Derivatives!

- Much of engineering (and Transport!) deals with how things change w/ position or time

- Captured by the derivative
- Suppose we have a function

$$y = f(x)$$

$$\frac{dy}{dx} \equiv \frac{df}{dx}$$

→ the local slope of the tangent to f(x)!

We also need higher derivatives:

$$\frac{d^2 f}{dx^2} \equiv \frac{d}{dx}\left(\frac{df}{dx}\right)$$

Note the units: $\left[\frac{d^2 f}{dx^2}\right] \equiv \frac{[f]}{[x]^2}$

not $\frac{[f]^2}{[x]^2}$!!!

Also note: $f'' < 0 \equiv$ concave down
$f'' > 0 \equiv$ concave up
$f'' = 0 \equiv$ inflection point!

You should all know common derivatives.

$$\frac{d}{dx}\left(x^{a}\right) = a\,x^{a-1}$$

$$\frac{d}{dx}\left(\ln x\right) = \frac{1}{x}$$

$$\frac{d}{dx}\left(e^{ax}\right) = a\,e^{ax}$$

$$\frac{d}{dx}\left(\sin x\right) = \cos x$$

$$\frac{d}{dx}\left(\cos x\right) = -\sin x$$

etc.....

Often we need to do chain rule differentiation:

Say $y = f(g(x))$ where f & g are functions

$$\therefore \frac{dy}{dx} = f' \frac{dg}{dx} \quad \text{where } f' \equiv \frac{df}{dg}$$

Let's look at an example:

$$f = e^{ax^2} \quad \therefore \; f = e^{ag}, \; g = x^2$$

$$\text{so } \frac{df}{dx} = \left(a e^{ax^2}\right)(2x)$$

$$= 2ax e^{ax^2}$$

This is similar to differentiation by parts

Say $y = f(x)\,g(x)$

$$\therefore \frac{dy}{dx} = \frac{df}{dx} g + \frac{dg}{dx} f$$

$$\text{So: } \frac{d}{dx}(\tan x) = \frac{d}{dx}\left(\frac{\overset{f(x)}{\sin x}}{\underset{g(x)}{\cos x}}\right)$$

$$= \frac{\overset{f'}{\cos x}}{\underset{g}{\cos x}} + \overset{f}{\sin x}\,\underbrace{\left(\frac{-1}{(\cos x)^2}\right)}_{\text{deriv of } \frac{1}{g} \text{ w.r.t. } g}\underbrace{(-\sin x)}_{g'}$$

$$= 1 + \frac{\sin^2 x}{\cos^2 x} = \frac{\cos^2 x + \sin^2 x}{\cos^2 x}$$

$$= \frac{1}{\cos^2 x} = \sec^2 x$$

We also have partial derivatives for functions of more than one indep. variable!

Let's take $h = f(x, y)$ (say, heights as fn position)

$$\therefore \frac{\partial h}{\partial x} \equiv \frac{\partial f}{\partial x}$$

$$\frac{\partial h}{\partial y} \equiv \frac{\partial f}{\partial y}$$

Again, these are slopes of a tangent - but in the x & y direction!

Say $h = x^2 + y^2$

$$\therefore \frac{\partial h}{\partial x} = 2x, \quad \frac{\partial h}{\partial y} = 2y$$

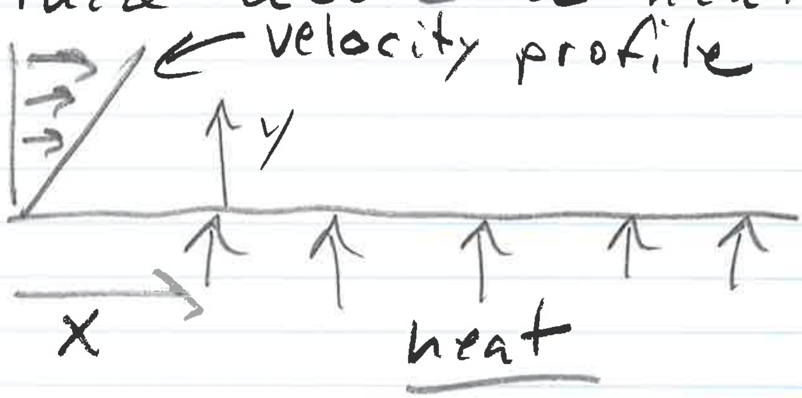
We hold one variable constant while taking the derivative w/ respect to the other!

Suppose $h = x^2 y^3$

Then $\dfrac{\partial h}{\partial x} = 2x\,y^3$

$$\frac{\partial h}{\partial y} = 3x^2 y^2$$

All this is often used together in transport! A classic problem: the temperature distribution in a fluid above a heated plane:



We find $T = x^{1/3} f(\eta)$; $\eta = \dfrac{y}{x^{1/3}}$

What is the derivative $\frac{\partial T}{\partial y}$??
(This is proportional to the heat flux)

const. w.r.t. y

$$\frac{\partial T}{\partial y} = \frac{\partial}{\partial y}\left(x^{1/3} f(\eta)\right) = x^{1/3}\frac{\partial f}{\partial y}$$

$$= x^{1/3}\frac{df}{d\eta}\frac{\partial \eta}{\partial y} = x^{1/3}\frac{df}{d\eta}\frac{1}{x^{1/3}}$$

chain rule

$$= \underline{f'}$$

we also need (for this problem)

$$\frac{\partial T}{\partial x} = \frac{\partial}{\partial x}\left(x^{1/3} f(\eta)\right) = \underbrace{\frac{1}{3}x^{-2/3} f + x^{1/3}\frac{\partial f}{\partial x}}_{\text{dif. by parts}}$$

$$= \frac{1}{3}x^{-2/3} f + x^{1/3}\frac{df}{d\eta}\frac{\partial \eta}{\partial x}$$

$$= -\frac{1}{3}\frac{y}{x^{4/3}}$$

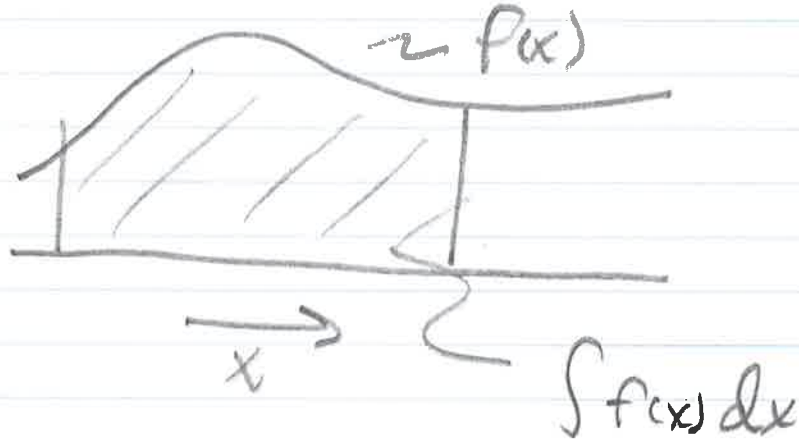
$$= \frac{1}{3}x^{-2/3} f - \frac{1}{3}\frac{y}{x} f'$$

$$= x^{-2/3}\eta$$

$$= \frac{1}{3}x^{-2/3}\left(f - \eta f'\right)$$

Integrals!

- This is the "anti-Derivative" inverse operation
- In one-d it's the area under the curve!



we have Definite integrals where we specify the limits (yields a const)

and indefinite integrals where it is a function of (usually) upper limit.

If neither upper or lower bound is specified we get a constant of integration

This is just the value of the integral at some point - usually the lower limit of integration.

So: $\int_0^1 x\,dx = \frac{1}{2}x^2\Big|_0^1 = \frac{1}{2}$ (definite)

$\int_0^x x'\,dx' = \frac{1}{2}x'^2\Big|_0^x = \frac{1}{2}x^2 - \frac{1}{2}(0) = \frac{1}{2}x^2$

dummy variable

$\int x\,dx = \frac{1}{2}x^2 + c$ (const. of int.)

Some integrals you should <u>know</u>:

$\int e^{ax}\,dx = \frac{1}{a}e^{ax}$ $(+c)$

$\int \sin x\,dx = -\cos x$

$\int x^a\,dx = \frac{1}{a+1}x^{a+1}$ $(a \neq -1)$

$\int \frac{1}{x}\,dx = \ln x$

etc.

Often you can figure out an integral by guessing from the derivative.

Suppose we want $\int \ln x \, dx$

We <u>expect</u> this to be related to $x \ln x$ (inc. power of x)

so: $\frac{d}{dx}(x \ln x) = \ln x + \frac{x}{x}$ $\; {}^{\prime\prime} 1$ (by parts)

so $\ln x = \frac{d}{dx}(x \ln x) - 1$

or $\int \ln x \, dx = x \ln x - \int dx$

$= x \ln x - x$

In general, we get this by integration by parts:

$$\frac{d(f(x)g(x))}{dx} = g\frac{df}{dx} + f\frac{dg}{dx}$$

$$\therefore \quad fg = \int g\frac{df}{dx}\,dx + \int f\frac{dg}{dx}\,dx$$

or

$$\int f g' dx = fg - \int g f' dx$$

This sort of integration by parts is useful in deriving transport relationships such as the minimum dissipation theorem (which we'll use) and the von Karman integral momentum balance (which we'll derive).

The analog to partial differentiation is multiple integration, usually over a surface or volume. Suppose we have some concentration distribution (mass/volume) $\phi(x, y, z)$ inside a rectangular prism volume of

$-a < x < a$, $-b < y < b$, $-c < z < c$

The total amount in the volume

is $I = \int_D \phi \, dV \equiv \int_{-a}^{a}\int_{-b}^{b}\int_{-c}^{c} \phi \, dz\, dy\, dx$

often shapes are more complicated! suppose instead the shape is a rectangular pyramid of height h:

$$0 < z < h, \quad -a\left(1-\frac{z}{h}\right) < x < a\left(1-\frac{z}{h}\right)$$

$$-b\left(1-\frac{z}{h}\right) < y < b\left(1-\frac{z}{h}\right)$$

In this example you integrate over x & y first as the limits leave a $f^n(z)$. Then you finish it off w/ z!

If $\phi = 1$ (uniform conc.) you get the volume of a pyramid is just $\frac{4}{3}abh$ (e.g., $\frac{1}{3}$ base area $\times$ height)

$$\equiv \int_0^h \int_{-a(1-\frac{z}{h})}^{a(1-\frac{z}{h})} \int_{-b(1-\frac{z}{h})}^{b(1-\frac{z}{h})} 1 \, dy\, dx\, dz$$

The Taylor Series

A *very* useful mathematical technique is the Taylor Series:

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + \ldots$$

$$\equiv \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

This *requires* $f(x)$ to be continuous & differentiable.

If $x_0 \equiv 0$ it's called the Maclaurin Series.

Effectively it's a polynomial expansion of a function about x_0.

We *often* truncate after the first (extra) term to *linearize* a problem.

Suppose we are looking at the motion of a pendulum. The restoring force is prop. to $-\sin\theta$ (where θ is the disp.)

Because $\sin\theta$ is non-linear in θ, this is inconvenient. We can linearize using T.S.:

expand about $\theta = 0$:

$$\sin\theta = \sin(0) + \cos(0)(\theta) - \frac{\sin(0)}{2}\theta^2 - \frac{\cos(0)}{3!}\theta^3 + \ldots$$

but $\sin(0) = 0$, $\cos(0) = 1$

$$\text{so} \quad \sin\theta = \theta - \frac{\theta^3}{6} + \frac{\theta^5}{5!} + \ldots$$

$$\approx \theta + O(\theta^3)$$

which makes analyzing a pendulum much easier!

we can truncate after any term by not specifying where the last deriv. is evaluated:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi)(x - x_0)^2$$

where $x_0 < \xi < x$.

There are a few series expansions you should know:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \ldots$$

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \ldots$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

etc.

The last two are interesting due to a finite radius of convergence. If x is too large, the series diverges!

In both cases it fails for $|x| > 1$

In general, the ratio of the $n+1$ to n^{th} term must vanish as $n \to \infty$

For some functions it doesn't exist at all! Wikipedia has a nice summary of this. You can do this w/ more than one variable as well:

$$f(\underset{\sim}{x}) = f(\underset{\sim}{x}_0) + (\underset{\sim}{x} - \underset{\sim}{x}_0) \cdot \underset{\sim}{\nabla} f \Big|_{\underset{\sim}{x}_0}$$

$$+ \frac{1}{2} (\underset{\sim}{x} - \underset{\sim}{x}_0) \cdot \underset{\sim}{\nabla}\underset{\sim}{\nabla} f \Big|_{\underset{\sim}{x}_0} \cdot (\underset{\sim}{x} - \underset{\sim}{x}_0)^T, \text{ etc.}$$

$\underset{\sim}{\nabla}$ is the gradient operator.

The Taylor Series is useful for many things, a good example being algorithm error. Suppose you want to estimate the derivative of a function at x_0 (say, the boundary) using the values at x_0 and $x_1 \equiv x_0 + \Delta x$

$$\therefore f'(x_0) \approx \frac{f(x_1) - f(x_0)}{\Delta x}$$

What is the error?

expand $f(x)$ about x_0:

$$f(x_1) = f(x_0) + \underbrace{(x_1 - x_0)}_{\Delta x} f'(x_0) + \frac{1}{2}(x_1 - x_0)^2 f''(\xi)$$

so:

$$\frac{f(x_1) - f(x_0)}{\Delta x} = \frac{\cancel{f(x_0)} + \Delta x f'(x_0) + \frac{1}{2}(\Delta x)^2 f''(\xi) - \cancel{f(x_0)}}{\Delta x}$$

$$= f'(x_0) + \frac{1}{2}\Delta x f''(\xi)$$

algorithm error!

So our error is $O(\Delta x)$

We can do better than this if we use the points at $x_{-1} = x_0 - \Delta x$, $x_{+1} = x_0 + \Delta x$:

$$f'(x_0) \approx \frac{f(x_1) - f(x_{-1})}{2\Delta x}$$

where $f(x_1) = f(x_0) + \Delta x f'(x_0) + \frac{1}{2}\Delta x^2 f''(x_0) + \frac{1}{6}\Delta x^3 f'''(\xi)$

and

$$f(x_{-1}) = f(x_0) - \Delta x f'(x_0) + \frac{1}{2}\Delta x^2 f''(x_0) - \frac{1}{6}\Delta x^3 f'''(\xi)$$

So $$\frac{f(x_1) - f(x_{-1})}{2\Delta x} = \frac{2\Delta x f'(x_0) + 0 + \frac{1}{3}\Delta x^3 f'''(\xi)}{2\Delta x}$$

$= f'(x_0) + \frac{1}{6}\Delta x^2 f'''(\xi)$

So our error is now $O(\Delta x^2)$!

This is the difference between a forward difference algorithm ($O(\Delta x)$) and a center difference algorithm ($O(\Delta x^2)$)

A problem in using a center dif. formula at the boundary is there's no x_{-1}! We can use our Taylor series to still get $O(\Delta x^2)$ error if we have a third point x_2!

Our error was

$$\frac{f(x_1) - f(x_0)}{\Delta x} = f'(x_0) + \frac{1}{2}\Delta x f''(\xi)$$

If we estimate $f''(\xi)$ we can subtract off the error!

f'' is just the derivative of the derivative!

Thus: (if $x_2 = x_0 + 2\Delta x$)

$$f''(x_1) \approx \frac{\dfrac{f(x_2)-f(x_1)}{\Delta x} - \dfrac{f(x_1)-f(x_0)}{\Delta x}}{\Delta x}$$

center dif. est. of f' at $\frac{x_2+x_1}{2}$

center dif est of f' at $\frac{x_1+x_0}{2}$

est of f''

The error of this formula is $O(\Delta x^2)$ (at x_1).

$$\text{So: } f''(x_1) \approx \frac{f(x_2)+f(x_0)-2f(x_1)}{\Delta x^2}$$

$$\therefore f'(x_0) \approx \frac{f(x_1)-f(x_0)}{\Delta x} - \frac{\Delta x}{2}\,\frac{f(x_2)+f(x_0)-2f(x_1)}{\Delta x^2}$$

$$= \frac{1}{\Delta x}\left(2f(x_1) - \frac{3}{2}f(x_0) - \frac{1}{2}f(x_2)\right)$$

What is the error? Using TS again we get:

$$f(x_0) = f(x_0)$$
$$f(x_1) = f(x_0) + \Delta x f'(x_0) + \frac{1}{2}\Delta x^2 f''(x_0) + \frac{1}{6}\Delta x^3 f'''(\xi)$$
$$f(x_2) = f(x_0) + 2\Delta x f'(x_0) + 2\Delta x^2 f''(x_0) + \frac{4}{3}\Delta x^3 f'''(\xi)$$

So:

$$f'(x_0) \approx \frac{1}{\Delta x}\left(2f(x_1) - \frac{3}{2}f(x_0) - \frac{1}{2}f(x_2)\right)$$

$$= f'(x_0) - \frac{1}{3}\Delta x^2 f'''(\xi)$$

(after lots of cancellation ...)

This error is twice that using $f(x_1)$ & $f(x_{-1})$ – but it works at boundaries!

You use these sorts of algorithms when doing finite difference solutions to PDE's, and it's also useful in estimating derivatives from data in the lab!

First Order Linear ODE's

An equation of the form:

$$\frac{dy}{dx} + p(x)\,y = f(x)$$

appears in many problems from accounting ($p(x)$ would be, say, an inflation rate - or neg. of interest rate and $f(x)$ would be revenue) to mass balances in a CSTR.

It is very convenient that it has an exact general solution!

$$y = e^{-\int p(x)\,dx}\left[\int e^{+\int p(x)\,dx} f(x)\,dx + K\right]$$

where K is a constant det. from an IC (or the value of $y(x)$ somewhere in domain).

Let's look at a couple of examples.

Suppose someone offers to give you 1\$/day for 10 years - what is it worth?

One answer is 1\$/day × 3650 days = \$3,650 - but you aren't accounting for inflation! Currently inflation is about 8.6%/yr, or a continuous rate of $\ln(1.086) = 0.0825$

So the present value of the stream of payments is:

$$\frac{dM}{dt} = -iM + s, \quad M\Big|_{t=0} = 0$$

where $i = 0.0825/\text{yr}$ and $s = 365\ \$/\text{yr}$

Putting this in our standard form:

$x \equiv t$, $y \equiv M$, $p(x) \equiv i$, $f(x) \equiv s$

So:
$$M = e^{-\int i\,dt}\left[\int s e^{+\int i\,dt}\,dt + K\right]$$

$$= e^{-it}\left[\int s e^{it}\,dt + K\right]$$

$$= e^{-it}\left[\frac{s}{i}e^{it} + K\right]$$

$$= \frac{s}{i} + K e^{-it}$$

Now $M(0) = 0 \quad \therefore \quad K = -\frac{s}{i}$

and $M = \frac{s}{i}\left(1 - e^{-it}\right)$

$$= \frac{365}{0.0825}\left(1 - e^{-0.825}\right)$$

$= \$2,485$, a lot less!

If inflation were <u>zero</u> you would get \$3,650.

In that case both ρ & f were constants, but in general that's not true! For cases like that, integrals get a bit messy so wolfram alpha or the equivalent are very useful!

Sometimes higher order DEs can be reduced to first order equations. An example: How large does a droplet need to be before a face shield can capture it? It's a lot like throwing something at a wall: if it's big it will hit it via inertial impaction, but if it's small enough viscosity slows it down and it never gets there! The trajectory is governed by $F = ma = m\frac{d^2x}{dt^2}$

So:

$$M\frac{d^2x}{dt^2} = -6\pi\mu a\frac{dx}{dt}$$

radius

viscous drag

viscosity (Stokes Law)

w/ $\left.\frac{dx}{dt}\right|_{t=0} = v_0$, $\left.x\right|_{t=0} = 0$

initial velocity

This is second order in x, <u>but</u> we can solve for the <u>velocity</u> as a first order eq^n!

$$v \equiv \frac{dx}{dt} \quad \therefore \quad \frac{dv}{dt} = -\frac{6\pi\mu a}{M}v$$

$$\left.v\right|_{t=0} = v_0$$

This is <u>really</u> easy! $p(x) = cst$, $f(x) = 0$

$$\therefore \quad v = v_0 e^{-\frac{6\pi\mu a}{M}t}$$

Finishing:

$$\frac{dx}{dt} = v_0 e^{-\frac{6\pi\mu a}{M}t}$$

so $x = \frac{U_0 m}{6\pi\mu a}\left(1 - e^{-\frac{6\pi\mu a}{m}t}\right)$

At long times $\Delta x = \frac{U_0 m}{6\pi\mu a}$ (max Δx)

For a face shield to work, $\Delta x \sim 3\,cm$

$U_0 \sim 1\,m/s$ (typical droplet velocity)

$m = \frac{4}{3}\pi a^3 \rho$ ← density of water

$$\therefore \quad \Delta x = \frac{U_0 \frac{4}{3}\pi a^3 \rho}{6\pi\mu a}$$

$$\text{or} \quad a = \left(\frac{9}{2}\frac{\mu\Delta x}{U_0 \rho}\right)^{1/2}$$

$$= \left(\frac{9}{2}\frac{(1.81\times10^{-4}\ g/cm\,s)(3\,cm)}{(100\ cm/s)(1\ g/cm^3)}\right)^{1/2}$$

$$= 0.0049\ cm = 49\,\mu m$$

The diameter is $2a \approx 100\,\mu m$

But typical oral drops are $\sim 25\,\mu m$!
so face shields aren't effective.

Higher Order Linear ODE's (5-1)

A linear ODE is linear in the dependent variable:

general form: $$\sum_{i=0}^{n} P_i(x) \frac{\partial^i y}{\partial x^i} = f(x)$$

$P_i(x)$: arb. $f^n(x)$ — $f(x)$: inhomog.

So: $$x^2 \frac{\partial^2 y}{\partial x^2} + x \frac{\partial y}{\partial x} + (x^2 - n^2)y = 0$$

is a linear second order homogeneous ODE. It's actually a very special equation: Bessel's Equation of order (n) which has well known solutions!

Let's look at a simpler equation:

$$\frac{\partial^2 y}{\partial x^2} + \omega^2 y = x$$

x: inhomog.

So this is a linear 2nd order inhomog. eqn.

Because it is 2^{nd} order we need 2 conditions! This could be 2 IC's : $y(0)=1$, $y'(0)=0$ or it could be BC's :

$$y(0)=1\,, \quad y(1)=0 \quad \text{(say)}$$

For an inhomogeneous eq'n it is usually convenient to break it up into a particular solution (which satisfies the inhomog.) and a homogeneous sol'n which satisfies the eq'n w/ the inhomog. removed. The solution (linear only!!) is the sum of y^p & y^h satisfying the BCs or ICs!

So : $$\frac{\partial^2 y}{\partial x^2} + \omega^2 y = x$$

$$y(0)=1\,, \quad y'(0)=0 \quad \text{(ICs)}$$

5-3

The particular sol'n by inspection is just $y^p = \frac{x}{\omega^2}$ (plug it into verify!)

so for y^h:

$$\frac{d^2 y^h}{dx^2} + \omega^2 y^h = 0$$

The solution is just sines & cosines!

$$y^h = A \sin \omega x + B \cos \omega x$$

$$\therefore y = y^h + y^p = \frac{x}{\omega^2} + A \sin \omega x + B \cos \omega x$$

$y(0) = 1 \therefore \quad 0 + 0 + B = 1$

$y'(0) = 0 \therefore \quad \frac{1}{\omega^2} + A\omega + 0 = 0$

$$A = -\frac{1}{\omega^3}$$

so $y = \frac{x}{\omega^2} - \frac{\sin \omega x}{\omega^3} + \cos \omega x$

This problem is more interesting if our inhomogeneity is periodic:

$$\frac{d^2y}{dx^2} + \omega^2 y = \sin ax$$

w/ IC $y(0) = y'(0) = 0$

Again by inspection we expect:

$y^p = C \sin ax$: Plug it in!

$$-Ca^2 \cancel{\sin ax} + C\omega^2 \cancel{\sin ax} = \cancel{\sin ax}$$

$$\therefore \quad C = \frac{1}{\omega^2 - a^2}$$

and $y^p = \dfrac{1}{\omega^2 - a^2} \sin ax$

The homogeneous sol'n is the <u>same</u>, so:

$$y = \frac{1}{\omega^2 - a^2} \sin ax + A \sin \omega x + B \cos \omega x$$

Now $y(0) = 0 \quad \therefore \quad B = 0$ (this time!)

$$y'(0) = 0 = \frac{a}{\omega^2 - a^2} + A\omega = 0$$

$$\therefore A = -\frac{a/\omega}{\omega^2 - a^2}$$

$$\text{so } y = \frac{1}{\omega^2 - a^2}\left(\sin ax - \frac{a}{\omega}\sin \omega x\right)$$

This is interesting when $a \to \omega$!
This is a driven undamped oscillator.
If the driving frequency approaches the natural frequency, it blows up! Resonance is a very big problem in engineering!

Special High Order ODEs (6-1)

Most high order ODEs are hard to solve, even if linear! There are two commonly found exceptions: const. coef. ODEs and Euler eq'ns. Let's look at these!

Const Coef ODEs:

$$\sum_{m=0}^{n} a_m \frac{d^m y}{dx^m} = 0 \quad \text{(homogeneous)}$$

Note that if it were inhomog, we'd have to get the particular solution as well!

In general, the solution to this eq'n is just an exponential!

Let $y = e^{rx}$

Plug in: $\frac{d}{dx}(e^{rx}) = re^{rx}$

$$\therefore \sum_{m=0}^{n} a_m r^m e^{rx} = 0$$

so $\sum_{m=0}^{n} a_m r^m = 0$ is an $n^{\underline{th}}$ order polynomial for all the rs!

There are n roots (if none are repeated!) so

$$y = \sum_{m=1}^{n} c_m e^{r_m x}$$

i'th root

Note that the r_m can be complex: this yields sines & cosines!

Let's look at our oscillator again:

$$\frac{d^2y}{dx^2} + \omega^2 y = 0$$

$$y = e^{rx}$$

$$\therefore r^2 \cancel{e^{rx}} + \omega^2 \cancel{e^{rx}} = 0$$

$$r^2 = -\omega^2 \qquad r = \pm\sqrt{-\omega^2} = \pm \omega i$$

So $y = c_1 e^{\omega i x} + c_2 e^{-\omega i x}$

but using the Euler formula we have the equivalent:

$$y = A \sin \omega x + B \cos \omega x$$

as before!

Let's look at the damped oscillator:

$$\frac{d^2 y}{dx^2} + y + 2\lambda \frac{dy}{dx} = 0$$

convenient damping coef.

So: $y = e^{rx}$ yields:

$$r^2 + 1 + 2\lambda r = 0$$

$$\therefore \; r = \frac{-2\lambda \pm \sqrt{4\lambda^2 - 4}}{2} = -\lambda \pm \sqrt{\lambda^2 - 1}$$

If $\lambda > 1$ we have two real roots:

$$r = -\lambda + \sqrt{\lambda^2 - 1}, \; -\lambda - \sqrt{\lambda^2 - 1}$$

which are both negative

This is an overdamped oscillator

If $\lambda < 1$ we have two complex roots:

$$r = -\lambda \pm (1-\lambda^2)^{1/2} i$$

The solution (since $e^{a+b} = e^a e^b$) is written:

$$y = Ae^{-\lambda x} \sin (1-\lambda^2)^{1/2} x + Be^{-\lambda x} \cos (1-\lambda^2)^{1/2} x$$

If $\lambda \equiv 1$ we have repeated roots!

$$r = -\lambda \equiv -1$$

The two solutions are $y = e^{-x}$, xe^{-x}

so $y = Ae^{-x} + Bxe^{-x}$

Higher order problems - such as the Taylor wiper problem we'll look at - are done the same way!

The Euler eq'n is very similar:

$$\sum_{m=0}^{n} a_m x^m \frac{d^m y}{dx^m} = 0$$

In this case, the solutions are just $y = x^r$

Plugging in you again get an n^{th} order polynomial for r

In fact, you can turn an Euler eq'n into a const. coef. eq'n w/ the transformation $x \equiv e^t$, although it gets a little messy.

Let's look at an example. In flow past a sphere you get an equation of the form:

$$x^4 \frac{d^4 y}{dx^4} - 4x^2 \frac{d^2 y}{dx^2} + 8x \frac{dy}{dx} - 8y = 0$$

So we let $y = x^r$:

$$x^4 x^{r-4}(r)(r-1)(r-2)(r-3)$$
$$-4x^2 x^{r-2}(r)(r-1)$$
$$+8x\, x^{r-1}(r) - 8x^r = 0$$

$$\therefore \quad r(r-1)(r-2)(r-3) - 4r(r-1) + 8r - 8 = 0$$

This has <u>4</u> roots:

$$r = -1, 1, 2, 4$$

$$\therefore \quad y = \frac{c_1}{x} + c_2 x + c_3 x^2 + c_4 x^4$$

In this case we have the BC's:

$$\lim_{x \to \infty} \frac{y(x)}{x^2} = 0, \quad \lim_{x \to \infty} \frac{dy/dx}{x} = 0$$

which means that <u>both</u> $c_3 = c_4 = 0$

The other BC's (at $x = 1$) are:

$$y(1) = -\frac{1}{2}, \quad \left.\frac{dy}{dx}\right|_{x=1} = -1$$

which yields $c_1 = \frac{1}{4}$, $c_2 = \frac{-3}{4}$

So:

$$y = \frac{1}{4}\frac{1}{x} - \frac{3}{4}x$$

Just as in the case of const coef. ODEs, you can get repeated roots! Suppose we have the repeated root $r = \lambda$. Then we have <u>two</u> solutions

$$y = x^{\lambda},\ x^{\lambda}\ln x$$

extra

which makes sense as we can go from one eq'n to the other from the transformation $x = e^t$.

Solving ODE's Via Transformation 7-1

- Linear 1st order, constant coef. & Euler eq'ns can be solved directly. While these are common, there are many other ODEs which arise in transport! Sometimes these can be solved via transformations of indep or dependent variables!
- Unlike the earlier examples, the approach is specific to each problem: no "one size fits all" approach.
- Let's look at some commonly seen examples!

⇒ 1st order non-linear ODEs

This is the easiest and most general example.

Suppose you squish a fluid between 2 plates w/ constant force (we'll demonstrate this!). The radius R of the fluid is governed by the non-linear 1st order eqn: depends on force, viscosity, etc.

$$\frac{dR}{dt} = \frac{\lambda}{R^7} \; ; \quad R\Big|_{t=0} = R_0$$

initial radius

R is the dependant variable, but we can invert it!

$$\frac{dt}{dR} = \frac{R^7}{\lambda} \qquad t\Big|_{R=R_0} = 0$$

This is linear in t!

$$t = \frac{R^8}{8\lambda} + C \; ; \quad C = -\frac{R_0^8}{8\lambda}$$

$$\therefore \; t = \frac{1}{8\lambda}\left(R^8 - R_0^8\right)$$

or

$$R = \left(R_0^8 + 8\lambda t\right)^{1/8}$$

There are other ways of looking at this: If we take the original problem and multiply by R^7 we can get a perfect differential:

$$R^7 \frac{dR}{dt} = \lambda$$

$$\frac{1}{8} \frac{dR^8}{dt} = \lambda$$

$$\therefore R^8 = 8\lambda t + R_0^8 \text{, etc.}$$

In fact, this works if the RHS is a <u>separable</u> <u>function</u> of R & t:

Let $\frac{dR}{dt} = f(R)\,g(t)$

$$\therefore \frac{1}{f(R)} dR = g(t)\,dt$$

$$\int \frac{1}{f(R)} dR = \int g(t)\,dt$$

which is also sometimes useful!

Perfect differentials can be used in higher order ODEs too - sometimes!

As we'll see in 356, the temperature distribution in a long fin cooling by radiation is governed by:

$$\frac{\partial^2 T}{\partial x^2} - \lambda T^4 = 0 \qquad T\Big|_{x=0} = T_0 \qquad T\Big|_{x\to\infty} = 0$$

$\frac{\partial^2 T}{\partial x^2}$ ↑ conduction along fin

λT^4 ↑ cooling due to radiation

Inverting doesn't help - but if we multiply by $\frac{\partial T}{\partial x}$ we can get a perfect differential!

$$\frac{\partial^2 T}{\partial x^2}\frac{\partial T}{\partial x} = \lambda T^4 \frac{\partial T}{\partial x}$$

$$\frac{1}{2}\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right)^2 = \frac{\lambda}{5}\frac{\partial T^5}{\partial x}$$

$$\therefore \frac{1}{2}\left(\frac{\partial T}{\partial x}\right)^2 = \frac{\lambda}{5} T^5 + cst$$

Now as $x \to \infty$ both T & $\frac{\partial T}{\partial x}$ vanish, so:

$$\left(\frac{\partial T}{\partial x}\right)^2 = \frac{2}{5}\lambda T^5 \qquad (cst = 0)$$

or $\frac{\partial T}{\partial x} = -\sqrt{\frac{2\lambda}{5}}\, T^{5/2}$ (neg root is physical)

so: $T^{-5/2}\frac{\partial T}{\partial x} = -\sqrt{\frac{2\lambda}{5}}$

$$-\frac{2}{3}\frac{\partial T^{-3/2}}{\partial x} = -\sqrt{\frac{2\lambda}{5}}$$

$$\therefore T^{-3/2} = +\frac{3}{2}\sqrt{\frac{2\lambda}{5}}\, x + cst$$

$$T\Big|_{x=0} = T_0 \quad \therefore cst = T_0^{-3/2}$$

So $T = \left(T_0^{-3/2} + \frac{3}{2}\sqrt{\frac{2\lambda}{5}}\, x\right)^{-2/3}$

which can be used to get the heat flux at the base - the rate of cooling provided by the fin. This is prop. to $\frac{\partial T}{\partial x}\Big|_{x=0}$

$$\therefore \frac{\partial T}{\partial x}\Big|_{x=0} = -\sqrt{\frac{2\lambda}{5}}\, T_0^{5/2}$$

Sometimes you can simplify DEs (and get perfect differentials) by differentiation! As an example, the unsteady heating of a slab yields (after some work!) the ODE:

$$\frac{d^2 f}{d\zeta^2} + \frac{1}{2}\zeta\frac{df}{d\zeta} - \frac{1}{2}f = 0$$

$$\left.\frac{df}{d\zeta}\right|_{\zeta=0} = -1 \quad , \quad \left. f\right|_{\zeta\to\infty} = 0$$

heat flux condition

We can solve this by rearranging and differentiation:

$$f'' = \frac{1}{2}(f - \zeta f')$$

Dif w.r.t ζ:

$$f''' = \frac{1}{2}(\cancel{f'} - \cancel{f'} - \zeta f'')$$

$$= -\frac{1}{2}\zeta f''$$

Now we can make this a perfect differential:

$$\frac{f'''}{f''} = -\frac{1}{2}\zeta$$

$$\parallel$$

$$\frac{d}{d\zeta}(\ln f'') = -\frac{1}{2}\zeta$$

So $\ln f'' = -\frac{1}{4}\zeta^2 + cst$

or $f'' = c\, e^{-\frac{1}{4}\zeta^2}$

This can just be integrated (twice) to get f' and f. Note that from the DE $f''(0) = \frac{1}{2}\left(f(0) - \underbrace{\zeta}_{0} f'(0)\right)$

$$= \frac{1}{2} f(0)$$

$$\therefore\ f'' = \frac{1}{2} f(0)\, e^{-\frac{1}{4}\zeta^2}$$

The final answer is the integral complementary error function $i\mathrm{erfc}(\zeta/2)$

A last approach which is sometimes useful is transformation of the dependent variable. This is very problem specific!

An example: suppose we have a chunk of uranium (fissile material). The neutron concentration for a spherical mass is approximated by:

$$\underbrace{\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\phi}{dr}\right)}_{\text{diffusion}} + \underbrace{\lambda^2\phi}_{\text{fission source}} + \underbrace{S}_{\text{source (radioactive)}} = 0$$

BCs: $\phi\big|_{r=0}$ = finite, $\phi\big|_{r=1} = 0$ (all escape)

dimensionless

This is a linear 2^{nd} order ODE & is inhomogeneous.

Because it is linear we can say

$$\phi = \phi_h + \phi_p \leftarrow \text{particular sol'n}$$

(8-2)

By inspection $\phi_p = -\frac{S}{\lambda^2}$

So:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \phi_h}{\partial r}\right) + \lambda^2 \phi_h = 0$$

We can solve this by transformation:

Let $\phi_h = \frac{f}{r}$

So: $\frac{\partial \phi_h}{\partial r} = \frac{\partial}{\partial r}\left(\frac{f}{r}\right) = \frac{1}{r}\frac{\partial f}{\partial r} - \frac{1}{r^2} f$

$\therefore \quad r^2 \frac{\partial \phi_h}{\partial r} = r\frac{\partial f}{\partial r} - f$

$$\frac{\partial}{\partial r}\left(r^2\frac{\partial \phi_h}{\partial r}\right) = r\frac{\partial^2 f}{\partial r^2} + \cancel{\frac{\partial f}{\partial r}} - \cancel{\frac{\partial f}{\partial r}}$$

$\therefore$ the DE is:

$$\frac{1}{r}\frac{\partial^2 f}{\partial r^2} + \lambda^2 \frac{f}{r} = 0$$

or $\frac{\partial^2 f}{\partial r^2} + \lambda^2 f = 0$!

8-3

This is a constant coef ODE and we know the solution!

$$f = A\sin\lambda r + B\cos\lambda r$$

$$\therefore \phi = A\frac{\sin\lambda r}{r} + B\frac{\cos\lambda r}{r} - \frac{S}{\lambda^2}$$

Now since $\phi\big|_{r\to 0}$ = finite, $B = 0$!

And since $\phi\big|_{r=1} = 0$ $\quad A = \frac{S}{\lambda^2 \sin\lambda}$

$$\therefore \phi = \frac{S}{\lambda^2}\left(\frac{\sin\lambda r}{r\sin\lambda} - 1\right)$$

Using the Taylor series for $\sin\lambda$, $\sin\lambda r$ we can look at the asymptote as $\lambda \to 0$:

$$\sin\lambda r = \lambda r - \frac{1}{6}(\lambda r)^3 + O(\lambda r)^5$$

$$\therefore \phi\big|_{\lambda\to 0} \cong \frac{S}{\lambda^2}\left(\frac{1-\frac{1}{6}(\lambda r)^2}{1-\frac{1}{6}\lambda^2} - 1\right) = \frac{1}{6}S(1-r^2) + O(\lambda^2)$$

higher order term

A more interesting issue is what happens for larger λ. The highest conc. is at $r=0$.

Again from a Taylor series:

$$\lim_{r \to 0} \frac{\sin \lambda r}{r} = \lambda$$

$$\therefore \phi\Big|_{r=0} = \frac{S}{\lambda^2}\left(\frac{\lambda}{\sin \lambda} - 1\right)$$

This is well behaved as $\lambda \to 0$ (keeping enough terms in the Taylor series) but when $\lambda = \pi$ $\sin \lambda = 0$ and it blows up! This is the critical mass for your sphere of uranium!

In general these tricks sometimes enable you to get an analytic sol'n to more complicated DEs. If it doesn't work, numerical sol'n is required.

Sometimes you can simplify a DE by transforming the <u>independent</u> <u>variable</u>. The conversion of Euler eq'ns to constant coef. ODEs via $x = e^t$ transform is a classic example. Indep. variable transforms are used <u>much</u> more often for PDEs - such as going from cartesian coord to cylindrical coord - as that sometimes lets you convert a PDE to an ODE - a <u>huge</u> simplification. We'll look at this later.

Numerical Solutions (9-1)

In general analytic solutions to ODEs are preferable, because you learn a lot more about the problem from the answer (e.g., the critical mass example). Often, however, (particularly for non-linear PDEs) an analytic solution either doesn't exist or is too hard to get! Thus we solve such ODEs numerically.

Most solution techniques are based on solving an ODE as a system of first order equations.

A classic example is from population dynamics: rabbits and foxes

Let $r \equiv$ rabbits, $f \equiv$ foxes we have coupled 1^{st} order equations:

$$\frac{dr}{dt} = \underset{\substack{\uparrow \\ \text{reproduction rate}}}{2r} - \underset{\substack{\uparrow \\ \text{foxes eat bunnies}}}{\alpha r f}$$

$$\frac{df}{dt} = (\underset{\substack{\uparrow \\ \text{more bunnies yields more foxes}}}{\alpha r} - 1)f$$

$\leftarrow$ foxes go away if not enough food

Let $y_1 \equiv r$, $y_2 \equiv f$

$$\therefore \begin{bmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \end{bmatrix} = \begin{bmatrix} 2y_1 - \alpha y_1 y_2 \\ (\alpha y_1 - 1) y_2 \end{bmatrix} \equiv \begin{bmatrix} g_1(t, \underset{\sim}{y}) \\ g_2(t, \underset{\sim}{y}) \end{bmatrix}$$

or $$\frac{d\underset{\sim}{y}}{dt} = \underset{\sim}{g}(t, \underset{\sim}{y})$$

This can be integrated numerically from some initial condition to demonstrate the fascinating (periodic) pop. dynamics!

This works just as well for high order ODEs! Consider the Falkner-Skan eq'n for boundary-layer flow past a wedge:

$$f''' + ff'' + \beta\left(1-(f')^2\right) = 0$$

w/ BCs: $f(0) = f'(0) = 0$, $f'(\infty) = 1$

If we have a third order eq'n, we need $\underline{\underline{3}}$ 1^{st} order equations!

Let $y_1 \equiv f$

$y_2 \equiv f'$

$y_3 \equiv f''$ ← stop here!!

So by definition $\dfrac{dy_1}{dt} = f' \equiv y_2$

$$\frac{dy_2}{dt} = f'' = y_3$$

and from the ODE:

$$\frac{dy_3}{dt} = f''' = -ff'' - \beta\left(1-(f')^2\right)$$
$$= -y_1 y_3 - \beta\left(1 - y_2^2\right)$$

So $$\frac{d\underset{\sim}{y}}{dt} = \underset{\sim}{g}(t,\underset{\sim}{y}) \equiv \begin{bmatrix} y_2 \\ y_3 \\ -y_1 y_3 - \beta(1-y_2^2) \end{bmatrix}$$

Most integrators start from an initial condition and integrate forward. We need 3 ICs:

$$y_1(0) = 0$$
$$y_2(0) = 0$$
$$y_3(0) = ??? \equiv x \quad \text{(unknown)}$$

Instead, we have a condition at the other boundary $f'(\infty) \equiv y_2(\infty) = 1$

We solve this using the shooting method. We guess $y_3(0) \equiv x$ (pick a reasonable value) and integrate to large t. We then adjust x until $y_2(\infty) - 1 = 0$. This is done automatically using a root finder! For higher order equations (e.g., natural convection from a heated wire) you may have 2 (or more) shooting parameters. This would yield some $\underset{\sim}{P}(\underset{\sim}{x}) = 0$ to satisfy the BCs on the other boundary. You solve this via multidimensional root finding - or sometimes it is more stable to minimize $\|\underset{\sim}{P}(\underset{\sim}{x})\|_2$ as an optimization problem - whatever works!

Runge-Kutta Integration

Ok, how do you solve an ODE numerically? It's all about a Taylor Series approximation!

Suppose we have the 1st order ODE:

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

We want to evaluate y at a set of discrete locations $t_0, t_1, \ldots t_n$

Our approximations are: $y_0, y_1, \ldots y_n$

The simplest approach is the Euler method:

$$y_{k+1}^{EM} = y_k^{EM} + f(t_k, y_k^{EM}) \cdot h_k$$

$$h_k \equiv t_{k+1} - t_k$$

This is just the first two terms of the Taylor Series!

What is the error in this method?

Use the Taylor Series again!

$$y(t_{k+1}) = y(t_k) + f(t_k, y(t_k))h_k + \frac{1}{2}y''(\xi)h_k^2$$

$\hookrightarrow$ local error

subtracting:

$$y_{k+1}^{EM} - y(t_{k+1}) = y_k^{EM} - y(t_k) + \left[f(t_k, y_k^{EM}) - f(t_k, y(t_k))\right]h_k - \frac{1}{2}y''(\xi)h_k^2$$

Now $f(t_k, y_k^{EM}) \approx f(t_k, y(t_k)) + \frac{\partial f}{\partial y}(y_k^{EM} - y(t_k))$

from a T-S expansion in $\underline{\underline{y}}$.

So:

amplification factor

$$y_{k+1}^{EM} - y(t_{k+1}) = \left(1 + h_k\frac{\partial f}{\partial y}\right)(y_k^{EM} - y(t_k)) - \frac{1}{2}y''(\xi)h_k^2$$

$\uparrow$ local error

So at each step we have a local error which is $O(h^2)$. Because the error is additive and the total number of steps is $O(1/h)$, the overall accuracy is $O(h)$ (first order).

The amplification factor is $|1+hJ|$ where J is the Jacobian $\frac{\partial f}{\partial y}$. This amplifies our error from previous steps! Note that if $hJ < -2$ (e.g., we have a stiff eq'n) our solution is numerically unstable, which is a problem for all explicit methods!

You can avoid this with an implicit method such as the Backward Euler Method:

$$y_{k+1}^{BE} = y_k^{BE} + f(t_{k+1}, y_{k+1}^{BE})\, h_k$$

But this requires solving a (usually non-linear) equation at every step.

The simplest extension of the EM is a 2-stage Runge Kutta rule based on Trapezoidal Rule integration:

$$y_{k+1}^{TR} \approx y_k^{TR} + \frac{1}{2}\left(f(t_k, y_k^{TR}) + f(t_{k+1}, y_{k+1}^{TR})\right)$$

This is also implicit (but more accurate and stable). Instead we use:

$$k_1 = h f(t_i, y_i)$$

$$k_2 = h f(t_{i+1}, y_i + k_1)$$

↳ EM est of y_{i+1}

and

$$y_{i+1}^{2S} = y_i^{2S} + \frac{1}{2}(k_1 + k_2)$$

This is explicit, has a local error of $O(h^3)$ and a global error of $O(h^2)$.

This can be extended to higher order - a popular 4-stage rule is:

$$k_1 = hf(t_i, y_i)$$

$$k_2 = hf(t_i + \tfrac{1}{2}h,\ y_i + \tfrac{1}{2}k_1)$$

$$k_3 = hf(t_i + \tfrac{1}{2}h,\ y_i + \tfrac{1}{2}k_2)$$

$$k_4 = hf(t_i + h,\ y_i + k_3)$$

$$y^{45}_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

which has a local error of $O(h^5)$ and an overall error of $O(h^4)$ - it's still explicit.

While R-K methods are easy to code up, it is more usual to use canned adaptive integration methods (adaptive methods change the step size h_k to satisfy error tolerance). In Matlab the codes used most often are:

ode 23 (low order, explicit, more points for plotting)

ode 45 (high order explicit)

and:
ode23s (implicit method for stiff [large, negative Jacobian] eq'ns)

All three require specification of the range of integration $[t_0, t_f]$ and initial conditions as well as the deriv. function.

If you are using the shooting method the one-d root finder in matlab is:

fzero

and the multi-dim. optimization routine is:

fminsearch

Use the "help" command to get examples.

Illustration of Numerical Solution Techniques

In this script we illustrate the numerical solution techniques of the Euler Method, the 2 stage Runge-Kutta method, the 4 stage R-K method, and an adaptive integration method built into matlab, ode23. We choose as an example an undamped forced oscillator: a problem for which there is a simple analytic solution. Our problem is:

y'' + y = sin(a*t)

y(0) = y'(0) = 0

The behavior of this oscillator depends on a, the dimensionless driving frequency. The amplitude blows up as a approaches 1, the natural frequency of the oscillator.

We solve this problem as a pair of first order equations, such that y(1) is y and y(2) is y'. The corresponding derivatives are:

ydot = @(t,y) [y(2) ; -y(1) + sin(a*t)];

With initial conditions: y0 = [0 ; 0];

We have the analytic solution to this equation:

yexact = @(t) (sin(a*t) - a* sin(t))/(1-a^2);

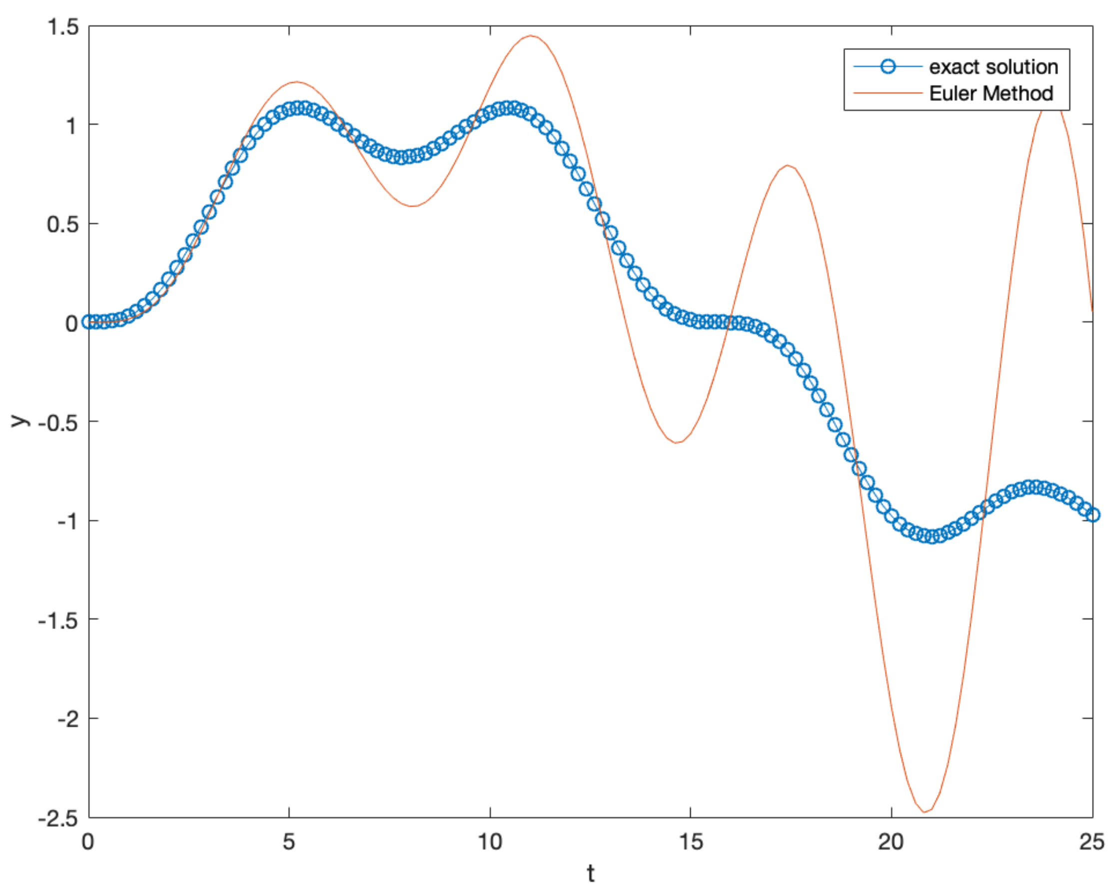
Contents

The Euler Method

We plot things up over four periods (e.g., up to 8pi). We just define the derivative and the time step. We also specify a, the frequency of the driving term.

```
a = 0.2;

% The exact solution:

yexact = @(t) (sin(a*t) - a* sin(t))/(1-a^2);
```

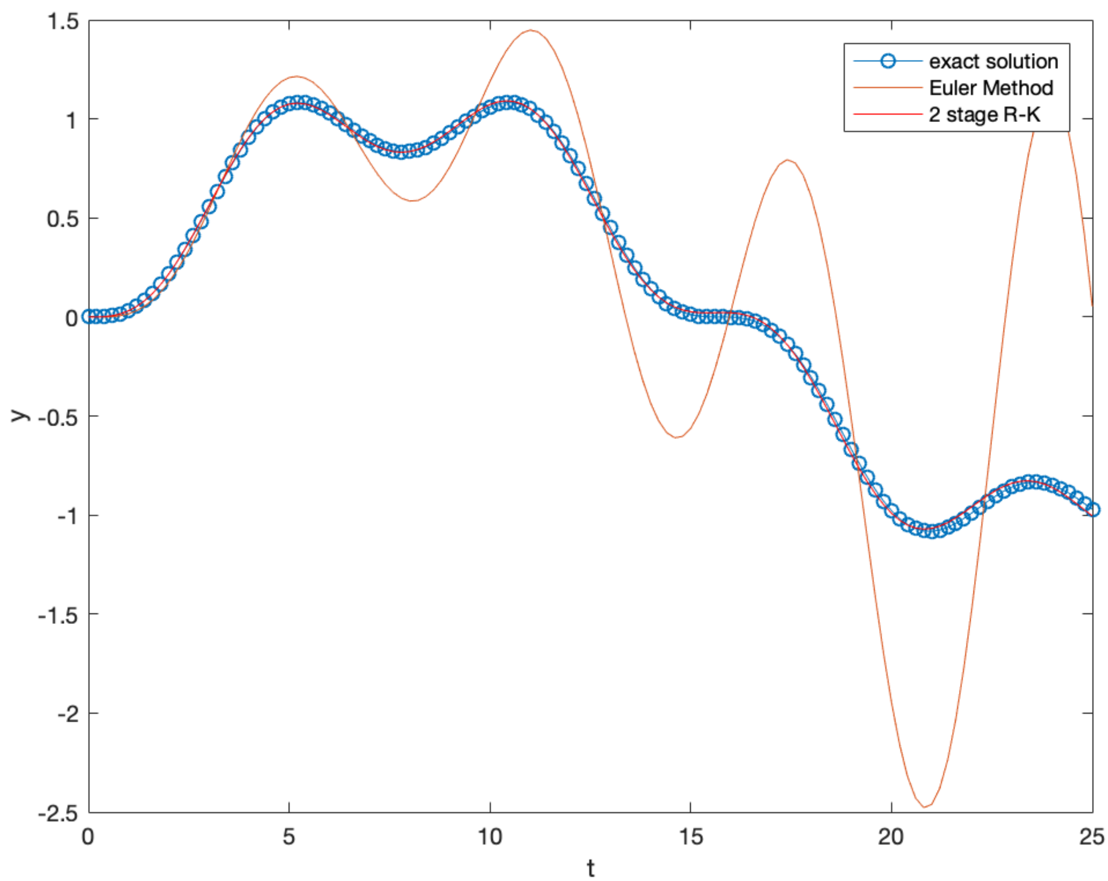
```
dt = 0.2;

tall = [0:dt:8*pi]';

y0 = zeros(2,1); % The initial condition

ydot = @(t,y) [y(2) ; -y(1) + sin(a*t)];

yem = zeros(2,length(tall)); % we keep both y1 and y2.

n = length(tall)-1; %the number of steps

for i = 1:n
    yem(:,i+1) = yem(:,i) + dt*ydot(tall(i),yem(:,i));
end

figure(1)
plot(tall,yexact(tall),'-o',tall,yem(1,:))
xlabel('t')
ylabel('y')
legend('exact solution','Euler Method')
```



The Two-Stage Runge-Kutta Technique

Very little needs to be added to the Euler method for the 2s R-K technique. The error, for this step size, is very small and is graphically indistinguishable from the exact solution.

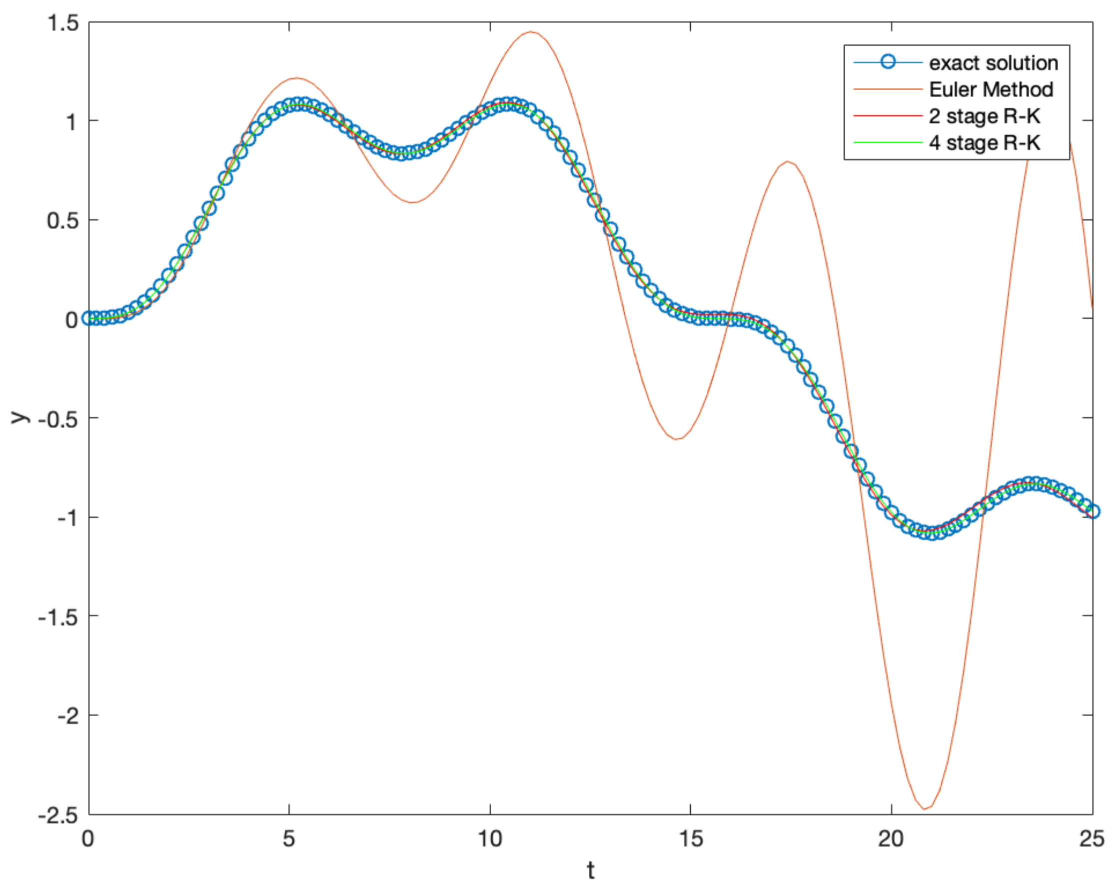
```
y2s = zeros(2,length(tall)); % we keep both y1 and y2.
for i = 1:n
    k1 = dt*ydot(tall(i),y2s(:,i));
    k2 = dt*ydot(tall(i+1),y2s(:,i)+k1);
    y2s(:,i+1) = y2s(:,i) + (k1+k2)/2;
end

figure(1)
hold on
plot(tall,y2s(1,:),'r')
hold off
legend('exact solution','Euler Method','2 stage R-K')
```



The Four-Stage Runge-Kutta Technique

This is very similar to the two stage technique, except there are four intermediate steps. The method is even more accurate.

```
y4s = zeros(2,length(tall)); % we keep both y1 and y2.
for i = 1:n
    k1 = dt*ydot(tall(i),y4s(:,i));
    k2 = dt*ydot(tall(i)+dt/2,y4s(:,i)+k1/2);
    k3 = dt*ydot(tall(i)+dt/2,y4s(:,i)+k2/2);
    k4 = dt*ydot(tall(i)+dt,y4s(:,i)+k3);
    y4s(:,i+1) = y4s(:,i) + (k1+2*k2+2*k3+k4)/6;
end

figure(1)
hold on
plot(tall,y4s(1,:),'g')
hold off
legend('exact solution','Euler Method','2 stage R-K','4 stage R-K')
```

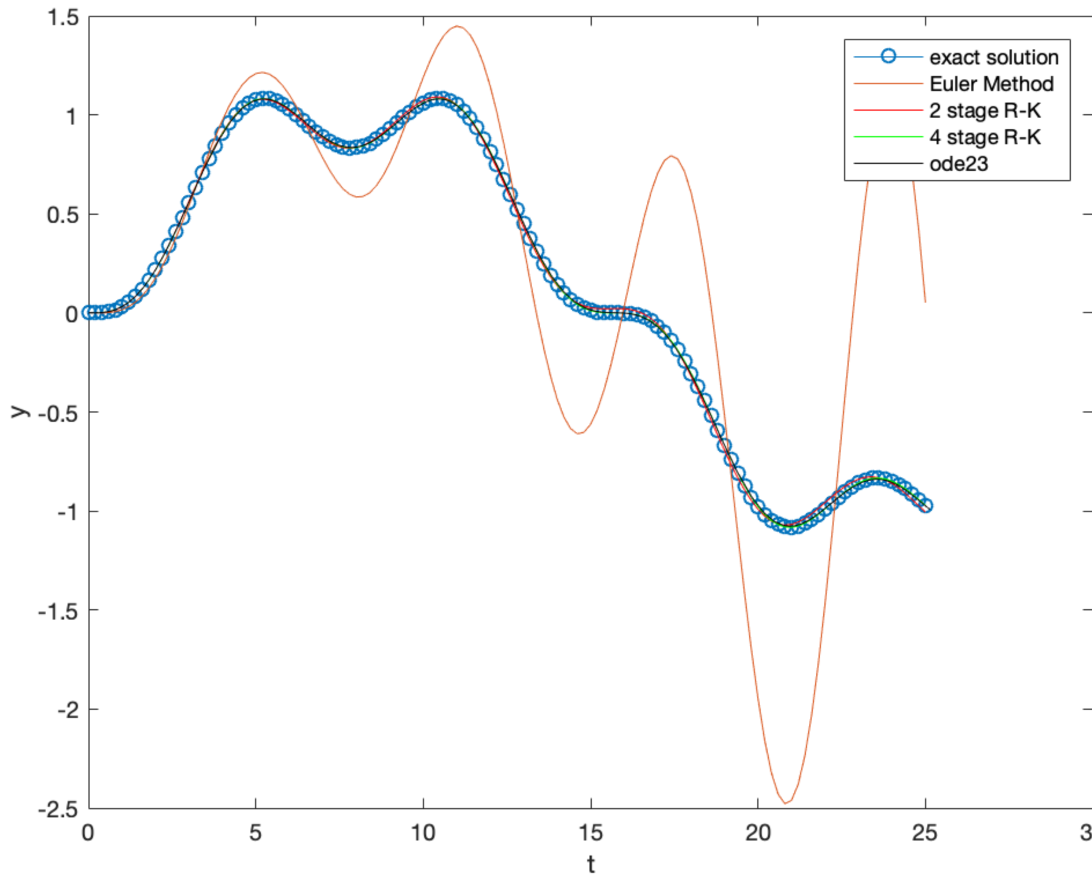


Adaptive Integrator

We use the adaptive integration method ode23.m supplied with matlab. It is also very accurate, and would require fewer steps.

```
[tout yout] = ode23(ydot,[0 8*pi],y0);
```

```
figure(1)
hold on
plot(tout,yout(:,1),'k')
hold off
legend('exact solution','Euler Method','2 stage R-K','4 stage R-K','ode23')
```



Error Comparison

A better way of looking at the different methods is to determine the maximum error of each. Here, since we have the exact solution, we can simply subtract it off and determine the maximum deviation.

```
errorem = max(abs(yem(1,:)-yexact(tall')))

error2s = max(abs(y2s(1,:)-yexact(tall')))

error4s = max(abs(y4s(1,:)-yexact(tall')))

errorode23 = max(abs(yout(:,1)-yexact(tout)))
```

```
errorem =
```

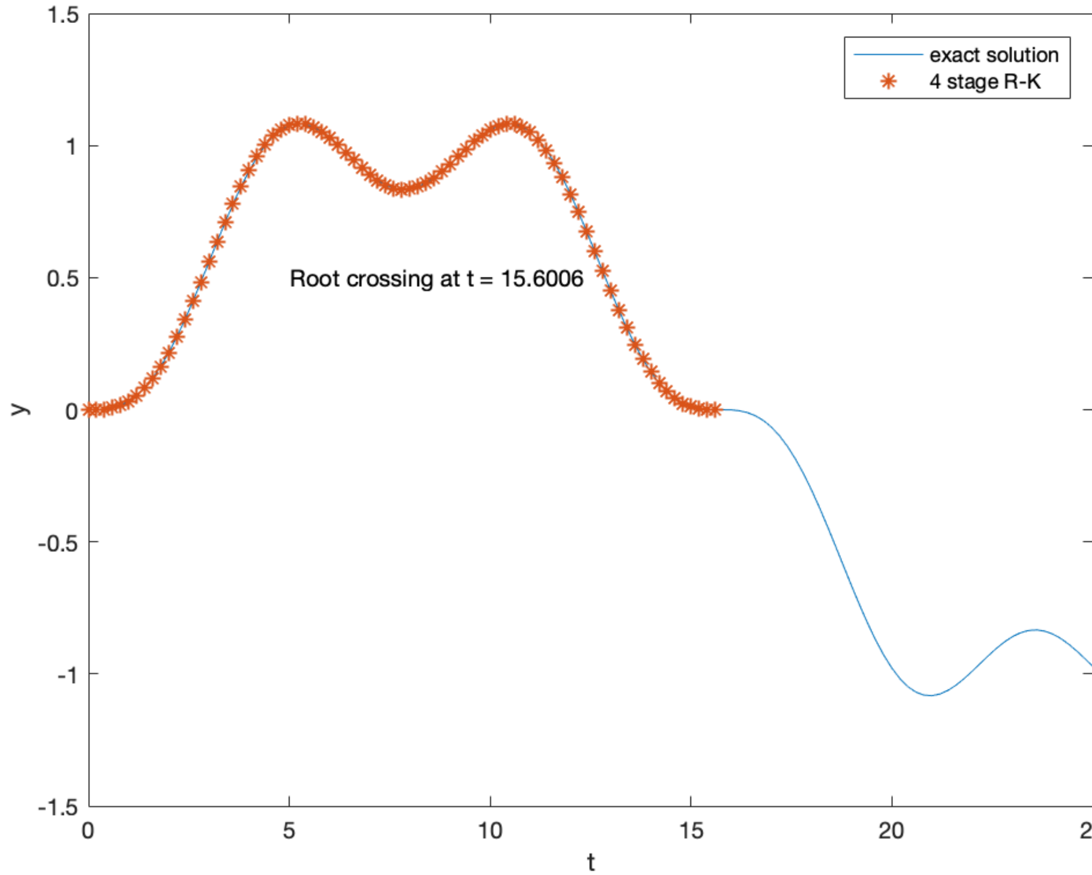
```
   1.9829e+00


error2s =

   3.3977e-02


error4s =

   6.5674e-05


errorode23 =

   6.3964e-03
```

Other Approaches

Often it is desired to integrate to some condition (e.g., where a function value reaches zero) rather than over a discrete range in time. This can be easily done for RK techniques using a while loop rather than a for loop (updating the index at each step). At the conclusion of the integration the array of function values and independent variables is trimmed (it needs to be predimensioned for efficiency) and the last element is adjusted via interpolation for improved accuracy. Note that you can also adjust the array size inside the while loop as well. This can also be done using the adaptive integrator ode23 via the "options" command. An implementation of a "while" loop approach is given below.

```
t0 = 0; %The initial time
y0 = [0; 0]; %The inital value (column vector)

nchunk = 20; %We will predimension the arrays and expand as necessary
tallw = zeros(1, nchunk); %We will keep time as a row vector
y4sw = zeros(length(y0),nchunk); %The y values are an array

tallw(1) = t0;
y4sw(:,1) = y0;

i = 1;

while (y4sw(1,i)>0)||(i == 1); %The truncation condition, doing it at least once

    if i+1 > length(tallw); %Redimensioning the array
        y4sw = [y4sw, zeros(length(y0),nchunk)];
        tallw = [tallw,zeros(1, nchunk)];
    end

    k1 = dt*ydot(tallw(i),y4sw(:,i));
    k2 = dt*ydot(tallw(i)+dt/2,y4sw(:,i)+k1/2);
```

```
    k3 = dt*ydot(tallw(i)+dt/2,y4sw(:,i)+k2/2);
    k4 = dt*ydot(tallw(i)+dt,y4sw(:,i)+k3);

    y4sw(:,i+1) = y4sw(:,i) + (k1+2*k2+2*k3+k4)/6;
    tallw(i+1) = tallw(i) + dt;

    i = i+1; %We update i
end

tallw = tallw(1:i); %We trim the values
y4sw = y4sw(:,1:i);

% Now for the interpolation for the root crossing:
f = y4sw(1,end-1)/(y4sw(1,end-1)-y4sw(1,end)); %Test on y = 0
tallw(end) = tallw(end-1)+f*dt;
y4sw(:,end) = y4sw(:,end-1) + f*(y4sw(:,end)-y4sw(:,end-1));

figure(2)
plot(tall,yexact(tall),tallw,y4sw(1,:),'*')
xlabel('t')
ylabel('y')
legend('exact solution','4 stage R-K')
text(5,.5,['Root crossing at t = ',num2str(tallw(end))])
```



Conclusion

As can be seen, the Euler Method isn't very accurate for this differential equation. The two RK methods are much more accurate, with the 4 stage method being three orders of magnitude better than the 2 stage method for this choice of dt. The accuracy of the adaptive quadrature routine is determined by its tolerances, and can be adjusted. For the choice of parameters used here it has the same total number of steps as the other methods and lies between the 2 stage and 4 stage methods in accuracy.

Published with MATLAB® R2017a

Sturm-Liouville Theory

Arguably the <u>most</u> <u>important</u> class of 2^{nd} order ODEs is Sturm-Liouville Boundary Value problems. These naturally arise from unsteady, bounded transport problems solved via separation of variables, and are described in Ch 11 of Boyce & DiPrima. Here we just state the theorem & look at a couple of examples.

A SL problem has the form:

$$\left[P(x)\, y'\right]' - q(x)\, y + \lambda\, w(x)\, y = 0$$

on the interval $0 < x < 1$

w/ <u>homogeneous</u> BCs:

$$a_1\, y(0) + a_2\, y'(0) = 0$$

$$b_1\, y(1) + b_2\, y'(1) = 0$$

where p, p', q & w are continuous functions of x on $[0,1]$ and $p(x) > 0$ and $w(x) > 0$ on $[0,1]$ (this condition is relaxed for singular SL problems which are very similar)

For such a problem we have:

1) All eigenvalues λ and corresponding eigenfunctions are real (not complex)

2) All eigenvalues are simple: one-to-one corresp. w/ eigenfunctions and may be ordered by magnitude s.t. $\lambda_n \to \infty$ as $n \to \infty$

3) Orthogonality: if $\phi_i(x)$ and $\phi_j(x)$ are two eigenfunctions (solutions) w/ corresp. eigenvalues λ_i, λ_j then:

$$\int_0^1 \phi_i \phi_j \, w(x) \, dx = 0 \text{ if } i \neq j$$

weight f^{n} from DE

4) The set of eigenfunctions is complete: you can represent any function over $[0,1]$ as a linear combination of the ϕ_i.

Example: A vibrating string has a spatial part governed by:

$$y'' + \lambda y = 0$$

$$y(0) = 0, \quad y(1) = 0$$

The solution to this eq'n is just $y = 0$! This is the trivial solution. But for specific values of λ there exist non-trivial solutions! These are the eigenvalues and eigenfunctions!

The solution is:

$$y = A \sin\sqrt{\lambda}\, x + B \cos\sqrt{\lambda}\, X$$

Apply the BC at $x=0$:

$$y(0) = A \underbrace{\sin\sqrt{\lambda}\,0}_{0} + B \underbrace{\cos\sqrt{\lambda}\,0}_{1} = 0$$

$\therefore B = 0$

apply BC at $x=1$:

$$y(1) = A \sin\sqrt{\lambda} = 0$$

$\therefore \sqrt{\lambda} = n\pi \quad n = 1, 2, \ldots$

We can represent any function over $[0,1]$ by a linear combination of the ϕ_i:

$$f(x) = \sum_{n=1}^{\infty} A_n \sin n\pi x$$

We get the A_n by orthogonality! Multiply both sides by $\sin m\pi x$ and $w(x)$ (but it's 1 here!)

and integrate:

$$\int_0^1 f(x)\sin(m\pi x)\,w(x)\,dx$$

$$= \sum_{n=1}^{\infty}\int_0^1 A_n \sin n\pi x \sin m\pi x \; w(x)\,dx$$

$$= \int_0^1 A_m (\sin m\pi x)^2 \, w(x)\,dx$$

so $A_n = \dfrac{\int_0^1 f(x)\sin(n\pi x)\,w(x)\,dx}{\int_0^1 (\sin(n\pi x))^2 w(x)\,dx}$

Now in this case $w(x) = 1$ and

$\int_0^1 \sin^2 n\pi x \, dx = \frac{1}{2}$

$\therefore \; A_n = 2\int_0^1 f(x)\sin(n\pi x)\,dx$

This is the Fourier Expansion of $f(x)$!

$$f(x) = \sum_{n=1}^{\infty} A_n \sin n\pi x$$

where A_n is given above.

The solution to <u>many</u> SL problems are available in the reference Carslaw & Jaeger, Conduction of Heat in Solids (since that <u>yields</u> SL problems). It is also easy to solve numerically in Matlab using an eigenvalue solver. A code which does this is provided in 356. We'll use this again when discussing PDE solutions via sep. of variables.

Div, Grad & Curl

So far we've reviewed ODEs - functions of a single indep. variable. Transport (& most of engineering) requires more than one! In addition, momentum is a vector - so there are multiple dependent variables too!

The derivative operator is:

$$\underset{\sim}{\nabla} \equiv \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$$

in cartesian coord.

So $\underset{\sim}{\nabla}\phi \equiv \left(\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z}\right)$

This is a vector prop. to local slope and points uphill. This is the gradient.

If $\underset{\sim}{u}$ is a vector, $\underset{\sim}{u} \equiv (u_x, u_y, u_z)$ (e.g., the velocity vector), then

$$\underset{\sim}{\nabla} \cdot \underset{\sim}{u} \equiv \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$$

which is a scalar (the divergence of $\underset{\sim}{u}$).
This is very different from $\underset{\sim}{u} \cdot \underset{\sim}{\nabla}$:

$$\underset{\sim}{u} \cdot \underset{\sim}{\nabla} = u_x \frac{\partial}{\partial x} + u_y \frac{\partial}{\partial y} + u_z \frac{\partial}{\partial z}$$

which is a scalar operator
and $\underset{\sim}{\nabla} \underset{\sim}{u}$ (vector composition product)
which is a matrix:

$$\underset{\sim}{\nabla} \underset{\sim}{u} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{bmatrix}$$

All three are very important in transport, so you have to pay attention!
You can take the divergence of the gradient:

$$\underset{\sim}{\nabla} \cdot \underset{\sim}{\nabla} \phi \equiv \underset{\sim}{\nabla} \cdot \left(\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z} \right)$$
$$= \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \equiv \nabla^2 \phi$$

This is the Laplacian of ϕ which is the 3-D analog of the 2nd derivative.

In heat transfer in solids:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T$$

α thermal diffusivity

So if $\nabla^2 T > 0$ ("concave up") the local temperature increases in time — which makes sense!

These operators can also apply to vectors (or tensors). In momentum transfer the velocity vector $\underset{\sim}{u}$ is governed by the Navier-Stokes equations:

$$\rho\left(\frac{\partial \underset{\sim}{u}}{\partial t} + \underset{\sim}{u}\cdot\underset{\sim}{\nabla}\underset{\sim}{u}\right) = -\underset{\sim}{\nabla}p + \mu\nabla^2\underset{\sim}{u} + \rho\underset{\sim}{g}$$

density   pressure gradient   viscosity   gravity

This is actually <u>3</u> equations - one for each u_x, u_y, u_z. That's because it is a <u>vector equation</u>.

There's another way $\underset{\sim}{\nabla}$ is used: the <u>curl</u> of a vector field. The <u>curl</u> of a magnetic field $\underset{\sim}{B}$ is the current density (from E&M):

$$\underset{\sim}{\nabla}\times\underset{\sim}{B} = \frac{4\pi}{c}\underset{\sim}{J} \leftarrow \text{current density}$$

In fluid mechanics the <u>vorticity</u> is the curl of the velocity:

$$\underset{\sim}{\omega} = \underset{\sim}{\nabla}\times\underset{\sim}{u}$$

The vorticity is useful in a number of ways, but curls are a bit messy!

$$\underset{\sim}{\omega} = \underset{\sim}{\nabla} \times \underset{\sim}{u} = \begin{vmatrix} \underset{\sim}{i} & \underset{\sim}{j} & \underset{\sim}{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u_x & u_y & u_z \end{vmatrix}$$

$$= \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}, \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}, \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right)$$

You can show that the vorticity is -2x the local rate of rotation of a fluid.

Manipulation of the vorticity & curl is <u>much</u> <u>easier</u> using <u>index notation</u> which we introduce elsewhere.

The last bit we'll talk about here is the <u>fundamental</u> <u>theorem</u> in multi-variable calculus: The <u>divergence</u> <u>theorem</u>

For an arbitrary closed surface ∂D w/ domain D and outward pointing unit normal $\underset{\sim}{n}$, we have:

$$\int_{\partial D} \phi \, \underset{\sim}{n} \, dA = \int_{D} \underset{\sim}{\nabla} \phi \, dV$$

surface integral

volume integral

(note that this requires ϕ be differentiable on D)

This works for vectors too:

$$\int_{\partial D} \underset{\sim}{u} \cdot \underset{\sim}{n} \, dA = \int_{D} \underset{\sim}{\nabla} \cdot \underset{\sim}{u} \, dV$$

We will use this extensively in deriving transport equations!

Solving PDEs: Simplification! (13-1)

PDEs are far more difficult to solve than ODEs. Thus the most common method is to turn them into ODEs!

One method is via simplification/change of variables. Suppose you fill a straw with a viscous liquid and let it drip out. The velocity is governed by the PDE:

$$\frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} = -\frac{\rho g}{\mu}$$

$$u_z \Big|_{x^2+y^2=a^2} = 0 \quad \text{(tube walls)}$$

We can simplify this via a change of variables:

$$r = (x^2+y^2)^{1/2}; \; x = r\cos\theta, \; y = r\sin\theta$$

The conversion from Cartesian coords

to cylindrical coords yields:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_z}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 u_z}{\partial \theta^2}=-\frac{\rho g}{\mu}$$

w/ BCs $u_z\big|_{r=0}=\text{finite}$, $u_z\big|_{r=a}=0$

In this case $u_z \neq f^n(\theta)$ (by symmetry)

so $\frac{\partial^2 u_z}{\partial \theta^2}=0$, thus we have the ODE:

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{du_z}{dr}\right)=-\frac{\rho g}{\mu}$$

In general, you don't want to break up the operator! Just multiply by r and integrate:

$$r\frac{du_z}{dr}=-\frac{1}{2}r^2\frac{\rho g}{\mu}+C_1$$

Divide by r:

$$\frac{du_z}{dr}=-\frac{1}{2}r\frac{\rho g}{\mu}+\cancel{\frac{C_1}{r}}$$

0 as finite at $r=0$!

And integrate again:

$$u_z = -\frac{1}{4} r^2 \frac{\rho g}{\mu} + c_2$$

Apply BC at $r = a$:

$$0 = -\frac{1}{4} a^2 \frac{\rho g}{\mu} + C_2$$

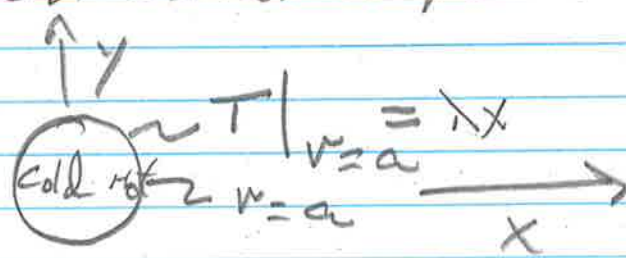
$$\therefore\ u_z = \frac{1}{4} \frac{\rho g a^2}{\mu} \left(1 - r^2/a^2\right)$$

which is much easier than solving it in Cartesian Coords!

$\Rightarrow$ In general, always pick a coordinate system in which the boundary has a convenient representation!

In addition to this, you often can simplify a problem by looking at the boundary conditions and allowing them to suggest the form of the solution!

An example: 2-D heat transfer from a circular dipole.



$$T\Big|_{r\to\infty} = 0$$

So: $\nabla^2 T = 0$ (Laplace's Equation)

$$T\Big|_{(x^2+y^2)^{1/2}=a} = \lambda x$$ (hot on one side, cold on the other)

$$T\Big|_{r\to\infty} = 0$$

$$\therefore \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

<u>First</u> since the boundary is a circle (cylinder) use cylindrical coords!

$$x = r\cos\theta, \quad y = r\sin\theta$$

In 2-D $$\nabla^2 T \equiv \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2}$$

And the BCs become:

$$T\Big|_{r\to\infty} = 0\ , \quad T\Big|_{r=a} = \lambda a \cos\theta$$

This time the problem is a function of both r & θ. But the ∇^2 operator only has 0^{th} or 2^{nd} deriv. w.r.t. θ. Thus, we allow the BC to suggest that:

$$T = \lambda \cos\theta\, f(r)$$

So: $f\Big|_{r=a} = a \qquad f\Big|_{r\to\infty} = 0$

and

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{df}{dr}\right)\cos\theta - \frac{f}{r^2}\cos\theta = 0$$

or $$\frac{1}{r}\frac{d}{dr}\left(r\frac{df}{dr}\right) - \frac{f}{r^2} = 0$$

Which is not only an ODE, it's an Euler equation!

So:

$$f \sim r^n$$

$$\therefore \quad n^2 r^{n-2} - r^{n-2} = 0$$

$$n^2 - 1 = 0$$

$$n = \pm 1$$

So $f = c_1 r + \frac{c_2}{r}$

$f(\infty) = 0 \quad \therefore \quad c_1 = 0$

$f(a) = a \quad$ so $\quad c_2 = a^2$

and $T = \frac{a^2}{r} \cos\theta$!

So:

1) Choose a coord system in which the BC (and problem) has a convenient rep.

2) Allow the BC to <u>suggest</u> the form of the solution!

Another important way to simplify PDEs arises for linear problems which are driven by periodic forcing. Examples include the pulsatile flow of your heart, to the periodic heating of the earth over the days & seasons!

For a linear problem we can solve these via analytic continuation into the complex plane!

$\Rightarrow$ key idea: $e^{i\omega t} = \cos\omega t + i\sin\omega t$

so if this is your driving force, then the real part of the solution is that driven by $\cos\omega t$ and the imaginary part is that driven by $\sin\omega t$!

Why does this help? Two ways:

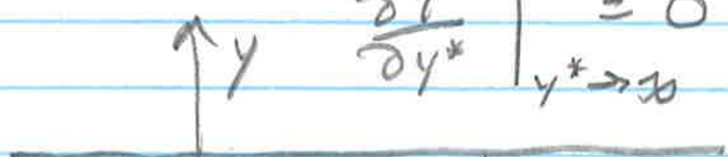
1) For a linear problem the solution will also be periodic!

2) Usually you have 1^{st} derivatives wrt t, so we have:

$$\frac{\partial}{\partial t}\left(e^{i\omega t}\right) = i\omega e^{i\omega t}$$

and you get $e^{i\omega t}$ back again!

An example: periodic heating



$$\left.\frac{\partial T^*}{\partial y^*}\right|_{y^*\to\infty} = 0$$

$$\left. T^*\right|_{y^*=0} = \cos t^*$$

We have the PDE:

$$\frac{\partial T^*}{\partial t^*} = \frac{\partial^2 T^*}{\partial y^{*2}}\ ; \quad \left. T^*\right|_{y^*=0} = \cos t^*, \quad \left.\frac{\partial T^*}{\partial y^*}\right|_{y^*\to\infty} = 0$$

Let's solve via analytic continuation:

$$\hat{T}\Big|_{y^*=0} = e^{it^*} \quad \therefore \; T^* = \mathrm{Re}\{\hat{T}\}$$

we expect $\hat{T}$ to be periodic

$$\therefore \quad \hat{T} = e^{it^*} f(y^*)$$

So: $$\frac{\partial \hat{T}}{\partial t^*} = i e^{it^*} f = \frac{\partial^2 \hat{T}}{\partial y^{*2}} = e^{it^*} f''$$

$$\therefore \quad f'' = if \quad \text{s.t.} \quad f\Big|_{y^*=0} = 1, \quad f'\Big|_{y^*\to\infty} = 0$$

This is a constant coef ODE! We want the decaying part (due to $f'(\infty)=0$)

$$\therefore \quad f = A e^{-\sqrt{i}\, y^*} \quad \text{w/ } f(0)=1 \;\therefore\; A=1$$

and $$\hat{T} = e^{it^*} e^{-\sqrt{i}\, y^*}$$

Now $\sqrt{i} = \frac{1+i}{\sqrt{2}}$

so $\hat{T} = e^{-y^*/\sqrt{2}} e^{i(t^* - \frac{y^*}{\sqrt{2}})}$

and since $T^* = Re\{\hat{T}\}$

we get:

$$T^* = e^{-y^*/\sqrt{2}} \cos(t^* - y^*/\sqrt{2})$$

so our temperature profile is an exponentially decaying wave w/ a phase shift!

Note that if you had a bounded domain (instead of $y \to \infty$) you would use hyperbolics rather than exponentials in y $\rightarrow$ more complex, but Matlab will plot up a complex solution just fine!

Simplification & Estimation via Scaling

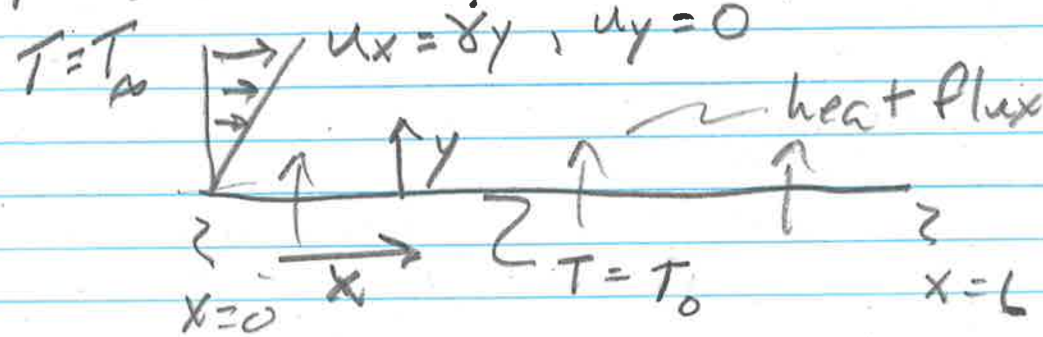
- A very useful technique for analyzing PDEs (ODEs too!) is scaling analysis
- This forms the basis of scale-up, used in many engineering problems!

⇒ The core idea is that variables (such as position, velocity, time, etc) all have units, but any relation should be able to be rendered dimensionless

⇒ scale all dependent (e.g., velocity, temperature, etc.) and indep. (e.g., position, time) variables with parameters so that they are of $O(1)$ in the region of interest!

To see how this works, look at a classic problem from ht transf:

Convective heat transfer from a plate at const. temperature!



we have the wall heat flux $q|_{y=0}$:

$$q\Big|_{y=0} = -k\frac{\partial T}{\partial y}\Big|_{y=0}$$

The total heat loss is:

$$\frac{Q}{W} = \int_0^L q\Big|_{y=0} dx = \int_0^L -k\frac{\partial T}{\partial y}\Big|_{y=0} dx$$

What is the rate of heat loss from the plate and how does it dep. on parameters such as the shear rate $\dot{\gamma}$ and material props?

To solve we will scale the energy equation:

$$\rho \hat{C}_p \underset{\sim}{u} \cdot \underset{\sim}{\nabla} T = k \nabla^2 T$$

- In 2-D we get:

$$\rho \hat{C}_p \left(u_x \frac{\partial T}{\partial x} + u_y \frac{\partial T}{\partial y} \right) = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

Now for simple shear $u_x = \dot{\gamma} y$, $u_y = 0$

$$\therefore \rho \hat{C}_p \dot{\gamma} y \frac{\partial T}{\partial x} = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

Let's scale!

From the geometry, $x^* = \frac{x}{L}$

(e.g., x^* goes from 0 to 1!)

Our problem is linear in T so we can subtract off a ref. temp!

$$T^* = \frac{T - T_\infty}{\Delta T_c}$$

From our B.C. at $y=0$:

$$T\Big|_{y=0} = T_0$$

$$\therefore \quad \Delta T_c T^*\Big|_{y^*=0} = T_0 - T_\infty$$

$$T^*\Big|_{y^*=0} = \frac{T_0 - T_\infty}{\Delta T_c} = \underline{\underline{1}}$$

so $\Delta T_c = T_0 - T_\infty$

and $T^*\Big|_{y=0} = 1$, $T^*\Big|_{y\to\infty} = 0$

Now for y:

we could use $y^* = y/L$ but it's not the best choice! Instead we scale w/ unknown param δ:

$$\underline{y^* = y/\delta}$$

Now we plug into the PDE:

$$\rho \hat{C}_p \dot{\gamma} \delta \frac{\Delta T_c}{L} y^* \frac{\partial T^*}{\partial x^*} = k\left(\frac{\Delta T_c}{L^2}\frac{\partial^2 T^*}{\partial x^{*2}} + \frac{\Delta T_c}{\delta^2}\frac{\partial^2 T^*}{\partial y^{*2}}\right)$$

To finish you must divide through by one of the groups of params!

Choose the group multiplying a term (physical mechanism) you think is important!

— For this problem, conduction away from the plate in the y-dir must matter, so divide by its scaling!

$$k\frac{\Delta T_c}{\delta^2}$$

So:

$$\left[\frac{\rho \hat{C}_p \dot{\gamma} \delta^3}{k L}\right] y^* \frac{\partial T^*}{\partial x^*} = \left[\frac{\delta^2}{L^2}\right]\frac{\partial^2 T^*}{\partial x^{*2}} + \frac{\partial^2 T^*}{\partial y^{*2}}$$

We determine δ by balancing y-conduction w/ another physical mechanism: x-dir convection!

Thus, set $\left[\frac{\rho \hat{C}_p \dot{\gamma} \delta^3}{kL}\right] = 1$

so: $\delta = \left[\frac{kL}{\rho \hat{C}_p \dot{\gamma}}\right]^{1/3}$

This tells us that the Boundary Layer Thickness (where T^* is going from 1 to 0) is of $O(\delta)$ and how it varies w/ parameters!

What about the heat loss?

$$\frac{Q}{W} = \int_0^L -k \frac{\partial T}{\partial y}\Big|_{y=0} dx$$

should be $O(1)$!

$$= -\frac{k \Delta T_c L}{\delta} \int_0^1 \frac{\partial T^*}{\partial y^*}\Big|_{y^*=0} dx^*$$

So $\dfrac{Q}{W} = \dfrac{k \Delta T_c L}{\left[\dfrac{kL}{\rho \hat{C}_p \dot{\gamma}}\right]^{1/3}} \displaystyle\int_0^1 - \left.\frac{\partial T^*}{\partial y^*}\right|_{y^*=0} dx^*$

If it's scaled right the integral is $O(1)$!

- We need to plug back into PDE:

$$y^* \frac{\partial T^*}{\partial x^*} = \frac{\partial^2 T^*}{\partial y^{*2}} + \frac{\delta^2}{L^2} \frac{\partial^2 T^*}{\partial x^{*2}}$$

where $\dfrac{\delta^2}{L^2} = \dfrac{\left[\dfrac{kL}{\rho \hat{C}_p \dot{\gamma}}\right]^{2/3}}{L^2} = \left[\dfrac{k}{\rho \hat{C}_p \dot{\gamma} L^2}\right]^{2/3}$

Now provided $\delta^2/L^2 << 1$, or

$$L >> \left(\frac{k}{\rho \hat{C}_p \dot{\gamma}}\right)^{1/2} \quad \text{(same thing!)}$$

diffusion in the x-direction is negligible! Thus we get the dimensionless eq'n:

$$y^* \frac{\partial T^*}{\partial x^*} = \frac{\partial^2 T^*}{\partial y^{*2}} + \cancel{\left(\frac{\delta^2}{L^2}\right)} \frac{\partial^2 T^*}{\partial x^{*2}}$$

small

w/ B.C.'s $T^*\big|_{y^*=0} = 1$, $T^*\big|_{y^* \to \infty} = 0$

and $\frac{Q}{W} = \left[\frac{k \Delta T_c L}{\delta}\right] \int_0^1 -\frac{\partial T^*}{\partial y^*}\bigg|_{y^*=0} dx^*$

So just from scaling we learn how Q/W depends on our parameters - and its approximate magnitude! We also learn how our BL thickness behaves - and how long the plate has to be for our BL approx to be valid! We also simplify our PDE!
⇒ This is most of what we want to know without solving the eqn!

Summary:

1) Render all dependent & indep. variables dimensionless w.r.t. parameters
2) often scaling parameters are determined from geometry or BCs!
3) If scalings aren't known (or obvious) use an "unknown char. value".
4) Divide out by the group of param. multiplying the term representing a physical mechanism you're pretty sure is important!

5) Determine unknown scaling param by setting other important groups equal to 1 — until you run out!
6) Check to make sure "leftovers" really are small — otherwise rescale!

Solving PDEs: Self-Similar Solutions

A very important (if special) class of problems are "length or time-scale" deficient - this includes problems such as Many Boundary Layer (BL) problems, unsteady diffusion from a point source, etc.

- Such problems often admit self-similar solutions where a transformed dependent variable is a function of a reduced set of independent variables: For example, you can turn a 2-D PDE into an ODE - a huge simplification!
- This is codified by Morgan's Theorem:

1) If a well-posed problem is invariant to a one parameter group of continuous transformations the number of

independent variables may be reduced by one.

2) The reduction is accomplished by choosing as new indep. & dep. variables combinations that are invariant under the transformation.

═

Ok, What does all this mean?

- well-posed problem: Consists of PDE, indep & dep. variables, BCs, location of BCs, etc.
- invariant: does not change (identical)
- one parameter: At least one parameter of the group of transformations is undetermined, but problem is still invariant
- continuous transformation: can include shifting or scaling, but doesn't include periodic (non-continuous) transformations.

- Reduction: new dependent and indep. variables are combinations of the old ones that are invariant under the transformation

There are many ways to apply this, but by far the simplest is Affine Stretching: Just stretch all dependent and indep. variables by an arbitrary factor. If some combination of these parameters exists that leaves the problem invariant but not all the parameters are forced to be 1, (e.g., "one free wish" is left over) you win!

Let's use the heated plate example again, but with the heat flux B.C.:

$$\left.\frac{\partial T^*}{\partial y^*}\right|_{y^*=0} = -1 \quad \text{(constant ht flux)}$$

So:

$$y^* \frac{\partial T^*}{\partial x^*} = \frac{\partial^2 T^*}{\partial y^{*2}} ; \quad T^*\Big|_{y^* \to \infty} = 0 ; \quad \frac{\partial T^*}{\partial y^*}\Big|_{y^*=0} = -1$$

Let's stretch dep. & indep. variables:

$$T^* = A\bar{T}, \quad x^* = B\bar{x}, \quad y^* = C\bar{y}$$

So the DE yields:

$$\frac{CA}{B} \bar{y} \frac{\partial \bar{T}}{\partial \bar{x}} = \frac{A}{C^2} \frac{\partial^2 \bar{T}}{\partial \bar{y}^2}$$

You want the minimum number of restrictions, so you divide out:

$$\frac{C^3}{B} \bar{y} \frac{\partial \bar{T}}{\partial \bar{x}} = \frac{\partial^2 \bar{T}}{\partial \bar{y}^2}$$

So the DE is invariant if $\frac{C^3}{B} = 1$

We must do the BCs too!

– in general, homogeneous (e.g., $= 0$) BCs don't add restrictions: just divide out!

So:
$$T^*\Big|_{y^*\to\infty} = 0$$

$$\therefore A\bar{T}\Big|_{C\bar{y}\to\infty} = 0$$

$$\text{or } \bar{T}\Big|_{\bar{y}\to\frac{\infty}{C}=\infty} = 0 \qquad \text{(no restrictions)}$$

(Note that a BC at $y^* = \underline{\underline{1}}$ would yield restriction $C = 1$ - and kill self-similarity - but ∞ is "homogeneous")

And we have $\dfrac{\partial T^*}{\partial y^*}\Big|_{y^*=0} = -1$

so $\dfrac{A}{C}\dfrac{\partial\bar{T}}{\partial\bar{y}}\Big|_{\bar{y}=0} = -1$ so $\dfrac{A}{C} = 1$

But we're done! 3 parameters - 2 restrictions = 1 free wish!

Any combination of T^*, x^*, y^* which are invariant under these transformations will work!

So:

$$\frac{A}{C} = 1 \; ; \; \frac{C^3}{B} = 1$$

$$\therefore \; \frac{T^*}{y^*} = f^n\left(\frac{y^{*3}}{x^*}\right)$$ would yield an ODE!

But this is a terrible choice!

Instead, use Canonical Form: Put all the complexity in the "time or time-like" variable.

⇒ Alternatively, put complexity in the variable whose highest deriv. in DE is the lowest (yields same result).

- For this problem we have 1^{st} deriv. w.r.t. x^* and 2^{nd} deriv. w.r.t. y^*

Thus, put complexity in x^*!

Now $x^* = B\bar{x}$, so put complexity in B!

We had:

$$\frac{A}{C} = 1, \quad \frac{C^3}{B} = 1$$

$\therefore \dfrac{C}{B^{1/3}} = 1$ and $\dfrac{A}{B^{1/3}}$ are equivalent!

So:

$$\frac{T^*}{x^{*1/3}} = f(\eta); \quad \eta = \frac{y^*}{x^{*1/3}}$$

or

$$\underbrace{T^* = x^{*1/3} f(\eta)}_{\text{similarity Rule}}; \quad \underbrace{\eta = \frac{y^*}{x^{*1/3}}}_{\text{similarity variable}}$$

Let's prove this works!

$$\frac{\partial T^*}{\partial y^*} = \frac{\partial}{\partial y^*}\left(x^{*1/3} f(\eta)\right) = x^{*1/3} \frac{df}{d\eta} \frac{\partial \eta}{\partial y^*}$$

Now $\dfrac{\partial \eta}{\partial y^*} = \dfrac{\partial}{\partial y^*}\left(\dfrac{y^*}{x^{*1/3}}\right) = \dfrac{1}{x^{*1/3}}$

$$\therefore \frac{\partial T^*}{\partial y^*} = \frac{df}{d\eta} \equiv f'$$

So BC at $y^* = 0$ is:

$$\left.\frac{\partial T^*}{\partial y^*}\right|_{y^*=0} = -1 \implies f'(0) = -1$$

We also need the second derivative:

$$\frac{\partial^2 T^*}{\partial y^{*2}} = \frac{\partial}{\partial y^*}\left(\frac{\partial T^*}{\partial y^*}\right) = \frac{\partial f'}{\partial y^*} = x^{*-1/3} f''$$

and finally:

$$\frac{\partial T^*}{\partial x^*} = \frac{\partial}{\partial x^*}\left(x^{*1/3} f\right) = \frac{1}{3} x^{*-2/3} f + x^{*1/3} f' \frac{\partial z}{\partial x^*}$$

Now $\frac{\partial z}{\partial x^*} = \frac{\partial}{\partial x^*}\left(\frac{y^*}{x^{*1/3}}\right) = -\frac{1}{3}\frac{y^*}{x^{*4/3}} = -\frac{1}{3}\frac{z}{x^*}$

In general if $z \equiv \frac{y^*}{x^{*n}}$ then $\frac{\partial z}{\partial x^*} = -n\frac{z}{x^*}$

So:

$$\frac{\partial T^*}{\partial x^*} = \frac{1}{3} x^{*-2/3}(f - zf')$$

and the DE becomes:

$$\frac{1}{3}\frac{y^*}{x^{*2/3}}(f - zf') = x^{*-1/3} f''$$

multiply by $x^{*1/3}$:

$$f'' = \frac{1}{3}\frac{y^*}{x^{*1/3}}(f - zf')$$

but $\frac{y^*}{x^{*1/3}} = z$!

so: $f'' = \frac{1}{3}\eta(f - \eta f')$

$f(\infty) = 0, \quad f'(0) = -1$

This is very easy to solve numerically, and even has an analytic solution!

Note that we can answer the question: what is the temperature of the plate?

$$T^*\Big|_{y^*=0} = x^{*\,1/3} f(0)$$

where $f(0)$ is an $O(1)$ constant...

Actually, we find $f(0) = 1.5363$ using just a few lines of code...

Solution via Separation of Variables

- Some PDEs can be solved via separation of variables. The PDE must be linear in the dependent variable, and it also needs a separable structure. Many transport problems (particularly transients) qualify!

Key steps

1) Determine the asymptotic solution. This should satisfy inhomogeneities in the DE or BCs - but not the initial condition.

2) Subtract off this solution to get DE (w/ homogeneous BCs) for the decaying part! This bit now has an inhomogeneous IC!

3) Assume that sol'n to decaying part is a product of a spatial part and a time-dependent part: Substitute in & divide out!

4) If one side of eq'n is a function only of time & other side is a function only of spatial variable, both must be constant: you get two ODEs!

5) Solve the time-dependent part – this is usually just an exponential!

6) The spatial part is a Sturm-Liouville eigenvalue problem! This is why you need homogeneous BCs!

7) Determine the eigenvalues & eigenfunctions of the S-L problem.

8) Get the coefficients of your now series solution (one term for each eigenvalue) using orthogonality and IC created by subtracting off asymptotic solution!

9) The complete solution is the sum of asymptotic and decaying solutions!

Let's see how this works: Start up flow in a tube!

A vertical straw of radius a is filled w/ a fluid of viscosity μ and density ρ. At $t=0$ you take your finger off the end and flow starts. What is the centerline velocity as a function of time??

DE: $\rho \frac{\partial u}{\partial t} = \mu \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \rho g$

BC: $u\big|_{r=a} = 0$, $\frac{\partial u}{\partial r}\Big|_{r=0} = 0$ (symmetry)

IC: $u\big|_{t=0} = 0$

We <u>start</u> by scaling!

$r^* = r/a$, $\quad u^* = \frac{u}{U_c}$, $\quad t^* = \frac{t}{t_c}$

$$\therefore \frac{\rho U_c}{t_c}\frac{\partial u^*}{\partial t^*} = \frac{\mu U_c}{a^2}\frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u^*}{\partial r^*}\right) + \rho g$$

<u>Divide</u> by ρg as that <u>must</u> matter!

$$\left[\frac{U_c}{g t_c}\right]\frac{\partial u^*}{\partial t^*} = \left[\frac{\mu U_c}{\rho g a^2}\right]\frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u^*}{\partial r^*}\right)$$

$$\left[\frac{U_c}{g t_c}\right] = 1, \quad \left[\frac{\mu U_c}{\rho g a^2}\right] = 1$$

$$\therefore U_c = \frac{\rho g a^2}{\mu} \quad \text{so} \quad t_c = \frac{U_c}{g} = \frac{\rho a^2}{\mu}$$

and:
$$\frac{\partial u^*}{\partial t^*} = \frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u^*}{\partial r^*}\right) + 1$$

$$u^*\Big|_{r^*=1} = 0, \quad \frac{\partial u^*}{\partial r^*}\Big|_{r^*=0} = 0; \quad u^*\Big|_{t^*=0} = 0$$

Now the BCs are <u>already</u> homogeneous, but the DE <u>isn't</u>!

At <u>long</u> <u>times</u> ($t^* \gg 1$) flow is <u>steady</u> (note: not <u>always</u> true!)

So: $$\frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u_\infty^*}{\partial r^*}\right) = -1$$

$$u_\infty^*\Big|_{r^*=1} = 0 \qquad \frac{\partial u_\infty^*}{\partial r^*}\Big|_{r^*=0} = 0$$

Integrating:
$$r^*\frac{\partial u_\infty^*}{\partial r^*} = -\frac{1}{2}r^{*2} + \cancel{C_1}$$
0 from BC at $r^*=0$

So $u_\infty^* = -\frac{1}{4}r^{*2} + C_2$ where $C_2 = \frac{1}{4}$

$\therefore u_\infty^* = \frac{1}{4}(1 - r^{*2})$ (Poiseuille flow)

Subtract off u_∞^*:

$$u^* = u_\infty^* + u_d^*$$

$$\therefore \underbrace{\cancel{\frac{\partial u_\infty^*}{\partial t^*}}}_{0} + \frac{\partial u_d^*}{\partial t^*} = \underbrace{\frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u_\infty^*}{\partial r^*}\right) + 1}_{=0} + \frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u_d^*}{\partial r^*}\right)$$

$$\therefore \frac{\partial u_d^*}{\partial t^*} = \frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u_d^*}{\partial r^*}\right)$$

$$u_d^*\Big|_{r^*=1} = 0 \,; \quad \frac{\partial u_d^*}{\partial r^*}\Big|_{r^*=0} = 0$$

But

$$u_d^*\Big|_{t^*=0} = -u_\infty^*\Big|_{t^*=0} = -\frac{1}{4}(1-r^{*2})$$

Let $u_d^* = F(r^*)\,G(t^*)$

$$\therefore\; F G' = G \frac{1}{r^*}(r^* F')'$$

Divide out by FG:

$$\frac{G'}{G} = \frac{(r^* F')'}{r^* F} = cst = -\sigma^2 \;(\text{or } \lambda)$$

choose neg as look for decaying sol'n!

$$\therefore\; G' = -\sigma^2 G \quad \therefore\; G = e^{-\sigma^2 t^*}$$

And

$$(r^* F')' + \sigma^2 r^* F = 0$$

$$F'(0) = 0,\quad F(1) = 0$$

This is a S-L eigenvalue problem!

The solution is Bessel functions of order zero - $J_0(\sigma r^*)$

Thus the eigenvalues are the roots

$$J_0(\sigma_n) = 0$$

So: $$u_d^* = \sum_{n=1}^{\infty} A_n e^{-\sigma_n^2 t^*} J_0(\sigma_n r^*)$$

Now for the A_n:

At $t^* = 0$ $$u_d^* = \sum_{n=1}^{\infty} A_n J_0(\sigma_n r^*) = -u_{ss}^*\Big|_{t^*=0}$$

By orthogonality: weight f^n from DE!

$$A_n = \frac{\int_0^1 -u_{ss}^*\Big|_{t^*=0} J_0(\sigma_n r^*)\, r^* dr^*}{\int_0^1 \left(J_0(\sigma_n r^*)\right)^2 r^* dr^*}$$

$$= \frac{-2}{\sigma_n^3 J_1(\sigma_n)}$$

Bessel f^n of $O(1)$

so:

$$u^* = \frac{1}{4}(1 - r^{*2}) + \sum_{n=1}^{\infty} \frac{-2}{\sigma_n^3 J_1(\sigma_n)} e^{-\sigma_n^2 t^*} J_0(\sigma_n r^*)$$

In this case you can get everything analytically (which is nice) but often there is no analytic solution! No problem: just solve the SL problem numerically!

What do we care about? In order:

1) Asymptotic solution: what's left at the end!
2) lead eigenvalue: tells us how fast the asymptotic solution is reached!
3) lead eigenfunction & coefficient: this is the dominant part of decaying sol'n at long times!
4) everything else...

For this problem, $J_0(\sigma_1) = 0$ so
$\sigma_1 = 2.4048$
Thus the u_d^* decays as $e^{-\sigma_1^2 t^*} = e^{-5.78 t^*}$
so it's <u>mostly gone</u> at $t^* \sim \frac{1}{\sigma_1^2} = 0.173$

and is 90% gone at $t^* = 0.4$!
Higher order terms decay faster (and have smaller coefficients!)

The lead coefficient $A_1 = -0.2770$
which is close to $u_\infty^*\big|_{r^*=0} = 0.25$

So to a good approximation:

$$u^*\big|_{r^*=0} \approx \frac{1}{4} - 0.2770\, e^{-5.78 t^*}$$

provided $t^* \gtrsim 0.1$ (higher σ decay faster!)

```
function [lambda,eigenvec]=slsolve(varargin)
%This function solves the Sturm-Liouville eigenvalue problem given by:
%
%  [p(x) y']' - q(x) y + lambda w(x) y = 0
%
% subject to the boundary conditions:
%
%  bc(1) y(0) + bc(2) y'(0) = 0
%
%  bc(3) y(1) + bc(4) y'(1) = 0
%
% over the domain 0< x <1.
%
% The function is called by the command:
%
% [lambda,eigenvec]=slsolve('pfun','qfun','wfun',bc,n);
%
% The function call requires that you provide the function names (or
% handles if you are using the anonymous function utility) for the
% functions p, q, and w.  These functions must be able to handle an array
% of values.  You also provide the boundary coefficients in the array bc.
% In addition, you may specify the degree of discretization n.  Its default
% value is 50.  The matrices which are generated are of size (n+1,n+1).
% The function returns the eigenvalues in the array lambda (sorted by size
% in ascending order) and the matrix eigenvec which contains the
% corresponding eigenfunctions.  The eigenfunctions are all normalized by
% their maximum value over the domain 0<x<1.
%
% A last note on error:  The code uses second order derivative
% approximations, so the error in the eigenvalues and eigenvectors will be
% of O(1/n^2).  In general, the first few eigenvalues will be reliable, but
% the accuracy will deteriorate as you look at the higher eigenvalues, with
% the last few being meaningless.

p=varargin{1};
q=varargin{2};
w=varargin{3};
bc=varargin{4};

if nargin<5;n=50;else;n=varargin{5};end

h=1/n; %set discretization
x=[0:h:1]'; %this is the array of x values

%Now we set up the arrays used in making the matrix A:
pp=zeros(1,n+1);
pm=zeros(1,n+1);
ww=zeros(1,n+1);
qq=zeros(1,n+1);
for i=2:n
 pp(i)=feval(p,x(i)+h/2);
 pm(i)=feval(p,x(i)-h/2);
 ww(i)=feval(w,x(i));
 qq(i)=feval(q,x(i));
end

%The matrix W is easy:
weight=-diag(ww);
```

```
%The matrix A is a bit more complex.  First we do
%the main diagonal:
a=diag(-pp-pm-qq*h^2);
%and then the super and sub diagonals:
a=a+diag(pp(1:n),1);
a=a+diag(pm(2:n+1),-1);

%Finally, we divide by h^2:
a=a/h^2;

%And now for the boundary conditions.  First at the left edge:
a(1,1)=bc(1)-bc(2)*1.5/h;
a(1,2)=bc(2)*2/h;
a(1,3)=-bc(2)/2/h;

%and at the right edge:
a(n+1,n+1)=bc(3)+bc(4)*1.5/h;
a(n+1,n)=-bc(4)*2/h;
a(n+1,n-1)=bc(4)/2/h;

%Now we are ready to calculate the eigenvalues:
[v,d]=eig(a,weight);

%The number of eigenvalues and vectors will be less
%than the size of A and W, thus:
evals=diag(d);
i=find(isfinite(evals)); %The matlab 7 form of finite!
evals=evals(i);
evecs=v(:,i);

%Now we sort the eigenvalues and eigenvectors according
%to the size of the eigenvalues:
[~,i]=sort(abs(real(evals)));
lambda=evals(i);
evecs=evecs(:,i);

%and finally, we normalize the eigenvectors by their
%maximum value.
eigenvec=zeros(size(evecs));
for j=1:length(lambda)
 eigenvec(:,j)=evecs(:,j)/norm(evecs(:,j),inf)/sign(evecs(2,j));
end
```

Contents

Separation of Variables Example: Startup of flow in a tube

We examine the numerical solution to the startup of a fluid flowing through a circular tube: what happens in a vertical straw filled with fluid when you take your finger off the end.

```
% We have the asymptotic solution uinf:

uinf = @(r) (1-r.^2)/4;

% And we have the functions p, q, and w from the resulting spatial
% Sturm-Liouville problem:

p = @(x) x;
q = @(x) zeros(size(x));
w = @(x) x;

% The boundary conditions are zero derivative at the center and zero
% magnitude at r = 1:
bc = [0,1,1,0];

% We set the number of eigenvalues we would like (only the earlier ones are
% accurate) and the degree of discretization.

n = 100; %The number of points we would like (the number of intervals)

% and we use our solver:

[lambda, eigenvecs] = slsolve(p,q,w,bc,n);

% And that's it!
```

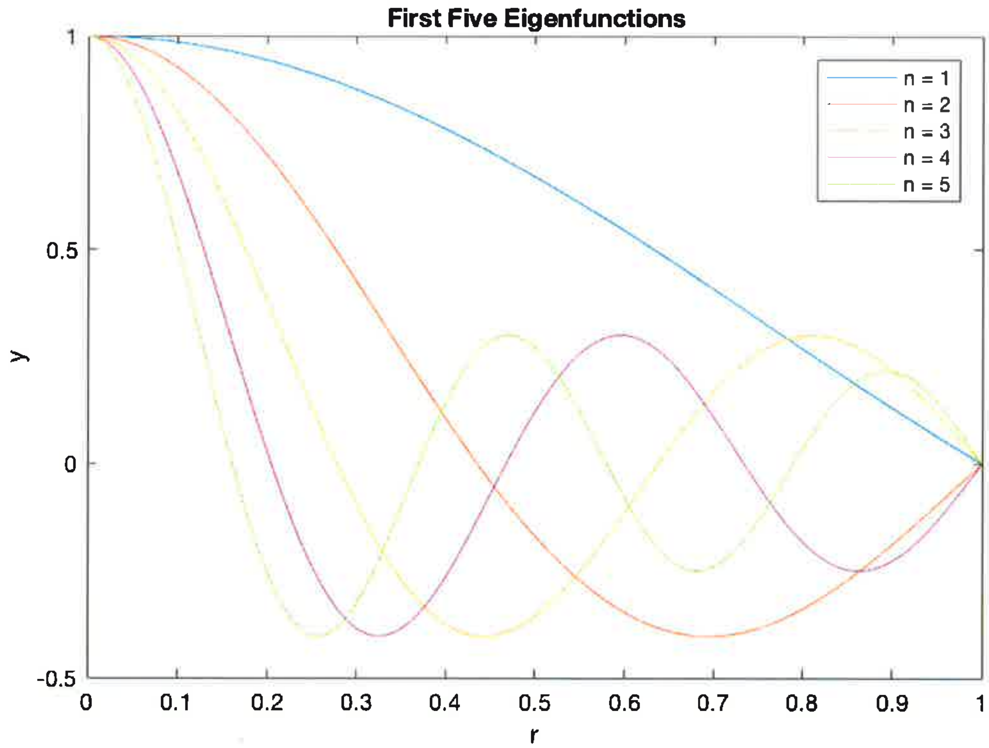
Eigenvalues, Eigenvectors, and Coefficients

We are interested in the lead eigenvalues, coefficients, and eigenvectors. We just look at the first five:

```
firsteigenvals = lambda(1:5)

% And we calculate the coefficients using the Trapezoidal Rule:
```

```
r = [0:1/n:1]';

% The Trapezoidal Rule weights:
weights = ones(1,n+1);
weights(1) = 0.5;
weights(n+1) = 0.5;
weights=weights/n;

a = zeros(length(lambda),1);

for i = 1:length(lambda)
    numerator = -weights*(w(r).*uinf(r).*eigenvecs(:,i));
    denominator = weights*(w(r).*eigenvecs(:,i).^2);
    a(i) = numerator/denominator;
end

firstcoefficients = a(1:5)

% And we plot the first five eigenfunctions:
figure(1)
plot(r,eigenvecs(:,1:5))
xlabel('r')
ylabel('y')
title('First Five Eigenfunctions')
legend('n = 1','n = 2','n = 3','n = 4','n = 5')
grid on
```

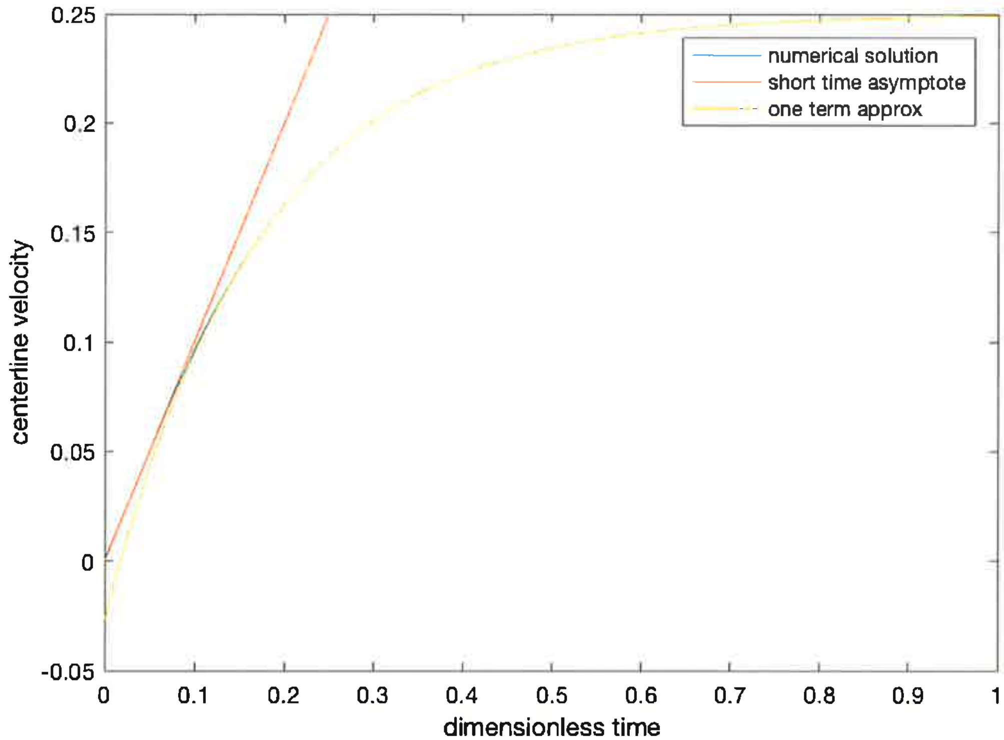
```
firsteigenvals =

    5.7829
   30.4633
   74.8397
  138.8784
  222.5174


firstcoefficients =

   -0.2770
    0.0350
   -0.0113
    0.0054
   -0.0028
```



Velocity at the Centerline

We are most interested in the velocity at the centerline. This will be uinf + udecaying evaluated at r = 0. So:

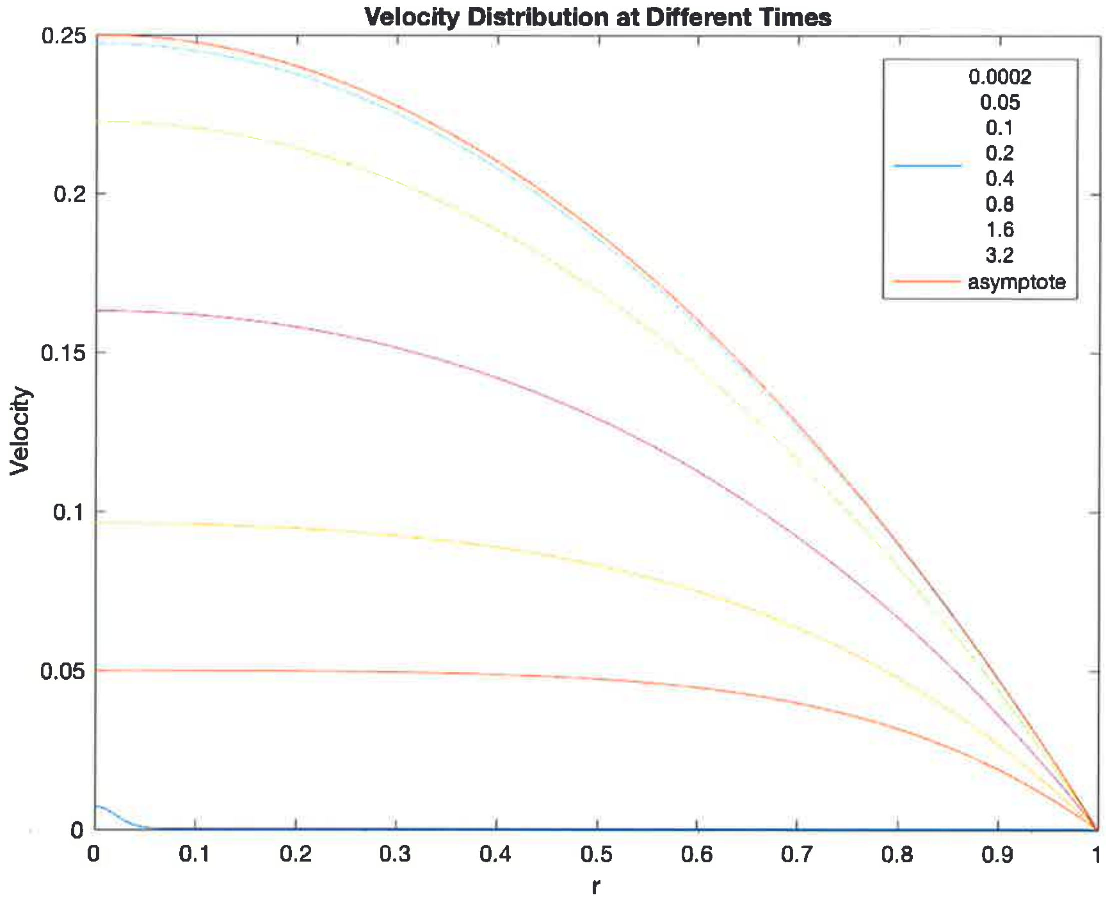
```
t = [0.0005:.001:1];
ucenter = zeros(size(t)); %We initialize the array

for i = 1:length(t)
    ucenter(i) = uinf(0) + sum(a.*exp(-lambda*t(i)).*eigenvecs(1,:)');
end

% We can also look at the one-term approximation:
ucenter1term = uinf(0) + a(1)*exp(-lambda(1)*t)*eigenvecs(1,1);

figure(2)
plot(t,ucenter,[0,.25],[0,.25],t,ucenter1term)
xlabel('dimensionless time')
ylabel('centerline velocity')
legend('numerical solution','short time asymptote','one term approx')
grid on
```



Velocity Profile at Various Times

We can also plot up the velocity distribution for specific times. You will note the issue near the origin at very short times. This is known as the Gibbs ringing phenomenon and is well known in signal processing.

```
tplot = [0.0002,.05,.1,.2,.4,.8,1.6,3.2]';

uprofile = zeros(length(r),length(tplot));
for j = 1:length(tplot)
    for i=1:length(r)
        uprofile(i,j) = uinf(r(i)) + sum(a.*exp(-lambda*tplot(j)).*eigenvecs(i,:)');
    end
end
figure(3)
plot(r,uprofile,r,uinf(r))
legend(num2str(tplot),'asymptote')
xlabel('r')
ylabel('Velocity')
title('Velocity Distribution at Different Times')
grid on
```



Published with MATLAB® R2017a

Finite Difference Marching Solutions

⇒ Many transport problems can't be solved analytically, but we still need the answer! As a result, numerical solutions have been developed

⇒ Many approaches have been used: implicit/explicit finite differences, finite element, spectral methods, lattice boltzman, etc. - A vast literature!

⇒ Here we introduce the simplest method: Explicit Euler Method / finite difference marching solutions.

⇒ Useful for many problems, but particularly for parabolic PDEs: 1st order deriv. in time, 2nd (or higher) deriv. in spatial variable

⇒ Good for both linear & non-linear probs & super easy to code up!

⇒ Key Idea: Discretize spatial domain, develop finite difference approx. for time derivative at each point, use EM to march forward in time!

Apply to startup flow in a pipe - it's better to do this analytically, but this method would even work for non-linear non-Newtonian rheology!

DE: $$\frac{\partial u^*}{\partial t^*} = \frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^*\frac{\partial u^*}{\partial r^*}\right) + 1$$

$$\left.\frac{\partial u^*}{\partial r^*}\right|_{r^*=0} = 0, \quad \left.u^*\right|_{r^*=1} = 0 \qquad \left.u^*\right|_{t^*=0} = 0$$

⇒ Always render eq'ns dimensionless (and scale properly) before doing any numerical solution!!

We discretize domain in r:

$\underset{\sim}{r} = [0 : dr : 1]$ so $n = \frac{1}{dr}$ intervals

$n+1$ node locations

Discretize u:

$$u_i = u(r_i)$$

So, apart from BCs:

$$\frac{\partial u_i}{\partial t} = \left[\frac{1}{r} \frac{\partial}{\partial r}\left(r \frac{\partial u}{\partial r} \right) \right]\Bigg|_{r_i} + 1$$

So we need a finite dif. rep. for RHS!

We want a <u>center</u> <u>difference</u> approx

so error is $O(dr^2)$!

Look at i^{th} node:

u_{i-1} dr u_i dr u_{i+1}

$i-1$ $\uparrow$ i $\uparrow$ $i+1$

r_{i-1} r_i r_{i+1}

midpoints!

17-4

We have the estimate of $\frac{\partial u}{\partial r}$ at $r_i + \frac{\delta r}{2}$ which is the center of the right interval:

$$\left.\frac{\partial u}{\partial r}\right|_{r_i + \frac{\delta r}{2}} \approx \frac{u_{i+1} - u_i}{\delta r}$$

and at $r_i - \frac{\delta r}{2}$:

$$\left.\frac{\partial u}{\partial r}\right|_{r_i - \frac{\delta r}{2}} \approx \frac{u_i - u_{i-1}}{\delta r}$$

So we get:

$$\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right]_{r_i} \approx \frac{1}{r_i}\left[\left(r_i + \frac{\delta r}{2}\right)\frac{(u_{i+1} - u_i)}{\delta r} - \left(r_i - \frac{\delta r}{2}\right)\frac{(u_i - u_{i-1})}{\delta r}\right] \Big/ \delta r$$

Marching forward is easy w/ EM!

Let u_i^k be estimate of u_i at t_k

$$\therefore \; u_i^{k+1} = u_i^k + \delta t * \left\{ \left.\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right]\right|_{r_i} + 1 \right\}$$

This works for <u>all</u> <u>interior</u> <u>nodes</u>.

<u>Not</u> for first & last! Those are determined from BCs!

<u>After</u> updating interior nodes, apply BCs!

At the outer edge $u = 0$

$$\therefore \quad u_{n+1}^{k+1} = 0$$

At the inner edge $\frac{\partial u}{\partial r} = 0$

We want to use an order dr^2 approx:

$$0 = \frac{u_2 - u_1}{dr} - \frac{dr}{2}\,\frac{u_3 + u_1 - 2u_2}{dr^2}$$

deriv est. at $r = \frac{dr}{2}$

2^{nd} deriv est + shift back to $r = 0$

$$\therefore$$

$$0 = u_2 - u_1 - \frac{u_3}{2} - \frac{u_1}{2} + u_2$$

$$= 2u_2 - \frac{3}{2}u_1 - \frac{1}{2}u_3$$

or $u_1^{k+1} = \frac{1}{3}\left(4u_2^{k+1} - u_3^{k+1}\right)$

This can be coded up in just a few lines!

Stability: Like *any* explicit numerical integration method, this is *numerically unstable* if dt is too large! For this method there is the Neumann condition for stability:

$$dt < \frac{1}{2} dr^2$$

Anything bigger than this blows up!

This requires a *really* small dt for small dr!

⇒ You can avoid this by using *implicit methods* (e.g., Crank-Nicholson), but these are *much harder* to code up!

⇒ For complicated problems, use canned PDE solvers: a bit of a learning curve but very useful!

```
%This script produces a movie of startup flow in a pipe.  It uses the Euler
%method marching forward in time.

n=20; %The spatial discretization.
dr=1/n;
r=[0:dr:1];
u=zeros(size(r));
udot=zeros(size(r));

i=[2:n]; %The interior nodes
dt=0.5*dr^2; %We choose a time discretization for stability.

t=0;

while t<.5
 t=t+dt;

 % The derivative for the interior nodes
 udot(i)=1+((u(i+1)-u(i))/dr.*(r(i)+dr/2)...
    -(u(i)-u(i-1))/dr.*(r(i)-dr/2))/dr./r(i);

 u(i)=u(i)+udot(i)*dt; %We update interior nodes

 u(n+1)=0; % The velocity at the outer wall remains zero.

 % We use a zero derivative condition at the center.
 u(1)=2/3*(2*u(2)-0.5*u(3));

 figure(1)
 plot(u,r,'x',0.25*(1-r.^2),r,[0 0],[0 1])
 axis([0 .3 0 1])
 xlabel('u*')
 ylabel('r*')
 title(['Velocity Profile in a Pipe for t* = ',num2str(t)])
%  drawnow

 %Let's compare this to the exact solution!  We have the eigenvalues (roots
 %of J0, we keep ten):
sigma =[2.40482555769577
   5.52007811028631
   8.65372791291101
  11.79153443901428
  14.93091770848779
  18.07106396791092
  21.21163662987925
  24.35247153074930
  27.49347913204025
  30.63460646843198]';
 %Which yields the coefficients:
 an=-2.0./sigma.^3.0./besselj(1,sigma);
 %and the solution:
 uexact=0.25*(1-r.^2)+(an.*exp(-sigma.^2*t))*besselj(0,sigma'*r);
 %and we plot it up:
 hold on
 plot(uexact,r,'g')
 hold off

 %We can also plot up a movie of the deviation:
```

```
 figure(2)
 plot(r,u-uexact)
 xlabel('r*')
 ylabel('deviation')
 title(['Deviation from exact solution (ten eigenvalues) for t* = ',num2str(t)])
 drawnow
end

%Note that most of this stuff was the comparison...
```

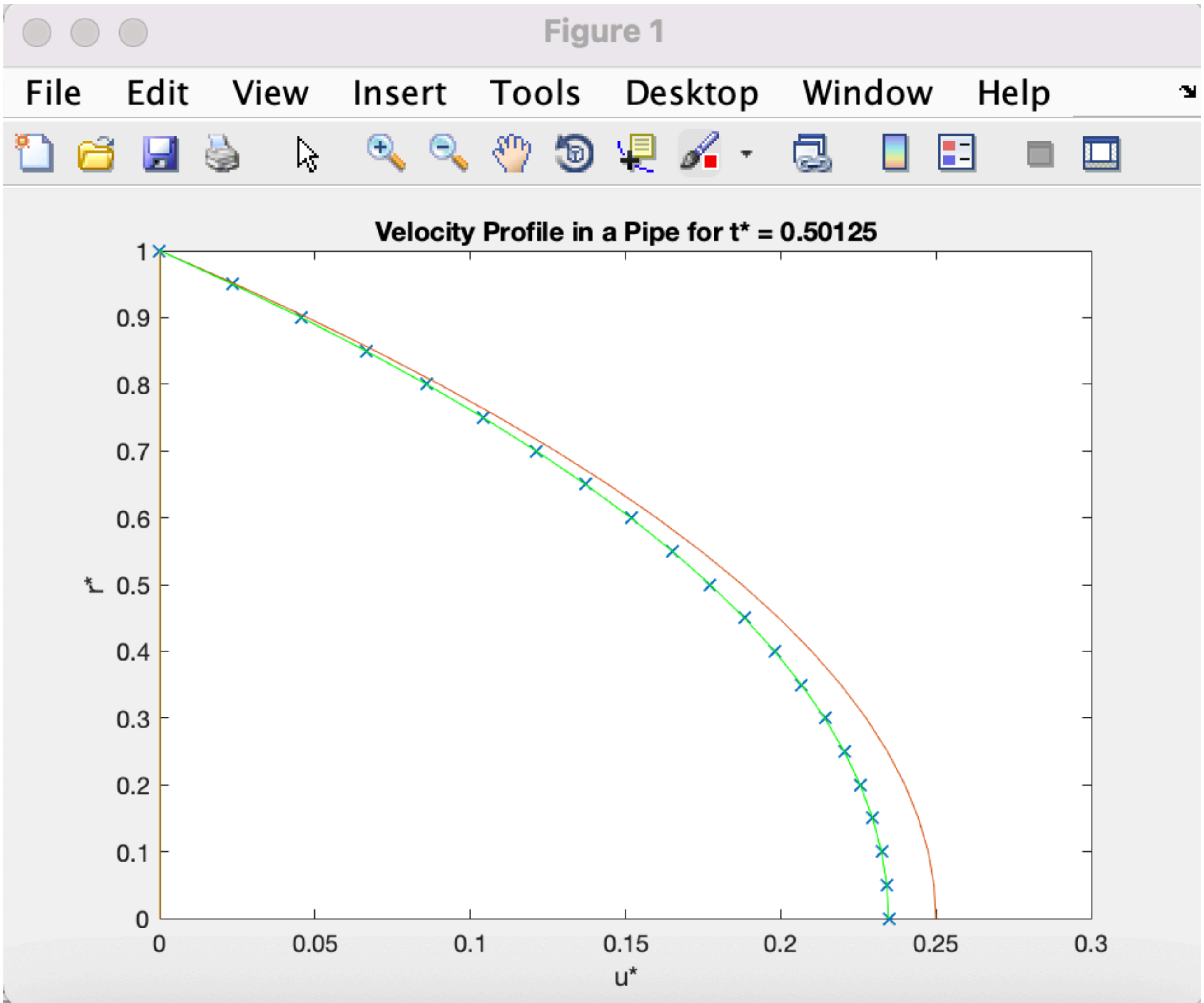


Figure 1
File Edit View Insert Tools Desktop Window Help
Velocity Profile in a Pipe for t* = 0.50125
r*
u*
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.05
0.15
0.25

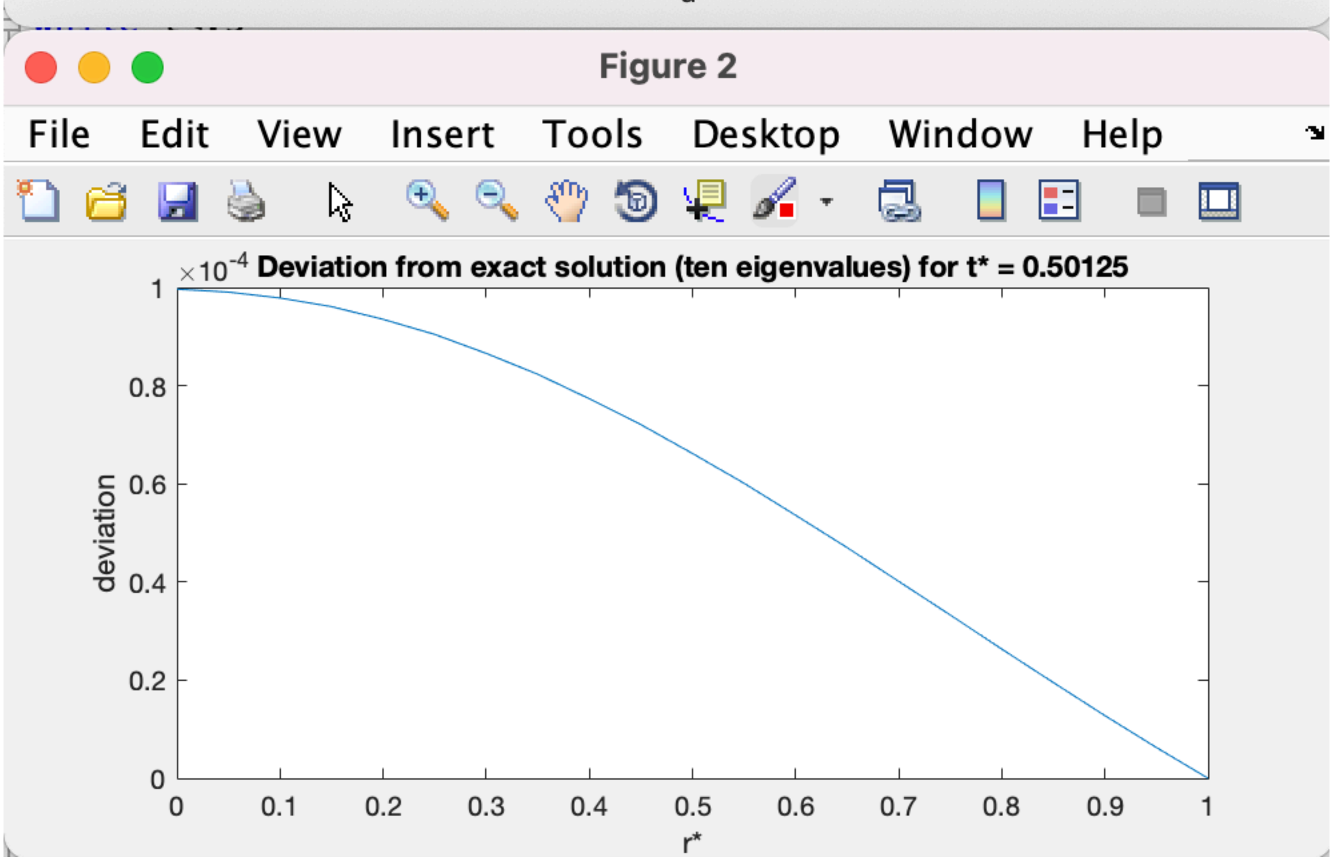


Figure 2
File Edit View Insert Tools Desktop Window Help
×10^-4 Deviation from exact solution (ten eigenvalues) for t* = 0.50125
deviation
r*
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9

Monte Carlo Solutions to Diffusion

- A <u>completely</u> <u>different</u> way to solve transient diffusion problems is via Monte Carlo simulations. It is particularly useful for linear problems with more complex domains. It's <u>computationally</u> <u>expensive</u> but <u>very</u> easy to code up! (Besides, it makes a nice movie...)

⇒ key idea: Diffusion of mass or energy can be approximated by a normally distributed random walk w/ step size w/ standard deviation of $(2D\Delta t)^{1/2}$. Thus, you just follow the motion of "tracers" (mass <u>or</u> energy) and give them a random kick at each time step!

18-2

Let's look at a simple example: a cube w/ sides $2a$ and initial temperature T_0 is cooled w/ surface temperature T_s. What is the <u>average</u> temperature of the cube as $f^{\underline{n}}(t)$?

So: $$\frac{\partial T}{\partial t} = \alpha \nabla^2 T \equiv \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right)$$

α ↳ thermal dif.

$$T\Big|_{t=0} = T_0 \qquad T\Big|_{x=\pm a} = T\Big|_{y=\pm a} = T\Big|_{z=\pm a} = T_s$$

Let's <u>scale</u>

From BCs: $$T^* = \frac{T - T_s}{T_0 - T_s}$$

and $x^* = \frac{x}{a}$, $y^* = y/a$, $z^* = \frac{z}{a}$

$$t^* = \frac{t}{t_c}$$

Plug in:

$$\frac{\Delta T}{t_c}\frac{\partial T^*}{\partial t^*} = \frac{\alpha \Delta T_c}{a^2}\nabla^{*2} T^*$$

Divide out:

$$\left[\frac{a^2}{\alpha t_c}\right]\frac{\partial T^*}{\partial t^*} = \nabla^{*2} T^*$$

$$\overset{\shortparallel}{1} \quad \underline{\text{so}} \quad t_c = \frac{a^2}{\alpha} \quad (\text{def}^{n} \text{ time})$$

$\therefore$

$$\frac{\partial T^*}{\partial t^*} = \frac{\partial^2 T^*}{\partial x^{*2}} + \frac{\partial^2 T^*}{\partial y^{*2}} + \frac{\partial^2 T^*}{\partial z^{*2}}$$

$$T^*\Big|_{t^*=1}, \quad T^*\Big|_{x^*=\pm 1} = T^*\Big|_{y^*=\pm 1} = T^*\Big|_{z^*=\pm 1} = 0$$

We can simplify using symmetry:

$$\frac{\partial T^*}{\partial x^*}\Big|_{x^*=0} = \frac{\partial T^*}{\partial y^*}\Big|_{y^*=0} = \frac{\partial T^*}{\partial z^*}\Big|_{z^*=0} = 0$$

To solve using MC, distribute N tracers uniformly over $0<x^*<1$, $0<y^*<1$, $0<z^*<1$. At each time step dt^* they get a random kick in x, y, z dir. w/ SD $= (2\,dt^*)^{1/2}$

The symmetry condition is simulated by reflection: $x^* = |x^*|$

The absorbing BC at $x^* = 1$ is even easier as you kill off tracers for $x^* > 1$ (or $y^* > 1$) etc. The average temp. is just the number left divided by N!

- This problem is implemented in the example code.

You can extend this to other BCs (say, partial reflection / absorption) and autocatalytic sources (the "bomb" problem). Convective diffusion is easy, as you just add in the convective velocity to tracer motion. Non-const (position dependent) diffusivities are a bit trickier, as you have to bias your random walk to get it right.

still, it's a fairly easy way to solve transient diffusion problems in even fairly complicated domains!

⇒ where this sort of approach is particularly useful is for analysis of separation and dispersion in convective techniques such as Field Flow Fractionation. Basically, you just follow solute molecules (or particles) in the flow and field (e.g., electric, gravity, cross-flow, etc.) and add in the Brownian kick! Very easy to simulate!

Simulation of a Quenched Cube

n thi imulation e ue onte Calo inteation to detemine the aeae temeatue of a cube a a function of time t initial temeatue i and the uface temeatue i eo e tat ith tace ditibuted oe the oitie chun of the cube eflect at the mmet lane and emoe tace hich eceed in an diection he aeae temeatue i ut the numbe of tace emainin in the domain ote that the te ie in time mut be mall enouh that the dt um i mall ith eect to the domain

```
N = 10000; % We start with lots of tracers

dt = 0.00005; % This yields a random walk step of 1%

x = rand(N,1); % The initial values of x
y = rand(N,1);
z = rand(N,1);

tfinal = .5; % How long we run it for.

tall = [0:dt:tfinal];

t = 0;
i = 1; % A counter.

isout = max(max(x,y),z); %This returns the maximum of x, y, and z for each
ikeep = find(isout<1);
nleft = N;

Tavgkeep = zeros(size(tall));

Tavg = 1; % Our initial temperature

Tavgkeep(1) = Tavg;

while nleft>0 & t<tfinal
    i = i+1;
    t = tall(i);
    x(ikeep) = x(ikeep) + (2*dt)^.5*randn(nleft,1);
    y(ikeep) = y(ikeep) + (2*dt)^.5*randn(nleft,1);
    z(ikeep) = z(ikeep) + (2*dt)^.5*randn(nleft,1);
    x = abs(x); % We reflect at zero
    y = abs(y);
    z = abs(z);
    isout = max(max(x,y),z); %This returns the maximum of x, y, and z for each
    ikeep = find(isout<1);
    nleft = length(ikeep);
    Tavgkeep(i) = nleft/N;
end
```

```
figure(1)
plot(tall, Tavgkeep)
xlabel('t')
ylabel('Average Temperature')
grid on
```