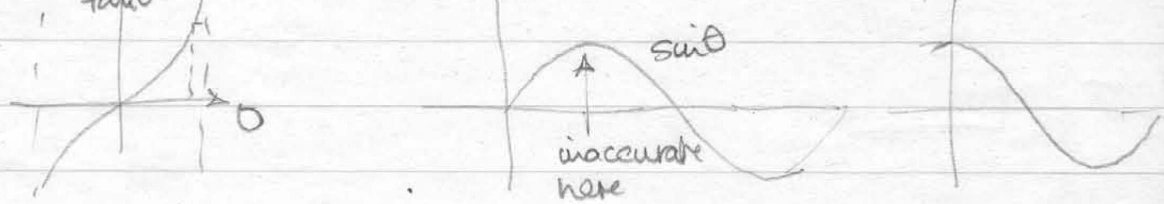
1. Photos
2. Access to mathematica / computers.
   Will use algebraic manip program, Will plot on cptr.
3. How to solve quadratic?
4. Moral: avoid getting solution from difference of 2 large numbers.
5. Given $\sin\theta$ and $\cos\theta$, what is numerically accurate way to get θ? $\sin^{-1}$? $\cos^{-1}$? $\tan^{-1}$
6. Why?



7. $\sinh\theta$ & $\cosh\theta$?
8. Given cubic eq $ax^3 + bx^2 + cx + d = 0$, how would you solve it?

not so useful (a) Analytic solution? ← ~~involves soln of quadratic~~ fastest, esp if you need all roots. Need $\tan^{-1}$ or $\sinh^{-1}$.

(b) Newton - Raphson? ← next fastest. Need init solution. Does NR work for $\mathbb{C}$?

(c) Root finder? ← most generic.

9. Quartic?
   (a) analytic - involves soln of cubic, then quadratic
   must polish real cubic root before completing.
10. General polynomial? All roots, including complex?
    NR recommend "Laguerre" iteration, followed by forward deflation. See NR.
    Convergence slowed if roots are multiple

Solution of quadratic

$$ax^2 + 2bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - ac}}{a}$$ Assume $b \geq 0$ wlog

can get large cancellation

$$\frac{-b - \sqrt{b^2 - ac}}{a}$$ is fine $(b > 0$

but $$\frac{-b + \sqrt{b^2 - ac}}{a}$$ gives large cancellation if $b^2 \gg ac$.

Alt:

$$a + \frac{2b}{x} + \frac{c}{x^2} = 0$$

$$\frac{1}{x} = \frac{-b \pm \sqrt{b^2 - ac}}{c}$$

$$x = \frac{c}{-b \mp \sqrt{b^2 - ac}}$$ is well-behaved alt to

Or,

$$x = \frac{-b + \sqrt{b^2 - ac}}{a}$$

$$= \frac{(-b + \sqrt{b^2 - ac})}{a} \frac{(-b - \sqrt{b^2 - ac})}{(-b - \sqrt{b^2 - ac})}$$

$$= \frac{b^2 - (b^2 - ac)}{a(-b - \sqrt{b^2 - ac})} = \frac{ac}{a(-b - \sqrt{\quad})}$$

$$= \frac{c}{-b - \sqrt{\quad}} \quad \checkmark$$

Extra]
Interpolation of functions

— various parts of Chs 3,4,5 of Press.

Given a set of points, fit a (reasonably) smooth line through them.

Common types of fit:
- Polynomial
- Rational function
- Chebychev polynomial
- Spline

never use it

Why do you need fit?
- Interpolation
- Integration (quadrature)
- ODEs
- PDEs

Need to solve continuum result in finite number of chunks

v. important! Foundation

Polynomial fit

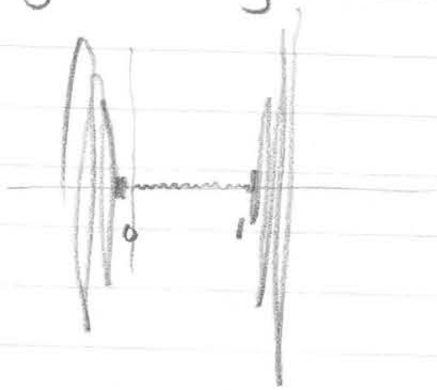
Justification: most differentiable functions 'of interest' are analytic:

$$f(z) = f(z_0) + \underset{\uparrow \atop z - z_0}{\Delta z} f'(z_0) + \frac{\Delta z^2}{2!} f''(z_0) + \dots$$

As $|\Delta z| \to 0$, approxn by first few terms becomes increasing good.

BUT (i) Your function may not be well-behaved
eg. it may have singularity,
or maybe it's not smooth

(ii) $\exists$ polynomials which are arbitrarily good approximations to $f(x) = 0$ in $x \in [0,1]$, but go crazy outside $[0,1]$.



Beware of extrapolation!

Lagrange formula (which you all know)

$$\frac{(x-x_1)\ldots(x-x_N)}{(x_0-x_1)\ldots(x_1-x_N)} f_0 + \frac{(x-x_0)(x-x_2)\ldots(x-x_N)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_N)} f_1 + \ldots$$
$$+ \frac{(x-x_0)\ldots(x-x_{N-1})}{(x_N-x_0)\ldots(x_N-x_{N-1})} f_N$$

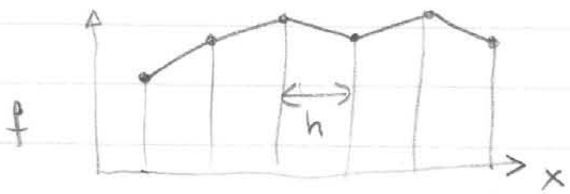
is polynomial of degree N through $N+1$ points $f(x_n) = f_n$

Press gives Neville's algorithm = neat way of building up Lagrange formula recursively.

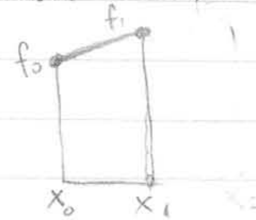
Application of polynomial fitting to integration

Take equally spaced intervals h

1. Trapezium rule



Interpolate each neighboring pair with linear polynomial



Want $\displaystyle\int_{x_0}^{x_1} f(x)\,dx$

$f = (1-t)f_0 + t f_1 \qquad t = \dfrac{x - x_0}{\underbrace{x_1 - x_0}_{h}}$

$$\int_{x_0}^{x_1} f(x)\,dx = h\int_0^1 f(t)\,dt$$

$$= h\int_0^1 (1-t)f_0 + t f_1 \; dt$$

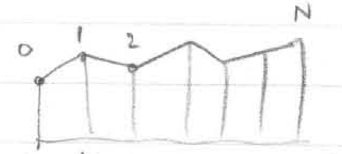
$$= h\left[\left(t - \frac{t^2}{2}\right) f_0 + \frac{t^2}{2} f_1\right]_0^1$$

$$= h\left(\frac{f_0}{2} + \frac{f_1}{2}\right)$$

f is accurate to $O(h^2)$

so $\int_{x_0}^{x_1} f\,dx$ is accurate to $O(h^3)$

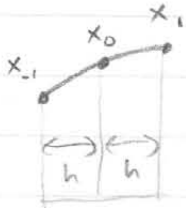
Sum N intervals : $h \sim 1/N$



weight on f_n → $\frac12$ $\frac12$ $\frac12$ $\frac12$ $\frac12$ $\frac12$ $\frac12$

$$\int_{x_0}^{x_N} f\,dx = h\left(\tfrac12 f_0 + f_1 + \dots + f_{N-1} + \tfrac12 f_N\right) + \overset{Nh^3}{O(N^{-2})}$$

2. Simpson rule – some surprises



Fit quadratic to 3 equally spaced points

$$\int_{x_{-1}}^{x_1} f\,dx = h\int_{-1}^{1} f\,dt \qquad t \equiv \frac{x - x_0}{x_1 - x_0 \leftarrow h}$$

$$f = \frac{(t^2-t)}{2}f_{-1} + (1-t^2)f_0 + \frac{(t^2+t)f_1}{2} + O(t^3)$$

$$\int_{x_{-1}}^{x_1} f\,dx = h\left[\left(\frac{t^3}{6}-\frac{t^2}{4}\right)f_{-1} + \left(t - \frac{t^3}{3}\right)f_0 + \left(\frac{t^3}{6}+\frac{t^2}{4}\right)f_1\right]_{-1}^{1}$$

$$= h\left(\frac{f_{-1}}{3} + \frac{4f_0}{3} + \frac{f_1}{3}\right)$$

f is accurate to $O(h^3)$

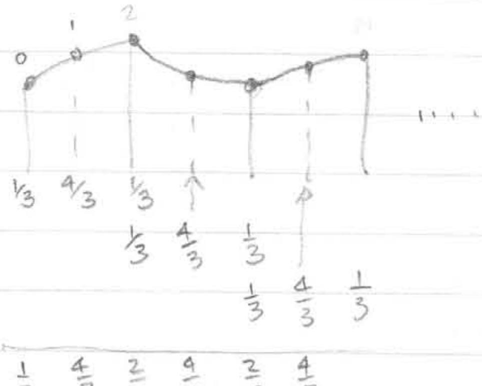
so you'd think $\int_{x_{-1}}^{x_1} f\,dx$ accurate to $O(h^4)$

but actually if you add cubic term:

$$\int_{-1}^{1} t^3\,dt = \left[\frac{t^4}{4}\right]_{-1}^{1} = 0 \qquad \text{disappears}$$

$\Rightarrow \int_{x_{-1}}^{x_1} f\,dx$ accurate to $O(h^5)$ surprise

Sum N intervals $\int_{x_0}^{x_N} f\,dx$

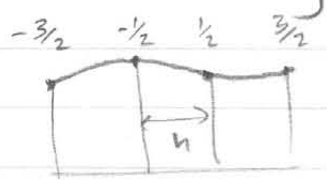


$$= h\left(\frac{f_0}{3} + \frac{4f_1}{3} + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \dots + \frac{4}{3}f_{N-1} + \frac{f_N}{3}\right)$$

$$+ O(\underset{Nh^5}{N^{-4}})$$

Notice way weights alternate. Mysterious! surprise! Are alternating weights somehow better than equal weights? Alternation comes from assymetry in way of treating intervals – some are left, some are right. Better idea to treat intervals on equal footing.

3. Press talks about Simpson's 3/8 rule, which comes from fitting cubic to 4 points:



$$\int_{x_{-3/2}}^{x_{3/2}} f\,dx = h\left(\frac{3}{8} f_{-3/2} + \frac{9}{8} f_{-\frac{1}{2}} + \frac{9}{8} f_{\frac{1}{2}} + \frac{3}{8} f_{\frac{3}{2}}\right)$$

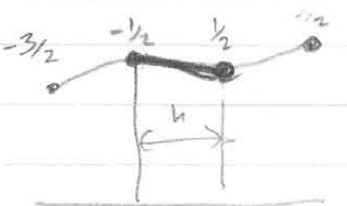
f is accurate to $O(h^4)$

So $\int_{x_{-3/2}}^{x_{3/2}} f\,dx$ is accurate to $O(h^5)$, like Simpson.

Press then meshes Simpson & Simpson 3/8 to get weights which are = 1 inside.

I prefer slightly different approach:

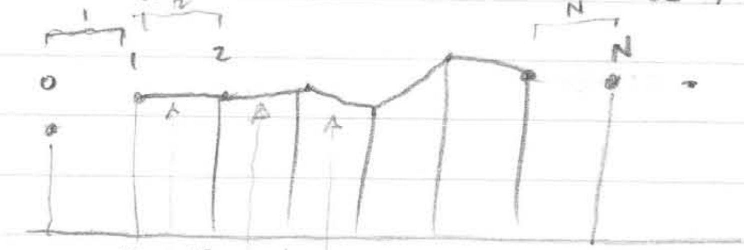
4. Like Simpson 3/8, fit cubic to 4 points:



but now look at integral only over central interval.

Result: $\int_{x_{-1/2}}^{x_{1/2}} f\,dx = h\left(-\frac{1}{24} f_{-3/2} + \frac{13}{24} f_{-\frac{1}{2}} + \frac{13}{24} f_{\frac{1}{2}} - \frac{1}{24} f_{3/2}\right) + O(h^5)$

Now sum over N intervals:

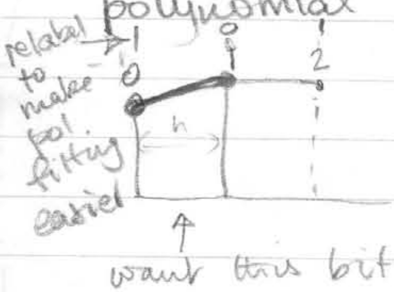


$\int$ each interval is accurate to $O(N^{-5})$

so sum is accurate to $O(N^{-4})$

$$
\begin{array}{cccccc}
-\frac{1}{24} & \frac{13}{24} & \frac{13}{24} & -\frac{1}{24} & & \\
 & -\frac{1}{24} & \frac{13}{24} & \frac{13}{24} & -\frac{1}{24} & \\
 & & -\frac{1}{24} & \frac{13}{24} & \frac{13}{24} & -\frac{1}{24} \\
 & & & -\frac{1}{24} & \frac{13}{24} & \cdots \\
 & & & & \ddots & \\
\hline
-\frac{1}{24} & \frac{12}{24} & \frac{25}{24} & 1 & 1 & \cdots
\end{array}
$$

But what to do about endintervals? Because only 2 endintervals, can take one order lower in polynomial fit and still get $O(N^{-4})$. (Press says you can't do this, but in fact you can)



So fit quadratic to first 3 points (same as Simpson), but take only first interval

$$\int_{x_{-1}}^{x_0} f\,dx = h\left(\frac{5}{12} f_{-1} + \frac{2}{3} f_0 - \frac{1}{12} f_1\right)$$

$$
\begin{array}{lcccccc}
\Rightarrow & -\frac{1}{24} & \frac{12}{24} & \frac{25}{24} & 1 & 1 & 1 \; \cdots \\
\text{+ endpoint contribution} & & \frac{10}{24} & \frac{16}{24} & -\frac{2}{24} & & \\
\hline
 & & \frac{9}{24} & \frac{28}{24} & \frac{23}{24} & &
\end{array}
$$

$$\Rightarrow \int_{x_0}^{x_N} f\,dx = h\left(\frac{3}{8} f_0 + \frac{7}{6} f_1 + \frac{23}{24} f_2 + f_3 + \cdots + \frac{3}{8} f_N\right) + O(N^{-4})$$

Question: is it better to go to higher order or to reduce stepsize h?

Answer: it depends. Going to higher order helps <u>only</u> if you know your function is sufficiently smooth. And h must certainly be smaller than radius of convergence.

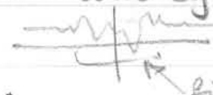
As regards solution of Diff Eqs, must also worry about stability: higher order may mean more spurious solutions to Eqs, which may be disastrous.

Recommendation: A cubic is good compromise in most everyday work. Linear is best when function is not smooth. For a few problems - eg. integration of solar system for long time - higher order scheme may be best

Rational function interpolation

$R(x) = \dfrac{P(x)}{Q(x)}$ ← polynomials is rational function.

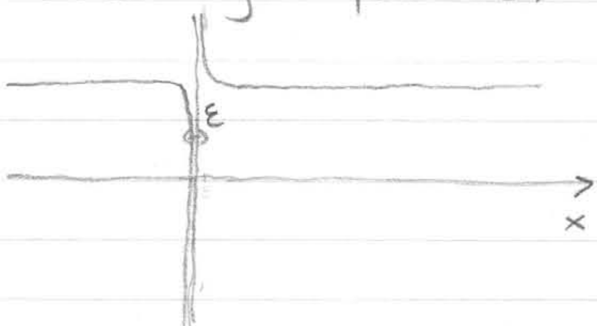
1. Rat. fn. is superior to polynomial in ability to model nearby poles. May work better for your function; may permit longer leaps. Press clearly likes 'em ... but as always "look before you leap"! Advantage



2. Beware of near cancelling factors:

e.g. $\dfrac{x+\varepsilon}{x}$ =

small number



≈ 1 except within $\sim \varepsilon$ of 0

eg.

looks like great fit, but it ain't.

Experience says this happens. Disadvantage of rational functions as opposed to polynomials.

3. Press gives "Neville" type neat, recursive algorithm for generating rat fn fit to set of points.

4. How to solve fit to rational fn?
Ans: rewrite as $RQ = P$, solve resulting linear eqn.

Example
Rational Function Approximation of Power Series / Polynomial

If $R(x) = \dfrac{P(x)}{Q(x)}$

$2n+1$ coeffs

$$r_0 + r_1 x + \dots + r_{2n} x^{2n} + \dots = \frac{p_0 + p_1 x + \dots + p_n x^n}{1 + q_1 x + \dots + q_n x^n} + O(x^{2n+1}) \quad \text{Padé approxn}$$

know r_i, want p_i & q_i

Then $\sum r_i x^i q_j x^j = \sum p_k x^k$ match coeffs of x to x^{2n+1}

$$\Rightarrow \sum_{j=0}^{n} r_{k-j} q_j = p_k \qquad k = 0 \dots 2n \qquad (\text{note } p_k = 0 \text{ for } k > n)$$

eg.
$$\begin{pmatrix} r_0 & & & & \\ r_1 & r_0 & & 0 & \\ r_2 & r_1 & r_0 & & \\ r_3 & r_2 & r_1 & r_0 & \\ r_4 & r_3 & r_2 & r_1 & r_0 \end{pmatrix} \begin{pmatrix} 1 \\ q_1 \\ q_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ 0 \\ 0 \end{pmatrix}$$

Reduces to 2 sets of equns:
$$\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} r_0 & 0 & 0 \\ r_1 & r_0 & 0 \\ r_2 & r_1 & r_0 \end{pmatrix} \begin{pmatrix} 1 \\ q_1 \\ q_2 \end{pmatrix}$$

&
$$\begin{pmatrix} r_3 & r_2 & r_1 \\ r_4 & r_3 & r_2 \end{pmatrix} \begin{pmatrix} 1 \\ q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Toeplitz matrices
= class of matrix with fast soln
– see Press.

which is
$$\begin{pmatrix} r_2 & r_1 \\ r_3 & r_2 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = - \begin{pmatrix} r_3 \\ r_4 \end{pmatrix} \Rightarrow \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \frac{1}{r_1 r_3 - r_2^2} \begin{pmatrix} r_2 & -r_1 \\ -r_3 & r_2 \end{pmatrix} \begin{pmatrix} r_3 \\ r_4 \end{pmatrix}$$

$$= \frac{1}{r_1 r_3 - r_2^2} \begin{pmatrix} r_2 r_3 - r_1 r_4 \\ r_2 r_4 - r_3^2 \end{pmatrix}$$

next order is
$$\begin{pmatrix} r_3 & r_2 & r_1 \\ r_4 & r_3 & r_2 \\ r_5 & r_4 & r_3 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} -r_4 \\ -r_5 \\ -r_6 \end{pmatrix}$$

In general
$$\boxed{\begin{aligned} p_k &= \sum_{j=0}^{k} r_{k-j} q_j && k = 0 \text{ to } n \\ \text{and} \quad \sum_{j=1}^{n} r_{k-j} q_j &= - r_k q_0 && k = n+1 \text{ to } 2n \end{aligned}}$$

In case $\underline{n = 1}$

$$q_0 = 1 \quad \text{(of course)}$$

$$r_1 q_1 = -r_2 \quad \Rightarrow \quad q_1 = -r_2/r_1$$

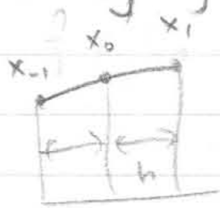
& $$p_0 = r_0 q_0 = r_0$$

$$p_1 = r_1 q_0 + r_0 q_1 = r_1 - \frac{r_0 r_2}{r_1}$$

$$\Rightarrow \frac{P}{Q} = \frac{r_0 + \left(r_1 - \frac{r_0 r_2}{r_1}\right) x}{1 - \frac{r_2}{r_1} x}$$

$$= \frac{r_0 r_1 + (r_1^2 - r_0 r_2) x}{r_1 - r_2 x} = r_0 + \frac{r_1^2 x}{r_1 - r_2 x}$$

Interpolation through 3 equally spaced points:
Previously got quadratic pol fit



$$t = \frac{x - x_0}{x_1 - x_0} \leftarrow = h$$

$$f = f_0 + \left(\frac{f_1}{2} - \frac{f_{-1}}{2}\right) t + \left(\frac{f_{-1}}{2} - f_0 + \frac{f_1}{2}\right) t^2$$

$$= r_0 + r_1 t + r_2 t^2$$

$n = 1$ rat fit

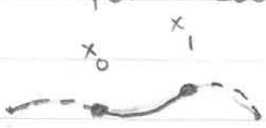
$$= \frac{f_0\left(\frac{f_1 - f_{-1}}{2}\right) + \left[\left(\frac{f_1 - f_{-1}}{2}\right)^2 - f_0\left(\frac{f_{-1}}{2} - f_0 + \frac{f_1}{2}\right)\right] t}{\frac{f_1 - f_{-1}}{2} - \left(\frac{f_{-1}}{2} - f_0 + \frac{f_1}{2}\right) t}$$

Sort of YUK.

Spline

Disadvantage of polynomial or rational interpolation is that derivatives are discontinuous at interpolation points.

Aim of spline is to make derivs continuous.
Most popular is <u>cubic</u> spline.
Idea: fit cubic polynomial between each pair of points



$$f \approx p_0 + p_1 x + p_2 x^2 + p_3 x^3 = P(x)$$

Adjust 4 coefficients so
(a) $P(x)$ goes through $f(x)$ at $x = x_0$ & x_1
(b) $\frac{dP}{dx}$ is continuous at x_0 & x_1 with P of adjacent interval.

= 4 conditions, except at endpoints.
At each of 2 endpoints, need to impose 1 more condition
eg. (c) $\frac{d^2P}{dx^2} = 0$ at endpoints

Requirement that $\frac{dP}{dx}$ is continuous accross interp points is non-local condition. End up needing to solve N linear equations, which look like $A_{ij} f''_j = B_i$
- see Press for details.

functions of x_k, f_k

A_{ij} is tridiagonal ⇝ can be solved in $O(N)$ steps = quick.

Press gives excellent discussion – if you need spline, read him.

Evaluation of functions

Summary of methods:

(1) Power series
- OK where series converges sufficiently strongly.

(2) Transformations:

eg. $\sin(x + 2\pi) = \sin x$

$\Gamma(1+z) = z\Gamma(z)$

$G(1/z) = -G(z) + \frac{1}{2}\ln^2 z + \pi^2/6$

(3) Recurrence relations:
transform a function into others which you have a better hope of evaluating
eg. Legendre polynomial

$$(l+1)P_{l+1}(x) = (2l+1)\,x\,P_l(x) - l\,P_{l-1}(x)$$

starting from $P_{-1} = 0$, $P_0 = 1$

Bessel function

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x)$$

(4) Continued fraction
= infinite rational approximation. eg. incomplete β fn

(5) Asymptotic series.
= series diverges, but first few terms give accurate result for sufficiently extreme values of argument.

We'll be finding each of these works best, depending on function, and value of argument of function.

(2′) Incomplete β-fn

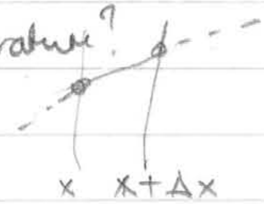
$$B_x(a,b) = \frac{x^a(1-x)^b}{a} + \frac{a+b}{a} B_x(a+1,b) = \int_0^x t^{a-1}(1-t)^{b-1}\,dt$$

$$= -\frac{x^a(1-x)^b}{b} + \frac{a+b}{a} B_x(a,b+1)$$

Note $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ for $a, b > 0$; extend by analytic cont.

N'th order
Runge - Kutta method

Integrate $\boxed{\dfrac{d\vec{y}}{dx} = \vec{f}(x, \vec{y})}$ How does this differ from quadrature?



numerically from x to $x + \Delta x$.

1. Basic assumption:

Each y approximates N'th order polynomial in x.

2. Advantage: simplicity. Depends only on evaluations within interval $[x, x+\Delta x]$, not on history.
3. Disadvantage: Not fastest, as does not use history of previous evaluations.
   Not most robust, nor most efficient.

Example: 2nd order RK — For simplicity take single y, but carries through to vector of y's.

Taylor series solution:

$$y(x+\Delta x) = y + \frac{dy}{dx}\Delta x + \frac{d^2y}{dx^2}\frac{\Delta x^2}{2!} + O(\Delta x^3)$$

$$\frac{dy}{dx} = f \qquad \frac{d^2y}{dx^2} = \frac{df}{dx} = \frac{\partial f}{\partial x} + \underbrace{\frac{dy}{dx}}_{f}\frac{\partial f}{\partial y}$$

so

$$y(x+\Delta x) = y + f\,\Delta x + \underbrace{\left(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right)}_{\text{Know all except this. How to estimate it?}}\frac{\Delta x^2}{2!} + O(\Delta x^3)$$

1st order guess is

$$\Delta y = f\,\Delta x \equiv f_0\,\Delta x$$

Try $f_1 = f(x+\Delta x,\ y+\Delta y)$

$$= f + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\underbrace{\Delta y}_{f_0\Delta x}$$

$$= f + \left(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right)\Delta x$$

Why OK to linear order?

Hey, this looks like desired. Yey!

Organise this as

$$\boxed{\begin{aligned} f_0 &= f(x, y) \\ f_1 &= f(x+\Delta x,\ y + f_0 \Delta x) \end{aligned}}$$

Then

$$\boxed{y(x+\Delta x) = y + \tfrac{1}{2}(f_0 + f_1)\Delta x}$$

General RK

$$y(x+\Delta x) = y + \Delta x \sum_{i=0}^{s} a_i f_i$$

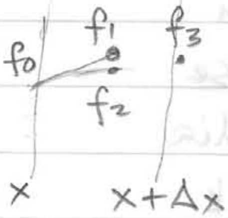
$$\begin{aligned} f_0 &= f(x, y) \\ f_1 &= f(x + c_1 \Delta x,\ y + b_{10} f_0 \Delta x) \\ &\vdots \\ f_i &= f(x + c_i \Delta x,\ y + \Delta x \sum_{j<i} b_{ij} f_j) \end{aligned}$$

$$\begin{array}{c|cccc} c_1 & b_{10} & & & \\ c_2 & b_{20} & b_{21} & & \\ \vdots & & & & \\ c_s & b_{s0} & \cdots & b_{s,s-1} & \\ \hline & a_0 & \cdots & \cdots & a_s \end{array}$$

4'th order RK

The most common (by far?) scheme is following 4th order RK.

$$
\boxed{
\begin{aligned}
f_0 &= f(x, y) \\
f_1 &= f\left(x + \tfrac{1}{2}\Delta x,\ y + \tfrac{1}{2} f_0 \Delta x\right) \\
f_2 &= f\left(x + \tfrac{1}{2}\Delta x,\ y + \tfrac{1}{2} f_1 \Delta x\right) \\
f_3 &= f\left(x + \Delta x,\ y + f_2 \Delta x\right) \\
y &= y + \left(\tfrac{1}{6} f_0 + \tfrac{1}{3} f_1 + \tfrac{1}{3} f_2 + \tfrac{1}{6} f_3\right)
\end{aligned}
}
$$



Run rk4.nb
- Moral: mathematica is pretty dumb,
  but good at checking whether something = 0

Spline

gimp iras-100um_2k.rgb

carina_acs_1920x1200.pnm ← Nathan Smith

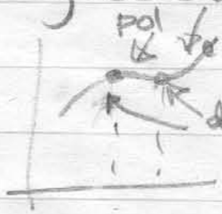
tools → color tools → curves

Fit to PS'

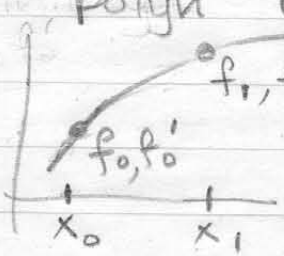
Spline (Nth order) fitting = closest approxn to "fitting w French curve".

A Nth order spline is a fit through a set of points such that

1. The fit on each segment between each pair of adjacent points is a Nth order polynomial.
2. The derivatives is continuous across segments.



Q. What is minimum order polyn that fits through $f(x)$ with specified values
$f(x_0) = f_0$, $f'(x_0) = f'_0$
$f(x_1) = f_1$, $f'(x_1) = f'_1$?



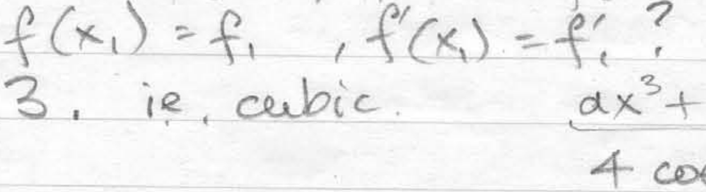
A. 3. ie. cubic $\underline{ax^3 + bx^2 + cx + d}$,
4 coeffs fits 4 things.

3. Can also require higher-order derivs to be cts, eg $f''(x_0) = f''_0$, $f''(x_1) = f''_1$ etc.

Q. Order polyn required to fit up to M'th deriv?

A. $N = 2M + 1$. ⇒ splines almost always odd order.

4. Most common (by far) spline is cubic, which fits values f_i and derivs f_i'.
5. What to do about derivs at endpoints?



For cubic (N'th order) spline, must choose 2 (N−1) boundary conds.

For cubic Most common choice: set $f_0'' = 0$, $f_n'' = 0$, corresponding to extrapolation with straight line. But can also choose f_0' & f_n' to be whatever.

Q: What do you need to know mathematically about spline fitting?
A: It's more complicated than polyn (or rat) fitting, but not ridiculously so.
For cubic spline through n points need to solve system of n linear equations.
Eqs link i'th pt to $i-1$'th and $i+1$'th pts, so mx is tridiagonal.

Q: When might you use spline-fitting?
A: When you want fit that:
(a) is robust (never fails)
(b) has continuous derivs
(c) gives decent (French curve) fit with few pts
(d) extrapolates nicely outside prescribed interval

Ex: fit atomic/molecular cross-sections from a modest number of tabulated pts, with prescribed asymptotic behavior.

Q: When not use spline-fitting?
A: In general numerical integrators,
- overhead of spline-solving too much
- more efficient to use polyn, more points.

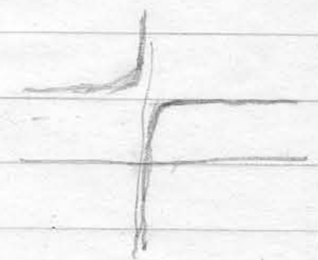
Rational function fitting

~~A~~ rational function $R(x)$ is a ratio of polynomials: $\boxed{R(x) = \frac{P(x)}{Q(x)}}$ ← order m polyn, order n polyn

Q. When would you NOT use rat fn fitting? *

A. In any canned numerical routine.

Why? $\frac{x-\varepsilon}{x} \rightarrow \begin{cases} 0 \text{ at } x = \varepsilon \\ \pm\infty \text{ at } x = 0 \\ 1 \text{ at } x \rightarrow \pm\infty \end{cases}$



Q. When might you use rat fn fitting?

A. When you want to approximate some specific formula(e)

Q: What is advantage of rat fn fitting over polynomial fitting?

A: Can remain well-behaved over all x.

E.g. $y = \frac{1}{1+x^2}$

B: If there are poles, rats can do 'em.

Q: What do you need to know mathematically about rat fn fitting?

A: It's not that hard to do.

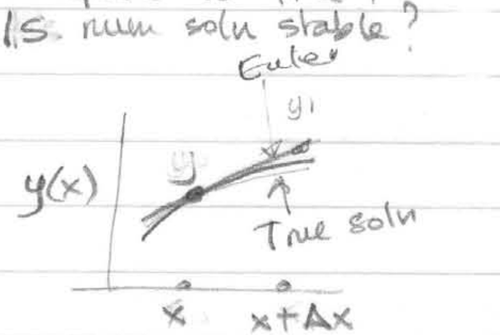
* Mathematica uses polyn interpolation to produce InterpolatingFunction objects.

Diff Eq – numerical solution

- How does num soln compare to true?
- Is num soln stable?

Take simplest possible scheme, 1st order RK (="Euler")

Solve $\dfrac{dy}{dx} = f(x,y)$



by $f_0 = f(x, y_0)$

$y(x+\Delta x) \equiv y + f_0 \Delta x$

Q: Does this come from polyn fitting? A: Yes. order 1.

Consider example

$$\frac{dy}{dx} = -y \qquad \text{with } y = y_0 \text{ at } x = 0$$

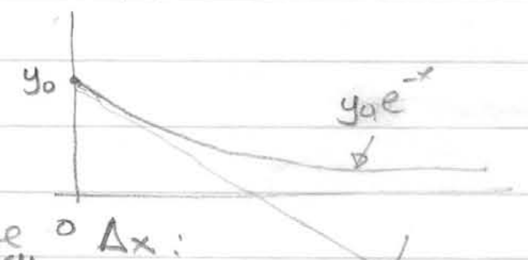
Q: Exact soln is?

A: $y = y_0 e^{-x}$

Euler gives

$f_0 = -y_0$

$y_1 \equiv y(x+\Delta x) = y_0 - y_0 \Delta x = y_0(1-\Delta x)$.



Repeat: at n'th step

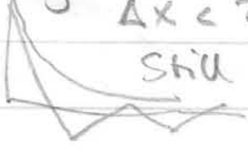
$y_n \equiv y(x+n\Delta x) = y_0 (1-\Delta x)^n$.

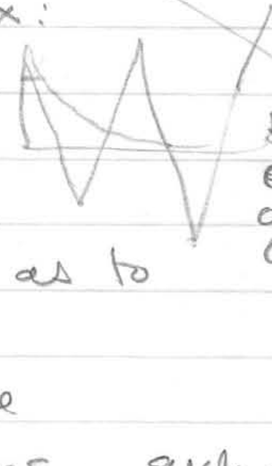
Behavior depends on step size Δx:

Clearly want $\Delta x \ll 1$ for good accuracy

$\Delta x < 1$: OK

$\Delta x < 2$: Still OK





$\Delta x > 2$: Blows up exply, oscillating. Ouch!

Q: Why would you be so stupid as to choose $\Delta x > 2$?

A: You've probably got a more complicated set of coupled eqs such as

$$\frac{d\vec{y}}{dx} = \underset{\text{matrix}}{A}\,\vec{y} \quad \text{ie} \quad \begin{pmatrix} y_1' \\ \vdots \\ y_n' \end{pmatrix} = \begin{pmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & & \\ A_{n1} & \cdots & A_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

For example, y_i might be:
- pops of energy levels of atom
- nuclear abundances in reaction chain
- ionization species
- molecular abundances in chemical chain.

Typically "rate coefficients" A_{ij} depend on eg temperature and other global quantities, but for present purpose treat them as constant.

Q: How to derive exact soln if A_{ij} are const?

A: Diagonalize $A = P^{-1} \Lambda P$

Λ: diagonal mx, whose elts are eigenvalues of A

Then
$$\frac{dy}{dx} = P^{-1} \Lambda P y$$

Define $Y = Py$ ← eigenvectors of Λ

Then
$$\frac{dY}{dx} = \Lambda Y$$

ie
$$\frac{dY_i}{dx} = \Lambda_i Y_i$$

solution
$$Y_i = Y_i(x_0)\, e^{\Lambda_i (x - x_0)}.$$

Problem with this is that some Λ_i may be tiny compared to others.

Y_i with tiny Λ_i are hardly changing;

Y_i with large +ve Λ_i are increasing exply,

Y_i with large −ve Λ_i are decaying exply.

Numerical scheme sees that latter Y_i have decayed to almost zero,

so it increases the step size.
Then bang, your tiny decaying Y_i start blowing up exponentially & oscillating.

Stepsize is limited by the fastest rate
largest $-ve$ Λ_i.
If increase stepsize above this, then solution will blow up. Trouble with this is you end up taking vast numbers of steps in which most Y_i hardly change at all. Not good.

Q: What to do about this?
A: First thing is, diagnose the problem. Having diagnosed problem, two generic options:
(a) remove rapidly varying y_i (the "stiff" y_i) from integration and "fix" them somehow;
(b) use "stiff" numerical integration method (discussed later).

Diff eqs like this are called stiff.

How to apply option (a)?

Typical physical situation:

$$\begin{tikzcd} y_1 \arrow[r, shift left, "fast"] \arrow[d, shift left] & y_2 \arrow[l, shift left] \arrow[dl, shift left, "medium"] \\ y_3 \arrow[u, shift left, "slow"] \arrow[ur, shift left] & \end{tikzcd}$$

might be nuclear species in nuc reaction network.

Fast rates (here $y_0 \leftrightarrows y_1$) tend to drive y_1 & y_2 into "relative "equilibrium". So solution to instability is to force set y_1 & y_2 into relative equilibrium. Have to implement criterion to do this.

Maybe:

(1) Is y_0 small, therefore unimportant? (abundance of species)

(2) Is stepsize becoming large enough that $y_1 \overset{\wedge}{\leftrightarrow} y_2$ is fast

ie $\Lambda \Delta x > 1$?

Λ: appropriate eigenvalue of $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ submatrix.

If so then set y_1 into equilib with y_2.

Problem of instability affects all "explicit" integration schemes, such as Runge-Kutta.

Explicit = calculate new value explicitly from earlier values

as opposed to

Implicit = calculate new value from eq involving new value as well as earlier.

↓ Postpone?

Consider, as next most complicated case, 2nd order RK

Recall: Solves $\frac{dy}{dx} = f(x,y)$

from $f_0 = f(x,y)$

$f_1 = f(x+\Delta x,\ y + f_0 \Delta x)$

$y(x+\Delta x) = y + \frac{1}{2}(f_0 + f_1)\Delta x.$

Example (again):

$\frac{dy}{dx} = -y$ starting from $y = y_0$ at $x = 0$.

2nd R-K gives

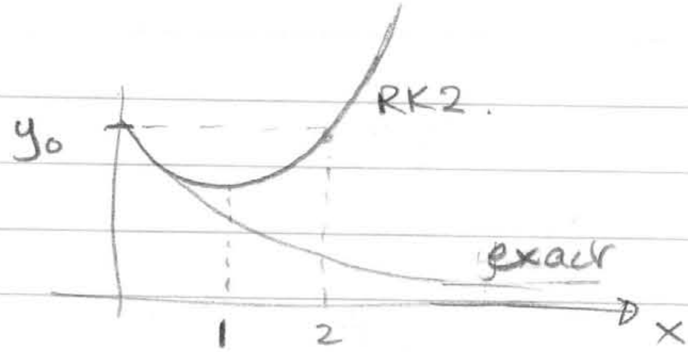
$f_0 = -y_0$ so $\underbrace{y_0 + f_0 \Delta x}_{\text{goes into } f_1} = y_0(1-\Delta x)$

$f_1 = -y_0(1-\Delta x)$

$y_1 \equiv y(x+\Delta x) = y_0\left(1 - \Delta x + \frac{\Delta x^2}{2}\right) = y_0\left[\frac{1}{2} + \frac{1}{2}(1-\Delta x)^2\right]$

whereas true solution is

$y(x+\Delta x) = y_0 e^{-\Delta x} = y_0\left(1 - \Delta x + \frac{\Delta x^2}{2!} - \ldots\right)$



Solution decays for $\Delta x < 2$,
but blows up exply + osc for $\Delta x > 2$.
For $\Delta x > 2$, blowup is more violent
than 1st order scheme.

Higher order integrators are more unstable
than lower order

Stiff integration methods

Are based on implicit schemes

Implicit 1st order RK (= Implicit Euler)

Instead of

explicit: $y_1 = y_0 + f(\underbrace{x_0, y_0}_{\text{value where you are}}) \Delta x$

solve

implicit: $y_1 = y_0 + f(\underbrace{x_1, y_1}_{\text{where you are heading}}) \Delta x$

Called implicit because solution for y_1 depends on $f(x,y)$ at $y = y_1$.
Could for example try iterative soln.
Clearly more work than explicit, especially if evaluating f is expensive

Example $y' = -y$

Instead of

explicit: $y_1 = y_0 - y_0 \Delta x$

ie $y_1 = y_0(1 - \Delta x)$

solve

implicit: $y_1 = y_0 - y_1 \Delta x$

ie $y_1 = \dfrac{y_0}{1 + \Delta x}$

(That was easy, but in general it may not be so easy to solve ..

At n'th step

$$y_n = \frac{y_0}{(1 + \Delta x)^n}$$

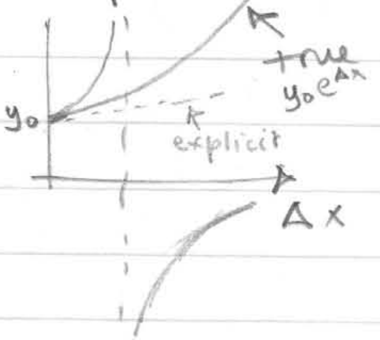
This is stable $\forall \Delta x$. Good

But how well does implicit method work for growing instead of decaying solutions?

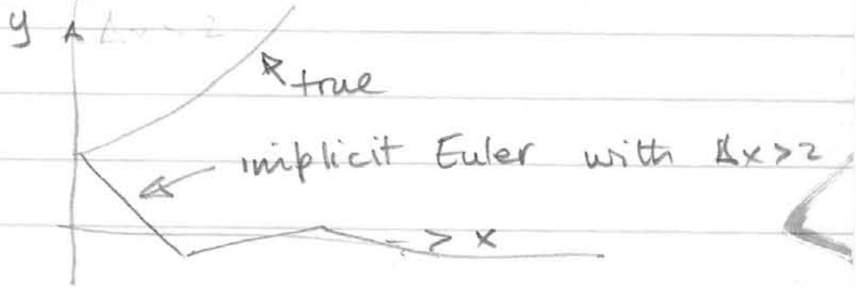
Ex/ $y' = y$
+ instead of −.

Here
explicit: $y_1 = y_0(1+\Delta x)$
whereas
implicit: $y_1 = \dfrac{y_0}{1-\Delta x}$

Implicit:



Soln grows if $\Delta x < 1$
grows & osc if $1 < \Delta x < 2$
decays (!) & osc if $\Delta x > 2$.



Explicit advertized too large stepsize for rapidly decaying modes by blowing up.
Implicit conceals too large stepsize for rapidly growing modes.
Which poison is worse?

$\boxed{\text{Implicit is safe provided that all modes decaying}}$

Q: Does that ever happen?
A: Yes: approach to thermodynamic equilib.

Before addressing what to do about that instability, let's worry about how to solve implicit eqns.

Linearly implicit Euler

Spose diffeqs are approx linear:

$$\frac{d\vec{y}}{dx} = \underset{\text{constant mx}}{A}\ \vec{y}$$

~~ie $\frac{dy_i}{dx} = A_{ij}\, y_j$ (y_i now denotes i'th y not i'th step)~~

explicit Euler;

$$\cancel{y_i(x+\Delta x) = y_i(x) + A_{ij}\, y_i(x)\, \Delta x}$$

or $\vec{y}(x+\Delta x) = \underbrace{(1 + A\Delta x)}_{\text{matrix}}\, \vec{y}(x)$

implicit Euler;

$$\vec{y}(x+\Delta x) = \vec{y}(x) + A\, \vec{y}(x+\Delta x)\, \Delta x$$

ie $(1 - A\Delta x)\, \vec{y}(x+\Delta x) = \vec{y}(x)$

ie $\boxed{\vec{y}(x+\Delta x) = \underbrace{(1 - A\Delta x)}_{\text{matrix}}{}^{-1}\, \vec{y}(x)}$ matrix inverse

So here soln of implicit eqn involves inverting a matrix.
Clearly more expensive numerically than explicit, but not ridiculously.

In general matrix A is not constant.
Linear implicit Euler approximates A by its value at x. Have

$\frac{dy_i}{dx} \equiv f_i = A_{ij}\, y_j$

so $A_{ij} = \partial f_i / \partial y_j$ is "Jacobian" mx

So with this method, derivative routine should return not only $f_i(x,y)$, but also Jacobian mx $\frac{\partial f_i}{\partial y_j}(x,y)$.

Obviously this involves more coding overhead than just coding f_i.
But sometimes that's OK,
because eg A_{ij} may just be rate coeffs
(in nuclear / atomic / ion / molecular network)

Back to stability issue...

Time symmetric approach
Consider again

$y' = Ay$ (just one y, for simplicity)

explicit:

$$y_1 = y_0(1 + A\Delta x)$$

problem: blows up for A -ve and $|A|\Delta x > 2$

implicit:

$$y_1 = \frac{y_0}{1 - A\Delta x}$$

problem: dies for A +ve and $A\Delta x > 2$.

Problem with implicit is just time-reversed version of explicit.

$\Rightarrow$ idea: what about time symmetric method:

$$y_1 = \left(\frac{1 + A\Delta x/2}{1 - A\Delta x/2}\right) y_0$$

Q: is this time symmetric?

A: Yes. Eqn is

$$y_0 = \left(\frac{1 - A\Delta x/2}{1 + A\Delta x/2}\right) y_1$$

which is same with $y_0 \leftrightarrow y_1$

and $\Delta x \to -\Delta x$.

Notice that method is actually 2nd order:

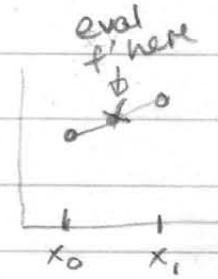
$$\left(\frac{1 + A\Delta x/2}{1 - A\Delta x/2}\right) = \underbrace{1 + A\Delta x + \frac{(A\Delta x)^2}{2!}}_{} + O(\Delta x^3)$$

$$= e^{A\Delta x} \text{ to 2nd order.}$$

Simplest time-symm method:

<u>Implicit midpoint Euler</u>

$$\boxed{y_1 = y_0 + f\left(\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2}\right)\Delta x}$$



Ex/ $y' = Ay$

$$y_1 = y_0 + A\left(\frac{y_0 + y_1}{2}\right)\Delta x$$

ie $\left(1 - \frac{A\Delta x}{2}\right) y_1 = y_0 \left(1 + \frac{A\Delta x}{2}\right)$

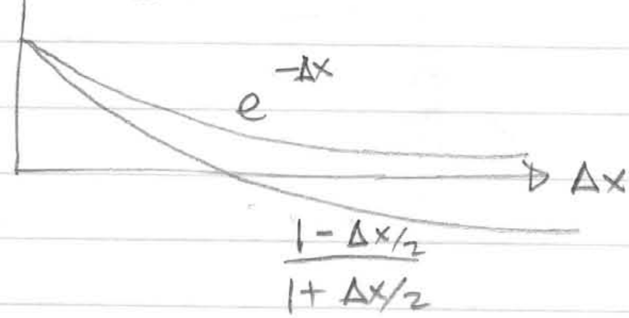
ie $y_1 = \frac{\left(1 + \frac{A\Delta x}{2}\right)}{\left(1 - \frac{A\Delta x}{2}\right)} y_0$ as desired.

At n'th step

$$y_n = \left(\frac{1 + A\Delta x/2}{1 - A\Delta x/2}\right)^n y_0$$

Stability?

$y' = -y$



Stable $\forall\, \Delta x$

As $\Delta x \to \infty$

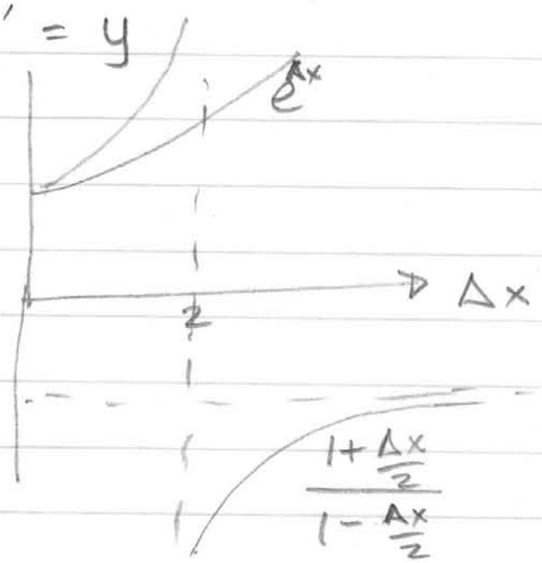
$y_n \to (-)^n y_0$

oscillates

with neutral stability.

OK.

$y' = y$



For $\Delta x > 2$,

oscillates,

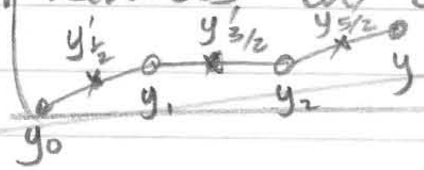
but grows.

At least it is not hiding growth.

Conclusion: time-symmetric approach is safer.

Implicit midpoint Euler is also nice in that it is 2nd order even though f is evaluated at 1 pt (the midpoint).

~~Methods in which y' is evaluated only~~ at midpoints of y are called "leap-frog".



Linearly implicit midpoint Euler

Approximates diffeqs as linear

$$\frac{d\vec{y}}{dx} = \underset{\substack{\uparrow \\ \text{matrix}}}{A} \vec{y}$$

Implicit mid Euler gives

$$y_1 = \underbrace{\left(1 - A\frac{\Delta x}{2}\right)^{-1}}_{\text{matrix}} \underbrace{\left(1 + A\frac{\Delta x}{2}\right)}_{\text{matrix}} y_0$$

← but no need for mx op here.

$$= y_0 + A y_0 \frac{\Delta x}{2}$$

$$= y_0 + f_0 \frac{\Delta x}{2}$$

So

$$\boxed{\begin{aligned} f_0 &= f(x_0, y_0) \\ y_1 &= \left(1 - A\frac{\Delta x}{2}\right)^{-1} \left(y_0 + f_0\frac{\Delta x}{2}\right) \end{aligned}}$$

Jacobean $\frac{\partial f}{\partial y}$ at x_0

Higher-order implicit methods

Can be quite complicated

eg Kaps-Rentrop (eg Timmes 1999
Bader-Deuflhard ApJS, 124, 241)

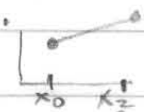
Stepsize control for integrator

Integrator should

- Compute to specified accuracy (relative or abs)
- Adjust stepsize to achieve required accuracy.

Common approach is to compare results at stepsize Δx and $2\Delta x$, use that as estimate of error.

Ex/ 4th order integrator (eg RK).



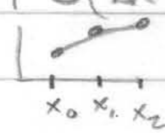
One step of length $2\Delta x$:

$$y_1(x+2\Delta x) = y_{true} + \underbrace{(2\Delta x)^5 a}_{\text{error in } y} + O(\Delta x^6)$$

a ← some coeff

Two steps of length Δx:

$$y_2(x+2\Delta x) = y_{true} + 2(\Delta x)^5 a + O(\Delta x^6)$$



So estimate of error is

$$\Delta y_2 = 2(\Delta x)^5 a$$

ie $$\sigma = \frac{y_2 - y_1}{(2^4-1)\Delta x} = \frac{y_2 - y_1}{15\,\Delta x}$$

If you want error σ,

set $$\frac{\sigma}{\Delta y_2} = \left(\frac{\Delta x}{\Delta x_{desired}}\right)^5$$

or $$\Delta x_{desired} = \Delta x \left(\frac{\sigma}{\Delta y_2}\right)^{1/5}.$$

More typically, change Δx by factor of 2, doubling or halving Δx as nec.

Thus if $\Delta y_2 > \sigma$, then halve Δx

$\Delta y_2 < \frac{1}{2^5}\sigma$, then double Δx

In practice you may want different errors σ_i for different y_i
Typically, you may want
large $|y|$: $\sigma_i = \epsilon |y_i|$ = constant relative err
small $|y|$: $\sigma_i = \epsilon$ = constant error.
Choose $\sigma_{desired}$ = smallest of individual σ_i desired.

<u>Improved estimate</u>

As by-product of need to estimate error, can improve estimate of y_{true} by combining y_1 & y_2:

$$\underbrace{\frac{2^4 y_2 - y_1}{2^4 - 1}}_{= \frac{16}{15} y_2 - \frac{1}{15} y_1} = y_{true} + O(\Delta x^6)$$

Gained an extra power of Δx in accuracy!

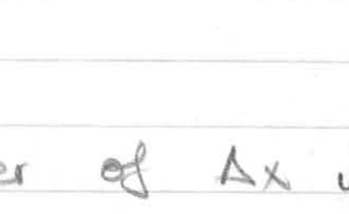
Notion of using multiple stepsizes (a) to estimate error, (b) to reduce error leads to:

<u>Richardson extrapolation</u>

Idea: use several stepsizes Δx over same interval, then extrapolate to $\Delta x \to 0$.

"sufficiently analytic"

For some systems of ODEs, this approach is best available.



Be warned: polyn extrap can fail, and rat fn extrap can fail!