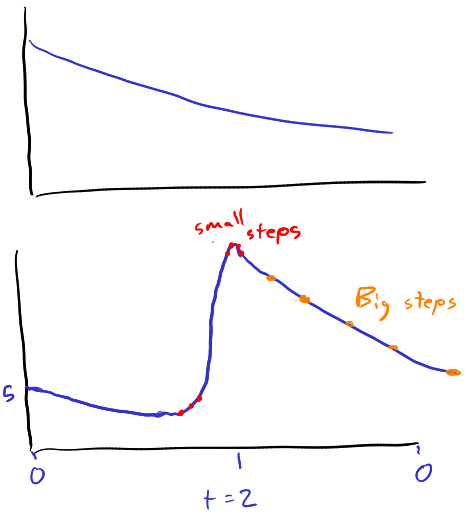
Lab 2, HW 4 due tomorrow

$\dot{x} = -0.6x \Rightarrow x = x_0 e^{-0.6t}$

$\dot{x} = 10e^{-(t-2)^2/[2(0.075)^2]} - 0.6x$

takes value of 10 at $t=2$





Option 1

Step Halving

Option 2: Adaptive Runge Kutta

```
1: function EULER_ADAPTIVE(f, [t_0, t_f], x_0, E_s)
2:     t ← [t_0],   x ← [x_0],   h ← (t_f − t_0)/100
3:     while t_end < t_f do
4:         h ← min(h, t_f − t_end)
5:         x_full ← x_end + h f(t_end, x_end)                ▷ or RK4
6:         x_mid ← x_end + (h/2) f(t_end, x_end)
7:         x_half ← x_mid + (h/2) f(t_end + h/2, x_mid)
8:         E_a ← |x_half − x_full|
9:         if E_a ≤ E_s then
10:            append t_end + h to t
11:            append x_half to x
12:        end if
13:        h ← 0.9 h sqrt(E_s/E_a)                           ▷ b/c E_a is O(h^2)
14:    end while
15:    return t, x
16: end function
```

full Euler's

2 Euler steps with h/2

```
1: function BS23_ADAPTIVE(f, [t_0, t_f], x_0, E_s)
2:     t ← [t_0],   x ← [x_0],   h ← (t_f − t_0)/100
3:     k_1 ← f(t_0, x_0)
4:     while t_end < t_f do
5:         h ← min(h, t_f − t_end)
6:         k_2 ← f(t_end + h/2, x_end + (h/2) k_1)
7:         k_3 ← f(t_end + 3h/4, x_end + (3h/4) k_2)
8:         x_3 ← x_end + h (2/9 k_1 + 1/3 k_2 + 4/9 k_3)     ▷ 3rd order
9:         k_4 ← f(t_end + h, x_3)
10:        x_2 ← x_end + h (7/24 k_1 + 1/4 k_2 + 1/3 k_3 + 1/8 k_4)   ▷ 2nd order
11:        E_a ← |x_3 − x_2|
12:        if E_a ≤ E_s then
13:            append t_end + h to t
14:            append x_3 to x
15:            k_1 ← k_4                                     ▷ slope at the new x_end
16:        end if
17:        h ← 0.9 h (E_s/E_a)^{1/3}                         ▷ b/c E_a is O(h^3)
18:    end while
19:    return t, x
20: end function
```

3 function evaluations per step

k_i for next iter

Local: $O(h^3)$

Start with this IRL

ode23

ode45 does the same thing with 4th/5th order

Stiff ODEs

A "stiff" system involves rapidly-changing and slowly-changing components



Van-der-pol Oscillator

$\ddot{x} - \mu(1-x^2)\dot{x} + x = 0$

first create system $\dot{\vec{x}} = f(t, \vec{x})$

$v \equiv \dot{x}$

$\dot{\vec{x}} = f(t, \vec{x})$

$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \mu(1-x^2)v - x \\ v \end{bmatrix}$$

system of 1st order ODEs

$\mu = 1$: not stiff

RK4 works

BS23 works

$\mu = 1000$: stiff

RK4 diverges

BS23 takes a lot of steps

Implicit methods

help stability not accuracy

Implicit Euler Update

Explicit $x_{i+1} \leftarrow x_i + f(t_i, x_i)\,h$

Implicit $x_{i+1} \leftarrow x_i + f(t_i+h, x_{i+1})\,h$

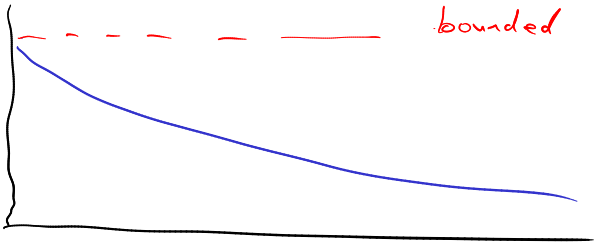
How do we find x_{i+1}?

Solve (possibly nonlinear equation) $x_{i+1} - x_i - f(t_{i+1}, x_{i+1})\,h = 0$

Use any nonlinear solver

Example

$\dot{x} = -ax \qquad a > 0$



unstable if errors grow exponentially for a bounded solution

Explicit Euler was conditionally stable

Implicit: $x_{i+1} = x_i - a x_{i+1} h$

$x_{i+1}(1+ah) = x_i$

$x_{i+1} = \frac{x_i}{1+ah}$ ← shrinks regardless of h

Implicit Euler is unconditionally stable

Implicit Euler on VDP equations

$$v_{i+1} = v_i + \mu(1 - x_{i+1}^2)\,v_{i+1} - x_{i+1}$$
$$x_{i+1} = x_i + v_{i+1}$$

unknowns

nonlinear system of equations

Newton-Raphson for systems

IRL if you have a stiff system consider this

ode23s in Matlab

↑ stiff