Day 20 Numerical Analysis,

Last time: Started talking about ODEs. and Numerical Solns.

given $y' = f(x,y)$
$y(x_0) = y_0$

EULERS METHOD

$y(x_0) = y_0$
$y(x_{n+1}) = y(x_n) + hf(x_n, y_n)$

FROM THE HW ⟶ what happened when you solved

$$y' = \lambda y + (1-\lambda)\cos(x) - (1+\lambda)\sin(x)$$
$$y(0) = 1$$

$\lambda = -1$ H worked $\lambda = 1$ III conditioned

$\lambda = -5$ H worked $\lambda = 5$ III conditioned.
but not for $h = .5$

There are lots of cases where Euler is not stable.

Even when well conditioned.

CONVERGENCE VS. STABILITY.

↓ as $h \to 0$ the solns $y_n \to y(x)$

↓ a small change in $y_0 \to y_0 + \epsilon$ gives small change in final soln.

Often we can show convergence using our numerical code.

A numerical method with truncation error of order two or higher is convergent if and only if it is stable.
* Comes from LAX EQUIVALENCE THEOREM.

To illistrate this
Lets consider two Model Problems:

1. $y' = \lambda y + g(x)$, $y(0) = y_0$

2. $y_\epsilon' = \lambda y_\epsilon + g(x)$, $y_\epsilon(0) = y_0 + \epsilon$
perturb the initial condition

Define $z_\epsilon(x) \equiv y_\epsilon(x) - y(x)$
change in soln due to ϵ.

To get an ode for z_ϵ we subtract $y_\epsilon' - y_\epsilon'$:

$$\left.\begin{aligned} z_\epsilon' &= \lambda z_\epsilon \\ z_\epsilon(0) &= \epsilon \end{aligned}\right\} \text{ Solving This} \longrightarrow z_\epsilon = \epsilon e^{\lambda x}$$

Use this idea

Typically for real world problems $\lambda < 0$ or $Re\{\lambda\} < 0$

$\lambda < 0$ means that the small change ϵ does not effect our soln. our method would be stable.

THE TEST

1. Solve the model problem $y' = \lambda y$, $y(0) = 1$ using the numerical method by hand.
2. See what solns do and require that $y_h(x_n) \to 0$ as $x_n \to \infty$
3. This should give a set of values $h\lambda$ for which the method converges and is stable.

If $y_h(x_n) \to 0$ as $x_n \to \infty$ for any choice of h the we have ABSOLUTE STABILITY.

Ex. Eulers Method

1 For The Model Problem:

$$y(n+1) = y_n + h\lambda y_n = (1+h\lambda) y_n$$

so
$$\begin{aligned} y(0) &= 1 \\ y(1) &= (1+h\lambda)1 = (1+h\lambda) \\ y(2) &= (1+h\lambda)(1+h\lambda) = (1+h\lambda)^2 \\ &\vdots \end{aligned}$$

$$y(n) = (1+h\lambda)^n$$

2. Require $y_h(n) \to 0$ as $n \to \infty$: $|1+h\lambda| < 1$

and for λ real and negative: $1 > |1+h\lambda| > -1$

or

$$0 > h\lambda > -2$$

$$0 < h < \frac{-2}{\lambda}$$

If λ is large then h must be small !!

From the homework: $y' = \lambda y + (1-\lambda)\cos(x) - (1+\lambda)\sin(x)$, $y(0) = 1$

when $\lambda = -5$ we need $0 < h < \frac{-2}{-5} = .4$

so $h = .5$ should not work!

Euler Method — simple and easy to program BUT we have strict restrictions on our step size to be sure that well conditioned problems converge.

Let's Talk about some improvements to this...

BACKWARD EULER METHOD:

$y' = f(x,y)$, $y(x_0) = y_0$ ⟶ Replace y' with a backward derivative.

Assume solns exist are unique and problem is well conditioned.

$$y' \approx \frac{y(x) - y(x-h)}{h}$$

$$\frac{y(x) - y(x-h)}{h} \approx f(x,y) \quad \text{or} \quad \frac{y(x_n) - y(x_{n-1})}{h} = f(x_n, y_n)$$

so we could solve for $y(x_{n-1})$ and literally march backward in time... but usually we move time forward so

$$y(x_n) = y(x_{n-1}) + hf(x_n, y_n) \quad \text{OR...}$$

Given some starting value $y(x_0) = y_0$

$$\boxed{y(x_{n+1}) = y(x_n) + hf(x_{n+1}, y_{n+1})}$$

Use the model problem to check convergence

$$y' = \lambda y \qquad y(0) = 1$$

$$y_{n+1} = y_n + h\lambda y_{n+1}$$

solve for y_{n+1} ... $y_{n+1} = (1-h\lambda)^{-1} y_n$

then $y_1 = (1-h\lambda)^{-1} y_0 = (1-h\lambda)^{-1}$

$y_2 = (1-h\lambda)^{-1}(1-h\lambda)^{-1} = (1-h\lambda)^{-2}$

$\vdots$

$y_n = (1-\lambda h)^{-n}$

when $\lambda < 0$, as $n \to \infty$ $y_n \to 0$ regardless of our choice of h
well conditioned

ABSOLUTELY STABLE
for any choice of h!

Why did we ever bother with EULER?

How would a computer solve $y' = \lambda y$, $y(0) = 1$?

$h = .1$ we can pick this as big or small as we want.
small → smooth soln curve.

$y(1) = 1$

then loop

$y(2) = y(1) + .1 f y(2)$; what is the problem?

↑ error we need to know our soln at y(2) to get our soln at y(2) !!

IMPLICIT METHOD

Finding y_{n+1} is a root finding problem,

At every x_n value... A few ways to deal with this!

A. For simple enough ODE's we can solve by hand:

Ex $y' = \lambda y + (1-\lambda)\cos(x) - (1+\lambda)\sin(x)$

Backward Euler:

$$y_{n+1} = y_n + h\left[\lambda y_{n+1} + (1-\lambda)\cos(x_{n+1}) - (1+\lambda)\sin(x_{n+1})\right]$$

solve for y_{n+1} ...

$$(1-h\lambda)\, y_{n+1} = y_n + h\left[(1-\lambda)\cos(x_{n+1}) - (1+\lambda)\sin(x_{n+1})\right]$$

and

$$y_{n+1} = \frac{y_n + h\left[(1-\lambda)\cos(x_{n+1}) - (1+\lambda)\sin(x_{n+1})\right]}{(1-h\lambda)}.$$

★ use this in your homework!

B. Use a Rootfinding method at every step

BRUTE FORCE

```
for i = 1:N
    err = 1
    while err > errtol
        solve y_{n+1} = y_n + h f(x_n, y_n)
        BISECTION
        NEWTONS
    end
end
```

Time Consuming.
Problem... if N = 100 then we are doing 100 rootfinding problems!

C. USUALLY – take one step with forward euler then use backward euler to iterate

```
for i = 1:N
    err = 1
    yold = y(i) + h f(x_i, y(i))   Euler
    while err > errtol
        ynew = y(i) + h f(x_{i+1}, yold)   Backward Euler
        err = abs(ynew - yold)
        yold = ynew
    end
    y(i+1) = yold
end.
```

Problem... now our method is no longer absolutely stable because of that first euler step.

consider the method: $y_{n+1} = y_n + h f(x_{n+1}, y_{n+1})$

vs.

the iteration: $y_{n+1}^{J+1} = y_n + h f(x_n, y_n^J)$

then error $= y_{n+1} - y_{n+1}^{J+1} = (y_n - y_n) + h\left[f(x_{n+1}, y_{n+1}) - f(x_n, y_n^J)\right]$

(error from iterating rather than using method At $J+1^{Th}$ iteration)

$$= h\left[f(x_{n+1}\, y_{n+1}) - f(x_n\, y_n^J)\right]$$

$$= h \frac{\partial}{\partial y} f(\xi, \zeta)\left[y_{n+1} - y_{n+1}^J\right] \approx h \left.\frac{\partial f}{\partial y}\right|_{x_{n+1}, y_{n+1}} \left[y_{n+1} - y_{n+1}^J\right]$$

Mean Value Thm

$x_n < \xi < x_{n+1}$ $\quad y_n < \zeta < y_{n+1}$

For error in successive terms to decrease.

In our iteration.

$$\left| h \left.\frac{\partial f}{\partial y}\right|_{x_{n+1}, y_{n+1}} \right| < 1$$

as long as h is small enough our iterations will converge.

* still restricted somewhat by h !!