Lecture 13 — Differential Equations I

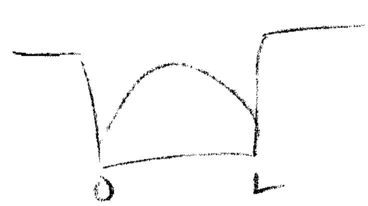
(Finite Difference Methods)

- Mainly ODE's -

Introduction:

Types of Problems:

	(easier)	(more difficult)
(A)	**ODE**	**PDE**
	ex. $M\frac{d^2\phi}{dt^2} = -K\phi(t)$ (simple harmonic oscillator)	ex. $\frac{\partial^2}{\partial x^2}\phi - \frac{1}{v^2}\frac{\partial^2\phi}{\partial t^2} = 0$ (wave equation)
(B)	**Linear**	**Nonlinear**
	ex. SHO, wave equations	ex. $\ddot{\phi} + w^2 \sin\phi = 0$ (pendulum), $w = \sqrt{g/k}$
(C)	**Initial Value**	**Boundary Value**
	$\phi(0)$ $\dot{\phi}(0)$	$\phi(0)$ $\phi(L)$



☆ **easier** — **More Difficult**

This lecture emphasis:

- ODE (linear/nonlinear)

$\swarrow$ $\searrow$

Initial Value

- Euler's (conceptual background)
- Runge-Kutta
- Midpoint & Richardson's Extrapolation
- others: predictor-corrector, leapfrog
- Stiffness problem
  - implicit methods

Boundary Value

- Shooting Methods
- Relaxation Methods

$\downarrow$

PDE (Next lecture)

☆☆ all of these rely on approximating

$$\dot{\phi}(t)\Big|_{x_k} \simeq \frac{1}{h}\left(\phi_{k+1} - \phi_k\right)$$

$$\ddot{\phi}(t)\Big|_{x_k} \simeq \frac{1}{h^2}\left(\phi_{k+1} - 2\phi_k + \phi_{k-1}\right)$$

by Finite Differences

General Form of ODE

- All ODE in any order can be written into a set of 1st order ODE!

$$\left\{ \frac{d^2y}{dt^2} + g(t)\frac{dy}{dt} = r(t) \right. \qquad \text{(2nd order)}$$

let $z(t) = \dfrac{dy}{dt}$

then $\dfrac{dz}{dt} = \dfrac{d^2y}{dt^2}$

$$\Rightarrow \quad \begin{cases} \dfrac{dz}{dt} = r(t) - g(t)\,z(t) & \text{(1st-order)} \\[2ex] \dfrac{dy}{dt} = z(t) \end{cases}$$

(☆ trade off → increase of dim by 1)

So, in general, we can always write ODE as:

$$\boxed{\dot{\underset{\sim}{y}}(t) = \underset{\sim}{F}(\underset{\sim}{y}, t)}$$

$\underset{\sim}{F}(\underset{\sim}{x}, t)$ could be linear or nonlinear!

Initial Value Problems

— Basic Idea behind Numerical ODE solver

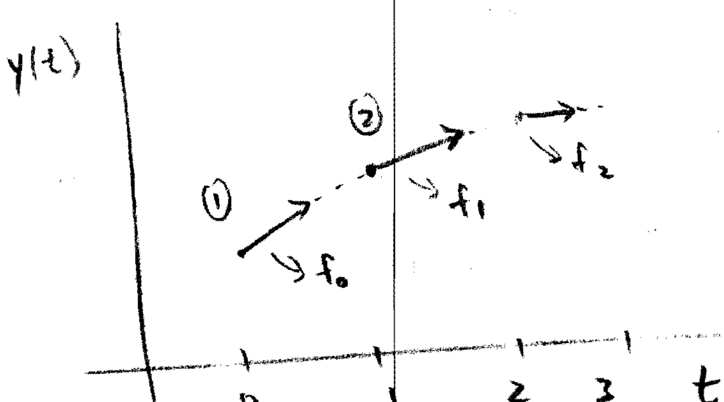
→ Euler's Method:

$$\dot{y}(t) = f(y, t)$$

Finite diff. $\Rightarrow \dfrac{y_{n+1} - y_n}{h} \hat{=} f(y_n, t_n)$; $y_n = y(t_n)$, $h = t_{n+1} - t_n$

Euler scheme → $y_{n+1} = y_n + h f(y_n, t_n)$ (*)

☆ start with I.C. y_0, use (*) to evolve y_n!



Note only derivative at t_n is used to get to Y_{n+1}! (assymetric)

☆ It is simple to see that:

By Taylor's expansion:

$$y_{n+1} = y_n + h f(y_n) + h^2 \frac{f'(y_n)}{2!} + \ldots$$

$$\uparrow$$

$$E \sim O(h^2)$$

☆ So, Euler is a 1st order Method!

Applet is open in a separate window.

This java applet demonstrates properties of vector fields. You may select one of many vector fields from the **Setup** menu in the upper right.

The applet shows the potential surface of the vector field, with particles following the field vectors. You may click and drag with the mouse to rotate the view. Also some of the field selections have parameters which may be adjusted.

There is also a 3-D version of this applet (a version with 3-D fields, that is). This version only does 2-D fields, but unlike the 3-D version it can display the potential surface, curl, and divergence, and can also demonstrate Green's theorem and the divergence theorem.

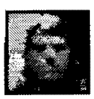
Full Directions.

The source.

More applets.

Zip archive of this applet.

Version 1.3, posted 2/22/05

java@falstad.com

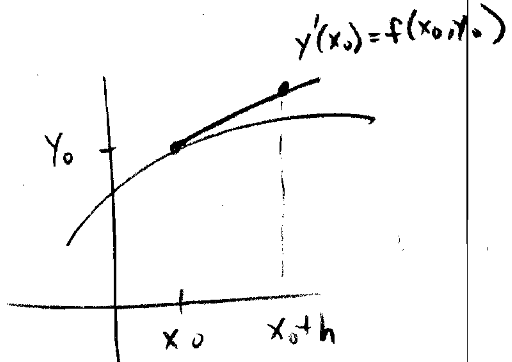
Show applet of vector field

Modified & Improved Euler's Methods

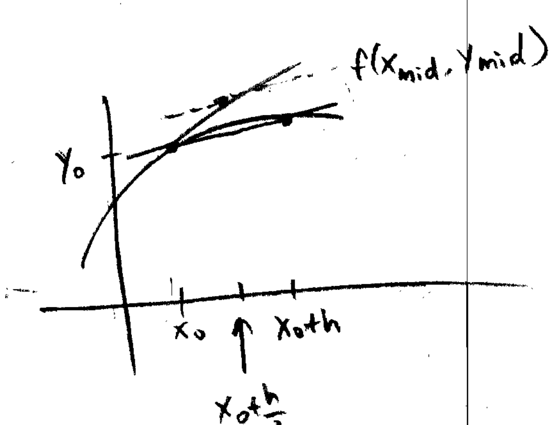
→ Euler is asymmetric & first order

☆ Can try to make it more <u>symmetric</u> and we will show that errors can be cancelled to <u>higher order</u> $\sim h^2$!

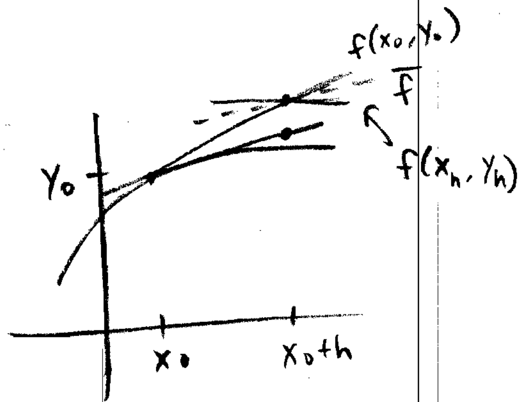
<u>Graphical</u>



<u>regular Euler</u>
- move forward with derivative $f(x_0, Y_0)$ at initial pt.



<u>modified Euler</u>
- move forward with approx. derivative $f(X_{mid}, Y_{mid})$ at $X_{mid} = X_0 + \frac{h}{2}$.



improved Euler

- move forward with "average" derivative $\frac{f_0+f_h}{2}$ at x_0 & x_0+h.

- let see how you do this?

Modified : - $x_{mid} = x_0 + \frac{h}{2}$

- $y_{mid} = y_0 + \frac{h}{2} f(x_0,y_0)$
(estimate mid pt with regular Euler)

- then $f(x_{mid}, y_{mid})$ (approx. derivative at midpt)

can be evaluated.

lastly - $y(x_0+h) = y_0 + h f(x_{mid}, y_{mid})$

Improved : – $x_h = x_0 + h$

– $\tilde{y}_h = y_0 + hf(x_0, y_0)$

(approx $\tilde{y}_h$ by regular Euler)

– then take average of derivative at x_0 (initial pt) & x_h (end pt).

$$\bar{f} = \frac{f(x_0,y_0) + f(x_h,\tilde{y}_h)}{2}$$

lastly – $y(x_0+h) = y_0 + h\left(\frac{f(x_0+y_0) + f(x_0+h, y_0+hf_0)}{2}\right)$

Modified & Improved Euler are 2nd order

Write them in general form:

$$y(x_0+h) = y_0 + h[\alpha f(x_0,y_0) + \beta f(x_0+\gamma h, y_0+\delta h f_0)]$$

Note that : Modified — $\alpha = 0$, $\gamma = \delta = \frac{1}{2}$, $\beta = 1$

Improved — $\alpha = \frac{1}{2}$, $\gamma = \delta = 1$, $\beta = \frac{1}{2}$

– other choices are possible –

Now, let look at Taylor's expansion of

$$f(x_0+\gamma h,\ y_0+\delta h f_0)$$

$$= f(x_0,y_0) + \frac{\partial f(x_0,y_0)}{\partial x}(\gamma h) + \frac{\partial f(x_0,y_0)}{\partial y}(\delta h f_0) + O(h^2).$$

Now, put this into our general Euler Eq.

$$y(x_0+h) = y_0 + h\alpha f_0 + h\beta\left[f_0 + \gamma h\frac{\partial f_0}{\partial x} + \delta h f_0 \frac{\partial f_0}{\partial y} + O(h^2)\right]$$

$$= y_0 + h(\alpha+\beta) f_0 + h^2\beta\left[\gamma\frac{\partial f}{\partial x}(x_0,y_0) + \delta f_0\frac{\partial f_0}{\partial y}\right] + O(h^3)$$

Compare with Taylor expansion of $y(x_0+h)$:

$$y(x_0+h) = y_0 + \frac{dy}{dx}(x_0,y_0)\,h + \frac{1}{2}\frac{d}{dx}\left(\frac{dy}{dx}\right)h^2 + O(h^3)$$

$$\downarrow$$

$$y_0 + f_0 h + \frac{1}{2}h^2\left[\frac{\partial f}{\partial x}(x_0,y_0) + f_0\frac{\partial f}{\partial y}(x_0,y_0)\right] + O(h^3)$$

Note: $\frac{d}{dx}\frac{dy}{dx} = \frac{d}{dx}(f(x,y)) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$

$$= \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f$$

→ everything will be consistent with Taylor to $O(h^3)$

$$\begin{cases} \alpha+\beta = 1 \\ \beta\gamma = \frac{1}{2} \\ \beta\delta = \frac{1}{2} \end{cases}$$

check with Modified ✓ $\alpha=0$, $\beta=1$, $\gamma=\delta=\frac{1}{2}$

& Improved ✓ $\alpha=\beta=\frac{1}{2}$, $\gamma=\delta=1$

⇒ Thus, Modified & Improved Euler Methods are 2nd order in h!

check:

Taylor's expanding y_{n+1} around y_n:

$$y_{n+1} = y_n + h f_n + \frac{h^2}{2} f_n' + O(h^3) \quad (*)$$

By finite difference,

$$\frac{(f_{n+1/2} - f_n)}{\frac{1}{2}h} \hat{=} f_n'$$

Substitute this into $(*)$,

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

$$\boxed{y_{n+1} = y_n + h \underbrace{f_{n+1/2}}_{k_2} + O(h^3)}$$

So, this "trick" step make method become 2nd order!

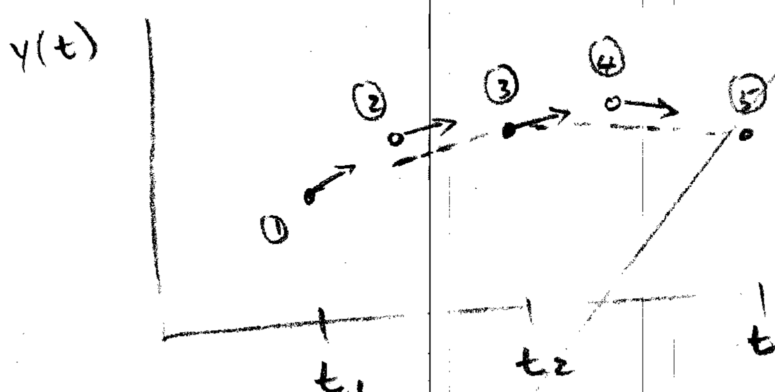
"RK - 2nd order" or "midpt" method

Will come back to this later!

Runge-Kutta Method (higher order method)

★ Similar to Numerical Integrators (or Mod/Imp Eulers) one can try to cancel out higher ordered terms in Taylor's expansion by including other fn evaluation pts in the integration step. (also symmetrize)

2nd-order RK:



- use Euler to take a "trial" step to midpt
- evaluate $\dot{y}(t_1 + \frac{1}{2}h)$ at midpt
- use this midvalue to compute the "real" step across the whole interval.

(slope at midpt is better than $\dot{y}(t_1)$ to get to t_2!)

★ $$\left\{ \begin{aligned} k_1 &= h f(t_n, y_n) \\ k_2 &= h f(t_n + \tfrac{1}{2}h, \; y_n + \tfrac{1}{2}k_1) \end{aligned} \right\} \quad y_{n+1} = y_n + k_2 + O(h^3)$$

Runge Kutta

4th order RK :

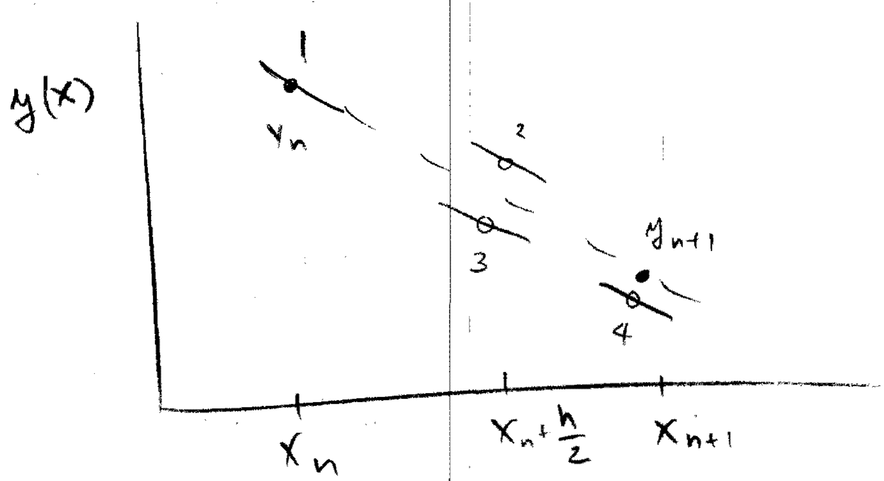
- Similarly, by evaluating f at different pts & by combining them in specific ways, one can in principle cancel higher order terms order by order!

- One particular choice is : 4th order RK

$$\begin{cases} k_1 = h f(y_n, x_n) \\ k_2 = h f(y_n + \frac{k_1}{2}, x_n + \frac{h}{2}) \\ k_3 = h f(y_n + \frac{k_2}{2}, x_n + \frac{h}{2}) \\ k_4 = h f(y_n + k_3, x_n + h) \\ y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{cases}$$

f' at mid point ~ modified Euler

~ weighted avg of y' ~ improved Euler

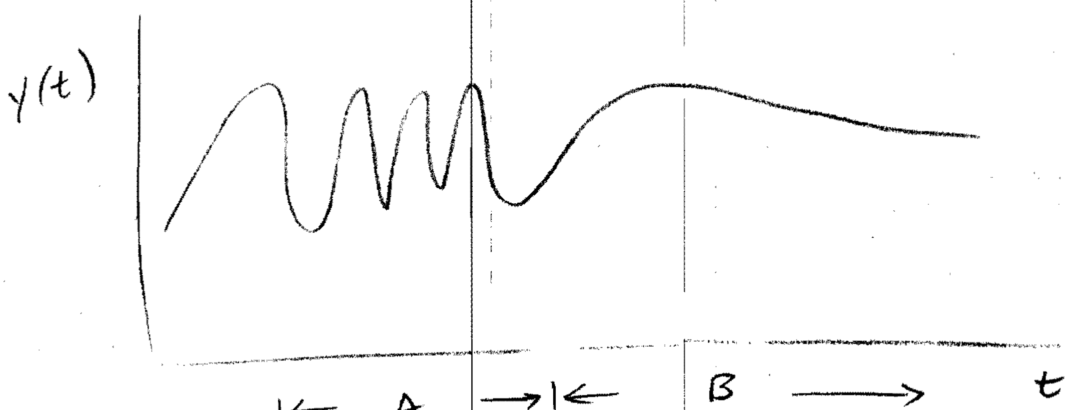


☆ Check that 4th RK is $O(h^5)$!

☆ RK4 → - Scientific Workhorse
- Method of choice for 1st tries
- But not necessary the most accurate!

Adaptive Stepsize for RK4

(Improvement on Efficiency)



y(t) varies quickly → need small steps

y(t) varies slowly → can use large steps.

☆ Optimize between accuracy & speed by varies the step sizes!

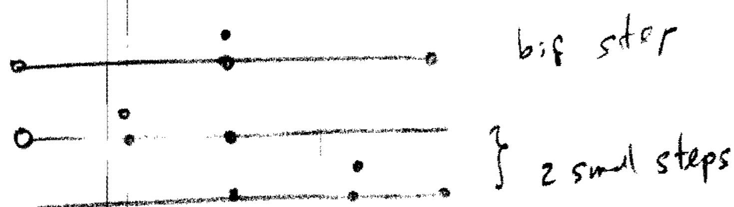
<u>Question</u>: How to adaptive vary the step size?

$\Rightarrow$ Need to be able to estimate the truncation error with a given step size!

☆ <u>Soln</u>: - Start with a trial value of h

- Calculate $y(t)$ in two different ways
    - 2 steps of h
    - 1 step of $2h$.

step doubling w/ correction



total # of fn evaluation = 11 (1st pts are the same)

- just use 2 small steps = 8

overhead cost in add. computation = $\frac{11}{8} \simeq 1.375$.

But, it gives you a handle on the truncation error:

(10)

true value ↘ numerical est. ↓

one step : $y(t+2h) = y_1 + (2h)^5 \frac{y^{(5)}}{5!} + O(h^6)$

two sm. steps : $y(t+2h) = y_2 + 2h^5 \frac{y^{(5)}}{5!} + O(h^6)$

(Note: RK4 is a $O(h^4)$ process)

(Calculated)

☆ $\Delta = y_2 - y_1$ indicator of truncation error

$\sim O(h^5)$

- We want to adjust h so that this is not too small (efficiency) AND not too large (accuracy)

- Now, let say ε_{max} - maximum toleratable truncation error
- Within the current interval $[t_k, t_{k+1}]$, we estimated $\Delta(t_k, h_k) = y_2 - y_1$ to be of $O(h^5)$
- Optimally, we want to set h_{k+1} for the next interval $[t_{k+1}, t_{k+2}]$ such that

$$\Delta(t_{k+1}, h_{k+1}) = \varepsilon_{max}$$

Since $\Delta \sim h^5$,

$$\left|\frac{\Delta_{k+1}}{\Delta_k}\right| = \left|\frac{\varepsilon_{max}}{\Delta(t_k, h_k)}\right| \approx \left(\frac{h_{k+1}}{h_k}\right)^5$$

☆☆ (We can calculate Δ_k & we know how it scales with h)

$$\Rightarrow \quad h_{k+1} \simeq h_k \left|\frac{\varepsilon_{max}}{\Delta(t_k, h_k)}\right|^{1/5}$$

This means that:

☆ $\begin{cases} \text{If } \Delta_k \geq \varepsilon_{max}, & h_{k+1} \text{ will be scaled back smaller} \\ \text{And, if } \Delta_k < \varepsilon_{max}, & h_{k+1} \text{ will be expanded bigger.} \end{cases}$

Note: ε_{max} should be stated in terms of fractional error

$$\varepsilon_{max} \sim \varepsilon\, y$$

ε: 10^{-6}, 10^{-16}, ... (single, double)

y → e.g. y_{max} for oscillatory functions. ↑y_{max}

☆ More strict criterion: ☆ (might be needed if ε accumulated with steps)

for small h, $\varepsilon_{max} \sim \epsilon h$ ← fractional accuracy in increments of y instead of y itself!

Runge-Kutta-Fehlberg

— Alternative to doubling/halfing method for
Adaptive step size control.

☆ (Advantage: 6 fn calls instead of 11)

☆ With six fn calls, Fehlberg found a way to construct

— a 5th order RK Y

AND — a 4th order RK. $\tilde{Y}$

— So, just like doubling method, we can get a handle on truncation error with these two estimates.

$$\Delta_K = Y_K - \tilde{Y}_K \sim \mathcal{O}(h^5)$$

↳ determined by the larger 4th order RK.

Again, $h_{K+1} = h_K \left| \dfrac{\varepsilon_{max}}{\Delta_K} \right|^{1/5}$

(11'')

— In the case when the total error E accumulated with the # of steps

$\Rightarrow$ reducing step size h (more steps), we need to require ε_{max} to be smaller accordingly!

$$\Rightarrow \quad \varepsilon_{max} \to \varepsilon_{max} h$$

— With this scaling, we need to adjust

$$\frac{\varepsilon_{max}}{\Delta_K} \to \frac{\varepsilon_{max} h_{K+1}}{\Delta_K} \sim \left(\frac{h_{K+1}}{h_K}\right)^5 \Rightarrow \frac{h_K \varepsilon_{max}}{\Delta_K} \sim \left(\frac{h_{K+1}}{h_K}\right)^4$$

$$\Rightarrow \quad h_{K+1} \sim h_K \left(\frac{h_K \varepsilon_{max}}{\Delta_K}\right)^{1/4}$$

Press suggestion:

(Combination of the two exponents)

$$h_{K+1} = \begin{cases} s\, h_K \left(\dfrac{\varepsilon_{max}}{\Delta_K}\right)^{1/5} & (\varepsilon_{max} \geq \Delta_K) \text{ expand} \\ s\, h_K \left(\dfrac{\varepsilon h_K}{\Delta_K}\right)^{1/4} & (\varepsilon h_K < \Delta_K) \text{ reduce} \end{cases}$$

$\Rightarrow$ the scaling should be $1/4$ instead of $1/5$ when $h \downarrow$!

A workable balance :

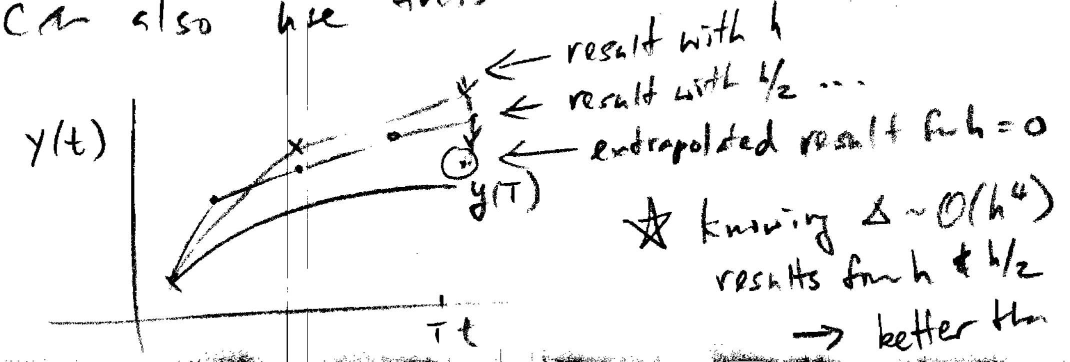
$$h_{k+1} = \begin{cases} S h_k \left| \dfrac{\varepsilon_{max}}{\Delta_k} \right|^{1/5} & \varepsilon_{max} \geq \Delta_k \quad \text{expand} \\ S h_k \left| \dfrac{\varepsilon_{max}}{\Delta_k} \right|^{1/4} & \varepsilon_{max} < \Delta_k \quad \text{reduce} \end{cases}$$

where S is a safety factor $\lesssim 1$.

Richardson's Extrapolation to $h \to 0$ (Bulirsch-Stoer Method)

Recall $\to$ Romberg integration : we use information of truncation error due to step size h to extrapolate result to $h \to 0$!

– We can also use this idea for ODE solver :



☆ knowing $\Delta \sim O(h^4)$ results for h & $h/2$ $\to$ better than $h/2$ alone.

- In principle, we can use any methods described before as the kernel for this scheme.

Recall <u>Mid-pt</u> Method:

$$y_{n+1} = y_n + hf(t_n + \tfrac{1}{2}h,\ y_n + \tfrac{1}{2}k_1)$$
$$k_1 = hf(t_n + \tfrac{1}{2}h,\ y_n)$$

For simplicity, assume we used fixed step size h.

Give $\quad y_n(T, h) \leftarrow$ result for step size h.

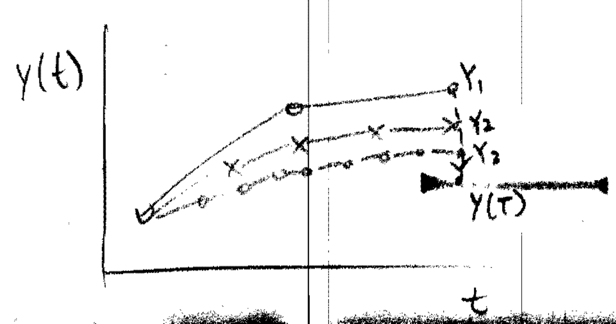
(Modified Mid-pt)

Recall in extrapolation (Neville's algorithm)

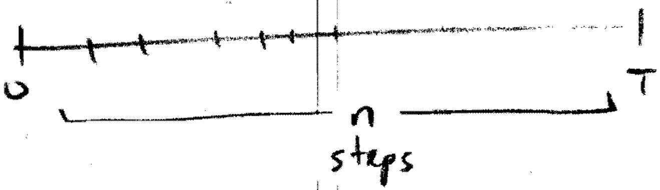
starting values

desired extrapolated result.

h	Y_1			
		Y_{12}		
$h/2$	Y_2		Y_{123}	
		Y_{23}		Y_{1234}
$h/4$	Y_3		Y_{234}	
		Y_{34}		
$h/8$	Y_4			



Modified Midpt



$h = T/n$

intermediate approx.
$$\begin{cases} z_0 = y(0) \\ z_1 = z_0 + h f(0, z_0) \\ \vdots \\ z_{m+1} = z_{m-1} + 2h f(x+mh, z_m) \quad m = 1, 2, \dots, n-1 \end{cases}$$

Same as "mid-pt" except at z_1 & z_n!

Final approx. $\leftarrow$ $Y_n(T, h) = \frac{1}{2}\left(z_n + z_{n-1} + h f(H, z_n)\right)$

$\rightarrow \mathcal{O}(h^2)$ in particular,

$y_n - y(T) = \sum \alpha_i h^{2i}$ even powers only!

☆ Simplest extrapolation:

assume we have y_n & $y_{n/2}$ (both $\mathcal{O}(h^2)$)

$\Rightarrow y(T) \approx \dfrac{4y_n - y_{n/2}}{3}$ this estimate is

4th order accurate!

Better Extrapolation

Rational Functions:

$$Y_n = \frac{P_\mu}{Q_\nu} = \frac{a_0 + \dots + a_\mu x^\mu}{b_0 + \dots + b_\nu x^\nu}$$

Using the trick with small differences:

elements in triangle

$$\Delta_{m,k} \equiv Y_{k\dots(k+m)} - Y_{k\dots(k+m-1)}$$

$$\Theta_{m,k} \equiv Y_{k\dots(k+m)} - Y_{(k+1)\dots(k+m)}$$

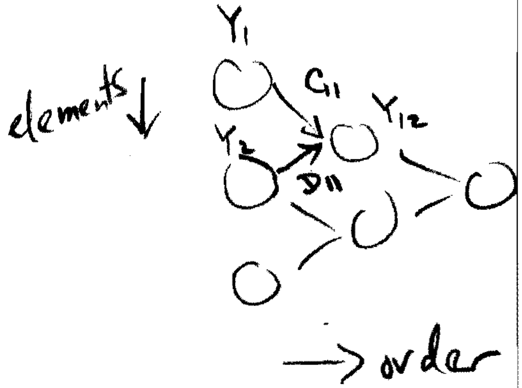
$\rightarrow$ recurrence relation:

$$\begin{cases} \Theta_{m+1,k} = \dfrac{\left(\Delta_{m(k+1)} - \Theta_{mk}\right)\Delta_{m(k+1)}}{\dfrac{h_k}{h_{m+k+1}}\Theta_{mk} - \Delta_{m(k+1)}} \\[2em] \Delta_{m+1,k} = \dfrac{\dfrac{h_k}{h_{m+k+1}}\Theta_{mk}\left(\Delta_{m(k+1)} - \Theta_{mk}\right)}{\dfrac{h_k}{h_{m+k+1}}\Theta_{mk} - \Delta_{m(k+1)}} \end{cases}$$

$$h_k = T/n_k$$

I.C. $\Theta_{0k} = \Delta_{0k} = Y(h_k)$

$n_k = 2, 4, 6, 8, 10, 12, 14 \dots$



$m=1$
$i=1$

$$C_{11} = P_{12} - P_1$$
$$D_{11} = P_{12} - P_2$$

At each level m,
Δ, θ are the corrections that give extrapolation one order higher

The final answer $Y_{1 \ldots N}$ is equal to the sume of any element Y_i + a set of Δ and/or Θ to get you thru the tree!

Stiffness

☆ Problem with very different time scale in ODE

- occurs wherever there are more than one 1st-order differential eq -

example:

$$u' = 998u + 1998v$$
$$v' = -999u - 1999v$$

with I.C.: $u(0) = 1$; $v(0) = 0$

By means of the transformation:

$$u = 2y - z \qquad v = -y + z$$

we find:

$$u = 2e^{-x} - e^{-1000x}$$
$$v = -e^{-x} + e^{-1000x}$$

two very different time scales!

In real soln: the e^{-1000x} term will die away quickly! → Not important in soln for x large.

☆ However, numerically it is unstable unless $h \ll 1/1000$!

→ To see this, let consider a simpler case

$$y' = -cy \qquad c > 0 \qquad y_0 = 1 \quad \rightarrow \quad y(t) = e^{-ct}$$

Euler scheme (other in principle similar)

$$Y_{n+1} = Y_n + hY_n' = (1 - ch)Y_n$$

This is <u>explicit</u> $\Rightarrow$ Y_{n+1} is a fn of Y_n.

☆☆ Note: Y_{n+1} will only be stable (not blow up)

if $\left| 1 - ch \right| < 1$ ☆ ($h < 2/c$ to be stable)

$\begin{cases} 1-ch < 1 \\ \underline{h > 0} \end{cases}$ and $\Rightarrow 0 < h < 2/c$! $(2/1000)$

$\begin{cases} 1-ch > -1 \\ -ch > -2 \rightarrow \underline{h < 2/c} \end{cases}$

A simple cure is to use <u>implicit</u> differencing:

backward Euler scheme:

$$Y_{n+1} = Y_n + h \underline{\underline{Y'_{n+1}}} \qquad Y'_{n+1} = -ch Y_{n+1}$$

$$Y_{n+1} = \frac{Y_n}{1+ch}$$

In this case, $\left| \frac{1}{1+ch} \right| < 1$ for all $h > 0$

And, Y_{n+1} is stable, i.e. $Y_{n+1} \rightarrow 0$ as $n \rightarrow \infty$

as required by the actual soln:

$$Y = e^{-ct} \rightarrow 0 \quad \text{as } t \rightarrow \infty.$$

☆ Stability for explicit method is true for linear system and it gives better performance in general

For general Nonlinear Equations:

$$\underset{\sim}{y}' = \underset{\sim}{f}(\underset{\sim}{y})$$

$$\rightarrow \quad \underset{\sim}{y}_{n+1} = \underset{\sim}{y}_n + h\,\underset{\sim}{f}(\underset{\sim}{y}_{n+1})$$

Implicit

→ One can solve the implicit equation directly

or Linearize:

$$\underset{\sim}{y}_{n+1} = \underset{\sim}{y}_n + h\left[f(\underset{\sim}{y}_n) + \nabla \underset{\sim}{f}\big|_{y_n} \cdot (\underline{y}_{n+1} - y_n)\right]$$

$$[\underset{\sim}{1} - h\nabla \underset{\sim}{f}]\,(y_{n+1} - y_n) = h\,f(y_n)$$

$$\Rightarrow \quad \underset{\sim}{y}_{n+1} = \underset{\sim}{y}_n + h\left[\underset{\sim}{1} - h\,\nabla \underset{\sim}{f}\right]^{-1} \cdot \underset{\sim}{f}(\underset{\sim}{y}_n)$$

Cost of implicit → need to evaluate inverse!

higher order implicit methods:

Generalization of RK4 → Rosenbrock

of BS → Bader and Deuflhard

"

⋮

[Press]

The Leapfrog Integrator

Let consider a 2nd order ODE (e.g. Newton's Equation) where the acceleration per unit mass of a particle is given by $f(x)$, i.e.

$$\frac{d^2x}{dt^2} = f(x) \longrightarrow \begin{cases} \dfrac{dx}{dt} = v \\ \dfrac{dv}{dt} = f(x) \end{cases}$$

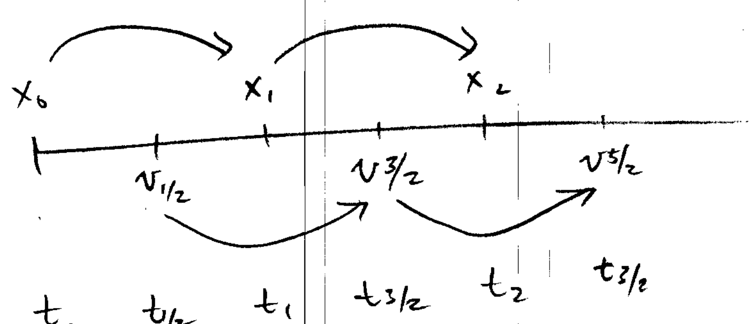
The Leapfrog integrator for this system can be defined as:

$$(*) \quad \begin{cases} x_{i+1} = x_i + v_{i+1/2}\, dt \\ v_{i+3/2} = v_{i+1/2} + f(x_{i+1})\, dt \end{cases}$$

a two-steps method

where $v(t) = \frac{dx}{dt}(t)$

$v_{i+1/2} = v(t + \frac{1}{2}dt)$



<u>Note</u> :

- symmetric with respect to the ways that x & v are advanced in time.

$$x_i \longrightarrow x_{i+1} \text{ with } v_{i+1/2} \text{ (at mid point)}$$

$$v_{i+1/2} \longrightarrow v_{i+3/2} \text{ with } x_{i+1} \text{ (at midpoint)}$$

- x & v advance in a staggered / leapfrog manner.

  - need "self-starting" :

    - to get x_i, one need $v_{1/2}$ from Euler, RK4, etc.

- $x(t)$ is 2nd order in accuracy :

$$x(t+dt) = x(t) + [\overbrace{v(t) + \tfrac{1}{2}dt\, f(t) + O(dt^2)}^{v(t+\frac{1}{2}dt)}]\,dt$$

$$x(t+dt) = x(t) + \underbrace{v(t)}_{\frac{dx}{dt}}\,dt + \tfrac{1}{2}\underbrace{f(t)}_{\frac{d^2x}{dt^2}}\,dt^2 + O(dt^3)$$

Leapfrog is <u>time-reversible</u>!

– One can invert the iteration (*) explicitly.

$$\begin{cases} v_{i+1/2} = v_{i+3/2} + f(x_{i+1})(-dt) \\ x_i = x_{i+1} + v_{i+1/2}(-dt) \end{cases}$$

→ these are precisely the steps (the same functional evaluations) that we took to advance the system in the first place.

→ $+dt$ ~~&~~ $-dt$, we get back to exactly the same starting point!

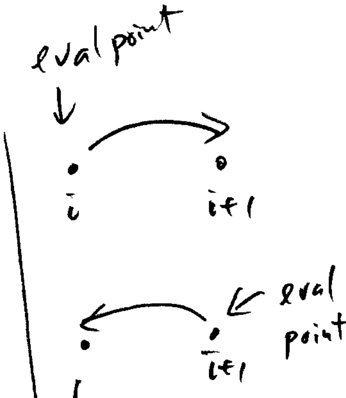
<u>Note</u>: Euler or RK4 – the derivatives are evaluated not symmetrically at different time.

e.g. Euler:

forward: $\begin{cases} x_{i+1} = x_i + \underline{v_i}\,dt \\ v_{i+1} = v_i + \underline{f(x_i)}\,dt \end{cases}$

backward: $v_i = v_{i+1} + \underline{f(x_{i+1})}(-dt)$

$x_i = x_{i+1} + \underline{v_{i+1}}(-dt)$



☆ Time reversibility in Leapfrog is explicit!

☆ In Euler / RK4, time reversibility is only approximated (up to prescribed precision)!

- Recall that in a physical system,

  the total energy of the system is conserved iff

  its Hamiltonian is time invariant.

- Thus, for problems where the explicit conservation of energy is required, explicit time reversibility in the ODE solver is important.

  e.g. long time solutions of particles in a conservative force field.

- another way to say this is that :

  → Euler / RK4 has intrinsic numerical dissipation

  while the leapfrog method is "amplitude" conserving.