Intro to Ordinary Differential Equations

CS 111: Introduction to Computational Science

Spring 2019 Lecture #14

Ziad Matni, Ph.D.

Administrative

- Homework #6
  - Due **WEDNESDAY (5/29) @ 6:00 pm**
- Homework #7 (last one)
  - Newly issued. Due **WEDNESDAY (6/5) @ 6:00 pm**

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.

- As opposed to "Partial Differential Equations"

System of *n* ODEs with *n* unknowns

- Examples:

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

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

Matrices!

- Notation notes:

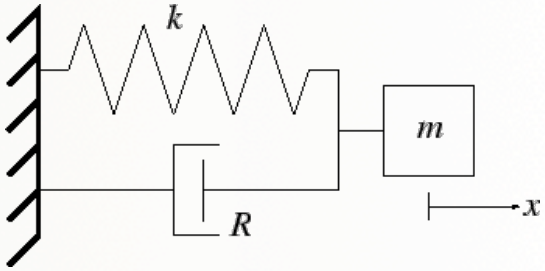
  $\dot{y}$ is the same as y'

  $\ddot{y}$ is the same as y''

Example:
The Mass-Spring Damper System

- Holds the general O.D.E: $mx'' + cx' + kx = 0$

  therefore, $x'' + (c/m)\,x' + (k/m)\,x = 0$



- Let **$y = x'$**
- So: $y' = x''$ and $x'' = -(k/m)\,x - (c/m)\,y$
- So:

$$\begin{cases} x' = y \\ y' = -(k/m)\,x - (c/m)\,y \end{cases}$$

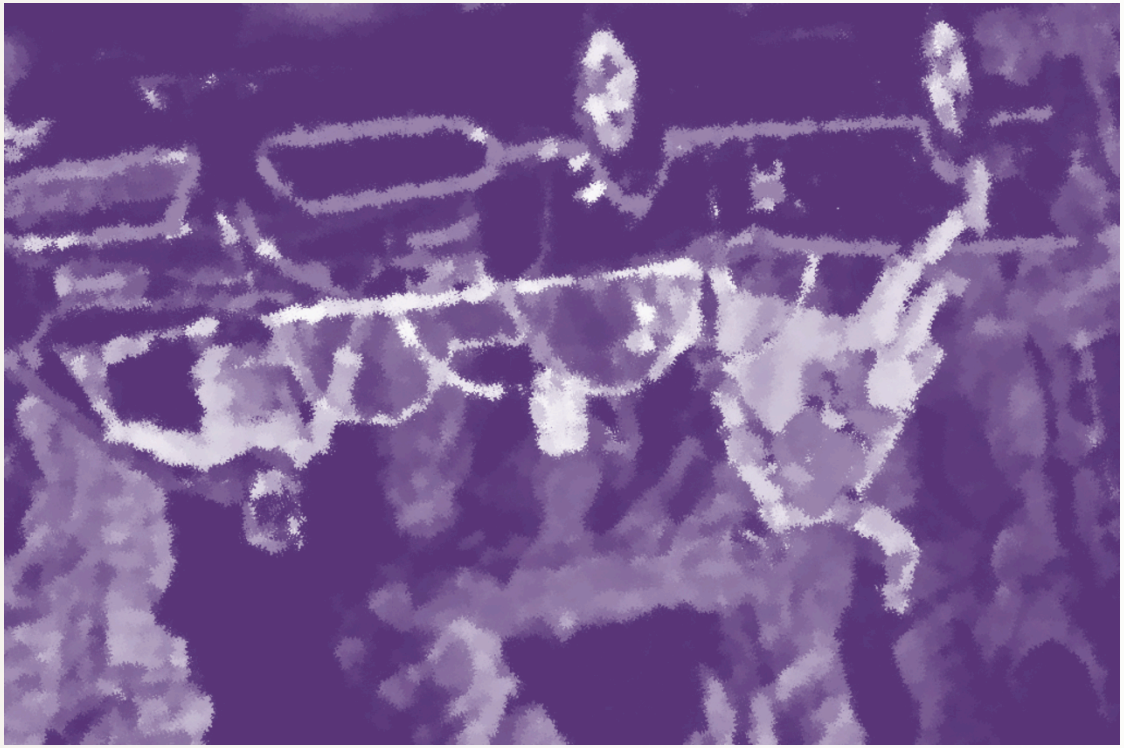
System of 2 equations w/ 2 unknowns:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-c}{m} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

ODEs in Matrices

- So we can express the ODE sytem as: $\mathbf{x}' = \mathbf{A}\mathbf{x}$
- For example, let $\mathbf{A} = \begin{bmatrix} 0 & 2 \\ 8 & 0 \end{bmatrix}$ *(not necessarily a mass-spring damper system)*

- Example continued on blackboard…



Python Function: `solve_ivp()`

- Found in the `scipy` module (in `scipy.integrate`)
  - Solves an ODE with initial conditions

- `solve_ivp(fun , t_span, y0, method)`
  - **`fun`** definition of the function to solve
  - **`t_span`** range of t
  - **`y0`** initial value $y_0 = y(t_0)$
  - **`method`** algorithmic method type (e.g. 'RK43' or 'RK23')
  - *There are other options that we can ignore to default*

Runge-Kutta Methods

- 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 numpy, **RK45** utilizes a 4^{th} order polynomial approach, **RK23** a 3^{rd} order
  - There are several others that are built-in and can be used
  - See https://docs.scipy.org/doc/scipy/reference/integrate.html for a full list
- 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**

- First define the function **`f(t,y)`**
  - Define `ydot` (i.e. y')

- Then: define `t_span` and `yinit` (i.e. y(0))
- Then: call the function `solve_ivp()` accordingly
  - With all its options set up correctly…

- Use the solution to plot a visual answer
  - Compare it against the actual answer, if you know it

Example: Non-Linear ODE

- Just as an example… we'll demonstrate

- The Lotka–Volterra equations
  - *aka* the predator–prey equations
  - Describe the dynamics of biological systems in which two species interact, one as a predator and the other as prey
  - Example: foxes vs. rabbits

Quick! To the Python-mobile!

Your To-Dos

- **Homeworks 6 and 7**

</LECTURE>