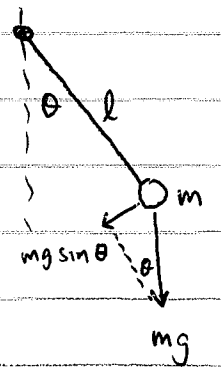
Numerical Solutions of Differential Equations

what is an ODE?

example: simple pendulum



torque: $\tau = -mgl\sin\theta$

moment of inertia: $I = ml^2$

equation of motion: $\tau = I\ddot{\theta}$ ($\ddot{\theta}$: angular acceleration)

(a dot represents a time derivative in mechanics)

$$-mgl\sin\theta = ml^2\ddot{\theta}$$

$$\boxed{\ddot{\theta} + \frac{g}{l}\sin\theta = 0}$$

alternative derivation

Lagrangian: $L = T - U = \frac{1}{2}m(l\dot{\theta})^2 - mgl(1-\cos\theta)$ (T: kinetic energy, U: potential energy)

eqn of motion: $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$

$$\frac{d}{dt}\left(ml^2\dot{\theta}\right) + mgl\sin\theta = 0$$

$$\boxed{\ddot{\theta} + \frac{g}{l}\sin\theta = 0}$$

this is a second order non-linear ODE.

for the solution to be unique, we must specify the initial position and velocity:

$$\theta(0) = \theta_0 \,, \quad \dot{\theta}(0) = \omega_0$$

Reduction to a first order system

introduce a new variable $\omega = \dot{\theta}$, define $y = \begin{pmatrix} \theta \\ \omega \end{pmatrix}$

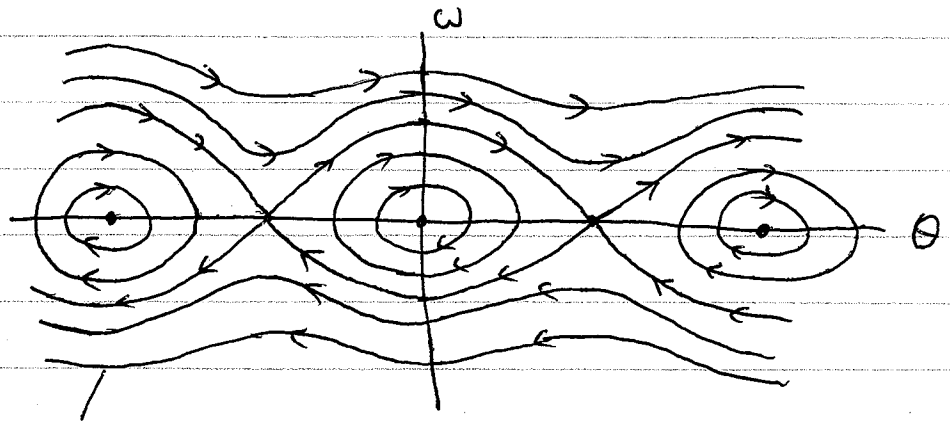
write equation as $\frac{d}{dt}\begin{pmatrix} \theta \\ \omega \end{pmatrix} = \begin{pmatrix} \omega \\ -\frac{g}{l}\sin\theta \end{pmatrix}$

or $\boxed{\dot{y} = f(y) \,, \quad f(y) = \begin{pmatrix} y_2 \\ -\frac{g}{l}\sin y_1 \end{pmatrix}}$

the initial conditions become $\boxed{y(0) = \begin{pmatrix} \theta_0 \\ \omega_0 \end{pmatrix} = \xi}$

it's always possible to reduce a higher order equation to a first order system. We'll assume from now on this has been done.

trajectories in phase space for pendulum:



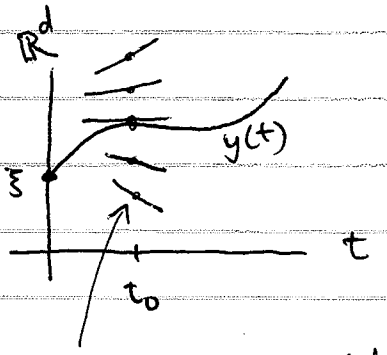
$$T + U = \frac{1}{2} m l^2 \omega^2 + mgl(1 - \cos\theta) = E = \text{const}$$

general setup:

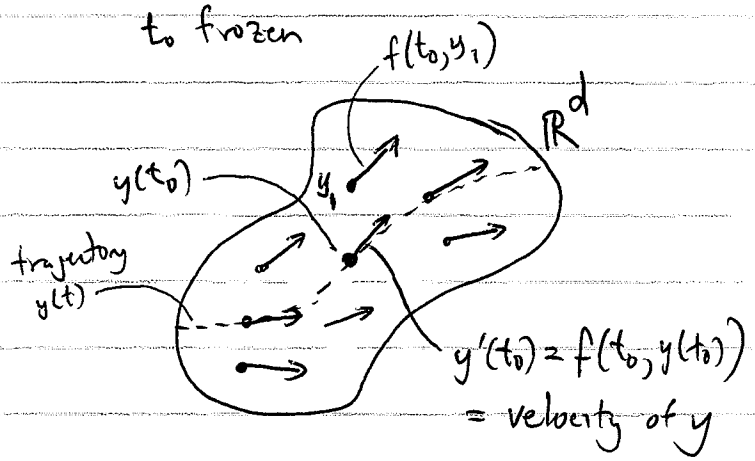
given a vector ξ and a vector-valued function $f(t,y)$, find a C^1 curve $y(t)$ defined on an interval I containing zero such that

$$y'(t) = f(t, y(t)) \qquad t \in I \qquad \left(' = \frac{d}{dt}\right)$$
$$y(0) = \xi$$

geometric interpretation (two pictures)



$f(t_0, y)$ gives the "slope" that the curve $y(t)$ must have when it passes through the point $(t_0, y(t_0))$



A system is autonomous if f doesn't depend on t.

in that case we write $y' = f(y)$

At the other extreme, if f depends on t only, we can integrate the solution exactly:

$$\left.\begin{aligned} \frac{dy}{dt} &= f(t) \\ y(0) &= \xi \end{aligned}\right\} \Rightarrow y(t) = \xi + \int_0^t f(s)\,ds$$

not so interesting...

requirements on f

1. $f: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is continuous. (the domain of f need only be an open subset $U \subset \mathbb{R} \times \mathbb{R}^d$ containing $(0, \xi)$, but for simplicity we'll assume $U = \mathbb{R} \times \mathbb{R}^d$)

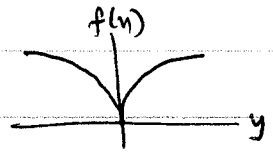
this is enough to ensure existence of solutions, but not uniqueness.

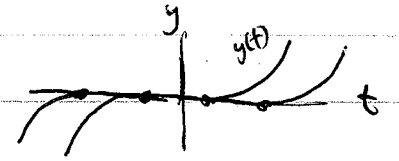
example: $d = 1$, $f(y) = 2\sqrt{|y|}$, $y(0) = 0$

$y(t) = 0$ is a solution (obvious)

$y(t) = \begin{cases} 0 & t \le a \\ (t-a)^2 & t > a \end{cases}$ is also a solution for any $a \in \mathbb{R}$.
(it is continuously differentiable even at $t = a$)

although f is continuous at $y = 0$, it has an infinite slope there which allows multiple solutions to branch apart





non unique solutions

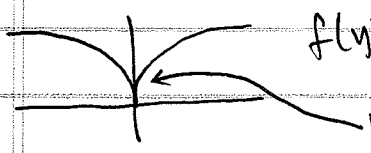
2. f satisfies a Lipschitz condition with respect to y,
i.e. $\exists L > 0$ s.t.

$$\|f(t,x) - f(t,y)\| \le L\|x - y\| \quad \text{for all } x, y \in \mathbb{R}^d,\ t \in \mathbb{R}$$

any norm is fine, e.g.:

$$\begin{cases} \|x\|_1 = \sum_{i=1}^{d} |x_i| & \text{one norm, manhattan norm} \\ \|x\|_2 = \sqrt{\sum_{i=1}^{d} |x_i|^2} & \text{two norm, Euclidean norm} \\ \|x\|_\infty = \max_i |x_i| & \text{infinity norm, max norm} \end{cases}$$

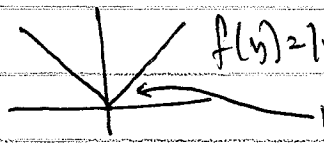
The above example does not satisfy this condition



no hope for $|f(x)-f(y)| \le L|x-y|$

(take $x=0$: $\frac{|f(x)-f(y)|}{|x-y|} = \frac{2\sqrt{|y|}}{|y|} = \frac{2}{\sqrt{|y|}}$ is not bounded by any L)

<u>example:</u> $f(y) = |y|$ is Lipschitz cont. (with $L=1$)

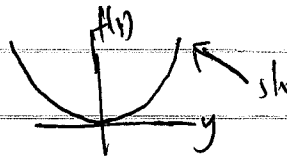


reverse triangle inequality

Nevertheless: $|f(x)-f(y)| = \big||x|-|y|\big| \le |x-y|$

$L=1$ works

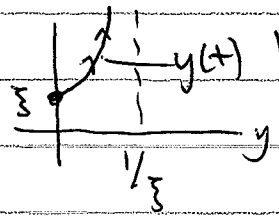
<u>example:</u> $f(y) = y^2$ is not Lipschitz on all of $\mathbb{R}$



slope $f'(y) \to \infty$ as $y \to \infty$ so $\frac{|f(x)-f(y)|}{|x-y|}$ can be made arbitrarily large

the solution of $y' = y^2$, $y(0) = \xi$ is

$$y(t) = \frac{1}{1/\xi - t}$$



→ The Lipschitz condition ensures that solutions are unique and exist for all time. (also: L enters into the error estimates)

3. $f: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is C^r for some $r \geq 1$.

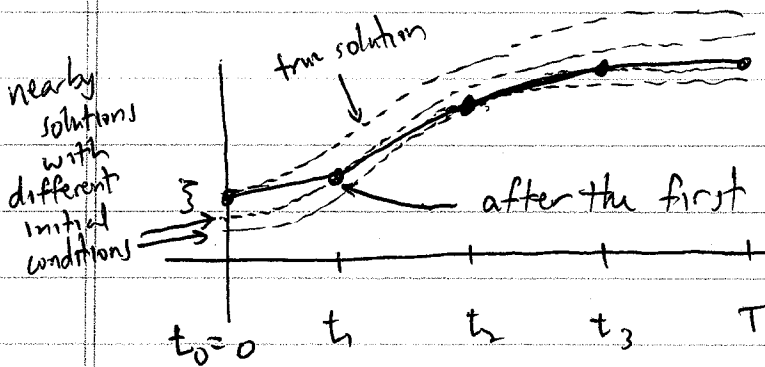
one may show that the solution $y(t, \xi)$ is also C^r as a function of t and the initial condition ξ.

→ useful since analysis of a method is often based on performing a Taylor-expansion of the solution.

Euler's method. Cornerstone of numerics and theory.

want to solve $\left\{ \begin{array}{l} y' = f(t, y(t)) \\ y(0) = \xi \end{array} \right\}$ for $0 \leq t \leq T$

idea: discretize time, construct piecewise linear function using slopes at left endpoint:



after the first step, we use this slope for the next step. This slope is wrong, or rather it is the correct slope of a wrong solution...

algorithm: $n = 0$

$t_0 = 0$, $y_0 \approx \xi$ (e.g. if $\xi = \sqrt{2}$ or π)

while $t_n < T$

$t_{n+1} = t_n + h$ ← stepsize

$y_{n+1} = y_n + h f(t_n, y_n)$

Euler's method is an example of an <u>explicit</u> method.

The value of y_n is known when $f(t_n, y_n)$ is evaluated.

For certain equations (stiff equations), implicit methods are useful.

<u>example</u>: trapezoidal rule

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right]$$

↑ what we're solving for!

Let's group the unknowns together:

$$y_{n+1} - \frac{1}{2}hf(t_{n+1}, y_{n+1}) = \underbrace{y_n + \frac{h}{2}f(t_n, y_n)}_{\text{call this } v}$$

under what conditions can we guarantee that

$$y - \frac{1}{2}hf(t_{n+1}, y) = v$$

has a unique solution $y \in \mathbb{R}^d$ for every $v \in \mathbb{R}^d$?

Such a y exists iff it is a <u>fixed point</u> (i.e. $T(y) = y$) of the operator $T: \mathbb{R}^d \to \mathbb{R}^d$ given by

$$T(y) = \frac{1}{2}hf(t_{n+1}, y) + v$$

def: $T: \mathbb{R}^d \to \mathbb{R}^d$ is a contraction if $\exists\, \alpha < 1$ s.t.

$$\|Tx - Ty\| \leq \alpha \|x - y\| \qquad \forall\, x, y \in \mathbb{R}^d$$

in our case, $\|Tx - Ty\| = \frac{1}{2} h \| f(t_{n+1}, x) - f(t_{n+1}, y) \|$

$$\leq \frac{1}{2} hL \|x - y\| \qquad \text{(Lipschitz condition)}$$

is a contraction as long as $\frac{1}{2} hL < 1$

Contraction mapping theorem: If T is a contraction on a complete metric space (such as $\mathbb{R}^d$), then T has a unique fixed point y such that $Ty = y$.

so as long as the timestep h is small enough, the trapezoidal rule is well-defined.

228A Lec 2

Today we're going to prove convergence of Euler's method and the trapezoidal rule.

useful lemma for estimating errors:

Taylor's theorem with remainder:

Suppose $g: (a,b) \to \mathbb{R}$ is C^{r+1} and $x_0 \in (a,b)$.

Then $$\boxed{g(x) = g(x_0) + g'(x_0)(x-x_0) + \cdots + \frac{g^{(r)}(x_0)}{r!}(x-x_0)^r + R_r(x)}$$

for all $x \in (a,b)$, where

$$\boxed{R_r(x) = \frac{g^{(r+1)}(\theta)}{(r+1)!}(x-x_0)^{r+1}}$$ (Lagrange form of remainder)

(for some θ between x_0 and x)

and

$$\boxed{R_r(x) = \int_{x_0}^{x} \frac{g^{(r+1)}(\theta)}{r!}(x-\theta)^r \, d\theta}$$ (Cauchy form of remainder)

The Cauchy form may be obtained inductively by integration by parts:

$$g(x) = g(x_0) + \int_{x_0}^{x} g'(\theta)\,d\theta \qquad u = g'(\theta),\ dv = d\theta,\ v = -(x-\theta)$$

$$= g(x_0) + g'(x_0)(x-x_0) + \int_{x_0}^{x} g''(\theta)\frac{(x-\theta)}{1!}\,d\theta \qquad u = g''(\theta),\ v = -\frac{(x-\theta)^2}{2!}$$

$$= \cdots$$

The Lagrange form follows from the Cauchy form and the intermediate value theorem:

if $m \le g^{(r+1)}(\theta) \le M$ for $\theta \in (x_0, x)$ then

$$R_r(x) = \int_{x_0}^{x} g^{(r+1)}(\theta) \frac{(x-\theta)^r}{r!} dt \begin{cases} \le \int_{x_0}^{x} M \frac{(x-\theta)^r}{r!} dt = \frac{M}{(r+1)!}(x-x_0)^{r+1} \\ \ge \int_{x_0}^{x} m \frac{(x-\theta)^r}{r!} dt = \frac{m}{(r+1)!}(x-x_0)^{r+1} \end{cases}$$

IVT $\Rightarrow$ $\exists \theta \in (x_0, x)$ s.t. $R_r(x) = \dfrac{g^{(r+1)}(\theta)}{(r+1)!}(x-x_0)^{r+1}$

Both forms imply that if $|g^{(r+1)}(x)| \le M$ for $x \in (a,b)$ then

$$|R_r(x)| \le \frac{M}{(r+1)!}(b-a)^{r+1} \qquad \forall x \in (a,b)$$

The Lagrange form is easier to remember but is clumsy to work with if $\vec{g}(x)$ is a vector as each component of $\vec{g}$ carries a different θ, i.e.

$$g_i(x) - \sum_{k=0}^{r} g_i^{(k)}(x_0)(x-x_0)^k = g_i^{(r+1)}(\theta_i) \frac{(x-x_0)^{r+1}}{(r+1)!}$$

(θ_i: different θ's)

The Cauchy form is identical in the vector case

$$\vec{g}(x) - \sum_{k=0}^{r} \vec{g}^{(k)}(x_0)(x-x_0)^k = \int_{x_0}^{x} \vec{g}^{(r+1)}(\theta) \frac{(x-\theta)^r}{r!} d\theta$$

Exercise: think about the Taylor expansion of $g(x) = \begin{cases} 0 & x \le 0 \\ e^{-1/x} & x > 0 \end{cases}$ at $x = 0$.

Error analysis of Euler's method.

Setup: suppose $f: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is C^2 and satisfies the Lipschitz condition

$$\|f(t,x) - f(t,y)\| \leq L \|x-y\| \qquad x,y \in \mathbb{R}^d, \; t \in \mathbb{R}$$

Then the solution $y(t,\xi)$ of $\begin{pmatrix} y' = f(t,y) \\ y(0) = \xi \end{pmatrix}$

exists for all time and is a C^2 function of t and ξ jointly.

Euler's method: $t_0 = 0$, $y_0 \approx \xi$, $t_{n+1} = t_n + h$ ← stepsize

$$\boxed{y_{n+1} = y_n + h f(t_n, y_n)}$$

goal: show that as $h \to 0$ and $y_0 \to \xi$,

$$\max_{0 \leq t_n \leq T} \|y_n - y(t_n)\| \to 0$$

↑ fixed window over which solution desired

Step 1: find a recursion for the error

Let $e_n = y_n - y(t_n)$

e_n: error; y_n: numerical solution; $y(t_n)$: exact solution

Euler: $y_{n+1} = y_n + hf(t_n, y_n)$

exact soln: $y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \tau_n$

The truncation error τ_n is what's left over when you plug the exact solution into the scheme. The second equation above defines τ_n.

Now subtract to obtain

$$e_{n+1} = e_n + h\left[f(t_n, y_n) - f(t_n, y(t_n))\right] - \tau_n$$

Next we take norms & use the triangle inequality & Lipschitz condition:

$$\begin{aligned} \|e_{n+1}\| &\le \|e_n\| + h\left\| f(t_n, y_n) - f(t_n, y(t_n))\right\| + \|\tau_n\| \\ &\le \|e_n\| + hL\,\|y_n - y(t_n)\| + \|\tau_n\| \\ &= (1+hL)\|e_n\| + \|\tau_n\| \end{aligned}$$

This recursion bounds the error at the next step in terms of the current error and the extent to which the exact solution fails to satisfy the scheme.

Step 2: find a bound on τ_n (show the scheme is consistent)

from Taylor's theorem, we know

$$y(t_{n+1}) = y(t_n) + \underbrace{h y'(t_n)}_{①} + \underbrace{\int_{t_n}^{t_{n+1}} y''(t)(t_{n+1}-t)\,dt}_{②}$$

Note that ① $= h f(t_n, y(t_n))$

and thus ② $= \tau_n$

since the exact solution $y(t)$ is C^2, there is a constant M such that

$$\|y''(t)\| \le M \quad \text{for} \quad 0 \le t \le T$$

$$\therefore \|\tau_n\| = \left\| \int_{t_n}^{t_{n+1}} y''(t)(t_{n+1}-t)\,dt \right\|$$

↘ triangle inequality for integrals

$$\le \int_{t_n}^{t_{n+1}} \|y''(t)\|\,(t_{n+1}-t)\,dt$$

$$\le M \int_{t_n}^{t_{n+1}} (t_{n+1}-t)\,dt = \frac{1}{2} M h^2$$

Thus, the local truncation error is $O(h^2)$.

def: A scheme is consistent of order p if $\|\tau_n\| = O(h^{p+1})$

So Euler's method is consistent of order 1 (or first order)

step 3: analyze the recurrence formula for the error (show that the scheme is stable, i.e. doesn't amplify the truncation errors too much)

By backward recursion, we have

$$\|e_n\| \le (1+hL)\|e_{n-1}\| + \|\tau_{n-1}\|$$

$$\le (1+hL)^2\|e_{n-2}\| + (1+hL)\|\tau_{n-2}\| + \|\tau_{n-1}\|$$

$$\vdots$$

$$\le (1+hL)^n\|e_0\| + (1+hL)^{n-1}\|\tau_0\| + (1+hL)^{n-2}\|\tau_1\| + \cdots + (1+hL)\|\tau_{n-2}\| + \|\tau_{n-1}\|$$

stability: no single error is amplified by more than $(1+hL)^n \le e^{nhL} \le e^{LT}$

making n mistakes kills a power of h

we know that $\|\tau_j\| \le \tau = \frac{1}{2}Mh^2$, so

$$\|e_n\| \le (1+hL)^n\|e_0\| + \left[(1+hL)^{n-1} + \cdots + (1+hL)^1 + 1\right]\tau$$

for any $x \ne 1$ we have $1 + x + \cdots + x^{n-1} = \frac{x^n - 1}{x-1}$

$$\therefore \|e_n\| \le (1+hL)^n\|e_0\| + \frac{(1+hL)^n - 1}{hL}\tau$$

finally, $1+hL \le 1 + hL + \frac{(hL)^2}{2!} + \cdots = e^{hL}$ ← different e

so $$\|e_n\| \le e^{nhL}\|e_0\| + \frac{e^{nhL}-1}{hL}\tau$$

note that $nh = t_n \le T$ so

$$\max_{0 \le t_n \le T} \|e_n\| \le e^{LT}\|e_0\| + \frac{e^{LT}-1}{LT}\frac{T}{h}\tau$$

$\frac{1}{2}MTh$

$\to 0$ as $h \to 0, y_0 \to \xi$

if LT is small, $e^{LT} \approx 1$, $\frac{e^{LT}-1}{LT} \approx 1$

this method of proof (consistency + stability ⇒ convergence) is very powerful. It works for PDE's as well as ODE's.

Trapezoidal rule. For Euler, we had $\tau_n = O(h^2)$. ← first order method

let's try to improve the order using symmetry:

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right]$$

As before, the truncation error is what's left over when you plug the exact solution into the scheme.

$$\tau_n = \underbrace{y(t_{n+1})}_{y(t_n) + hy'(t_n) + \frac{1}{2}h^2y''(t_n) + R_1} - y(t_n) - \frac{h}{2}\left[\overbrace{f(t_n, y(t_n))}^{y'(t_n)} + \overbrace{f(t_{n+1}, y(t_{n+1}))}^{y'(t_n) + hy''(t_n) + R_2}\right]$$

$$= R_1 - \frac{h}{2}R_2 = \int_{t_n}^{t_{n+1}} y'''(t)\frac{(t_{n+1}-t)^2}{2}\,dt - \frac{h}{2}\int_{t_n}^{t_{n+1}} y'''(t)(t_{n+1}-t)\,dt$$

$$= \int_{t_n}^{t_{n+1}} y'''(t)\frac{t_{n+1}-t}{2}\underbrace{\left[t_{n+1}-t - \overbrace{(t_{n+1}-t_n)}^{h}\right]}_{-(t-t_n)}dt$$

$$\therefore \|\tau_n\| \le \frac{M}{2}\int_0^h (h-t)t\,dt = \frac{Mh^3}{12} = \tau$$

$O(h^3)$ now. (second order method)

$$M = \max_{0\le t\le T} \|y'''(t)\|$$

The recursion for the error is very similar to the Euler case:

$$e_{n+1} = e_n + \frac{h}{2}\left[f(t_n, y_n) - f(t_n, y(t_n))\right] + \frac{h}{2}\left[f(t_{n+1}, y_n) - \cdots\right] - \tau_n$$

$$\|e_{n+1}\| \le \|e_n\| + \frac{h}{2}L\|e_n\| + \frac{h}{2}L\|e_{n+1}\| + \tau$$

$$\|e_{n+1}\| \le \frac{1+\frac{1}{2}hL}{1-\frac{1}{2}hL}\|e_n\| + \frac{\tau}{1-\frac{1}{2}hL}$$

remember that we want $\frac{1}{2}hL < 1$ for the trap. rule anyway (to guarantee there is a solution of the implicit equation)

By backward recursion, we have

$$\|e_n\| \le \left(\frac{1+\frac{1}{2}hL}{1-\frac{1}{2}hL}\right)^n \|e_0\| + \underbrace{\left[1 + \cdots + \left(\frac{1+\frac{1}{2}hL}{1-\frac{1}{2}hL}\right)^{n-1}\right]\frac{\tau}{1-\frac{1}{2}hL}}$$

with a bit of algebra $\longrightarrow$ $\left[\left(\frac{1+\frac{1}{2}hL}{1-\frac{1}{2}hL}\right)^n - 1\right]\frac{\tau}{hL}$

now, $\left(\frac{1+\frac{1}{2}hL}{1-\frac{1}{2}hL}\right)^n = \left(1 + \frac{hL}{1-\frac{1}{2}hL}\right)^n \le \exp\left(\frac{nhL}{1-\frac{1}{2}hL}\right)$

So, since $nh \le T$

$$\|e_n\| \le \exp\left(\frac{LT}{1-\frac{1}{2}hL}\right)\|y_0 - \xi\| + \frac{\exp\left(\frac{LT}{1-\frac{1}{2}hL}\right) - 1}{LT}\underbrace{\frac{T}{h}\tau}_{\frac{1}{12}MTh^2}$$

Again, $\frac{T}{h}$ is the number of steps taken (i.e. errors committed) and the scheme doesn't amplify any single error by more than a constant as $h \to 0$.

method is 2nd order

Error estimation and Richardson extrapolation

For Euler's method

$$y_{n+1} = y_n + h f(t_n, y_n)$$

we showed that if $y_0 = \xi$ (i.e. we start with the correct initial condition) then

$$\| y_n - y(t_n) \| \leq \frac{e^{LT} - 1}{LT} \cdot \frac{1}{2} M T h \quad \text{for} \quad 0 \leq t_n \leq T$$

with $M = \max_{0 \leq t \leq T} \| y''(t) \|$

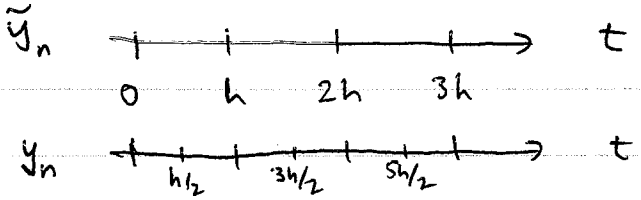
Today we want to understand the structure of this error. It turns out there is a function $\varepsilon(t)$ depending on f and ξ but independent of h such that

$$y_n = y(t_n) + h\varepsilon(t_n) + O(h^2)$$

- y_n: numerical solution via Euler's method
- $y(t_n)$: exact solution
- $h\varepsilon(t_n)$: leading order estimate of the error
- $O(h^2)$: higher order garbage

why do we care? It allows us to estimate the error and improve the calculation.

Suppose you run two calculations, one with stepsize h the other with $h/2$



we can compare the two solutions at their common time values:

$$\tilde{y}_n = y(nh) + h\,\varepsilon(nh) + O(h^2)$$

$$y_{2n} = y\left(2n\cdot\tfrac{h}{2}\right) + \tfrac{h}{2}\,\varepsilon\left(2n\cdot\tfrac{h}{2}\right) + O(h^2) \qquad \leftarrow \left(\tfrac{h}{2}\right)^2 = O(h^2)$$

subtract: $\tilde{y}_n - y_{2n} = \left(h - \tfrac{h}{2}\right)\varepsilon(nh) + O(h^2)$

Thus, the difference between the two solutions is roughly the error on the fine grid. Now let's eliminate ε instead of y:

$$2y_{2n} - \tilde{y}_n = (2-1)\,y(nh) + O(h^2)$$

↖ Richardson extrapolation

Thus, a linear combination of the two solutions is a more accurate estimate of the exact solution than either calculation alone! (it's 2nd order rather than first)

But... we've lost the ability to tell what the error is.

this is typical: if you know the error, you can use it to improve the calculation, but then you don't have an estimate of the error anymore.

variational equations. Before working out what $\varepsilon(t)$ is, let's review how ODE's depend on initial conditions and parameters.

$y' = f(t, y, \mu)$ $\quad$ μ is a parameter

$y(0) = \xi(\mu)$ $\quad$ $y(t, \mu)$ is the solution

$$\frac{\partial}{\partial t}\frac{\partial y}{\partial \mu} = \frac{\partial}{\partial \mu}\frac{\partial y}{\partial t} = \frac{\partial}{\partial \mu} f(t, y, \mu) = D_y f(t, y, \mu)\frac{\partial y}{\partial \mu} + \frac{\partial f}{\partial \mu}$$

here $D_y f = \left(\frac{\partial f_i}{\partial y_j}\right)$ is the Jacobian matrix of f

so $$D_y f \frac{\partial y}{\partial \mu} = \sum_{j=1}^{d} \frac{\partial f}{\partial y_j}\frac{\partial y_j}{\partial \mu}$$

Thus $v(t) = \frac{\partial y}{\partial \mu}(t, \mu_0)$ satisfies the ODE:

$$v'(t) = A(t)v(t) + b(t)$$
$$v(0) = \frac{\partial \xi}{\partial \mu}(\mu_0)$$

← variational equation

where $A(t) = D_y f(t, y(t, \mu_0), \mu_0)$

$$b(t) = \frac{\partial f}{\partial \mu}(t, y(t, \mu_0), \mu_0)$$

↑

are evaluated along the exact trajectory $y(t, \mu_0)$.

this allows us to compute the derivative of the solution with respect to a parameter in the ODE by solving an auxiliary ODE

As a special case, we may consider

$$y' = f(t,y) \quad \leftarrow f \text{ indep. of } \mu$$

$$y(0) = \xi + \mu e_j \quad \leftarrow e_j = (0,\dots,0,1,0,\dots,0)^T \in \mathbb{R}^d$$

(jth slot)

Then $\frac{\partial y}{\partial \mu} = \frac{\partial y}{\partial \xi_j}$ gives the rate of change of the solution with respect to the initial conditions and satisfies:

$$\frac{d}{dt}\left(\frac{\partial y}{\partial \xi_j}\right) = D_y f(t, y(t)) \cdot \left(\frac{\partial y}{\partial \xi_j}\right)$$

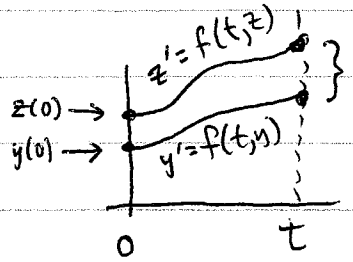
$$\frac{\partial y}{\partial \xi_j}(0) = e_j$$

We can compute the Jacobian $Y(t) = D_\xi y(t) = \left(\frac{\partial y_i}{\partial \xi_j}\right)$ by solving the matrix equation

$$Y'(t) = D_y f(t, y(t))\, Y(t)$$

$$Y(0) = I$$

interpretation: $Y(t)$ tells us how two solutions that start close together evolve together:



$z(t) - y(t) \approx Y(t)(z(0) - y(0))$

Jacobian of the solution w.r.t. initial conditions.

Now let's figure out the equation $\varepsilon(t)$ should satisfy:

try inserting $y_n = y(t_n) + h\varepsilon(t_n) + h^2\varepsilon_2(t_n) + \cdots$
into the scheme:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

$t = t_n$:

$$y(t+h) + h\varepsilon(t+h) + h^2\varepsilon_2(t+h) + \cdots$$
$$= y(t) + h\varepsilon(t) + h^2\varepsilon_2(t) + hf\left(t, y(t) + h\varepsilon(t) + \cdots\right) + \cdots$$

Now Taylor expand y, ε, f:

$$y + hy' + \frac{h^2}{2}y'' + h\left(\varepsilon + h\varepsilon'\right) + h^2\varepsilon_2$$
$$= y + h\varepsilon + h^2\varepsilon_2 + h\left[f + D_y f \cdot (h\varepsilon)\right] + O(h^3)$$

cancel terms, use $y' = f$:

$$h^2\left[\frac{1}{2}y'' + \varepsilon' - D_y f \cdot \varepsilon\right] = O(h^3)$$

divide by h^2 and take the limit as $h \to 0$. We see that ε should satisfy

$$\varepsilon'(t) = D_y f(t, y(t))\,\varepsilon(t) - \frac{1}{2}y''(t)$$
$$\varepsilon(0) = 0$$

← variational equation

So the systematic errors we make using Euler's method lead to a leading order error term similar to varying a parameter in an ODE

The equation for ε is a non-homogeneous linear equation of the form

$$\varepsilon' = A(t)\varepsilon + b(t)$$

where $A(t)$ and $b(t)$ are continuous functions on $0 \le t \le T$.
$\Rightarrow$ solution exists and is unique.

Exercise: show that $L = \max_{0 \le t \le T} \|A(t)\|$ is a Lipschitz constant for the equation.

Now that we think we know $\varepsilon(t)$, we have to actually show that it gives the leading order behavior of the error.

plan: repeat Euler convergence proof replacing $y(t_n)$ by $y(t_n) + h\varepsilon(t_n)$ everywhere. We want a bound on

$$E_n = \underbrace{y_n - y(t_n)}_{e_n \text{ from Lecture 2}} - h\varepsilon(t_n)$$

numerical solution: $y_{n+1} = y_n + hf(t_n, y_n)$

modified truncation error: $y(t_n+h) + h\varepsilon(t_n+h) = y(t_n) + h\varepsilon(t_n) + hf\left(t_n, y(t_n) + h\varepsilon(t_n)\right) + \tau_n$

↖ this equation defines τ_n

subtract: $E_{n+1} = E_n + h\left[f(t_n, y_n) - f(t_n, y(t_n) + h\varepsilon(t_n))\right] - \tau_n$

Take norm, use Lip. cond: $\|E_{n+1}\| \le \|E_n\| + hL\|E_n\| + \|\tau_n\|$

Analyzing the recursion as we did for Euler, we obtain

$$\|E_n\| \le \frac{e^{LT}-1}{LT} \cdot \frac{T}{h}\tau \qquad \left(\frac{T}{h} = \#\text{ of steps}, \quad \tau = \max_n \|\tau_n\|\right)$$

But this time, instead of $\tau = \frac{1}{2}Mh^2$, we have $\tau = O(h^3)$.

proof: expand and match terms:

$$\begin{aligned}\tau_n = \; & y(t_n+h) && + h\varepsilon(t_n+h) \\ & -y(t_n) && -h\varepsilon(t_n) \\ & && -hf\left(t_n, y(t_n)+h\varepsilon(t_n)\right)\end{aligned}$$

$$y(t_n+h) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + R_1$$

$$\varepsilon(t_n+h) = \varepsilon(t_n) + h\varepsilon'(t_n) + R_2$$

$$f\left(t_n, y(t_n)+h\varepsilon(t_n)\right) = g(h) = g(0) + hg'(0) + R_3$$

$$g(s) = f\left(t_n, y(t_n)+s\varepsilon(t_n)\right)$$

$$g'(s) = D_y f\left(t_n, y(t_n)+s\varepsilon(t_n)\right)\cdot\varepsilon(t_n)$$

$$g''(s) = \underbrace{D_y^2 f\left(t_n, y(t_n)+s\varepsilon(t_n)\right)\left(\varepsilon(t_n), \varepsilon(t_n)\right)}$$

$$\text{"}\varepsilon^T H \varepsilon\text{"} = \sum_{j=1}^{d}\sum_{k=1}^{d}\frac{\partial^2 f}{\partial y_j \partial y_k}\varepsilon_j\varepsilon_k$$

↑ Hessian matrix (although f is a vector, so this is really a vector of Hessian matrices)

ugly...

ith component: $g_i''(s) = \varepsilon(t_n)^T H_i\left(t_n, y(t_n)+s\varepsilon(t_n)\right)\varepsilon(t_n)$

$$(H_i)_{jk} = \frac{\partial^2 f_i}{\partial y_j \partial y_k}$$

collecting terms, we get

$$\tau_n = \left[y(t_n) - y(t_n)\right] + h\left[y'(t_n) - f(t_n, y(t_n))\right]$$
$$+ h^2\left[\frac{1}{2}y''(t_n) + \varepsilon'(t_n) - D_y f(t_n, y(t_n)) \cdot \varepsilon(t_n)\right]$$
$$+ \int_{t_n}^{t_{n+1}} y'''(t)\frac{(t_{n+1}-t)^2}{2}\,dt + h\int_{t_n}^{t_{n+1}} \varepsilon''(t_n)(t_{n+1}-t)\,dt + h\int_0^h g''(s)(h-s)\,ds$$

this is still an exact formula. The first 3 terms in the expansion are zero due to the equations satisfied by $y(t)$ and $\varepsilon(t)$.

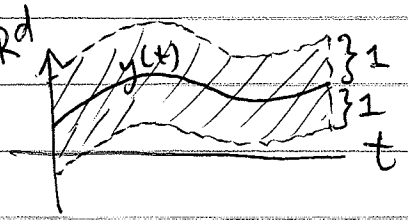
$$\therefore \|\tau_n\| \le \tau = h^3\left[\frac{1}{6}\max_{0\le t\le T}\|y'''(t)\| + \frac{1}{2}\max_{0\le t\le T}\|\varepsilon''(t)\|\right.$$
$$\left. + \frac{1}{2}\max_{\substack{0\le t\le T \\ \|x\|\le 1}}\left\|D_y^2 f(t, y(t)+x)\right\| \cdot \|\varepsilon(t)\|^2\right]$$

a constant independent of h →

we assume here that $(h)\left(\max_{0\le t\le T}\|\varepsilon(t)\|\right) \le 1$ so that

for each t_n we have $\|s\,\varepsilon(t_n)\| \le 1$ i.e. we stay within a band of size 1 around $y(t)$ when evaluating

$$g''(s) = D_y^2 f(t_n, y(t_n) + s\varepsilon(t_n))(\varepsilon, \varepsilon)$$



final result: comp y_n = exact $y(t_n)$ + exact $h\varepsilon(t_n)$ + $O(h^2)$

extremely useful in practice...

Last time: we showed that Euler's method has a predictable leading order error

$$y_n = y(t_n) + h\,\varepsilon(t_n) + O(h^2)$$

numerical solution: $\begin{cases} y_{n+1} = y_n + h f(t_n, y_n) \\ y_0 = \xi \end{cases}$

exact solution of: $\begin{cases} y' = f(t,y) \\ y(0) = \xi \end{cases}$

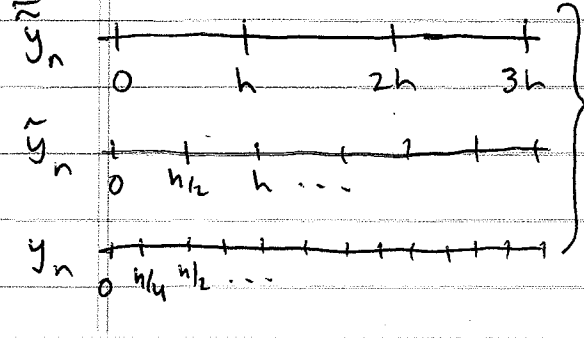
exact solution of variational equation: $\begin{cases} \varepsilon'(t) = D_y f(t, y(t))\,\varepsilon(t) - \frac{1}{2} y''(t) \\ \varepsilon(0) = 0 \end{cases}$

There are a few situations where it's worthwhile to solve the ODE for ε. But the primary use of this analysis is theoretical: knowing that ε exists allows us to compare solutions obtained with different meshsizes to each other to get an idea of how fast the solution is converging, how big we think the error is, etc.

Example: suppose we use a new numerical scheme and we don't know the order of the method. We can guess that

$$y_n = y(t_n) + C(t_n) h^p + O(h^{p+1})$$

and then eliminate $y(t_n)$ and $C(t_n)$ to find p.



compare result for 3 meshes

$$\frac{\tilde{\tilde{y}}_n - \tilde{y}_{2n}}{\tilde{y}_{2n} - y_{4n}} \approx \frac{C(nh)\left(h^p - \left(\frac{h}{2}\right)^p\right)}{C(nh)\left(\left(\frac{h}{2}\right)^p - \left(\frac{h}{4}\right)^p\right)}$$

$$= \frac{4^p - 2^p}{2^p - 1} = 2^p$$

it's also frequently useful to use more than 3 stepsizes and make log-log plots of the error to study the convergence properties of a scheme

e.g. if you run the scheme on a test problem where you know the exact solution, you'll find that

$$\log \underbrace{|y_n - y(t_n)|}_{\text{error}} \approx \log |C(t_n)| + p \log h$$

(h: meshsize)

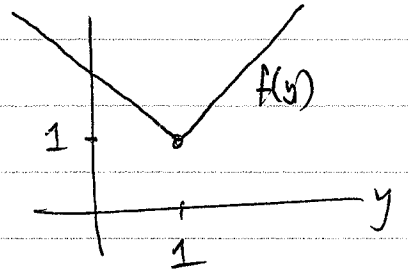
is linear with slope p once h is small enough that $O(h^{p+1})$ can be neglected.

In our convergence proofs, the error estimates are usually in terms of norms of higher derivatives of y.
Let's now consider what happens when f is not smooth.

Example: $y' = f(y)$, $y(0) = 0$

$$f(y) = 1 + |y-1| = \begin{cases} 2-y & y \le 1 \\ y & y \ge 1 \end{cases}$$

this f is Lipschitz continuous:

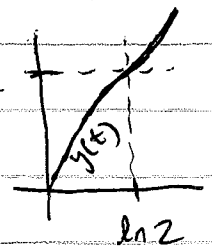


$$|f(x) - f(y)| = \big||x-1| - |y-1|\big|$$
$$\le |(x-1) - (y-1)| = |x-y|$$

↑ reverse triangle inequality

so there's no problem with existence and uniqueness!

in fact, the solution can be worked out explicitly:



$$y(t) = \begin{cases} 2 - 2e^{-t} & 0 \le t \le \ln 2 \\ \frac{1}{2} e^{t} & t \ge \ln 2 \end{cases}$$

$y(t)$ and $y'(t)$ are continuous at $t = \ln 2$. $\boxed{y'(\ln 2) = 1}$

whereas $y''(t)$ has a jump discontinuity: $\boxed{y''(\ln 2^{\pm}) = \pm 1}$

and $y'''(t)$ has a δ-function lurking inside it.

$$y'(t) = \begin{cases} 2e^{-t} & t \le \ln 2 \\ \frac{1}{2} e^{t} & t \ge \ln 2 \end{cases}$$

$$y''(t) = \begin{cases} -2e^{-t} & t \le \ln 2 \\ \frac{1}{2} e^{t} & t \ge \ln 2 \end{cases}$$

This problem is simple enough to carry out Euler's method by hand:

for small t_n, we have

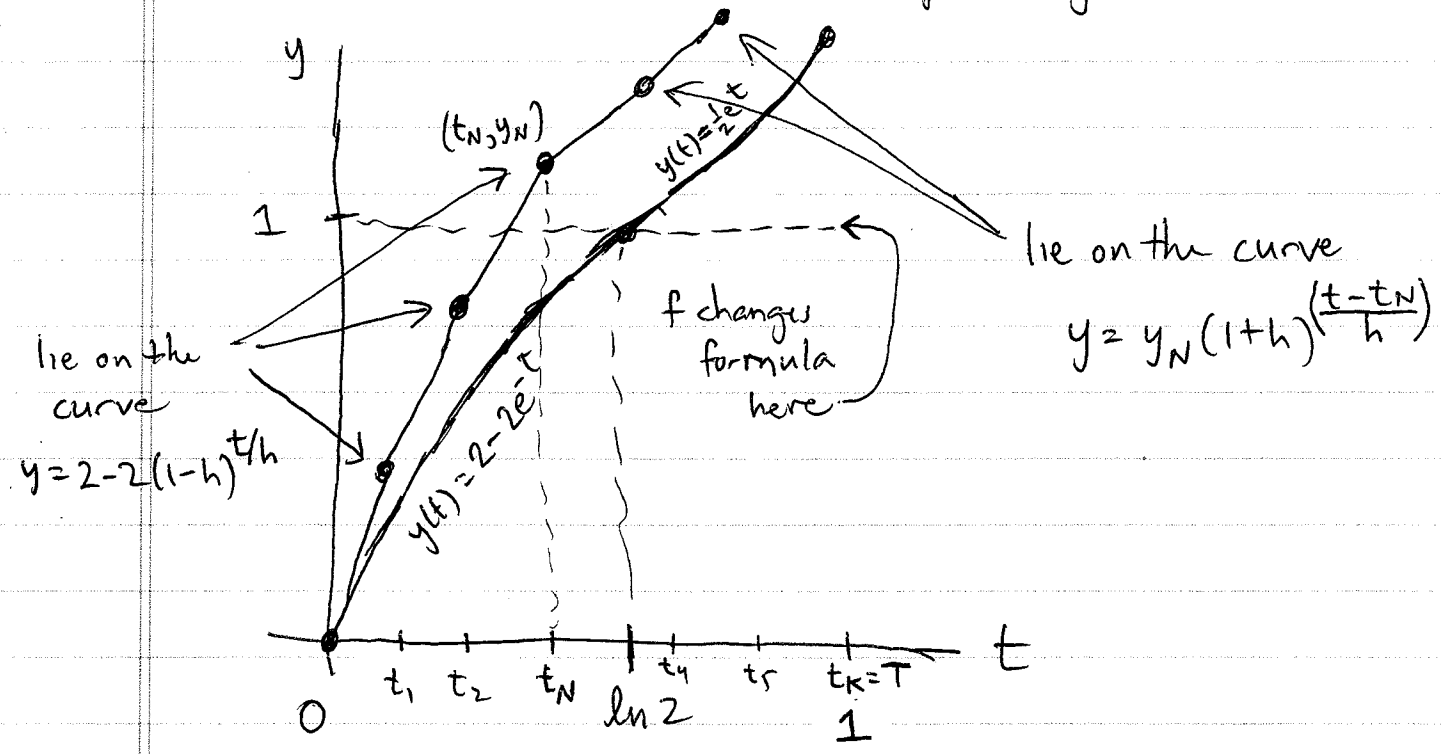
$$y_{n+1} = y_n + h(2 - y_n) = (1-h) y_n + 2h$$

looks familiar:

$$\begin{aligned} y_n &= (1-h) y_{n-1} + 2h \\ &= (1-h)^2 y_{n-2} + [(1-h) + 1] 2h \\ &\vdots \\ &= (1-h)^n y_0 + \frac{1 - (1-h)^n}{1 - (1-h)} 2h \\ &= 2 - 2(1-h)^n \end{aligned}$$

$(n = t_n / h)$

we should continue this until we step over $y=1$:



crossing point: $2-2(1-h)^{t^*/h} = 1$

$$2(1-h)^{t^*/h} = 1$$

$$\frac{t^*}{h} = \frac{\ln 1/2}{\ln(1-h)}$$

$N = \lceil t^*/h \rceil$ ← ceil function $\lceil 4.1 \rceil = 5$, $\lceil 6 \rceil = 6$

now let's continue our trajectory to $T=1$. Assume $h = \frac{T}{K}$

$$y_{N+1} = y_N + hf(y_N) = (1+h)y_N$$

$$y_{N+2} = (1+h)^2 y_N$$

$$\vdots$$

$$y_K = (1+h)^{K-N} y_N$$ ← solution at $T=1$ via Euler

Summary: exact solution: $y(t) = \begin{cases} 2-2e^{-t} & 0 \le t \le \ln 2 \\ \frac{1}{2}e^{t} & t \ge \ln 2 \end{cases}$

numerical solution: $y_n = \begin{cases} 2-2(1-h)^n & 0 \le n \le N \\ (1+h)^{n-N} y_N & n \ge N \end{cases}$

$$N = \frac{\ln(1/2)}{\ln(1-h)} + \theta, \quad 0 \le \theta < 1$$

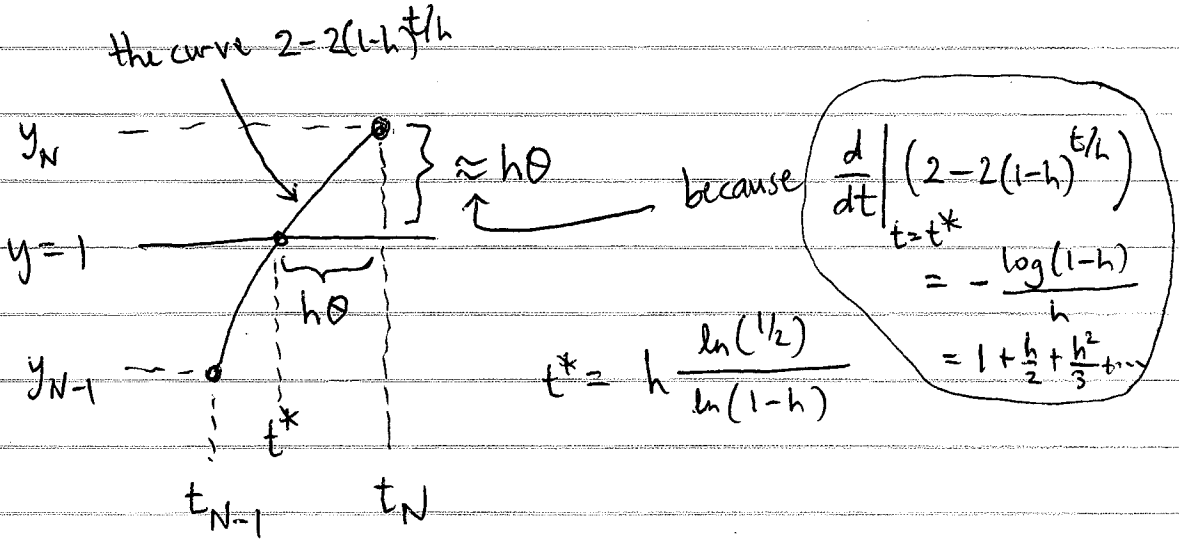
here θ tells us how close we came to landing on $y_N = 1$ exactly

Now we try to expose the dependence of y_K on h. We are interested in whether

$$y_K = y(T) + h\mathcal{E}(T) + O(h^2) \qquad (T=1)$$

still holds in spite of the fact that f is not smooth.

If we zoom in on segment N, we have the picture



because $\left.\frac{d}{dt}\right|_{t=t^*} \left(2-2(1-h)^{t/h}\right) = -\frac{\log(1-h)}{h} = 1 + \frac{h}{2} + \frac{h^2}{3} + \cdots$

$$t^* = h\frac{\ln(1/2)}{\ln(1-h)}$$

If we want the next term in the expansion as well:

$$y_N = 2 - 2(1-h)^N$$

$$= 2 - 2(1-h)^{\left(\frac{\ln(1/2)}{\ln(1-h)} + \theta\right)}$$

$$a^b = e^{b \ln a}$$

$$a = 1-h$$

$$b = \frac{\ln(1/2)}{\ln a}$$

$$= 2 - 2 \cdot \frac{1}{2}(1-h)^\theta$$

$$= 2 - e^{\theta \ln(1-h)}$$

Taylor expand around $h=0$:

$$= 1 + \underbrace{h\theta}_{\text{what we expected}} + \underbrace{\frac{h^2}{2}\theta(1-\theta)}_{\text{hard to guess}} + O(h^3)$$

To evaluate $y_K = (\text{numerical soln at } t_K = T) = (1+h)^{K-N} y_N$

we also need to expand

$$(1+h)^{K-N} = (1+h)^{K - \frac{\ln(1/2)}{\ln(1-h)} - \theta}$$

$K = 1/h$

$$= \exp\left[\left(\frac{1}{h} - \frac{\ln(1/2)}{\ln(1-h)} - \theta\right)\ln(1+h)\right]$$

$$= \exp\left[\frac{\ln(1+h)}{h} - \ln\left(\frac{1}{2}\right)\frac{\ln(1+h)}{\ln(1-h)}\right]\exp\left[-\theta \ln(1-h)\right]$$

(mathematica)

$$= \frac{e}{2}\left[1 + \left(\ln 2 - \frac{1}{2}\right)h + \left(\frac{(\ln 2)^2}{2} - \ln 2 + \frac{11}{24}\right)h^2 + O(h^3)\right]$$

$$\times\left[1 - h\theta + \frac{h^2}{2}\theta(1+\theta) + O(h^3)\right]$$

Combining these results:

the linear terms cancelled... $(1+h\theta+\cdots)(1-h\theta+\cdots) = O(h^2)$

$$y_K = (1+h)^{K-N} y_N$$

$$= \frac{e}{2}\underbrace{\left[1+\left(\ln 2 - \frac{1}{2}\right)h + \cdots\right]}_{\text{a nice expansion in } h \text{ (independent of } \theta)}\left[1+\underbrace{h^2\theta(1-\theta)}_{\text{first effect of discontinuity}} + O(h^3)\right]$$

So the computed solution at $t_K = T = 1$ is equal to

$$y_K = (\text{exact}) + \left(\begin{array}{c}\text{leading error term} \\ \text{of order } h\end{array}\right) + \left(\begin{array}{c}\text{"random" error} \\ \text{term of order } h^2\end{array}\right)$$

$y(1) = \frac{e}{2}$

$\underbrace{\frac{e}{2}\left(\ln 2 - \frac{1}{2}\right)h}_{\varepsilon(1)}$

Can solve the variational equation explicitly to check this

$C(\theta)h^2$

We have little control of θ. It jumps around between 0 and 1 depending on when the numerical solution crosses $y=1$.

Conclusion: making one mistake of order h^2 in the middle of the calculation does not destroy the $O(h^2)$ corrected error

$$E_n = y_n - y(t_n) - h\varepsilon(t_n)$$

If we go back to the backward recursion analysis, we see this makes perfect sense:

$$\|E_n\| \le (1+hL)^n \|E_0\| + (1+hL)^{n-1} \underbrace{\|\tau_0\|} + \cdots + (1+hL) \underbrace{\|\tau_{n-2}\|} + \underbrace{\|\tau_{n-1}\|}$$

If one or just a handful of these τ_i's are $O(h^2)$, the error remains 2nd order. (Most of the terms need to be $O(h^3)$ so that when we add $\frac{1}{h}$ of them, we're left with $O(h^2)$, but a few exceptions are OK)

The same thing would happen with the trapezoidal rule method (method remains 2nd order)

but higher order methods would degrade to 2nd order methods when used to solve this ODE.

---

our book does a similar analysis when f (rather than f') is discontinuous. Usually you don't consider such ODE's, but occasionally you want to model discontinuities in velocity (bat hits a ball, etc.) In this case, an error of size $O(h)$ is committed when the Euler path crosses the discontinuity, so Euler's method remains 1st order, the Trap. rule ceases to be 2nd order, and the leading term of Euler ceases to be a nice function. ($2\,\text{comp}_{h/2} y_{2n} - \text{comp}_h y_n$ will no longer be 2nd order.)

multistep methods

Euler's method and the trapezoidal rule are both examples of one-step methods. All you need to know to compute y_{n+1} is y_n and $f(t,y)$

In a multistep method, we also make use of previously computed values: ($s \geq 1$ is the number of steps)

$$a_0 y_{n+1} + a_1 y_n + \cdots + a_s y_{n+1-s} = h\left[b_0 f_{n+1} + b_1 f_n + \cdots + b_s f_{n+1-s}\right]$$

a_0 is usually normalized to unity

method is explicit if $b_0 = 0$, otherwise implicit

$f_n = f(t_n, y_n)$

equivalently: $a_0 y_{n+s} + \cdots + a_s y_n = h\left[b_0 f_{n+s} + \cdots + b_s f_n\right]$

advantages

1. it's easy to construct methods that are consistent to a high order
2. inexpensive: we re-use data we've already computed
3. theory is interesting

disadvantages

1. it can be difficult to design stable schemes
2. starting the method and changing stepsize is awkward
3. the theory of ODE's works like a one-step method: the equation and an initial condition uniquely determine the solution (past values are irrelevant/redundant)

simplest multistep methods: Adams-Bashforth:

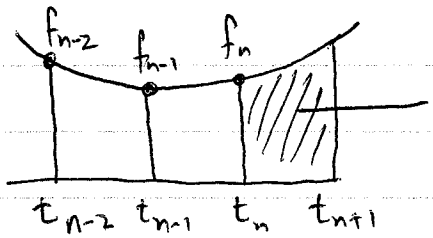
$$y_{n+1} - y_n = h\left[b_1 f_n + \cdots + b_s f_{n+1-s}\right]$$

starting point: $y'(t) = f(t, y(t))$

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t)\,dt = \int_{t_n}^{t_{n+1}} f(t, y(t))\,dt$$

idea: approximate $f(t, y(t))$ between t_n and t_{n+1} using the values of f at $t_{n+1-s}, \ldots, t_n$:

$s = 3$



$$\int_{t_n}^{t_{n+1}} f(t, y(t))\,dt \approx \int_{t_n}^{t_{n+1}} p(t)\,dt$$

we may use Lagrange interpolation to determine the function $p(t)$ passing through these points:

$$p(t) = \frac{(t-t_{n-1})(t-t_{n-2})}{(t_n-t_{n-1})(t_n-t_{n-2})} f_n + \frac{(t-t_n)(t-t_{n-2})}{(t_{n-1}-t_n)(t_{n-1}-t_{n-2})} f_{n-1} + \frac{(t-t_n)(t-t_{n-1})}{(t_{n-2}-t_n)(t_{n-2}-t_{n-1})} f_{n-2}$$

Then $\int_{t_n}^{t_{n+1}} p(t)\,dt = h\int_0^1 p(t_n+\theta h)\,d\theta$

$t = t_n + \theta h \Rightarrow dt = h\,d\theta$

$\begin{cases} t - t_n = (\theta+0)h \\ t - t_{n-1} = (\theta+1)h \\ t - t_{n-2} = (\theta+2)h \end{cases}$

so

$$p(t_n+\theta h) = \frac{(\theta+1)h\cdot(\theta+2)h}{h\cdot 2h} f_n + \frac{\theta h\cdot(\theta+2)h}{-h\cdot h} f_{n-1} + \frac{\theta h\cdot(\theta+1)h}{(-2h)(-h)} f_{n-2}$$

$$= \tfrac{1}{2}(\theta+1)(\theta+2) f_n - \theta(\theta+2) f_{n-1} + \tfrac{1}{2}\theta(\theta+1) f_{n-2}$$

$$\therefore \int_{t_n}^{t_{n+1}} p(t)\,dt = h\int_0^1 \tfrac{1}{2}(\theta^2+3\theta+2) f_n - (\theta^2+2\theta) f_{n-1} + \tfrac{1}{2}(\theta^2+\theta) f_{n-2}\,d\theta$$

$$= h\left[\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right]$$

if we had done the same thing with $s=1$ or 2, we would have obtained:

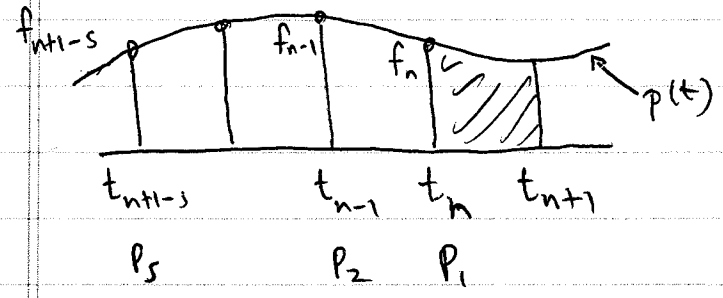
$y_{n+1} = y_n + h f_n$ Euler = 1 step A.B.

$y_{n+1} = y_n + h\left[\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\right]$ 2 step A.B.

$y_{n+1} = y_n + h\left[\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right]$ 3 step A.B.

General procedure for s-step Adams-Bashforth method:

$$p(t) = p_1(t) f_n + p_2(t) f_{n-1} + \cdots + p_s(t) f_{n+1-s}$$



$$p_m(t) = \prod_{\substack{j=1 \\ j \neq m}}^{s} \frac{t - t_{n+1-j}}{t_{n+1-m} - t_{n+1-j}} \qquad 1 \le m \le s$$

$p_m(t)$ is the unique polynomial of degree s satisfying

$$p_m(t_{n+1-j}) = \delta_{mj}$$

The scheme is $y_{n+1} - y_n = \int_{t_n}^{t_{n+1}} p(t)\, dt$

$$= h\left[b_1 f_n + b_2 f_{n-1} + \cdots + b_s f_{n+1-s} \right]$$

where

$$b_m = h^{-1} \int_{t_n}^{t_{n+1}} p_m(t)\, dt = \int_0^1 p_m(t_n + \theta h)\, d\theta = \int_0^1 \tilde{p}_m(\theta)\, d\theta$$

$$\tilde{p}_m(\theta) = \prod_{\substack{j=1 \\ j \neq m}}^{s} \frac{t_n + \theta h - t_{n+1-j}}{t_{n+1-m} - t_{n+1-j}} = \prod_{\substack{j=1 \\ j \neq m}}^{s} \frac{j - 1 + \theta}{j - m}$$

equally spaced case $(t_k - t_\ell = (k-\ell)h)$
otherwise h means $t_{n+1} - t_n$ here (i.e. h_n)

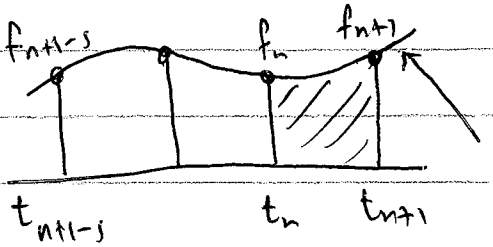
check: $s = 3$

$$\tilde{P}_1(\theta) = \frac{(1+\theta)(2+\theta)}{(1)(2)}, \quad \tilde{P}_2(\theta) = \frac{(\theta)(2+\theta)}{(-1)(1)}, \quad \tilde{P}_3(\theta) = \frac{(\theta)(1+\theta)}{(-2)(-1)}$$

same as before...

Adams-Bashforth is an explicit method.
if we keep the same structure but allow $b_0 \neq 0$, the method becomes implicit.

Adams-Moulton method



interpolate f at t_{n+1}, too.

$$p(t) = p_0(t) f_{n+1} + p_1(t) f_n + \cdots + p_s(t) f_{n+1-s}$$

$$p_m(t) = \prod_{\substack{j=0 \\ j \neq m}}^{s} \frac{t - t_{n+1-j}}{t_{n+1-m} - t_{n+1-j}} \qquad 0 \leq m \leq s$$

only difference is that j starts at zero instead of one.

scheme: $$y_{n+1} - y_n = h\left[b_0 f_{n+1} + \cdots + b_s f_{n+1-s}\right]$$

$$b_m = \int_0^1 \tilde{P}_m(\theta)\, d\theta$$

$$\tilde{P}_m(\theta) = p_m(t_n + \theta h_n) = \prod_{\substack{j=0 \\ j \neq m}}^{s} \frac{t_n + \theta h_n - t_{n+1-j}}{t_{n+1-m} - t_{n+1-j}} = \prod_{\substack{j=0 \\ j \neq m}}^{s} \frac{j - 1 + \theta}{j - m}$$

$h_n = t_{n+1} - t_n$ (equally spaced case)

in class exercise: compute the $s=1$ Adams-Moulton scheme.

$s=3$

$$\tilde{P}_0(\theta) = \frac{(\theta)(1+\theta)(2+\theta)}{(1)(2)(3)} \rightarrow b_0 = \frac{9}{24}$$

$$\tilde{P}_1(\theta) = \frac{(-1+\theta)(1+\theta)(2+\theta)}{(-1)(1)(2)} \rightarrow b_1 = \frac{19}{24}$$

$$\tilde{P}_2(\theta) = \frac{(-1+\theta)(\theta)(2+\theta)}{(-2)(-1)(1)} \rightarrow b_2 = -\frac{5}{24}$$

$$\tilde{P}_3(\theta) = \frac{(-1+\theta)(\theta)(1+\theta)}{(-3)(-2)(-1)} \rightarrow b_3 = \frac{1}{24}$$

compare:

3 step A.B: $$y_{n+1} - y_n = h\left[\frac{23}{12}f_n - \frac{16}{12}f_{n-1} + \frac{5}{12}f_{n-2}\right]$$

3 step A.M: $$y_{n+1} - y_n = h\left[\frac{9}{24}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right]$$

As we will see, 3 step A.B. is 3rd order while
3 step A.M is 4th order

moreover, A.M. methods have better stability properties.

but... because A.M. is an implicit method, you need a good starting guess for the solution using e.g. Newton's method. Idea: use A.B. for the starting guess. (example of a predictor corrector method.)

So how would we implement this predictor/corrector method?

initial guess

$$y_{n+1}^{(0)} = y_n + h\left[\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right]$$

we're trying to solve

$$y_{n+1} - h\frac{9}{24} f(t_{n+1}, y_{n+1}) = \underbrace{y_n + h\left[\frac{19}{24} f_n - \frac{5}{24} f_{n-1} + \frac{1}{24} f_{n-2}\right]}_{u}$$

(u known, so compute it once and give it a name)

define

$$F(y) = y - h\frac{9}{24} f(t_{n+1}, y) - u$$

note that the Jacobian of F is closely related to that of f:

$$DF(y) = I - h\frac{9}{24} D_y f(t_{n+1}, y)$$

So Newton's method would look like

predictor → $\nu = 0$, $y^{(0)} = y_n + h\left[\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right]$

$u = \cdots$

$F^{(0)} = F(y^{(0)})$

corrector:

while $\|F^{(\nu)}\| > tol$ ← e.g. $tol = 10^{-12}$

$y^{(\nu+1)} = y^{(\nu)} - DF(y^{(\nu)})^{-1} F^{(\nu)}$

$F^{(\nu+1)} = F(y^{(\nu)})$

$\nu = \nu + 1$

$y_{n+1} = y^{(\nu)} - DF(y^{(\nu)})^{-1} F^{(\nu)}$ ← one more iteration to get the last digits right

sometimes forming the Jacobian of f (and hence F) is too expensive or tedious to code up.

An alternative to Newton's method is the iteration

$$y^{(\nu+1)} = y_n + h\left[f(t_{n+1}, y^{(\nu)}) + b_1 f(y_n) + \cdots + b_s f_{n+1-s}\right]$$

($y^{(\nu)}$: lags behind by one iteration)

in the previous notation, we're solving $F(y) = 0$ iteratively via

$$F(y^{(\nu+1)}) \simeq y^{(\nu+1)} - h b_0 f(t_{n+1}, y^{(\nu)}) - u = 0$$

This works because

$$y^{(\nu+1)} - y^{(\nu)} = h b_0 \left[f(t_{n+1}, y^{(\nu)}) - f(t_{n+1}, y^{(\nu-1)}) \right]$$

so $$\| y^{(\nu+1)} - y^{(\nu)} \| \leq \underbrace{h|b_0|L}_{\alpha} \| y^{(\nu)} - y^{(\nu-1)} \| \leq \alpha^{\nu} \| y^{(1)} - y^{(0)} \|$$

Thus successive corrections become smaller and smaller and we have

$$\| y^{(\infty)} - y^{(\nu)} \| \leq \sum_{j=\nu}^{\infty} \| y^{(j+1)} - y^{(j)} \| \leq \left(\sum_{j=\nu}^{\infty} \alpha^j \right) \| y^{(1)} - y^{(0)} \|$$

$$= \left(\frac{\| y^{(1)} - y^{(0)} \|}{1 - \alpha} \right) \alpha^{\nu} \qquad \alpha = h|b_0|L < 1$$

(assume h small enough)

pros: easy to implement, don't need to compute Df

cons: converges only linearly with ν (the number of correct digits increases linearly with the # of iterations)

by contrast, Newton's method converges quadratically. $\| y^{(\infty)} - y^{(\nu)} \| \leq C\alpha^{2^{\nu}-1}$
(# of correct digits grows exponentially with ν)

228A Lec 6

Before continuing with our analysis of multistep methods, let's review our multivariable calculus.

Advanced calculus is the study of approximating non-linear functions

$$f: \mathbb{R}^n \to \mathbb{R}^m$$

by simpler linear (or polynomial) functions:

$$f(x) \approx f(x_0) + \underbrace{Df(x_0)}_{\substack{\text{Jacobian} \\ \text{matrix} \\ m \times n}} \underbrace{(x - x_0)}_{\substack{\text{displacement} \\ \text{vector} \\ (\text{belongs to } \mathbb{R}^n)}} + O(\|x - x_0\|^2)$$

↑ matrix multiply

2 easy cases

$m = 1$ $f: \mathbb{R}^n \to \mathbb{R}$ is a scalar function of several variables

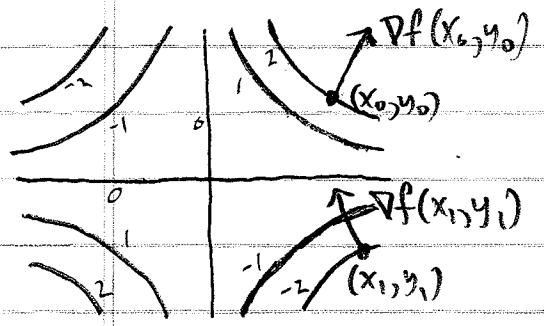
The gradient $\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$ is a column vector pointing in the direction of maximum increase. The derivative $Df = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right)$ is a linear mapping (represented as a $1 \times n$ Jacobian matrix or a row vector in this case) telling how $f(x)$ changes locally as we vary x. The linear approximation of f at x_0 is

$$P_{x_0}(x) = f(x_0) + \nabla f(x_0) \cdot (x - x_0) = f(x_0) + Df(x_0)(x - x_0)$$

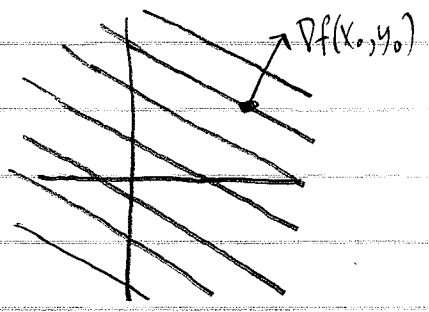
↑ dot product

↑ matrix multiplication.

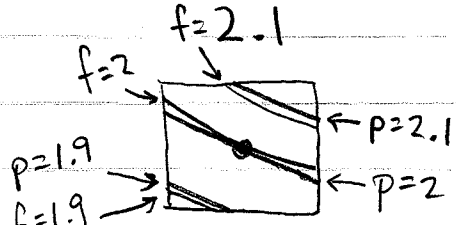
Example: $f(x,y) = xy$, $\nabla f(x_0,y_0) = \begin{pmatrix} y_0 \\ x_0 \end{pmatrix}$



contour plot of f
∇f is perpendicular to contour lines



contour plot of $P_{(x_0,y_0)}(x,y)$
contours are parallel straight lines



zoomed in

Near (x_0,y_0), the contours of p and f are very close to each other

other easy case: $n=1$, $f: \mathbb{R} \to \mathbb{R}^m$

now f is a vector function of a scalar parameter

Example: the solution $y(t)$ of our ODE's is a vector valued function of time

This is the case in which Taylor's theorem with remainder applies

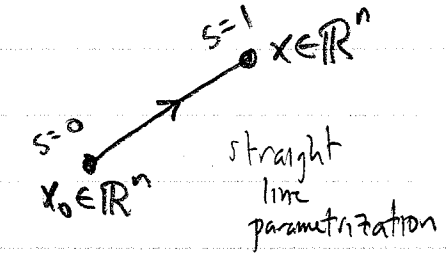
$$y(t) = y(t_0) + y'(t_0)(t-t_0) + \cdots + y^{(r)}(t_0)\frac{(t-t_0)^r}{r!} + R_r(t,t_0)$$

$$R_r(t,t_0) = \int_{t_0}^{t} y^{(r+1)}(s)\frac{(t-s)^r}{r!}\,ds$$

The general case $f: \mathbb{R}^n \to \mathbb{R}^m$ can be thought of component by component as the "$m=1$" case, and Taylor's theorem can be used by parametrizing the change from x_0 to x with a straight line (which reduces the problem to the "$n=1$" case)

$$f(x) - f(x_0) = g(1) - g(0)$$

$$g(s) = f\big(x_0 + s(x - x_0)\big)$$



important cases:

chain rule

(1) $$f(x) - f(x_0) = \int_0^1 g'(s)\,ds = \int_0^1 Df\big(x_0 + s(x - x_0)\big)(x - x_0)\,ds$$

$$\|f(x) - f(x_0)\| \le \int_0^1 \|Df(x_0 + s(x - x_0))\| \cdot \|x - x_0\|\,ds$$

$$\le \left(\max_{0 \le s \le 1} \|Df(x_0 + s(x - x_0))\|\right)\|x - x_0\| \le L\|x - x_0\|$$

so the maximum norm of the Jacobian matrix

$$L = \max_{x \in \mathbb{R}^n} \|Df(x)\|$$

can serve as a Lipschitz constant for f.

(2) $$f(x) - f(x_0) - Df(x_0)(x - x_0) = \int_0^1 g''(s)(1 - s)\,ds$$

$$g'(s) = Df\big(x_0 + s(x - x_0)\big)(x - x_0) = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\big(x_0 + s(x - x_0)\big)\underbrace{(x_j - x_{0j})}_{\text{a number (scalar)}}$$

note: $$\frac{d}{ds}\frac{\partial f}{\partial x_j}\big(x_0 + s(x - x_0)\big) = \sum_{k=1}^{d} \frac{\partial^2 f}{\partial x_k \partial x_j}\big(x_0 + s(x - x_0)\big)(x_k - x_{0k})$$

$\frac{\partial f}{\partial x_j}$ is a vector

so

$$g''(s) = \sum_{j=1}^{n}\sum_{k=1}^{n} \underbrace{\frac{\partial^2 f}{\partial x_k \partial x_j}\big(x_0 + s(x - x_0)\big)}_{\text{a vector}}\underbrace{(x_j - x_{0j})(x_k - x_{0k})}_{\text{a scalar}}$$

alternative notation → $D^2 f\big(x_0 + s(x - x_0)\big)(x - x_0, x - x_0)$

We conclude that

$$\|f(x) - f(x_0) - Df(x_0)(x-x_0)\| \le \int_0^1 \|g''(s)\|\,(1-s)\,ds \le \Big(\max_{0\le s\le 1}\|g''(s)\|\Big)\underbrace{\int_0^1 1-s\,ds}_{1/2}$$

How we proceed from here depends on the norm. It's not hard to show that

$$\|g''(s)\|_1 \le C_1\|x-x_0\|_1^2, \quad C_1 = \max_{x\in\mathbb{R}^n}\max_{j,k}\left\|\frac{\partial^2 f}{\partial x_k \partial x_j}(x)\right\|_1$$

$$\|g''(s)\|_\infty \le C_\infty\|x-x_0\|_\infty^2, \quad C_\infty = \max_{x\in\mathbb{R}^n}\sum_{j,k}\left\|\frac{\partial^2 f}{\partial x_k \partial x_j}(x)\right\|_\infty$$

$$\|g''(s)\|_2 \le C_2\|x-x_0\|_2^2, \quad C_2 = \max_{x\in\mathbb{R}^n}\sqrt{\sum_{j,k}\left\|\frac{\partial^2 f}{\partial x_k \partial x_j}(x)\right\|_2^2}$$

Exercise: verify this claim. For the last one, you'll need the Cauchy-Schwarz inequality

$$\Big|\sum_i a_i b_i\Big| \le \|a\|_2\|b\|_2 = \Big(\sum_i a_i^2\Big)^{1/2}\Big(\sum_j b_j^2\Big)^{1/2}$$

Conclusion:

$$f(x) = f(x_0) + Df(x_0)(x-x_0) + O\big(\|x-x_0\|^2\big)$$

where the constant in the error depends on f via its second partial derivatives. This technique of parametrizing via $x_0 + s(x-x_0)$ can also be used to show

$$f(x) = f(x_0) + Df(x_0)(x-x_0) + \tfrac{1}{2}D^2 f(x_0)(x-x_0, x-x_0) + O(\|x-x_0\|^3)$$

$$f(x) = \text{"} \quad \text{"} \quad \text{"} + \tfrac{1}{6}D^3 f(x_0)(x-x_0, x-x_0, x-x_0) + O(\cdot^4)$$

etc.

Newton's method. $F: \mathbb{R}^n \to \mathbb{R}^n$ (nonlinear function), solve $F(x) = 0$

idea: start with an initial guess $x^{(0)}$

then let $x^{(\nu+1)}$ solve the linearized equation

$$F(x) \approx \underbrace{F(x^{(\nu)}) + DF(x^{(\nu)})(x - x^{(\nu)})}_{\text{linearization of } F \text{ at } x^{(\nu)}} = 0$$

solution: $x^{(\nu+1)} = x^{(\nu)} - \underbrace{DF(x^{(\nu)})^{-1}}_{\text{inverse of the } n \times n \text{ Jacobian matrix}} F(x^{(\nu)})$

$$DF(x^{(\nu)})_{ij} = \frac{\partial F_i}{\partial x_j}(x^{(\nu)})$$

Theorem

if $F \in C^2$ near x^*, $F(x^*) = 0$

and $DF(x^*)$ is invertible, then $\exists \rho > 0, C > 0$

s.t.

if $\|x^{(\nu)} - x^*\| < \rho$ then $\|x^{(\nu+1)} - x^*\| \leq C\|x^{(\nu)} - x^*\|^2$

(quadratic convergence)

we may decrease ρ if necessary

so that $\rho C = \alpha < 1$

proof: ① Since $DF(x^*)$ is continuous and invertible,

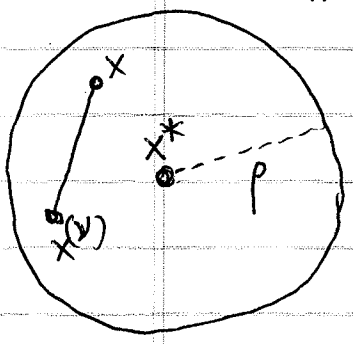
$\exists \rho > 0, C_1 > 0$ s.t.

$$\|DF(x)^{-1}\| \leq C_1 \quad \text{for} \quad \|x - x^*\| < \rho$$

② Since F is C^2 (twice continuously differentiable) $\exists C_2 > 0$ s.t.

$$\|F(x) - F(x^{(\nu)}) - DF(x^{(\nu)})(x - x^{(\nu)})\| \leq C_2 \|x - x^{(\nu)}\|^2$$

whenever $\|x - x^*\| < \rho$ and $\|x^{(\nu)} - x^*\| < \rho$



← we showed earlier how the error in the linear approximation depends on $\frac{\partial^2 F}{\partial x_j \partial x_k}$ along the line joining $x^{(\nu)}$ to x.

③ $\therefore \quad F(x^*) = F(x^{(\nu)}) + DF(x^{(\nu)})(x^* - x^{(\nu)}) + E = 0$

$$\|E\| \leq C_2 \|x^{(\nu)} - x^*\|^2$$

$$x^* = -DF(x^{(\nu)})^{-1}\left[F(x^{(\nu)}) + E\right] \quad \leftarrow \text{exact formula}$$

$$x^{(\nu+1)} = -DF(x^{(\nu)})^{-1} F(x^{(\nu)}) \quad \leftarrow \text{exact formula}$$

$$\therefore \|x^{(\nu+1)} - x^*\| = \|DF(x^{(\nu)})^{-1} E\| \leq \underbrace{C_1 C_2}_{C} \|x^{(\nu)} - x^*\|^2$$

as claimed.

Significance: Consider 2 algorithms for computing x^* iteratively.

algorithm 1: $\|x^{(\nu+1)} - x^*\| \leq \alpha \|x^{(\nu)} - x^*\|$ — linearly convergent algorithm

algorithm 2: $\|x^{(\nu+1)} - x^*\| \leq C \|x^{(\nu)} - x^*\|^2$ — quadratically convergent algorithm

case 1: $\|x^{(\nu)} - x^*\| \leq \alpha \|x^{(\nu-1)} - x^*\| \leq \alpha^2 \|x^{(\nu-2)} - x^*\|$

$\leq \cdots \leq \alpha^\nu \|x^{(0)} - x^*\|$

expect the number of correct digits in the result to grow linearly with the number of iterations ν.

case 2: $e^{(\nu)} = x^{(\nu)} - x^*$

$\|e^{(1)}\| \leq C \|e^{(0)}\|^2$

$\|e^{(2)}\| \leq C \|e^{(1)}\|^2 \leq C^3 \|e^{(0)}\|^4$

$\|e^{(3)}\| \leq C \|e^{(2)}\|^2 \leq C^7 \|e^{(0)}\|^8$

$\vdots$

$\|e^{(\nu)}\| \leq C^{2^\nu - 1} \|e^{(0)}\|^{2^\nu} \leq \left(C\|e^{(0)}\|\right)^{2^\nu - 1} \|e^{(0)}\|$

$\leq \alpha^{2^\nu - 1} \|e^{(0)}\|$

↑ $\|e^{(0)}\| < \rho$, $\rho C = \alpha < 1$

$\alpha = 0.9$

ν	α^ν	$\alpha^{2^\nu - 1}$
1	.9	.9
2	.81	.729
4	.656	.206
8	.430	2.15×10^{-12}

so we expect the number of correct digits to grow exponentially with the number of iterations ν (as long as we start close enough so $\|e^{(0)}\| < \rho$)

implementing predictor-corrector methods

$$y_{n+1}^{(0)} = y_n + h\sum_{j=1}^{s} \tilde{b}_j f_{n+1-j}$$ ← predictor: s-step Adams-Bashforth

$$y_{n+1} - hb_0 f(t_{n+1}, y_{n+1}) = \underbrace{y_n + h\sum_{j=1}^{s} b_j f_{n+1-j}}_{u}$$ corrector: s-step Adams-Moulton

requires implicit solve.

option 1: use Newton's method with $F(y) = y - hb_0 f(t_{n+1}, y) - u$

$$DF(y) = I - hb_0 D_y f(t_{n+1}, y)$$

$y^{(0)}$ = predictor

option 2: use fixed point iteration

$$y^{(\nu+1)} = hb_0 f(t_{n+1}, y^{(\nu)}) + u$$

($y^{(\nu)}$: lags behind by one iteration)

exact solution satisfies

$$y^* = hb_0 f(t_{n+1}, y^*) + u$$

so

$$\|y^{(\nu+1)} - y^*\| \le \underbrace{h|b_0|L}_{\alpha} \|y^{(\nu)} - y^*\| \le \alpha^{\nu+1} \underbrace{\|y^{(0)} - y^*\|}_{\text{error in predictor}}$$

pros & cons: (1) Newton requires the Jacobian DF. could be expensive to compute...

(2) fixed point iteration easier to implement

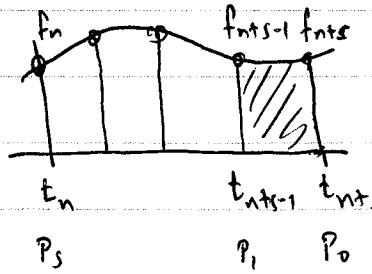
but could require many more iterations (but each iteration is cheaper)

(3) Sometimes we want to take bigger steps than $\alpha = h|b_0|L < 1$ allows. With a good initial guess, Newton will still work while fixed point iteration won't.

In lecture 5, we obtained Adams-Bashforth and Adams-Moulton methods by approximating the right hand side of

$$y' = f(t, y(t))$$

by polynomial interpolation and integrating:



$$y_{n+s} - y_{n+s-1} = \int_{t_{n+s-1}}^{t_{n+s}} f(t, y(t))\, dt$$

$$\approx \int_{t_{n+s-1}}^{t_{n+s}} \sum_{j=0}^{s} P_j(t)\, f_{n+s-j}\, dt$$

$$= h \sum_{j=0}^{s} b_j f_{n+s-j}, \quad b_j = \frac{1}{h} \int_{t_{n+s-1}}^{t_{n+s}} P_j(t)\, dt$$

$$P_m(t_{n+s-j}) = \delta_{mj} = \begin{cases} 1 & m = j \\ 0 & \text{o.w.} \end{cases}$$

Today we're going to study the consistency (i.e. order) of a general multistep method. We'll see that the Adams methods are in some sense optimal and find an algebraic alternative to the above geometric approach to computing the coefficients a_j, b_j.

<u>Setup:</u> $s \geq 1$ is the number of steps.

scheme: $y_0, \ldots, y_{s-1}$ given

$$a_0 y_{n+s} + \cdots + a_s y_n = h\left[b_0 f_{n+s} + \cdots + b_s f_n \right]$$

As in the Euler and Trapezoidal Rule methods, it's useful to define the local truncation error as what's left over when you plug the exact soln into the scheme:

$$\tau_n = a_0 y(t_{n+s}) + \cdots + a_s y(t_n) - h\left[b_0 \underbrace{f(t_{n+s}, y(t_{n+s}))}_{y'(t_{n+s})} + \cdots + b_s \underbrace{f(t_n, y(t_n))}_{y'(t_n)}\right]$$

y is the exact solution, so it satisfies $y' = f(t,y)$

Now we Taylor expand about $t = t_n$:

$$y(t_{n+j}) = y(t_n) + \underbrace{(jh)}_{t_{n+j} - t_n} y'(t_n) + \cdots + \frac{1}{p!}(jh)^p y^{(p)}(t_n) + O(h^{p+1})$$

$$h\,y'(t_{n+j}) = h\,y'(t_n) + (h)(jh)y''(t_n) + \cdots + \frac{h}{(p-1)!}(jh)^{p-1} y^{(p)}(t_n) + O(h^{p+1})$$

plugging these into

$$\tau_n = \sum_{j=0}^{s} \left[a_j y(t_{n+s-j}) - h b_j y'(t_{n+s-j})\right]$$

and matching like powers of h, we obtain

$$\tau_n = c_0 y(t_n) + c_1 y'(t_n) h + c_2 y''(t_n) h^2 + \cdots + c_p y^{(p)}(t_n) h^p + O(h^{p+1})$$

where $c_0 = \sum_{j=0}^{s} a_j$

$$c_1 = \sum_{j=0}^{s} \left[(s-j) a_j - b_j \right]$$

$$c_m = \sum_{j=0}^{s} \left[\frac{(s-j)^m}{m!} a_j - \frac{(s-j)^{m-1}}{(m-1)!} b_j \right] \quad m = 2 \ldots p$$

and the $O(h^{p+1})$ term is <u>exactly</u> equal to

$$\sum_{j=0}^{s} \left[a_j \int_{t_n}^{t_{n+s-j}} y^{(p+1)}(t) \frac{(t_{n+s-j} - t)^p}{p!} dt - h b_j \int_{t_n}^{t_{n+s-j}} y^{(p+1)}(t) \frac{(t_{n+s-j} - t)^{p-1}}{(p-1)!} dt \right]$$

Thus, the method is consistent of order p iff

$$c_0 = c_1 = \cdots = c_p = 0$$

In that case $\tau = O(h^{p+1})$ is rigorously bounded by

$$\|\tau_n\| \le \sum_{j=0}^{s} \left[|a_j| \int_{t_n}^{t_{n+s-j}} M \frac{(t_{n+s-j} - t)^p}{p!} dt + h |b_j| \int_{t_n}^{t_{n+s-j}} M \frac{(t_{n+s-j} - t)^{p-1}}{(p-1)!} dt \right]$$

$$= \sum_{j=0}^{s} \left[\frac{(s-j)^{p+1}}{(p+1)!} |a_j| + \frac{(s-j)^p}{p!} |b_j| \right] M h^{p+1}$$

where $M = \max_{0 \le t \le T} \|y^{(p+1)}(t)\|$

so as usual, $\|\tau_n\| \le C h^{p+1}$ with C depending on y through its $(p+1)$st derivative

Note: the constants c_m depend only on the coefficients $a_0 \dots a_s$, $b_0 \dots b_s$ and not on which equation we're solving. In particular, if they're zero for any equation, then they're zero for all equations.

Let's see what happens for the ODE

$$y' = y \xrightarrow{\text{exact}} y(t) = e^t$$
$$y(0) = 1$$

the truncation error is

$$\tau_n = a_0 e^{(n+s)h} + a_1 e^{(n+s-1)h} + \dots + a_s e^{nh} - h\left[b_0 e^{(n+s)h} + \dots + b_s e^{nh}\right]$$

$$= e^{nh}\left[a_0 e^{sh} + a_1 e^{(s-1)h} + \dots + a_s e^{0h} - h\left(b_0 e^{sh} + b_1 e^{(s-1)h} + \dots + b_s e^{0h}\right)\right]$$

$$= e^{nh}\left[\rho(e^h) - h\sigma(e^h)\right]$$

where

$$\rho(z) = a_0 z^s + a_1 z^{s-1} + \dots + a_{s-1} z + a_s$$
$$\sigma(z) = b_0 z^s + b_1 z^{s-1} + \dots + b_{s-1} z + b_s$$

are the polynomials associated with the multistep method.

Now, the previous expansion

$$\tau_n = c_0 y(t_n) + c_1 h y'(t_n) + \cdots + c_p h^p y^{(p)}(t_n) + O(h^{p+1})$$
$$= e^{t_n}\left[c_0 + c_1 h + \cdots + c_p h^p\right] + O(h^{p+1})$$

is also valid for this ODE. Comparing the two formulas for τ_n, we learn that

$$e^{nh}\left[\rho(e^h) - h\sigma(e^h)\right] = e^{nh}\left[c_0 + \cdots + c_p h^p\right] + O(h^{p+1})$$

dividing through by e^{nh} will not modify the $O(h^{p+1})$ form of the remainder since

$$1 \le e^{nh} \le e^T \qquad \text{(bounds indep of } h\text{)}$$

conclusion:

$$\rho(e^h) - h\sigma(e^h) = c_0 + \cdots + c_p h^p + O(h^{p+1})$$

$$\underbrace{c_0 = c_1 = \cdots = c_p = 0}_{\text{condition that the scheme is consistent of order } p \text{ (i.e. } \|\tau\| = O(h^{p+1}))} \iff \underbrace{\rho(e^h) - h\sigma(e^h) = O(h^{p+1})}_{\text{equivalent condition (easier to check)}}$$

As $h \to 0$, $e^h \to 1$, so the behavior of the polynomials $\rho(z)$, $\sigma(z)$ at $z=1$ tells us everything about the order of the method.

The substitution $z = e^h = 1 + h + \frac{h^2}{2} + \cdots$

$$h = \ln z = \ln(1 + z - 1) = (z-1) - \frac{(z-1)^2}{2} + \frac{(z-1)^3}{3} \cdots$$

yields the equivalent condition

$$\rho(z) - \ln z \, \sigma(z) = O(|z-1|^{p+1})$$

for a multistep method to be order p. (if and only if)

Examples:

Euler: $y_{n+1} - y_n = h f_n$

$\rho(z) = z - 1$

$\sigma(z) = 1$

$z = 1 + \zeta$

$$\rho(z) - (\ln z)\sigma(z) = z - 1 - \ln z = \zeta - \ln(1+\zeta)$$

$$= \zeta - \left(\zeta - \frac{\zeta^2}{2} + \frac{\zeta^3}{3} - \cdots\right) = \frac{1}{2}\zeta^2 + \cdots$$

$$= O(|\zeta|^2)$$

$\Rightarrow$ first order

3-step Adams-Bashforth

$$y_{n+3} - y_{n+2} = h\left[\frac{23}{12} f_{n+2} - \frac{16}{12} f_{n+1} + \frac{5}{12} f_n\right]$$

$$\rho(z) = z^3 - z^2$$

$$\sigma(z) = \frac{23}{12} z^2 - \frac{16}{12} z + \frac{5}{12}$$

$$\rho(z) - \ln z\, \sigma(z) = (1+\zeta)^3 - (1+\zeta)^2 - \left(\zeta - \frac{\zeta^2}{2} + \frac{\zeta^3}{3} - \frac{\zeta^4}{4} + \cdots\right)\left(\frac{23}{12}(1+\zeta)^2 - \frac{16}{12}(1+\zeta) + \frac{5}{12}\right)$$

($z = 1+\zeta$)

mathematica

$$= \frac{3}{8}\zeta^4 + \cdots = O(|\zeta|^4)$$

$\longrightarrow$ 3rd order.

Note: $\sigma(z) = \dfrac{\rho(z)}{\ln z} + O(|z-1|^s)$ can be used to generate the coefficients b_j

2-step Adams-Moulton

$$y_{n+2} - y_{n+1} = h\left[\frac{5}{12} f_{n+2} + \frac{8}{12} f_{n+1} - \frac{1}{12} f_n\right]$$

$$\rho(z) = z^2 - z$$

$$\sigma(z) = \frac{5}{12} z^2 + \frac{8}{12} z - \frac{1}{12}$$

$$\rho(z) - (\ln z)\sigma(z) = -\frac{\zeta^4}{24} + \cdots = O(|\zeta|^4)$$

$\longrightarrow$ 3rd order. ($\sigma(z) = \dfrac{\rho(z)}{\ln z} + O(|z-1|^{s+1})$ can be used to generate the b_j)

Dahlquist barrier theorems

an s-step method has $2s+1$ parameters $(a_1, \dots, a_s, b_0, \dots, b_s)$ that we can play with to try to maximize the order p in

$$\rho(z) - \ln z \, \sigma(z) = O(|z-1|^{p+1})$$

it turns out that for any $p \leq 2s$ there exist parameters that do this, but the resulting scheme is unstable unless

$p \leq s+2$ if s is even
$p \leq s+1$ if s is odd
$p \leq s$ if the scheme is explicit ($b_0 = 0$)

the Milne method

$$y_{n+1} - y_{n-1} = h\left[\frac{1}{3} f_{n+1} + \frac{4}{3} f_n + \frac{1}{3} f_{n-1}\right]$$

is a stable 2-step method of order 4.

A BDF method is an s-order s-step method such that $\sigma(z) = b_0 z^s$ for some number b_0. One may show

$$b_0 = \left(\sum_{m=1}^{s} \frac{1}{m}\right)^{-1} \quad \text{and} \quad \rho(z) = b_0 \sum_{m=1}^{s} \frac{1}{m} z^{s-m} (z-1)^m$$

They work well for stiff problems. (stable iff $s \leq 6$)

beginning of class: discuss HW1

Last time: the multistep method

$$a_0 y_{n+s} + \cdots + a_s y_n = h\left[b_0 f_{n+s} + \cdots + b_s f_n \right]$$

is consistent of order p (i.e. $\tau_n = O(h^{p+1})$) iff

$$\rho(z) - \ln z\, \sigma(z) = O(|z-1|^{p+1})$$

where ρ, σ are the polynomials

$$\rho(z) = a_0 z^s + a_1 z^{s-1} + \cdots + a_{s-1} z + a_s$$
$$\sigma(z) = b_0 z^s + b_1 z^{s-1} + \cdots + b_{s-1} z + b_s$$

associated with the scheme.

These conditions are easier to check (and much easier to remember!) than the equivalent conditions

$$c_0 = c_1 = \cdots = c_p = 0$$

where

$$\tau_n = c_0 y(t_n) + c_1 y'(t_n) h + \cdots + c_p y^{(p)}(t_n) h^p + O(h^{p+1})$$

i.e. $c_0 = \sum_{j=0}^{s} a_j$, $\quad c_1 = \sum_{j=0}^{s} \left[(s-j) a_j - b_j \right]$

$$c_m = \sum_{j=0}^{s} \left[\frac{(s-j)^m}{m!} a_j - \frac{(s-j)^{m-1}}{(m-1)!} b_j \right] \qquad m = 2, \ldots, p$$

note that the coefficients of the scheme appear in ρ, σ when they are expressed in terms of z, while the order condition depends on their behavior when expanded around $z=1$

$$\tilde{\rho}(\zeta) = \rho(1+\zeta), \quad \tilde{\sigma}(\zeta) = \sigma(1+\zeta) \qquad z = 1+\zeta$$

requirement: $\tilde{\rho}(\zeta) - \ln(1+\zeta)\tilde{\sigma}(\zeta) = O(|\zeta|^{p+1})$

Example: find the coefficients b_j of the 2 step Milne method

$$y_{n+2} - y_n = h \sum_{j=0}^{2} b_j f_{n+2-j}$$

solution: $\rho(z) = z^2 - 1 \Rightarrow \tilde{\rho}(\zeta) = (1+\zeta)^2 - 1 = \zeta^2 + 2\zeta$

want $\tilde{\sigma}(\zeta) = \dfrac{\tilde{\rho}(\zeta)}{\ln(1+\zeta)} = (\zeta^2+2\zeta)\underbrace{\left(\frac{1}{\zeta} + \frac{1}{2} - \frac{\zeta}{12} + \frac{\zeta^2}{24} + \cdots\right)}_{\text{expansion of } \frac{1}{\log(1+\zeta)}}$

$$= \underbrace{2 + 2\zeta + \frac{\zeta^2}{3}}_{\text{we have to truncate here or else } \sigma \text{ will not be a polynomial of degree} \le s = 2} - \frac{\zeta^4}{90} + \cdots$$

but we get lucky with this scheme in that the 3rd order term involving ζ^3 is zero. For this reason, the scheme is 4th order instead of 3rd

finally, $\sigma(z) = \tilde{\sigma}(z-1) = 2 + 2(z-1) + \dfrac{(z-1)^2}{3}$

$$= \underbrace{\frac{1}{3}}_{b_0} z^2 + \underbrace{\frac{4}{3}}_{b_1} z + \underbrace{\frac{1}{3}}_{b_2}$$

$\tilde{\rho}(\zeta) - \ln(1+\zeta)\tilde{\sigma}(\zeta) = O(|\zeta|^5)$

Stability

it turns out that the polynomials ρ, σ also determine whether the scheme is stable.

Theorem: An s-step method is stable iff ρ satisfies the root condition: all roots $z_1, \dots, z_m$ of $\rho(z) = 0$ satisfy $|z_j| \leq 1$, and those with $|z_j| = 1$ are simple (i.e. not repeated) roots.

remark: z_j is a simple root iff $\rho(z_j) = 0$, $\rho'(z_j) \neq 0$

this is clear from the formula $\rho(z) = (z - z_1)^{\mu_1}(z - z_2)^{\mu_2} \cdots (z - z_m)^{\mu_m}$

μ_j = multiplicity of root z_j (so $\mu_1 + \mu_2 + \cdots + \mu_m = s$) ↖ $a_0 = 1$

Example: every Adams method is of the form

$$y_{n+s} - y_{n+s-1} = h \sum_{j=0}^{s} b_j f_{n+s-j}$$

↑ $a_0 = 1$ ↑ $a_1 = -1$

so $\rho(z) = z^s - z^{s-1} = z^{s-1}(z - 1)$

↑ multiple root at the origin ↑ simple root on unit circle

∴ root condition is satisfied. ∴ Method is stable.

Note: $z = 1$ is always a root of $\rho(z)$ (i.e. $\rho(1) = 0$)

reason: $\rho(z) - \ln z \, \sigma(z) = \rho(1) + O(|z-1|)$ ← first term in expansion.

the proof of the stability theorem is considerably more complicated than the 1-step case. It starts out the same:

define the error $e_n = y_n - y(t_n)$ and obtain a recursion for e_n:

$$a_0 y_{n+s} + \cdots + a_s y_n = h\left[b_0 f(t_{n+s}, y_{n+s}) + \cdots + b_s f(t_n, y_n)\right]$$

$$a_0 y(t_{n+s}) + \cdots + a_s y(t_n) = h\left[b_0 f(t_{n+1}, y(t_{n+s})) + \cdots + b_s f(t_n, y(t_n))\right] + \tau_n$$

subtract:

$$\circledast \quad a_0 e_{n+s} + \cdots + a_s e_n = h\left[b_0 g_{n+s} + \cdots + b_s g_n\right] + \tau_n$$

where $g_n = f(t_n, y_n) - f(t_n, y(t_n))$

satisfies $\|g_n\| \le L \|y_n - y(t_n)\| = L\|e_n\|$

and $\|\tau_n\| = O(h^{p+1})$ with a constant involving $\max_{0 \le t \le T} \|y^{(p+1)}(t)\|$

3 steps to analyzing the difference eqn. $\circledast$:

1) solve the homogeneous problem (RHS = 0)
2) solve the inhomogeneous problem when the RHS is known in advance (Duhamel's principle, a.k.a. variation of parameters)
3) solve the inhomogeneous problem when the RHS involves the solution (discrete Gronwall inequality)

s-step scalar homogeneous difference equations

$$\circledast \qquad a_0 u_{n+s} + \cdots + a_s u_n = 0$$

$u_0, u_1, \ldots, u_{s-1}$ given ← so solution space is s-dimensional (a solution is an infinite sequence of numbers)

Question: under what conditions will the solution remain bounded for all $n \geq 0$?

idea: seek solutions of the form $u_n = r^n$ where r is a root of the polynomial

$$\rho(r) = a_0 r^s + \cdots + a_s = a_0 (r-r_1)^{\mu_1} (r-r_2)^{\mu_2} \cdots (r-r_m)^{\mu_m}$$

($r_1, r_2, \ldots, r_m$: distinct roots of ρ)

claim: a basis for the solution space of the difference equation $\circledast$ consists of m pure power solutions

$$u_n = r_j^n \qquad (j = 1, \ldots, m) \qquad\qquad k = 0$$

together with the secular (or parasitic) solutions

$$u_n = \frac{1}{k!} \frac{d^k}{dr^k} \Big|_{r=r_j} r^n = \begin{cases} 0 & n < k \\ \binom{n}{k} r_j^{n-k} & n \geq k \end{cases} \qquad 1 \leq k \leq \mu_j - 1$$

for example, if $\mu_j = 4$, the four solutions associated with the root r_j are

	$n=0$	1	2	3	4	5 ...	
$k=0$:	1	r_j	r_j^2	r_j^3	r_j^4	$r_j^5 \ldots$	↘ differentiate
$k=1$:	0	1	$2r_j$	$3r_j^2$	$4r_j^3$	$5r_j^4 \ldots$	↘ diff, divide by 2
$k=2$:	0	0	1	$3r_j$	$6r_j^2$	$10r_j^3 \ldots$	↘ diff, divide by 3
$k=3$:	0	0	0	1	$4r_j$	$10r_j^2 \ldots$	↘

Example: the Fibonacci equation

$$u_{n+2} = u_{n+1} + u_n \quad , \quad u_0 = 1, \quad u_1 = 1$$

we have $\rho(r) = r^2 - r - 1$

so $r_1 = \frac{1+\sqrt{5}}{2} \approx 1.618$, $r_2 = \frac{1-\sqrt{5}}{2} \approx -.618$

we conclude that the general solution is

$$u_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

the initial conditions are

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ r_1 & r_2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}$$

which gives $\begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \frac{1}{r_2 - r_1} \begin{pmatrix} r_2 & -1 \\ -r_1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} r_1 \\ -r_2 \end{pmatrix}$

$r_2 - r_1 = -\sqrt{5}$, $r_2 - 1 = -r_1$, $1 - r_1 = r_2$

$$u_n = \frac{1}{\sqrt{5}} r_1^{n+1} - \frac{1}{\sqrt{5}} r_2^{n+1} \qquad n = 0, 1, 2, 3, \ldots$$

$= (1, 1, 2, 3, 5, 8, 13, \ldots)$ eventually grows like 1.618^{n+1}

check: plugging in $u_n = \frac{1}{k!}\frac{d^k}{dr^k}\Big|_{r=r_j} r^n$ gives

$$a_0 u_{n+s} + \cdots + a_s u_n = \frac{1}{k!}\frac{d^k}{dr^k}\Big|_{r_j}\left(a_0 r^{n+s} + \cdots + a_s r^n\right)$$

$$= \frac{1}{k!}\frac{d^k}{dr^k}\Big|_{r_j}\left(r^n \rho(r)\right) = \circledast$$

since r_j is a root of ρ of multiplicity μ_j, there is a polynomial $q(r)$ s.t.

$$r^n \rho(r) = (r - r_j)^{\mu_j} q(r)$$

thus for $k = 0, \ldots, \mu_j - 1$ we have

$$\circledast = \frac{1}{k!}\sum_{\ell=0}^{k}\binom{k}{\ell}\underbrace{\left[\frac{d^\ell}{dr^\ell}\Big|_{r_j}(r-r_j)^{\mu_j}\right]}_{0 \text{ since } \ell < \mu_j}\left[\frac{d^{k-\ell}}{dr^{k-\ell}}\Big|_{r_j} q(r)\right] = 0$$

so u_n satisfies the difference equation.

$=$

we have just enumerated s solutions

$$u_n^{(jk)} = \frac{1}{k!}\frac{d^k}{dr^k}\Big|_{r=r_j} r^n \qquad \begin{array}{l} 1 \le j \le m \\ 0 \le k \le \mu_j - 1 \end{array}$$

of an s-dimensional solution space. As long as they are linearly independent, this is all of them.

linear independence: let $u_n = \sum_{jk} \alpha_{jk} u_n^{(jk)}$

The initial conditions of this linear combination satisfy

$$\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{s-1} \end{pmatrix} = \underbrace{\begin{pmatrix} R_1 & R_2 & \cdots & R_m \end{pmatrix}}_{\text{generalized Vandermonde matrix } V} \begin{pmatrix} \alpha_{10} \\ \vdots \\ \alpha_{m,\mu_m-1} \end{pmatrix}$$

← sometimes also called confluent Vandermonde matrix if you drop the $k!$ here

here R_j is the $s \times \mu_j$ matrix $(R_j)_{nk} = \frac{1}{k!} \frac{d^k}{dr^k} \Big|_{r=r_j} r^n$ $\quad 0 \le n \le s-1$, $0 \le k \le \mu_j - 1$

e.g. if $s=4$ and $\mu_j = 3$ we have

$$R_j = \begin{pmatrix} 1 & 0 & 0 \\ r_j & 1 & 0 \\ r_j^2 & 2r_j & 1 \\ r_j^3 & 3r_j^2 & 3r_j \end{pmatrix}$$

One can show (see link on course webpage) that

$$\det(V) = \prod_{1 \le i < j \le m} (r_j - r_i)^{\mu_i \mu_j} \neq 0$$

so V is invertible and these solutions are linearly independent.
(any initial condition can be represented with appropriately chosen α_{jk})

remark: $u_n = n^k r_j^n$ $\quad 1 \le j \le m$, $0 \le k \le \mu_j - 1$

is also an acceptable basis if $r_j \neq 0$ or $\mu_j = 1$.

228A Lec 9

Stability of multistep schemes, continued

scheme: $a_0 y_{n+s} + \cdots + a_s y_n = h\left[b_0 f_{n+s} + \cdots + b_s f_n\right]$

polynomials: $\rho(z) = a_0 z^s + \cdots + a_s$, $\sigma(z) = b_0 z^s + \cdots + b_s$

<u>theorem:</u> the scheme is stable iff ρ satisfies the root condition: all roots z_j of $\rho(z)$ satisfy $|z_j| \le 1$ and those with $|z_j| = 1$ are simple (i.e. not repeated) roots.

→ finish lecture 8

<u>conclusion</u> the basis functions $u_n^{(j,k)} = \begin{cases} 0 & n < k \\ \binom{n}{k} r_j^{n-k} & n \ge k \end{cases}$ $\quad 1 \le j \le m$, $0 \le k \le \mu_j - 1$

all remain bounded as $n \to \infty$ iff ρ satisfies the root condition.

note that for fixed k, $\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!} \sim n^k$

so $\binom{n}{k} r_j^{n-k} \sim n^k r_j^n \to 0$ as $n \to \infty$ if $|r_j| < 1$

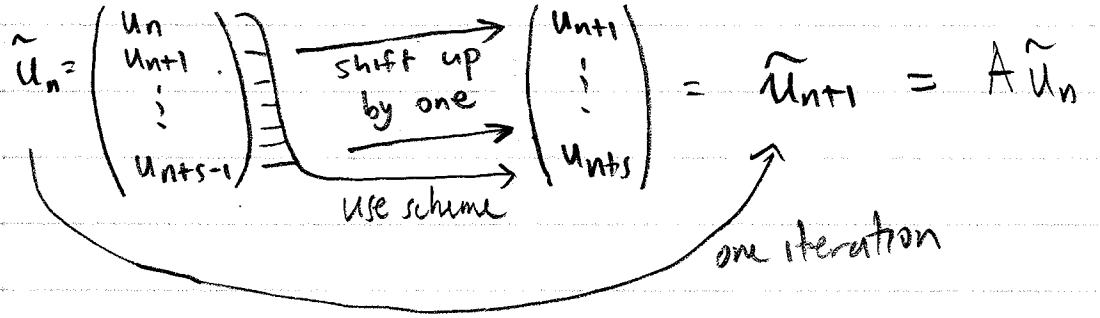
(polynomial growth is eventually dominated by exponential decay:

$$\log n^k r^n = k \log n + n \log r = nk\left(\frac{\log n}{n} + \frac{\log r}{k}\right)$$

$$\le nk\left(\frac{\log r}{2k}\right) \to -\infty \text{ as } n \to \infty$$

↑ if n large enough that $\frac{\log n}{n} < \frac{1}{2}\left|\frac{\log r}{k}\right|$

As with differential equations, an s-step difference equation can be transformed into a first order matrix iteration:



$$A = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_s & \cdots & -a_2 & -a_1 \end{pmatrix}$$ ← shift up by 1

← scalar difference equation (assume $a_0 = 1$)

$$u_{n+s} + a_1 u_{n+s-1} + \cdots + a_s u_n = 0$$

companion matrix of the polynomial $\rho(r)$

The generalized Vandermonde matrix V comes as close as possible to diagonalizing A: it puts A in Jordan canonical form:

$$V^{-1}AV = \underbrace{\begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_m \end{pmatrix}}_{J} \qquad J_j = \begin{pmatrix} r_j & 1 & 0 & 0 \\ 0 & r_j & 1 & 0 \\ 0 & 0 & r_j & 1 \\ 0 & 0 & 0 & r_j \end{pmatrix} \leftarrow M_j = 4 \text{ example}$$

Note that $\det(\lambda I - A) = \det(\lambda I - V^{-1}AV)$

$$= (\lambda - r_1)^{M_1} \cdots (\lambda - r_m)^{M_m} = \rho(\lambda)$$

so ρ is the characteristic determinant of A.

Jordan blocks correspond to subspaces of eigenvectors and <u>associated vectors</u> — parasites

$$V^{-1}AV = J$$

$$\Leftrightarrow \quad AV = VJ$$

$$\Leftrightarrow \quad A(R_1,\dots,R_m) = (R_1,\dots,R_m)\begin{pmatrix} J_1 & & \\ & \ddots & \\ & & J_m \end{pmatrix}$$

$$\Leftrightarrow \quad AR_j = R_j J_j \qquad 1 \le j \le J$$

Let's drop the subscript j and write $R = (w_0,\dots,w_{\mu-1})$ (columns of R)

Then $AR = RJ$, $J = \begin{pmatrix} r & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & r \end{pmatrix}$

means $A(w_0,\dots,w_{\mu-1}) = (w_0,\dots,w_{\mu-1})J$ (what right multiplication by J does)

$$= (rw_0, w_0 + rw_1, w_1 + rw_2, \dots, w_{\mu-2} + rw_{\mu-1})$$

or $Aw_0 = rw_0$ ← genuine eigenvector

$Aw_1 = w_0 + rw_1$ ← first associated vector

rw_1: tries to be an eigenvector

w_0: needs w_0 to help out

$Aw_2 = w_1 + rw_2$ ← second associated vector

⋮

this is the meaning of Jordan canonical form in general.

In our case, $W_0 = \begin{pmatrix} 1 \\ r \\ \vdots \\ r^{s-1} \end{pmatrix}$, $W_1 = \begin{pmatrix} 0 \\ 1 \\ 2r \\ \vdots \\ (s-1)r^{s-2} \end{pmatrix}$, $(W_k)_n = \begin{cases} 0 & 0 \le n < k \\ \binom{n}{k} r^{n-k} & k \le n \le s-1 \end{cases}$

and we can check that

$$AW_0 = \begin{pmatrix} r \\ r^2 \\ \vdots \\ r^s \end{pmatrix} = rW_0, \quad AW_1 = \begin{pmatrix} 1 \\ 2r \\ \vdots \\ sr^{s-1} \end{pmatrix} = W_0 + rW_1$$

<u>for $1 \le k \le \mu - 1$:</u>

$$(AW_k)_n = \left.\begin{cases} 0 & 0 \le n < k-1 \\ \binom{n+1}{k} r^{n+1-k} & k-1 \le n \le s-1 \end{cases}\right\} = (W_{k-1} + rW_k)_n$$

Pascal's triangle

$$\binom{n+1}{k} = \begin{cases} \binom{n}{k-1} & n = k-1 \\ \binom{n}{k-1} + \binom{n}{k} & n > k-1 \end{cases}$$

so indeed V puts the companion matrix A in Jordan canonical form.

Our matrix iteration $\tilde{u}_{n+1} = A\tilde{u}_n$, $\tilde{u}_n = \begin{pmatrix} u_n \\ \vdots \\ u_{n+s-1} \end{pmatrix}$ implies

$$\tilde{u}_n = A^n \tilde{u}_0 \qquad n = 0, 1, 2, \ldots$$

Thus $\|\tilde{u}_n\| = \|A^n \tilde{u}_0\| \le \|A^n\| \cdot \|\tilde{u}_0\|$

and $A^n = (VJV^{-1})^n = VJ^nV^{-1} = V \begin{pmatrix} J_1^n & & & \\ & J_2^n & & \\ & & \ddots & \\ & & & J_m^n \end{pmatrix} V^{-1}$

so $\|A^n\| \le \underbrace{\|V\| \cdot \|V^{-1}\|}_{\text{condition number of } V.} \cdot \max_{1 \le j \le m} \|J_j^n\|$

(could be big but doesn't depend on n)

So the growth of solutions of matrix iterations boils down to powers of Jordan blocks. (true for any matrix A, not just companion matrices)

write $J_j = r_j I + Z$, $I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$, $Z = \underbrace{\begin{pmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & 0 \end{pmatrix}}_{\mu_j}\Big\}\mu_j$

then $J_j^n = (r_j I + Z)^n = \sum_{k=0}^{n} \binom{n}{k} r_j^{n-k} Z^k$

but Z^k is zero for $k \geq \mu_j$

off-diagonal shifts up

e.g. if $\mu = 4$: $Z = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $Z^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $Z^3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $Z^4 = 0$

$\therefore J_j^n = \sum_{k=0}^{\min(n, \mu_j - 1)} \underbrace{\binom{n}{k} r_j^{n-k}}_{\text{entries along } k\text{th superdiagonal}} Z^k$

$\mu_j = 4$ example:

$$J^2 = \begin{pmatrix} r_j^2 & 2r_j & 1 & 0 \\ 0 & r_j^2 & 2r_j & 1 \\ 0 & 0 & r_j^2 & 2r_j \\ 0 & 0 & 0 & r_j^2 \end{pmatrix}, \quad J^3 = \begin{pmatrix} r_j^3 & 3r_j^2 & 3r_j & 1 \\ 0 & r_j^3 & 3r_j^2 & 3r_j \\ 0 & 0 & r_j^3 & 3r_j^2 \\ 0 & 0 & 0 & r_j^3 \end{pmatrix}, \quad J^7 = \begin{pmatrix} r_j^7 & 7r_j^6 & 21r_j^5 & 35r_j^4 \\ 0 & r_j^7 & 7r_j^6 & 21r_j^5 \\ 0 & 0 & r_j^7 & 7r_j^6 \\ 0 & 0 & 0 & r_j^7 \end{pmatrix}$$

coefficients grow polynomially, entries grow or decay exponentially unless $|r_j| = 1$. (root test same for matrix iterations)

Example: the companion matrix for the Fibonacci equation $u_{n+1} = u_n + u_{n-1}$ is

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Its eigenvalues are $r_1 = \frac{1+\sqrt{5}}{2}$, $r_2 = \frac{1-\sqrt{5}}{2}$

it is diagonalized by $V = \begin{pmatrix} 1 & 1 \\ r_1 & r_2 \end{pmatrix}$, $V^{-1} = \frac{1}{r_2 - r_1}\begin{pmatrix} r_2 & -1 \\ -r_1 & 1 \end{pmatrix}$

So $A^n = V \begin{pmatrix} r_1^n & \\ & r_2^n \end{pmatrix} V^{-1}$

Thus the recurrence $\tilde{u}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\tilde{u}_{n+1} = A\tilde{u}_n$ gives the same result as before:

$$\tilde{u}_n = V \begin{pmatrix} r_1^n & \\ & r_2^n \end{pmatrix} \underbrace{V^{-1}\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\frac{1}{\sqrt{5}}\begin{pmatrix} r_1 \\ r_2 \end{pmatrix}} = \frac{1}{\sqrt{5}}\begin{pmatrix} 1 & 1 \\ r_1 & r_2 \end{pmatrix}\begin{pmatrix} r_1^{n+1} \\ r_2^{n+1} \end{pmatrix} = \begin{pmatrix} u_n \\ u_{n+1} \end{pmatrix}$$

$$u_n = \frac{1}{\sqrt{5}} r_1^{n+1} - \frac{1}{\sqrt{5}} r_2^{n+1}$$

since $r_1 \approx 1.618 > 1$ and $|r_2| \approx |-.618| < 1$, u_n blows up (like r_1^n) for large n.

228A Lec 10

Last time: given a matrix A, the norms $\|A^n\|$ remain bounded for $0 \le n < \infty$ iff all eigenvalues λ_j of A satisfy $|\lambda_j| \le 1$, and those with $|\lambda_j| = 1$ have only 1×1 Jordan blocks associated with them.

We also saw that if A is a companion matrix, i.e. $A = \begin{pmatrix} 0 & 1 & & \\ & & \ddots & \\ & & & 1 \\ -a_s & & & -a_2 - a_1 \end{pmatrix}$

then each eigenvalue corresponds to a single Jordan block of size equal to its multiplicity

$$\left[\text{so} \quad A = UJU^{-1}, \; J = \begin{pmatrix} 1/2 & 1 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{pmatrix} \quad \text{can't happen} \right]$$

we therefore have a complete understanding of the matrix iteration

$$\tilde{u}_{n+1} = A\tilde{u}_n, \quad \tilde{u}_0 \text{ given}$$

namely that $\tilde{u}_n = A^n \tilde{u}_0$ remains bounded iff. the eigenvalues of A (i.e. the roots of ρ) satisfy the root condition. Let's assume from now on that this is so, i.e.

$$\exists K > 0 \text{ s.t. } \|A^n\| \le K \quad \text{for} \quad 0 \le n < \infty$$

Next consider the inhomogeneous case

$$\tilde{u}_{n+1} = A\tilde{u}_n + \tilde{\tau}_n$$

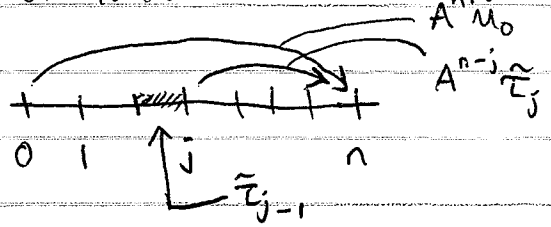
($\tilde{\tau}_n$: a given sequence of vectors $\tilde{\tau}_0, \tilde{\tau}_1, \tilde{\tau}_2, \ldots$)

over the interval $0 \le n \le T/h$

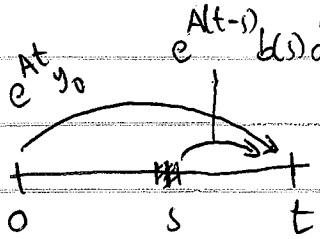
Then
$$\tilde{u}_1 = A\tilde{u}_0 + \tilde{\tau}_0$$
$$\tilde{u}_2 = A^2\tilde{u}_0 + A\tilde{\tau}_0 + \tilde{\tau}_1$$
$$\vdots$$
$$\tilde{u}_n = A^n\tilde{u}_0 + A^{n-1}\tilde{\tau}_0 + \cdots + A\tilde{\tau}_{n-2} + \tilde{\tau}_{n-1}$$

or
$$\tilde{u}_n = A^n\tilde{u}_0 + \sum_{j=1}^{n} A^{n-j}\tilde{\tau}_{j-1}$$

this is a discrete version of the Duhamel principle from ODE's and PDE's: input from the right hand side propagates forward like a new initial condition at a later time.



usual (continuous version): $y' = Ay \longrightarrow y(t) = e^{At}y_0$



$$y' = Ay + b(t) \longrightarrow y(t) = e^{At}y_0 + \int_0^t e^{A(t-s)}b(s)\,ds$$

taking norms, we obtain

$$\|\tilde{u}_n\| \le \|A^n\| \cdot \|\tilde{u}_0\| + \sum_{j=1}^{n} \|A^{n-j}\| \cdot \|\tilde{\tau}_{j-1}\|$$

$$\le K\|\tilde{u}_0\| + K\left(\sum_{j=1}^{n} \|\tilde{\tau}_{j-1}\|\right)$$

$$\le K\|\tilde{u}_0\| + TK \max_{0 \le j \le T/h} \frac{\|\tau_j\|}{h}$$

$nh = t_n \le T$

↑ sum of all the garbage you put in. No error gets amplified by more than K.

So the root condition ensures stability in the case that the right hand side is given.
Finally, consider the case that the RHS depends on the solution:

$$\tilde{e}_{n+1} = A\tilde{e}_n + h\tilde{g}_n + \tilde{\tau}_n$$

here $\tilde{e}_n = \begin{pmatrix} e_n \\ e_{n+1} \\ \vdots \\ e_{n+s-1} \end{pmatrix} = \begin{pmatrix} y_n - y(t_n) \\ y_{n+1} - y(t_{n+1}) \\ \vdots \\ y_{n+s-1} - y(t_{n+s-1}) \end{pmatrix}$ ← but each of these are vectors...

so A is actually a block companion matrix here

$$A = \begin{pmatrix} 0 & I & & \\ & & \ddots & \\ & & & I \\ -a_s I & \cdots & & -a_2 I - a_1 I \end{pmatrix}$$

$0, I$ are $d \times d$

A is $sd \times sd$

by permuting the rows and columns of $\mathbb{A}$ you'll end up with d copies of a plain companion matrix A:

$$\mathbb{P}^{-1}\mathbb{A}\mathbb{P} = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}, \qquad A = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_s & & \cdots & -a_1 \end{pmatrix}$$

$$\mathbb{P}_{d(i-1)+\mathring{i},\, s(\mathring{j}-1)+j} = \delta_{ij}\,\delta_{\mathring{i}\mathring{j}} \qquad \begin{matrix} 1 \le i, j \le s \\ 1 \le \mathring{i}, \mathring{j} \le d \end{matrix}$$

↑ permutation matrix to re-organize blocks $\left(\mathbb{P}^{-1} = \mathbb{P}^T,\ \|\mathbb{P}x\| = \|x\|\right)$

$$\|\mathbb{A}^n\| = \|(\mathbb{P}^{-1}\mathbb{A}\mathbb{P})^n\| = \left\|\begin{pmatrix} A^n & & \\ & \ddots & \\ & & A^n \end{pmatrix}\right\| = \|A^n\| \le K, \quad n = 0, 1, 2, \ldots$$

recall that $\tilde{e}_{n+1} = \mathbb{A}\tilde{e}_n + h\tilde{g}_n + \tilde{\tau}_n$

came from the recursion

$$e_{n+s} + a_1 e_{n+s-1} + \cdots + a_s e_n = h\left[b_0 g_{n+s} + \cdots + b_s g_n\right] + \tau_n$$

$$g_n = f(t_n, y_n) - f(t_n, y(t_n))$$

$$\|g_n\| \le L\|y_n - y(t_n)\| \le L\|e_n\|$$

thus the first $s-1$ blocks of $\tilde{e}_{n+1}$ are still just a shift of the last $s-1$ blocks of $\tilde{e}_n$, and

$$\tilde{g}_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_0 g_{n+s} + \cdots + b_s g_n \end{pmatrix}, \qquad \tilde{\tau}_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \tau_n \end{pmatrix}$$

using the triangle inequality, we find that

$$\|\bar{g}_n\| \le \sum_{k=0}^{s} |b_k| \, \|g_{n+s-k}\| \le \sum_{k=0}^{s} |b_k| L \, \|e_{n+s-k}\|$$

$$\le |b_0| L \underbrace{\|e_{n+s}\|}_{\le \|\tilde{e}_{n+1}\|} + \left(\sum_{k=1}^{s} |b_k|\right) L \underbrace{\|\tilde{e}_n\|}_{\tilde{e}_n = \begin{pmatrix} e_n \\ e_{n+1} \\ \vdots \\ e_{n+s-1} \end{pmatrix}}$$

↑ split off implicit term

so $\|e_{n+s-k}\| \le \|\tilde{e}_n\|$

while $\|\tilde{\tau}_j\| = \|\tau_j\|$

the discrete Duhamel principle gives

$$\|\tilde{e}_n\| \le K \|\tilde{e}_0\| + K \sum_{j=0}^{n-1} \left(h \|\tilde{g}_j\| + \|\tau_j\| \right)$$

now we match up terms:

$$\|\tilde{g}_0\| \le |b_0| L \, \|\tilde{e}_1\| + \left(\sum_{1}^{s} |b_k|\right) L \, \|\tilde{e}_0\|$$

(absorb)

$$\|\tilde{g}_1\| \le |b_0| L \, \|\tilde{e}_2\| + \left(\sum_{1}^{s} |b_k|\right) L \, \|\tilde{e}_1\|$$

$$\vdots$$

(absorb)

$$\|\tilde{g}_{n-1}\| \le |b_0| L \, \|\tilde{e}_n\| + \left(\sum_{1}^{s} |b_k|\right) L \, \|\tilde{e}_{n-1}\|$$

so $$K \sum_{j=0}^{n-1} h \|\tilde{g}_j\| \le \underbrace{h |b_0| L K \|\tilde{e}_n\|}_{\text{move to Left Hand Side}} + h B L K \sum_{j=0}^{n-1} \|\tilde{e}_j\|$$

$B = \sum_{k=0}^{s} |b_k|$

$$\therefore \underbrace{(1 - h|b_0|LK)}_{\text{assume } h \text{ is small enough that this coefficient is } \geq \frac{1}{2}} \|\tilde{e}_n\| \leq K\|\tilde{e}_0\| + hBLK\sum_{j=0}^{n-1}\|\tilde{e}_j\| + K\sum_{j=0}^{n-1}\|\tau_j\|$$

(automatically true for explicit schemes)

Then $\|\tilde{e}_n\| \leq 2(\text{RHS})$

Thus we have bounded $\|\tilde{e}_n\|$ in terms of its previous values and the perturbations $\tilde{\tau}_j$ introduced along the way.

A discrete Gronwall inequality allows us to bound $\|\tilde{e}_n\|$ by $\|\tilde{e}_0\|$ and the $\|\tau_j\|$'s alone.

usual Gronwall inequality: suppose $r(t)$ is continuous and satisfies

$$0 \leq r(t) \leq C + L\int_0^t r(s)\,ds$$

Then $r(t)$ is bounded by

$$r(t) \leq Ce^{Lt}$$

← proof: see any book on ODE theory, e.g. Coddington & Levinson

discrete Gronwall: suppose ε_n satisfies

$$0 \leq \varepsilon_n \leq C + L\sum_{j=0}^{n-1}\varepsilon_j \qquad n = 0, 1, 2, \ldots$$

Then

$$\varepsilon_n \leq Ce^{Ln} \quad \text{for } n = 0, 1, 2, \ldots$$

proof: Let $u_n = C + L\sum_{j=0}^{n-1} \varepsilon_j$ so $\varepsilon_n \le u_n$.

Then $u_0 = C$

and $u_{n+1} - u_n = L\varepsilon_n \le L u_n$

$\Rightarrow u_{n+1} \le (1+L)u_n$

$\therefore \varepsilon_n \le u_n \le (1+L)^n u_0 \le (1+L)^n C$

as usual, $1+L \le 1+L+\frac{L^2}{2}+\cdots = e^L$

so $\varepsilon_n \le C e^{Ln}$ as claimed.

Applying this to $\underbrace{\|\tilde{e}_n\|}_{\text{"}\varepsilon_n\text{"}} \le \underbrace{2K\left(\|\tilde{e}_0\| + \sum_{j=0}^{n-1}\|\tau_j\|\right)}_{\text{"}C\text{"}} + \underbrace{2hBLK}_{\text{"}L\text{"}} \sum_{j=0}^{n-1}\|\tilde{e}_j\|$

We obtain

$$\|\tilde{e}_n\| \le 2K\left(\|\tilde{e}_0\| + \sum_{j=0}^{n-1}\|\tau_j\|\right) e^{2BLKt_n}$$

($t_n = nh$)

this sum is less than $t_n \max_j \frac{\|\tau_j\|}{h}$

so no single error in the initial conditions or truncation error is amplified by more than $2Ke^{2BLKT}$ over the interval $[0,T]$.

$\longrightarrow$ scheme is stable $\longleftarrow$

228A Lec 11

Today we'll finish our discussion of multistep methods

Asymptotic expansion of the error:

recall from Lec 3 that the error in using Euler's method satisfies

$$y_n = y(t_n) + h\,\varepsilon(t_n) + O(h^2)$$

where $\varepsilon(t)$ satisfies the variational equation

$$\varepsilon'(t) = D_y f(t, y(t))\,\varepsilon(t) - \frac{1}{2} y''(t)$$

a similar formula holds for general multistep methods. To derive it, recall from lecture 7 that the truncation error

$$\tau_n = a_0 y(t_{n+s}) + \cdots + a_s y(t_n) - h\left[b_0 f(t_{n+s}, y(t_{n+s})) + \cdots + b_s f(t_n, y(t_n))\right]$$

has the expansion

$$\tau_n = c_0 y(t_n) + c_1 y'(t_n) h + \cdots + c_p y^{(p)}(t_n) h^p + O(h^{p+1})$$

where

$$c_0 = \sum_{j=0}^{s} a_j$$

$$c_m = \sum_{j=0}^{s} \left[\frac{(s-j)^m}{m!} a_j - \frac{(s-j)^{m-1}}{(m-1)!} b_j\right] \qquad m = 2, \ldots, p$$

If the scheme is order p, then $c_0 = c_1 = \cdots = c_p = 0$ and

$$\tau_n = c_{p+1} y^{(p+1)}(t_n) h^{p+1} + O(h^{p+2})$$ ← include one more term than before

We know that the error $e_n = y_n - y(t_n)$ satisfies

$$\circledast \qquad a_0 e_{n+s} + \cdots + a_1 e_n = h\left[b_0 g_{n+s} + \cdots + b_s g_n\right] - \tau_n$$

where $g_{n+j} = f(t_{n+j}, y_{n+j}) - f(t_{n+j}, y(t_{n+j}))$

$$\approx D_y f(t_{n+j}, y(t_{n+j}))\, e_{n+j} + O(\|e_{n+j}\|^2)$$

finally, we note that $y^{(p+1)}(t_n) = y^{(p+1)}(t_{n+j}) + O(h)$

so that

$$\tau_n = c_{p+1} h^{p+1}\left[\frac{b_0 y^{(p+1)}(t_{n+s}) + \cdots + b_s y^{(p+1)}(t_n)}{b_0 + \cdots + b_s}\right] + O(h^{p+2})$$

$$= h^{p+1}\left[b_0\left(\frac{c_{p+1}}{\sigma(1)} y^{(p+1)}(t_{n+s})\right) + \cdots b_s\left(\frac{c_{p+1}}{\sigma(1)} y^{(p+1)}(t_n)\right)\right] + O(h^{p+2})$$

so if we write $e_n = h^p \varepsilon(t_n) + O(h^{p+1})$

we can interpret $\frac{\circledast}{h^p}$ as the multistep method applied to the equation

$$\varepsilon'(t) = D_y f(t, y(t))\,\varepsilon(t) - \frac{c_{p+1}}{\sigma(1)} y^{(p+1)}(t)$$

$$\varepsilon(0) = 0$$

One can now mimic our proof in the Euler case to show that the solution $\varepsilon(t)$ of this variational equation gives the leading order error, i.e.

$$y_n = y(t_n) + h^p \varepsilon(t_n) + O(h^{p+1})$$

↑ numerical solution; ↑ exact solution; ↑ exact soln of variational equation

the constant $\frac{C_{p+1}}{\sigma(1)}$ is known as the error constant of the scheme. It does not change if you multiply the scheme coefficients $a_0 \dots a_s, b_0 \dots b_s$ by a constant (since C_{p+1} and $\sigma(1)$ both scale the same)

For Euler's method: $s=p=1$, $a_0=1$, $a_1=-1$, $b_1=1$

$$C_{p+1} = \sum_{j=0}^{s} \left[\frac{(s-j)^{p+1}}{(p+1)!} a_j - \frac{(s-j)^p}{p!} b_j \right] = \frac{1}{2}(1) + \frac{0}{2}(-1) - \frac{1}{1}(0) - \frac{0}{1}(1)$$

$$= 1/2$$

$$\therefore \sigma(1) = b_0 + b_1 = 1 \qquad \boxed{\frac{C_{p+1}}{\sigma(1)} = 1/2} \text{ as expected}$$

As before, the main use of this analysis is to justify comparing the solution to itself with different mesh-sizes to infer the order of convergence and determine if further refinement is necessary.

Schema	order	error constant (s=4 ↓, s=5 ↓)
Adams-Bashforth	s	$1/2, 5/12, 3/8, \frac{251}{720}, \frac{95}{288}$
Adams-Moulton	s+1	$-1/12, -1/24, \frac{-19}{720}, -3/160, -863/60480$
also called leapfrog → Nystrom (s=2)	2	$1/6$
Nystrom (s>2)	s	$1/6, \frac{29}{180}, \frac{7}{45}$
Milne (s=2)	4	$-1/180$
Milne (s>2)	s+1	$-1/180, -1/180, -\frac{37}{7560}$
BDF	s	$-1/2, -1/3, -1/4, -1/5, -1/6$

all else being equal, a smaller error constant is better.

multiple roots on the unit circle

in a nutshell, our convergence proof for multistep methods was:

① by grouping together adjacent error terms $\tilde{e}_n = \begin{pmatrix} e_n \\ \vdots \\ e_{n+s-1} \end{pmatrix} = \begin{pmatrix} y_n - y(t_n) \\ \vdots \\ y_{n+s-1} - y(t_{n+s-1}) \end{pmatrix}$

we obtain the recurrence

$$\tilde{e}_{n+1} = \mathbb{A}\tilde{e}_n + h\tilde{g}_n + \tilde{\tau}_n, \quad \mathbb{A} = A \otimes I = \begin{pmatrix} 0 & I & & \\ & \ddots & \ddots & \\ & & 0 & I \\ -a_s I & \cdots & -a_2 I & -a_1 I \end{pmatrix}$$

(Kronecker product)

② by the discrete Duhamel principle,

$$\tilde{e}_n = \mathbb{A}^n \tilde{e}_0 + \sum_{j=1}^{n} \mathbb{A}^{n-j} \left(h\tilde{g}_{j-1} + \tilde{\tau}_{j-1} \right)$$

(3) if ρ satisfies the root condition, $\exists K$ s.t. $\|A^n\| \le K \quad \forall n \ge 0$.

(4) from $\|g_j\| \le L\|e_j\|$ it follows that $\beta = \sum_{k=0}^{s} |b_k|$

$$\|\tilde{e}_n\| \le 2K\|e_0\| + 2K\sum_{j=0}^{n-1}\|\tau_j\| + 2hBLK\sum_{j=0}^{n-1}\|\tilde{e}_j\|$$

(5) the discrete Gronwall inequality eliminates the previous errors:

$$0 \le \varepsilon_n \le C + L\sum_{j=0}^{n-1}\varepsilon_j \Rightarrow \varepsilon_n \le Ce^{Ln} \quad (nh)$$

$$\text{so} \quad \|\tilde{e}_n\| \le \left(2K\|e_0\| + 2K\sum_{j=0}^{n-1}\|\tau_j\|\right)e^{2BLKt_n}$$

Now suppose ρ has a double root on the boundary. Then step (3) would become

$$\exists K \text{ s.t. } \|A^n\| \le (n+1)K \quad \forall n \ge 0$$

↑ linear growth due to 2×2 Jordan block

$A = VJV^{-1}$, $J = \begin{pmatrix} * & & & \\ & * & & \\ & & \begin{pmatrix}\lambda & 1\\ & \lambda\end{pmatrix} & \\ & & & * \end{pmatrix}$ ← $|\lambda| = 1$

$\|A = A \otimes I$

$$\begin{pmatrix}\lambda & 1\\ & \lambda\end{pmatrix}^n = \begin{pmatrix}\lambda^n & n\lambda^{n-1}\\ 0 & \lambda^n\end{pmatrix}$$

As a result, step 4 becomes

$$\|\tilde{e}_n\| \le (n+1)K\|\tilde{e}_0\| + K\sum_{j=0}^{n-1}(n-j)\|\tau_j\| + hBLK\sum_{j=0}^{n-1}(n-j)\|\tilde{e}_j\|$$

not a big deal. Just lose an order of accuracy (but scheme could still converge)

problem. discrete Gronwall can't handle the n here.

example 1. the scheme

$$y_{n+3} + y_{n+2} - y_{n+1} - y_n = h\left(\frac{8}{3} f_{n+2} + \frac{2}{3} f_{n+1} + \frac{2}{3} f_n\right)$$

has $\rho(z) = (z-1)(z+1)^2$ ← double root at $z=-1$

and $\rho(z) - \ln z\, \sigma(z) = O(|z-1|^4)$ ← $\tau = O(h^4)$

but if you try it out, it doesn't just degrade to a second order method... the solution blows up like mad.

example 2. the scheme

$$y_{n+3} - 2y_{n+2} + y_{n+1} = h\left[\frac{3}{2} f_{n+2} - 2 f_{n+1} + \frac{1}{2} f_n\right]$$

has $\rho(z) = z^3 - 2z^2 + z = z(z-1)^2$, $\tau = O(h^4)$

and this time the scheme behaves like a second order method (so it converges, but loses an order of accuracy)

Irreducible methods

the reason for the mysterious convergence of example 2 is that

$$\sigma(z) = \frac{3}{2} z^2 - 2z + \frac{1}{2} = (z-1)\left(\frac{3}{2} z - \frac{1}{2}\right)$$

shares a common factor of $z-1$ with $\rho(z)$. Now consider the simpler scheme generated by

$$\tilde{\rho}(z) = z(z-1), \quad \tilde{\sigma}(z) = \frac{3}{2} z - \frac{1}{2}$$

← cancel common factor of $z-1$

Now take the solution of the 3-step scheme and define δ_n to be the amount that it fails to satisfy the 2-step scheme:

$$y_{n+3} - y_{n+2} = h\left(\frac{3}{2}f_{n+2} - \frac{1}{2}f_{n+1}\right) + \delta_{n+1}$$

$$- \left[y_{n+2} - y_{n+1} = h\left(\frac{3}{2}f_{n+1} - \frac{1}{2}f_n\right) + \delta_n\right]$$

$$y_{n+3} - 2y_{n+2} + y_{n+1} = h\left(\frac{3}{2}f_{n+2} - 2f_{n+1} + \frac{1}{2}f_n\right) + \delta_{n+1} - \delta_n$$

the 3-step scheme; $\delta_{n+1} - \delta_n$ must be zero

δ_0 is determined by the initial condition y_2:

y_0 y_1 y_2 ← initial conditions for 3-step scheme

$$\delta_0 = y_2 - y_1 - h\left(\frac{3}{2}f_1 - \frac{1}{2}f_0\right)$$

after that, $\delta_n = \delta_0$ for all $n \geq 0$. (no feedback from solution!)

so now the convergence of the (stable!) two step scheme can be applied to this problem by absorbing δ_n into the truncation error τ_n.

⇒ expect 2nd order convergence ✓

the same trick works any time ρ, σ share a factor of $z - \alpha$ for some $\alpha \in \mathbb{C}$. In that case

$$\delta_{n+1} - \alpha\delta_n = 0 \implies \delta_n = \alpha^n \delta_0$$

So if $|\alpha| \leq 1$, no problem... δ_n remains bounded.
$|\alpha| > 1$, δ_n will grow exponentially, solution will diverge.

Evidently, there is no benefit to using a scheme in which ρ, σ share a common factor (the reduced scheme determines the order of convergence anyway)

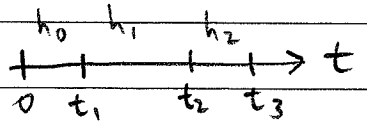
assuming they don't (i.e. the scheme is irreducible), then double roots of ρ on the unit circle lead to catastrophic feedback of the errors and

$$\text{stability} + \text{consistency} \iff \text{convergence}$$

stability ↑ root condition

consistency ↑ $\rho(z) - \ln z\, \sigma(z) = O(|z-1|^{p+1})$, $p \geq 1$

variable stepsize



$$a_{0n} y_{n+s} + \cdots + a_{0s} y_0 = h_{n+s-1}\left[b_{0n} f_{n+s} + \cdots + b_{sn} f_n\right]$$

method is order p if:

$$a_{0n} q(t_{n+s}) + \cdots + a_{sn} q(t_n) = h_{n+s-1}\left[b_{0n} q'(t_{n+s}) + \cdots + b_{sn} q'(t_n)\right]$$

for all polynomials $q(t)$ of degree $\leq p$.

the coefficients can be constructed geometrically as in Lec 5.

error analysis: $\tilde{e}_{n+1} = A_n \tilde{e}_n + h_{n+s-1} \tilde{g}_n + \tilde{\tau}_n$

$\mathbb{A}_n = A_n \otimes I$

$$A_n = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_{sn} & \cdots & -a_{2n} & -a_{1n} \end{pmatrix}$$

stability:

$\|\mathbb{A}_{n+\ell} \cdots \mathbb{A}_{n+1} \mathbb{A}_n\| \leq K$

$\forall n \geq 0,\ \ell \geq 1$

(guaranteed for Adams methods, where $\mathbb{A}_n$ is indep. of n)

Runge-Kutta methods

an RK method is a one step method that uses several f evaluations to advance from t_n to t_{n+1}

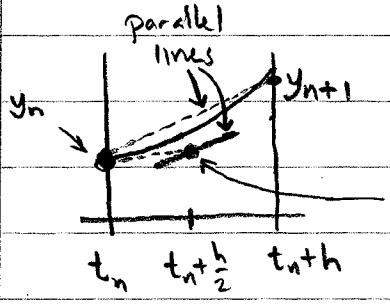
examples: Euler: $k_1 = f(t_n, y_n)$

$$y_{n+1} = y_n + h k_1$$

Runge's 2nd order method: $k_1 = f(t_n, y_n)$

$$k_2 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_1)$$

$$y_{n+1} = y_n + h k_2$$



use Euler to predict midpoint
compute slope at predicted midpoint
use this slope to go from y_n to y_{n+1}

trapezoidal rule with Euler predictor: $k_1 = f(t_n, y_n)$

$$k_2 = f(t_n + h, y_n + h k_1)$$

$$y_{n+1} = y_n + h\left(\tfrac{1}{2}k_1 + \tfrac{1}{2}k_2\right)$$

"the" Runge-Kutta method (RK4) $k_1 = f(t_n, y_n)$

$$k_2 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_1)$$

$$k_3 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_2)$$

$$k_4 = f(t_n + h, y_n + h k_3)$$

$$y_{n+1} = y_n + h\left[\tfrac{1}{6}k_1 + \tfrac{2}{6}k_2 + \tfrac{2}{6}k_3 + \tfrac{1}{6}k_4\right]$$

A general RK method has s stages:

$$k_1 = f\left(t_n + c_1 h,\ y_n + h\left(a_{11}k_1 + \cdots + a_{1s}k_s\right)\right)$$

$$\vdots$$

$$k_s = f\left(t_n + c_s h,\ y_n + h\left(a_{s1}k_1 + \cdots + a_{ss}k_s\right)\right)$$

$$y_{n+1} = y_n + h\left(b_1 k_1 + \cdots + b_s k_s\right)$$

the coefficients are conveniently represented in a Butcher array

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

Examples: Euler

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

Runge's 2nd order:

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

trap. rule with Euler predictor

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

classical RK4

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

if entries on or above the diagonal are non-zero, the method is implicit (IRK):

implicit Euler: $y_{n+1} = y_n + h\underbrace{f(t_n+h, y_{n+1})}$

only one stage, so they must be k_1

in RK notation: $k_1 = f(t_n+h, y_n+hk_1)$

$y_{n+1} = y_n + hk_1$

Butcher array:

1	1
	1

trapezoidal rule: $k_1 = f(t_n, y_n)$

$k_2 = f(t_n+h, y_{n+1}) = f\left(t_n+h, y_n+h\left(\frac{k_1}{2}+\frac{k_2}{2}\right)\right)$

$y_{n+1} = y_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)$

0	0	0
1	1/2	1/2
	1/2	1/2

Butcher theory of attainable order:

theorem: All RK methods of order $p \leq 4$ have the same order whether y is a scalar or a vector. For $p \geq 5$, the order for a system may be lower than for a scalar equation.

theorem: any s-stage ERK method has order $p \leq s$

theorem: An ERK method of order p must have at least:

p+1 stages if $p > 4$, p+2 stages if $p > 6$, p+3 stages if $p > 7$.

highest order method I know of: 10th order, 17 stages

it's a pain to keep writing $f(t,y)$ ... the formulas are about to get very messy. There's a standard trick to reduce a time-dependent ODE to an autonomous system:

$$y' = f(t,y), \quad y(0) = \xi \qquad \longrightarrow \qquad \tilde{y}' = \tilde{f}(\tilde{y}), \quad \tilde{y}(0) = \begin{pmatrix} 0 \\ \xi \end{pmatrix}$$

$$\tilde{y}(t) = \begin{pmatrix} t \\ y(t) \end{pmatrix} \in \mathbb{R}^{d+1} \quad \leftarrow \text{ just add a time component}$$

$$\tilde{f}(\tilde{y}) = \begin{pmatrix} 1 \\ f(\tilde{y}^0, (\tilde{y}^1, \ldots, \tilde{y}^d)^T) \end{pmatrix} = \begin{pmatrix} 1 \\ f(t,y) \end{pmatrix}$$

$\tilde{y}^0$: time; $(\tilde{y}^1, \ldots, \tilde{y}^d)^T$: y in the original formulation

(superscripts denote components of vectors today)

The first equation of the system $\tilde{y}' = \tilde{f}(\tilde{y})$, $\tilde{y}(0) = \begin{pmatrix} 0 \\ \xi \end{pmatrix}$ decouples from the rest:

$$\tilde{y}^{0\prime} = 1, \quad \tilde{y}^0(0) = 0$$

solution: $\tilde{y}^0(t) = t$

The remaining equations of $\tilde{y}' = \tilde{f}(\tilde{y})$, $\tilde{y}(0) = \begin{pmatrix} 0 \\ \xi \end{pmatrix}$ are then equivalent to the original ODE. Thus, if one has a unique solution, so does the other and they agree.

if we apply our RK method to the new problem, we would like to get the same numerical solution. The ith stages are:

original problem: $k_i = f\left(t_n + c_i h,\ y_n + h\left(a_{i1}k_1 + \cdots + a_{is}k_s\right)\right)$

new formulation: $\tilde{k}_i = \tilde{f}\left(\tilde{y}_n + h\left(a_{i1}\tilde{k}_1 + \cdots + a_{is}\tilde{k}_s\right)\right)$

$$= \begin{pmatrix} 1 \\ f\left(\tilde{y}_n^0 + h\left(a_{i1}\tilde{k}_1^0 + \cdots + a_{is}\tilde{k}_s^0\right),\ \tilde{y}_n^* + h\left(a_{i1}\tilde{k}_1^* + \cdots + a_{is}\tilde{k}_s^*\right)\right) \end{pmatrix}$$

the zeroth component of each $\tilde{k}_j$ is 1 → $\tilde{y}_n^0 + (a_{i1} + \cdots + a_{is})h$

$\tilde{y}_n^* = \begin{pmatrix} y_n^1 \\ \vdots \\ y_n^d \end{pmatrix}$

so as long as $\tilde{y}_n^0 = t_n$, $\tilde{y}_n^* = y_n$ and $a_{i1} + \cdots + a_{is} = c_i$

then $\tilde{k}_i = \begin{pmatrix} 1 \\ k_i \end{pmatrix}$

the final update $\tilde{y}_{n+1} = \tilde{y}_n + h\left(b_1\tilde{k}_1 + \cdots + b_s\tilde{k}_s\right)$

yields $\tilde{y}_{n+1}^0 = \tilde{y}_n^0 + h(b_1 + \cdots + b_s) = t_n + h = t_{n+1}$

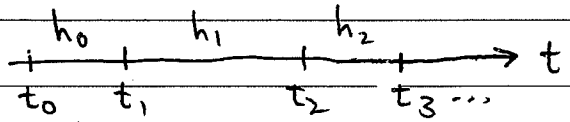
$$\tilde{y}_{n+1}^* = \tilde{y}_n^* + h\left(b_1\tilde{k}_1^* + \cdots + b_s\tilde{k}_s^*\right)$$
$$= y_n + h\left(b_1k_1 + \cdots + b_sk_s\right) = y_{n+1}$$

as long as $b_1 + \cdots + b_s = 1$.

we'll assume $c_i = \sum_{j=1}^{s} a_{ij}$ and $\sum_{j=1}^{s} b_j = 1$ from now on and analyze the autonomous case only.

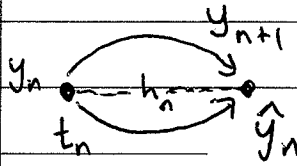
stepsize control. a major advantage of RK methods over multistep methods is that the update from y_n to y_{n+1} does not involve previous values $y_{n-1}, y_{n-2}, \ldots$ or $f_{n-1}, f_{n-2}, \ldots$ so it is easy to

- start the method going at $t=0$
- change stepsize $h_n = t_{n+1} - t_n$



goal = choose h_n as you go so you take big steps where the solution is smooth and small steps where it changes rapidly.

idea: use one scheme to advance the solution from t_n to t_{n+1}
use another scheme to predict the error of this step and decide whether to increase or decrease h.

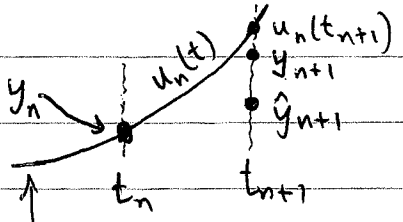


use order p method to compute y_{n+1}
" " $p-1$ " " " $\hat{y}_{n+1}$

we expect that $\hat{y}_{n+1} - y_{n+1}$ is a good estimate of the error of the lower order scheme (i.e. in $\hat{y}_{n+1}$) in advancing from t_n to t_{n+1} with initial condition (at t_n) given by y_n. (we use stepsize control to estimate local errors rather than accumulated global errors all the way from $t=0$)

in other words, if $u_n(t)$ is the exact solution of the ODE

$$u_n'(t) = f(u_n(t))$$
$$u_n(t_n) = y_n$$



exact solution of $y' = f(y)$ which agrees with the numerical soln at time t_n

then we expect

$$\|\hat{y}_{n+1} - u_n(t_{n+1})\| \leq \underbrace{\|\hat{y}_{n+1} - y_{n+1}\|}_{O(h^p) \text{ and available to us}} + \underbrace{\|y_{n+1} - u_n(t_{n+1})\|}_{O(h^{p+1}) \text{ so neglect}}$$

now we make a big leap of faith and assume

$$\|y_{n+1} - u_n(t_{n+1})\| \lessapprox h \|\hat{y}_{n+1} - u_n(t_{n+1})\|$$

↑ approximately less than (has no rigorous meaning)

in other words, we expect the order p method to have an error about h times smaller than the order $p-1$ method and use this as a heuristic for deciding the stepsize.

our goal is to choose steps to keep the rate at which errors are being generated per unit time approximately constant:

$$\text{err} = \text{error rate} = \frac{\|y_{n+1} - u_n(t_{n+1})\|}{h} \lessapprox \|\hat{y}_{n+1} - y_{n+1}\|$$

numerator: error introduced in this step
denominator: how far you got to go for this error

↑ our estimate of the error rate.

we expect $err = \|\hat{y}_{n+1} - y_{n+1}\|$ to satisfy $err \approx Ch^p$

So replacing h by h_{new} would cause err to become

$$err_{new} = (err)\left(\frac{h_{new}}{h}\right)^p = \varepsilon$$

(ε ← what we want (ε is the desired error rate))

so our next timestep should be

$$h_{new} = h_{old} \cdot \left(\frac{\varepsilon}{err}\right)^{\frac{1}{p}}$$

In commercial codes (like matlab), weighted norms are used to try to maintain high relative accuracy of each component:

want $|\hat{y}^i_{n+1} - y^i_{n+1}| \le \varepsilon_i$, $\quad \varepsilon_i = \underbrace{Atol_i}_{\text{absolute tolerance}} + |y^i_n| \cdot \underbrace{Rtol_i}_{\text{relative tolerance}}$

define

a weighted 2-norm → $$err = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\frac{\hat{y}^i_{n+1} - y^i_{n+1}}{\varepsilon_i}\right)^2}$$

($Atol_i$, $Rtol_i$ ← supplied by the user)

want $err_{new} \approx 1$. Should choose $h_{new} = h_{old} \cdot \left(\frac{1}{err}\right)^{1/p}$

to be safe, we choose (2, 1/2, 0.8)

$$h_{new} = h_{old} \cdot \min\left(facmax, \max\left(facmin, fac \cdot \left(\frac{1}{err}\right)^{\frac{1}{p}}\right)\right)$$

If $err \le 1$, accept the step and use the new stepsize for the next step

If $err > 1$, reject the step and try again with the new stepsize

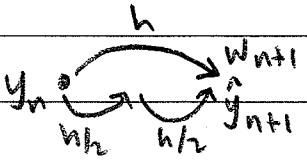
so how should we choose y_{n+1} and $\hat{y}_{n+1}$?

two options: (1) Richardson extrapolation
(2) embedded RK methods ← next week

option 1 - pick your favorite RK scheme of order p

(1) Let w_{n+1} be the result of taking a step of size h.

(2) Let $\hat{y}_{n+1}$ be the result of taking two steps of size h/2.



we expect ($u_n(t_n+h)$ ← exact soln through (t_n, y_n))

$$w_{n+1} = u_n(t_n+h) + h^p \varepsilon(t_n+h) + O(h^{p+2})$$
$$\hat{y}_{n+1} = u_n(t_n+h) + \left(\frac{h}{2}\right)^p \varepsilon(t_n+h) + O(h^{p+2})$$

where $\varepsilon(t)$ satisfies a variational equation with initial condition $\varepsilon(t_n) = 0$.

$$\therefore \quad \varepsilon(t_n+h) = 0 + \varepsilon'(t_n)h + \frac{\varepsilon''(t_n)}{2}h^2 + \cdots$$

$$w_{n+1} = u_n(t_{n+1}) + h^{p+1}\varepsilon'(t_n) + O(h^{p+2})$$
$$\hat{y}_{n+1} = u_n(t_{n+1}) + \frac{h^{p+1}}{2^p}\varepsilon'(t_n) + O(h^{p+2})$$

so the following linear combination cancels the leading error term:

$$y_{n+1} = \hat{y}_{n+1} + \frac{\hat{y}_{n+1} - w_{n+1}}{2^p - 1} = u_n(t_{n+1}) + O(h^{p+2})$$

method becomes order p+1. (so replace p by p+1 in stepsize control algorithm)

use this to advance the solution to the next step.

Runge-Kutta order conditions (consistency of RK methods)

Butcher array

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad \Longleftrightarrow \quad \begin{cases} k_1 = f(t_n + c_1 h,\ y_n + h(a_{11}k_1 + \cdots + a_{1s}k_s)) \\ \vdots \\ k_s = f(t_n + c_s h,\ y_n + h(a_{s1}k_1 + \cdots + a_{ss}k_s)) \\ y_{n+1} = y_n + h(b_1 k_1 + \cdots + b_s k_s) \end{cases}$$

A: $s \times s$ matrix

$b, c \in \mathbb{R}^s$

for simplicity, we'll assume the system is autonomous: $y' = f(y)$ ← no explicit time dependence

(requires that $c_i = \sum_j a_{ij}$, $\sum_j b_j = 1$)

Let's start by deriving a 2-stage ERK (explicit) method of maximal accuracy

$$\begin{array}{c|cc} 0 & 0 & 0 \\ a & a & 0 \\ \hline & b_1 & b_2 \end{array}$$

$$k_1 = f(y_n)$$

$$k_2 = f(y_n + h a k_1)$$

$$y_{n+1} = y_n + h(b_1 k_1 + b_2 k_2)$$

← most general 2-stage ERK method

The truncation error is

$$\tau_n = \underbrace{y(t_n + h)}_{y + hy' + \frac{h^2}{2}y'' + \frac{h^3}{6}y''' + O(h^4)} - y - h b_1 f(y) - h b_2 \underbrace{f(y + h a f(y))}_{k_2(h) = k_2(0) + k_2'(0)h + \frac{1}{2}k_2''(0)h^2 + \cdots}$$

where y stands for $y(t_n)$, y' for $y'(t_n)$, etc.

to compare terms, we need to express everything in terms of f

$y' = f(y)$ ⤵ chain rule: $\frac{d}{dt} f(y(t)) = \sum_{j=1}^{d} \frac{\partial f}{\partial y^j}(y(t)) \frac{dy^j}{dt}(t)$

$y'' = Df(y)(y') = Df(y)(f(y))$ ⤵ chain rule and product rule

$y''' = D^2 f(y)(f(y), f(y)) + Df(y)(Df(y)(f(y)))$

↑ $\sum_{j=1}^{d} \sum_{l=1}^{d} \frac{\partial^2 f}{\partial y^j \partial y^l}(y(t)) f^j(y(t)) f^l(y(t))$

↑ $\sum_{j=1}^{d} \sum_{l=1}^{d} \frac{\partial f}{\partial y^j}(y(t)) \frac{\partial f^j}{\partial y^l}(y(t)) f^l(y(t))$

$k_2(h) = f(y + h\, a f(y)) \longrightarrow k_2(0) = f(y)$

$k_2'(h) = Df(y + h\, af(y))(af(y)) \rightarrow k_2'(0) = a\, Df(y)(f(y))$

$k_2''(h) = D^2 f(y + h af(y))(af(y), af(y)) \rightarrow k_2''(0) = a^2 D^2 f(y)(f(y), f(y))$

final result:

$$\tau_n = (y - y) + (y' - b_1 f(y) - b_2 g(0))h + (\tfrac{1}{2} y'' - b_2 g'(0))h^2 + (\tfrac{1}{6} y''' - \tfrac{1}{2} b_2 g''(0))h^3 + \dots$$

$$= (1 - b_1 - b_2) f(y) h + (\tfrac{1}{2} - b_2 a) Df(y)(f(y)) h^2 + \left[\left(\tfrac{1}{6} - \tfrac{1}{2} b_2 a^2 \right) D^2 f(y)(f(y), f(y)) + \tfrac{1}{6} Df(y)(Df(y)(f(y))) \right] h^3 + O(h^4)$$

So we need $b_1 + b_2 = 1$ for first order or better

$b_2 a = \frac{1}{2}$ for 2nd order or better

and there's no way to achieve 3rd order

examples:

0	
1/2	1/2
	0 1

midpoint rule

$(a = 1/2, b_1 = 0, b_2 = 1)$ ✓

0	
1	1
	1/2 1/2

explicit trap. rule

$a = 1, b_1 = b_2 = 1/2$ ✓

all else being equal, we might as well kill the $O(h^3)$ term we have control over:

$$\frac{1}{6} - \frac{1}{2} b_2 a^2 = 0 \Rightarrow b_2 a^2 = \frac{1}{3}$$

$$b_2 a = \frac{1}{2}$$

$$a = \frac{2}{3}, \; b_2 = \frac{3}{4}, \; b_1 = \frac{1}{4}$$

0	
2/3	2/3
	1/4 3/4

↑ optimal 2-stage explicit RK method.

note: there is little geometric intuition for this choice of coefficients — they come out of tedius algebraic manipulations. For implicit RK methods,

this approach is known as collocation

{ one often chooses the c_i to coincide with Gaussian quadrature points (to solve $y' = f(t)$ accurately, i.e. to order $2s$: $y_{n+1} = y_n + \sum_{j=1}^{s} b_j f(t_n + c_j h)$)

Gaussian quadrature weights and abscissas ↑ ↑ more later...

Now we're ready to analyze the general case:

$$T_n = y(t_n+h) - y - h\sum_{j=1}^{s} b_j k_j(h) \qquad (y \text{ means } y(t_n))$$

$$k_i(h) = f\left(y + h\sum_{j=1}^{s} a_{ij} k_j(h)\right)$$

we have 2 expansions to work out:
$$\begin{cases} y(t_n+h) - y = hy' + \frac{h^2}{2}y'' + \frac{h^3}{6}y''' + \cdots \\ k_i(h) = k_i + h k_i' + \frac{h^2}{2}k_i'' + \cdots \end{cases}$$

what's the pattern in the following sequence of formulas?

$$y' = f$$
$$y'' = Df(f)$$
$$y''' = D^2f(f,f) + Df(Df(f))$$
$$y^{(4)} = D^3f(f,f,f) + 3D^2f(Df(f),f) + Df(D^2f(f,f)) + Df(Df(Df(f)))$$

$$y^{(5)} = D^4f(f,f,f,f) + 6D^3f(Df(f),f,f) + 4D^2f(D^2f(f,f),f) + 3D^2f(Df(f),Df(f)) + 4D^2f(Df(Df(f)),f) + Df(D^3f(f,f,f)) + 3Df(D^2f(Df(f),f)) + Df(Df(D^2f(f,f))) + Df(Df(Df(Df(f))))$$

clearly we wouldn't want to proceed any further directly. It turns out that graph theory can be used to concisely represent all these terms.

machinery:

1. graphs consist of nodes and edges
2. the order of a graph is the # of nodes
3. a (rooted) tree is a graph in which every two nodes are joined by a single path (no cycles) and one node is singled out as the root. (Rooted) trees have a parent/child hierarchy

(from now on "tree" means "rooted tree".)

idea: set up a correspondence between trees and elementary differentials:

denote the mapping by F:

$\bullet \longleftrightarrow f$ $\qquad F(\bullet)(y) = f(y)$

$\longleftrightarrow Df(f)$ $\qquad F(\,)(y) = Df(y)(f(y))$

$\longleftrightarrow D^2f(f,f)$

etc.

$\longleftrightarrow D^4f\big(D^2f(Df(f),f), Df(Df(f)), f, f\big)$

trees can be differentiated pictorally by adding a branch to each node in turn:

$$\frac{d}{dt}F(\,) = F(\,) + F(\,) + F(\,) + F(\,)$$

here I mean the composite function $F(\,)(y(t))$

example: $F(\bigvee)(y(t)) = D^2 f(y(t))\left(f(y(t)), f(y(t))\right)$

$$= \sum_{jk} \frac{\partial^2 f}{\partial y^j \partial y^k}(y(t))\, f^j(y(t))\, f^k(y(t)) = f_{jk} f^j f^k$$

↑ shorthand (implied sums)

$$\frac{d}{dt}\left(f_{jk} f^j f^k\right) = f_{jkl} f^j f^k f^l + f_{jk} f^j_l f^k f^l + f_{jk} f^j f^k_l f^l$$

↑ product & chain rule

these sums are actually equal (change dummy indices $j \leftrightarrow k$, use symmetry: $\frac{\partial^2 f}{\partial y^j \partial y^k} = f_{jk} = f_{kj}$)

equivalent graph manipulation:

$$\frac{d}{dt} F(\bigvee) = F(\bigvee\!\!\circ) + F(\text{◡}\!\circ) + F(\circ\!\text{◡}) = F(\bigvee\!\!\circ) + 2F(\text{◡}\!\circ)$$

same tree

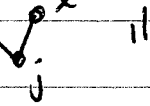
The number of trees of order q grows rapidly (superexponentially):

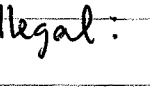
q:	1	2	3	4	5	6	7	8	9	10	...
$\#(T_q)$:	1	1	2	4	9	20	48	115	286	719	...

↑ T_q is the set of trees with q vertices

Theorem: $$y^{(q)}(t) = \sum_{\phi \in T_q} \alpha(\phi) F(\phi)(y(t)), \quad q = 1, 2, 3, \ldots$$

where $\alpha(\phi)$ is the number of ways of labelling the q nodes of ϕ with q distinct letters such that the parent of any node comes earlier in the alphabet.

legal: 

illegal: 

Last time: the truncation error of a general RK method

$$\tau_n = y(t_n+h) - y - h\sum_{j=1}^{s} b_j k_j(h)$$

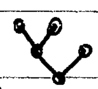
has the expansion

$$\tau_n = \Big(y' - \sum_j b_j k_j\Big)h + \Big(\tfrac{1}{2}y'' - \sum_j b_j k_j'\Big)h^2$$

$$+ \cdots + \Big(\frac{1}{p!}y^{(p)} - \frac{1}{(p-1)!}\sum_j b_j k_j^{(p-1)}\Big)h^p + O(h^{p+1})$$

where $y^{(q)}$ means $y^{(q)}(t_n)$
and $k_i^{(q)}$ means $k_i^{(q)}(0)$, where $k_i(h) = f\big(y + h\sum_j a_{ij} k_j(h)\big)$

we showed how to use graph theory to represent the time derivatives of y in terms of elementary differentials of f, e.g.

$F\Big($  $\Big)(y) = D^2f(y)\Big(D^2f(y)(f(y), f(y)),\ f(y)\Big)$

Theorem: if $y(t)$ satisfies $y' = f(y)$, then

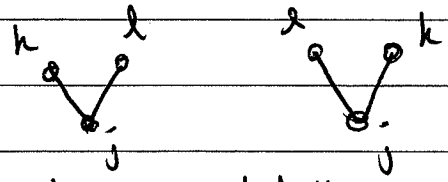
$$y^{(q)}(t) = \sum_{\phi\in T_q} \alpha(\phi)\, F(\phi)(y(t)) \qquad q = 1, 2, 3, \ldots$$

where T_q = set of rooted trees of order q (i.e. with q vertices)
and $\alpha(\phi)$ = # of ways of labeling the nodes of ϕ such that the parent of any node comes earlier in the alphabet.

$\alpha(\circ) = 1$

$\alpha(\;) = 1$

$\alpha(\;) = 1$ ← tricky:

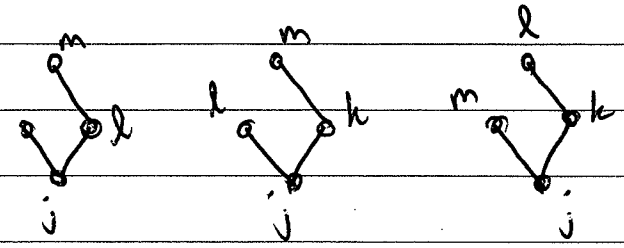


are the same labelling since you can map the nodes of one tree to the other without changing the parent/child relationship

$\alpha(\;) = 1$

$\alpha(\;) = 1$

$\alpha(\;) = 3$ ←

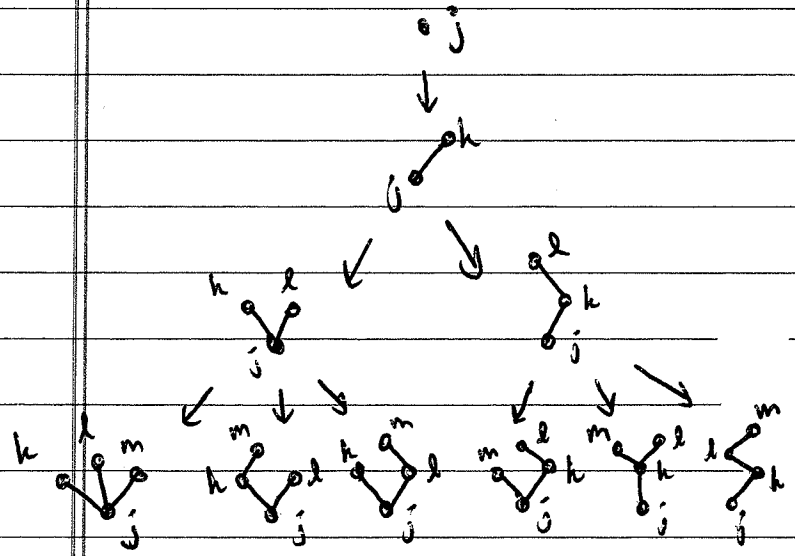


$\alpha(\;) = 1$, $\alpha(\;) = 1$

That covers all the trees of order ≤ 4.

---

proof of theorem: each labeled tree appears exactly once if you don't consolidate terms using symmetry of mixed partial derivatives



$$(y^j)' = f^j$$

$$(y^j)'' = f^j_k f^k$$

$$(y^j)''' = f^j_{kl} f^k f^l + f^j_k f^k_l f^l$$

$$(y^j)^{(4)} = f^j_{klm} f^k f^l f^m + f^j_{kl} f^k_m f^l f^m + f^j_{kl} f^k f^l_m f^m + f^j_{km} f^k f^l f^m + f^j_k f^k_{lm} f^l f^m + f^j_k f^k_l f^l_m f^m$$

Next we need to work out the derivatives of

$$k_i(h) = f\Big(\underbrace{y + h\sum_j a_{ij} k_j(h)}_{g_i(h)}\Big)$$

By the chain rule and product rules:

$$k_i'(h) = Df(g_i(h))(g_i'(h))$$

$$k_i''(h) = D^2 f(g_i(h))(g_i'(h), g_i'(h)) + Df(g_i(h))(g_i''(h))$$

$$k_i'''(h) = D^3 f(g_i(h))(g_i'(h), g_i'(h), g_i'(h)) + 3D^2 f(g_i(h))(g_i''(h), g_i'(h)) + Df(g_i(h))(g_i'''(h))$$

etc.

Note that

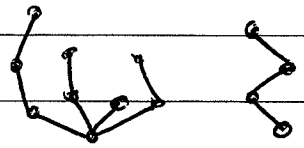
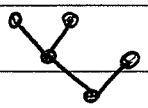
$$g_i'(h) = \sum_j a_{ij} k_j(h) + h\sum_j a_{ij} k_j'(h) \xrightarrow{h\to 0} \sum_j a_{ij} k_j(0)$$

$$g_i''(h) = 2\sum_j a_{ij} k_j'(h) + h\sum_j a_{ij} k_j''(h) \to 2\sum_j a_{ij} k_j'(0)$$

$$g_i'''(h) = 3\sum_j a_{ij} k_j''(h) + h\sum_j a_{ij} k_j'''(h) \to 3\sum_j a_{ij} k_j''(0)$$

Since we're only interested in the derivatives $k_i^{(q)}(h)$ <u>at $h=0$</u>, even when the method is implicit, the formulas for the derivatives $k_i^{(q)}(0)$ may be expressed explicitly in terms of f and its derivatives evaluated at y. Graph theory will allow us to evaluate these formulas recursively in an elegant way.

Let S_q be the set of special trees with no ramifications (i.e. branching nodes) except at the root

examples:  not allowed: 

For any tree $\phi \in S_{q+1}$, we associate a formula involving as many derivatives of f as there are branches at the root, and plug in derivatives of g according to how many nodes are on each branch:

$$G\left(\vee\right)(h) = D^2 f(g(h))\left(g'(h), g'(h)\right)$$

$$G\left(\curlyvee\right)(h) = D^3 f(g(h))\left(g'''(h), g''(h), g'(h)\right)$$

Faá diBruno formula: suppose $k(h) = f(g(h))$.

Then $$k^{(q)}(h) = \sum_{\phi \in S_{q+1}} \alpha(\phi)\, G(\phi)(h) \qquad q = 1, 2, 3, \ldots$$

where $\alpha(\phi)$ is again the number of ways to label the tree (in order along branches). It's worth comparing this to Leibniz' rule for derivatives of a product:

$$\frac{d^q}{dh^q}\left(f(h)g(h)\right) = \sum_{j=0}^{q} \binom{q}{j} f^{(j)}(h)\, g^{(j)}(h)$$

$\alpha(\phi)$ takes the place of the binomial coefficient.

All that remains is to plug $g_i^{(q)}(0) = q \sum_j a_{ij} k_j^{(q-1)}(0)$

into $k_i^{(q)}(0) = \sum_{\phi \in S_{q+1}} \alpha(\phi)\, (\Gamma_i(\phi))(0)$ recursively.

The amazing thing is that even when the method is implicit, the derivatives <u>at $h=0$</u> depend explicitly on f:

$$k_i(0) = f(y)$$

$$k_i'(0) = \overbrace{Df(y)(g'(0))}^{\Gamma(\bullet)} = \Big(\sum_j a_{ij}\Big) Df(y)(f(y))$$

(pull the sum outside — Df is linear)

$$k_i''(0) = \overbrace{D^2 f(y)(g'(0), g'(0))}^{\Gamma(\vee)} + \overbrace{Df(y)(g''(0))}^{\Gamma(\curlyvee)}$$

$$\Big(\sum_j a_{ij}\Big)\Big(\sum_k a_{ik}\Big) D^2 f(y)(f(y), f(y)) \qquad 2 \sum_j a_{ij} k_j'(0)$$

(pull sums outside — $D^2 f$ is bilinear)

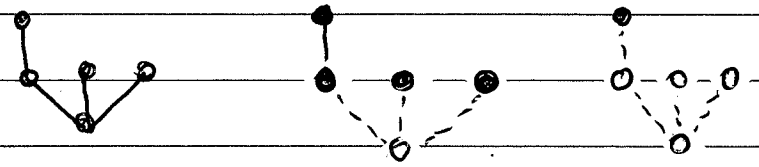
$$\Big(\sum_k a_{ik}\Big) Df(y)(f(y))$$

one may prove inductively that

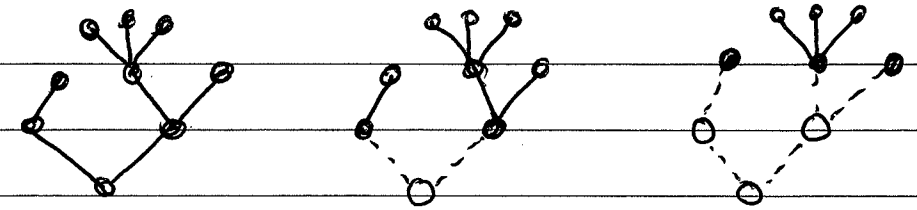
$$k_i^{(q-1)}(0) = \frac{1}{q} \sum_{\phi \in T_q} \alpha(\phi)\, \gamma(\phi)\, \Phi_i(\phi)\, F(\phi)(y)$$

- $\alpha(\phi)$: number of labelings
- $\gamma(\phi)$: products of subtree orders
- $\Phi_i(\phi)$: sums of products of the a_{ij} that were pulled outside the multilinear operators
- $F(\phi)(y)$: elementary differential

Here $\gamma(\phi)$ is the product of the orders of the trees obtained when you repeatedly remove the roots of the trees:

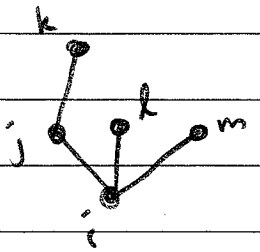
$\phi =$ 

$\gamma(\phi) = 5 \cdot 2 \cdot 1 \cdot 1 \cdot 1 = 10$

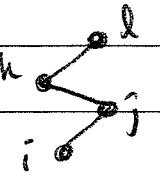
$\phi =$ 

$\gamma(\phi) = 9 \cdot 2 \cdot 6 \cdot 1 \cdot 4 \cdot 1 = 432$

and $\underset{i}{\Phi}(\phi)$ is the sum over all non-root nodes of products of the a_{jk} corresponding to branch segments (edges of the graph)

$\phi =$ 

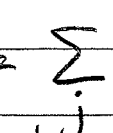
$$\underset{i}{\Phi}(\phi) = \sum_{jklm} a_{ij} a_{jk} a_{il} a_{im}$$

$\phi =$ 

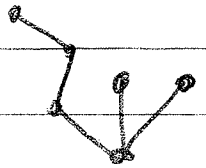
$$\underset{i}{\Phi}(\phi) = \sum_{jkl} a_{ij} a_{jk} a_{kl}$$

parent index

Note that the sum over terminal nodes yields c_*:

$\underset{i}{\Phi}$  $= \underbrace{\sum_j a_{ij} c_j c_i^2}_{(\mathrm{diag}(Ac)\,\mathrm{diag}(c)\,c)_i}$ $\qquad \underset{i}{\Phi}$  $= \underbrace{\sum_{jk} a_{ij} a_{jk} c_k}_{(A^2 c)_i}$

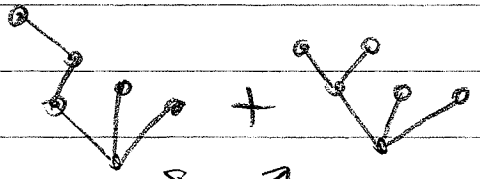
the idea of the induction step is that every time you plug a derivative of g into a slot from the Faá di Bruno formula, you can use multilinearity to expand that branch (which initially has no ramifications, i.e. splits) into all possible splits:



$\leftarrow$ belongs to S_6 (from Faá di Bruno formula)

$\downarrow$ plug $g_i''' = 3\sum_j a_{ij} k_j''$ into first slot

the factor of 3 cancels with the $\frac{1}{3}$ in the formula for k_j'' and the new a_{ij} gets absorbed by Φ in each term



everything in T_6 of the form $[3,1,1]$

Finally, we go back to page 1 and recall that

$$\tau_n = \left(y' - \sum_j b_j k_j\right)h + \cdots + \left(\frac{1}{p!}y^{(p)} - \frac{1}{(p-1)!}\sum_j b_j k_j^{(p-1)}\right)h^p + O(h^{p+1})$$

$$= \tau_n^{(1)} h + \tau_n^{(2)}\frac{h^2}{2} + \cdots + \tau_n^{(p)}\frac{h^p}{p!} + O(h^{p+1})$$

where

$$\tau_n^{(q)} = \sum_{\phi \in T_q} \alpha(\phi)\left(1 - \gamma(\phi)\sum_{j=1}^{s} b_j \Phi_j(\phi)\right)F(\phi)(y(t_n))$$

Conclusion: A Runge-Kutta method is order p iff

$$\sum_{j=1}^{s} b_j \Phi_j(\phi) = \frac{1}{\gamma(\phi)} \quad \text{for all trees } \phi \text{ of order} \le p.$$

(order conditions)

Stability of RK methods

All RK methods can be written in the form

$$y_{n+1} = y_n + h\Psi(y_n, h) \quad \leftarrow \text{ or } \Phi(t_n, y_n, h) \text{ if non-autonomous}$$

Ψ: Increment function of the method

example: general 2 step ERK:

$$\begin{array}{c|cc} 0 & 0 & \\ a & a & 0 \\ \hline & b_1 & b_2 \end{array}$$

$$k_1 = f(y_n)$$
$$k_2 = f(y_n + hak_1)$$
$$y_{n+1} = y_n + h(b_1k_1 + b_2k_2)$$

so $\Psi(y_n, h) = b_1 f(y_n) + b_2 f(y_n + haf(y_n))$

the general case is trickier

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

$$k_1 = f\left(y_n + h\sum_{j=1}^{s} a_{1j}k_j\right)$$
$$\vdots$$
$$k_s = f\left(y_n + h\sum_{j=1}^{s} a_{sj}k_j\right)$$

} have to be solved implicitly

$$\Psi(y_n, h) = b_1 k_1(y_n, h) + \cdots + b_s k_s(y_n, h)$$

Theorem: If f is Lip. continuous, $\exists h_0 > 0$ s.t. for $0 \le h \le h_0$ the implicit equations have a unique solution and the resulting functions $k_i(y_n, h)$ are also Lip. continuous with respect to y_n (hence so is $\Psi(y_n, h)$)

proof: first we use the contraction mapping theorem to show that the equations have a unique solution (recall we did something similar in Lec 1 for the trapezoidal rule)

it's convenient to group the stage derivatives together into a giant "block" vector $\mathbb{k} \in \mathbb{R}^{sd}$, $\mathbb{k} = \begin{pmatrix} k_1 \\ \vdots \\ k_s \end{pmatrix}$ ← vectors in $\mathbb{R}^d$

now define $\mathbb{T} : \mathbb{R}^{sd} \to \mathbb{R}^{sd}$ via

$$\mathbb{T}\underbrace{\begin{pmatrix} k_1 \\ \vdots \\ k_s \end{pmatrix}}_{\mathbb{k}} = \begin{pmatrix} f(y_n + h(a_{11}k_1 + \cdots + a_{1s}k_s)) \\ \vdots \\ f(y_n + h(a_{s1}k_1 + \cdots + a_{ss}k_s)) \end{pmatrix} = \begin{pmatrix} T_1(\mathbb{k}) \\ \vdots \\ T_s(\mathbb{k}) \end{pmatrix}$$

the infinity norm is convenient: $\|\mathbb{k}\|_\infty = \max_{1 \le j \le s} \|k_j\|_\infty = \max_{\substack{1 \le j \le s \\ 1 \le i \le d}} |k_j^i|$

Since f is Lipschitz continuous,

$$\|T_i(\mathbb{k}) - T_i(\mathbb{r})\|_\infty = \left\| \begin{array}{l} f(y_n + h(a_{i1}k_1 + \cdots + a_{is}k_s)) \\ -f(y_n + h(a_{i1}r_1 + \cdots + a_{is}r_s)) \end{array} \right\|_\infty$$

$$\le L \left\| h \sum_{j=1}^{s} a_{ij}(k_j - r_j) \right\|_\infty$$

$$\le hL \left(\sum_{j=1}^{s} |a_{ij}| \right) \underbrace{\left(\max_j \|k_j - r_j\|_\infty \right)}_{\|\mathbb{k} - \mathbb{r}\|_\infty}$$

$$\therefore \|T(\mathbb{k}) - T(\mathbb{r})\|_\infty \le hL \underbrace{\left(\max_i \sum_j |a_{ij}| \right)}_{\|A\|_\infty} \|\mathbb{k} - \mathbb{r}\|_\infty$$

maximum absolute row sum

Now set $h_0 = \frac{1}{2L\|A\|_\infty}$ so that

$$0 \le h \le h_0 \Rightarrow \|\Pi(\mathbb{K}) - \Pi(\mathbb{K}')\|_\infty \le \frac{1}{2}\|\mathbb{K} - \mathbb{K}'\|_\infty$$

for all $\mathbb{K}, \mathbb{K}' \in \mathbb{R}^{sd}$

$\Rightarrow \Pi$ is a contraction

$\Rightarrow \Pi$ has a unique fixed point $\mathbb{K}$ such that

$$\boxed{\Pi(\mathbb{K}) = \mathbb{K}}$$

this is what it means to solve the equations implicitly.

so the functions $k_i(y_n, h)$ exist for $0 \le h \le h_0$, $y_n \in \mathbb{R}^d$.

Now let's check that they're Lipschitz continuous.
Freeze h and drop it from the notation.

we have $k_i(y_n) = f\left(y_n + h\sum_j a_{ij} k_j(y_n)\right)$

so changing y_n from x to y causes k_i to change by at most

$$\|k_i(x) - k_i(y)\|_\infty \le L\left(\|x-y\|_\infty + h\sum_j |a_{ij}|\,\|k_j(x) - k_j(y)\|_\infty\right)$$

Let $w_i = \|k_i(x) - k_i(y)\|_\infty \qquad 1 \le i \le s$

w is a vector in $\mathbb{R}^s$ and $\|\mathbb{K}(x) - \mathbb{K}(y)\|_\infty = \max_{1\le i\le s} |w_i| = \|w\|_\infty$

we have: $0 \le w_i \le L\left(\|x-y\|_\infty + h\sum_j |a_{ij}| w_j\right)$

$$= L\|x-y\|_\infty + hL(Gw)_i$$

where $G = \begin{pmatrix} |a_{11}| & \cdots & |a_{1s}| \\ \vdots & & \vdots \\ |a_{s1}| & \cdots & |a_{ss}| \end{pmatrix}$ is an $s \times s$ matrix

$$\therefore \|w\|_\infty \le L\|x-y\|_\infty + hL\|G\|_\infty \|w\|_\infty$$

note that $\|G\|_\infty = \|A\|_\infty$ (same absolute row sums)

we already assumed $hL\|A\|_\infty \le \frac{1}{2}$

$$\therefore \|w\|_\infty \le L\|x-y\|_\infty + \frac{1}{2}\|w\|_\infty$$

$$\frac{1}{2}\|w\|_\infty \le L\|x-y\|_\infty$$

$$\|w\|_\infty \le 2L\|x-y\|_\infty$$

recalling the definition of w, we obtain

$$\|k_i(x) - k_i(y)\|_\infty \le 2L\|x-y\|_\infty$$

and

$$\|\Phi(x,h) - \Phi(y,h)\|_\infty \le 2L\overbrace{(|b_1| + \cdots + |b_s|)}^{\text{usually equals 1.}}\|x-y\|_\infty$$

$\therefore$ k_i and Φ are Lipschitz continuous and the Lipschitz constant is closely related to that of f.

Theorem: if f is C^r in a neighborhood of y_n, then $k_i(y,h)$ and $\Phi(y,h)$ are also C^r functions of y and h near $(y_n, 0)$.

proof: apply the implicit function theorem to solve $\mathbb{F}(\mathbb{k}, y, h) = 0$ for $\mathbb{k}$ in terms of y and h, where

$$\mathbb{F}(\mathbb{k}, y, h) = \mathbb{k} - \mathbb{T}(\mathbb{k}) = \begin{pmatrix} k_1 - f\left(y + h\sum_j a_{1j} k_j\right) \\ \vdots \\ k_s - f\left(y + h\sum_j a_{sj} k_j\right) \end{pmatrix}$$

this is possible because $\mathbb{F}\left(\begin{pmatrix} f(y_n) \\ \vdots \\ f(y_n) \end{pmatrix}, y_n, 0\right) = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$

and $D_{\mathbb{k}}\mathbb{F}\left(\begin{pmatrix} f(y_n) \\ \vdots \\ f(y_n) \end{pmatrix}, y_n, 0\right) = \begin{pmatrix} I & & \\ & \ddots & \\ & & I \end{pmatrix}$ is invertible

Implicit function theorem: suppose $F: \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^m$ is C^r in a neighborhood of $(x_0, y_0) \in \mathbb{R}^m \times \mathbb{R}^d$ and suppose

partial Jacobian matrix $\left(\frac{\partial F_i}{\partial x_j}\right)$ $m \times m$ matrix

$$F(x_0, y_0) = 0, \quad D_x F(x_0, y_0) \text{ is invertible}$$

Then there is a neighborhood U of y_0 and a unique function $g: U \to \mathbb{R}^m$ (also C^r) such that

$$F(g(y), y) = 0 \qquad y \in U$$

i.e. you can uniquely solve $F(x,y) = 0$ for x in terms of y.

(Our discussion above of the Lipschitz case contains the first half of the proof of the implicit function theorem.)

In Lecture 14 we worked out what the derivatives of the k_i with respect to h are:

$$k_i^{(q-1)}(0) = \frac{1}{q} \sum_{\phi \in T_q} \alpha(\phi)\, \gamma(\phi)\, \Phi_i(\phi)\, F(\phi)(y)$$

Now we have justified our assumption that the k_i's could be differentiated.

Stability proof: Let's add a new feature: non-uniform stepsizes.

$$y_{n+1} = y_n + h_n \Psi(y_n, h_n)$$

$$y(t_{n+1}) = y(t_n) + h_n \Psi(y(t_n), h_n) + \tau_n \quad \text{(def. of } \tau_n\text{)}$$

$$e_{n+1} = e_n + h_n \left[\Psi(y_n, h_n) - \Psi(y(t_n), h_n) \right] - \tau_n$$

$$\| e_{n+1} \| \leq \| e_n \| + h_n L \| e_n \| + \| \tau_n \|$$

(L: Lip. const. for Ψ now)

In the uniform case, we used $1+hL \le e^{hL}$ to make it easy to raise $1+hL$ to various powers.

In the non-uniform case, we will instead use

$$1+h_nL \le e^{h_nL} = e^{(t_{n+1}-t_n)L}$$

Then

$$\|e_n\| \le e^{(t_n-t_{n-1})L}\|e_{n-1}\| + \|\tau_{n-1}\|$$

$$\le e^{(t_n-t_{n-1})L}\left(e^{(t_{n-1}-t_{n-2})L}\|e_{n-2}\| + \|\tau_{n-2}\|\right) + \|\tau_{n-1}\|$$

$$= e^{(t_n-t_{n-2})L}\|e_{n-2}\| + e^{(t_n-t_{n-1})L}\|\tau_{n-2}\| + \|\tau_{n-1}\|$$

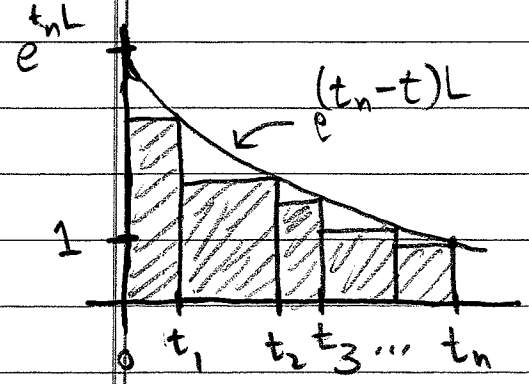
$$\vdots$$

$$\le e^{(t_n-t_0)}\underbrace{\|e_0\|}_{\text{usually zero}} + e^{(t_n-t_1)L}\|\tau_0\| + \cdots + e^{(t_n-t_{n-1})L}\|\tau_{n-2}\| + \|\tau_{n-1}\|$$

$$\sum_{j=1}^{n} e^{(t_n-t_j)L}\|\tau_{j-1}\|$$

Now assume $\|\tau_j\| \le C(t_j)h_j^{p+1}$ for $0 \le j \le n-1$

Then $\|e_n\| \le \left(\max_j C(t_j)h_j^p\right)\sum_{j=1}^{n} e^{(t_n-t_j)L}h_{j-1}$



a lower Riemann sum

$$\le \left(\max_j C(t_j)h_j^p\right)\int_0^{t_n} e^{(t_n-t)L}\,dt$$

$$= (\quad)\frac{e^{t_nL}-1}{L}$$

Very similar to result for Euler's method.

If a handful of τ_j's are only $O(h_j^p)$, they can be separated from the rest of the sum like errors in the initial condition:

$$\|e_n\| \le \left(\max_{\text{good } j\text{'s}} C(t_j) h_j^p \right) \frac{e^{t_n L} - 1}{L} + e^{t_n L} \underbrace{\left(\|e_0\| + \sum_{\text{bad } j\text{'s}} \|\tau_{j-1}\| \right)}$$

just need to be sure this doesn't exceed the sum of all the "typical terms" by too much (method remains order p)

The point of stepsize control is to estimate $C(t_j)$ as you go and take big steps where $C(t_j)$ is small and vice versa so that $C(t_j) h_j^p$ remains approximately constant throughout the computation.

for a method of order p, we have

$$\tau_j = \frac{h_j^{p+1}}{(p+1)!} \tau_j^{(p+1)} + O(h^{p+2})$$

where $$\tau_j^{(p+1)} = \sum_{\phi \in T_p} \alpha(\phi) \left[1 - \gamma(\phi) \sum_{i=1}^{s} b_i \bar{\Phi}_i(\phi) \right] F(\phi)(y(t_n))$$

which gives an explicit formula for $C(t_j)$ to first order, but of course it would be extremely expensive to evaluate. It's much better to estimate $C(t_j)$ using two methods of different orders

Embedded Runge-Kutta methods and stepsize control

we saw last time that the error $e_n = y_n - y(t_n)$ is bounded by

$$\|e_n\| \le \left(\max_{0 \le j \le n-1} C(t_j) h_j^p \right) \frac{e^{t_n L} - 1}{L} + e^{t_n L} \|e_0\|$$

where $\|\tau_j\| \le C(t_j) h_j^{p+1}$

we interpret $err = C(t_j) h_j^p$ as an error rate, i.e. a bound on how much error we're introducing into the numerical solution per unit time. We would like to keep this approximately constant for an optimally efficient code.

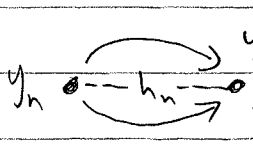
want $C(t_j) h_j^p \approx \varepsilon$ ← prescribed parameter (chosen by user)

The formula $\tau_j = \frac{h_j^{p+1}}{(p+1)!} \tau_j^{(p+1)} + O(h_j^{p+2})$ yields a first order estimate of what $C(t_j)$ should be, namely

$$C(t_j) = \left\| \frac{1}{(p+1)!} \sum_{\phi \in T_{p+1}} \alpha(\phi) \left(1 - \gamma(\phi) \sum_{i=1}^{s} b_i \Phi_i(\phi) \right) F(\phi)(y(t_j)) \right\| + O(h_j)$$

but it would be very inefficient to evaluate all the terms in this equation.

So instead we use two different schemes to try to guess what $C(t_n)$ is as we go, and adjust h_n accordingly.



$y_{n+1} \leftarrow$ order p method, $\|\tau_n\| \leq C(t_n) h_n^{p+1}$

$\hat{y}_{n+1} \leftarrow$ order $p-1$ method $\|\hat{\tau}_n\| \leq \hat{C}(t_n) h_n^{p}$

note: we've changed our definition of truncation error a bit:

Since we don't know $y(t_n)$, we do our best with what we have:

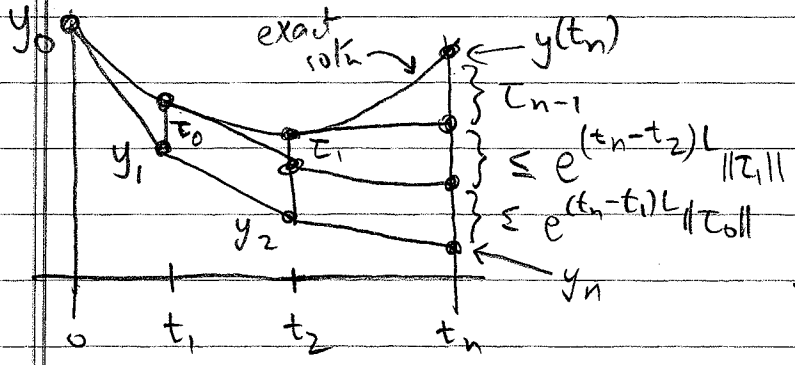
let $u_n(t)$ = exact soln of $\begin{cases} u' = f(u) \\ u(t_n) = y_n \end{cases}$ = trajectory you jumped onto at step n.

define τ_n by

$$u_n(t_{n+1}) = \underbrace{u_n(t_n) + h_n \Phi(u_n(t_n), h_n)}_{\text{also equals } y_{n+1} \text{ and } u_{n+1}(t_{n+1})} + \tau_n$$

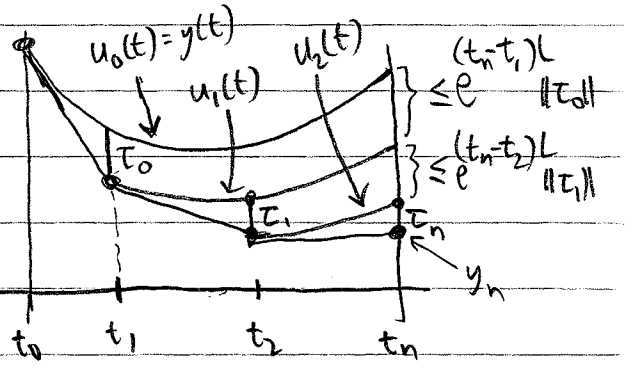
(y_n under $u_n(t_n)$; y_n under $u_n(t_n)$)

usual convergence proof



propagate error discretely using the scheme

alternative proof



propagate errors continuously using nearby exact solutions.

the alternative proof is complicated by the fact that the error bound involves the largest truncation error you could get in a band around the exact solution $y(t)$, and you have to prove that the numerical solution doesn't leave this band. (not too hard...)

so then $y_{n+1} = u_n(t_{n+1}) - \tau_n$

$\hat{y}_{n+1} = u_n(t_{n+1}) - \hat{\tau}_n$

$$\|\hat{y}_{n+1} - y_{n+1}\| = \|\hat{\tau}_n - \tau_n\| \approx \|\hat{\tau}_n\| = \hat{C}(t_n) h_n^p$$

Now we make the big assumption (unjustified) that $C(t_n) \approx \hat{C}(t_n)$ and expect that $C(t_{n+1}) \approx C(t_n)$ (C really is a continuous function of t)

we choose our next step so that $C(t_{n+1}) h_{n+1}^p = \varepsilon$, i.e.

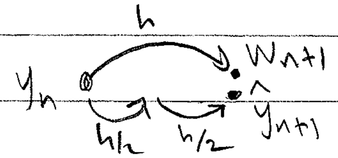
$$h_{n+1} = \left(\frac{\varepsilon}{C(t_{n+1})}\right)^{1/p} \approx \left(\frac{\varepsilon}{\hat{C}(t_n)}\right)^{1/p} \approx h_n \cdot \left(\frac{\varepsilon}{\|\hat{y}_{n+1} - y_{n+1}\|}\right)^{1/p}$$

A more rigorous assumption would be $C(t_{n+1}) \leq \hat{C}(t_n) h_n^{-1}$ (which amounts to assuming $\|\tau_n\| \leq \|\hat{\tau}_n\|$ rather than $\|\tau_n\| \leq h\|\hat{\tau}_n\|$, i.e. the higher order method is better) which would lead to

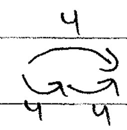
$$h_{n+1} = h_n \cdot \left(\frac{\varepsilon h_n}{\|\hat{y}_{n+1} - y_{n+1}\|}\right)^{1/p}$$

In practice, the two approaches are comparable in performance and accuracy, the former being a little faster and the latter a bit safer.

In the homework, we used Richardson extrapolation to turn our fourth order RK4 method into a 5th order stepsize control algorithm.



$$y_{n+1} = \hat{y}_{n+1} + \frac{\hat{y}_{n+1} - W_{n+1}}{2^p - 1}$$

this is actually quite inefficient:  = 12 f evaluations per step.

the idea of an embedded RK method is to construct two schemes that share most of their f evaluations.

notation

$$\begin{array}{c|c} c & A \\ \hline & b^T \\ \hline & \hat{b}^T \end{array}$$

A ← both use same stages

b^T, $\hat{b}^T$ ← different combination of slopes

$$y_{n+1} = y_n + h(b_1 k_1 + \dots + b_s k_s)$$
$$\hat{y}_{n+1} = y_n + h(\hat{b}_1 k_1 + \dots + \hat{b}_s k_s)$$

example:

0	0			
2/3	2/3	0		
2/3	0	2/3	0	
	1/4	3/8	3/8	← order 3
	1/4	3/4	0	← order 2

← a (3,2) method

you can check that $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T(c^2) = \frac{1}{3}$, $b^T A c = \frac{1}{6}$

$\hat{b}^T e = 1$, $\hat{b}^T c = \frac{1}{2}$ | $b^T A^2 c = 0 \neq \frac{1}{24}$, $\hat{b}^T A c = 0 \neq \frac{1}{6}$

RKF45 (Fehlberg 1968) is a (4,5) method with 6 stages

↑ the lower order method is intended to advance the soln (see book p. 84)

DOPRI (Dormand Prince 1980) is a (5,4) method with 7 stages

↑ higher order method used to advance the soln.

this method was optimized to make the coefficients in the leading term in the truncation error as small as possible (like we did for the 2-stage ERK in Lec 13)

↳ it also has $c_7 = 1$, b^T = 7th row of A

"first same as last" property (FSAL). Even when a step is accepted, f doesn't have to be re-evaluated at the start of the next step.

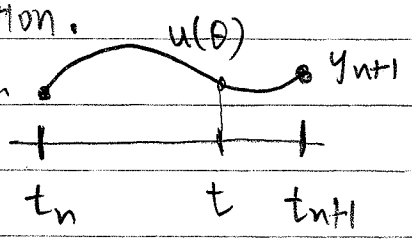
→ only 6 f evaluations per step

DOPRI is the best known general purpose RK method.

There are a handful of higher order methods (Fehlberg 7(8), DOPRI(8,6), DVERK 6(5)) but they are rarely used except in specialized applications that call for extreme accuracy (usually need more than double precision arithmetic for these to pay off)

Dense output. sometimes you want to know the solution at intermediate values of t, where you didn't compute a y_n. E.g. the crossing point of the pendulum problem.

for 4th order methods, a good solution is Hermite interpolation.



$t = t_n + \theta h \qquad 0 \le \theta \le 1$

known: $y_n, f(y_n), y_{n+1}, f(y_{n+1})$

interpolate by the unique cubic polynomial that agrees with the values and slopes at the endpoints

$$u(\theta) = (1-\theta)y_n + \theta y_{n+1} - \theta(1-\theta)\Big((1-2\theta)(y_{n+1}-y_n) - (1-\theta)hf_n + \theta h f_{n+1}\Big)$$

If more accuracy is required, you can bootstrap.

example: suppose y_n, y_{n+1} are known to 5th order. $\left(y_n = y(t_n) + \alpha(h^5)\right)$

cubic interpolation yields a 4th order approximation for $t_n \le t \le t_{n+1}$, but we want 5th order accuracy.

solution pick $\alpha \in (0,1)$ and define $u(\theta)$ via interpolation:

$u(0) = y_0 \quad u'(0) = hf(t_n, y_n) \quad u'(\alpha) = hf\left(t_n + \alpha h, \tilde{u}(\alpha)\right)$

$u(1) = y_1 \quad u'(1) = hf(t_{n+1}, y_n)$

↑ from cubic interpolation gives an extra order

(works as long as $\alpha \neq 1/2$)

higher order bootstrapping to any order is possible by repeating this procedure and adding an interpolation point on each successive iteration (though I doubt you'd ever go for more than 5th order in practice)

Collocation methods the easiest way to construct high order implicit methods is via collocation.

idea: choose $c_1 < c_2 < \cdots < c_s$ and find the polynomial $u(t)$ of degree $\leq s$ satisfying

$$u(t_n) = y_n$$
$$u'(t_n + c_i h) = f(t_n + c_i h,\ u(t_n + c_i h)) \qquad 1 \leq i \leq s$$

then define $y_{n+1} = u(t_n + h)$.

Claim: Collocation is a special case of Runge-Kutta:

proof: write $u'(t_n + c_i h) = k_i$ and use the fact that $u(t)$ is a polynomial to obtain

$$u'(t_n + \theta h) = \sum_{j=1}^{s} k_j \ell_j(\theta), \qquad \ell_j(\theta) = \prod_{k \neq j} \frac{(\theta - c_k)}{(c_j - c_k)}$$

↖ Lagrange polynomials (degree $= s-1$)

then

$$u(t_n + c_i h) = y_n + h \sum_{j=1}^{s} \underbrace{\left(\int_0^{c_i} \ell_j(\theta)\, d\theta \right)}_{a_{ij}} k_j$$

$$u(t_n + h) = y_n + h \sum_{j} \underbrace{\left(\int_0^{1} \ell_j(\theta)\, d\theta \right)}_{b_j} k_j$$

↖ $u(t_n) + \int_0^{c_i} \frac{d}{d\theta} u(t_n + \theta h)\, d\theta$

Theorem: Let $P(\theta) = \prod_{i=1}^{s} (\theta - c_i)$ and suppose

$P(\theta)$ is orthogonal to all polynomials of degree $\leq r-1$,

i.e.

$$\int_0^1 P(\theta)\, \theta^q d\theta = 0 \qquad 0 \leq q \leq r-1$$

Then the collocation method is order $p = s+r$.

proof: see Hairer/Norsett/Wanner or Iserles.

examples: implicit midpoint rule:

$1/2$	$1/2$
	1

$s=1, r=1 \Rightarrow$ **2nd order**

$\uparrow$ $\int_0^1 (\theta - \frac{1}{2}) \cdot 1 \, d\theta = 0$

Hammer-Hollingsworth:

$\frac{3-\sqrt{3}}{6}$	$1/4$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
$\frac{3+\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$1/4$
	$1/2$	$1/2$

$s=2, r=2 \Rightarrow$ **4th order**

$\uparrow$ $\frac{3 \pm \sqrt{3}}{6}$ are the zeros of $\tilde{P}_2(\theta)$, the 2nd shifted Legendre polynomial

$\tilde{P}_0(\theta) = 1$, $\tilde{P}_1(\theta) = 2\theta - 1$, $\tilde{P}_2(\theta) = 6\theta^2 - 6\theta + 1$, etc.

$$\int_0^1 \tilde{P}_i(\theta) \tilde{P}_j(\theta)\, d\theta = \frac{1}{2n+1} \delta_{ij}, \quad \text{so } \tilde{P}_2 \perp \text{span}\{\tilde{P}_0, \tilde{P}_1\} = \text{span}\{1, \theta\}$$

Stiff ODE's

example: (Prothero-Robinson equation)

$$y' = -L(y - \varphi(t)) + \varphi'(t) \qquad y(0) = \xi$$

here we imagine L is very large and $\varphi(t)$ is a nice smooth function (e.g. $\varphi(t) = 1$ or $\varphi(t) = \sin t$)

The exact solution is easily found:

$$(y - \varphi)' + L(y - \varphi) = 0$$

$$\frac{d}{dt}\left[e^{Lt}(y - \varphi)\right] = 0$$

$$e^{Lt}(y - \varphi) = \text{const} = \overbrace{y(0) - \varphi(0)}^{\xi}$$

$$\boxed{y(t) = e^{-Lt}(\xi - \varphi(0)) + \varphi(t)}$$

if L is very large, the first term decays to zero very rapidly (e.g. if $L = 1000$ and $t \geq 1/10$ then $e^{-Lt} \leq e^{-100} \sim 10^{-40}$) and the exact solution is as smooth as $\varphi(t)$ after the transient behavior has died.

problem: the Lipschitz constant for $f(t,y) = -L(y-\varphi(t)) + \varphi'(t)$ is L, which is big, so our error estimate

$$\max_{0 \le t_n \le T} \|e_n\| \le \left(\max_n \frac{\|\tau_n\|}{h} \right) \frac{e^{LT}-1}{L} + e^{LT} \|e_0\|$$

is basically useless for $T \ge \frac{10}{L}$.

on second thought, the intermediate stages could "feel" L and cause τ_n to be large

Although we have no problem taking small steps to make

$$\tau_n = y(t_{n+1}) - y(t_n) - h\Psi(t, y(t_n), h)$$

Ψ ← increment fcn of method

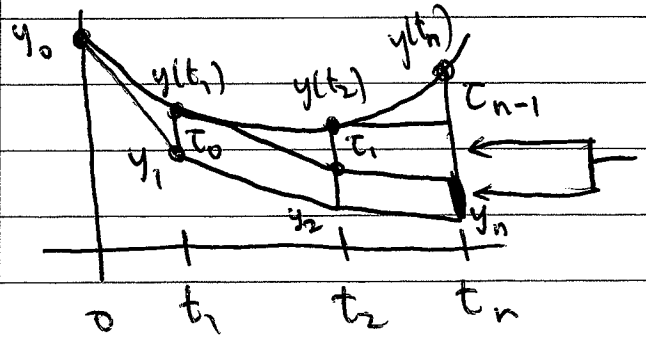
as small as we please (since $y \approx \varphi$ is very smooth), the requirement

$$\|\tau_n\| \le hLe^{-LT}\varepsilon$$

ε ← desired global error

is totally impractical!

go back to pictorial proof of convergence:



result when scheme propagates the truncation errors τ_1, τ_2 from t_1, t_2 to t_n (bounded by $e^{L(t_n-t_1)}\|\tau_0\|$ and $e^{L(t_n-t_2)}\|\tau_1\|$ respectively)

if we can somehow convince the scheme not to amplify errors when solving ODE's in which nearby solutions stay close to each other, we could drop the exponentials from the error bound. E.g. if the scheme didn't amplify errors <u>at all</u>, our final error bound would look like

$$\|e_n\| \le \|\tau_0\| + \cdots + \|\tau_{n-1}\| \le \left(\max_{0\le j\le n-1} \frac{\|\tau_j\|}{h_j}\right)\underbrace{\left(\sum_j h_j\right)}_{t_n} + \|e_0\|$$

which we can handle.

$\Rightarrow$

goals: ① take steps small enough to make $\|\tau_n\| = O(h_n^{p+1})$ small, but not necessarily small enough to make hL small.

(so fixed point iteration won't work)

② get rid of exponential growth in error estimates when the underlying ODE exhibits exponential <u>decay</u> instead of growth.

A particularly nice family of ODE's are the <u>contractive</u> ODE's:

<u>def</u>: $y' = f(t,y)$ is contractive if every pair of solutions $y(t), z(t)$ satisfies

$$\|y(t) - z(t)\| \le \|y(s) - z(s)\| \quad \text{whenever } t \ge s$$

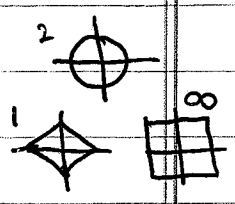
when is an ODE contractive?

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t)-z(t)\|^2\right) = \left(y'(t)-z'(t)\right)\cdot\left(y(t)-z(t)\right)$$

(dot product)

$$= \left(f(t,y)-f(t,z)\right)\cdot(y-z) \le 0$$

2-norm is best since it's smooth (no corners in unit balls)



def: $f: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is dissipative if

$$(f(t,y) - f(t,z)) \cdot (y-z) \leq 0 \qquad \forall t \in \mathbb{R},\ \forall y, z \in \mathbb{R}^d$$

example: $f(t,y) = -L(y - \varphi(t)) + \varphi'(t)$

$$(f(t,y) - f(t,z)) \cdot (y-z) = -L(y-z) \cdot (y-z)$$

$$= -L\|y-z\| \leq 0 \quad \checkmark$$

def: A method is B-stable (or contractive) if every pair of numerical solutions y_n, z_n of a dissipative ODE $y' = f(t,y)$ satisfies

$$\|y_{n+1} - z_{n+1}\| \leq \|y_n - z_n\| \qquad \forall n \geq 0$$

Example 1: backward Euler is B-stable.

$$y_{n+1} = y_n + h f(t_{n+1}, y_{n+1})$$

$$z_{n+1} = z_n + h f(t_{n+1}, z_{n+1})$$

$$e_n = y_n - z_n$$

$$e_{n+1} \cdot e_{n+1} = \Big(e_n + \overbrace{h\big[f(t_{n+1}, y_{n+1}) - f(t_{n+1}, z_{n+1})\big]}^{\circledast} \Big) \cdot e_{n+1}$$

$\circledast \cdot e_{n+1} = \circledast \cdot (y_{n+1} - z_{n+1}) \leq 0$ since f is dissipative

$\therefore \|e_{n+1}\|^2 \leq e_n \cdot e_{n+1} \leq \|e_n\| \cdot \|e_{n+1}\|$

$\therefore \|e_{n+1}\| \leq \|e_n\|$.

↑ Cauchy-Schwarz: $|x-y| \leq \|x\| - \|y\|$, $x, y \in \mathbb{R}^d$

Example 2: the implicit midpoint rule is B-stable

$$y_{n+1} = y_n + h f\left(t_n + \frac{h}{2}, \frac{y_n + y_{n+1}}{2}\right)$$

$$\|e_{n+1}\|^2 - \|e_n\|^2 = (e_{n+1} + e_n) \cdot (e_{n+1} - e_n)$$

$$= \underbrace{(e_{n+1} + e_n)} \cdot \left[h f\left(t_n + \frac{h}{2}, \frac{y_n + y_{n+1}}{2}\right) - h f\left(t_n + \frac{h}{2}, \frac{z_n + z_{n+1}}{2}\right) \right]$$

$$2\left(\frac{y_n + y_{n+1}}{2} - \frac{z_n + z_{n+1}}{2}\right)$$

≤ 0 since f is dissipative.

$\therefore \|e_{n+1}\| \leq \|e_n\|$ ✓ B-stable.

Example 3: the trapezoidal rule is not B-stable

Let $\varepsilon < \frac{1}{5}$, $f(y) = \begin{cases} -y & y \geq 0 \\ -y/\varepsilon & y \leq 0 \end{cases}$, $h = 1$

↑ clearly dissipative

↑ B-stability is supposed to hold for all $h > 0$

then $y_n = 0 \Rightarrow y_{n+1} = 0$

$z_n = -\varepsilon \quad z_{n+1} = \frac{1 - 2\varepsilon}{3}$

so $\|y_{n+1} - z_{n+1}\| > \|y_n - z_n\|$

check: $\frac{1 - 2\varepsilon}{3} = -\varepsilon + \frac{h}{2}\left(1 + \underbrace{-\frac{1 - 2\varepsilon}{3}}\right)$

z_{n+1} z_n $1/2$ $f(z_n)$ $f(z_{n+1})$

fact: most ODE's are not dissipative and most methods are not B-stable...

Best you can do is linearize the problem and choose a method that suppresses growth of modes that should decay. (there's nothing you can do about modes that grow — if exact solutions that start close together end up far apart, you're stuck with small timesteps!)

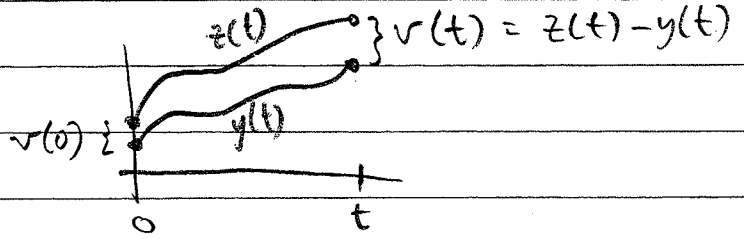
Linearization: let $y(t)$ be one solution of $y' = f(t,y)$ and let $z(t) = y(t) + v(t)$ be a nearby solution.

then $z'(t) = y'(t) + v'(t) = f(t, z(t))$

$$= \underbrace{f(t, y(t))}_{y'(t)} + D_y f(t, y(t)) v(t) + O(\|v(t)\|^2)$$

So $$v(t) = D_y f(t, y(t)) v(t) \underbrace{+ O(\|v(t)\|^2)}_{\text{neglect}}$$

which is just the variational equation we talked about in Lecture 3. Solutions of this (linear) equation tell us how nearby solutions evolve together.



When I talk about growth and decay modes, I just mean solutions of the variational equation that grow or decay.

We can also linearize the numerical solution.

$$y_{n+1} = y_n + h\Phi(t_n, y_n, h) \quad \leftarrow \text{main solution}$$

$$z_n = y_n + v_n \quad \leftarrow \text{nearby solution}$$

$$v_{n+1} = z_{n+1} - y_{n+1} = \underbrace{z_n - y_n}_{v_n} + h\left[\Phi(t_n, z_n, h) - \Phi(t_n, y_n, h)\right]$$

$$= v_n + h D_y\Phi(t_n, y_n, h)v_n + O(\|v_n\|^2)$$

Note that $D_y\Phi(t_n, y_n, h)v_n = \sum_{i=1}^{s} b_i D_y k_i(t_n, y_n, h)v_n$

and $k_i(t_n, y_n, h) = f\left(t_n + c_i h,\ y_n + h\sum_j a_{ij} k_j(t_n, y_n, h)\right)$

$$\Rightarrow D_y k_i(t_n, y_n, h)v_n = D_y f(\cdots)\left(I + h\sum_j a_{ij} D_y k_j(t_n, y_n, h)\right)v_n$$

we can now interpret $\boxed{v_{n+1} = v_n + h D_y\Phi(t_n, y_n, h)v_n}$ as the Runge-Kutta scheme applied to the variational equation $\boxed{v' = D_y f(t, y(t))v}$, namely

$$l_i = D_y f\left(t_n + c_i h,\ y(t_n + c_i h)\right)\left(v_n + h\sum_j a_{ij} l_i\right) \qquad 1 \le i \le s$$

$$v_{n+1} = v_n + h\sum_i b_i l_i$$

except that we replace $y(t_n + c_i h)$ by $y_n + h\sum_j a_{ij} k_j(t_n, y_n, h)$

example: forward Euler: $\Phi(t_n, y_n, h) = f(t_n, y_n)$

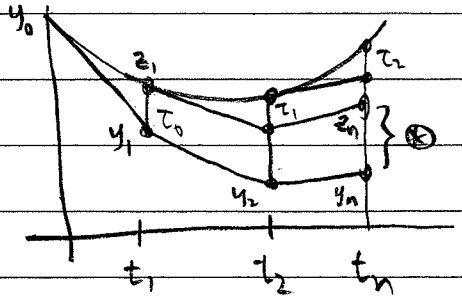
linearization: $v_{n+1} = v_n + h D_y f(t_n, y_n) v_n$

variational equation: $v' = D_y f(t, y(t)) v$ ← only difference

solving the variational equation via forward Euler: $v_{n+1} = v_n + h D_y f(t_n, y(t_n)) v_n$

Conclusion: the transport of errors under the scheme is largely determined by what the scheme would do if used to solve the variational equation

$$v'(t) = Df(t, y(t))\, v(t)$$



note that v here represents the transport of any of the truncation errors, e.g.

$v_1 = \tau_0, \ldots, v_n = \circledast$

We can learn a lot about how the scheme will behave on non-autonomous linear problems by studying the constant coefficient case

$$v' = Bv \quad , \quad B \text{ a constant matrix}$$

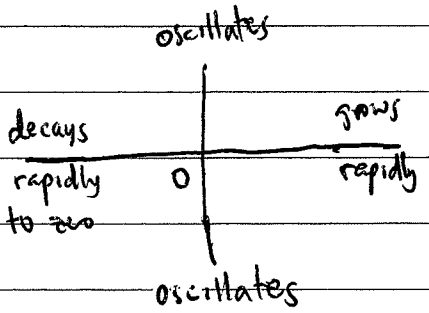
This in turn boils down to the scalar case $v' = \lambda v$, $\lambda \in \mathbb{C}$.
This simple equation holds the key to understanding stiff ODE's...

Linear stability analysis

simplest possible linear ODE: $y' = \lambda y$ (scalar, const. coeff.)

want to know which values of h, λ cause the numerical solution to remain bounded as $n \to \infty$

exact soln: $y(t) = Ce^{\lambda t}$ λ in complex plane:



for Euler, we have

$$y_{n+1} = y_n + h\lambda y_n = (1+h\lambda) y_n$$

$$\to y_n = (1+h\lambda)^n y_0 = R(h\lambda)^n y_0$$

$R(z) = 1 + z$ = stability function of Euler's method.

the numerical solution

diverges exponentially if $h\lambda \in \{z \in \mathbb{C} : |R(z)| > 1\}$

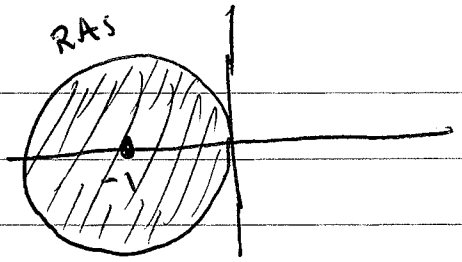
remains bounded if $h\lambda \in \{z \in \mathbb{C} : |R(z)| \leq 1\}$

converges to zero as $n \to \infty$ if $h\lambda \in \{z \in \mathbb{C} : |R(z)| < 1\}$

<u>def</u>: the region of absolute stability (or stability domain) is

$$RAS = \{z \in \mathbb{C} : |R(z)| \leq 1\}$$

Euler: $R(z) = 1+z$



Every Runge-Kutta method has a stability function

$$k_i = f\left(y_n + h\left(a_{i1}k_1 + \cdots + a_{is}k_s\right)\right)$$
$$= \lambda y_n + h\lambda \sum_j a_{ij}k_j$$

$$(I - h\lambda A)k = \lambda y_n e \qquad e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad k = \begin{pmatrix} k_1 \\ \vdots \\ k_s \end{pmatrix}$$

scalar case $k_i \in \mathbb{C}$

$$k = \lambda(I - h\lambda A)^{-1} e\, y_n$$

$$y_{n+1} = y_n + hb^T k = \underbrace{\left[1 + h\lambda b^T (I - h\lambda A)^{-1} e\right]}_{R(h\lambda)} y_n$$

$$y_n = R(h\lambda)^n y_0$$

The stability function is a rational function of the form

$$R(z) = \frac{P(z)}{Q(z)} \quad \text{with} \quad \deg P \le s, \ \deg Q \le s$$

pf: $(I - zA)^{-1} = \dfrac{\mathrm{adj}(I - zA)}{\det(I - zA)}$

$\mathrm{adj}(B)$ = adjugate of B = transpose of cofactor matrix C

$C_{ij} = (-1)^{i+j} \det(M_{ij})$

M_{ij} = matrix minor (with row i and column j deleted)

Since $\det(I - zA)$ is a polynomial in z of degree $\le s$ while $\mathrm{adj}(I - zA)_{ij}$ has degree $\le s-1$, the result follows from the formula $R(z) = 1 + z\, b^T (I - zA)^{-1} e$

if the method is explicit, $I - zA$ is lower triangular with ones on the diagonal, hence $\det(I - zA) = 1$ and $R(z)$ is a polynomial of degree $\le s$.

another way to see this:

$$(I - zA)^{-1} = \sum_{j=0}^{\infty} (zA)^j = I + zA + \cdots + z^{s-1}A^{s-1}$$

↑ series terminates $(A^s = A^{s+1} = \cdots = 0)$

$\therefore$ $R(z) = 1 + z b^T (I - zA)^{-1} e$ is a polynomial of deg $\le s$

__Example:__ explicit midpoint rule

0		
1/2	1/2	
	0	1

$$(I - zA)^{-1} = I + z\begin{pmatrix} 0 & 0 \\ 1/2 & 0 \end{pmatrix}$$

$$b^T(\quad)^{-1}e = b^T e + z \underbrace{b^T A e}_{c} = 1 + \frac{z}{2}$$

(• b–c ← order conditions)

$$R(z) = 1 + z b^T (\quad)^{-1} e = 1 + z + \frac{z^2}{2}$$

it's no accident that these are the first 3 terms of $e^z = 1 + z + \cdots$

__Theorem:__ the stability function of an RK method of order p satisfies

$$R(z) = e^z + O(z^{p+1})$$

proof: the expansion $(I - zA)^{-1} = I + zA + z^2A^2 + \cdots$ is valid for small z. So

$$R(z) = 1 + z\,b^T e + z^2 b^T A e + z^3 b^T A^2 e + z^4 b^T A^3 e + \cdots$$

$$= 1 + z\underbrace{b^T e}_{1} + z^2\underbrace{b^T c}_{1/2} + z^3\underbrace{b^T A c}_{1/6} + z^4\underbrace{b^T A^2 c}_{1/24} + \cdots$$

for each of these graphs through order p, $\sum_i b_i \Phi_i(\phi) = \frac{1}{\gamma(\phi)}$

if ϕ has q nodes and no ramifications, then $\gamma(\phi) = q!$

so $R(z) = 1 + z + \frac{z^2}{2!} + \cdots + \frac{z^p}{p!} + O(z^{p+1})$

alternative (direct) proof:

for the ODE $y' = \lambda y$, the truncation error is

$$\tau_n = \underbrace{y(t_{n+1})}_{e^{\lambda(t_n+h)}} - \underbrace{\left[y(t_n) + h\Phi(y(t_n), h)\right]}_{R(h\lambda)\,y(t_n) \;\leftarrow\; y(t_n) = e^{\lambda t_n}}$$

$$= \left[e^{\lambda h} - R(h\lambda)\right] e^{\lambda t_n} = O(h^{p+1})$$

$\uparrow$ method is order p

$\therefore R(z) = e^z + O(z^{p+1})$

Note: for a multistep method, we had

$$\text{method is order } p \iff \rho(e^h) - h\sigma(e^h) = O(h^{p+1})$$
$$\iff \rho(z) - \ln z\, \sigma(z) = O(|z-1|^{p+1})$$

but for Runge-Kutta we only have

$$\text{method is order } p \implies R(z) = e^z + O(z^{p+1})$$

(not $\Leftarrow$)

Corollary: every s-stage s-order explicit RK method has the stability function

$$R(z) = 1 + z + \cdots + \frac{z^s}{s!}$$

↖ (Note: for $s \geq 5$, there is no ERK method of order $p = s$, so this statement is not very general...)

proof: on one hand it's a polynomial of degree $\leq s$
on the other hand it satisfies $R(z) = e^z + O(z^{s+1})$
this condition uniquely determines the first $s+1$ terms.

Corollary: An s-stage ERK can't be of order $\geq s+1$.

proof: this would require $R(z) = 1 + \cdots + \frac{z^s}{s!} + \frac{z^{s+1}}{(s+1)!} + O(z^{s+2})$

can't match this term ↗
since $\deg R(z) \leq s$

Note the region of absolute stability $RAS = \{z \in \mathbb{C} : |R(z)| \leq 1\}$ of an explicit scheme tends to increase as p gets bigger, which is surprising- (usually higher order means less stable)

reason: $e^z = 1 + z + \frac{z^2}{2} + \cdots$ is a convergent series

and $\{z : |e^z| \leq 1\} = \underbrace{\{z : \mathrm{Re}\, z \leq 0\}}_{\mathbb{C}^-}$ ← left half of complex plane

as p increases, $R(z)$ contains more and more terms in the series, and being close to e^z means being less than 1 in magnitude for $z \in \mathbb{C}^-$. (this reasoning breaks down somewhat for $p \geq 5$ since $S > p$ after that)

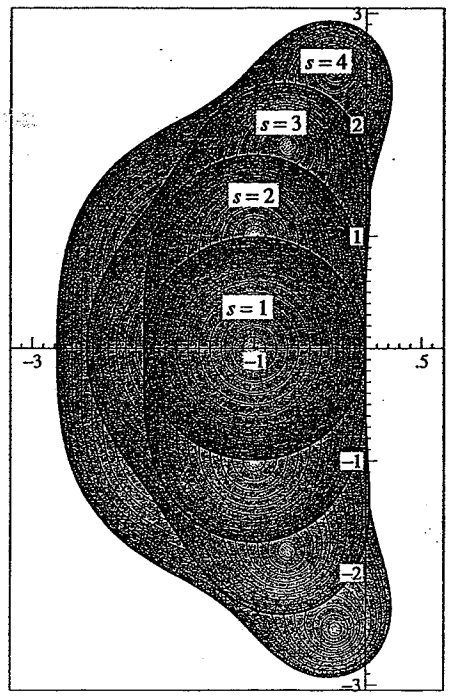


Fig. 2.1. Stability domains for explicit Runge-Kutta methods of order $p = s$

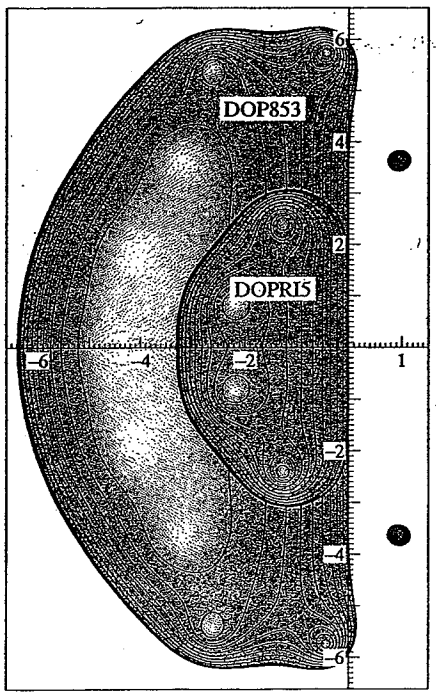


Fig. 2.2. Stability domains for DOPRI methods

the stability regions grow for $p = 1, 2, 3, 4$ but the one for DOPRI5 does not fully contain RK4 since the 6th stage of DOPRI5 adds an extra term:

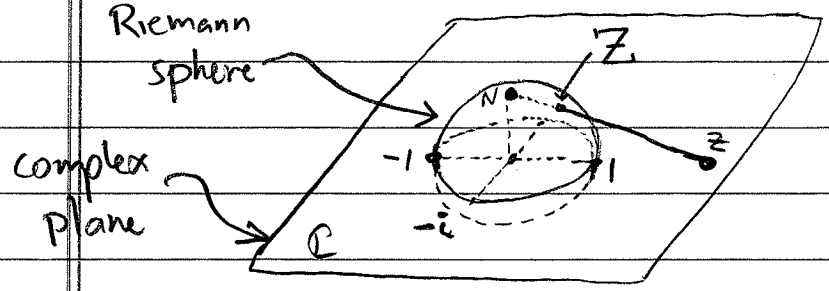
$R(z) = 1 + z + \cdots + \frac{z^5}{120} + \frac{z^6}{600}$

would be $\frac{z^6}{720}$ for 6th order.

def: A method is A-stable if the RAS includes $\mathbb{C}^-$

note: no explicit method is A-stable since $R(z)$ is a polynomial (polynomials diverge as $z \to \infty$)

in complex analysis, ∞ is thought of as a single point that you add to the complex plane to make it look topologically like a sphere. (in $\mathbb{R}$, we usually distinguish between $\pm\infty$, but in $\mathbb{C}$ there is only one ∞)



straight line to north pole sets up correspondence between points on sphere and complex numbers. The north pole itself corresponds to infinity.

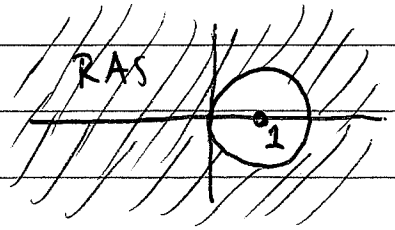
Examples of A-stable methods

backward Euler: $y_{n+1} = y_n + h\underbrace{f(y_{n+1})}_{\lambda y_{n+1}}$

$$(1-h\lambda)y_{n+1} = y_n$$

$$y_{n+1} = (1-h\lambda)^{-1} y_n \quad , \quad R(z) = \frac{1}{1-z}$$

$$|R(z)| \le 1 \iff \frac{1}{|R(z)|} \ge 1 \iff |1-z| \ge 1$$



so $\mathbb{C}^- \subseteq$ RAS. $\therefore$ A-stable.

every B-stable method is A-stable.

proof: if $\lambda \in \mathbb{C}^-$, $y' = \lambda y$ is contractive.

$$\frac{d}{dt}\frac{1}{2}|y(t)-z(t)|^2 = \frac{1}{2}\Big[(y'(t)-z'(t))\cdot(\overline{y(t)-z(t)}) + (\overline{y'(t)-z'(t)})\cdot(y(t)-z(t))\Big]$$

$$= \mathrm{Re}\Big[(f(y(t))-f(z(t)))\cdot(\overline{y(t)-z(t)})\Big]$$

$$= \mathrm{Re}\Big[\lambda|y(t)-z(t)|^2\Big] = (\mathrm{Re}\,\lambda)|y(t)-z(t)|^2 \le 0$$

$\therefore$ $|y_{n+1}-0| \le |y_n-0|$

↑ ↑ two solutions

$\therefore$ $|R(\lambda h)y_n| \le |y_n|$

$\therefore$ $|R(\lambda h)| \le 1$ $\therefore$ $|R(z)| \le 1$ for $z \in \mathbb{C}^-$

$\therefore$ method is A-stable.

so backward Euler and implicit midpoint are both A-stable.

the trapezoidal rule is A-stable but not B-stable.

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

$$\left(1 - \frac{h\lambda}{2}\right) y_{n+1} = \left(1 + \frac{h\lambda}{2}\right) y_n$$

$$R(z) = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}$$

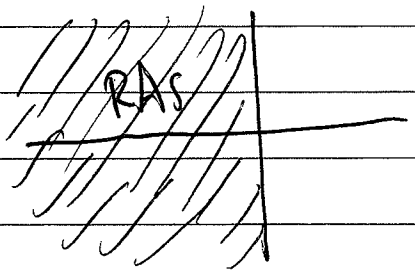
$$|R(z)| \le 1 \iff \left|1 + \frac{z}{2}\right| \le \left|1 - \frac{z}{2}\right| \qquad z = x + iy$$

$$\iff \left(1 + \frac{x}{2}\right)^2 + \left(\frac{y}{2}\right)^2 \le \left(1 - \frac{x}{2}\right)^2 + \left(\frac{y}{2}\right)^2$$

$$\iff 1 + x + \frac{x^2}{4} \le 1 - x + \frac{x^2}{4}$$

$$\iff x \le 0$$

So $RAS = \mathbb{C}^-$ in this case



So if you solve $y' = \lambda y$ with $\mathrm{Re}\,\lambda \le 0$, the numerical solution remains bounded no matter how big your stepsize h is chosen. This is the same behavior that the exact solution $y(t) = Ce^{\lambda t}$ exhibits, except that for $\mathrm{Re}\,\lambda \ll 0$, the decay rate of the exact solution is very fast while $R(z) \to 1$ as $z \to \infty$ (so $|R(h\lambda)|$ increases to 1 as $\mathrm{Re}\,\lambda \to -\infty$ holding h fixed)

last time:

stability function of a scheme: $y_{n+1} = R(h\lambda) y_n \quad (y' = \lambda y)$

$R(z) = 1 + z b^T (I - zA)^{-1} e, \quad e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

Region of Absolute Stability: $RAS = \{ z \in \mathbb{C} : |R(z)| \le 1 \}$

A scheme is A-stable if $\mathbb{C}^- \subseteq RAS$

B-stable schemes are A-stable (Backward-Euler, implicit midpt.)

the trapezoidal rule is A-stable but not B-stable.

$=$

why do we care about A-stability?

example: $y' = By$, $B = \begin{pmatrix} -100 & 0 & 0 \\ 101 & 0 & 1 \\ 99 & -1 & 0 \end{pmatrix}$, $y(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

diagonalize B: $\det(\lambda I - B) = (\lambda + 100)(\lambda^2 + 1)$

find eigenvectors:

$$B \underbrace{\begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & i & -i \end{pmatrix}}_{U} = \begin{pmatrix} 100 & 0 & 0 \\ -100 & i & -i \\ -100 & -1 & -1 \end{pmatrix} = U \begin{pmatrix} -100 & & \\ & i & \\ & & -i \end{pmatrix}$$

$U \leftarrow$ columns are eigenvectors

(found by finding nullspaces of $B - \lambda_i I$, $\lambda_1 = -100, \lambda_2 = i, \lambda_3 = -i$)

$$U^{-1} = \frac{1}{2} \begin{pmatrix} -2 & 0 & 0 \\ 1-i & 1 & -i \\ 1+i & 1 & i \end{pmatrix}$$

$$y(t) = e^{Bt} y_0 = U e^{\Lambda t} U^{-1} y_0 = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & i & -i \end{pmatrix} \begin{pmatrix} e^{-100t} & & \\ & e^{it} & \\ & & e^{-it} \end{pmatrix} \begin{pmatrix} -1 \\ \frac{1-i}{2} \\ \frac{1+i}{2} \end{pmatrix}$$

it's a bit nicer to work with real numbers when B is real.

In general, if $\lambda = \alpha \pm i\beta$ is a complex conjugate pair of eigenvalues, the eigenvectors will also be conjugates:

$$[B - (\alpha \pm i\beta)I](u \pm iv) = 0$$

$$\begin{aligned} Bu &= \alpha u - \beta v \\ Bv &= \beta u + \alpha v \end{aligned} \qquad B(u,v) = (u,v)\begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}$$

In our case, $\alpha = 0, \beta = 1$, $u = \mathrm{Re}\begin{pmatrix} 0 \\ i \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$, $v = \mathrm{Im}\begin{pmatrix} 0 \\ i \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$$B \underbrace{\begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}}_{U} = U \underbrace{\begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}}_{\Lambda}$$

$$U^{-1} = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$B = U\Lambda U^{-1}$ ← real "diagonal" form

exact solution:

$$y(t) = e^{Bt} y_0 = U e^{\Lambda t} U^{-1} y_0$$

$$= \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} e^{-100t} & & \\ & \cos t & \sin t \\ & -\sin t & \cos t \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} e^{-100t} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \cos t + \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \sin t$$

If $AB = BA$ then $e^{A+B} = e^A e^B$

so $\exp\left[\alpha\begin{pmatrix} 1 & \\ & 1 \end{pmatrix} + \beta\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\right]$

$= e^{\alpha} \exp\begin{pmatrix} 0 & \beta \\ -\beta & 0 \end{pmatrix}$

$= e^{\alpha}\left(I + \beta\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} - \frac{\beta^2}{2} I - \frac{\beta^3}{6}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} + \cdots\right)$

$= e^{\alpha}\begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix}$

The numerical solution can also be found by diagonalizing B

claim: the numerical solution of $y' = By$ satisfies $y_{n+1} = R(hB)y_n$

$$\text{So } B = U\Lambda U^{-1} \Rightarrow y_n = U R(h\Lambda)^n U^{-1} y_0$$

proof: in the scalar case $y' = \lambda y$, the solution of

$$k_i = \lambda y_n + h\lambda \sum_j a_{ij} k_j \qquad 1 \le i \le s$$

is given by $k_i = e_i^T (I - h\lambda A)^{-1} e \, \lambda y_n = \dfrac{P_i(h\lambda)}{Q(h\lambda)} \lambda y_n$

where $Q(z) = \det(I - zA)$, $P_i(z) = e_i^T \operatorname{adj}(I - zA) e$

are polynomials with $\deg Q(z) \le s$, $\deg P_i(z) \le s-1$

and $e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ ← ith slot, $e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$. (proved in Lec 18)

I claim that in the system case, the solution of

$$k_i = By_n + hB \sum_j a_{ij} k_j$$

is $k_i = Q(hB)^{-1} P_i(hB) B y_n$

where $Q(hB)^{-1}$ is the matrix inverse of $Q(hB) = q_0 I + \cdots + q_s (hB)^s$

coefficients of $Q(z) = q_0 + q_1 z + \cdots + q_s z^s$

to show this, we use the key property of the adjugate matrix

$$(I - zA)\,\mathrm{adj}(I-zA) = \underbrace{\det(I-zA)}_{Q(z)} I$$

which implies

$$P_i(z) = e_i^T \mathrm{adj}(I-zA)e = e_i^T \Big[Q(z)I + zA\,\mathrm{adj}(I-zA) \Big] e$$

(insert $I = \sum_j e_j e_j^T$)

$$= Q(z) + z \sum_{j=1}^{s} a_{ij} P_j(z)$$

Since the left and right sides are just polynomials in z, we can replace z by the matrix hB and the equation will still hold.

$$\therefore \quad P_i(hB) = Q(hB) + hB \sum_j a_{ij} P_j(hB)$$

finally, we apply these matrices to By_n and left multiply by $Q(hB)^{-1}$ to obtain

(B and $Q(hB)^{-1}$ commute)

$$\underbrace{Q(hB)^{-1} P_i(hB) By_n}_{k_i} = By_n + hB \sum_j a_{ij} \underbrace{Q(hB)^{-1} P_j(hB) By_n}_{k_i}$$

The final update $y_{n+1} = y_n + h\sum_i b_i k_i$ yields

$$y_{n+1} = R(hB) y_n \quad \text{with} \quad R(hB) = I + hB \frac{\sum_i b_i P_i(hB)}{Q(hB)} = \frac{P(hB)}{Q(hB)}$$

where $R(z) = 1 + z b^T (I-zA)^{-1} e = \dfrac{P(z)}{Q(z)}$

(note: $\frac{P(hB)}{Q(hB)}$ means either $Q(hB)^{-1}P(hB)$ or $P(hB)Q(hB)^{-1}$. They're equal)

back to example: $y' = By$, $B = \begin{pmatrix} -100 & 0 & 0 \\ 101 & 0 & 1 \\ 99 & -1 & 0 \end{pmatrix}$, $y_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

exact solution: $y(t) = U e^{\Lambda t} U^{-1} y_0 = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & i & -i \end{pmatrix} \begin{pmatrix} e^{-100t} & & \\ & e^{it} & \\ & & e^{-it} \end{pmatrix} \begin{pmatrix} -1 \\ \frac{1-i}{2} \\ \frac{1+i}{2} \end{pmatrix}$

numerical solution: $y_n = U R(h\Lambda)^n U^{-1} y_0 = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & i & -i \end{pmatrix} \begin{pmatrix} R(-100h)^n & & \\ & R(ih)^n & \\ & & R(-ih)^n \end{pmatrix} \begin{pmatrix} -1 \\ \frac{1-i}{2} \\ \frac{1+i}{2} \end{pmatrix}$

we want to take timesteps small enough that

$$R(\pm ih)^n \approx e^{\pm inh} \quad \leftarrow \text{accuracy requirement}$$

and $R(-100h)^n \approx 0$ $\leftarrow$ stability requirement (large steps OK as long as $-100h \in RAS$)

this example also shows why we care about complex $\lambda \in \mathbb{C}$. although B is a real matrix, it has complex eigenvalues, and the accuracy and stability of the method is determined by $R(\lambda_j)$. (The final formula for y_n is actually real, of course) If we used real "diagonal" form instead and $\alpha \pm i\beta$ were eigenvalues, we'd be faced with evaluating 2×2 matrices:

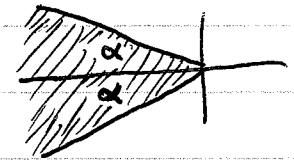
$$R\begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} = R\left(\alpha \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \beta \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}\right) = \underbrace{\frac{R(\alpha+i\beta)+R(\alpha-i\beta)}{2}}_{\text{should be close to } e^{\alpha}\cos\beta} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \underbrace{\frac{R(\alpha+i\beta)-R(\alpha-i\beta)}{2i}}_{\text{should be close to } e^{\alpha}\sin\beta} \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}$$

$$R\begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} \begin{pmatrix} R(\alpha+i\beta) & \\ & R(\alpha-i\beta) \end{pmatrix} \begin{pmatrix} 1/2 & -i/2 \\ 1/2 & i/2 \end{pmatrix}$$

there are a few other types of stability.

a method is <u>L-stable</u> if it is A-stable and $\lim_{z\to\infty} R(z) = 0$.

note: $\lim_{z\to\infty} R(z) = 0 \iff R(z) = \frac{P(z)}{Q(z)}$ with $\deg Q(z) > \deg P(z)$

a method is A(α) stable if the RAS contains the wedge 

A method is I-stable if $|R(z)| \leq 1$ for z on the imaginary axis. We care about I-stability because it's easier to check than A-stability and:

<u>theorem</u>: A method is A-stable iff it is I-stable and all the poles of $R(z)$ have positive real part.

<u>proof</u>: $\Rightarrow$ rational functions are continuous except at their poles, where they approach infinity. So

$|R(z)| \leq 1$ for $z \in \mathbb{C}^-$ $\Rightarrow$ no poles $z_0 \in \mathbb{C}^-$

and $\Rightarrow$ if $t_0 \in \mathbb{R}$ then

$$|R(it_0)| = \lim_{\substack{z\to it_0 \\ z\in\mathbb{C}^-}} |R(z)| \leq 1$$

by continuity

$\Leftarrow$ Let $R(z) = \dfrac{P(z)}{Q(z)} = \dfrac{p_0 + p_1 z + \cdots + p_\alpha z^\alpha}{q_0 + q_1 z + \cdots + q_\beta z^\beta}$, $p_\alpha \neq 0$, $q_\beta \neq 0$

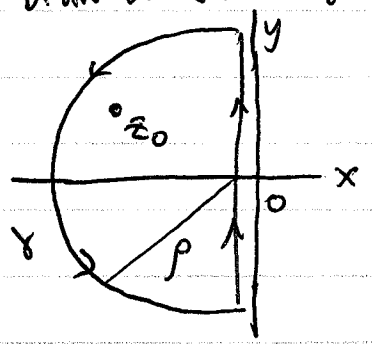
I-stability $\Rightarrow$ $\alpha \leq \beta$ and if $\alpha = \beta$ then $|p_\alpha| \leq |q_\beta|$

(otherwise $\lim_{t\to\infty} |R(it)|$ would blow up ($\alpha > \beta$) or approach $\frac{|p_\alpha|}{|q_\beta|} > 1$ ($\alpha = \beta$))

so $\lim_{z\to\infty} |R(z)|$ exists and is ≤ 1.

↖ in any direction, not just along imaginary axis

Since $R(z)$ has no poles in $\mathbb{C}^-$, it is analytic in $\mathbb{C}^-$ and continuous on $\overline{\mathbb{C}^-} = \{z : \operatorname{Re} z \leq 0\}$. Suppose there were a point $z_0 \in \mathbb{C}^-$ such that $|R(z_0)| > 1$. Draw a curve γ around z_0 as shown.



choose ρ large enough that

$$|R(z)| \leq 1 + \frac{R(z_0) - 1}{2}$$

for all z satisfying $|z| = \rho$. (possible since $\lim_{z\to\infty} |R(z)| \leq 1$)

Since $|R(z)| \leq 1$ along the imaginary axis, for all points $z \in \gamma$,

$$|R(z)| \leq 1 + \frac{R(z_0) - 1}{2} < R(z_0)$$

this contradicts the maximum principle (if $f(z)$ is analytic in a region Ω and continuous on $\overline{\Omega}$ then $|f(z)|$ achieves its maximum on the boundary of Ω, not in the interior.)

example: suppose you determined that for your favorite scheme,

$$R(z) = \frac{1 + 2z/5 + z^2/20}{1 - \frac{3z}{5} + \frac{3z^2}{20} - \frac{z^3}{60}} = \frac{P(z)}{Q(z)}$$

I-stability

to check I-stability, we have to determine if

$$E(y) = |Q(iy)|^2 - |P(iy)|^2 = Q(iy)Q(-iy) - P(iy)P(-iy)$$

is non-negative for all y. Expanding this out, we obtain

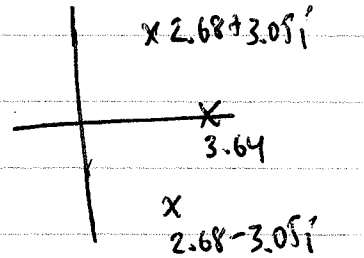
$$E(y) = \frac{y^6}{3600} \geq 0 \quad \text{as required.}$$

poles of R

we can factor $Q(z)$ with the help of a computer:

$$Q(z) = -\frac{1}{60}(z - 3.63783)(z - (2.68108 + 3.05043i))(z - (2.68 - 3.05i))$$

thus the poles of $R(z)$ have positive real part.



∴ method is A-stable

since $\deg(P) = 2 < \deg(Q) = 3$ $\quad \lim_{z \to \infty} R(z) = 0$

∴ method is L-stable.

Prothero Robinson equation

$$y' = \lambda(y - \varphi(t)) + \varphi'(t)$$
$$y(0) = \varphi(0)$$

exact solution: $y(t) = \varphi(t)$

(more generally, if $y(0) = \xi$
$y(t) = e^{\lambda t}(\xi - \varphi(0)) + \varphi(t)$)

how do we implement an implicit method for this equation?

want: $k_i = f\left(t_n + c_i h,\ y_n + h\sum_{j=1}^{s} a_{ij} k_j\right)$

$$= \lambda\left[y_n + h\sum_j a_{ij}k_j - \varphi(t_n + c_i h)\right] + \varphi'(t_n + c_i h)$$

solution:

$$\vec{k} = (I - h\lambda A)^{-1}\left[\lambda y_n e - \lambda\varphi(t_n + \vec{c}h) + \varphi'(t_n + \vec{c}h)\right]$$

$e = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}$ ↑ $\begin{pmatrix}\varphi(t_n + c_1 h)\\ \vdots \\ \varphi(t_n + c_s h)\end{pmatrix}$ same idea

$$y_{n+1} = y_n + h b^T k$$ ← how to advance solution numerically

for error analysis, want recursion for the error. Goal: simplify RHS

$$y_{n+1} - \varphi(t_{n+1}) = y_n - \varphi(t_n) + \varphi(t_n) - \varphi(t_n + h)$$
$$+ h b^T (I - h\lambda A)^{-1}\left[\lambda y_n e - \lambda\varphi(t_n + \vec{c}h) + \varphi'(t_n + \vec{c}h)\right]$$

next we define the stage errors Δ_0, $\Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_s \end{pmatrix}$ via

$$\varphi(t_n + c_j h) = \varphi(t_n) + h \sum_\ell a_{j\ell} \varphi'(t_n + c_\ell h) + \Delta_j \qquad 1 \le j \le s$$

$$\varphi(t_n + h) = \varphi(t_n) + h \sum_\ell b_\ell \varphi'(t_n + c_\ell h) + \Delta_0$$

abscissas — weights

these measure how accurately the quadrature formulas $(c_\ell, a_{j\ell})$, (c_ℓ, b_ℓ) integrate the exact solution from t_n to $t_n + c_\ell h$ or $t_n + h$.

the bracketed term in the formula for y_{n+1} becomes

$$\left[\lambda y_n e - \lambda\left(\varphi(t_n) e + hA\varphi'(t_n + \vec{c}h) + \Delta \right) + \varphi'(t_n + \vec{c}h) \right]$$

$$= \lambda(y_n - \varphi(t_n)) e + (I - h\lambda A)\varphi'(t_n + \vec{c}h) - \lambda\Delta$$

and we learn that

$$y_{n+1} - \varphi(t_{n+1}) = \left[1 + h\lambda b^T (I - h\lambda A)^{-1} e \right] (y_n - \varphi(t_n))$$

$$-\Delta_0 \longrightarrow \qquad + h b^T \varphi'(t_n + \vec{c}h) + \varphi(t_n) - \varphi(t_n + h)$$

$$- h\lambda b^T (I - h\lambda A)^{-1} \Delta$$

or $$y_{n+1} - \varphi(t_{n+1}) = R(h\lambda)(y_n - \varphi(t_n)) - \tau_n$$

$$\tau_n = h\lambda b^T (I - h\lambda A)^{-1} \Delta + \Delta_0$$

the usual backward iteration (with $e_n = y_n - \varphi(t_n)$, $z = h\lambda$) gives

$$\begin{aligned} e_n &= R(z)e_{n-1} - \tau_{n-1} \\ &\vdots \\ &= R(z)^n \underbrace{e_0}_{0} - R(z)^{n-1}\tau_0 - \cdots - R(z)\tau_{n-2} - \tau_{n-1} \end{aligned}$$

take absolute value and add it all up

$$\begin{aligned} |e_n| &\le \left(1 + |R(z)| + \cdots + |R(z)|^{n-1}\right) \max\left(|\tau_0|, \ldots, |\tau_{n-1}|\right) \\ &= \frac{1 - |R(z)|^n}{1 - |R(z)|} \max_j |\tau_j| \\ &\le \min\left(n, \frac{1}{1-|R(z)|}\right) \max_j |\tau_j| \quad \leftarrow \text{if } z \in RAS \text{ i.e. } R(z) \le 1 \end{aligned}$$

compare this to the Euler proof:

$$\begin{aligned} |e_n| &\le \left(1 + (1+hL) + \cdots + (1+hL)^{n-1}\right) \max\left(|\tau_0|, \ldots, |\tau_{n-1}|\right) \\ &\le \frac{(1+hL)^n - 1}{(1+hL) - 1} \max_j |\tau_j| \le \frac{e^{Lt_n} - 1}{L} \max_j \frac{|\tau_j|}{h} \end{aligned}$$

so in the new proof, we don't have exponential growth in the prefactor <u>and</u> we haven't lost a power of h in the truncation errors! (for fixed λ, as $h \to 0$ $R(h\lambda) \to 1$ and $\min\left(n, \frac{1}{1-|R(z)|}\right)$ becomes $n = \frac{t_n}{h}$. But for stiff problems we're hoping to make the $|\tau_j|$ small with $h\lambda$ large and negative so that $\frac{1}{1-|R(z)|} = O(1)$)

Now let's estimate $\Delta_0, \Delta_1, \Delta_2$ for a 2-stage method $\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$

Taylor expand and match terms

$$\Delta_0 = \overbrace{\varphi(t_n+h)}^{\varphi + h\varphi' + \cdots} - \varphi(t_n) - h\sum_j b_j \overbrace{\varphi'(t_n + c_j h)}^{\varphi' + (c_j h)\varphi'' + \frac{(c_j h)^2}{2}\varphi''' + \cdots}$$

$$= \underbrace{(\varphi - \varphi)}_{0} + \underbrace{\left(1 - \sum_j b_j\right) h\varphi'}_{0} + \underbrace{\left(1 - 2\sum_j b_j c_j\right)\frac{h^2}{2}\varphi''}_{0 \text{ if method 2nd order}}$$

$$+ \underbrace{\left(1 - 3\sum_j b_j c_j^2\right)\frac{h^3}{6}\varphi'''}_{0 \text{ if method is 3rd order}} + O(h^4)$$

the coefficients of Δ_0 correspond to the bushy trees $\bullet$  etc

the order conditions for these trees determine the order of the quadrature rule with abscissas c_j and weights b_j when solving $y' = f(t)$ ← no dependence on y

next we compute Δ_1, Δ_2:

$$\Delta_i = \varphi(t_n + c_i h) - \varphi(t_n) - h\sum_j a_{ij}\varphi'(t_n + c_j h)$$

$$= \overbrace{(\varphi - \varphi)}^{0} + \overbrace{\left(c_i - \sum_j a_{ij}\right) h\varphi'}^{0} + \left(c_i^2 - 2\sum_j a_{ij} c_j\right)\frac{h^2}{2}\varphi''$$

$$+ \left(c_i^3 - 3\sum_j a_{ij} c_j^2\right)\frac{h^3}{6}\varphi''' + O(h^4)$$

→ these are new conditions measuring how well the ith row of A works as a quadrature rule for $\int_{t_n}^{t_n + c_i h} \varphi'(t)\,dt$.
(they are not automatically zero for a high order method)

for SDIRK, we have $\begin{array}{c|cc} \beta & \beta & \\ 1-\beta & 1-2\beta & \beta \\ \hline & 1/2 & 1/2 \end{array}$ which gives

$$\Delta_1 = \left(\beta^2 - 2\beta^2\right)\frac{h^2}{2}\varphi'' + \cdots = -\frac{\beta^2 h^2}{2}\varphi''(t_n) + O(h^3)$$

$$\Delta_2 = \left((1-\beta)^2 - 2\left[(1-2\beta)\beta + \beta(1-\beta)\right]\right)\frac{h^2}{2}\varphi'' + \cdots$$

$$= \left(7\beta^2 - 6\beta + 1\right)\frac{h^2}{2}\varphi''(t_n) + O(h^3)$$

$$\Delta_0 = \left(1 - 3\left(\frac{1}{2}\beta^2 + \frac{1}{2}(1-\beta)^2\right)\right)\frac{h^3}{6}\varphi''' + \cdots$$

$$= \underbrace{\left(-\frac{1}{2} + 3\beta - 3\beta^2\right)}_{\text{zero if } \beta = \frac{3\pm\sqrt{3}}{6}}\frac{h^3}{6}\varphi'''(t_n) + O(h^4)$$

if $z = O(h)$, then $\tau_n = \Delta_0 + z\, b^T (I - zA)^{-1}\Delta$

$$\approx \Delta_0 + z b^T \Delta = O(h^{p+1})$$

$$p = \begin{cases} 3 & \gamma = \frac{3 \pm \sqrt{3}}{6} \\ 2 & \text{o.w.} \end{cases}$$

but if z is large and negative, the order is reduced

$$(I - zA)^{-1} \overset{|z|>>1}{\approx} (-zA)^{-1} = -z^{-1}\begin{pmatrix} \beta & 0 \\ 1-2\beta & \beta \end{pmatrix}^{-1} = \frac{-z^{-1}}{\beta^2}\begin{pmatrix} \beta & 0 \\ 2\beta - 1 & \beta \end{pmatrix}$$

so $\tau_n = \Delta_0 + z b^T (I - zA)^{-1} \Delta$

$$\approx \Delta_0 - b^T \left[\frac{1}{\beta^2} \begin{pmatrix} \beta & 0 \\ 2\beta - 1 & \beta \end{pmatrix} \right] \Delta$$

$$\approx \underbrace{\Delta_0}_{O(h^3)} - \frac{1}{\beta^2} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \beta & 0 \\ 2\beta - 1 & \beta \end{pmatrix} \begin{pmatrix} -\frac{\beta^2 h^2}{2} \varphi''(t_n) \\ (7\beta^2 - 6\beta + 1) \frac{h^2}{2} \varphi''(t_n) \end{pmatrix} + O(h^3)$$

$$= \frac{h^2}{4\beta} (1 - 5\beta + 4\beta^2) \varphi''(t_n) + O(h^3)$$

so τ_n degrades from $\begin{cases} O(h^4) & \beta = \frac{3 \pm \sqrt{3}}{6} \\ O(h^3) & \text{o.w.} \end{cases}$ to $\begin{cases} O(h^3) & \beta = 1, \frac{1}{4} \\ O(h^2) & \text{o.w.} \end{cases}$

when $|\lambda| \gg \frac{1}{h}$

Note: it may seem strange to change the ODE as the mesh is refined, but this is exactly what happens when you discretize PDE's. Refinement in the space variables leads to stiffer problems.

final global error estimates:

$$|y_n - \varphi(t_n)| \leq \underbrace{\frac{1 - |R(z)|^n}{1 - |R(z)|}}_{C + O(\frac{1}{z})} \underbrace{\max(|\tau_0|, \dots, |\tau_{n-1}|)}_{\frac{h^2}{4\beta}(1 - 5\beta + 4\beta^2)\varphi''(t_n) + O(h^3)}$$

$$R(z) = \frac{1 + z(1 - 2\beta) + z^2(\beta^2 - 2\beta + \frac{1}{2})}{(1 - \beta z)^2}$$

β	$\lim_{z\to\infty} \lvert R(z)\rvert$	C
$\frac{3-\sqrt{3}}{6}$	2.73	blows up
$\frac{3+\sqrt{3}}{6}$	0.732	3.732
$1\pm\frac{\sqrt{2}}{2}$	0	1

L stability makes the constant C smaller, but doesn't change the order of convergence. (Best choice of β for this problem: $\beta = 1$ or $1/4$ makes $\tau_n = O(h^3)$. both are A-stable).

Also note that although $1+\frac{\sqrt{2}}{2}$ is L-stable, it has the undesirable feature that $c_1 = \beta > 1$ and $c_2 = 1-\beta < 0$, i.e. the quadrature points are outside the interval $[0,1]$. (doesn't seem to cause trouble, though...)

summary: if $z = h\lambda$ is not small, the order conditions we derived using trees to make τ_n small are no longer necessary or sufficient. Instead, we found that $1-5\beta+4\beta^2$ needs to be zero for

$$\tau_n = \Delta_0 + zb^T(I-zA)^{-1}\Delta = O(h^3)$$

in the $h \ll 1$, $\lambda \ll -\frac{1}{h}$ regime.

for Radau IIa, we have

1/3	5/12	-1/12
1	3/4	1/4
	3/4	1/4

(3rd order method)

so $\Delta_0 = O(h^4)$

$$\Delta_1 = \underbrace{\left(c_1^2 - 2\sum_j a_{1j} c_j\right)}_{0} \frac{h^2}{2}\varphi'' + \underbrace{\left(c_1^3 - 3\sum_j a_{1j} c_j^2\right)}_{4/27} \frac{h^3}{6}\varphi''' + \cdots$$

$$\Delta_1 = \frac{2}{81} h^3 \varphi'''$$

$$\Delta_2 = \Delta_0 = O(h^4) \quad (\text{since } a_{2j} = b_j)$$

and therefore

$$\tau_n = \overbrace{(0\ 1)\begin{pmatrix}\Delta_1\\ \Delta_2\end{pmatrix}}^{\Delta_0} + z b^T (I - zA)^{-1} \begin{pmatrix}\Delta_1\\ \Delta_2\end{pmatrix}$$

$$= \left[(0\ 1)(I - zA) + z b^T\right](I - zA)^{-1}\begin{pmatrix}\Delta_1\\ \Delta_2\end{pmatrix}$$

(cancel since b^T = last row of A)

$$= (0\ 1)(I - zA)^{-1}\begin{pmatrix}\Delta_1\\ \Delta_2\end{pmatrix}$$

$$= \frac{1}{z}\underbrace{(0\ 1)A^{-1}\begin{pmatrix}\Delta_1\\ \Delta_2\end{pmatrix}}_{O(h^3)} + O(h^4)$$

($\frac{1}{z}$: $O(h)$)

$$\Rightarrow \boxed{\tau_n = O(h^4)}$$

L-stability $\Rightarrow$ prefactor $\frac{1 - |R(z)|^n}{1 - |R(z)|} \approx 1$

$\therefore$ global error <u>improves</u> to $O(h^4)$ for stiff problems.

for Hammer-Hollingsworth:

$1/2-\sqrt{3}/6$	$1/4$	$1/4-\sqrt{3}/6$
$1/2+\sqrt{3}/6$	$1/4+\sqrt{3}/6$	$1/4$
	$1/2$	$1/2$

$\Delta_0 = O(h^5)$ (method is 4th order)

$$\Delta_1 = \underbrace{\left(c_1^2 - 2\sum_j a_{1j} c_j\right)}_{0}\frac{h^2}{2}\varphi'' + \underbrace{\left(c_1^3 - 3\sum_j a_{1j} c_j^2\right)}_{\frac{1}{12\sqrt{3}}}\frac{h^3}{6}\varphi''' + \cdots$$

$$\Delta_2 = \underbrace{\left(c_2^2 - 2\sum_j a_{2j} c_j\right)}_{0}\frac{h^2}{2}\varphi'' + \underbrace{\left(c_2^3 - 3\sum_j a_{2j} c_j^2\right)}_{-\frac{1}{12\sqrt{3}}}\frac{h^3}{6}\varphi''' + \cdots$$

so $$\tau_n = \underbrace{b^T A^{-1}\begin{pmatrix}1\\-1\end{pmatrix}\cdot\frac{h^3}{12\sqrt{3}}}\varphi''' + O(h^4)$$

$$-\frac{1}{36}h^3\varphi''' + O(h^4)$$

why isn't the method globally 3rd order then?

$$R(z) = \frac{z^2+6z+12}{z^2-6z+12} \to 1 \quad \text{as} \quad z\to\infty$$

Let $h = 1/N$, $\quad z = h\lambda = -N$

then $R(-N) = \dfrac{1 - 6/N + 12/N^2}{1 + 6/N + 12/N^2} \approx 1 - \dfrac{12}{N} + O\left(\dfrac{1}{N^2}\right)$

$$\log |R(-N)|^N \approx N \log\left(1 - \frac{12}{N}\right) \approx -12$$

so $\quad \dfrac{1 - |R(-N)|^N}{1 - |R(N)|} \approx \dfrac{1 - e^{-12}}{1 - (1 - \frac{12}{N})} \approx \dfrac{N}{12} \quad (e^{-12} \ll 1)$

thus, the global error is

$$\approx \frac{h^{-1}}{12}\left(\frac{1}{36} h^3 \max_{0 \le t \le T} \varphi'''(t)\right) = O(h^2)$$

method degraded from 4th order to 2nd order

linear stability of multistep methods

$$a_0 y_{n+s} + \cdots a_s y_n = h\left[b_0 f_{n+s} + \cdots + b_s f_n \right]$$

problem of interest: $\boxed{y' = \lambda y}$ $\longrightarrow$ λy_{n+s} λy_n

$$\circledast \qquad \underbrace{(a_0 - h\lambda b_0)}_{\tilde{a}_0} y_{n+s} + \cdots + \underbrace{(a_s - h\lambda b_s)}_{\tilde{a}_s} y_n = 0$$

when do all solutions of this linear difference equation remain bounded? precisely when the polynomial

$$\tilde{\rho}(r) = \rho(r) - h\lambda\sigma(r) = \tilde{a}_0 r^s + \tilde{a}_1 r^{s-1} + \cdots + \tilde{a}_s$$

satisfies the root condition. (all zeros of $\tilde{\rho}(r) = 0$ satisfy $|r_j| \le 1$ and, if $|r_j| = 1$, it is a simple root.)

recall how this works: if r_j is a zero of $\tilde{\rho}(r)$ of multiplicity μ_j, then for $0 \le k \le \mu_j - 1$ the sequence

$$y_n = \begin{cases} 0 & n < k \\ \binom{n}{k} r_j^{n-k} & n \ge k \end{cases}$$

solves $\circledast$, and these solutions form a basis for the solution space.

$\binom{n}{k}$ grows polynomially for fixed k (like n^k)

r_j^{n-k} decays exponentially (like $|r_j|^n$ if $|r_j|<1$)

or remains of constant magnitude (if $|r_j|=1$)

One difference between one-step methods and multistep methods is that the latter can have solutions that grow for a while and then decay while the former either grow or decay by the same amount at every step ($y_n = R(h\lambda)^n y_0$)

<u>def:</u> the RAS of a multistep method is the set

$$RAS = \left\{ z \in \mathbb{C} : \text{all roots of } \rho(r) - z\sigma(r) \text{ satisfy } |r_j| \le 1 \right.$$
$$\left. \text{and if } |r_j| = 1 \text{ then } r_j \text{ is a simple root.} \right\}$$

(note that a method is stable iff $0 \in RAS$)

Example: 2-step BDF

$$\frac{3}{2} y_{n+2} - 2 y_{n+1} + \frac{1}{2} y_n = h f_{n+2}$$

$$\rho(r) - z\sigma(r) = \left(\frac{3}{2} - z\right) r^2 - 2r + \frac{1}{2}$$

$$r_j = \frac{2 \pm \sqrt{4 - 4\left(\frac{3}{2} - z\right)\cdot\frac{1}{2}}}{2\left(\frac{3}{2} - z\right)} = \frac{2 \pm \sqrt{1 + 2z}}{3 - 2z}$$

how do we figure out which z's cause both r_j's to satisfy $|r_j| \le 1$?

trick: boundary locus technique. Instead of asking which z leads to $|r_j| \le 1$, imagine $|r| = 1$ is given and solve for z.

$$r = e^{i\theta}$$

$$z = \frac{\rho(r)}{\sigma(r)} = \frac{\frac{3}{2}r^2 - 2r + \frac{1}{2}}{r^2} = \frac{3}{2} - \frac{2}{r} + \frac{1}{2r^2}$$

$$= \frac{3}{2} - 2e^{-i\theta} + \frac{1}{2}e^{-2i\theta}$$

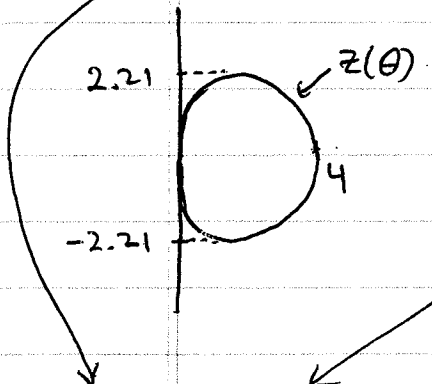
$$= \frac{3}{2} - 2(\cos\theta - i\sin\theta) + \underbrace{\frac{1}{2}(\cos\theta - i\sin\theta)^2}$$

$$\frac{1}{2}\cos^2\theta - i\cos\theta\sin\theta - \frac{1}{2}\sin^2\theta$$

$$= (1-\cos\theta)^2 + i\sin\theta(2-\cos\theta) \qquad -\frac{1}{2} + \frac{1}{2}\cos^2\theta$$

only on this curve can our polynomial have roots of modulus 1.

note that in this example, the real part is always positive.



question: which side is which?

try $z = 1$: $r_j = \frac{2 \pm \sqrt{3}}{3-2} = 2 \pm \sqrt{3}$

result: one is bigger than one, the other smaller.
(same must be true for any z inside curve)

now try $z = -\frac{1}{2}$: $r_j = \frac{2 \pm \sqrt{0}}{3 + (2)(\frac{1}{2})} = \frac{1}{2}$

so both are smaller than one.

conclusion:

every $z \in \mathbb{C}^-$ belongs to the RAS.

∴ method is A-stable.

the other multistep methods can be treated similarly.

1- plot the curve $z(\theta) = \dfrac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}$ for $0 \le \theta \le 2\pi$

2- find the roots r_j of $\rho(r) - z\sigma(r) = 0$ for any convenient choice of z in each region bounded by the curve. The region is part of the RAS iff each root r_j satisfies $|r_j| < 1$.

results:

(1) the stability regions shrink as the order increases (unlike RK methods where they grow)

(2) The RAS of the A.B. method is generally tiny.
" " " " A.M. method is bigger, but nowhere near A-stable.

(3) the RAS of the Nystrom (e.g. leapfrog $y_{n+1} = y_{n-1} + 2h f_n$) and Milne (e.g. $y_{n+1} - y_{n-1} = h\left[\frac{1}{3} f_{n+1} + \frac{4}{3} f_n + \frac{1}{3} f_{n-1}\right]$) methods consists of the point $z = 0$.
(only $\lambda = 0$, i.e. $y' = 0$, has all solutions bounded. Thus, even though Milne is a 2-step 4th order method, it's not very stable — oscillatory errors tend to grow exponentially in time like our worst case error analysis in Lec. 10)

(4) The BDF methods are not A-stable for $s \ge 3$. They're not even stable for $s \ge 7$. (unbounded solutions for $\lambda = 0$, $y' = 0$ exist.) They are, however $A(\alpha)$ stable for $s \le 6$.

stability domains of various multistep methods. (Reference: Hairer/Norsett/Wanner volume 2)

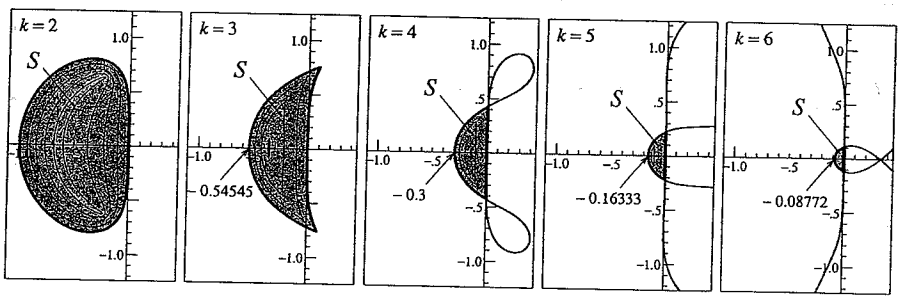


Fig. 1.2. Stability domains for explicit Adams methods

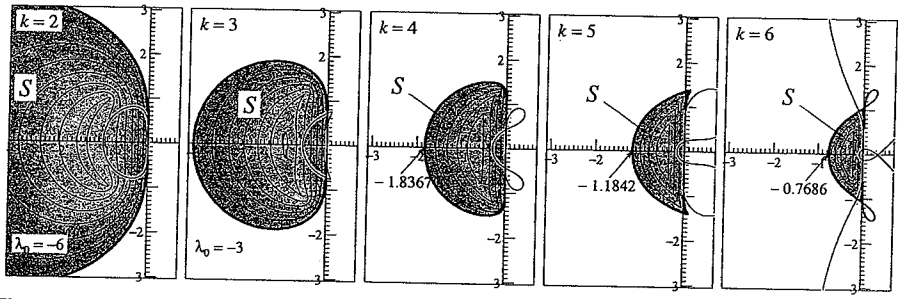


Fig. 1.3. Stability domains of implicit Adams methods, compared to those of the explicit ones

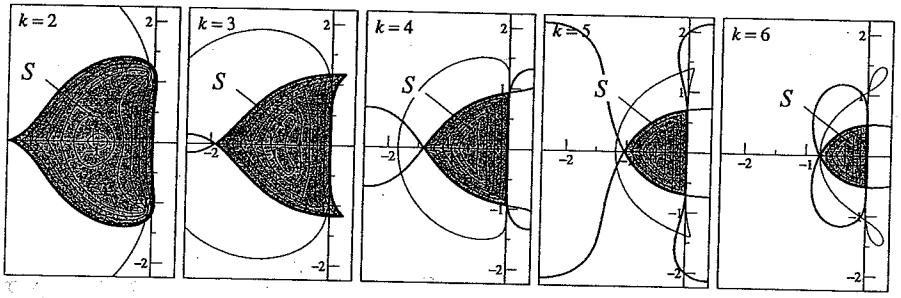


Fig. 1.4. Stability domains for PECE compared to original implicit methods

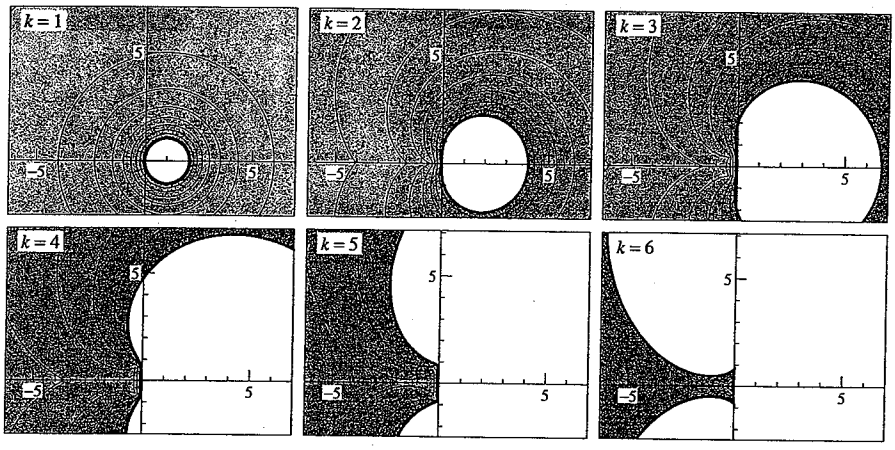
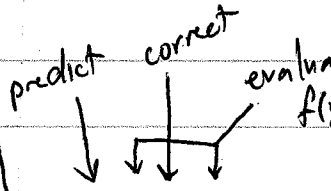


Fig. 1.6. Root locus curves and stability domains of BDF methods

PECE methods are a compromise between explicit and implicit multistep methods. (they are explicit methods, not really multistep anymore)



PECE

do one fixed point iteration to "solve" the implicit Adams-Moulton equation.

predictor: (AB):

$$y^{(0)} = y_{n+s-1} + h\left[b_1 f_{n+s-1} + \cdots + b_s f_n\right]$$

fixed point iteration:

$$y^{(\nu+1)} = y_{n+s-1} + h\left[b_0 f(t_{n+1}, y^{(\nu)}) + b_1 f_{n+s-1} + \cdots + b_s f_n\right]$$

PECE:

$$y_{n+s} = y^{(1)}$$

P(EC)^{2}E:

$$y_{n+s} = y^{(2)}$$

PECE is enough to achieve order of A.M. method.

P(EC)^{2}E is enough to achieve principal term in the truncation error of the A.M. method.

implicit-explicit (imex) methods

discretizing PDE's often leads to stiff equations where the source of stiffness is due to a high order differential operator in space, which is linear.

example: viscous Burger's equation: $u_t + uu_x = \nu u_{xx}$

(uu_x: nonlinearity; νu_{xx}: source of stiffness)

method of lines

a common way to solve PDE's like this is to discretize in space first, and then choose your favorite ODE solver to evolve the resulting ODE.

notation: U_j^n is numerical solution representing $u(j\Delta x, n\Delta t)$

step 1: discretize in space. (e.g. finite differences or spectral method)

$$u_x \approx \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

$$u_{xx} \approx \frac{\left(\frac{u_{j+1} - u_j}{\Delta x}\right) - \left(\frac{u_j - u_{j-1}}{\Delta x}\right)}{\Delta x} = \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2}$$

ODE to solve

$$\frac{d}{dt} u_j(t) = -u_j(t) \frac{u_{j+1}(t) - u_{j-1}(t)}{2\Delta x} + \nu \frac{u_{j+1}(t) - 2u_j(t) + u_{j-1}(t)}{\Delta x^2}$$

$$u' = f(u) + g(u),$$

f is non-linear

g requires small timesteps if explicit method used.

problems f makes it hard to implement an implicit method
g " " expensive " " " " explicit "

solution: treat the two parts differently

multistep approach:

$$a_0 u^{n+s} + \cdots + a_s u^n = h\Big[\overbrace{b_1 f^{n+s-1} + \cdots + b_s f^n}^{\text{treat } f \text{ explicitly}} + \overbrace{c_0 g^{n+s} + \cdots + c_s g^n}^{\text{treat } g \text{ implicitly}}\Big]$$

$h = \Delta t$ (usually $h = \Delta x$, $k = \Delta t$ but we'll stick with $h = \Delta t$)

match terms in Taylor series:

$\tilde{f}(t) = f(t, u(t))$

$$\tau_n = a_0 u(t_{n+s}) + \cdots + a_s u(t_n) - h\Big[\begin{array}{l} b_1 \tilde{f}(t_{n+s-1}) + \cdots + b_s \tilde{f}(t_n) \\ + c_0 \tilde{g}(t_{n+s-1}) + \cdots + c_s \tilde{g}(t_n) \end{array}\Big]$$

$u(t_n)$: exact solution of ODE (after we discretized space)

$$u(t_{n+j}) = u(t_n) + (jh)u'(t_n) + \cdots + \frac{1}{p!}(jh)^p u^{(p)}(t_n) + \cdots$$

$$\tilde{f}(t_{n+j}) = \tilde{f}(t_n) + (jh)\tilde{f}'(t_n) + \cdots$$

$$u^{(p)}(t_n) = \frac{d^{p-1}}{dt^{p-1}}\left(\tilde{f}(t_n) + \tilde{g}(t_n)\right) = \tilde{f}^{(p-1)}(t_n) + \tilde{g}^{(p-1)}(t_n)$$

$$\tau_n = \left(\sum_j a_j\right) u(t_n) + \left(\sum_j (s-j) a_j u'(t_n) - \sum_j b_j \tilde{f}(t_n) - \sum_j c_j \tilde{g}(t_n)\right) h$$

$$+ \cdots + \left(\sum_j \frac{(s-j)^p}{p!} a_j u^{(p)}(t_n) - \sum_j \frac{(s-j)^{p-1}}{(p-1)!} b_j \tilde{f}^{(p-1)}(t_n) - \sum_j \frac{(s-j)^{p-1}}{(p-1)!} c_j \tilde{g}^{(p-1)}(t_n)\right) h^p$$

$$+ O(h^{p+1})$$

so $\tau_n = O(h^{p+1})$ and method is order p as long as

$$\sum_{j=0}^{s} a_j = 0 \qquad \sum_{j=0}^{s} (s-j)a_j = \sum_{j=1}^{s} b_j = \sum_{j=0}^{s} c_j$$

$$\sum_{j=0}^{s} \frac{(s-j)^m}{m!} a_j = \sum_{j=1}^{s} \frac{(s-j)^{m-1}}{(m-1)!} b_j = \sum_{j=0}^{s} \frac{(s-j)^{m-1}}{(m-1)!} c_j \quad (m=2\ldots p)$$

summary: all you need is for (a_j, b_j) and (a_j, c_j) to each be order p when solving $u' = f(u)$ or $u' = g(u)$. the combined scheme for $u' = f(u) + g(u)$ will then also be order p.

popular choices:

2nd order

$$u^{n+1} - u^n = h\left[\underbrace{\frac{3}{2}f^n - \frac{1}{2}f^{n-1}}_{\text{2nd order Adams Bashforth}} + \underbrace{\frac{1}{2}(g^{n+1} + g^n)}_{\text{trapezoidal rule}}\right]$$

this scheme is called CNAB (CN: Crank-Nicholson, i.e. trapezoidal rule for PDE's)

2nd order

$$u^{n+1} - u^{n-1} = h\left[f^n + \frac{1}{2}(g^{n+1} + g^{n-1})\right]$$

CNLF: Crank-Nicholson, Leapfrog

example:
SBDF4: $f(u) = au_x$
$g(u) = u_{xx}$
treat space derivatives spectrally.
stability requirement:
$a\Delta t \le \frac{1}{2}\Delta x$

2nd order

$$\frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1} = h\left[2f^n - f^{n-1} + g^{n+1}\right]$$

SBDF: split Backward Differentiation Formula

4th order SBDF

$$\frac{25}{12}u^{n+1} - 4u^n + 3u^{n-1} - \frac{4}{3}u^{n-2} + \frac{1}{4}u^{n-3} = h\left[4f^n - 6f^{n-1} + 4f^{n-2} - f^{n-3} + g^{n+1}\right]$$

3rd order SBDF

$$\frac{11}{6}u^{n+1} - 3u^n + \frac{3}{2}u^{n-1} - \frac{1}{3}u^{n-2} = h\left[3f^n - 3f^{n-1} + f^{n-2} + g^{n+1}\right]$$

implicit explicit methods

$y' = f(t,y) + g(t,y)$ — f nonlinear, g source of stiffness

last time: multistep approach:

$$a_0 y_{n+s} + \cdots + a_s y_n = h\Big[\overbrace{b_1 f_{n+s-1} + \cdots + b_s f_n}^{\text{treat } f \text{ explicitly}} + \overbrace{\underbrace{c_0 g_{n+s}}_{c_0 g(t_{n+s}, y_{n+s})} + \cdots + c_s g_n}^{\text{treat } g \text{ implicitly}}\Big]$$

we saw that the combined method is as accurate as the least accurate of the two methods individually. (no extra coupling conditions)

IMEX Runge Kutta

two Butcher arrays: (A, b for f; $\hat{A}$, $\hat{b}$ for g)

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \qquad \begin{array}{c|c} \hat{c} & \hat{A} \\ \hline & \hat{b}^T \end{array}$$

scheme:

$$k_i = f\Big(t_n + c_i h,\ y_n + h\sum_j a_{ij} k_j + h\sum_j \hat{a}_{ij} l_j\Big)$$

$$l_i = g\Big(t_n + \hat{c}_i h,\ y_n + h\sum_j a_{ij} k_j + h\sum_j \hat{a}_{ij} l_j\Big)$$

$$y_{n+1} = y_n + h\sum_j b_j k_j + h\sum_j \hat{b}_j l_j$$

assumptions: $a_{ij} = 0$ if $j \geq i$ (treat f explicitly)

$\hat{a}_{ij} = 0$ if $j > i$ (only consider DIRK schemes for g)

these assumptions allow you to solve for k_i, l_i one stage at a time.

skip this step if $\hat{a}_{i1} = 0$ for all i → solve for l_1 in g equation (implicit g)
plug l_1 into f, get k_1 (explicit f)
plug l_1, k_1 into g, solve for l_2 (implicit g)
plug into f to obtain k_2 (explicit f)
etc.

in principle, you could allow nonzero entries in $\hat{A}$ above the diagonal as long as appropriate entries in A are zero, (but I've never seen anyone do that). For example:

A: $\begin{array}{|cccc} 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & * & * & 0 \end{array}$ $\hat{A}$: $\begin{array}{|cccc} * & 0 & 0 & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & * \end{array}$

this would be OK because you could solve the system

$$l_2 = g(t_n + \hat{c}_2 h, \underbrace{y_n + h a_{21} k_1 + h \hat{a}_{21} l_1}_{\text{known}} + \underbrace{h \hat{a}_{22} l_2 + h \hat{a}_{23} l_3}_{\text{unknown}})$$
$$l_3 = g(t_n + \hat{c}_3 h, \underbrace{y_n + h a_{21} k_1 + h \hat{a}_{31} l_1}_{\text{known}} + \underbrace{h \hat{a}_{32} l_2 + h \hat{a}_{33} l_3}_{\text{unknown}})$$

$=$ without getting f involved in this implicit stage (since $a_{32} = 0$)

reduction to autonomous case: (doesn't matter how you split the 1 between f and g)

$$\tilde{y} = \begin{pmatrix} t \\ y \end{pmatrix}, \quad \tilde{y}' = \begin{pmatrix} 1 \\ f(\tilde{y}^0, y) \end{pmatrix} + \begin{pmatrix} 0 \\ g(\tilde{y}^0, y) \end{pmatrix}$$

requires $c_i = \hat{c}_i = \sum_j a_{ij}$

for simplicity, we usually also assume that $\hat{c}_i = \sum_j \hat{a}_{ij}$ so that

order conditions

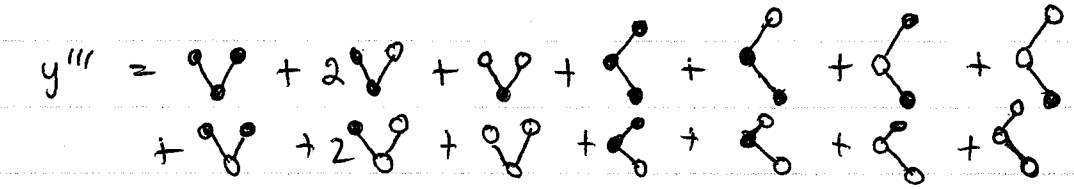
differentiating the exact solution, we find that

$$y' = f(y) + g(y)$$

$$y'' = Df(y)(f(y)+g(y)) + Dg(y)(f(y)+g(y))$$

$$= Df(f) + Df(g) + Dg(f) + Dg(g)$$

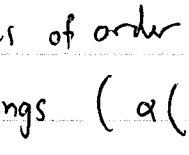
$$y''' =$$

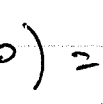


$$y^{(p)}(t) = \sum_{\phi \in PT_p} \alpha(\phi) F(\phi)(y(t))$$

each node is "fat" or "meager"

where PT_p are the partitioned trees of order p

$\alpha(\phi)$ is the number of labelings ($\alpha($  $) = 2$:  or )

$F($  $) = D^2 g(Df(g), g)$

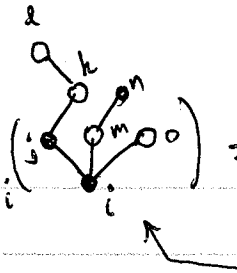
f: meager nodes
g: fat nodes

if we also differentiate the stage derivatives and match terms in the truncation error expansion, we learn that:

An imex scheme is of order $\geq p$ iff

$$\sum_{j=1}^{s} b_j \Phi_j(\phi) = \frac{1}{\gamma(\phi)} \quad \text{and} \quad \sum_{j=1}^{s} \hat{b}_j \hat{\Phi}_j(\phi) = \frac{1}{\gamma(\phi)}$$

for all P-trees of order $\leq p$.

here e.g. $\Phi_i\left(\right.$  $\left.\right) = \sum_{j,k,l,m,n,o} a_{ij}\hat{a}_{jk}\hat{a}_{kl}\hat{a}_{im}a_{mn}\hat{a}_{io}$

and γ works as before, e.g. $\gamma(\quad) = 7 \cdot 3 \cdot 2 \cdot 2 = 84$

=

the number of partitioned trees grows very rapidly with p

p	1	2	3	4	5	6
$\#T_p$	1	1	2	4	9	20
$\#PT_p$	2	4	14	52	214	916

but many of the new coupling conditions become redundant with simplifying assumptions such as $\underbrace{\hat{c}_i = c_i}$ and $\underbrace{\hat{b}_i = b_i}$

$\hat{c}_i = c_i$: color of terminal nodes don't matter

$\hat{b}_i = b_i$: root color doesn't matter

examples:

forward-backward Euler:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad , \quad \begin{array}{c|c} \hat{c} & \hat{A} \\ \hline & \hat{b}^T \end{array} \quad = \quad \begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & 1 & 0 \end{array} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 0 & 1 \\ \hline & 0 & 1 \end{array}$$

l_1 and k_2 don't have to be computed, actually...

note that requiring $\hat{c}_i = c_i = \sum_j a_{ij}$ makes $\hat{c}_i = 0$, so you can't construct a one-stage IMEX method with $\hat{A}, \hat{b}$ implicit and $\hat{c}_i = \sum_j \hat{a}_{ij}$ ← This assumption isn't strictly necessary, so

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

is also valid

and

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

would work too if we do the autonomous reduction as $\tilde{y}' = \begin{pmatrix} 0 \\ f \end{pmatrix} + \begin{pmatrix} 1 \\ g \end{pmatrix}$

imex midpoint rule: (2nd order)

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

3-stage 3rd order:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \beta & \beta & 0 & 0 \\ 1-\beta & \beta-1 & 2(1-\beta) & 0 \\ \hline & 0 & 1/2 & 1/2 \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \beta & 0 & \beta & 0 \\ 1-\beta & 0 & 1-2\beta & \beta \\ \hline & 0 & 1/2 & 1/2 \end{array} \qquad \beta = \frac{3+\sqrt{3}}{6}$$

more sophisticated examples with better stability properties for the implicit scheme, embedded formulas (for stepsize control) and higher orders (through order 5) can be found in the paper:

> Additive Runge-Kutta schemes for convection-diffusion-reaction equations
> by Kennedy / Carpenter Applied Numerical Mathematics 44 (2003) 139-181

Spectral methods

many PDE's are of the form

$$u_t = Lu$$

where L is a (possibly non-linear) differential operator involving derivatives with respect to space variables.

ARK4(3)6L[2]SA–ERK

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
$\frac{83}{250}$	$\frac{13861}{62500}$	$\frac{6889}{62500}$	0	0	0	0
$\frac{31}{50}$	$\frac{-116923316275}{2393684061468}$	$\frac{-2731218467317}{15368042101831}$	$\frac{9408046702089}{11113171139209}$	0	0	0
$\frac{17}{20}$	$\frac{-451086348788}{2902428689909}$	$\frac{-2682348792572}{7519795681897}$	$\frac{12662868775082}{11960479115383}$	$\frac{3355817975965}{11060851509271}$	0	0
1	$\frac{647845179188}{3216320057751}$	$\frac{73281519250}{8382639484533}$	$\frac{552539513391}{3454668386233}$	$\frac{3354512671639}{8306763924573}$	$\frac{4040}{17871}$	0
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
$\hat{b}_i$	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$\frac{-3700637}{11593932}$	$\frac{61727}{225920}$

ARK4(3)6L[2]SA–ESDIRK

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$\frac{-1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$\frac{-654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$\frac{-71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
$\hat{b}_i$	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$\frac{-3700637}{11593932}$	$\frac{61727}{225920}$

examples: $u_t = u_{xx}$ heat equation in 1d

$u_t = \Delta u$ " " " higher dimensions

$\left(\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} = \text{Laplacian}\right)$

$u_{tt} = c^2 u_{xx}$ wave equation

has desired form if we make it into a system: $v = \begin{pmatrix} u_x \\ u_t \end{pmatrix}$

$$v_t = \begin{pmatrix} u_{xt} \\ u_{tt} \end{pmatrix} = \begin{pmatrix} u_{xt} \\ c^2 u_{xx} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ c^2 & 0 \end{pmatrix}}_{A} \begin{pmatrix} u_x \\ u_t \end{pmatrix}_x = A v_x$$

convection diffusion: $u_t = -cu_x + \nu u_{xx}$

Burger's: $u_t = -uu_x + \nu u_{xx}$

KdV: $u_t = -uu_x - \nu u_{xxx}$

etc.—

in a spectral method, we discretize space uniformly and compute space derivatives using the fast fourier transform. This yields an ODE in time that we solve using one of the methods we've studied this semester.

for simplicity, let's assume we're dealing with periodic b/c's on the interval

$$0 \le x \le 2\pi$$

other interval lengths can be reduced to this one by a change of variables, and sometimes other boundary conditions can be reduced to the periodic case on a larger interval by even or odd reflection.

We will represent the solution as a Fourier series

$$u(x,t) = \sum_{k=-\infty}^{\infty} \hat{u}_k(t)\, e^{ikx}, \qquad \hat{u}_k(t) = \frac{1}{2\pi}\int_0^{2\pi} u(x,t) e^{-ikx}\, dx$$

properties:

① f real $\Rightarrow \hat{f}_k^* = \hat{f}_{-k}$ ← $f(x) = \sum_k \hat{f}_k e^{ikx}$; $\frac{1}{2\pi}\int_0^{2\pi} e^{ikx} e^{-ilx}\, dx = \delta_{kl}$

② $\frac{1}{2\pi}\int_0^{2\pi} |f(x)|^2\, dx = \sum_k |\hat{f}_k|^2$ Parseval (follows from orthogonality of e^{ikx})

③ decay of fourier modes is determined by the smoothness of f

let $L = \max_{0\le x\le 2\pi} |f^{(m)}(x)|$. Then $|\hat{f}_k| \le \frac{L}{|k|^m}$

proof: integrate by parts.

$\Rightarrow$ if $f \in C_p^\infty[0,2\pi]$, the fourier amplitudes decay faster than any algebraic power of $\frac{1}{k}$. (they decay exponentially if f is analytic in a strip around the real axis)

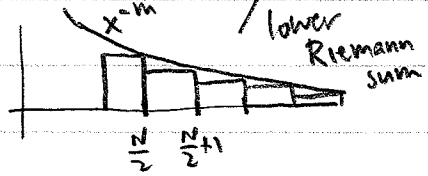
④ truncation error: $P_N f(x) = \sum_{k=-\frac{N}{2}+1}^{N/2} \hat{f}_k e^{ikx}$

$$\|f - P_N f\|_2^2 = \frac{1}{2\pi}\int_0^{2\pi} |f(x) - P_N f(x)|^2\, dx = \sum_{\substack{k\le -N/2 \\ \text{or } k > N/2}} |\hat{f}_k|^2$$

$$\le 2\sum_{k=N/2}^{\infty} \frac{L}{k^m} \le 2L\int_{\frac{N}{2}-1}^{\infty} x^{-m}\, dx = \frac{2L}{m-1}\left(\frac{N}{2}-1\right)^{-m+1}$$

$$\le \frac{C_m}{N^{m-1}}$$

assuming $f \in C_p^m[0,2\pi]$



today we'll continue our discussion of spectral methods for solving evolution equations

$$u_t = Lu \quad \text{or} \quad u_t = Lu + f$$

with periodic boundary conditions $u(0) = u(2\pi)$

$=$

Fourier series

$$L^2(0,2\pi) = \{ \text{square integrable functions } f : (0,2\pi) \to \mathbb{C} \}$$

norm: $\|f\|_{L^2} = \sqrt{(f,f)_{L^2}} = \sqrt{\frac{1}{2\pi}\int_0^{2\pi} |f(x)|^2 \, dx}$

inner product: $(f,g)_{L^2} = \frac{1}{2\pi}\int_0^{2\pi} f(x)\,\overline{g(x)}\,dx$

$$\ell^2(\mathbb{Z}) = \{ \text{square summable sequences } \{c_j\}_{j=-\infty}^{\infty}, \ c_j \in \mathbb{C} \}$$

norm: $\|c\|_{\ell^2} = \sqrt{(c,c)_{\ell^2}} = \sqrt{\sum_{j=-\infty}^{\infty} |c_j|^2}$

inner product $(c,d)_{\ell^2} = \sum_j c_j d_j^*$ $\qquad d^* = \bar{d} =$ complex conjugate

L^2 and ℓ^2 are both examples of Hilbert spaces $\Leftarrow$ complete inner product spaces

Cauchy-Schwarz inequality:

$$|(f,g)| \le \|f\| \cdot \|g\|$$

the trigonometric functions $\varphi_k(x) = e^{ikx}$, $k \in \mathbb{Z}$

form an orthonormal basis for $L^2(0,2\pi)$ ← (i.e. $(\varphi_j, \varphi_k) = \delta_{jk}$ and if $(f, \varphi_j) = 0$ for all j then $f \equiv 0$)

As a result, the Fourier transform

$$L^2(0,2\pi) \xrightarrow{\mathcal{F}} \ell^2(\mathbb{Z})$$

$$f(x) \longmapsto \hat{f}_k = (\mathcal{F}f)_k = (f, \varphi_k) = \frac{1}{2\pi}\int_0^{2\pi} f(x) e^{-ikx}\, dx$$

$$c^\vee(x) = \mathcal{F}^{-1} c(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} \longleftarrow \{c_k\}_{k=-\infty}^{\infty}$$

is an isometric (norm preserving) isomorphism (continuous linear bijection) from $L^2(0,2\pi)$ to $\ell^2(\mathbb{Z})$. This happens any time you start with an orthonormal basis for a Hilbert space H and define the mapping $\mathcal{F}: H \to \ell^2$ via $(\mathcal{F}f)_k = (f, \varphi_k)$

$$\|\mathcal{F}f\|_{\ell^2} = \|f\|_H \quad \forall f \in H \leftarrow \text{what it means for } \mathcal{F} \text{ to be an isometry}$$

proof: $\|f\|_H^2 = (f,f)_H = \left(\sum_j (f,\varphi_j)\varphi_j, \sum_k (f,\varphi_k)\varphi_k \right)$

$$= \sum_{jk} (f,\varphi_j)\overline{(f,\varphi_k)}\underbrace{(\varphi_j,\varphi_k)}_{\delta_{jk}} = \sum_j |(f,\varphi_j)|^2 = \|\mathcal{F}f\|_{\ell^2}$$

adjoint = conjugate transpose

example: if $H = \mathbb{C}^n$ with inner product $(x,y) = x^T \bar{y}$

and if A is a Hermitian $n \times n$ matrix ($A^* = A$, $(A^*)_{ij} = \bar{A}_{ji}$)

then A can be diagonalized by a unitary matrix U:

$$A = U \Lambda U^{-1}, \quad U^* U = I \leftarrow \text{columns of } U \text{ are an orthonormal basis for } H$$

↑ columns of U are the eigenvectors of A

we can think of U as an isometric isomorphism from $\ell^2(\{1,\dots,n\})$ to $H = \mathbb{C}^n$

$$U = (\varphi_1, \dots, \varphi_n) \quad , \quad U\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = c_1\varphi_1 + \dots + c_n\varphi_n$$

↑ ↑ the columns of U form a basis for $H = \mathbb{C}^n$

(although normally when $H = \mathbb{C}^n$ you think of $\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$ belonging to $\mathbb{C}^n$ as well instead of introducing the notation $\ell^2(\{1,\dots,n\})$ for the domain of U)

the significance of the Fourier transform is that it diagonalizes the derivative operator

$$D : L^2(0,2\pi) \to L^2(0,2\pi) \qquad \leftarrow \text{an unbounded operator... not defined for all } f \in L^2$$
$$f \mapsto f'$$

$$D = \mathcal{F}^{-1} \Lambda \mathcal{F}$$

$$\Lambda : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z}) \qquad \leftarrow \text{an infinite matrix}$$

$$\Lambda_{jk} = ik\,\delta_{jk}$$

$$\Lambda = \begin{pmatrix} \ddots & & & & & \\ & -3i & & & & \\ & & -2i & & & \\ & & & -i & & \\ & & & & 0 & \\ & & & & & i \\ & & & & & & 2i \\ & & & & & & & 3i \\ & & & & & & & & \ddots \end{pmatrix}$$

$j=-1$, $j=0$, $j=1$

$k=-1$, $k=0$, $k=1$ etc.

proof: $f(x) = \sum_k \hat{f}_k e^{ikx}$

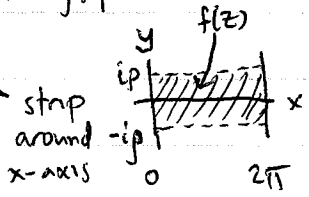
$$f'(x) = \sum_k \underbrace{ik\hat{f}_k}_{\text{Fourier coefficients of } f'(x)} e^{ikx}$$

$$(\widehat{f'})_k = ik\hat{f}_k = (\Lambda \hat{f})_k$$

another nice feature of the Fourier transform is that the Fourier coefficients of a smooth function decay rapidly as $|k| \to \infty$

$$f \in C_p^m[0,2\pi] \;:\quad |\hat{f}_k| \le \frac{L}{|k|^m}, \quad k \neq 0, \quad L = \max_{0 \le x \le 2\pi} |f^{(m)}(x)|$$

$$f \text{ analytic \& periodic near x-axis} \;:\quad |\hat{f}_k| \le M e^{-\rho|k|}, \quad M = \max_{\substack{0 \le x \le 2\pi \\ -\rho \le y \le \rho}} |f(x+iy)|$$



thus, storing Fourier coefficients gives a very compact representation of smooth functions to extremely high accuracy.

truncation: $P_N f(x) = \sum_{k=-\frac{N}{2}+1}^{N/2} \hat{f}_k e^{ikx}$

$$\|f - P_N f\|^2 = \sum_{\substack{k \le -N/2 \\ \text{or } k > N/2}} |\hat{f}_k|^2 \le 2 \sum_{k=N/2}^{\infty} \frac{L^2}{k^{2m}} \le \frac{2L^2}{2m-1}\left(\frac{N}{2}-1\right)^{1-2m}$$

$2L^2 \int_{\frac{N}{2}-1}^{\infty} x^{-2m}\,dx$

so if $N \ge 8$ and $m \ge 1$ then $\|f - P_N f\| \le \frac{L}{\sqrt{m-1/2}}\left(\frac{8}{3N}\right)^{m-\frac{1}{2}}$

analytic case: $\|f - P_N f\|^2 \le 2\sum_{k=N/2}^{\infty} M^2 e^{-2\rho|k|} \le 2M^2 \int_{\frac{N}{2}-1}^{\infty} e^{-2\rho x}\,dx$

$$\Rightarrow \|f - P_N f\| \le M \frac{e^{\rho}}{\sqrt{\rho}} e^{-\rho N/2}$$

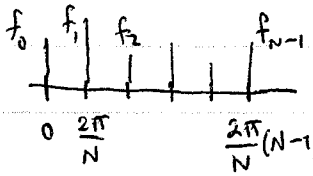
$\frac{e^{-2\rho\left(\frac{N}{2}-1\right)}}{2\rho}$

so the error we commit in truncating the Fourier expansion of a smooth or analytic function decays very rapidly to zero as $N \to \infty$.

(truncate expansion → space becomes finite dimensional → can represent f in physical space via its values at N equally spaced points)

discrete orthogonality. consider $\mathbb{C}^N$ with inner product

$$(f,g)_N = \frac{1}{N}\sum_{j=0}^{N-1} f_j \bar{g}_j$$



$f_j = f(x_j)$

$x_j = \frac{2\pi j}{N}$

the basis functions $\varphi_k(x_j) = e^{ikx_j} = e^{2\pi i kj/N}$ satisfy

$$(\varphi_k, \varphi_j)_N = \begin{cases} 1 & k = j + N\ell \text{ for some } \ell \in \mathbb{Z} \\ 0 & \text{o.w.} \end{cases}$$

so for any N consecutive integers k (e.g. $-\frac{N}{2}+1 \le k \le \frac{N}{2}$ or $0 \le k \le N-1$)
the functions φ_k form a basis for $\mathbb{C}^N$.

Note that $\varphi_{k+\ell N}$ and φ_k have identical values on the grid:

$$\varphi_{k+\ell N}(x_j) = e^{2\pi i (k+\ell N)j/N} = e^{2\pi i kj/N} \underbrace{e^{2\pi i \ell j}}_{1} = \varphi_k(x_j)$$

this phenomenon is known as aliasing. We generally want to choose the lowest frequency version (with $-\frac{N}{2}+1 \le k \le \frac{N}{2}$)

Discrete Fourier Transform (the FFT is a fast algorithm for computing this)

since the basis functions $\varphi_k(x_j) = e^{ikx_j}$, $k \in K_N = \{-\frac{N}{2}+1, \dots, \frac{N}{2}\}$ are orthonormal, we can expand any $f \in \mathbb{C}^N$, $f_j = f(x_j)$, $0 \le j \le N-1$ into its Fourier components:

$$f_j = \sum_{k \in K_N} \tilde{f}_k e^{ikx_j}, \quad \tilde{f}_k = (f, \varphi_k)_N = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ikx_j}$$

$$\|\tilde{f}\|_{\ell^2(K_N)} = \|f\|_N \quad \text{i.e.} \quad \sum_{k \in K_N} |\tilde{f}_k|^2 = \frac{1}{N}\sum_{j=0}^{N-1} |f_j|^2$$

Note that if f_j is obtained by sampling a function $f \in L^2(0,2\pi)$, then $\tilde{f}_k$ is the trapezoidal rule approximation of $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x) e^{-ikx}\,dx$

when we sample a function and compute the DFT, we commit aliasing error

$$\tilde{f}_k = \hat{f}_k + \sum_{\substack{l=-\infty \\ l\neq 0}}^{\infty} \hat{f}_{k+lN} \qquad k \in K_N = \left\{-\tfrac{N}{2}+1, \dots, \tfrac{N}{2}\right\}$$

$\hat{f}_{k+lN}$ ← these are very small if f is smooth.

proof: $\tilde{f}_k = (f, \varphi_k)_N = \left(\sum_l \hat{f}_l \varphi_l, \varphi_k\right)_N = \sum_{l \in k+N\mathbb{Z}} \hat{f}_l$

$(\varphi_l, \varphi_k)_N = \begin{cases} 1 & k-l \in N\mathbb{Z} \\ 0 & \text{o.w.} \end{cases}$

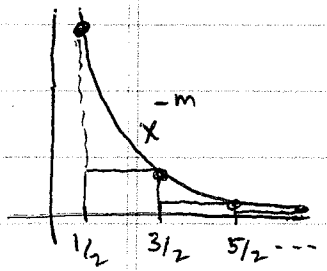
Thus, the error $|\tilde{f}_k - \hat{f}_k|$ is governed by the smoothness of f.

→ if $f \in C_p^m[0,2\pi]$ and $\max_{0\le x\le 2\pi} |f(x)| = L$ and $m \ge 2$ then

$$|\tilde{f}_k - \hat{f}_k| \le \sum_{l\neq 0} |\hat{f}_{k+lN}| \le \sum_{l\neq 0} \frac{L}{|k+lN|^m}$$

$$\le \frac{L}{N^m} \sum_{l\neq 0} \frac{1}{|l+\frac{k}{N}|^m} \le \frac{2L}{N^m} \sum_{l=1}^{\infty} \frac{1}{|l-\frac{1}{2}|^m}$$

$|l+\frac{k}{N}| \ge \big||l|-|\frac{k}{N}|\big| \ge \left(|l|-\frac{1}{2}\right)$



$$\left(\tfrac{1}{2}\right)^{-m} + \left(\tfrac{3}{2}\right)^{-m} + \dots \le 2^m + 1 + \int_{3/2}^{\infty} x^{-m}\,dx = 2^m + 1 + \frac{(3/2)^{1-m}}{m-1} \le 2^m + 2 \le 2^{m+1}$$

$$\Rightarrow \boxed{|\tilde{f}_k - \hat{f}_k| \le 4L\left(\tfrac{2}{N}\right)^m \to 0 \text{ rapidly as } N\to\infty}$$

and if f is analytic on a strip $\{z : |\mathrm{Im}\, z| \le \rho\}$ around the real axis,

$$|\tilde{f}_k - \hat{f}_k| \le \sum_{\ell \neq 0} M e^{-\rho|k+\ell N|} \le 2M e^{\frac{N}{2}\rho} \sum_{\ell=1}^{\infty} (e^{-\rho N})^{\ell} = \frac{2M e^{-\rho\frac{N}{2}}}{1 - e^{-\rho N}} \le 4M e^{-\rho\frac{N}{2}}$$

$|k+\ell N| \ge (|\ell| - \frac{1}{2})N$; $N \ge \rho^{-1}$

=

special case: $k = 0$

$$|\tilde{f}_0 - \hat{f}_0| = \left| \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) - \frac{1}{2\pi} \int_0^{2\pi} f(x)\, dx \right| \le 4M e^{-\rho\frac{N}{2}} \xrightarrow[N\to\infty]{} 0$$

$\therefore$ the trapezoidal rule approximation of an analytic function over its period converges exponentially fast (like Newton's method: double N → double # of correct digits)

=

interpolation: sample f, fill in data using trigonometric polynomials:

$$I_N f(x) = \sum_{k \in K_N} \tilde{f}_k e^{ikx} \qquad K_N = \left\{-\tfrac{N}{2}+1, \dots, \tfrac{N}{2}\right\}$$

differentiation: $I_N f'(x_j) = \sum_{k \in K_N} (ik\tilde{f}_k) e^{ikx_j}$ ← fast way to compute a derivative (use FFT to compute $\tilde{f}_k$)

integration: $\int_0^{x_j} I_N f(x)\, dx = x_j \tilde{f}_0 + \sum_{\substack{k \in K_N \\ k \neq 0}} \frac{\tilde{f}_k}{ik} e^{ikx_j}$ ← the mean $\tilde{f}_0$ must be zero for the result to be periodic

interpolation error: $\|I_N f - f\|^2 = \|I_N f - P_N f\|^2 + \|P_N f - f\|^2 \xrightarrow[N\to\infty]{\text{rapidly}} 0$

pythagorean theorem (these are orthogonal) ⇒

$\|I_N f - P_N f\|^2$: $\sum_{k \in K_N} (\tilde{f}_k - \hat{f}_k) e^{ikx}$

$\|P_N f - f\|^2$: $\sum_{k > N \text{ or } k \le -\frac{N}{2}} \hat{f}_k e^{ikx}$

as opposed to Fourier-Galerkin methods that involve projecting the nonlinearity back into the Fourier space to eliminate aliasing errors

Fourier collocation, pseudo-spectral methods (do this in HW7)

(1) represent the approximate solution in physical space, not Fourier space

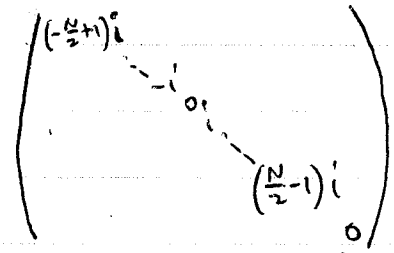
(2) discretize in space first $u_j(t) \approx u(\frac{2\pi j}{N}, t)$, then in time $u_j^n \approx u(\frac{2\pi j}{N}, t_n)$ (method of lines)

(3) compute space derivatives using the FFT.

$$D_N = \mathcal{F}_N^{-1} \Lambda_N \mathcal{F}_N$$

IFFT ↑ FFT

$$\frac{d}{dx} I_N u(x,t) = \sum_{k \in K_N} ik \hat{u}_k(t) e^{ikx}$$



zero out Nyquist frequency $k = \frac{N}{2}$

this discrete derivative operator is equivalent to the application of a skew symmetric matrix

$$D_N^* = -D_N, \quad (D_N)_{lj} = \begin{cases} \frac{1}{2}(-1)^{l+j} \cot\left(\frac{(l-j)\pi}{N}\right) & l \neq j \\ 0 & l = j \end{cases}$$

and the eigenvalues of D_N are $\lambda_k = \begin{cases} ik & -\frac{N}{2}+1 \leq k \leq \frac{N}{2}-1 \\ 0 & k = N/2 \end{cases}$

(4) evaluate non-linearities on the grid, e.g. to compute uu_x, evaluate $w_j = \frac{1}{2}u_j^2$ for $j = 0..N-1$ and then compute $D_N w$

(5) choose an ODE method suitable for solving the resulting finite dimensional ODE

$$\frac{du}{dt} = L(u), \quad u = \{u_j\}_{j=0}^{N-1}$$

for example, if $L(u) = D_N^2 u$ (heat equation), the eigenvalues of the $n \times n$ matrix D_N^2 are $\lambda_k = -k^2$, $-\frac{N}{2}+1 \leq k \leq \frac{N}{2}-1$

so $h = \Delta t$ must be small enough that each $\lambda_k \in$ RAS of method.

ODE's to solve:

Burgers': $u_t = -D_N w + \nu D_N^2 u$

KdV: $u_t = -D_N w - \nu D_N^3 u$

$w_j = \frac{1}{2} u_j^2 \qquad u = \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix} \in \mathbb{R}^N$

hints on the homework

Burgers': $u_t = -uu_x + \nu u_{xx} = f(u) + g(u)$

KdV: $u_t = -uu_x - \nu u_{xxx} = f(u) + g(u)$

multistep IMEX:

$$a_0 u^{n+s} + \cdots + a_s u^n = h\left[b_1 f^{n+s-1} + \cdots + b_s f^n + c_0 g^{n+s} + \cdots + c_s g^n \right]$$

divide coefficients by a_0 first (so assume $a_0 = 1$ now). Have to solve

$$\underbrace{(I - c_0 h g)}_{\mathcal{F}_N^{-1}(I - c_0 h \hat{g})\mathcal{F}_N} u^{n+s} = v := \underbrace{-a_1 u^{n+s-1} - \cdots - a_s u^n + h\left[b_1 f^{n+s-1} + \cdots + b_s f^n + c_1 g^{n+s-1} + \cdots + c_s g^n\right]}_{\text{known already}}$$

$$\boxed{u^{n+s} = \mathcal{F}_N^{-1}(I - c_0 h \hat{g})^{-1} \mathcal{F}_N v} \qquad g = \mathcal{F}_N^{-1} \hat{g} \mathcal{F}_N$$

Burgers': $g = \nu D_N^2$, $\hat{g} = \mathrm{diag}(-\nu k^2)$, $(I - c_0 h \hat{g})^{-1} = \mathrm{diag}(1 + h c_0 \nu k^2)^{-1}$

KdV: $g = -\nu D_N^3$, $\hat{g} = \mathrm{diag}(i \nu k^3)$, $(I - c_0 h \hat{g})^{-1} = \mathrm{diag}(1 - i h c_0 \nu k^3)^{-1}$

IMEX RK

have to solve $l_i = g\Big(\overbrace{u^n + h\sum_{j=1}^{i-1} a_{ij} k_j + h\sum_{j=1}^{i-1} \hat{a}_{ij} l_j}^{v \text{ (known)}} + h\hat{a}_{ii} l_i\Big)$

$= g(v) + h\hat{a}_{ii} g(l_i)$

(two kinds of i, sorry...)

$(I - h\hat{a}_{ii} g) l_i = gv$

$$\boxed{l_i = (I - h\hat{a}_{ii} g)^{-1} g v = \mathcal{F}_N^{-1}\left[(I - h\hat{a}_{ii}\hat{g})^{-1}\hat{g}\right]\mathcal{F}_N v}$$

Burgers': $(I - h\hat{a}_{ii}\hat{g})^{-1}\hat{g} = \mathrm{diag}\left(\frac{-\nu k^2}{1 + h\hat{a}_{ii}\nu k^2}\right)$

KdV: $(I - h\hat{a}_{ii}\hat{g})^{-1}\hat{g} = \mathrm{diag}\left(\frac{i\nu k^3}{1 - ih\hat{a}_{ii}\nu k^3}\right) = \mathrm{diag}\left(\frac{\nu k^3(-h\hat{a}\nu k^3 + i)}{1 + (h\hat{a}\nu k^3)^2}\right)$

coping with matlab's fft

our definition: $\tilde{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ikj\frac{2\pi}{N}} \qquad -\frac{N}{2}+1 \le k \le \frac{N}{2}$

matlab: $\tilde{f}_k = \sum_{j=1}^{N} f_j e^{-i(j-1)(k-1)\frac{2\pi}{N}} \qquad 1 \le k \le N$

so you just have to be careful of "off by one" errors and map the indices above the Nyquist frequency to their proper negative values by subtracting N. (The $\frac{1}{N}$ out front is unimportant since the inverse transform corrects it)

if we want to compute $uu_x = \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right)$ on the grid, we:

define $w_j = \frac{1}{2}u_j^2 \qquad 1 \le j \le N$

define $\xi_k = \begin{cases} k-1 & 1 \le k \le N/2 \\ k-1-N & \frac{N}{2}+1 \le k \le N \end{cases}$

$N=8$ example

k	1	2	3	4	5	6	7	8
ξ_k	0	1	2	3	± 4	-3	-2	-1

↑ Nyquist always ambiguous. (zero it out)

define $\hat{w} = \text{fft}(w)$

$\longrightarrow \quad w(x) = \frac{1}{N}\sum_{k=1}^{N} \hat{w}_k e^{i\xi_k x}$

so $\frac{dw}{dx}(x) = \frac{1}{N}\sum_{k=1}^{N} i\xi_k \hat{w}_k e^{i\xi_k x}$

set $\hat{z}_{\text{Nyquist}} = 0$

define $\hat{z}_k = i\xi_k \hat{w}_k \qquad 1 \le k \le N$

$x_j = (j-1)\frac{2\pi}{N}$

define $z = \text{ifft}(\hat{z})$. Then $z_j \approx u(x_j)u_x(x_j)$

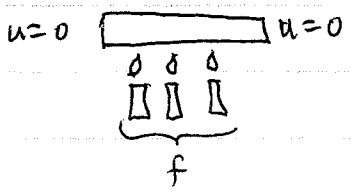
the heat equation and the Poisson equation

heat equation: $u_t = u_{xx} + f$ $\quad f$ = heat source

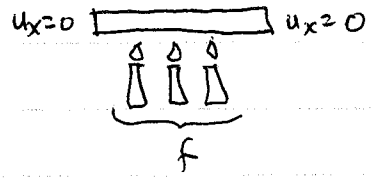
Poisson equation (steady state): $-u_{xx} = f$

common boundary conditions:

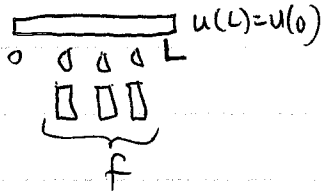
Dirichlet:



Neumann (insulating)



periodic



the exact solution can be found by expanding f in terms of the eigenfunctions of the laplacian.

periodic case: $\left.\begin{array}{l} -u_{xx} = \lambda u \\ u(0) = u(L) \\ u'(0) = u'(L) \end{array}\right\} \Rightarrow \varphi_k(x) = e^{2\pi i k \frac{x}{L}}, \quad \lambda_k = \left(\frac{2\pi k}{L}\right)^2 \qquad k \in \mathbb{Z}$

Dirichlet: $\left.\begin{array}{l} -u_{xx} = \lambda u \\ u(0) = u(L) = 0 \end{array}\right\} \Rightarrow \varphi_k(x) = \sin\frac{k\pi x}{L}, \quad \lambda_k = \left(\frac{k\pi}{L}\right)^2 \qquad k = 1, 2, 3, \ldots$

Neumann: $\left.\begin{array}{l} -u_{xx} = \lambda u \\ u_x(0) = u_x(L) = 0 \end{array}\right\} \Rightarrow \varphi_k(x) = \cos\frac{k\pi x}{L}, \quad \lambda_k = \left(\frac{k\pi}{L}\right)^2 \qquad k = 0, 1, 2, 3, \ldots$

expand $f(x) = \sum_k \hat{f}_k \varphi_k(x)$

solution of Poisson equation: $u(x) = \sum_{k \neq 0} \lambda_k^{-1} \hat{f}_k \varphi_k(x)$

(need $\hat{f}_0 = 0$ in Neumann & periodic cases)

for the heat equation, we also need an initial condition

$$u_0(x) = \sum_k \hat{u}_{0k} \varphi_k(x)$$

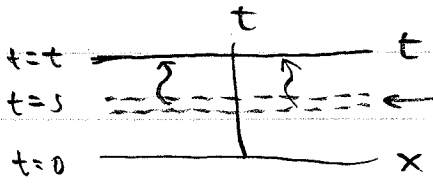
exact solution of $u_t = u_{xx}$ is $\quad u(x,t) = \sum_k \hat{u}_{0k} e^{-\lambda_k t} \varphi_k(x)$

if we define the evolution operator $\quad E(t)u_0 = \sum_k \hat{u}_{0k} e^{-\lambda_k t} \varphi_k(x)$

then we can use Duhamel's principle to solve

$$u_t = u_{xx} + f$$

solution: $\quad u(x,t) = E(t)u_0 + \int_0^t E(t-s) f(x,s)\, ds$



propagate the source forward like an initial condition starting at time s.

if f is independent of time, then

$$u(x,t) = \sum_k \hat{u}_{0k} e^{-\lambda_k t} \varphi_k(x) + \sum_k \underbrace{\frac{1 - e^{-\lambda_k t}}{\lambda_k}}_{t \text{ if } \lambda_k = 0} \hat{f}_k \varphi_k(x)$$

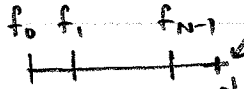
and we recover the Poisson solution as $t \to \infty$ (assuming $\hat{f}_0 = 0$)

spectral methods for these problems basically amount to evaluating the Fourier coefficients with the FFT and using the exact formulas.

It turns out we need slightly different spaces for the various boundary conditions

---

periodic case (N dof)



inner product $(f,g) = \frac{1}{N}\sum_{j=0}^{N-1} f_j \bar{g}_j$

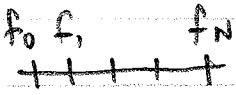
Dirichlet ($N-1$ dof)

$f_0 = f_{N-1} = 0$ not represented

inner product: $(f,g) = \frac{1}{N}\sum_{j=1}^{N-1} f_j \bar{g}_j$

we still divide by N (rather than $N-1$) since there are still N subintervals between $x=0$ and $x=L$ (there's a reflective symmetry relating this case to the periodic case on a larger interval. Changing N would break the relationship)

Neumann: ($N+1$ dof)

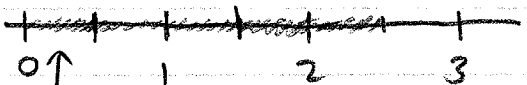


← all values can be assigned independently

inner product:

$$(f,g) = \frac{1}{2N}\left[f_0\bar{g}_0 + f_N\bar{g}_N\right] + \frac{1}{N}\sum_{j=1}^{N-1} f_j\bar{g}_j$$

the $\frac{1}{2}$ is due to the end intervals being cut in half (again to make the symmetry work out with periodic case)



half as much mass is associated with the endpoints (vs interior points)

---

with these discrete inner products, our previous basis functions (sampled on the grid) turn out to be orthogonal.

discrete orthonormal bases: $x_j = \frac{jL}{N}$

periodic case: $\varphi_k(x_j) = e^{2\pi i k x_j / L}$ $\quad 0 \le j \le N-1$, $\quad -\frac{N}{2}+1 \le k \le \frac{N}{2}$ $\quad \left(k = N/2 \text{ Nyquist } +1, -1, +1, -1, \ldots\right)$

Dirichlet case: $\varphi_k(x_j) = \sqrt{2} \sin \pi k \frac{x_j}{L}$ $\quad 1 \le j \le N-1$, $\quad 1 \le k \le N-1$

Neumann case: $\varphi_k(x_j) = \begin{cases} 1 & k=0 \\ \sqrt{2}\cos \pi k \frac{x_j}{L} & k>0 \end{cases}$ $\quad 0 \le j \le N$, $\quad 0 \le k \le N$ $\quad \left(k = N \text{ Nyquist } +1, -1, +1, -1, \ldots\right)$

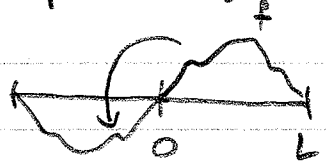
The FFT (Fourier), FST (Sine), FCT (Cosine) transforms are fast algorithms ($O(n \log n)$) for computing all the Fourier coefficients simultaneously (or vice-versa)

$$f(x_j) = \sum \hat{f}_k \varphi_k(x_j)$$

(and hence the accuracy of the spectral method)

the decay of the Fourier coefficients $\hat{f}$ in the Dirichlet and Neumann cases depends on how smooth the function is when extended via symmetry (using our previous theory for the periodic case)

Dirichlet: extend using odd symmetry

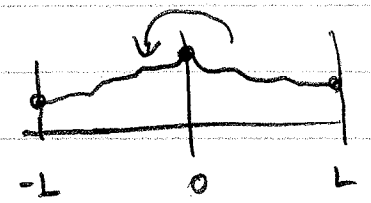


$f(-x) = -f(x)$

(note: $f(x) \equiv 1$ becomes discontinuous at $x=0$ and $x=L$. Fourier coefficients decay like $1/k$ bad news for spectral method)

problem: f doesn't satisfy the b.c.'s

Neumann: extend using even symmetry



$f(-x) = f(x)$

the procedure for solving $-u_{xx} = f$, $u(0) = u(L) = 0$ spectrally is:

(1) discretize space, evaluate $f_j = f\left(\frac{jL}{N}\right)$ $\quad 1 \le j \le N-1$

(2) use the FST to compute $\hat{f}_k = \frac{1}{N} \sum_{j=1}^{N-1} f_j \sqrt{2} \sin \frac{\pi k j}{N}$ $\quad 1 \le k \le N-1$

(3) define $\hat{u}_k = \lambda_k^{-1} \hat{f}_k$, $\lambda_k = \left(\frac{k\pi}{L}\right)^2$ $\quad 1 \le k \le N-1$

(4) use the IFST to compute $u_j = \sum_{k=1}^{N-1} \hat{u}_k \sqrt{2} \sin \frac{\pi k j}{N}$ $\quad 1 \le j \le N-1$

the only error occurs at step 2 where we commit aliasing and truncation errors. The procedure for the other boundary conditions and the heat equation is similar. For the heat equation you can either timestep the ODE using the method of lines, or if f is independent of time, just use the <u>exact formula</u>

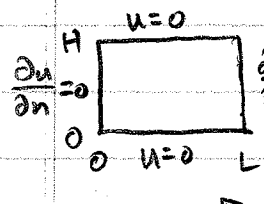
$$u(x_j, t) = \sum_k \hat{u}_{0k} e^{-\lambda_k t} \varphi_k(x_j) + \sum_k \frac{1 - e^{-\lambda_k t}}{\lambda_k} \hat{f}_k \varphi_k(x_j)$$

the same thing can be done in 2d on a rectangle or 3d in a box

heat $\rightarrow$ $u_t = \Delta u + f$ $\qquad \Delta u = u_{xx} + u_{yy}$ $(+ u_{zz})$

Poisson $\rightarrow$ $-\Delta u = f$

the eigenfunctions of the Laplacian are just tensor products of the 1d eigenfunctions, e.g.:



mixed b/c's

$$-\Delta u = \lambda u \rightarrow \begin{cases} u_{k,l}(x,y) = \varphi_k(x)\,\psi_l(y) & k = 0,1,2,\ldots \\ & l = 1,2,3,\ldots \\ \lambda_{k,l} = \left(\frac{k\pi}{L}\right)^2 + \left(\frac{l\pi}{H}\right)^2 \end{cases}$$

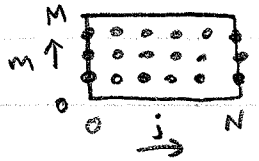
$$\varphi_k(x) = \begin{cases} 1 & k = 0 \\ \sqrt{2}\cos\frac{k\pi x}{L} & k > 0 \end{cases}$$

$$\psi_l(y) = \sqrt{2}\sin(l\pi y / H)$$

exact solution of Poisson is: $u(x,y) = \sum_{k,l} \lambda_{k,l}^{-1} \hat{f}_{k,l} \varphi_k(x) \psi_l(y)$

↖ omit $(k,l) = (0,0)$ in the [Neumann-Neumann, Neumann-Periodic, Periodic-Periodic] cases ($\hat{f}_{0,0}$ must be zero for a solution to exist)

and our spectral method boils down to discretizing $f(x_j, y_m)$ on the grid and computing the Fourier coefficients one direction at a time



$$\hat{f}_{k,l} = \frac{1}{M} \sum_{m=1}^{M-1} \left[\frac{1}{2N} \left(f_{0,m} \overline{\varphi_k(x_0)} + f_{N,m} \overline{\varphi_k(x_N)} \right) + \frac{1}{N} \sum_{j=1}^{N-1} f_{j,m} \overline{\varphi_k(x_j)} \right] \overline{\psi_l(y_m)}$$

↑ sine transform in y-direction of result.

cosine transform in x-direction holding m constant

Spectral methods are the most accurate and fastest methods when they are applicable, but they are the least flexible.

→ domain must be rectangular

→ the source must be smooth (so Fourier coefficients decay fast)

→ the equation must be constant coefficient

$$\nabla \cdot (\sigma \nabla u) = f \quad \text{not allowed with variable } \sigma(x,y)$$

→ some boundary conditions break the method

e.g. $u_t = u_{xx} + u_x$, $u(0) = u(L) = 0$

this equation would be fine with periodic b/c's, but with Dirichlet conditions the u_x term breaks the reflective symmetry and the basis functions $\sin\left(\frac{k\pi x}{L}\right)$ cease to be effective (high frequency fourier modes grow → truncation and aliasing errors become large)

Iterative methods for linear systems

Chris H. Rycroft*

November 20th, 2007

Introduction

For many elliptic PDE problems, finite-difference and finite-element methods are the techniques of choice. In a finite-difference approach, we search for a solution u_k on a set of discrete gridpoints $1, \ldots, k$. The discretized partial differential equation and boundary conditions give us linear relationships between the different values of u_k. In finite-element method, we express our solution as a linear combination u_k of basis functions λ_k on the domain, and the corresponding finite-element variational problem again gives linear relationships between the different values of u_k.

Regardless of the precise details, all of these approaches ultimately end up with having to find the u_k which satisfy all the linear relationships prescribed by the PDE. This can be written as a matrix equation of the form

$$Au = b$$

where we wish to find a solution u, given that A is a matrix capturing the differentiation operator, and b corresponds to any source or boundary terms. Theoretically, this problem could be solved on a computer by any of the standard methods for dealing with matrices. However, the real challenge for PDEs is that frequently, the dimensionality of the problem can be enormous. For example, for a two dimensional PDE problem, a 100×100 grid would be a perfectly reasonable size to consider. Thus u would be a vector with 10^4 elements, and A would be a matrix with 10^8 elements. Even allocating memory for such a large matrix may be problematic. Direct approaches, such as the explicit construction of A^{-1}, are impractical.

The key to making progress is to note that in general, the matrix A is extremely sparse, since the linear relationships usually only relate nearby gridpoints together. We therefore seek methods which do not require ever explicitly specifying all the elements of A, but exploit its special structure directly. Many of these methods are *iterative* – we start with a guess u_k, and apply a process that yields a closer solution u_{k+1}.

*Electronic address: chr@math.berkeley.edu

Typically, these iterative methods are based on a *splitting* of A. This is a decomposition $A = M - K$, where M is non-singular. Any splitting creates a possible iterative process. We can write

$$\begin{aligned} Au &= b \\ (M-K)u &= b \\ Mu &= Ku + b \\ u &= M^{-1}Ku + M^{-1}b \end{aligned}$$

and hence a possible iteration is

$$u_{k+1} = M^{-1}Ku_k + M^{-1}b.$$

Of course, there is no guarantee that an arbitrary splitting will result in an iterative method which converges. To study convergence, we must look at the properties of the matrix $R = M^{-1}K$. For convergence analysis, it is helpful to introduce the *spectral radius*

$$\rho(R) = \max_j \{|\lambda_j|\}$$

where the λ_j are the eigenvalues of R. It can be shown [2] that an iterative scheme converges if and only if $\rho(R) < 1$. The size of the spectral radius determines the convergence rate, and ideally we would like to find splittings which result in as small a $\rho(R)$ as possible.

An example: a two dimensional Poisson problem

In the convergence analysis later, we will consider a two dimensional Poisson problem on the square $-1 \leq x \leq 1, -1 \leq y \leq 1$, given by the equation

$$-\nabla^2 u = f,$$

subject to the Dirichlet conditions that $u(x, y)$ vanishes on the boundary. We use a source function of the form

$$f(x,y) = \begin{cases} 1 & \text{if } |x| < 0.5 \text{ and } |y| < 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

This is plotted on a 33×33 grid in figure 1. For convergence properties, the eigenfunctions and eigenvalues of this function are very important, and to determine these, it is helpful to consider an associated one-dimensional Poisson problem on the interval $-1 \leq x \leq 1$,

$$-\frac{d^2u}{dx^2} = f(x),$$

subject to Dirichlet boundary conditions $u(-1) = u(1) = 0$. We consider a discretization into $N+2$ gridpoints such that $x_j = -1+2j/(N+1)$ for $j = 0, \ldots, N+1$. When constructing

the corresponding matrix problem, u_0 and u_{N+1} need not be considered, since their values are always fixed to zero. By discretizing the second derivative according to

$$\left.\frac{d^2u}{dx^2}\right|_{x=x_j} = \frac{u_{j-1} + u_{j+1} - 2u_j}{2h^2}$$

where $h = 2/N$, we can write the corresponding linear system as

$$T_N \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & & \\ -1 & 2 & -1 & \ddots & \\ 0 & -1 & \ddots & -1 & 0 \\ & \ddots & -1 & 2 & -1 \\ & & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{pmatrix} = 2h^2 \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

Motivated by previous lectures on the spectral method, we expect that the eigenvectors of T_N may be based on sine functions. A reasonable guess for the jth eigenfunction is

$$z_j(k) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi kj}{N+1}.$$

To verify this is an eigenfunction, and find its eigenvalue, we apply T_N to obtain

$$\left(T_N z_j\right)(k) = \sqrt{\frac{2}{N+1}} \left(2 \sin \frac{\pi kj}{N+1} - \sin \frac{\pi (k+1) j}{N+1} - \sin \frac{\pi (k-1) j}{N+1} \right).$$

Note that this expression will always be valid for the range $k = 1, 2, \ldots, N$, and the boundary values just work out. The last two sine functions can be rewritten using a trigonometric identity to give

$$\begin{aligned} \left(T_N z_j\right)(k) &= \sqrt{\frac{2}{N+1}} \left(2 \sin \frac{\pi kj}{N+1} - 2 \sin \frac{\pi kj}{N+1} \cos \frac{\pi j}{N+1} \right) \\ &= \sqrt{\frac{2}{N+1}} 2 \left(1 - \cos \frac{\pi j}{N+1} \right) \sin \frac{\pi kj}{N+1} \\ &= 2 \left(1 - \cos \frac{\pi j}{N+1} \right) z_j(k) \end{aligned}$$

and thus we see that z_j is an eigenvector with eigenvalue $\lambda_j = 2(1 - \cos \pi j/(N+1))$. The smallest eigenvalue is $\lambda_1 = 2(1 - \cos \pi/(N+1))$ and the largest is $\lambda_N = 2(1 - \cos N\pi/(N+1))$.

Returning to the two dimensional problem, we see that the corresponding derivative matrix $T_{N \times N}$ can be written as the tensor product of two one dimensional problems T_N. Its eigenvectors can be expressed as the tensor product of the one dimensional eigenvectors, namely

$$z_{i,j}(k, l) = z_i(k) z_j(l)$$

and their corresponding eigenvalues are

$$\lambda_{i,j} = \lambda_i + \lambda_j.$$

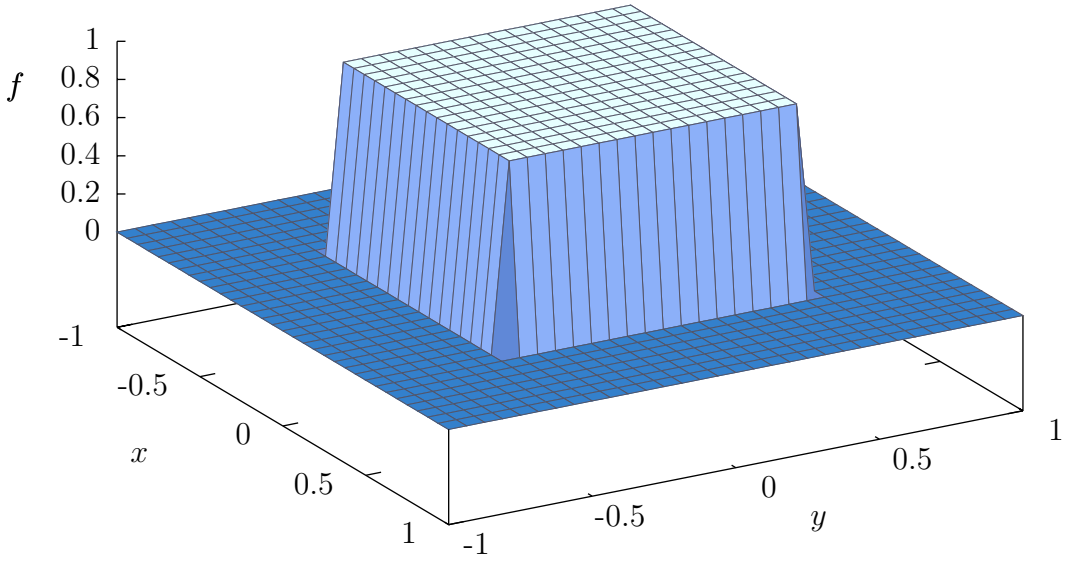


Figure 1: A sample source function $f(x, y)$ on a 33×33 grid.

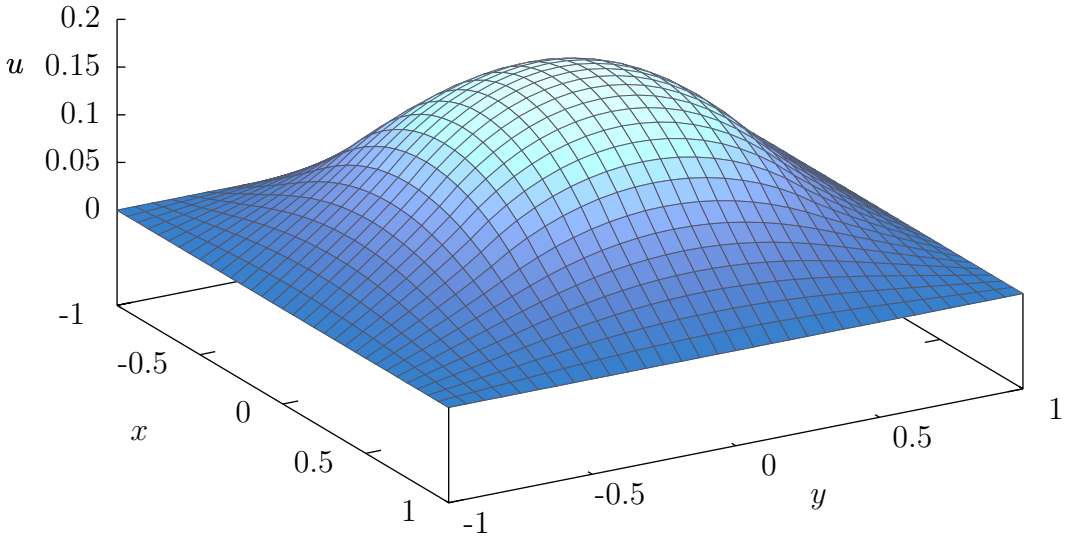


Figure 2: The exact solution to the 2D Poisson problem $-\nabla^2 u = f$, with zero boundary conditions and a source term given in figure 1.

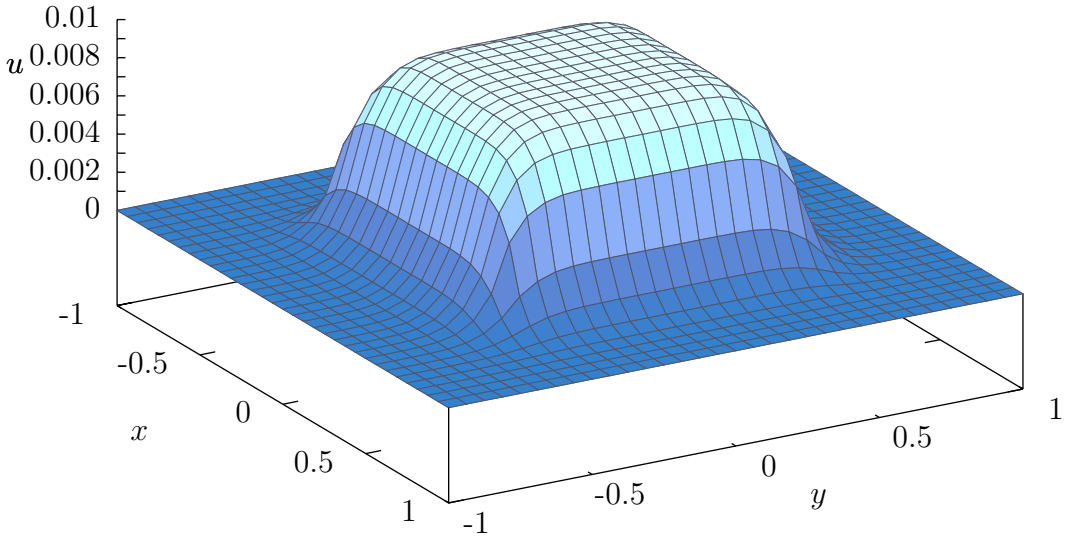


Figure 3: The solution to the example 2D Poisson problem after ten iterations of the Jacobi method.

The Jacobi Method

The Jacobi method is one of the simplest iterations to implement. While its convergence properties make it too slow for use in many problems, it is worthwhile to consider, since it forms the basis of other methods. We start with an initial guess u_0, and then successively improve it according to the iteration

for $j = 1$ to N **do**
  $u_{m+1,j} = \frac{1}{a_{jj}} \left(b_j - \sum_{k \neq j} a_{jk} u_{m,k} \right)$
end for

In other words, we set the jth component of u so that it would exactly satisfy equation j of the linear system. For the two dimensional Poisson problem considered above, this corresponds to an iteration of the form

for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    $u_{m+1,i,j} = \left(h^2 f_j + u_{m,i,j+1} + u_{m,i,j-1} + u_{m,i+1,j} + u_{m,i-1,j} \right) / 4$
  end for
end for

To find the corresponding matrix form, write $A = D - L - U$ where D is diagonal, L is lower-triangular, and U is upper-triangular. Then the above iteration can be written as

$$u_{m+1} = D^{-1}(L + U)u_m + D^{-1}b.$$

The convergence properties, discussed later, are then set by the matrix $R_J = D^{-1}(L + U)$.

The Jacobi method has the advantage that for each m, the order in which the components of u_{m+1} are computed has no effect – this may be a favorable property to have in some parallel implementations. However, it can also be seen that u_m must be retained until after u_{m+1} is constructed, meaning we must store u_{m+1} in a different part of the memory. The listing given in appendix A.1 carries out the Jacobi iteration on the Poisson test function. It makes use of two arrays for the storage of u, computing the odd u_k in one and the even u_k in the other. Figure 3 shows a the progress of the Jacobi method after ten iterations.

The Gauss–Seidel Method

The Gauss–Seidel method improves on the Jacobi algorithm, by noting that if we are updating a particular point $u_{m+1,j}$, we might as well reference the already updated values $u_{m+1,1},\ldots,u_{m+1,j-1}$ in the calculation, rather than using the original values $u_{m,1},\ldots,\ u_{m,j-1}$. The iteration can be written as:

for $j = 1$ to N **do**
  $u_{m+1,j} = \frac{1}{a_{jj}} \left(b_j - \sum_{k=1}^{j-1} a_{jk} u_{m+1,k} - \sum_{k=j+1}^{N} u_{m,k} \right)$
end for

The Gauss–Seidel algorithm has the advantage that in a computer implementation, we no longer need to allocate two arrays for u_{m+1} and u_m. Instead, we can make just a single array for u_m, and carry out all the updates *in situ*. However, the Gauss–Seidel implementation introduces an additional complication that the order in which the updates are applied will affect the values of u_m. For a two dimensional problem, two particular orderings are worth special attention:

- *Natural ordering* – this is the typical ordering that would result in a **for** loop. We first loop successively through all elements of the first row $(1, 1), \ldots, (1, n)$ before moving onto the second row, and so on.

- *Red–Black ordering* – this is the ordering that results by coloring the gridpoints red and black in a checkerboard pattern. Specifically, we color a gridpoint (i, j) red if $i + j$ is even, and black if $i + j$ is odd. During the Gauss–Seidel update, all red points are updated before the black points. For the two dimensional Poisson problem, we see that updating a red grid point only requires information from the black gridpoints, and vice versa. Hence the order in which points in each set are updated does not matter. We can think of the whole Gauss–Seidel update being divided into a red grid point update and black gridpoint update, and this can be helpful in the convergence analysis.

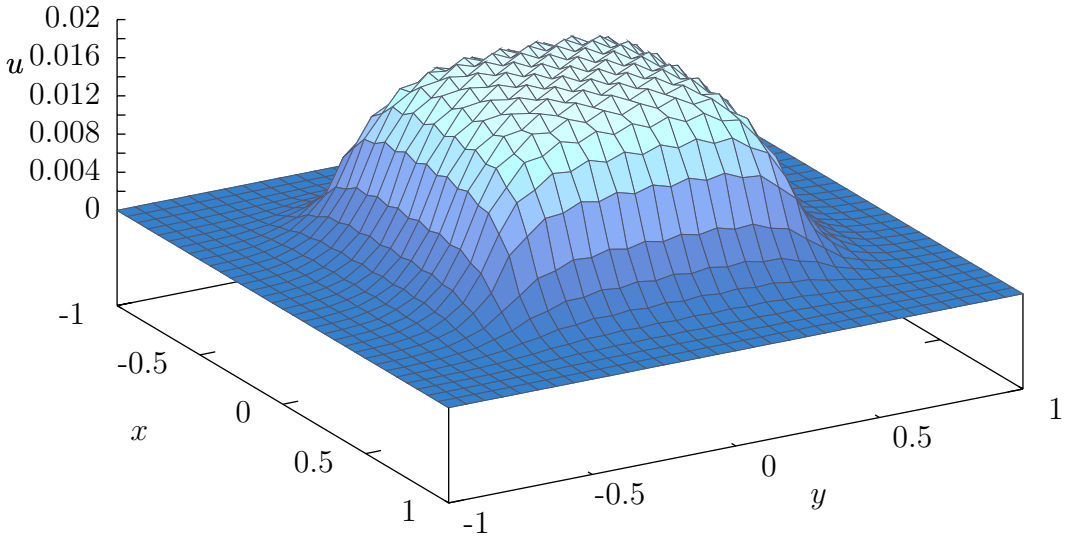


Figure 4: The Gauss–Seidel solution to the example 2D Poisson problem after ten iterations. The crinkles in the solution are due to the Red–Black update procedure.

From the algorithm above, we can write down the corresponding matrix splitting for the Gauss–Seidel method as

$$\begin{aligned}(D-L)u_{m+1} &= Uu_m + b \\ u_{m+1} &= (D-L)^{-1}Uu_m + (D-L)^{-1}b.\end{aligned}$$

Appendix A.2 contains a C++ code to carry out a Gauss–Seidel method on the example problem, and the result after ten iterations is shown in figure 4.

Successive Over-Relaxation

Successive Over-Relaxation (SOR) is a refinement to the Gauss–Seidel algorithm. At each stage in the Gauss–Seidel algorithm, a value $u_{m,j}$ is updated to a new one $u_{m+1,j}$, which we can think of as displacing $u_{m,j}$ by an amount $\Delta u = u_{m+1,j} - u_{m,j}$. The SOR algorithm works by displacing the values by an amount $\omega\Delta u$, where typically $\omega > 1$, in the hope that if Δu is a good direction to move in, we might as well move further in that direction. The iteration can be written as:

for $j = 1$ to N **do**
$u_{m+1,j} = (1-\omega)u_{m,j} + \frac{\omega}{a_{jj}}\left(b_j - \sum_{k=1}^{j-1} a_{jk}u_{m+1,k} - \sum_{k=j+1}^{N} u_{m,k}\right)$
end for

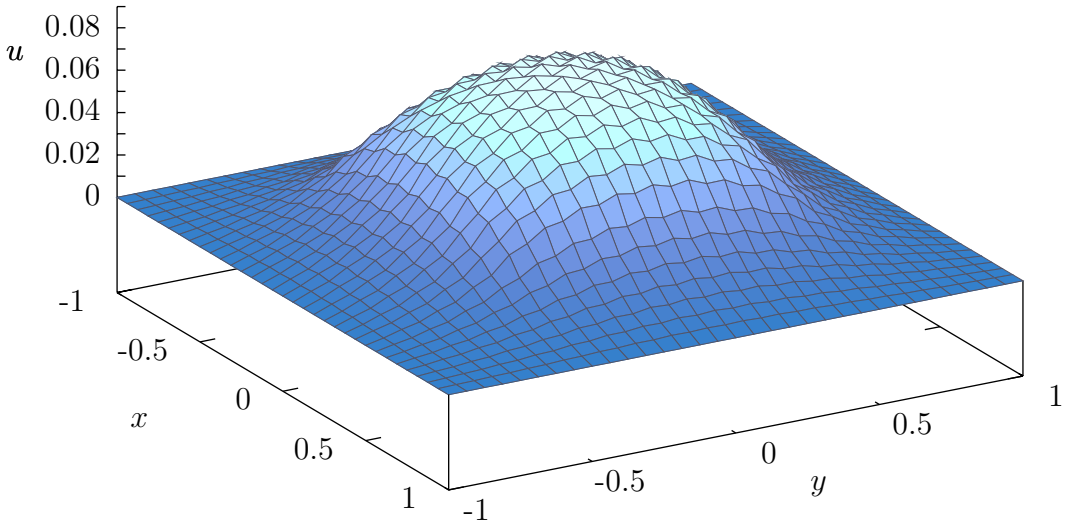


Figure 5: The SOR solution (using the theoretically optimal ω) to the example 2D Poisson problem after ten iterations. The solution is closer to the answer than the Jacobi or Gauss–Seidel methods.

The corresponding matrix form is

$$\begin{aligned}(D+\omega L)u_{m+1} &= [(1-\omega)D - U\omega]\,u_m + \omega b \\ u_{m+1} &= (D+\omega L)^{-1}\,[(1-\omega)D - U\omega]\,u_m + (D+\omega L)^{-1}\omega b.\end{aligned}$$

Appendix A.3 contains a C++ code to carry out the SOR iteration on the example problem, and the result is shown in figure 5. In the SOR algorithm, we are free to choose the value of ω, and the best choices can be found by considering the eigenfunctions of the associated problem. This is discussed in more detail below.

Convergence analysis and complexity

To examine the convergence properties of the different methods, we need to look at the associated spectral radii. For the Jacobi method, we had $R_J = D^{-1}(L+U)$. For the 2D Poisson problem, $D = 4I$ so we can write $R_J = (4I)^{-1}(4I - T_{N\times N}) = I - T_{N\times N}/4$. The largest eigenvalue of R_J corresponds to the smallest of $T_{N\times N}$, namely

$$\begin{aligned}\lambda_{1,1} &= 4 - 2\cos\left(\frac{\pi}{N+1}\right) - 2\cos\left(\frac{\pi}{N+1}\right) \\ &= 4 - 4\cos\left(\frac{\pi}{N+1}\right).\end{aligned}$$

Hence

$$\rho(R_J) = \cos\left(\frac{\pi}{N+1}\right).$$

A Taylor series expansion shows us that $\rho(R_J) = 1 - 2\pi^2/(N+1)^2$. The time for us to gain an extra digit of accuracy is approximately

$$\frac{1}{\log_{10}\rho(R_J)} \propto N^2.$$

Thus we must run the algorithm for $O(N^2)$ iterations. Since there are $O(N^2)$ gridpoints for the 2D problem, the total running time is $O(N^4)$. For detailed proofs of the convergence properties of the other methods, the reader should refer elsewhere [2]. It can be shown that

$$\rho(R_{GS}) = \cos^2\left(\frac{\pi}{N+1}\right),$$

so that one iteration of the Gauss–Seidel method is equivalent to two Jacobi iterations. Note however the complexity is the same: we still need $O(N^2)$ iterations. For the SOR algorithm, it can be shown that the optimal value of ω is

$$\frac{2}{1+\sqrt{1-\rho(R_J)^2}}.$$

and that for this value,

$$\rho(R_{SOR}) \approx 1 - 2\frac{2\pi}{N+1}.$$

Since there is a factor of N in the denominator as opposed to N^2, the order of computation decreases to $O(N)$ per grid point.

Figure 6 shows a plot of mean square error against the number of iterations for the model problem with the Jacobi, Gauss–Seidel, and optimal SOR method. The lines agree with the above results. The SOR method reaches numerical precision within 1200 iterations, while the other two methods have not fully converged even after 10^4 iterations.

Multigrid

One of the major problems with the three methods considered so far is that they only apply locally. Information about different cell values only propagates by one or two gridpoints per iteration. However, for many elliptic problems, a point source may cause an effect over the entire domain. The above methods have a fundamental limitation that they will need to be applied for at least at many iterations as it takes for information to propagate across the grid. As such, we should not expect to ever do better than $O(N)$ operations per point. This can also be seen by considering the eigenvalues. The maximal eigenvalue of R_J was set by the $\lambda_{1,1}$, corresponding to the lowest order mode. While the methods may effectively damp

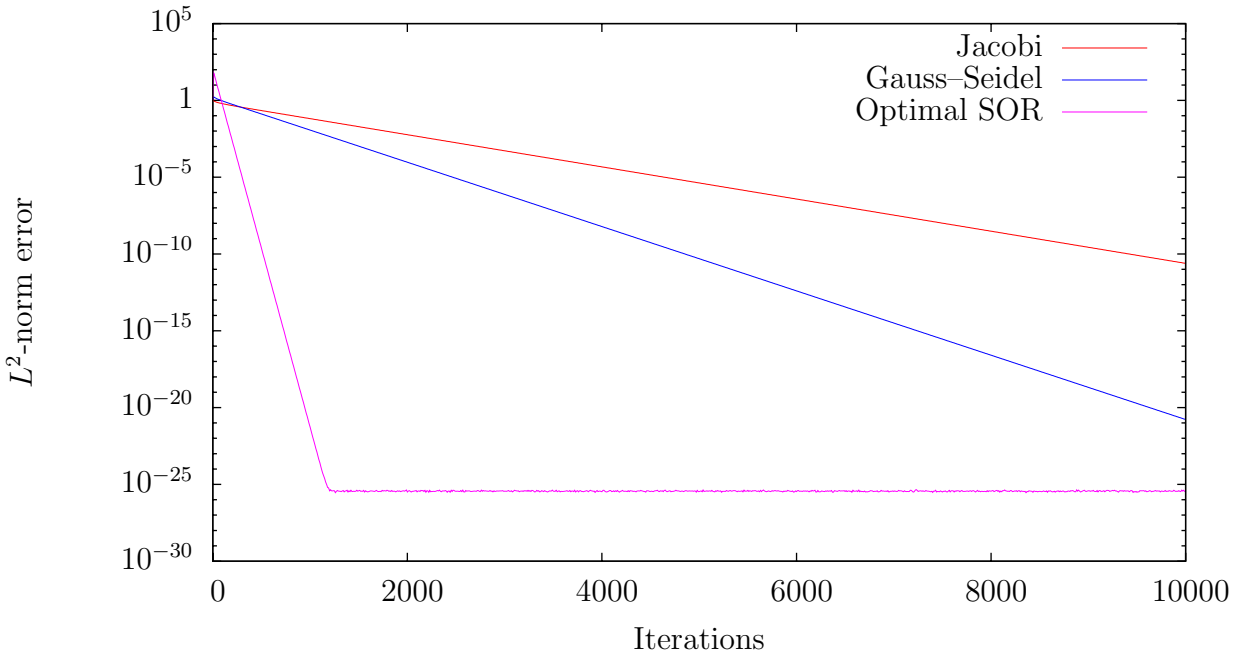


Figure 6: Errors versus the number of effective iterations for the Jacobi, Gauss–Seidel, and SOR methods, applied to the example 2D Poisson problem on a 65×65 grid. The plots are in line with the theoretical results of the text. The Gauss–Seidel method is faster than the Jacobi method, but has still not reached double numerical precision after 10000 iterations. The SOR method is significantly faster, but still requires 1200 iterations to reach double numerical precision.

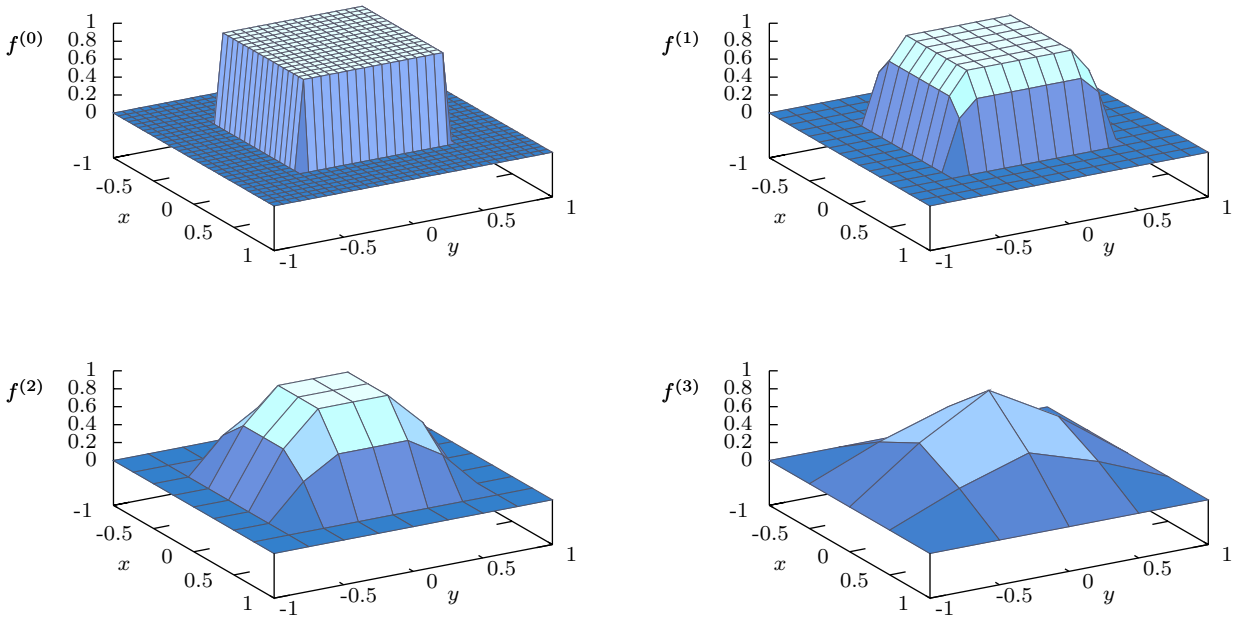


Figure 7: The restriction of the example source term f using the multigrid method with $v_{\text{down}} = 0, v_{\text{up}} = 2$. Top left: the initial grid, with 33×33 gridpoints. Top right: grid 1, with 17×17 gridpoints. Bottom left: grid 2, with 9×9 gridpoints. Bottom right: grid 3, with 5×5 gridpoints.

out high frequency oscillations, it will take a very long time to correctly capture the lowest modes with the largest wavelengths.

The multigrid method circumvents these limitations by introducing a hierarchy of coarser and coarser grids. Typically, at each level, the number of gridpoints is reduced by a factor of two in each direction, with the coarsest grid having ten to twenty points. To find a solution, we restrict the source term to the coarse grids, refine the solution on each, and interpolate up to the original grid. On the coarser grids, the lower frequency modes in the final solution can be dealt with much more effectively. Since the coarser grids have progressively fewer gridpoints, the time spent computing them is minimal. Because of this, the multigrid algorithm requires only $O(1)$ computation per point, which is the best order of complexity that we could hope for.

To be more specific, we let the original problem be on grid 0, and we then introduce a sequence of other successively coarser grids $1, \ldots, g$. We write $u^{(i)}$ and $b^{(i)}$ to represent the solution and source terms on the ith grid. A multigrid algorithm requires the following:

- A solution operator $S(u^{(i)}, b^{(i)})$ which returns a better approximation $u^{(i)}$ to the solution on the ith level
- An interpolation operator $T(u^{(i)})$ which returns an interpolation $u^{(i-1)}$ on the $(i-1)$th level

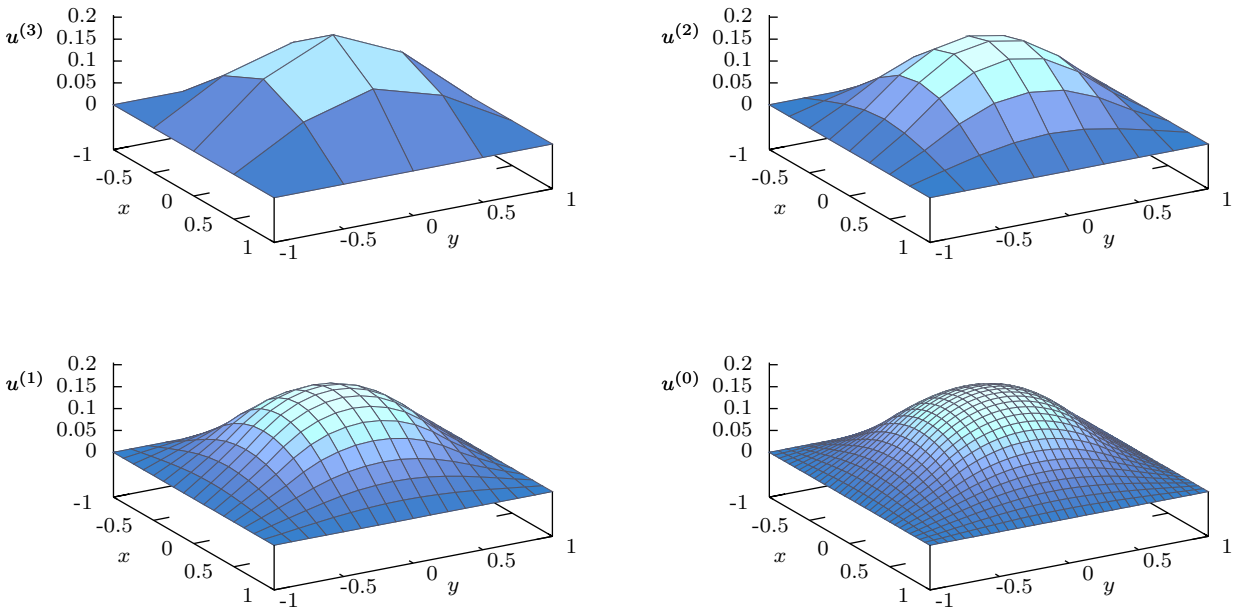


Figure 8: The solution of the v in the first multigrid V-cycle for the example 2D Poisson problem. Top left: the exact solution on grid 3 to the $f^{(3)}$ source term in figure 7. Top right: the interpolation and refinement on grid 2. Bottom left: the interpolation and refinement on grid 1. Bottom right: the interpolation and refinement on grid 0. Even after a single V-cycle, the solution is closer to the exact solution than the plots of 3, 4, and 5.

- A restriction operator $R(b^{(i)})$ which returns a restriction $b^{(i+1)}$ on the $(i+1)$th level

The interpolation and restriction operators can be thought of as rectangular matrices. As an example, consider a problem with 9 equally-spaced gridpoints on the unit interval, at $(0, 1/8, 2/8, \ldots, 1)$. Let grid 1 have 5 points at $(0, 1/4, 2/4, 3/4, 1)$, and let grid 2 have 3 gridpoints at $(0, 1/2, 1)$. Interpolation operators between the grids can be written as

$$T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \qquad T^{(2)} \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

where we are keeping the values at gridpoints common between the two levels, and introducing extra ones at the midpoints between each. Similarly, the restriction operators can be written as

$$R^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad R^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

It should be noted that without the boundary points, $R^{(i)}$ is the proportional to the transpose of $T^{(i+1)}$, and this property is very useful in proving the convergence of the multigrid method. However, this property is not strictly necessary to create an efficient multigrid algorithm. Other methods of interpolating and restricting are also possible, and it should also be noted that while a grid of size $2^n + 1$ has a particularly convenient multigrid formulation, the multigrid method can be applied to grids of arbitrary size. Given a differential operator matrix $A = A^{(0)}$ on the top level, we can define corresponding matrices on the lower levels according to

$$A^{(i)} = R^{(i-1)} A^{(i-1)} T^{(i)}.$$

With this definition, we can construct a solution operator $S(u^{(i)}, b^{(i)})$ as a single red–black Gauss–Seidel sweep. Given this, we can write a multigrid formulation as

function Multi$(u^{(i)}, b^{(i)})$
  if $i = g$ **then**
    compute exact solution to $A^{(i)} u^{(i)} = b^{(i)}$
    return $u^{(i)}$
  else

for $j = 1$ to v_{down} **do**
$u^{(i)} = S(u^{(i)}, b^{(i)})$
end for
$r^{(i)} = b^{(i)} - A^{(i)}u^{(i)}$
$d^{(i)} = T(\text{Multi}(0^{(i+1)}, R(r^{(i)})))$
$u^{(i)} = u^{(i)} + d^{(i)}$
for $j = 1$ to v_{up} **do**
$u^{(i)} = S(u^{(i)}, b^{(i)})$
end for
end if

The function is applied recursively, and at each stage, the remainder of the problem on the level above is sent to the lower level. The algorithm starts on level 0, descends to level g, and then ascends to level 0 again, following the shape of a V. It is therefore referred to as the *multigrid V-cycle*. Other more elaborate methods of moving between grids are possible, although the V-cycle is extremely efficient in many situations. In the algorithm, we are free to choose the number of times the solution operator is applied on the way down and on the way up, and typical good values to try may be $v_{\text{down}} = v_{\text{up}} = 2$, or even $v_{\text{down}} = 0, v_{\text{up}} = 2$. It may also be worthwhile to carry out more iterations on the coarser grids, since the computation is much cheaper there.

Figure 7 shows the restriction of the source term in the test problem on the coarser grids. Figure 8 shows the solution being successively refined on the grids. Even after a single V-cycle, the solution is close to the exact answer. Figure 9 shows the computation times for two different multigrid algorithms, compared with the previous three methods considered. The multigrid algorithms reach numerical precision extremely quickly, much faster even than SOR. Only twenty Gauss–Seidel iterations are applied at the top level before double numerical precision has been reached.

A Code listings

The following codes were used to generate the Jacobi, Gauss–Seidel, and SOR diagrams in these notes. They are written in C++ and were compiled using the GNU C++ compiler. Each of the first three routines calls a common code listed in appendix A.4 for setting up useful constants and defining common routines. This common code also contains a function for outputting the 2D matrices in a matrix binary format that is readable by the plotting program *Gnuplot* [1]. This output routine could be replaced in order to save to different plotting programs.

A.1 Jacobi method – jacobi.cc

```
// Load common routines and constants
#include "common.cc"
```

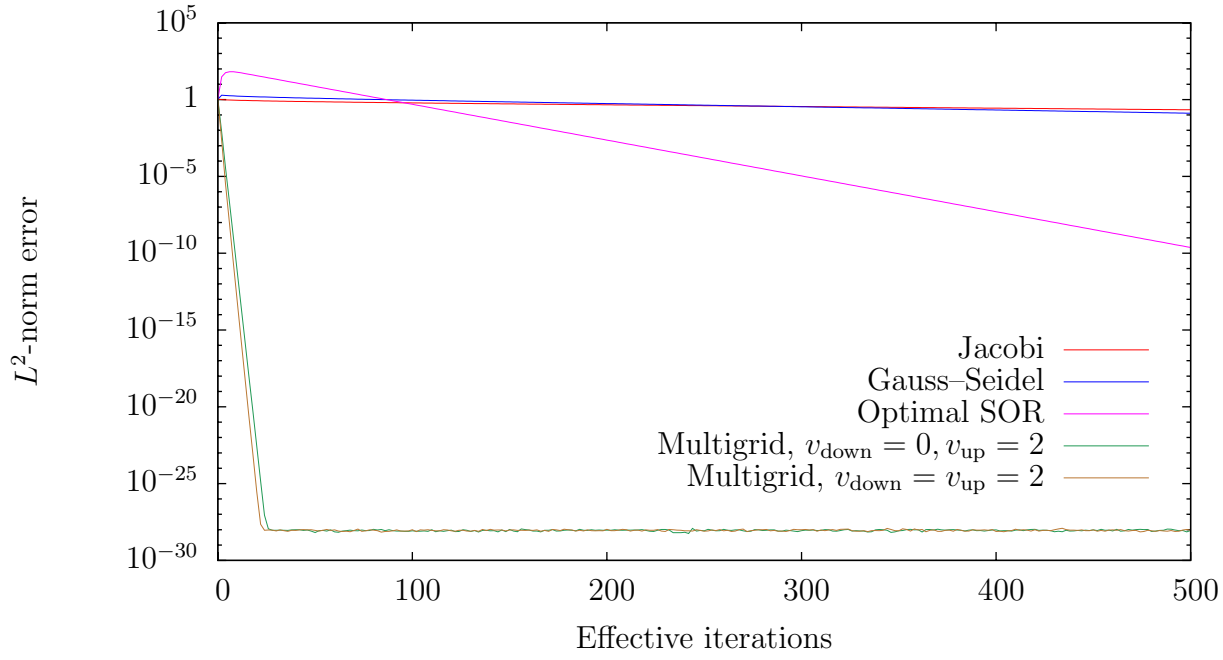


Figure 9: Errors versus the number of effective iterations for the several different iteration techniques. Here, to allow a direct comparison, "effective iterations" for the multigrid methods is defined by the number of Gauss–Seidel iterations that are applied on the top grid level, since the Gauss–Seidel iterations on the coarser grids are small in comparison. For the $v_{\text{down}} = 0, v_{\text{up}} = 2$ method, the effective number of iterations is twice the number of V-cycles. For the $v_{\text{down}} = v_{\text{up}} = 2$ method, the algorithm was slightly modified, so that only two Gauss–Seidel iterations were applied at the top level each time, instead of the expected four. Thus the effective number of iterations is also twice the number of V-cycles. The speed of the multigrid methods is startling when compared to any of the other three iterations.

```
int main() {
  int i,j,ij,k;
  double error,u[m*n],v[m*n],z;
  double *a,*b;

  // Set initial guess to be identically zero
  for(ij=0;ij<m*n;ij++) u[ij]=v[ij]=0;
  output_and_error("jacobi_out",u,0);

  // Carry out Jacobi iterations
  for(k=1;k<=total_iters;k++) {
    // Alternately flip input and output matrices
    if (k%2==0) {a=u;b=v;} else {a=v;b=u;}

    // Compute Jacobi iteration
    for(j=1;j<n-1;j++) {
      for(i=1;i<m-1;i++) {
        ij=i+m*j;
        a[ij]=(f(i,j)+dxxinv*(b[ij-1]+b[ij+1])
          +dyyinv*(b[ij-m]+b[ij+m]))*dcent;
      }
    }

    // Save and compute error if necessary
    output_and_error("jacobi_out",a,k);
  }
}
```

A.2 Gauss–Seidel – gsrb.cc

```
// Load common routines and constants
#include "common.cc"

int main() {
  int i,j,ij,k;
  double error,u[m*n],z;

  // Set initial guess to be identically zero
  for(ij=0;ij<m*n;ij++) u[ij]=0;
  output_and_error("gsrb_out",u,0);

  // Compute Red-Black Gauss-Seidel iteration
  for(k=1;k<=total_iters;k++) {
    for(j=1;j<n-1;j++) {
      for(i=1+(j&1);i<m-1;i+=2) {
```

```
        ij=i+m*j;
        u[ij]=(f(i,j)+dxxinv*(u[ij-1]+u[ij+1])
          +dyyinv*(u[ij-m]+u[ij+m]))*dcent;
      }
    }
    for(j=1;j<n-1;j++) {
      for(i=2-(j&1);i<m-1;i+=2) {
        ij=i+m*j;
        u[ij]=(f(i,j)+dxxinv*(u[ij-1]+u[ij+1])
          +dyyinv*(u[ij-m]+u[ij+m]))*dcent;
      }
    }

    // Save the result and compute error if necessary
    output_and_error("gsrb_out",u,k);
  }
}
```

A.3 Successive Over-Relaxation – sor.cc

```
// Load common routines and constants
#include "common.cc"

int main() {
  int i,j,ij,k;
  double error,u[m*n],z;

  // Set initial guess to be identically zero
  for(ij=0;ij<m*n;ij++) u[ij]=0;
  output_and_error("sor_out",u,0);

  // Compute SOR Red-Black iterations
  for(k=1;k<=total_iters;k++) {
    for(j=1;j<n-1;j++) {
      for(i=1+(j&1);i<m-1;i+=2) {
        ij=i+m*j;
        u[ij]=u[ij]*(1-omega)+omega*(f(i,j)
             +dxxinv*(u[ij-1]+u[ij+1])
             +dyyinv*(u[ij-m]+u[ij+m]))*dcent;
      }
    }
    for(j=1;j<n-1;j++) {
      for(i=2-(j&1);i<m-1;i+=2) {
        ij=i+m*j;
        u[ij]=u[ij]*(1-omega)+omega*(f(i,j)
             +dxxinv*(u[ij-1]+u[ij+1])
```

```
            +dyyinv*(u[ij-m]+u[ij+m]))*dcent;
      }
    }

    // Save the result and compute error if necessary
    output_and_error("sor_out",u,k);
  }
}
```

A.4 Common routine for setup and output – common.cc

```
// Load standard libraries
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

// Set grid size and number of iterations
const int save_iters=20;
const int total_iters=200;
const int error_every=2;
const int m=33,n=33;
const double xmin=-1,xmax=1;
const double ymin=-1,ymax=1;

// Compute useful constants
const double pi=3.14159265358979323846264338327795;
const double omega=2/(1+sin(2*pi/n));
const double dx=(xmax-xmin)/(m-1);
const double dy=(ymax-ymin)/(n-1);
const double dxxinv=1/(dx*dx);
const double dyyinv=1/(dy*dy);
const double dcent=1/(2*(dxxinv+dyyinv));

// Input function
inline double f(int i,int j) {
  double x=xmin+i*dx,y=ymin+j*dy;
  return abs(x)>0.5||abs(y)>0.5?0:1;
}

// Common output and error routine
void output_and_error(char* filename,double *a,const int sn) {
  // Computes the error if sn%error_every==0
  if(sn%error_every==0) {
```

```
    double z,error=0;int ij;
    for(int j=1;j<n-1;j++) {
      for(int i=1;i<m-1;i++) {
        ij=i+m*j;
        z=f(i,j)-a[ij]*(2*dxxinv+2*dyyinv)
          +dxxinv*(a[ij-1]+a[ij+1])
          +dyyinv*(a[ij-m]+a[ij+m]);
        error+=z*z;
      }
    }
    cout << sn << " " << error*dx*dy << endl;
  }

  // Saves the matrix if sn<=save_iters
  if(sn<=save_iters) {
    int i,j,ij=0,ds=sizeof(float);
    float x,y,data_float;const char *pfloat;
    pfloat=(const char*)&data_float;

    ofstream outfile;
    static char fname[256];
    sprintf(fname,"%s.%d",filename,sn);
    outfile.open(fname,fstream::out
                |fstream::trunc|fstream::binary);

    data_float=m;outfile.write(pfloat,ds);
    for(i=0;i<m;i++) {
      x=xmin+i*dx;
      data_float=x;outfile.write(pfloat,ds);
    }

    for(j=0;j<n;j++) {
      y=ymin+j*dy;
      data_float=y;
      outfile.write(pfloat,ds);
      for(i=0;i<m;i++) {
        data_float=a[ij++];
        outfile.write(pfloat,ds);
      }
    }
    outfile.close();
  }
}
```

References

[1] http://gnuplot.info/.

[2] J. W. Demmel, *Applied numerical linear algebra*, SIAM, 1997.

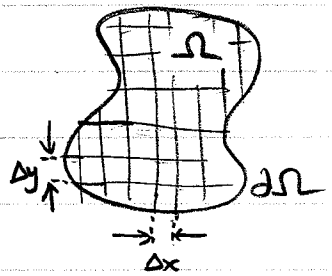
[3] G. H. Golub and C. H. Van Loan, *Matrix computations*, Johns Hopkins University Publishers, 1996.

Error analysis of finite difference methods for the Poisson equation
(reference: Morton & Mayers, Numerical Solution of PDE's)

$$-\nabla\cdot(\sigma\nabla u) = f \quad \text{in } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

$\sigma(x,y)$ diffusion constant (or conductivity if u is the electric potential in a conductor)

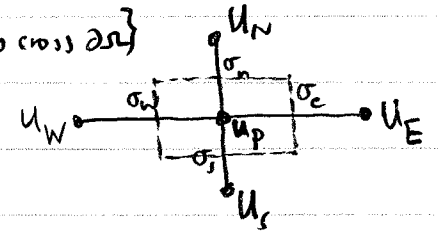
physical interpretation: $-\sigma\nabla u$ is a flux



lay down a grid, use control volumes to derive equations

$\Omega_h = \{\text{gridpoints in } \Omega\}$

$\partial\Omega_h = \{\text{points where gridlines cross } \partial\Omega\}$



interior points:

$$\text{flux out of box} = \iint_{CV}(-\sigma\nabla u)\cdot n\,ds = \iint_{CV}\overbrace{-\nabla\cdot(\sigma\nabla u)}^{f}\,dA = \text{amount created inside}$$

$$-\sigma_n\left(\frac{u_N-u_P}{\Delta y}\right)\Delta x - \sigma_e\left(\frac{u_E-u_P}{\Delta x}\right)\Delta y - \sigma_s\left(\frac{u_S-u_P}{\Delta y}\right)\Delta x - \sigma_w\left(\frac{u_W-u_P}{\Delta x}\right)\Delta y = f_P\,\Delta x\Delta y$$

if $\sigma(x,y)=1$ and $\Delta x = \Delta y$, this equation becomes

$$\frac{1}{\Delta x^2}\left(4u_P - u_N - u_E - u_S - u_W\right) = f_P$$

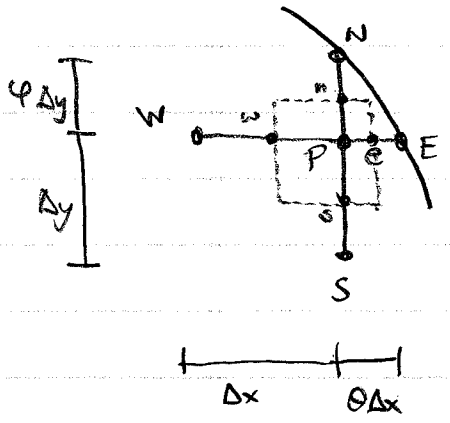
which has the stencil

$$\begin{array}{ccc} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{array}$$

↑ coefficient array of a sparse, localized linear equation

← this would be the stencil if we left Δx^2 on the RHS

points near bdry : if the stencil at P is chipped by the boundary, we shrink the control volume



$0 \le \theta \le 1, \quad 0 \le \varphi \le 1$

$$-\sigma_n\left(\frac{u_N - u_P}{\varphi \Delta y}\right)\left(\frac{1+\theta}{2}\Delta x\right) - \sigma_e\left(\frac{u_E - u_P}{\theta \Delta x}\right)\left(\frac{1+\varphi}{2}\Delta y\right)$$

$$-\sigma_s\left(\frac{u_s - u_P}{\Delta y}\right)\left(\frac{1+\theta}{2}\Delta x\right) - \sigma_w\left(\frac{u_W - u_P}{\Delta x}\right)\left(\frac{1+\varphi}{2}\Delta y\right)$$

$$= f_P\left(\frac{1+\theta}{2}\Delta x\right)\left(\frac{1+\varphi}{2}\Delta y\right)$$

if $\sigma(x,y) = 1$ and $\Delta x = \Delta y$, we get

$$\frac{1}{\Delta x^2}\left\{\left[\frac{1}{\frac{1}{2}(1+\varphi)\varphi} + \frac{1}{\frac{1}{2}(1+\theta)\theta} + \frac{1}{\frac{1}{2}(1+\varphi)} + \frac{1}{\frac{1}{2}(1+\theta)}\right]u_P - \frac{1}{\frac{1}{2}(1+\varphi)}u_s - \frac{1}{\frac{1}{2}(1+\theta)}u_W\right\}$$

$$= \frac{1}{\Delta x^2}\left(\frac{1}{\frac{1}{2}(1+\varphi)\varphi}u_N + \frac{1}{\frac{1}{2}(1+\theta)\theta}u_E\right) + f_P$$

here we've moved the known quantities u_N, u_E ($u_N = g_N$, $u_E = g_E$) to the right hand side

each grid point P contributes a linear equation in the unknowns u_P, u_N, u_E, u_s, u_W

we'll write the linear system as $\boxed{-L_h u = f + g}$ where e.g. (g: bdry data (weighted appropriately))

if P is in the interior and $\Delta x = \Delta y$ then $L_h u_P = \left(-4u_P + u_N + u_E + u_s + u_W\right)\cdot\frac{1}{\Delta x^2}$

↖ Pth component of $L_h u$

Key feature of this construction:

the matrix representing L_h is monotone (and hence diagonally dominant)

$$L_h U_P = \sum_{Q \neq P} C_{PQ} U_Q - C_{PP} U_P$$

$$C_{PQ} \geq 0$$

$$\sum_{Q \neq P} C_{PQ} \leq C_{PP}$$

↑ sum is over nearest neighbors in the domain ($Q = N, E, S, W$)

↑ inequality is strict if P is next to the boundary

main drawback: L_h is not symmetric (curved boundary causes $C_{PQ} \neq C_{QP}$ if P near $\partial\Omega$, Q interior)

(some linear algebra solvers, notably the conjugate gradient method, need symmetric matrices)

(but Jacobi, Gauss-Seidel, SOR, multigrid are OK)

Let's move g back to LHS: include known values U_P, $P \in \partial\Omega$ in U. $L_h: \mathbb{R}^{N+d} \to \mathbb{R}^N$ (points in Ω, on $\partial\Omega$)

truncation error. Let U be the numerical solution $-L_h U = f$

Let u be the exact solution $-\Delta u = f$

we define the truncation error τ as what's left over when you plug the exact solution into the scheme:

$$-L_h U_P = f_P$$

$$-L_h u_P = f_P + \tau_P \quad \leftarrow \text{definition of } \tau$$

subtract: $$-L_h(U_P - u_P) = -\tau_P$$

or $$L_h e_P = \tau_P$$

$e_P = U_P - u_P$ ← error ($e_P = 0$ if $P \in \partial\Omega$)

step 1: bound τ ← consistency step 2: show $L_h e = \tau \Rightarrow \|e\| \le C\|\tau\|$ ← stability

step 1: we assume the exact solution $u(x,y)$ is C^4 on $\bar{\Omega}$

This requires that the boundary is C^4, or is a rectangle. (Corners lead to singularities in the solution of elliptic equations. But 90° corners usually have polynomial singularities (which aren't singular). 90° corners do cause trouble if f is nonzero near the corner, though...)

now Taylor expand: (let's assume $\sigma \equiv 1$ for simplicity.)

case 1: P interior: $L_h u_P = D_x^+ D_x^- u_P + D_y^+ D_y^- u_P$

$$D_x^+ D_x^- u_P = \frac{u(x_P+\Delta x, y_P) - 2u(x_P,y_P) + u(x_P-\Delta x, y_P)}{\Delta x^2}$$

$$= \frac{1}{\Delta x^2}\left\{ \begin{array}{l} u + \Delta x\, u_x + \frac{\Delta x^2}{2} u_{xx} + \frac{\Delta x^3}{6} u_{xxx} + \frac{\Delta x^4}{24} u_{xxxx} + \cdots \\ -2u \\ +u - \Delta x\, u_x + \frac{(-\Delta x)^2}{2} u_{xx} - \frac{\Delta x^3}{6} u_{xxx} + \frac{\Delta x^4}{24} u_{xxxx} + \cdots \end{array} \right\}$$

$$= u_{xx}(x_P,y_P) + \frac{\Delta x^2}{12} u_{xxxx}(x_P,y_P) + O(\Delta x^4)$$

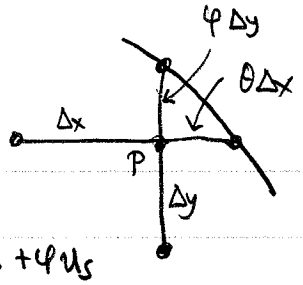
if we used Taylor's theorem with remainder instead, we'd get

$$D_x^+ D_x^- u_P = u_{xx}(x_P,y_P) + R_P, \quad |R_P| \le \frac{M_1 \Delta x^2}{12}, \quad M_1 = \max_{(x,y)\in\Omega} |u_{xxxx}(x,y)|$$

$$D_y^+ D_y^- u_P = u_{yy}(x_P,y_P) + S_P, \quad |S_P| \le \frac{M_2 \Delta y^2}{12}, \quad M_2 = \max |u_{yyyy}(x,y)|$$

$$\tau_P = -L_h u_P - f_P = -R_P - S_P, \qquad \boxed{|\tau_P| \le \frac{1}{12}\left(M_1 \Delta x^2 + M_2 \Delta y^2\right)}$$

case 2: P near bdry. example:



$$L_h u_P = \underbrace{\frac{u_E - (1+\theta)u_P + \theta u_W}{\frac{1}{2}\theta(1+\theta)\Delta x^2}} + \frac{u_N - (1+\varphi)u_P + \varphi u_S}{\frac{1}{2}\varphi(1+\varphi)\Delta y^2}$$

↖ similar

$$\frac{1}{\frac{1}{2}\theta(1+\theta)\Delta x^2}\left(\begin{matrix} u + \theta\Delta x u_x + \frac{(\theta\Delta x)^2}{2}u_{xx} + \frac{(\theta\Delta x)^3}{6}u_{xxx} + \cdots \\ -(1+\theta)u \\ +\theta u - \theta\Delta x u_x + \theta\frac{\Delta x^2}{2}u_{xx} - \theta\frac{\Delta x^3}{6}u_{xxx} + \cdots \end{matrix}\right)$$

$\theta^3 - \theta = \theta(\theta-1)(\theta+1)$

$$= (u_{xx})_P - \frac{1}{3}(1-\theta)\Delta x (u_{xxx})_P + O(\Delta x^2)$$

and using Taylor's theorem with remainder:

$\theta^3 + \theta \le \theta^2 + \theta$

$$\tau_P = -L_h u_P - f_P \quad \text{satisfies} \quad \boxed{|\tau_P| \le \frac{1}{3}(M_3\Delta x + M_4\Delta y)}$$

$M_3 = \max |u_{xxx}|$

$M_4 = \max |u_{yyyy}|$

so we lose an order of accuracy near the boundary due to the broken symmetry of the stencil

the same thing happens when you allow $\sigma(x,y)$ to vary:

$$\tau_P = \begin{cases} O(\Delta x^2 + \Delta y^2) & \text{at interior points} \\ O(\Delta x + \Delta y) & \text{at boundary points} \end{cases}$$

(but the error constants are more complicated... depend on σ too)

step 2: (stability) Now we know τ is small, must show

$$L_h e = \tau \implies e \text{ is small too.}$$

discrete maximum principle: suppose every grid point has a pathway (via stencils) to a Dirichlet boundary node and suppose L_h is monotone:

$$\left(L_h u_p = \sum_{Q \neq P} c_{PQ} u_Q - c_{PP} u_P \quad \leftarrow \text{sum is over bdry nodes too, i.e. } \sum_{Q \in \Omega_h \cup \partial\Omega_h \setminus \{P\}} \cdots \right.$$

$$\left. c_{PQ} \geq 0, \quad c_{PP} \geq \sum_{Q \neq P} c_{PQ} \right)$$

then

$$L_h u_p \geq 0 \quad \forall P \in \Omega_h$$

$$\implies \max_{P \in \Omega_h} u_p \leq \max_{P \in \partial\Omega_h} u_p$$

proof: Let $M = \max_{P \in \Omega_h} u_p$ and choose P so $u_p = M$.

then $$0 \leq L_h u_p = \underbrace{\sum_{Q \neq P} c_{PQ} u_Q}_{\leq} \underbrace{- c_{PP} u_P}_{=}$$

$$\sum_{Q \neq P} c_{PQ} M - c_{PP} M \leq 0$$

If this inequality were strict, we'd have $0 < 0$, a contradiction.

$$\therefore \sum_{Q \neq P} c_{PQ} u_Q = \sum_{Q \neq P} c_{PQ} M$$

$$\therefore \sum_{Q \neq P} c_{PQ} (M - u_Q) = 0$$

in the first case, we repeat the argument at each node on a path to the boundary.

two possibilities:

① $c_{PQ} = 0 \ \forall Q \in \partial\Omega_h$
$\therefore u_Q = M$ whenever $c_{PQ} \neq 0$

② $\exists Q \in \partial\Omega_h$ s.t. $c_{PQ} > 0$,
$\therefore u_Q \geq M$ for some $Q \in \partial\Omega_h$ as claimed.

a similar argument shows that $L_h u_p \le 0 \quad \forall p \in \Omega_h$

$$\Rightarrow \min_{p\in\Omega_h} u_p \ge \min_{p\in\partial\Omega_h} u_p$$

(replace u by $-u$ in proof)

Now we use the maximum principle to prove that $L_h e = \tau \Rightarrow \|e\| \le C\|\tau\|$.

Let Φ be a mesh function satisfying $L_h\Phi_p \ge 1$, $\Phi_p \ge 0 \ \forall p$.

[in the case that $\sigma \equiv 1$, $\Phi(x,y) = \frac{x^2+y^2}{4}$ works (our finite difference formulas are exact for quadratic functions when σ = const.)]

Let $T = \max_{p\in\Omega_h} |\tau_p|$, $C = \max_{p\in\partial\Omega_h} \Phi_p$.

Then $|e_p| \le CT \quad \forall p \in \Omega_h$.

proof: $L_h(T\Phi + e)_p \ge T + \tau_p \ge 0$ ($L_h e_p = \tau_p$)

$$\therefore \max_{p\in\Omega_h}(e_p) \le \max_{p\in\Omega_h}(T\Phi_p + e_p) \le \max_{p\in\partial\Omega_h}(T\Phi_p + e_p)$$

$\Phi_p \ge 0$; maximum principle ; 0 when $p \in \partial\Omega_h$

$$\therefore \max_p(e_p) \le CT$$

a similar argument shows that $\max_p(-e_p) \le CT$

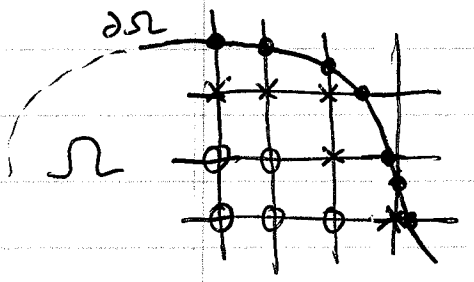
$$\therefore \max_p |e_p| \le CT$$

Unfortunately T is only $O(\Delta x + \Delta y)$ due to large errors near bdry.

Next time we'll see how to sharpen this argument to prove global 2nd order accuracy in spite of first order truncation errors near the boundary.

Today we'll finish talking about finite difference methods for the Poisson and heat equations.

notation: Let's give names to the various types of grid points



J	set of all grid points	$\bullet, \times, \circ$
J_1	interior pts far from bdry	$\circ$
J_2	interior pts near bdry	$\times$
J_3	boundary points	$\bullet$

$$J = \overbrace{J_1 \cup J_2}^{J_{12}} \cup J_3 \qquad N_1 = \#J_1, \quad N_{12} = \#J_{12}, \text{ etc}$$

our finite difference operator L_h is a linear mapping

$$L_h : R^J \to R^{J_{12}}$$

where a grid function $u \in R^J$ assigns a number u_p to each $p \in J$
and a " " $f \in R^{J_{12}}$ " " " f_p " " $p \in J_{12}$

we can represent L_h as an $N_{12} \times N$ matrix by enumerating the points in J_{12} (to give the rows of L_h) and the points in J (to give the columns). It's probably best to enumerate the columns corresponding to interior points first (putting points in J_3 at the end) so that the row and column indices of a point $p \in J_{12}$ are the same.

with this enumeration, the linear system

$$-L_h u = f$$

has the form $(A, B)\begin{pmatrix} u \\ g \end{pmatrix} = (f)$ or $Au = f - Bg$

(columns corresponding to bounding nodes $Q \in J_3$ when the solution $u = g$ is known)

so the linear system you actually solve is $Au = f - Bg$

where A is $N_{12} \times N_{12}$, but in the error analysis we'll work directly with L_h.

Since L_h is a linear operator, it has the form

$$L_h u_P = \sum_{Q \in J \setminus \{P\}} C_{PQ} u_Q - C_{PP} u_P \qquad (P \in J_{12})$$

def: $\text{neigh}(P) = \{Q \in J \setminus \{P\} : C_{PQ} \neq 0\}$

in our construction, $\text{neigh}(P) = \{$the 4 nearest neighbors of $P\}$ (we labeled them N, E, S, W last time). But the analysis below can handle larger stencils than this as long as they're monotone.

def: a path from $P \in J_{12}$ to $Q \in J$ is a sequence $P_1, \ldots, P_s$ such that $P_1 \in \text{neigh}(P)$, $P_2 \in \text{neigh}(P_1), \ldots, P_s \in \text{neigh}(P_{s-1})$, $Q \in \text{neigh}(P_s)$.

def: L_h is monotone if $C_{PQ} \geq 0 \quad \forall P \in J_{12}, Q \in J$

and $$C_{PP} \geq \sum_{Q \in \text{neigh}(P)} C_{PQ}$$

by construction, our scheme is monotone and this is actually an equality.
(it becomes an inequality if $P \in J_2$ and we omit $Q \in J_3$ from the sum, i.e. if we look only at the rows of A and not of (A, B) above)

maximum principle: if L_h is monotone and every $P \in J_{12}$ has a path to the boundary, then

$$\left(L_h u_P \geq 0 \;\; \forall P \in J_{12}\right) \Rightarrow \left(\max_{P \in J_{12}} u_P \leq \max_{P \in J_3} u_P \right)$$

proof: suppose the largest value occurs in the interior, i.e.

$$\exists\, P \in J_{12} \text{ s.t. } u_Q \leq u_P \;\; \forall Q \in J$$

then

$$c_{PP} u_P \leq \sum_{Q \in \text{neigh}(P)} c_{PQ} u_Q \leq \left(\sum_{Q \in \text{neigh}(P)} c_{PQ}\right) u_P \leq c_{PP} u_P$$

- $c_{PP} u_P \leq \sum_{Q \in \text{neigh}(P)} c_{PQ} u_Q$: $L_h u_P \geq 0$
- $\sum_{Q \in \text{neigh}(P)} c_{PQ} u_Q \leq \left(\sum_{Q \in \text{neigh}(P)} c_{PQ}\right) u_P$: strict unless $u_Q = u_P \;\forall Q \in \text{neigh}(P)$
- $\left(\sum_{Q \in \text{neigh}(P)} c_{PQ}\right) u_P \leq c_{PP} u_P$: monotonicity

conclusion: $u_Q = u_P \;\forall Q \in \text{neigh}(P)$. repeat this argument along a path to the boundary to get $u_Q = u_P$, $Q \in J_3$.

→ so interior maxima can only occur for functions that are constant on connected components. The values at the boundary of a constant function also serve as maxima.

Last time we saw how to use the maximum principle to show that

$$L_h e = \tau \;\Rightarrow\; \|e\|_\infty \leq C \|\tau\|_\infty$$

where $C = \max_{P \in J_3} \Phi_P$ for any grid function satisfying $\Phi_P \geq 0 \;\; P \in J$, $L_h \Phi_P \geq 1 \;\; P \in J_{12}$

Today will sharpen this result to deal with the loss of accuracy in τ_p when $p \in J_2$.

Theorem: Suppose $L_h e = \tau$ and we can find a grid function Φ satisfying

(1) $\quad \Phi_p \geq 0 \qquad p \in J$

(2) $\begin{cases} L_h \Phi_p \geq C_1 > 0 & p \in J_1 \\ L_h \Phi_p \geq C_2 > 0 & p \in J_2 \end{cases}$

(3) $\begin{cases} |\tau_p| \leq T_1 & p \in J_1 \\ |\tau_p| \leq T_2 & p \in J_2 \end{cases}$

then

$$|e_p| \leq \left(\max_{p \in J_3} \Phi_p \right) \max\left(\frac{T_1}{C_1}, \frac{T_2}{C_2} \right), \quad p \in J.$$

proof: Let $K = \max\left(\frac{T_1}{C_1}, \frac{T_2}{C_2} \right)$ and observe that

$$L_h(K\Phi + e)_p = K L_h \Phi_p + \tau_p \geq \begin{Bmatrix} KC_1 - T_1 & p \in J_1 \\ KC_2 - T_2 & p \in J_2 \end{Bmatrix} \geq 0$$

$$\therefore \; e_p \leq K\Phi_p + e_p \leq \max_{p \in J_3} \underbrace{\left(K\Phi_p + e_p \right)}_{\uparrow \; 0 \text{ when } p \in J_3} = \left(\max_{p \in J_3} \Phi_p \right) K$$

similarly, $L_h(K\Phi - e)_p = K L_h \Phi_p - \tau_p \geq \begin{Bmatrix} KC_1 - T_1 & p \in J_1 \\ KC_2 - T_2 & p \in J_2 \end{Bmatrix} \geq 0$

so $\quad -e_p \leq K\Phi_p - e_p \leq \max_{p \in J_3} \left(K\Phi_p - e_p \right) = \left(\max_{p \in J_3} \Phi_p \right) K$

$\therefore \; |e_p| \leq \cdots$ as claimed.

In our model problem $\left(\begin{array}{l} -\Delta u = f \text{ in } \Omega \\ u = g \text{ on } \partial\Omega \end{array}\right)$ we have

$$T_1 = \frac{1}{12}\left(M_1 \Delta x^2 + M_2 \Delta y^2\right) \qquad M_1 = \max_{(x,y)\in\Omega} |u_{xxxx}(x,y)| \,, \; M_2 = \max |u_{yyyy}|$$

$$T_2 = \frac{1}{3}\left(M_3 \Delta x + M_4 \Delta y\right) \qquad M_3 = \max |u_{xxx}| \,, \; M_4 = \max |u_{yyy}|$$

Let's assume $\Delta x = \Delta y$ and define $K_1 = \frac{1}{12}(M_1 + M_2)$, $K_2 = \frac{1}{3}(M_3 + M_4)$ so that

$$T_1 \le K_1 \Delta x^2 \,, \quad T_2 \le K_2 \Delta x$$

So we just need to construct a Φ s.t. $L_h \Phi_p \ge \frac{1}{\Delta x}$ for $p \in J_2$

Let's try adding a constant to the boundary points:

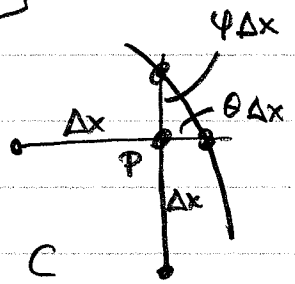
$$\Phi_p = \begin{cases} \dfrac{x_p^2 + y_p^2}{4} & p \in J_{12} \\ \frac{1}{4}\left(x_p^2 + y_p^2\right) + C & p \in J_3 \end{cases}$$

$C_1 = 1$ in theorem

Then $L_h \Phi_p = \left(D_x^+ D_x^- + D_y^+ D_y^-\right)\left(\frac{x^2+y^2}{4}\right)_p = \frac{1}{2} + \frac{1}{2} = 1 \qquad p \in J_1$

[reason: $D_x^+ D_x^-(x^2) = \frac{(x+\Delta x)^2 - 2x^2 + (x-\Delta x)^2}{\Delta x^2} = 2$]

and if $p \in J_2$ with e.g. the geometry



then

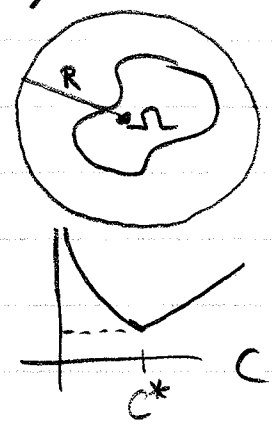
$$L_h \Phi_p = 1 + \frac{C}{\frac{1}{2}\theta(1+\theta)\Delta x^2} + \frac{C}{\frac{1}{2}\varphi(1+\varphi)\Delta x^2} \ge \frac{C}{\Delta x^2}$$

↑ in all cases (for any geometry in which stencil is clipped by bdry)

So $C_2 = \dfrac{C}{\Delta x^2}$ in the theorem.

So we learn that

$$|e_p| \le \underbrace{\left(\max_{p \in J_3} \Phi_p\right)}_{\left(\frac{R^2}{4}+C\right)} \max\left(\frac{K_1 \Delta x^2}{1}, \frac{K_2 \Delta x}{C/\Delta x^2}\right)$$



the RHS in minimized (varying C) when the two terms in $\max(\cdot,\cdot)$ are equal.

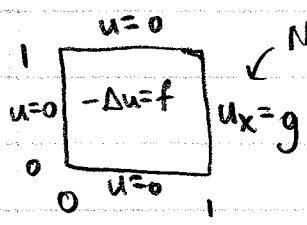
$$C^* = \frac{K_2 \Delta x^3}{K_1 \Delta x^2} = \frac{K_2 \Delta x}{K_1}$$

$$|e_p| \le \left(\frac{R^2}{4} + \frac{K_2}{K_1}\Delta x\right) K_1 \Delta x^2 = \frac{K_1 R^2}{4}\Delta x^2 + K_2 \Delta x^3$$

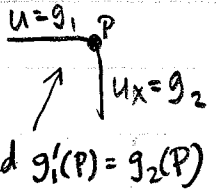
So even though we lost an order of accuracy in the truncation error near the boundary, the global error is still $O(\Delta x^2)$ with an error constant governed by the interior (J_1) equations.

=

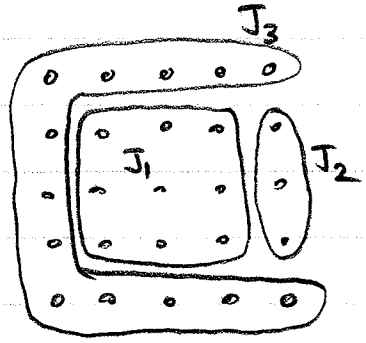
As another example of this proof technique, consider the problem



Neumann b/c's on one side. (Be careful of singularities at points where b/c changes type. 90° is OK if f=0 near the corner and the b/c's are compatible

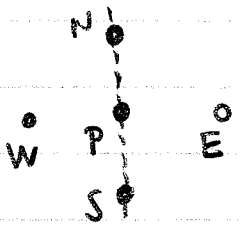


partitioning of the nodes:



The points $P \in J_2$ are treated as interior nodes since we don't know their values.

use ghost points to derive equations on right boundary



$$U_E = U_W + 2\Delta x\, g_P$$

$$\underbrace{-\frac{U_N + U_S + 2U_E - 4U_P}{\Delta x^2}}_{L_h U_P} = \frac{2g_P}{\Delta x} + f_P = \tilde{f}_P$$

stencil: $\frac{1}{2}\ \frac{1}{-4}\ \frac{1}{1}$

linear system: $-L_h U = \tilde{f}$ (L_h is still monotone)

truncation error: $|\tau_P| \le K_1 \Delta x^2$ $\quad P \in J_1$

$|\tau_P| \le K_2 \Delta x$ $\quad P \in J_2$

$K_1 = \frac{1}{12}(M_1 + M_2)$, $K_2 = \frac{1}{3}M_3 + \frac{1}{12}M_2\Delta x$

$M_1 = \max|u_{xxxx}|$
$M_2 = \max|u_{yyyy}|$
$M_3 = \max|u_{xxx}|$

grid function: $\Phi_P = (x_P - 2)^2 + y_P^2$ (needed this to be positive)

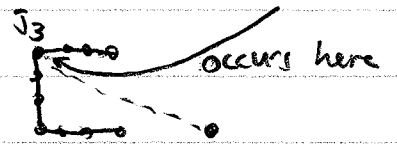
$$\underbrace{-\frac{2}{\Delta x}\left(\frac{\partial \Phi}{\partial x}\right)_P}_{\frac{\Phi_W - \Phi_E}{\Delta x^2}} \le \frac{4}{\Delta x}$$

$P \in J_1$: $L_h \Phi_P = 4$

$P \in J_2$: $L_h \Phi_P = \frac{\Phi_N + \Phi_S + 2\Phi_W - 4\Phi_P}{\Delta x^2} = 4 + \frac{\Phi_W - \Phi_E}{\Delta x^2}$

So now we apply the theorem and learn that

$$|e_p| \le \left(\max_{p \in J_3} \Phi_p \right) \max\left(\frac{K_1 \Delta x^2}{4} , \frac{K_2 \Delta x}{4/\Delta x} \right)$$



$$\max_{p \in J_3} \Phi_p = 2^2 + 1^2 = 5$$

$$|e_p| \le \frac{5}{4} \max(K_1, K_2) \Delta x^2 = O(\Delta x^2)$$

again we find that losing an order of accuracy in the truncation error (at the boundary) does not destroy 2nd order accuracy of the method.

final comments:

(1) it's not hard to produce higher order stencils for the Laplacian, but any time the boundary conditions don't screw things up, you would have been better off using a spectral method. For curved boundaries, higher order methods are easier to achieve using finite elements.

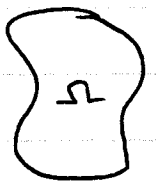
(2) we can convert any solver for the Poisson equation into an implicit method for the heat equation. $u_t = \Delta u + f$

example: Backward Euler: $u^{n+1} = u^n + \Delta t (L_h u^{n+1} + f^{n+1})$

system to solve: $(I - \Delta t L_h) u^{n+1} = u^n + f^{n+1} \Delta t$

a discrete maximum principle holds for this operator, too, which implies that the inverse is bounded in the infinity norm.

This week we're going to study boundary integral methods for the Laplace equation:



$$\Delta u = 0 \quad \text{in } \Omega$$

$$\begin{cases} u = g & \text{on } \partial\Omega \leftarrow \text{Dirichlet b/c's. (voltage or temperature specified on boundary)} \\ \text{or} \\ \dfrac{\partial u}{\partial n} = g & \text{on } \partial\Omega \leftarrow \text{Neumann b/c's (current or flux b.c.'s)} \end{cases}$$

idea: take advantage of the fact that the Laplace equation is linear and use the superposition principle to represent the solution.

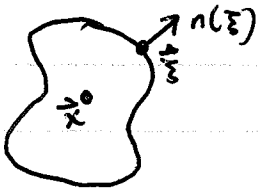
if we had access to the Green's function $G(\vec{x},\vec{\xi})$ satisfying

$$-\Delta G(\vec{x},\vec{\xi}) = \delta(\vec{x}-\vec{\xi}) \qquad \vec{x}\in\Omega \qquad \leftarrow \vec{\xi} = (\xi,\eta) \text{ fixed}$$

$$G(\vec{x},\vec{\xi}) = 0 \qquad \vec{x}\in\partial\Omega \qquad \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

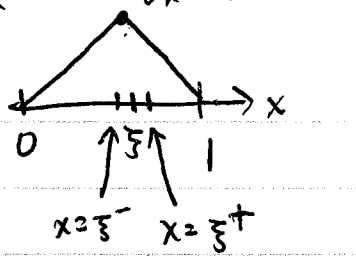
we could write down the solution of $\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$ explicitly:

$$u(\vec{x}) = \int_\Omega G(\vec{x},\vec{\xi})\, f(\vec{\xi})\, d\vec{\xi} - \int_{\partial\Omega} \underbrace{\frac{\partial G}{\partial n_\xi}(\vec{x},\vec{\xi})}_{\text{normal derivative with respect to } \xi,\ \text{i.e. } \nabla_\xi G(\vec{x},\vec{\xi})\cdot n(\vec{\xi})} g(\vec{\xi})\, ds_\xi$$



f & g serve as weights to use when summing up elementary solutions $G(\vec{x},\vec{\xi})$ with different values of $\vec{\xi}$.

$$\frac{\partial G}{\partial x}(\xi^+,\xi) - \frac{\partial G}{\partial x}(\xi^-,\xi) = -1$$



in 1-d, this is actually very useful because we know the Green's function:

$$G(x,\xi) = \begin{cases} (1-\xi)x & 0 \le x \le \xi \\ (1-x)\xi & \xi \le x \le 1 \end{cases}$$

$x \to -x$, $1-x$

$$u(x) = \int_0^1 G(x,\xi) f(\xi)\, d\xi \underbrace{- \left[\frac{\partial G}{\partial \xi}(x,1)\, g(1) - \frac{\partial G}{\partial \xi}(x,0)\, g(0) \right]}$$

$$+ x\, g(1) + (1-x)\, g(0)$$

but in 2-d and 3-d, finding the Green's function is at least as hard as solving the original problem in most cases.
(exceptions: $\Omega = \mathbb{R}^n$ free space problem
$\Omega = \{(x,y) : y > 0\}$ or $\{(x,y,z) : z > 0\}$ half space problem
Ω = unit ball ← can be conformally mapped to half space.)

So instead of G, we'll use the Newtonian potential N, which satisfies the equation but not the b.c's:

$$-\Delta N = \delta(\vec{x} - \vec{\xi})$$ ← N is the free space Green's function.

N is spherically symmetric (but doesn't satisfy $N(\vec{x},\vec{\xi}) = 0$ for $\vec{x} \in \partial\Omega$)

$$N(\vec{x},\vec{\xi}) = \begin{cases} -\frac{1}{2\pi} \log(|\vec{x} - \vec{\xi}|) & 2d \\ \frac{1}{4\pi |\vec{x} - \vec{\xi}|} & 3d \end{cases} \qquad |\vec{x} - \vec{\xi}| = \sqrt{(x-\xi)^2 + (y-\eta)^2 + (z-\zeta)^2}$$

our new representation is

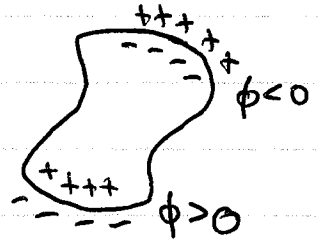
$$u(\vec{x}) = \int_\Omega N(\vec{x},\vec{\xi}) f(\vec{\xi})\, d\vec{\xi} - \int_{\partial\Omega} \frac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi})\, \phi(\vec{\xi})\, ds_\xi$$

the price we pay for using N instead of G is that ϕ is now unknown (whereas before the given boundary values were used).

When $f=0$ (we'll assume this from now on (when $f\neq 0$, better to use finite elements...)), u is known as the double layer potential with moment ϕ or dipole density

The study of this representation is known as potential theory.

physically, we can think of ϕ as a surface distribution of dipoles



or just think of this as another instance of the superposition principle

($\frac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi})$ with $\vec{\xi}\in\partial\Omega$ satisfies $\Delta u = 0$ inside Ω)

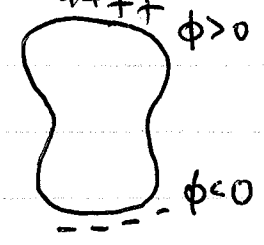
similarly, for the Neumann problem $\begin{cases} \Delta u = 0 & \text{in } \Omega \\ \frac{\partial u}{\partial n} = g & \text{on } \partial\Omega \end{cases}$

we look for solutions of the form

$$u(\vec{x}) = \int_{\partial\Omega} N(\vec{x},\vec{\xi})\, \underbrace{\phi(\vec{\xi})}_{\text{moment or charge density}}\, ds_\xi$$

↑ single layer potential

physical interp: surface distribution of charges



our goal now is to reduce the problem to an integral equation for ϕ in terms of g that we can solve numerically.

<u>case 1:</u> 2d Dirichlet problem

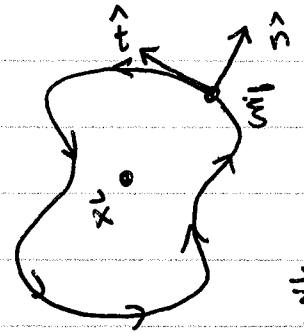
$$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \vec{\xi} = \begin{pmatrix} \xi \\ \eta \end{pmatrix}, \quad r = |\vec{x} - \vec{\xi}| = \sqrt{(x-\xi)^2 + (y-\eta)^2}$$

$$N = -\frac{1}{2\pi} \log r$$

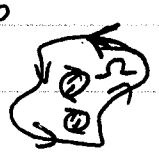
$$-\nabla_\xi N = \frac{1}{2\pi} \cdot \frac{1}{r} \cdot \frac{2(\xi - x, \eta - y)}{2\sqrt{(\xi-x)^2 + (\eta-y)^2}} = \frac{1}{2\pi} \frac{\vec{\xi} - \vec{x}}{r^2}$$

direction: $\frac{\vec{\xi} - \vec{x}}{r}$ magnitude: $\frac{1}{2\pi r}$

we want to compute $-\frac{\partial N}{\partial n_\xi} = -\nabla_\xi N \cdot \vec{n}_\xi$ ← unit normal at $\vec{\xi}$



suppose the curve is parametrized so Ω lies to the left of $\Gamma = \partial\Omega$



$\vec{\xi}(t)$ $0 \le t \le a$ parametrizes Γ

unit tangent vector: $\hat{t} = \frac{d\vec{\xi}}{ds} = \frac{d\vec{\xi}}{dt} \cdot \frac{dt}{ds} = \frac{d\vec{\xi}}{dt} \cdot \left(\frac{ds}{dt}\right)^{-1}$

arclength →

$$\frac{ds}{dt} = \sqrt{\dot{\xi}^2 + \dot{\eta}^2}$$

outward normal: $\hat{n} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} t_2 \\ -t_1 \end{pmatrix} = \frac{1}{\sqrt{\dot{\xi}^2 + \dot{\eta}^2}} \begin{pmatrix} \dot{\eta} \\ -\dot{\xi} \end{pmatrix}$

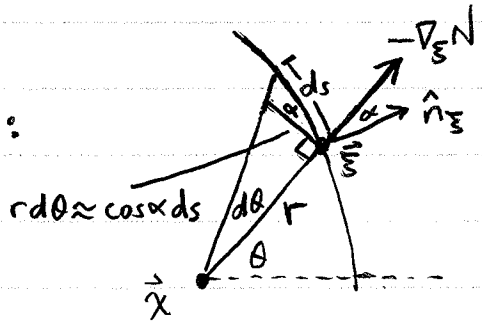
clockwise rotation by 90°

so $$-\frac{\partial N}{\partial n_\xi} = \frac{1}{2\pi}\frac{(\xi-x)\dot\eta - (\eta-y)\dot\xi}{(\xi-x)^2+(\eta-y)^2}\cdot\frac{1}{\sqrt{\dot\xi^2+\dot\eta^2}}$$

and $$u(\vec x) = \int_\Gamma -\frac{\partial N}{\partial n_\xi}\phi\, ds = \frac{1}{2\pi}\int_0^a \frac{(\xi(t)-x)\dot\eta(t) - (\eta(t)-y)\dot\xi(t)}{(\xi(t)-x)^2+(\eta(t)-y)^2}\phi(\vec\xi(t))\,dt$$

($ds = \frac{ds}{dt}dt$)

$-\frac{\partial N}{\partial n_\xi}$ has a geometric interpretation:



$$\frac{d\theta}{ds} \approx \frac{\cos\alpha}{r}$$

$$-\nabla_\xi N\cdot\vec n_\xi = |\nabla_\xi N|\cos\alpha = \frac{\cos\alpha}{2\pi r}$$

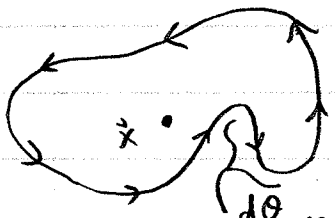
$$-\frac{\partial N}{\partial n_\xi} = \frac{1}{2\pi}\frac{d\theta}{ds}$$ ← $\vec x$ fixed, $\vec\xi$ varying along Γ

θ = angle from horizontal to $\vec\xi - \vec x$

so $$u(\vec x) = \frac{1}{2\pi}\int_\Gamma \frac{d\theta}{ds}\phi(\xi)\,ds = \frac{1}{2\pi}\int_0^{2\pi}\phi(\xi(\theta))\,d\theta = \frac{1}{2\pi}\int_0^a \frac{d\theta}{dt}\phi(\vec\xi(t))\,dt$$

angle parametrization ($d\theta$ = signed measure, can be negative)

arbitrary parametrization



$\frac{d\theta}{ds}$ is negative

note:
$$\tan\theta = \frac{\eta-y}{\xi-x} \Rightarrow \sec^2\theta\,\dot\theta = \frac{(\xi-x)\dot\eta-(\eta-y)\dot\xi}{(\xi-x)^2}$$
$$\Rightarrow \dot\theta = \frac{\cdots}{\cdots}\cos^2\theta = \frac{\cdots}{\cdots}\frac{(\xi-x)^2}{(\xi-x)^2+(\eta-y)^2} = \frac{(\xi-x)\dot\eta-(\eta-y)\dot\xi}{(\xi-x)^2+(\eta-y)^2}$$

same formula as before: $u(\vec x) = \frac{1}{2\pi}\int_0^a (\ \nwarrow\ )\,\phi(\vec\xi(t))\,dt$

Suppose $\vec{x} \in \Gamma$. what happens to the integrand when $\vec{\xi}$ approaches $\vec{x}$ along Γ? (say $\vec{\xi}(t_0) = \vec{x}$)

use l'Hopital's rule: (or do a Taylor expansion... same result)

$$\dot{\theta}(t_0) = \lim_{t \to t_0} \frac{(\xi(t)-x)\dot{\eta}(t) - (\eta(t)-y)\dot{\xi}(t)}{(\xi(t)-x)^2 + (\eta(t)-y)^2}$$

$$= \lim_{t \to t_0} \frac{(\xi-x)\ddot{\eta} - (\eta-y)\ddot{\xi} + \dot{\xi}\dot{\eta} - \dot{\eta}\dot{\xi}}{2[(\xi-x)\dot{\xi} + (\eta-y)\dot{\eta}]}$$

$$= \lim_{t \to t_0} \frac{\dot{\xi}\ddot{\eta} - \dot{\eta}\ddot{\xi} + \overset{0}{(\xi-x)}\dddot{\eta} - \overset{0}{(\eta-y)}\dddot{\xi}}{2[\dot{\xi}^2 + \dot{\eta}^2] + 2\overset{0}{(\xi-x)}\ddot{\xi} + 2\overset{0}{(\eta-y)}\ddot{\eta}}$$

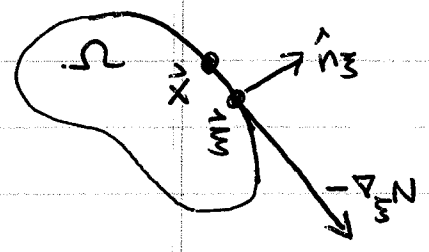
$$= \frac{1}{2}\kappa(\vec{x})\frac{ds}{dt} \qquad \kappa(x) = \frac{\dot{\xi}\ddot{\eta} - \dot{\eta}\ddot{\xi}}{(\dot{\xi}^2+\dot{\eta}^2)^{3/2}}, \quad \frac{ds}{dt} = \sqrt{\dot{\xi}^2+\dot{\eta}^2}$$

$\therefore \frac{d\theta}{ds}(t_0) = \frac{1}{2}\kappa(\vec{x})$ curvature Ω ⌒ $\kappa > 0$ Ω ⌒ $\kappa < 0$

result: the function $K : \Gamma \times \Gamma \to \mathbb{R}$ given by

$$K(\vec{x},\vec{\xi}) = \begin{cases} -\frac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi}) = \frac{1}{2\pi}\frac{d\theta}{ds}(\vec{x},\vec{\xi}) & \vec{x} \in \Gamma,\ \vec{\xi} \in \Gamma,\ \vec{x} \neq \vec{\xi} \\ \frac{1}{4\pi}\kappa(\vec{x}) & \vec{x} = \vec{\xi} \in \Gamma \end{cases}$$

is continuous in spite of the fact that $|\nabla_\xi N| = \frac{1}{2\pi|\vec{\xi}-\vec{x}|}$ blows up.

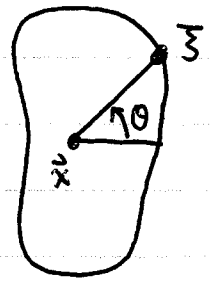


Reason: $\nabla_\xi N$ and $\hat{n}_\xi$ are close to orthogonal

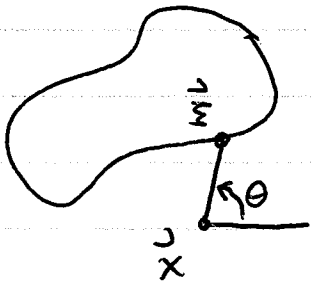
Next we check what happens if $\vec{x}$ is close to Γ but not on Γ.

Note that if $\phi \equiv 1$ then

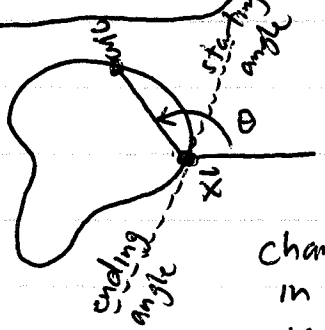
$$u(\vec{x}) = \frac{1}{2\pi}\int_0^a \frac{d\theta}{dt}\,dt = \begin{cases} 1 & x \in \Omega \\ 1/2 & x \in \Gamma \\ 0 & \text{o.w.} \end{cases}$$



change in angle is 2π when you go around once.



change in angle is Zero when you go around once.

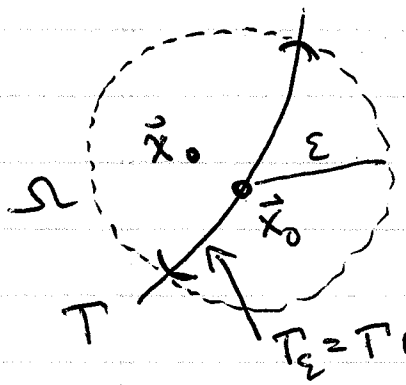


change in angle is π

Also, if $\phi(x_0) = 0$, then $u(\vec{x}_0^-) = u(\vec{x}_0)$

proof: $\underbrace{u(\vec{x}_0^-)}_{\lim_{\substack{\vec{x}\to\vec{x}_0 \\ \vec{x}\in\Omega}} u(\vec{x})} = \underbrace{\frac{1}{2\pi}\int_{\Gamma_\varepsilon} \frac{d\theta}{ds}(\vec{x}_0^-, \vec{\xi})\,\phi(\vec{\xi})\,ds}_{\circledast \downarrow 0 \text{ as } \varepsilon\to 0} + \underbrace{\frac{1}{2\pi}\int_{\Gamma\setminus\Gamma_\varepsilon} \frac{d\theta}{ds}(\vec{x}_0^-, \vec{\xi})\,\phi(\vec{\xi})\,ds}_{\downarrow u(\vec{x}_0) \text{ as } \varepsilon\to 0}$

(take limit as $\vec{x}\to\vec{x}_0$ first, then let $\varepsilon\to 0$)



$$|\circledast| \le \underbrace{\left(\max_{\vec{\xi}\in\Gamma_\varepsilon} |\phi(\vec{\xi})|\right)}_{\substack{\text{Converges to zero as } \varepsilon\to 0 \\ \text{since } \phi \text{ is cont. at } \xi = x_0}} \underbrace{\int_{\Gamma_\varepsilon} \frac{1}{2\pi}\left|\frac{d\theta}{ds}(\vec{x}_0^-, \vec{\xi})\right| ds}_{\le \frac{1}{2} \text{ when } \varepsilon \text{ is small}}$$

Then for general ϕ we have

$$u(\vec{x}_0^-) = \frac{1}{2\pi}\int_\Gamma \frac{d\theta}{ds}(\vec{x}_0^-, \vec{\xi})\Big[\overbrace{\phi(\vec{x}_0)}^{\text{constant function}} + \overbrace{\phi(\vec{\xi})-\phi(\vec{x}_0)}^{\to 0 \text{ as } \vec{\xi}\to\vec{x}_0}\Big]\,ds$$

$$= \phi(\vec{x}_0) + \frac{1}{2\pi}\int_\Gamma \frac{d\theta}{ds}(\vec{x}_0, \vec{\xi})\big[\phi(\vec{\xi}) - \underbrace{\phi(\vec{x}_0)}_{\text{const fcn.}}\big]\,ds$$

but now the integral has $\vec{x}_0 \in \Gamma$ (instead of $\vec{x}_0^- \in \Omega$, but very close to $\vec{x}_0$)

$$= \frac{1}{2}\phi(\vec{x}_0) + \underbrace{\frac{1}{2\pi}\int_\Gamma \frac{d\theta}{ds}(\vec{x}_0, \vec{\xi})\,\phi(\vec{\xi})\,ds}_{u(\vec{x}_0)}$$

similarly, if you approach from the outside:

$$u(\vec{x}_0^+) = -\frac{1}{2}\phi(\vec{x}_0) + u(\vec{x}_0)$$

so $u(\vec{x})$ jumps by $-\phi(\vec{x}_0)$ when $\vec{x}$ crosses Γ at $\vec{x}_0$ from inside to outside.

the boundary condition we want to impose is $u(\vec{x}_0^-) = g(\vec{x}_0)$, so $\phi(\vec{\xi})$ should satisfy

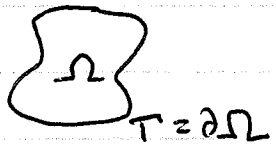
$$\frac{1}{2}\phi(\vec{x}_0) + \int_\Gamma K(\vec{x}_0, \vec{\xi})\,\phi(\vec{\xi})\,ds = g(\vec{x}_0) \qquad (\vec{x}_0 \in \Gamma)$$

where $K : \Gamma \times \Gamma \to \mathbb{R}$ was defined previously. This is a second kind Fredholm integral equation. (A very nice property). We'll see how to solve it numerically with spectral accuracy next time; and also talk about the Neumann problem.

Boundary integral (and boundary element) methods

Last time we saw that the solution of the Dirichlet problem

$$\Delta u = 0 \text{ in } \Omega, \qquad u = g \text{ on } \Gamma$$



can be represented as a double layer potential

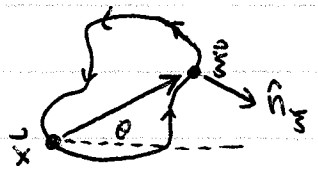
$$u(\vec{x}) = \int_\Gamma -\frac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi})\,\underbrace{\phi(\vec{\xi})}_{\text{dipole density (assumed to be continuous on } \Gamma)}\,ds \qquad \leftarrow N(\vec{x},\vec{\xi}) = \frac{-1}{2\pi}\log|\vec{x}-\vec{\xi}|$$

as long as ϕ satisfies the integral equation

$$\frac{1}{2}\phi(\vec{x}) + \int_\Gamma K(\vec{x},\vec{\xi})\,\phi(\vec{\xi})\,ds = g(\vec{x}) \qquad (\vec{x}\in\Gamma)$$

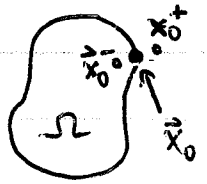
where $K:\Gamma\times\Gamma\to\mathbb{R}$ is the continuous function

$$K(\vec{x},\vec{\xi}) = \begin{cases} -\dfrac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi}) = \dfrac{1}{2\pi}\dfrac{d\theta}{ds}(\vec{x},\vec{\xi}) \\ \kappa(\vec{x})/4\pi \quad \leftarrow \text{if } \vec{x}=\vec{\xi} \end{cases}$$



This is because the double layer potential satisfies the jump conditions

$$u(\vec{x}_0^{\pm}) = \mp\frac{1}{2}\phi(\vec{x}_0) + u(\vec{x}_0) \qquad (\vec{x}_0\in\Gamma)$$

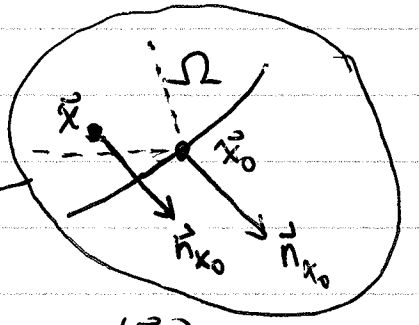


and we want $u(\vec{x}_0^-) = g(\vec{x}_0)$.

For the Neumann problem $\begin{cases} \Delta u = 0 & \text{in } \Omega \\ \frac{\partial u}{\partial n} = g & \text{on } \Gamma \end{cases}$

we will use a single layer potential (see handout for contour plots)

$$u(\vec{x}) = \int_\Gamma N(\vec{x},\vec{\xi})\,\phi(\vec{\xi})\,ds$$



and we want to enforce

$$\frac{\partial u}{\partial n_x}(\vec{x}_0^{-}) = \lim_{\substack{\vec{x}\to\vec{x}_0 \\ \vec{x}\in\Omega}} \underbrace{\nabla_x u(x)\cdot\hat{n}_{x_0}}_{\int_\Gamma \left(\nabla_x N(\vec{x},\vec{\xi})\cdot\vec{n}_{x_0}\right)\phi(\vec{\xi})\,ds} = g(\vec{x}_0)$$

approach from inside a cone (stay away from bdry)

with some effort (see e.g. Garabedian — it's much harder than the double layer case), one may show that

$$\frac{\partial u}{\partial n_x}(\vec{x}_0^{\pm}) = \mp\frac{1}{2}\phi(\vec{x}_0) + \frac{\partial u}{\partial n_x}(\vec{x}_0)$$

where $\frac{\partial u}{\partial n_x}(\vec{x}_0)$ is not the derivative of anything, but instead is the integral

$$\frac{\partial u}{\partial n_x}(\vec{x}_0) \overset{\text{def}}{=} \int_\Gamma \frac{\partial N}{\partial n_x}(\vec{x}_0,\vec{\xi})\,\phi(\vec{\xi})\,ds \qquad (\vec{x}_0\in\Gamma)$$

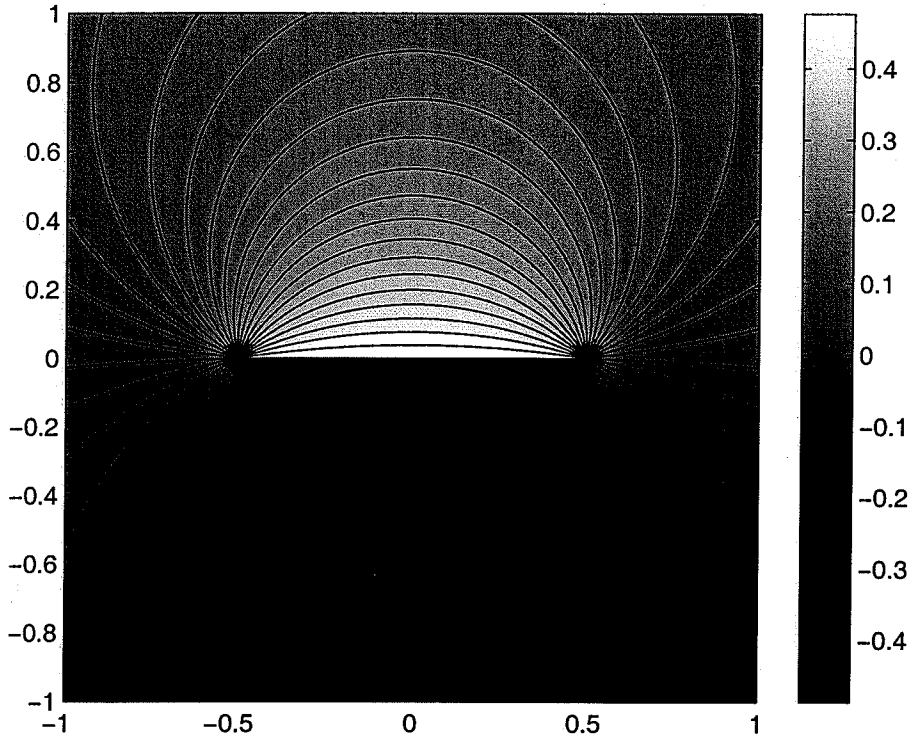
Note that if $\vec{x},\vec{\xi}\in\Gamma$, $\frac{\partial N}{\partial n_x} = \nabla_x N(\vec{x},\vec{\xi})\cdot\hat{n}_x$

$= \nabla_x N(\vec{\xi},\vec{x})\cdot\hat{n}_x$ ← $N(\vec{x},\vec{\xi}) = N(\vec{\xi},\vec{x})$

recall

$K(\vec{x},\vec{\xi}) = -\frac{\partial N}{\partial n_\xi}(\vec{x},\vec{\xi})$ → $= -K(\vec{\xi},\vec{x})$

double layer potential with density $\phi = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & |x| > 1/2 \end{cases}$

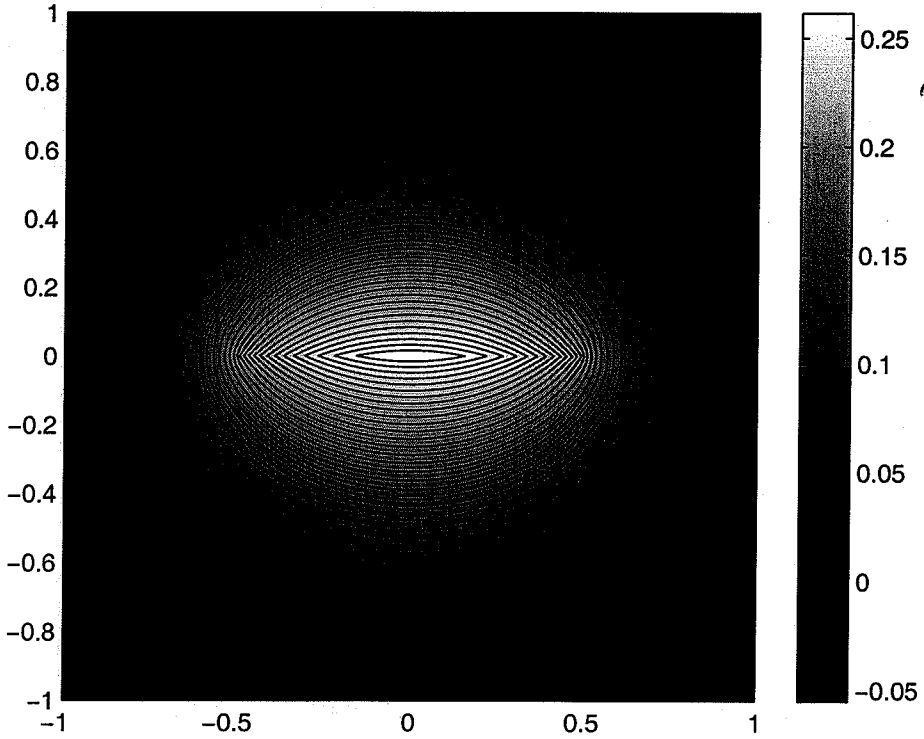


along the real axis (with $\hat{n}$ pointing down)

u jumps by 1 as you cross the dipole sheet

(but $\frac{du}{dy}$ is actually continuous across the sheet)

single layer potential, same density



u is continuous but $\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial y}$ jumps by 1 as you cross the sheet of charge from outside to inside

(from the lower half plane to the upper half-plane)

so if we want $\frac{\partial u}{\partial n_x}(\vec{x}_0^-) = g(\vec{x}_0)$, we need ϕ to satisfy

$$\frac{1}{2}\phi(\vec{x}) - \int_\Gamma K(\vec{\xi},\vec{x})\,\phi(\vec{\xi})\,d\xi = g(\vec{x})$$

summary: the integral equations of potential theory are:

interior Dirichlet: $\left(\frac{1}{2}\mathbb{I} + \mathbb{K}\right)\phi = g$

exterior Dirichlet: $\left(\frac{1}{2}\mathbb{I} - \mathbb{K}\right)\phi = -g$

interior Neumann: $\left(\frac{1}{2}\mathbb{I} - \mathbb{K}^*\right)\phi = g$

exterior Neumann: $\left(\frac{1}{2}\mathbb{I} + \mathbb{K}^*\right)\phi = -g$

adjoint pairs of equations

$g = \frac{\partial u}{\partial n} = \nabla u \cdot n$, n = outward normal from Ω in both cases

order reversed. it's like a transpose

$$\mathbb{K}\phi(\vec{x}) = \int_\Gamma K(\vec{x},\vec{\xi})\,\phi(\vec{\xi})\,ds, \quad \mathbb{K}^*\phi(\vec{x}) = \int_\Gamma K(\vec{\xi},\vec{x})\,\phi(\vec{\xi})\,ds$$

$\mathbb{K}$ and $\mathbb{K}^*$ are adjoints on $L^2(\Gamma)$: $(\mathbb{K}\phi,\psi) = (\phi,\mathbb{K}^*\psi) \quad \forall \phi,\psi \in L^2(\Gamma)$

$(\phi,\psi) = \int_\Gamma \phi\bar{\psi}\,ds$

they are both compact operators (almost finite rank)

Fredholm alternative: Suppose $\mathbb{A}: L^2(\Gamma) \to L^2(\Gamma)$ has the form $\mathbb{A} = \alpha\mathbb{I} + \mathbb{K}$ (nonzero scalar, compact operator)

then either: (1) $\mathbb{A}\phi = g$ and $\mathbb{A}^*\psi = \gamma$ have unique solutions $\forall g,\gamma \in L^2(\Gamma)$

- or -

(2) $\mathbb{A}\phi = g$ is solvable iff $(g,\psi) = 0 \quad \forall \psi \in N(\mathbb{A}^*)$

and $\mathbb{A}^*\psi = \gamma$ is solvable iff $(\gamma,\phi) = 0 \quad \forall \phi \in N(\mathbb{A})$

and these nullspaces have the same (finite) dimension.

the same thing happens in finite dimensions

example: $A = \begin{pmatrix} 3 & 2 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$, $A^* = \begin{pmatrix} 3 & 0 & 1 \\ 2 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $N(A^*) = \text{span}\{w\}$, $w = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$

$N(A) = \text{span}(e_3)$, $e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

is there a solution of $Ax = b$, $b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$? yes $b^T w = 0$ ✓

- - - - - - - - - - - - , $b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$? no $b^T w = 1 \neq 0$

The Fredholm alternative reduces the question of solvability (for which g is there a solution?) to the question of uniqueness (if $g = 0$, how many solutions are there?). The only complication is that the second question pertains to the adjoint problem

(interior Dirichlet $\leftrightarrow$ exterior Neumann
exterior " $\leftrightarrow$ interior ") more later...

Numerical method (collocation approach, Nystrom's method)

Step 1: choose a parametrization $\vec{\xi}(t)$ for Γ

Let's assume t runs from 0 to 2π. (can always rescale t so this is true)

the equation (in the interior Dirichlet case) becomes

$$\frac{1}{2}\phi(\vec{x}) + \int_0^{2\pi} k(\vec{x}, \vec{\xi}(t))\, \phi(\vec{\xi}(t)) \frac{ds}{dt}\, dt = g(\vec{x})$$

must hold for all $\vec{x} \in \Gamma$, i.e.
for $\vec{x} = \vec{\xi}(t_0)$, $0 \leq t_0 \leq 2\pi$

$\frac{ds}{dt} = \sqrt{\dot{\xi}^2 + \dot{\eta}^2}$

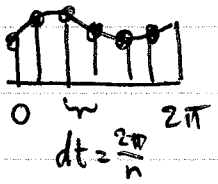
Step 2: choose a quadrature scheme for the integral.

want abscissas t_j $\quad 1 \le j \le n$

and weights w_j

so that $\int_0^{2\pi} f(t)\,dt \approx \sum_{j=1}^{n} f(t_j) w_j$ for all sufficiently smooth f.

Since our curve is smooth and periodic, you can't do better than the trapezoidal rule. $t_j = \frac{j-1}{n}$, $w_j = \frac{2\pi}{n}$



Step 3: enforce the discretized integral equation at the quadrature nodes only.

$$\vec{x}_k = \vec{\xi}(t_k), \quad \vec{\xi}_j = \vec{\xi}(t_j), \quad \dot{\vec{\xi}}_j = \dot{\vec{\xi}}(t_j), \quad \phi_k = \phi(\vec{\xi}(t_k)), \quad g_k = g(\vec{\xi}(t_k))$$

$$\frac{1}{2}\phi_k + \sum_{j=1}^{n} K(\vec{x}_k, \vec{\xi}_j)\,\phi_j \frac{ds}{dt}(t_j)\frac{2\pi}{n} = g_k$$

$A\phi = g$ ← finite dim'l linear system

$$A_{kj} = \frac{1}{2}\delta_{kj} + \frac{2\pi}{n} K(\vec{x}_k, \vec{\xi}_j)\frac{ds}{dt}(t_j)$$

$$= \begin{cases} \frac{1}{2} + \frac{1}{2n}\,\dfrac{\dot{\xi}_k \ddot{\eta}_k - \dot{\eta}_k \ddot{\xi}_k}{\dot{\xi}_k^2 + \dot{\eta}_k^2} & j = k \\ \frac{1}{n}\,\dfrac{(\xi_j - x_k)\dot{\eta}_j - (\eta_j - y_k)\dot{\xi}_j}{(\xi_j - x_k)^2 + (\eta_j - y_k)^2} & j \neq k \end{cases}$$

step 4: solve the equation (Gaussian elimination or GMRES)
now you know the dipole moments at the quadrature nodes.

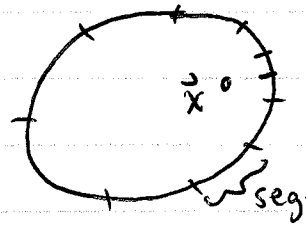
step 5: evaluate $u(\vec{x}) = \frac{1}{2\pi}\int_0^{2\pi} \frac{d\theta}{dt}(\vec{x},\vec{\xi}(t))\,\phi(\vec{\xi}(t))\,dt$

$$\simeq \frac{1}{n}\sum_{j=1}^{n} \underbrace{\frac{d\theta}{dt}(\vec{x},\vec{\xi}_j)}_{\frac{(\xi_j - x)\dot{\eta}_j - (\eta_j - y)\dot{\xi}_j}{(\xi_j - x)^2 + (\eta_j - y)^2}} \phi_j$$

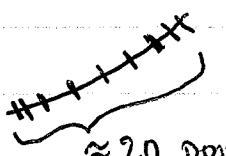
at the desired interior or exterior points.

when $\vec{x}$ is close to Γ, you actually need more resolution to do the integral than is needed to represent ϕ.

2 choices: (1) abandon the trapezoidal rule and break Γ into unequally spaced segments with Gaussian quadrature nodes on each.



on each segment:

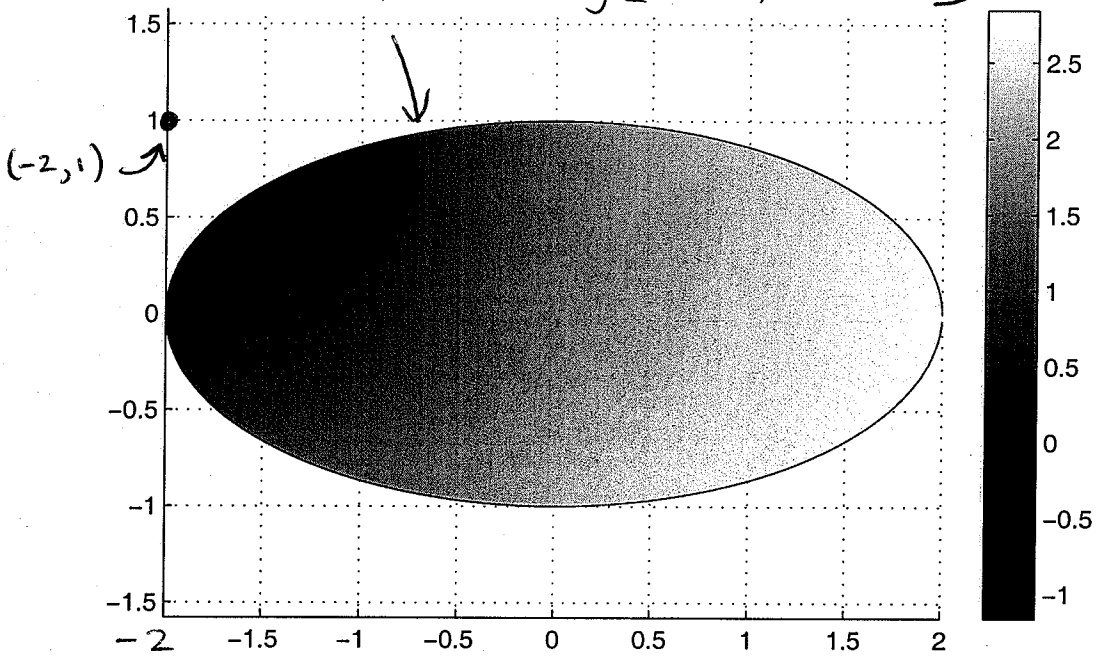


-or-

(2) refine the trap. rule by adding more equally spaced nodes.

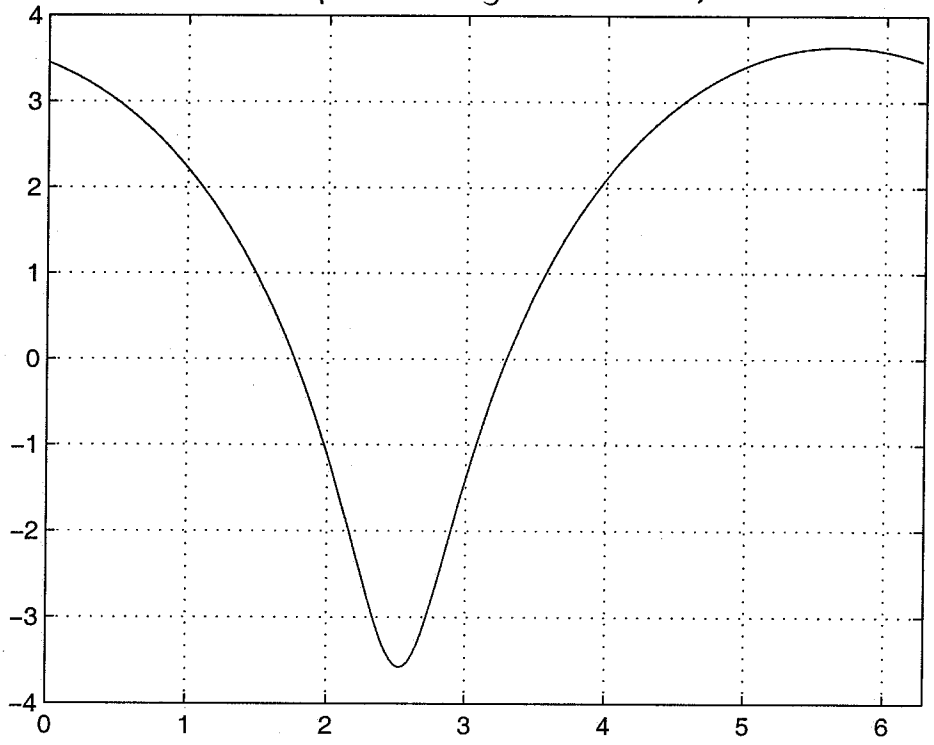
Dirichlet problem on the ellipse $\begin{cases} \xi(t) = 2\cos t \\ \eta(t) = \sin t \end{cases}$ $0 \le t \le 2\pi$

exact soln: $\log[(x+2)^2 + (y-1)^2]$



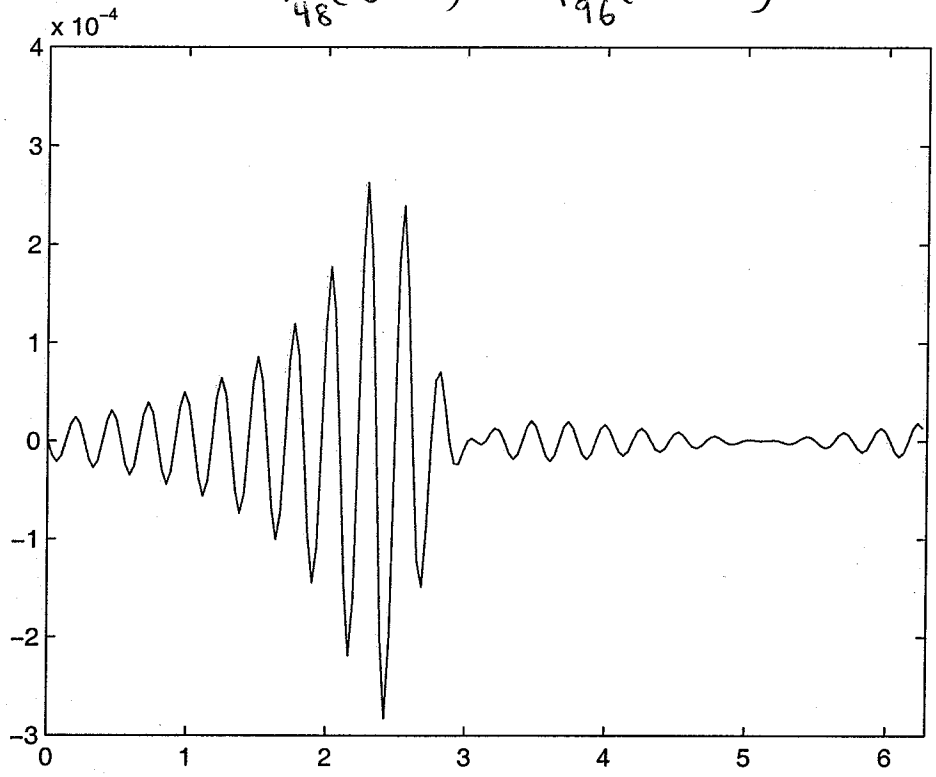
use double layer potential to represent solution.

dipole density $\phi(\vec{\xi}(t))$

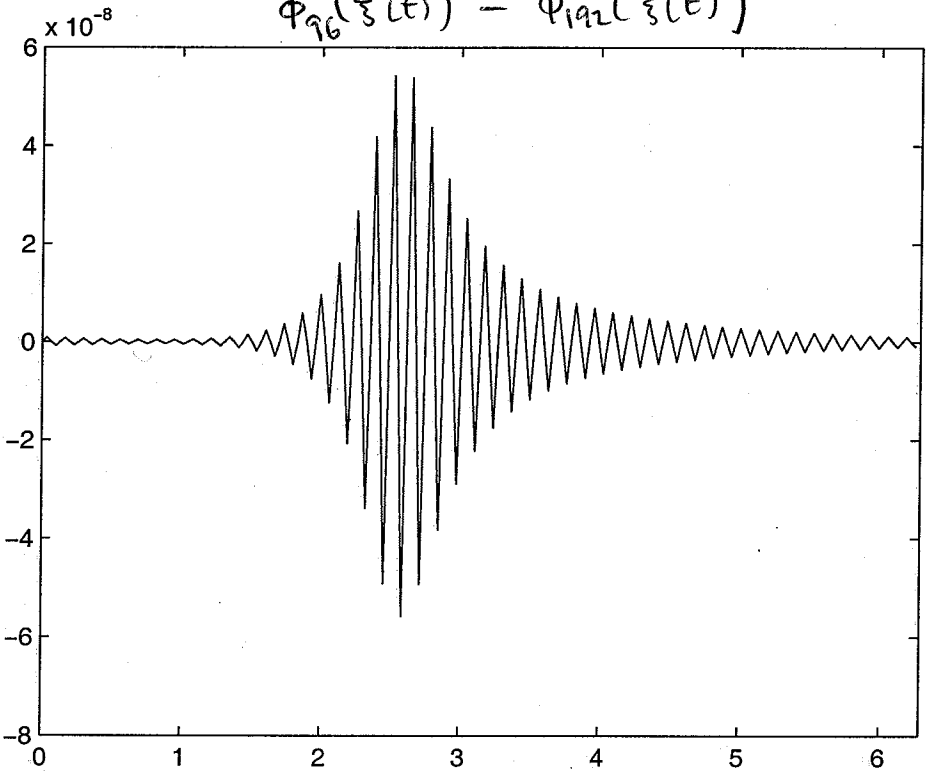


← found using $n = 192$ in Nystrom's method with the trapezoidal rule $(0 \le t \le 2\pi)$

$\phi_{48}(\vec{\xi}(t)) - \phi_{96}(\vec{\xi}(t))$



$\phi_{96}(\vec{\xi}(t)) - \phi_{192}(\vec{\xi}(t))$



fourier coefficients, n=48

1.47506510306981
1.17157287183890 + 0.828427126215221
-0.17157286794385 - 0.485281373008801
-0.06700338554892 + 0.236892698007991
0.10303038680077 - 0.083261102060361
-0.07587334071957 + 0.004877295776111
0.03872149516501 + 0.023808943359891
-0.01098900011898 - 0.025702224445371
-0.00368291894068 + 0.017156807978591
0.00823468351049 - 0.007577459426111
-0.00716619062192 + 0.000924975706091
0.00416444433007 + 0.002206000281461
-0.00139838587175 - 0.002765811788341
-0.00030161556632 + 0.002030100677751
0.00094508423384 - 0.000987908373071
-0.00089857283098 + 0.000174194257671
0.00056058229380 + 0.000253879565191
-0.00020896270414 - 0.000358777374481
-0.00002629007908 + 0.000279612047191
0.00012560662045 - 0.000142838435491
-0.00012595052726 + 0.000026611663531
0.00007770344372 + 0.000038777603751
-0.00002121318915 - 0.000054499696311
-0.00001962492630 + 0.000035644469491
0.00003415751729

fourier coefficients, n=96

1.47506510158731
1.17157287525381 + 0.828427124746191
-0.17157287525381 - 0.485281374238571
-0.06700337559222 + 0.236892706218251
0.10303038033001 - 0.083261120685231
-0.07587334789365 + 0.004877323527901
0.03872152643899 + 0.023808916454701
-0.01098905908391 - 0.025702218651921
-0.00368284505318 + 0.017156849861801
0.00823463163057 - 0.007577569415381
-0.00716622079474 + 0.000925147162061
0.00416462132028 + 0.002205824712541
-0.00139873816053 - 0.002765752763931
-0.00030115427146 + 0.002030323177141
0.00094472809211 - 0.000988551197731
-0.00089868820232 + 0.000175243316301
0.00056158829895 + 0.000252744078501
-0.00021108779664 - 0.000358279099501
-0.00002337659956 + 0.000280793217451
0.00012315984041 - 0.000146654482691
-0.00012624705775 + 0.000033149044251
0.00008350914076 + 0.000031309507431
-0.00003431562222 - 0.000050525334871
-0.00000079071118 + 0.000041906223361
0.00001707875864 - 0.000023211383231
-0.00001883421512 + 0.000006261753871
0.00001310243307 + 0.000003974361441
-0.00000580569705 - 0.000007468096311
0.00000029653049 + 0.000006537380721
0.00000244678005 - 0.000003816047191
-0.00000291347952 + 0.000001181170251
0.00000212509252 + 0.000000498274981
-0.00000100600517 - 0.000001135486661
0.00000011537135 + 0.000001049058561
0.00000035614181 - 0.000000642824531
-0.00000046129514 + 0.000000222499241
0.00000035228945 + 0.000000059024301
-0.00000017699121 - 0.000000175567941
0.00000003017333 + 0.000000171453071
0.00000005188256 - 0.000000109983671
-0.00000007389876 + 0.000000041876831
0.00000005899189 + 0.000000005790811
-0.00000003131688 - 0.000000026868291
0.00000000720361 + 0.000000027637591
0.00000000655065 - 0.000000018394931
-0.00000001032546 + 0.000000007906181
0.00000000812217 - 0.000000001093131
-0.00000000455321 - 0.000000000979361
0.00000000296501

fourier coefficients n=192

1.47506510158731
1.17157287525381 + 0.828427124746191
-0.17157287525381 - 0.485281374238571
-0.06700337559222 + 0.236892706218251
0.10303038033001 - 0.083261120685231
-0.07587334789365 + 0.004877323527901
0.03872152643899 + 0.023808916454701

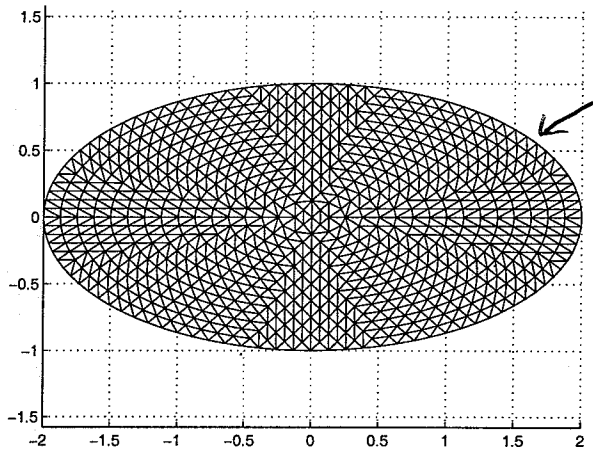
Fourier coefficients of $\phi(\vec{\xi}(t))$ found using the FFT of the sdn for n=48, 96, 192

I've only listed $c_0, c_1, \ldots, c_{n/2}$

(the rest are complex conjugates of these)

ϕ is fully resolved when n=192

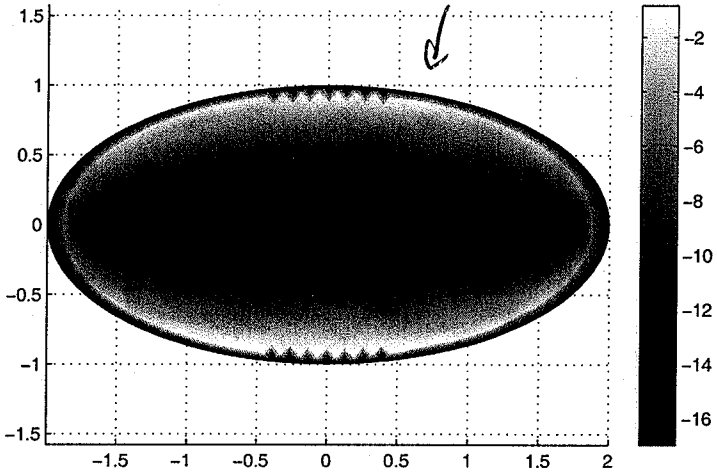
-0.01098905908391 - 0.025702218651921
-0.00368284505319 + 0.017156849861801
0.00823463163057 - 0.007577569415381
-0.00716622079474 + 0.000925147162061
0.00416462132029 + 0.002205824712551
-0.00139873816053 - 0.002765752763951
-0.00030115427148 + 0.002030323177161
0.00094472809214 - 0.000988551197741
-0.00089868820237 + 0.000175243316281
0.00056158829899 + 0.000252744078571
-0.00021108779664 - 0.000358279099601
-0.00002337659965 + 0.000280793217561
0.00012315984060 - 0.000146654482741
-0.00012624705802 + 0.000033149044151
0.00008350914100 + 0.000031309507771
-0.00003431562222 - 0.000050525335451
-0.00000079071165 + 0.000041906224021
0.00001707875973 - 0.000023211383601
-0.00001883421668 + 0.000006261753411
0.00001310243450 + 0.000003974363231
-0.00000580569723 - 0.000007468099551
0.00000029652806 + 0.000006537384611
0.00000244678604 - 0.000003816049701
-0.00000291348849 + 0.000001181168261
0.00000212510125 + 0.000000498284621
-0.00000100600739 - 0.000001135504941
0.00000011535897 + 0.000001049081501
0.00000035617489 - 0.000000642840941
-0.00000046134683 + 0.000000221491181
0.00000035234265 + 0.000000059075921
-0.00000017701054 - 0.000000175671221
0.00000003011116 + 0.000000171588461
0.00000005206542 - 0.000000110089551
-0.00000007419730 + 0.000000041849691
0.00000005931559 + 0.000000006066871
-0.00000003146583 - 0.000000027453621
0.00000000689640 + 0.000000028438421
0.00000000756386 - 0.000000019071891
-0.00000001205663 + 0.000000007860361
0.00000001009416 + 0.000000000381061
-0.00000000563273 - 0.000000004310651
0.00000000148251 + 0.000000004757061
0.00000000107952 - 0.000000003331291
-0.00000000197199 + 0.000000001474191
0.00000000173117 - 0.000000000045821
-0.00000000101321 - 0.000000000676961
0.00000000030721 + 0.000000000800841
0.00000000014895 - 0.000000000585331
-0.00000000032371 + 0.000000000276071
0.00000000029855 - 0.000000000027141
-0.00000000018286 - 0.000000000105871
0.00000000006217 + 0.000000000135391
0.00000000001933 - 0.000000000103281
-0.00000000005320 + 0.000000000051621
0.00000000005168 - 0.000000000008071
-0.00000000003308 - 0.000000000016411
0.00000000001238 + 0.000000000022941
0.00000000000222 - 0.000000000018281
-0.00000000000873 + 0.000000000009641
0.00000000000897 - 0.000000000002001
-0.00000000000599 - 0.000000000002511
0.00000000000243 + 0.000000000003891
0.00000000000018 - 0.000000000003241
-0.00000000000143 + 0.000000000001801
0.00000000000156 - 0.000000000000451
-0.00000000000109 - 0.000000000000371
0.00000000000047 + 0.000000000000661
-0.00000000000000 - 0.000000000000581
-0.00000000000023 + 0.000000000000331
0.00000000000027 - 0.000000000000101
-0.00000000000020 - 0.000000000000051
0.00000000000009 + 0.000000000000111
-0.00000000000001 - 0.000000000000101
-0.00000000000004 + 0.000000000000061
0.00000000000005 - 0.000000000000021
-0.00000000000004 - 0.000000000000011
0.00000000000002 + 0.000000000000021
-0.00000000000000 - 0.000000000000021
-0.00000000000001 + 0.000000000000011
0.00000000000001 - 0.000000000000001i
-0.00000000000001 - 0.000000000000001i
0.00000000000000 + 0.000000000000001i
-0.00000000000000 - 0.000000000000001i
-0.00000000000000 + 0.000000000000001i
0.00000000000000 - 0.000000000000001i
-0.00000000000000 + 0.000000000000001i
-0.00000000000000 + 0.000000000000001i
-0.00000000000000 - 0.000000000000001i
-0.00000000000000 + 0.000000000000001i
-0.00000000000000



$n = 96$ points around boundary

(interior points are for contour plots)

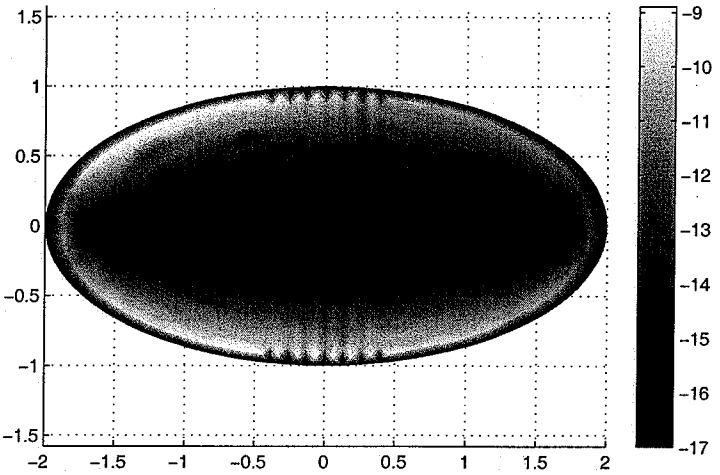
contour plot of $\log_{10}|u_{approx} - u_{exact}|$



$$u(\vec{x}) = \int_\Gamma -\frac{\partial N}{\partial n_\xi}\,\phi(\vec{\xi})\,ds$$

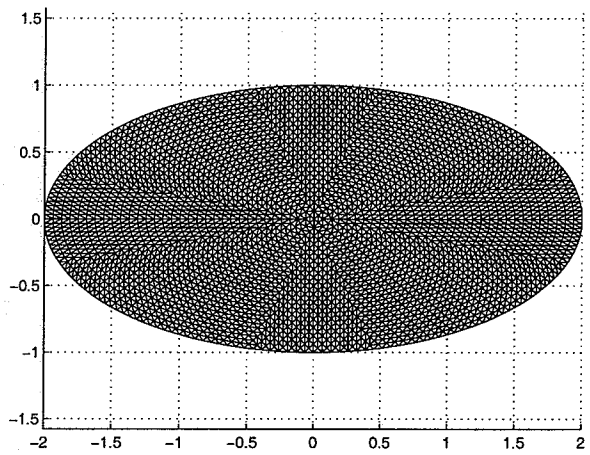
using the trapezoidal rule with 96 points

integration errors are large when x is close to Γ

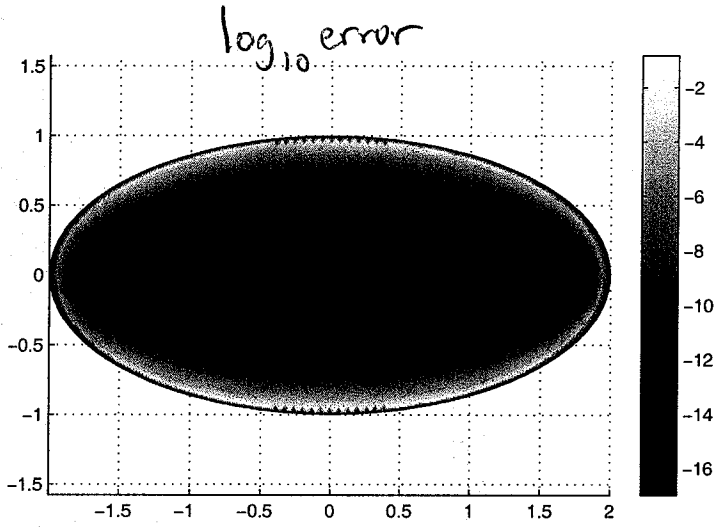


same density ϕ used but its fourier transform has been used to reconstruct ϕ on the boundary in between grid points.

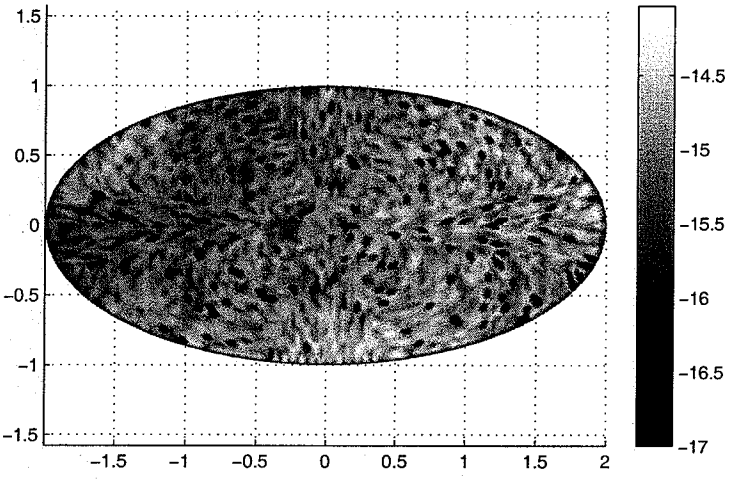
$u(\vec{x})$ is then evaluated in the interior using $96 \cdot 12$ points in the trapezoidal rule.



← n = 192

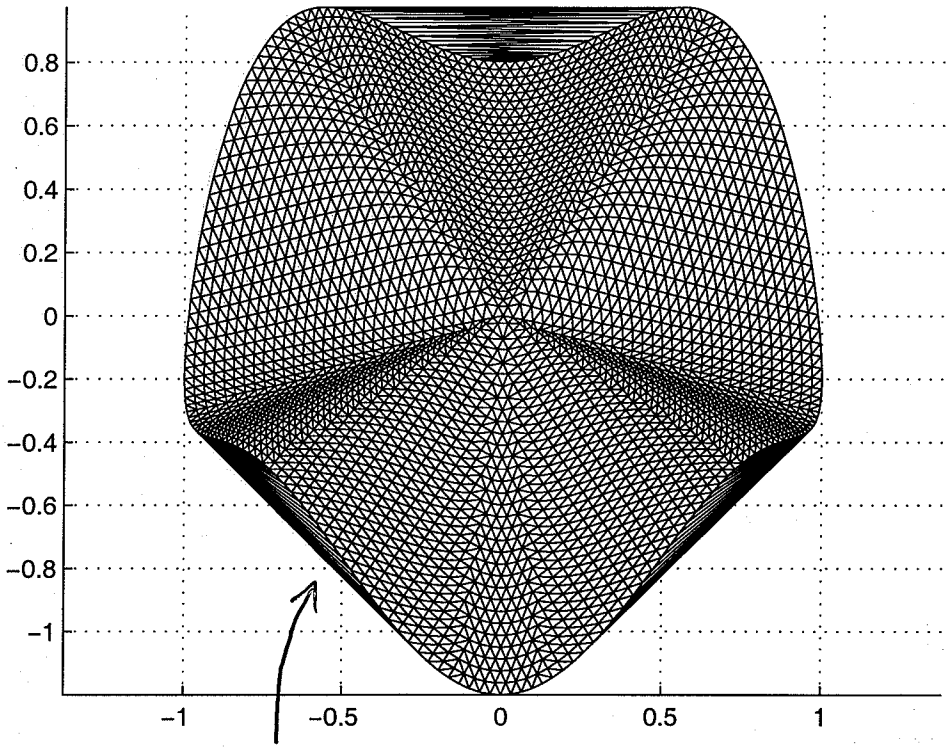


trap. rule with
192 integration points
(big errors near T)



trap rule with
192 · 12
integration points.
(soln is essentially exact)

Dirichlet problem on a more complicated (non-convex) geometry



$\xi(t) = \cos t$

$\eta(t) = \sin t - .2\cos 4t$

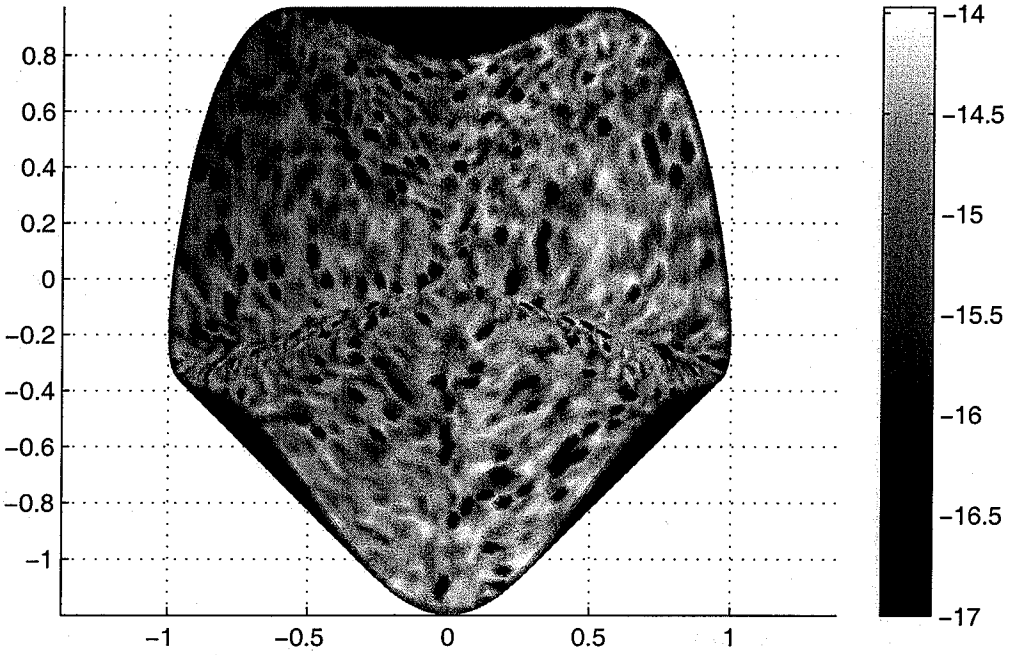
exact soln:

$u(x,y) = \log[(x+2)^2 + (y-1)^2]$

n = 192

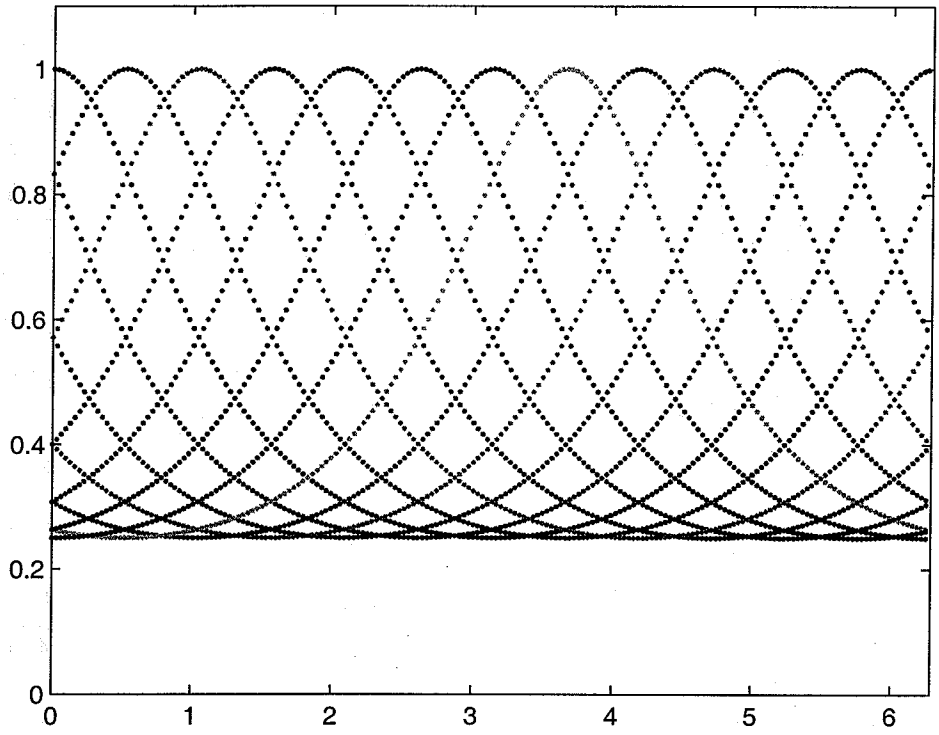
$\log_{10}$ error

(still essentially exact with n = 192)



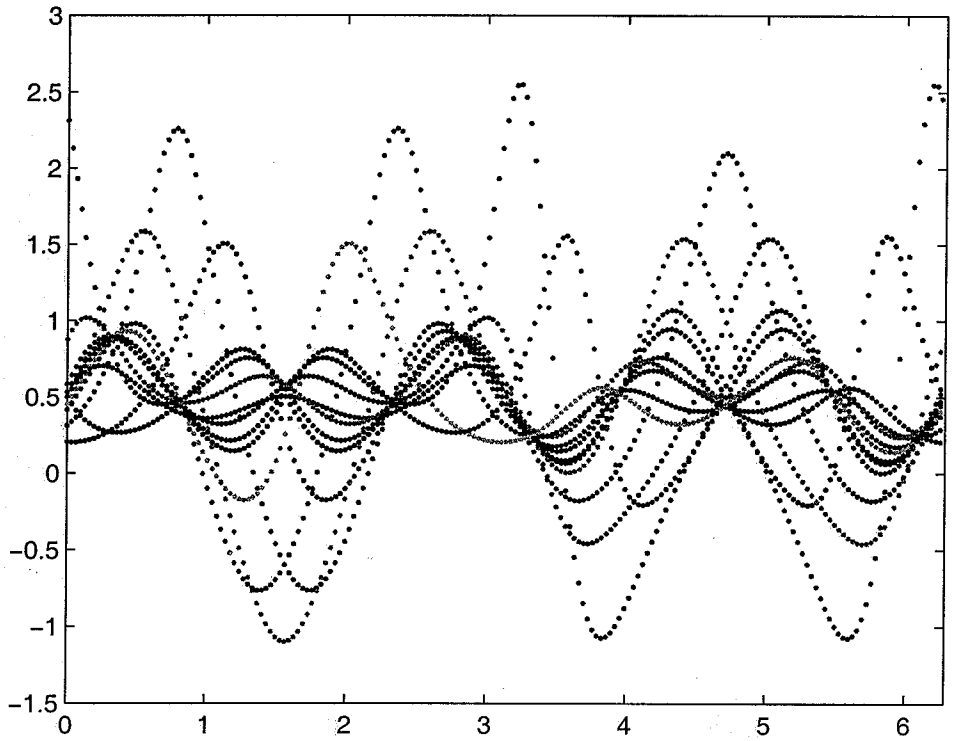
reconstruction used 12n quadrature points in trapezoidal rule.

plot of rows 1:12:192 of $192\left(A - \frac{1}{2}I\right)$



← for ellipse.

This geometry is special. Normally the rows aren't translates of each other and $A \neq A^T$ (although they are equal here)



← for  domain.

This time $A \neq A^T$

note that nothing bad (i.e. discontinuous) happens when $i = j$ (as ξ passes through x)

ndsideways

```
eig  n=48

  1.00000000000000
  0.66666666666667
  0.55555555555556
  0.51851851851852
  0.50617283950617
  0.50205761316872
  0.50068587105624
  0.50022862368541
  0.50007620789514
  0.50002540263171
  0.50000846754390
  0.50000282251463
  0.50000094083821
  0.50000031361274
  0.50000010453758
  0.50000003484586
  0.50000001161529
  0.50000000387176
  0.50000000129059
  0.50000000043020
  0.50000000014342
  0.50000000004787
  0.50000000001613
  0.50000000000590
  0.50000000000354
  0.49999999999410
  0.49999999998387
  0.49999999995213
  0.49999999985658
  0.49999999956980
  0.49999999870941
  0.49999999612824
  0.49999998838471
  0.49999996515414
  0.49999989546242
  0.49999968638726
  0.49999905916179
  0.49999717748537
  0.49999153245610
  0.49997459736829
  0.49992379210486
  0.49977137631459
  0.49931412894376
  0.49794238683128
  0.49382716049383
  0.48148148148148
  0.44444444444444
  0.33333333333333

eig  n=96

  1.00000000000000
  0.66666666666667
  0.55555555555556
  0.51851851851852
  0.50617283950617
  0.50205761316872
  0.50068587105624
  0.50022862368541
  0.50007620789514
  0.50002540263171
  0.50000846754390
  0.50000282251463
  0.50000094083821
  0.50000031361274
  0.50000010453758
  0.50000003484586
  0.50000001161529
  0.50000000387176
  0.50000000129059
  0.50000000043020
  0.50000000014340
  0.50000000004780
  0.50000000001593
  0.50000000000531
  0.50000000000177
  0.50000000000059
  0.50000000000020
  0.50000000000007
  0.50000000000002
  0.50000000000001
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.49999999999998
  0.49999999999993
  0.49999999999980
  0.49999999999941
  0.49999999999823
  0.49999999999469
  0.49999999998407
  0.49999999995220
  0.49999999985660
  0.49999999956980
  0.49999999870941
  0.49999999612824
  0.49999998838471
  0.49999996515414
  0.49999989546242
  0.49999968638726
  0.49999905916179
  0.49999717748537
  0.49999153245610
  0.49997459736829
  0.49992379210486
  0.49977137631459
  0.49931412894376
  0.49794238683128
  0.49382716049383
  0.48148148148148
  0.44444444444444
  0.33333333333333

eig  n=192

  1.00000000000000
  0.66666666666667
  0.55555555555556
  0.51851851851852
  0.50617283950617
  0.50205761316872
  0.50068587105624
  0.50022862368541
  0.50007620789514
  0.50002540263171
  0.50000846754390
  0.50000282251463
  0.50000094083821
  0.50000031361274
  0.50000010453758
  0.50000003484586
  0.50000001161529
  0.50000000387176
  0.50000000129059
  0.50000000043020
  0.50000000014340
  0.50000000004780
  0.50000000001593
  0.50000000000531
  0.50000000000177
  0.50000000000059
  0.50000000000020
  0.50000000000007
  0.50000000000002
  0.50000000000001
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
   ** snip **
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.50000000000000
  0.49999999999998
  0.49999999999993
  0.49999999999980
  0.49999999999941
  0.49999999999823
  0.49999999999469
  0.49999999998407
  0.49999999995220
  0.49999999985660
  0.49999999956980
  0.49999999870941
  0.49999999612824
  0.49999998838471
  0.49999996515414
  0.49999989546242
  0.49999968638726
  0.49999905916179
  0.49999717748537
  0.49999153245610
  0.49997459736829
  0.49992379210486
  0.49977137631459
  0.49931412894376
  0.49794238683128
  0.49382716049383
  0.48148148148148
  0.44444444444444
  0.33333333333333
```

eigenvalues of A for ellipse.

condition number $\|A\| \cdot \|A^{-1}\|$ is 3 in all three cases (linear system does not become ill conditioned as $n \to \infty$)

↑ beauty of 2^{nd} kind Fredholm integral equations.

```
eig  n=48
1.00000002720137
0.66212897440957
0.64231156658645
0.62032358422807
0.59581918898569
0.55112633382626
0.51930655798058
0.51516312728687
0.51404581914189
0.50842356107427
0.50473226034420
0.50303352472482
0.50176382446735
0.50130204733410
0.50065466472906
0.50050403896410
0.50048432085585
0.50019218165161
0.50012688574654
0.50005365102457
0.50004365740823
0.50003386970351
0.50001207474031
0.50001036642304
0.50000392730356
0.49999914852075
0.49999802910553
0.49999304465157
0.49998318547147
0.49996121263169
0.49989993727444
0.49988849231401
0.49972571269869
0.49959952654497
0.49937153221209
0.49871981129077
0.49836070349255
0.49708850128669
0.49530334814088
0.49158009828610
0.48595680816705
0.48484405741088
0.48069797004928
0.44887365172745
0.40418091237555
0.37967648090731
0.35768840841099
0.33787102486619
```

```
eig  n=96

1.00000000000000
0.66212896862754
0.64231141404019
0.62032322986622
0.59581904853187
0.55112622046272
0.51930352782531
0.51515854374798
0.51404413661521
0.50842140295797
0.50471184419596
0.50296512983482
0.50169454299370
0.50128956422816
0.50063828548969
0.50045475363902
0.50038944734053
0.50017349857739
0.50011817541801
0.50005270965319
0.50005039099150
0.50004143470352
0.50001722357618
0.50001130202252
0.50000616869649
0.50000528403599
0.50000317904233
0.50000172735693
0.50000121013106
0.50000069274124
0.50000055567158
0.50000023594675
0.50000016288449
0.50000012509997
0.50000006186926
0.50000004290250
0.50000001731966
0.50000000958640
0.50000000778092
0.50000000299483
0.50000000197248
0.50000000141892
0.50000000071189
0.50000000037221
0.50000000010957
0.50000000003676
0.50000000003560
0.50000000000621
0.50000000000356
0.49999999999677
0.49999999998850
0.49999999994581
0.49999999992783
0.49999999965044
0.49999999962187
0.49999999904219
0.49999999773795
0.49999999610650
0.49999999251747
0.49999999008534
0.49999998298065
0.49999996023797
0.49999994268182
0.49999989468670
0.49999985622903
0.49999976484992
0.49999950036287
0.49999935821658
0.49999886084587
0.49999833997524
0.49999682384356
0.49999477532345
0.49999389999518
0.49998871641082
0.49998280072006
0.49995856782748
0.49994962072850
0.49994729598764
0.49988182509640
0.49982650231465
0.49961055276122
0.49954524637874
0.49936171450156
0.49871043576747
0.49830545698490
0.49703487015307
0.49528815580210
0.49157859704191
0.48595586338479
0.48484145625209
0.48069647217487
0.44887377953728
0.40418095146814
0.37967677013362
0.35768858595970
0.33787103137245
```

```
eig  n=192

1.00000000000000
0.66212896862755
0.64231141404026
0.62032322986638
0.59581904853193
0.55112622046278
0.51930352782563
0.51515854374840
0.51404413661533
0.50842140295811
0.50471184419786
0.50296512984404
0.50169454300822
0.50128956423105
0.50063828549617
0.50045475363939
0.50038944731381
0.50017349816612
0.50011817517278
0.50005270692324
0.50005038530691
0.50004143346844
0.50001721174193
0.50001129303289
0.50000613575146
0.50000525561491
0.50000317768638
0.50000169779726
0.50000117984102
0.50000067696326
0.50000054002961
0.50000023598509
0.50000016858220
0.50000013655539
0.50000006759563
0.50000005499611
0.50000001788559
0.50000001676041
0.50000001644194
0.50000000640607
0.50000000552262
0.50000000212796
0.50000000175713
0.50000000122791
0.50000000059452
0.50000000055101
0.50000000027808
0.50000000017891
0.50000000008642
0.50000000005517
0.50000000005479
0.50000000003731
0.50000000001808
0.50000000000612
0.50000000000546
0.50000000000541
0.50000000000472
0.50000000000180
0.50000000000071
0.50000000000057
0.50000000000044
0.50000000000039
0.50000000000016
0.50000000000008
0.50000000000005
0.50000000000004
0.50000000000003
0.50000000000001
0.50000000000001
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.50000000000000
0.49999999999999
0.49999999999999
0.49999999999998
0.49999999999997
0.49999999999996
0.49999999999993
0.49999999999988
0.49999999999981
0.49999999999961
0.49999999999956
0.49999999999940
0.49999999999923
0.49999999999819
0.49999999999528
0.49999999999459
0.49999999999454
0.49999999999388
0.49999999998193
0.49999999996271
0.49999999994522
0.49999999994484
0.49999999991358
0.49999999982110
0.49999999972193
0.49999999944899
0.49999999940548
0.49999999877209
0.49999999824287
0.49999999787204
0.49999999447738
0.49999999359393
0.49999998355806
0.49999998323959
0.49999998211441
0.49999994500389
0.49999993240437
0.49999986344461
0.49999983141780
0.49999976401491
0.49999945997039
0.49999932303674
0.49999882015898
0.49999830220274
0.49999682231362
0.49999474438509
0.49999386424855
0.49998870696711
0.49998278825807
0.49995856653156
0.49994961469309
0.49994729307676
0.49988182482722
0.49982650183388
0.49961055268619
0.49954524636061
0.49936171450384
0.49871043576894
0.49830545699178
0.49703487015596
0.49528815580214
0.49157859704189
0.48595586338467
0.48484145625160
0.48069647217437
0.44887377953722
0.40418095146807
0.37967677013362
0.35768858595974
0.33787103137245
```

eigenvalues of A for

domain.

Although A is not self-adjoint, it's singular values (from the SVD) are very similar to its eigenvalues.

condition numbers:

n=48: 3.21

n=96: 3.21

n=192: 3.21

max singular value: 1.0296

2nd largest: 0.6816

smallest 0.3212

in either case you have to interpolate the ϕ_j values away from the original quadrature points t_j. Since we're aiming for spectral accuracy, use the fft:

$$\phi(t) \approx \sum_{k=-\frac{n}{2}+1}^{n/2} \hat{\phi}_k e^{ikt}, \qquad \hat{\phi}_k = \frac{1}{n} \sum_{j=1}^{n} \phi_j e^{-2\pi i k(j-1)/n}$$

↑ evaluate this at the extra grid points (between the original t_j's where ϕ_j is known)

$=$

For the Neumann problem, steps 3 & 5 are slightly different.

step 3: $\frac{1}{2}\phi_k - \sum_{j=1}^{n} K(\vec{\xi}_j, \vec{x}_k)\phi_j \frac{ds}{dt}(\vec{\xi}_j)\frac{2\pi}{n} = g_k$

$\Rightarrow B^* \phi = g$ finite dim'l linear system

$$B^*_{kj} = \frac{1}{2}\delta_{kj} - \frac{2\pi}{n} K(\vec{\xi}_j, \vec{x}_k) \frac{ds}{dt}(\vec{\xi}_j)$$

$$= \begin{cases} \frac{1}{2} - \frac{1}{2n} \dfrac{\dot{\xi}_k \ddot{\eta}_k - \dot{\eta}_k \ddot{\xi}_k}{\dot{\xi}^2 + \dot{\eta}^2} & j = k \\ \frac{1}{n} \dfrac{(\xi_j - x_k)\dot{\eta}_k - (\eta_j - y_k)\dot{\xi}_k}{(\xi_j - x_k)^2 + (\eta_j - y_k)^2} \sqrt{\dfrac{\dot{\xi}_j^2 + \dot{\eta}_j^2}{\dot{\xi}_k^2 + \dot{\eta}_k^2}} & j \neq k \end{cases}$$

problem: if u is a solution of the Neumann problem, so is $u+C$.

$\rightarrow$ $B^* = \frac{1}{2}I - K^*$ is not invertible ($\underbrace{N(B) = \text{span}\{1\}}$)

$\phi = 1$ yields $g = 0$ for the exterior Dirichlet problem.

so as n becomes large, the matrix B^* becomes singular.

In fact, for large n, $R(B^*) = \{e^\perp\}$, $e = \frac{2\pi}{n}\begin{pmatrix} \frac{ds}{dt}(t_1) \\ \vdots \\ \frac{ds}{dt}(t_n) \end{pmatrix} \in \mathbb{R}^n$

But the right hand side g (discrete version) ends up belonging to $R(B^*)$ for large n since $\int_\Gamma g\, ds = 0$ (↑ continuous version)

by assumption. Note that $e^T g$ is the trapezoidal rule approximation of $\int_0^{2\pi} g \frac{ds}{dt}\, dt$.

So the equation can be solved via $\phi = \underbrace{\text{pinv}(B^*)}\, g$ for example.

pseudo-inverse

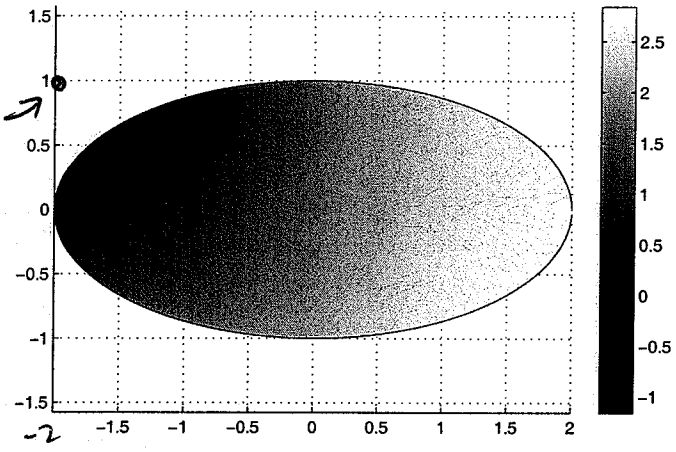
$=$

Step 5 for Neumann problem:

evaluate $u(\vec{x}) = \int_\Gamma N(\vec{x}, \vec{s})\, \phi(\vec{s})\, ds$

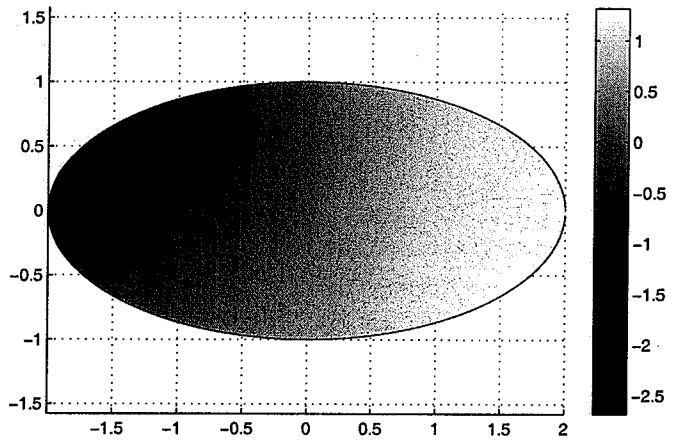
at the desired field points $\vec{x}$. For points in the interior it's best to stick with the trapezoidal rule. We still need to increase n by interpolation near the bdry, but only by a factor of 4 or so since $N = \frac{-1}{2\pi} \log|\vec{x} - \vec{s}|$ has a mild singularity.

Neumann Problem on the ellipse $\begin{cases} \xi(t) = 2\cos t \\ \eta(t) = \sin t \end{cases}$ $0 \le t \le 2\pi$



solution I started with is

$$u(x,y) = \log\left[(x+2)^2 + (y-1)^2\right]$$

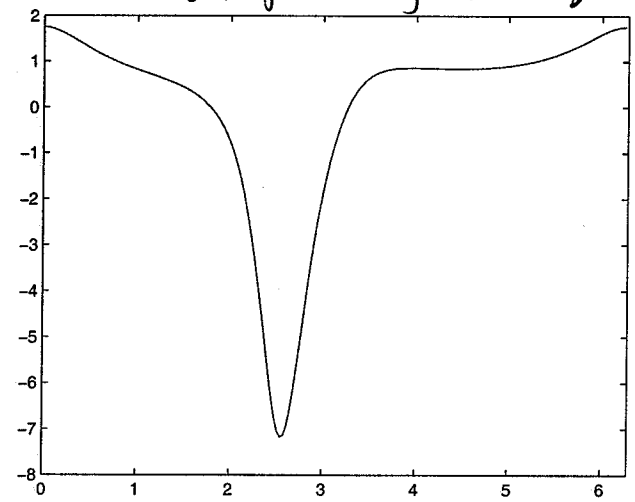


reconstructed solution from single layer potential

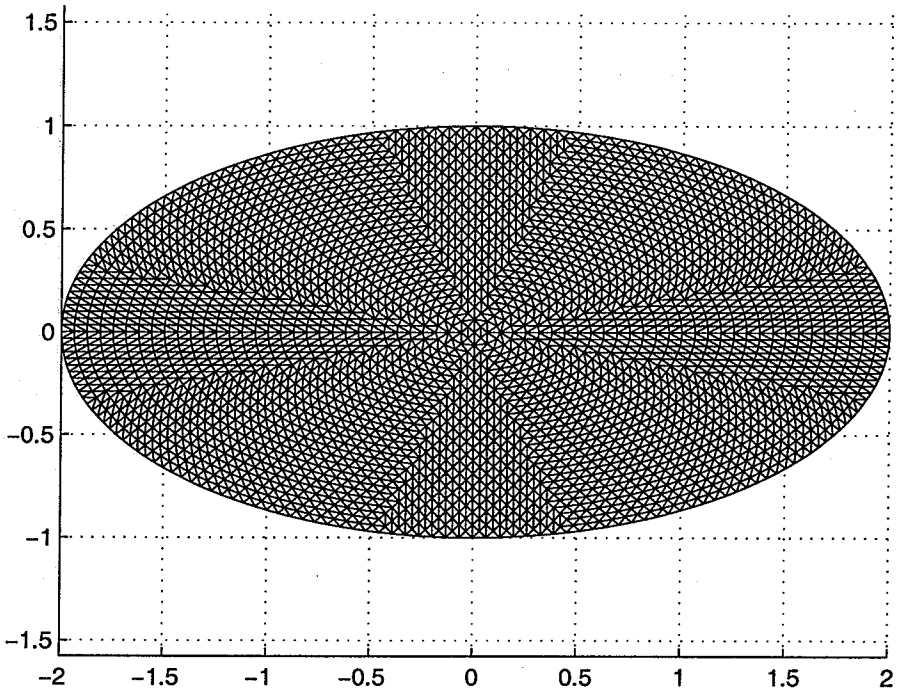
$$u(\vec{x}) = \int_\Gamma N(\vec{x},\vec{\xi})\,\phi(\vec{\xi})\,ds$$

differs from the original function by a constant

charge density $\phi(\vec{\xi}(t))$



← found using $n = 192$ in Nyström's method with the trapezoidal rule on $0 \le t \le 2\pi$

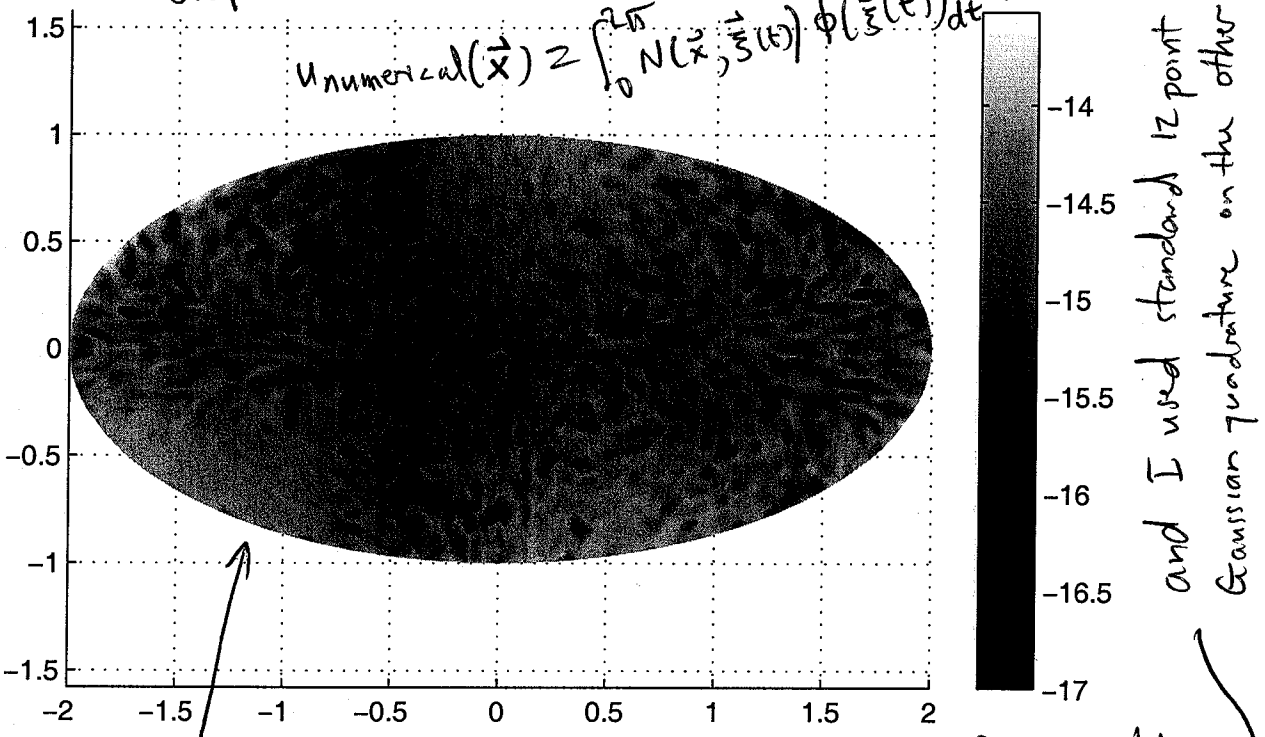


$n = 192$ points around boundary (interior points are for contour plots)

contour plot of

$$\log_{10}\left| u_{numerical} - u_{exact} - \overbrace{\left(u_{numerical}(0,0) - u_{exact}(0,0)\right)}^{\text{a constant}} \right|$$

$$u_{numerical}(\vec{x}) = \int_0^{2\pi} N(\vec{x}, \vec{\xi}(t))\, \phi(\vec{\xi}(t)) \frac{ds}{dt}\, dt$$



and I used standard 12 point Gaussian quadrature on the other segments

for interior nodes I used $12n$ equally spaced ($\ln t$) interpolation points for the trapezoidal rule

for boundary points, I broke the boundary into 100 segments and used the Ma/Rokhlin/Wandzura quadrature rule to integrate the segments touching the node (segments with a singularity)

SIAM J. NUMER. ANAL.
Vol. 33, No. 3, pp. 971-996, June 1996

GENERALIZED GAUSSIAN QUADRATURE RULES FOR SYSTEMS OF ARBITRARY FUNCTIONS*

J. MA†, V. ROKHLIN†, AND S. WANDZURA‡

Abstract. A numerical algorithm is presented for the construction of generalized Gaussian quadrature rules, originally introduced by S. Karlin and W. Studden over three decades ago. The quadrature rules to be discussed possess most of the desirable properties of the classical Gaussian integration formulae, such as positivity of the weights, rapid convergence, mathematical elegance, etc. The algorithm is applicable to a wide class of functions, including smooth functions (not necessarily polynomials), as well as functions with end-point singularities, such as those encountered in the solution of integral equations, complex analysis, potential theory, and several other areas. The performance of the algorithm is illustrated with several numerical examples.

Key words. numerical integration, quadrature rule, Gaussian quadrature

AMS subject classifications. 65D30, 65D32

1. Introduction. Classical Gaussian quadrature rules are extremely efficient when the functions to be integrated are well approximated by polynomials. When the functions to be integrated are very different from polynomials, Gaussian quadratures do not perform well; many particularly difficult and important problems involve the integration of functions of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i \cdot \varphi_i, \tag{1}$$

where each of the functions φ_i has its own singularity at one of the ends of the interval, and the function f can only be evaluated *in toto*, the coefficients α_i being unavailable. This problem is encountered in the solution of integral equations with singular kernels, in the numerical complex analysis, in the numerical solution of elliptic partial differential equations on regions with corners, and in many other situations. While such problems are normally dealt with by means of various ad hoc procedures (see, for example, [1], [8]), these schemes lack the rapid convergence, stability, and elegance of the Gaussian rules.

In fact, in [6], a far-reaching generalization of the classical Gaussian quadratures is introduced, replacing the polynomials with functions from an extremely wide class. The quadrature rules of [6] possess most of the desirable properties of the classical Gaussian integration formulae, such as positivity of the weights, rapid convergence, mathematical elegance, etc. Unfortunately, it is not clear from [6] how such quadrature rules can be obtained numerically.

In this paper, we present a numerical scheme for the construction of such generalized Gaussian quadratures. The algorithm is applicable to a variety of functions, including smooth functions (not necessarily polynomials), as well as functions with end-point singularities.

The paper is organized as follows. In §2, we restate the relevant results from [6], and in §3, we summarize the numerical techniques to be used in this paper. In §4, we develop the analytical apparatus to be used in the numerical construction of the generalized Gaussian quadrature rules. We extend those analytical tools to functions with end-point singularities in §5. The actual numerical algorithm is presented in §6, and the performance of the algorithm is demonstrated with numerical examples in §7.

*Received by the editors February 16, 1994; accepted for publication July 22, 1994.

†Department of Computer Science, Yale University, New Haven, CT 06520 (majh@math.ucla.edu). The research of these authors was supported in part by Defense Advanced Research Project Agency contract F49620/91/C/0084 and in part by Office of Naval Research grant N00014-89-J-1527.

‡Hughes Research Laboratories, 3011 Malibu Canyon Road, Malibu, CA 90265. The research of this author was supported in part by Defense Advanced Research Projects Agency contract F49620/91/C/0064.

EXAMPLE 7.1. *Gaussian quadratures with respect to the system of functions*

$$1, \ln x, x, x \ln x, x^2, x^2 \ln x, \ldots, x^{n-1}, x^{n-1} \ln x \tag{128}$$

on [0, 1] *are given in Table* 1, *and tested on selected functions in Table* 5.

EXAMPLE 7.2. *Gaussian quadratures with respect to the systems of functions*

$$1, x^\alpha, x, x^{1+\alpha}, x^2, x^{2+\alpha}, \ldots, x^{n-1}, x^{n-1+\alpha} \tag{129}$$

on [0, 1] *are given respectively in Tables* 2–4 *for*

$$\alpha = \frac{1}{3}, -\frac{1}{3}, -\frac{2}{3},$$

and tested on selected functions in Tables 6–8, *respectively.*

REMARK 7.1. *The algorithm of this paper has been applied to a large selection of functions* $\{\varphi_i\}$. *Space limitations prevent us from presenting more examples here. A detailed report on our numerical experiments can be found in* [7].

8. Conclusions. A numerical algorithm has been presented for the construction of the generalized Gaussian quadrature rules, introduced in [6]. The quadrature rules of this paper possess most of the desirable properties of the classical Gaussian integration formulae, such as positivity of the weights, rapid convergence, mathematical elegance, etc. The algorithm is applicable to a wide class of functions, including smooth functions (not necessarily polynomials), as well as functions with end-point singularities, such as those encountered in the solution of integral equations, complex analysis, potential theory, and several other areas.

TABLE 1

Gaussian quadrature for products of polynomials and logarithmic function.

$$\int_0^1 \varphi_k(x)dx = \sum_{i=1}^{N} w_i \varphi_k(x_i) \quad \text{for} \quad k = 1, 2, \ldots, 2N$$

where $\{\varphi_i\} = \{1, \ln x, x, x \ln x, \ldots, x^{N-1}, x^{N-1} \ln x\}$

N	Nodes x_i	Weights w_i
5	0.565222820508010E-02	0.210469457918546E-01
	0.734303717426523E-01	0.130705540744447E+00
	0.284957404462558E+00	0.289702301671314E+00
	0.619482264084778E+00	0.350220370120399E+00
	0.915758083004698E+00	0.208324841671986E+00
10	0.482961710689630E-03	0.183340007378985E-02
	0.698862921431577E-02	0.134531223459918E-01
	0.326113965946776E-01	0.404971943169583E-01
	0.928257573891660E-01	0.818223696589036E-01
	0.198327256895404E+00	0.129192342770138E+00
	0.348880142979353E+00	0.169545319547259E+00
	0.530440555787956E+00	0.189100216532996E+00
	0.716764648511655E+00	0.177965753961471E+00
	0.875234557506234E+00	0.133724770615462E+00
	0.975245698684393E+00	0.628655101770325E-01
15	0.105784548458629E-03	0.403217724648460E-03
	0.156624383616782E-02	0.306297843478700E-02
	0.759521890320709E-02	0.978421211876615E-02
	0.228310673939862E-01	0.215587522255813E-01
	0.523886301568200E-01	0.383230673708892E-01
	0.100758685201213E+00	0.588981990263004E-01
	0.170740768849943E+00	0.811170299392595E-01
	0.262591206118993E+00	0.102122101972069E+00
	0.373536505184558E+00	0.118789059030401E+00
	0.497746358414533E+00	0.128210316446694E+00
	0.626789031392373E+00	0.128163327417093E+00
	0.750516103461408E+00	0.117489465888492E+00
	0.858255335207861E+00	0.963230185695904E-01
	0.940141291212346E+00	0.661345398318934E-01
	0.988401595986342E+00	0.296207140035355E-01

TABLE 1
(cont.)

N	Nodes x_i	Weights w_i
20	0.352330453033401E-04	0.1344996764677 58E-03
	0.526093982517410E-03	0.103477692295062E-02
	0.258751954058141E-02	0.337726367723322E-02
	0.793447194838041E-02	0.767355619359468E-02
	0.186828881374457E-01	0.142054962855420E-01
	0.370976733697505E-01	0.229844384632086E-01
	0.653124886740214E-01	0.337363605577136E-01
	0.105048504711551E+00	0.459147630734522E-01
	0.157359691819002E+00	0.587404799428040E-01
	0.222430062767455E+00	0.712650131611020E-01
	0.299443765654100E+00	0.824518089775832E-01
	0.386542446943882E+00	0.912682015163873E-01
	0.480876453826790E+00	0.967797159091613E-01
	0.578747932205507E+00	0.982381433400897E-01
	0.675835475840038E+00	0.951553030540297E-01
	0.767482460872564E+00	0.873556504104574E-01
	0.849025253970320E+00	0.750027772122717E-01
	0.916133703241664E+00	0.585972958082337E-01
	0.965135427900256E+00	0.389472505496114E-01
	0.993303536456954E+00	0.171372052681059E-01
25	0.148805205646725E-04	0.568460660251700E-04
	0.223091159576968E-03	0.439997585769338E-03
	0.110464364905582E-02	0.145071890475996E-02
	0.341946946888592E-02	0.334401873817378E-02
	0.815052929503389E-02	0.630809954735919E-02
	0.164289374947977E-01	0.104488723103534E-01
	0.294459835598650E-01	0.157795036631362E-01
	0.483575697079336E-01	0.222157908473762E-01
	0.741870939197324E-01	0.295777024141012E-01
	0.107732955883526E+00	0.375970456071838E-01
	0.149486638258087E+00	0.459308515949819E-01
	0.199566730959288E+00	0.541797236657000E-01
	0.257673355831325E+00	0.619100915223073E-01
	0.323066266406720E+00	0.686790748928476E-01
	0.394568512069187E+00	0.740604961651023E-01
	0.470596049553408E+00	0.776705045127631E-01
	0.549212146433180E+00	0.791912877674548E-01
	0.628203942243430E+00	0.783914525088150E-01
	0.705177201069316E+00	0.751418416138532E-01
	0.777664184415814E+00	0.694258212157633E-01
	0.843238762138573E+00	0.613433919048206E-01
	0.899632416106180E+00	0.511088512980029E-01
	0.944844733405715E+00	0.390421895640092E-01
	0.977242575226693E+00	0.255554713626328E-01
	0.995647215456441E+00	0.111503547267078E-01
30	0.732379744272606E-05	0.279892154309547E-04
	0.110044700457775E-03	0.217365526502542E-03
	0.546918326183967E-03	0.720703586534390E-03
	0.170185751910164E-02	0.167446096505499E-02
	0.408386360971437E-02	0.319128240641147E-02
	0.830004117688234E-02	0.535378831352934E-02
	0.150229781560800E-01	0.820962136808196E-02
	0.249539236157546E-01	0.117680292130848E-01
	0.387833861710629E-01	0.159981435048914E-01
	0.571508984811764E-01	0.208290410936243E-01
	0.806057414726552E-01	0.261515976613093E-01
	0.109570394234518E+00	0.318220682694564E-01
	0.144308373001501E+00	0.376672559998067E-01
	0.184897949427532E+00	0.434910622632234E-01
	0.231213001548255E+00	0.490821532284030E-01
	0.282911960197208E+00	0.542224286612626E-01
	0.339435481190853E+00	0.586959442909769E-01
	0.400013113109450E+00	0.622979181149365E-01
	0.463678856939306E+00	0.648434457072330E-01
	0.529295142741036E+00	0.661755598512544E-01
	0.595584395325645E+00	0.661722952772005E-01
	0.661167040396915E+00	0.647524589229209E-01
	0.724604528176033E+00	0.618798583499546E-01
	0.784445734734942E+00	0.575658036634420E-01
	0.839274951478663E+00	0.518697691924657E-01
	0.887759597617343E+00	0.448981784955672E-01
	0.928695795963228E+00	0.368013625003695E-01
	0.961050059187410E+00	0.277688703774136E-01
	0.983995703521289E+00	0.180238737841607E-01
	0.996945958679763E+00	0.782767019549676E-02

N	Nodes x_i	Weights w_i
35	0.401098093701052E-05	0.153323972344389E-04
	0.603496493961345E-04	0.119314925936177E-03
	0.300616431782387E-03	0.396956727600450E-03
	0.938413243142584E-03	0.926719742189694E-03
	0.226110954001356E-02	0.177722277468937E-02
	0.461861900484087E-02	0.300450660226082E-02
	0.840960506728443E-02	0.464967479805987E-02
	0.140655627785534E-01	0.673701850949898E-02
	0.220332949959123E-01	0.927268937587387E-02
	0.327563552513284E-01	0.122439660322322E-01
	0.466560658935003E-01	0.156191410212273E-01
	0.641127372512586E-01	0.193480355175793E-01
	0.854477111324605E-01	0.233631296006668E-01
	0.110906829994115E+00	0.275812764915292E-01
	0.140645892246187E+00	0.319059507976849E-01
	0.174718595617135E+00	0.362299639512362E-01
	0.213067396094360E+00	0.404385652051307E-01
	0.255517621930679E+00	0.444128342291967E-01
	0.301775083274460E+00	0.480332619050301E-01
	0.351427311172301E+00	0.511834096518771E-01
	0.403948448291627E+00	0.537535347215710E-01
	0.458707701144855E+00	0.556440694728935E-01
	0.514981153335511E+00	0.567688466612984E-01
	0.571966634792308E+00	0.570579701395159E-01
	0.628801246341147E+00	0.564602408366056E-01
	0.684581055249250E+00	0.549450611514993E-01
	0.738382408163506E+00	0.525037565635151E-01
	0.789284255328327E+00	0.491502708662106E-01
	0.836390845802486E+00	0.449212104911785E-01
	0.878854138847408E+00	0.398752335254109E-01
	0.915895282821554E+00	0.340918004630718E-01
	0.946824542300002E+00	0.276693300400168E-01
	0.971059120835061E+00	0.207228620420109E-01
	0.988138551329524E+00	0.133817082903205E-01
	0.997739722313237E+00	0.579513447814728E-02
40	0.237684143879143E-05	0.908716648479552E-05
	0.357941198659536E-04	0.708096570276615E-04
	0.178564191524854E-03	0.236107924249676E-03
	0.558572352357276E-03	0.552941324982586E-03
	0.134949361265519E-02	0.106471907669186E-02
	0.276558618788719E-02	0.180898827559081E-02
	0.505522381417743E-02	0.281624906827513E-02
	0.849332816182978E-02	0.410894500639753E-02
	0.133728905726761E-01	0.570065525023859E-02
	0.199957958342846E-01	0.759550947534350E-02
	0.286631788855304E-01	0.978784093517871E-02
	0.396655579573064E-01	0.122620873954733E-01
	0.532729943956627E-01	0.149929436836712E-01
	0.697255301257928E-01	0.179457635281876E-01
	0.892241483693333E-01	0.210772023407832E-01
	0.111922491943615E+00	0.243360867646222E-01
	0.137919556510407E+00	0.276644913094057E-01
	0.167253553899293E+00	0.309989973545750E-01
	0.199897113629064E+00	0.342721053423017E-01
	0.235753959625130E+00	0.374137672104434E-01
	0.274657164616264E+00	0.403530031221956E-01
	0.316369047605705E+00	0.430195644057483E-01
	0.360582741027397E+00	0.453456033756603E-01
	0.406925414636825E+00	0.472673103983455E-01
	0.454963103785795E+00	0.487264791953468E-01
	0.504207051427098E+00	0.496719629360194E-01
	0.554121436898801E+00	0.500609861199254E-01
	0.604132331112845E+00	0.498602805316812E-01
	0.653637688015750E+00	0.490470175905769E-01
	0.702018156811904E+00	0.476095141198307E-01
	0.748648479051785E+00	0.455476938170290E-01
	0.792909219783803E+00	0.428732923992053E-01
	0.834198572923854E+00	0.396098004030230E-01
	0.871943978073835E+00	0.357921438452470E-01
	0.905613289425416E+00	0.314661094009314E-01
	0.934725247515507E+00	0.266875279291532E-01
	0.958859023272801E+00	0.215124114456 73E-01
	0.977662642064842E+00	0.160399123812805E-01
	0.990860246620079E+00	0.103230260805101E-01
	0.998259972471242E+00	0.446223271379884E-02

n: 5

0.046910077030668018	0.1184634425280945
0.23076534494715845	0.23931433524968324
0.5	0.28444444444444444
0.7692346550528415	0.23931433524968324
0.95308992296933193	0.1184634425280945

n: 10

0.013046735741414128	0.03333672154342931
0.067468316655507732	0.074725674575290266
0.16029521585048778	0.10954318125799108
0.28330230293537639	0.13463335965499812
0.42556283050918442	0.14776211235737646
0.57443716949081558	0.14776211235737646
0.71669769706462361	0.13463335965499812
0.83970478414951222	0.10954318125799108
0.93253168334449232	0.074725674575290266
0.98695326425858587	0.03333672154342931

n: 20

0.0034357004074525577	0.0088070035695757155
0.018014036361043095	0.020300714900193525
0.043882785874337027	0.03133602416705452
0.080441514088890553	0.041638370788352377
0.1268340467699246	0.050965059908620242
0.18197315963674249	0.059097265980759157
0.24456649902458644	0.065844319224588291
0.31314695564229023	0.071048054659190965
0.38610707442917747	0.074586493236301912
0.46173673943325133	0.076376693565362988
0.53826326056674867	0.076376693565362988
0.61389292557082253	0.074586493236301912
0.68685304435770977	0.071048054659190965
0.75543350097541362	0.065844319224588291
0.81802684036325757	0.059097265980759157
0.8731659532300754	0.050965059908620242
0.91955848591110945	0.041638370788352377
0.95611721412566297	0.03133602416705452
0.98198596363895696	0.020300714900193525
0.99656429959254744	0.0088070035695757155

Quadrature rules for integrating polynomials through order 2n on the interval [0,1]

(algorithm from Numerical Recipes)

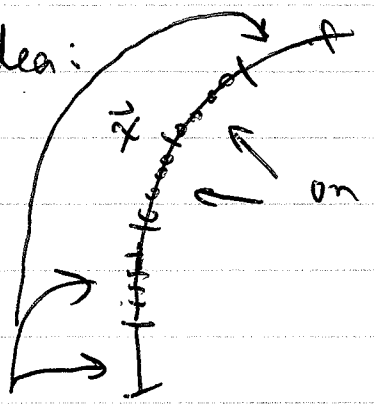
n: 50

0.00056679778996449048	0.001454311276577494
0.0029840152839546441	0.0033798995978727396
0.0073229579759970798	0.0052952741918255233
0.013567807446653979	0.007190411380742787
0.021694522378596037	0.0090577803567446682
0.031671690527561025	0.010890121585062399
0.043460721672104075	0.012680336785006156
0.057016010238193471	0.014421496790267576
0.072285115285026957	0.016106864111788983
0.089208964570332006	0.017729917807573041
0.1077220835498004	0.019284378306293839
0.12775284888696575	0.02076423154507382
0.1492237656465889	0.02216375216940164
0.17205176715728032	0.02347752565197422
0.19614853640752489	0.02470046922473319
0.22142084774267495	0.025827851534790579
0.24777092754626789	0.026855310944498136
0.27509683251298062	0.02777887240310627
0.30329284405121743	0.02859496282386419
0.33224987729028133	0.029300424906611226
0.36185590311023397	0.02989252935213273
0.39199638156197913	0.030368985420885106
0.42255470500092707	0.030727949795158378
0.45341264921995694	0.030968033710341607
0.48445083083640555	0.031088308327673654
0.51554916916359439	0.031088308327673654
0.54658735078004306	0.030968033710341607
0.57744529499907293	0.030727949795158378
0.60800361843802087	0.030368985420885106
0.63814409688976603	0.02989252935213273
0.66775012270971867	0.029300424906611226
0.69670715594878252	0.02859496282386419
0.72490316748701944	0.02777887240310627
0.75222907245373216	0.026855310944498136
0.7785791522573251	0.025827851534790579
0.80385146359247517	0.02470046922473319
0.82794823284271968	0.02347752565197422
0.8507762343534111	0.02216375216940164
0.8722471511130343	0.02076423154507382
0.8922779164501996	0.019284378306293839
0.91079103542966799	0.017729917807573041
0.92771488471497299	0.016106864111788983
0.94298398976180653	0.014421496790267576
0.95653927832789587	0.012680336785006156
0.96832830947243898	0.010890121585062399
0.97830547762140396	0.0090577803567446682
0.98643219255334602	0.007190411380742787
0.99267704202400298	0.0052952741918255233
0.99701598471604536	0.0033798995978727396
0.99943320221003551	0.001454311276577494

But this time we don't know u on the boundary.

→ need a special quadrature rule to handle the logarithmic singularity.

idea:



on adjacent segments, use Ma/Rokhlin/Wandzura scheme. (integrates $1, \log x, x, x\log x, \ldots, x^n \log x$ exactly from 0 to 1)

on all other segments, use classical Gaussian quadrature. (integrates $1, x, x^2, \ldots, x^{2n}$ exactly from 0 to 1)

note: in the numerical example above, the solution differs from the one I started with by a constant. A different solution of $B^t\phi = g$ gives a different constant.

=

Exterior problems and multiply connected domains.

the equation $B\phi = g$, $B = \frac{1}{2}I - K$ corresponds to the exterior Dirichlet problem. Like B^t, B is not invertible, so $B\phi = g$ has a solution iff $(g, \phi_0) = 0$ whenever $B^t\phi_0 = 0$.

But this time there is a solution of the PDE for every g, it just happens not to be representable by the double layer potential unless $(g, \phi_0) = 0$.

example:  exterior problem

problem: no way to represent $u \equiv$ const outside Ω

freedom: $\phi =$ const is mapped to $u \equiv 0$ outside Ω

solution: Represent u via a modified potential:

$$u(\vec{x}) = \int_\Gamma \left(-\frac{\partial N}{\partial n_\xi}(\vec{x}, \vec{\xi}) + 1 \right) \phi(\vec{\xi})\, d\xi$$

↑ any constant $\neq 0$ will do.

example:  multiply-connected domain.

problem: no way to represent a soln with different constant values on $\Gamma_0, \Gamma_1, \Gamma_2$ using double-layers alone.

freedom: $\phi =$ const on Γ_1, Γ_2 are mapped to $u \equiv 0$ in Ω.

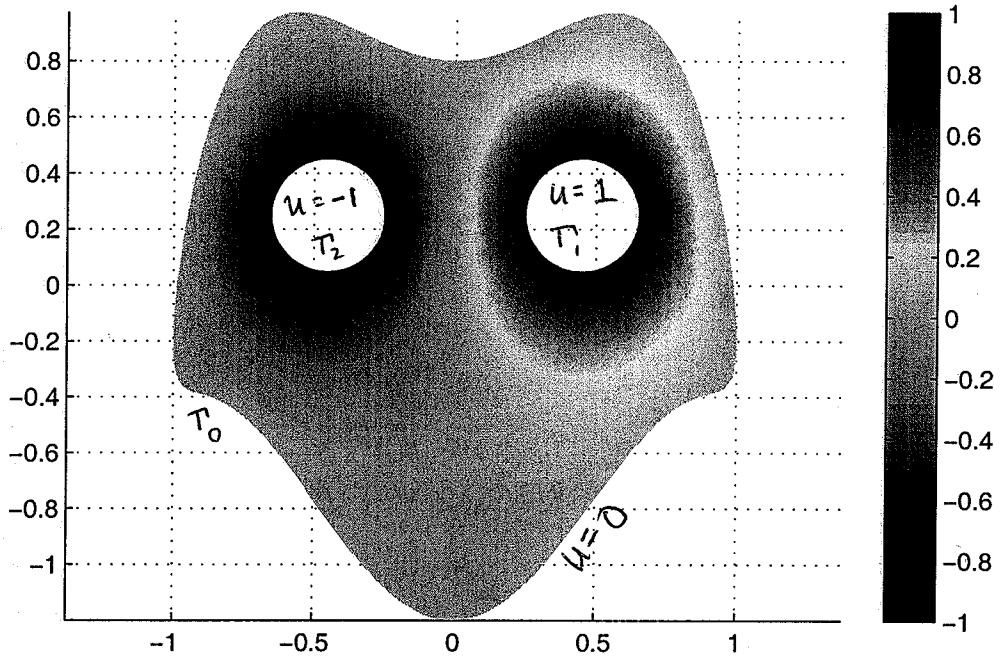
solution: represent u via

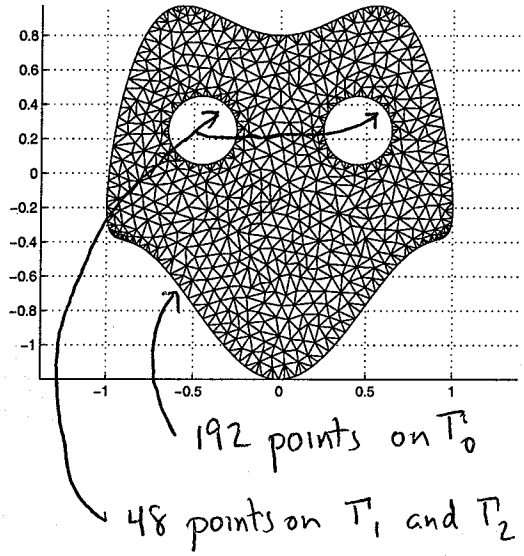
$$u(\vec{x}) = \int_{\Gamma_0} -\frac{\partial N}{\partial n_\xi}(\vec{x}, \vec{\xi}) \phi(\vec{\xi})\, d\xi + \sum_{j=1}^{2} \int_{\Gamma_j} \left[-\frac{\partial N}{\partial n_\xi}(\vec{x}, \vec{\xi}) + \log\left(|\vec{x} - \vec{a}_j|^2 \right) \right] \phi(\vec{\xi})\, d\xi$$

↑ $\vec{a}_j$ inside Γ_j (so $\vec{a}_j \notin \Omega$)

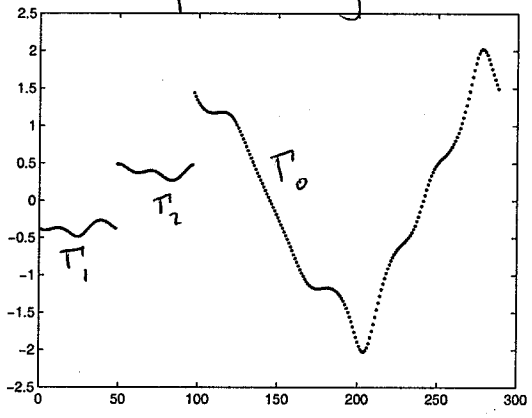
it's not hard to show that this modified representation leads to a uniquely solvable integral equation $\tilde{B}\phi = g$.

double layer potential augmented with two harmonic functions





dipole density



-0.52065832197224	0.49999990309552	-4.28583284359856	0.49999982962007
-0.51060351506924	0.49999992225624	-3.65651656593380	0.49999988756418
-0.50030108744344	0.49999994251571	-0.51076284922798	0.49999994220995
-0.50010932959409	0.49999996493349	-0.50842278110165	0.49999995767616
-0.50000283340495	0.49999996819157	-0.50014472938116	0.49999996679213
-0.50000084840917	0.49999999110907	-0.50010350140679	0.49999998238959
-0.50000002026557	0.49999999190111	-0.50000138102249	0.49999999145038
-0.50000000633295	0.49999999354090	-0.50000081372182	0.49999999339758
-0.50000000009103	0.49999999659317	-0.50000000947750	0.49999999571664
-0.50000000002736	0.49999999778149	-0.50000000608233	0.49999999731344
-0.49999999999901	0.49999999926190	-0.50000000007330	0.49999999901562
-0.49999999999848	0.49999999967449	-0.50000000000922	0.49999999960704
-0.49999999998825	0.49999999997794	-0.49999999999847	0.49999999986208
-0.49999999997275	0.49999999999699	-0.49999999999810	0.49999999989614
-0.49999999994549	0.49999999999931	-0.49999999998823	0.49999999997801
-0.49999999992197	0.49999999999955	-0.49999999997270	0.49999999999931
-0.49999999982405	0.50000000000048	-0.49999999993421	0.49999999999949
-0.49999999974822	0.50000000001275	-0.49999999987338	0.50000000000060
-0.49999999827031	0.50000000004927	-0.49999999982405	0.50000000000534
-0.49999999804568	0.50000000016840	-0.49999999974781	0.50000000002474
-0.49999999790356	0.50000000022631	-0.49999999826870	0.50000000021540
-0.49999999595736	0.50000000121400	-0.49999999803905	0.50000000063025
-0.49999998556914	0.50000000331034	-0.49999999641256	0.50000000136319
-0.49999998408056	0.50000000541433	-0.49999999594357	0.50000000377700
-0.49999993126937	0.50000000813788	-0.49999998556314	0.50000000772632
-0.49999987522313	0.50000001052260	-0.49999998407925	0.50000000812450
-0.49999987402098	0.50000002975986	-0.49999988599892	0.50000002025255
-0.49999983888815	0.50000003749383	-0.49999987503754	0.50000002312035
-0.49999898020868	0.50000013464991	-0.49999987374142	0.50000005767542
-0.49999894575449	0.50000019981830	-0.49999983382581	0.50000019910970
-0.49999751034814	0.50000031982831	-0.49999897984240	0.50000027899333
-0.49999369422765	0.50000044307331	-0.49999894553136	0.50000041240989
-0.49999145029870	0.50000044583621	-0.49999585502320	0.50000043278444
-0.49999126034474	0.50000163319496	-0.49999347385503	0.50000089146267
-0.49992896537819	0.50000279373019	-0.49999145013783	0.50000259318914
-0.49992762390932	0.50000444303690	-0.49999119889463	0.50000375049260
-0.49988686452835	0.50000528908883	-0.49992896537728	0.50000528663153
-0.49975866035564	0.50000616572263	-0.49992728927142	0.50000546135725
-0.49939617262992	0.50001662685070	-0.49981069693233	0.50001167012446
-0.49939343429651	0.50002422850653	-0.49974182418655	0.50002256071501
-0.49506159973799	0.50004848577407	-0.49939512092949	0.50003595916363
-0.49497558301096	0.50008447273129	-0.49937825385771	0.50004016688524
-0.49394582092970	0.50012955447815	-0.49499527710867	0.50010719935498
-0.49102118694073	0.50018319320583	-0.49482347386717	0.50017371661676
-0.46351455376776	0.50022816238941	-0.49123455260462	0.50020524348073
0.44843384112131	0.50054543579346	-0.48657012470465	0.50034947246574
0.00000000003497	0.50104776112123	-0.46321560324892	0.50044713047744
0.00000000004863	0.50157336534512	-0.44186443553013	0.50136072171949
0.32385819457219	0.50221045102385	0.31685379350958	0.50182423063333
0.35626859748955	0.50335868575391	0.31836402539314	0.50293765506613
0.37563476293734	0.50678982010713	0.36148885792176	0.50434489343459
0.39206569598309	0.51190725286231	0.37255927325793	0.50697599655674
0.44605873917311	0.51464436110477	0.44300911582767	0.51403438005759
0.47691106813361	0.51700825779264	0.46191990545019	0.51449675621348
0.48360073643116	0.54860431362837	0.47475802716952	0.52764036930954
0.48370402240970	0.58871487660325	0.48346506761819	0.58155277737920
0.48849023224655	0.61748184953063	0.48514447187992	0.60696123804056
0.49431292496145	0.64140563963200	0.49211453326806	0.62596277909663
0.49552116856581	0.65249214828552	0.49409790683052	0.63473606054158
0.49789535830500	0.99999999997928	0.49755396200650	1.02390151017872
0.49801106799247		0.49801106604955	
0.49883351564860		0.49851517577007	
0.49917947131805		0.49882718953165	
0.49930545886573		0.49928594076781	
0.49961253860413		0.49956869785231	
0.49964543608412		0.49964417818438	
0.49986427894321		0.49983491194059	
0.49987619914115		0.49985582811136	
0.49991079449209		0.49991067030452	
0.49993258918616		0.49992969001240	
0.49995463119155		0.49995089750524	
0.49997426454393		0.49997370873858	
0.49998349652612		0.49998260000074	
0.49998997064212		0.49998858555835	
0.49999225010889		0.49999223184703	
0.49999491873851		0.49999405861489	
0.49999501034996		0.49999487075742	
0.49999811012097		0.49999761059282	
0.49999872712242		0.49999849237269	
0.49999899495194		0.49999876996837	
0.49999928531623		0.49999906542560	
0.49999949807176		0.49999948845896	
0.49999973916466		0.49999973513951	
0.49999975066667		0.49999973750026	

eigenvalues of A

← double layer potential

Same, but augmented with two harmonic functions →

not singular now

A is nearly singular (here 24 points were used on Γ_1, Γ_2 and 96 were used on Γ_0)

When 48,48,192 are used, A is exactly singular with 2-dim kernel

References

Fourier Series

Dym and McKean, Fourier Series and Integrals
T. W. Korner, Fourier Analysis
Ivar Stakgold, Green's Functions and Boundary Value Problems
Numerical Recipes (often a good place to start)
Any book on signal processing (look up aliasing)

Potential Theory

O. D. Kellog, Foundations of Potential Theory
(a very old but classic text)
Paul Garabedian, Partial Differential Equations
S. L. Sobolev, Partial Differential Equations of Mathematical Physics
Gerald Folland, Partial Differential Equations

Boundary Integral and Boundary Element Methods

Rainer Kress, Linear Integral Equations
(very mathematical -- Fredholm theory, compact operators, etc.
this is the book for mathematicians)

C. Pozrikidis, Boundary integral and singularity methods for
linearized viscous flow
(excellent engineering book -- but only covers Stokes flow)

Marc Bonnet, Boundary Integral Equation Methods for Solids and Fluids
(this book isn't great, but it's pretty complete and doesn't require
too much math background, unlike Kress's book above)