$$\frac{dy}{dt} = Ay$$

$$\frac{d^2y}{dt^2} = -Sy$$

Matrices!

Lecture 18:
Ordinary Differential Equations

CS 111: Intro to Computational Science

Spring 2023

Ziad Matni, Ph.D.

Dept. of Computer Science, UCSB

Administrative

- Last homework (Assignment 9) due Wednesday
  - Remember: it's in 2 parts!

- Last Lab Thursday

- Questions about the Final Exam?

Lecture Outline

- Reviewing what we know about ODEs
- Characteristics of ODEs as Matrices
- Solving ODEs using Polynomial Approaches
    - The Runge-Kutta Methods using numpy’s **`solve_ivp()`**

Ordinary Differential Equations

- Recall: you are given a diff. equation of the form

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

- Examples:

$$\frac{dy}{dt} = ay + q(t)$$ Linear first-order D.E.

$$\frac{d^2y}{dt^2} = -ky$$ Linear second-order D.E.

Ordinary Differential Equations

- Also, recall that there are certain "forms" for answers to typical ODEs
  - But not for all of them!

- For example, an ODE of the type
  - $\mathbf{y}' = k$ has a solution of $\mathbf{y(t)} = \boldsymbol{kt + C}$
  - $\mathbf{y}' = k\mathbf{y}$ has a solution of $\mathbf{y(t)} = \boldsymbol{Ce^{At}}$
  - $\mathbf{y}'' + a\mathbf{y} = k$ has a solution of $\mathbf{y(t)} = \boldsymbol{C_1} \mathbf{\sin(\omega t)} + \boldsymbol{C_2} \mathbf{\cos(\omega t)} + \boldsymbol{k}$
  - Etc…

- But don't worry about re-memorizing this stuff
  - We have computer programs to solve these ODEs for us, after all!!!!

System of *n* ODEs with *n* unknowns

- Examples:

$$\frac{dy}{dt} = Ay$$

A and S are Matrices!

$$\frac{d^2y}{dt^2} = -Sy$$

- Notation notes:

$\dot{y}$ is the same as y' is the same as **dy/dt**

$\ddot{y}$ is the same as y'' is the same as **d^2y/dt^2**

ODEs in Matrices

- So we can express the ODE system as: **x’ = Ax**
- For example, let **A** = $\begin{bmatrix} 0 & 2 \\ 8 & 0 \end{bmatrix}$
- So: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Remember: x(t) and y(t) are functions of *t*

Therefore: $\begin{cases} x' = 2y \Rightarrow x'' = 2y' \Rightarrow y' = \frac{1}{2} x'' \\ y' = 8x \end{cases}$

Therefore:
x" – 16x = 0 (2nd order ODE)

Example Continued…

$x'' - 16x = 0$ (2nd order ODE)

- We can solve this in the following fashion *(reach back to your calculus courses!):*

$S^2 - 16 = 0$

$\Rightarrow$ $S = 4, -4$

$\Rightarrow$ The general solution to this ODE is: $\underline{\mathbf{x(t) = C_1 e^{4t} + C_2 e^{-4t}}}$

- Since $y(t) = ½\, x'(t)$

$\Rightarrow$ $\mathbf{y(t) = ½\,(4.C_1 e^{4t} - 4.C_2 e^{-4t}) =}$ $\underline{\mathbf{2.C_1 e^{4t} - 2.C_2 e^{-4t}}}$

- BTW, how might we figure out what C_1 and C_2 are??

Example Continued…

- So: with **x**' = **Ax**

the sqrt(5) factor isn't important…

$$x = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} C_1e^{4t} + C_2e^{-4t} \\ 2C_1e^{4t} - 2C_2e^{-4t} \end{bmatrix} = C_1e^{4t}\underbrace{\begin{bmatrix} 1 \\ 2 \end{bmatrix}}_{v_1} + C_2e^{-4t}\underbrace{\begin{bmatrix} 1 \\ -2 \end{bmatrix}}_{v_2}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 2 \\ 8 & 0 \end{bmatrix}$$

- Note that: $Av_1 = 4v_1$ and $Av_2 = -4v_2$
  - **So, what does that make 4 and -4, and v_1 and v_2 ????**

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

```
a = np.array([[0,2],[8,0]])
d,V = npla.eig(a)
print('evalues =', d, '\n')
print('evectors =\n', V, '\n')
print('K.evectors (where K = sqrt(5)) =\n', np.sqrt(5)*V)
```

```
evalues = [ 4. -4.]

evectors =
 [[ 0.4472136  -0.4472136 ]
 [ 0.89442719  0.89442719]]

K.evectors (where K = sqrt(5)) =
 [[ 1. -1.]
 [ 2.  2.]]
```

Eigenvalues are the values in the solution:
$y = C_1e^{\lambda 1}$ v1 + $C_2e^{\lambda 2}$ v2

Eigenvectors are the values in the solution:
$y = C_1e^{\lambda 1}$ v1 + $C_2e^{\lambda 2}$ v2

But Computing Eigen-“stuff” is Expensive…

… computationally, speaking

Is there a more efficient way to solve ODEs?

Even if it uses approximations?

(common theme in this course, isn’t it?)

https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html

Python Function: `solve_ivp()`

- Found in the `scipy` module (in `scipy.integrate`)
  - Solves an ODE with initial conditions using a method called "Runge-Kutta"

`solve_ivp(fun , t_span, y0, method)`

*i.e. Given an ODE, find **y(t)***

- **`fun`** function to solve (as a **callback**, *aka* function pointer)
- **`t_span`** range of t
- **`y0`** initial value $y_0 = y(t_0)$
- **`method`** algorithmic method type, e.g. 'RK45' (default) or 'RK23'
- *There are other options that we can ignore to default*

Runge-Kutta Methods

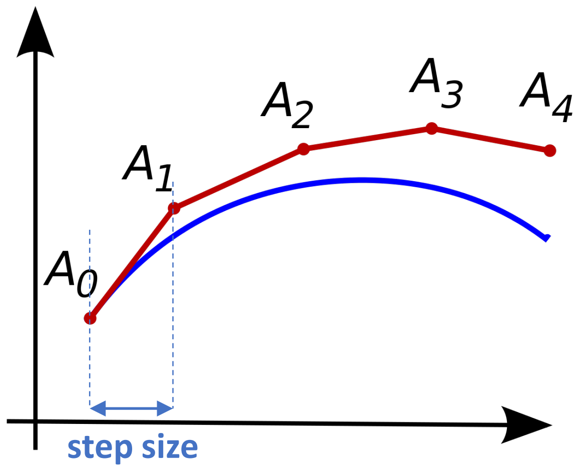
- Family of **iterative numerical methods** used in *temporal discretization* for the *approximate solutions* of *ordinary differential equations*.

- We will use them as the *primary engine* in our ODE solvers

- In **`scipy`**, **RK45** utilizes a 4th order polynomial approach, **RK23** a 3rd order one
  - There are several others that are built-in and can be used
  - Note that the Math literature will sometimes call these by other names
  - We will explore some of them with more details, in the next lesson
  - See https://docs.scipy.org/doc/scipy/reference/integrate.html for a full list

Runge-Kutta Methods

- **Iterative** numerical methods used in *temporal discretization* for the *approximate solutions* of *ordinary differential equations*.
- Precision of the approximation is determined by a **step-size**



- Nice write-up on Wikipedia: https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

Example: Solve for $y' = 0.5\,y, \quad y(0) = 2$

Preview answer:
$y(t) = 2e^{0.5t}$

- First define the function `f(t,y)`
  - Define `ydot` (i.e. y'), example: `ydot = y/2`
- Then: define `yinit` and `t_span`
  - Example: `yinit = [2]` and `tspan = (0,10)`
- Then: call the function `solve_ivp()` accordingly
  - With all its options set up correctly…
- Use the solution to **plot** a visual answer
  - You can Compare it against the actual answer, if you know it

Quick! To the Python-mobile!

Your TO DOs!

- Finish assignment 9 by Wednesday!
- Lab Thursday!

</LECTURE>