

# SESSION 2: MATLAB NUMERICS

<http://uiuc-cse.github.io/matlab-sp17/>

# OUTLINE

- Review: Matlab basics
  - Control Flow, Function
- *Example: Heat conduction*
- *Review: Function*
- Example: Radioactive decay chain
- *Example: Systems of ODE*

# FOR LOOP, IF-ELSE

- for loop :  
    **for** loop\_index = vector  
        block of code  
    **end**
- if-elseif – else statement:  
    **if** condition 1  
        do something 1  
    **elseif** condition 2  
        do something 2  
    **else**  
        do something 3  
    **end**

# RECAP - FUNCTION

- Method 1: Explicitly define a function through m file.
- Method 2: Use a function handle

(ex)

```
fun = @(x) sin(x);  
f1 = @(f,a,b,c) c*f(a+b);  
f1(fun,2,3,4)
```

# EXAMPLE: HEAT CONDUCTION (1)

- Transient heat conduction problem

Initial condition  $u(x, t=0) = \sin(\pi x/L)$ ,

Boundary conditions  $u(x=0, t) = u(x=L, t) = 0$  and length  $L = 1$

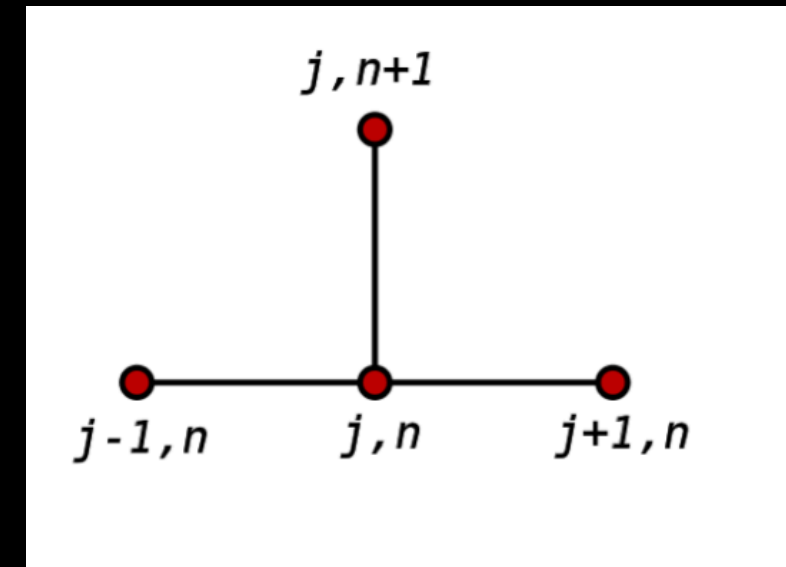
$$\frac{1}{\alpha} \frac{du}{dt} = \frac{d^2 u}{dx^2}$$

thus

$$\frac{1}{\alpha} \frac{u_i^m}{\delta t} = \frac{u_{i-1}^{m-1} - 2u_i^{m-1} + u_{i+1}^{m-1}}{\delta x^2}$$

Finite difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad \text{thus} \quad f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$



# EXAMPLE: RADIOACTIVE DECAY CHAIN (1)

- $\frac{dN}{dt} = -\lambda N$

where “N” is the number of atoms and  $\lambda$  is the decay constant of the material.

- Now, this system has an analytic solution of the following form:

$$N(t) = N_0 e^{(-\lambda t)}$$

Write a function (dNdt) to return the analytical value for N given N0,  $\lambda$ , t.

- Matlab's built-in solvers: ode23, ode45, etc...
- Compare the results visually:
  - analytical vs. ode23
  - ode23 vs. ode 45

# EXAMPLE: RADIOACTIVE DECAY CHAIN (2)

- Now let's introduce another species.
- System of Linear ODE:

$$\begin{aligned}\frac{dc_1}{dt} &= -\lambda_1 c_1 \\ \frac{dc_2}{dt} &= \lambda_1 c_1 - \lambda_2 c_2 \\ \frac{dc_3}{dt} &= \lambda_2 c_2 - \lambda_3 c_3\end{aligned}$$



Matrix Form

$$\begin{bmatrix} c_1^{(1)} \\ c_2^{(1)} \\ c_3^{(1)} \end{bmatrix} = \begin{bmatrix} -\lambda_1 & 0 & 0 \\ \lambda_1 & -\lambda_2 & 0 \\ 0 & \lambda_2 & -\lambda_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- Re-write dNdt function with vectorized equations with two given decay constants as follows:
  - $\text{lambda\_A} = \log(2);$
  - $\text{lambda\_B} = \log(2)/20;$



# SYSTEMS OF NONLINEAR ODE(1)

- System of Lorenz equations:

$$\begin{aligned}\frac{dx}{dt} &= -\sigma x + \sigma y \\ \frac{dy}{dt} &= \rho x - y - xz \\ \frac{dz}{dt} &= -\beta z + xy,\end{aligned}$$

- Solving a system of nonlinear ODE in MATLAB is quite similar to solving a single equation, but we must define a function for a system of equation as an M-file (not by function control)



# SYSTEMS OF ODE (2)

- Set the initial condition and parameters:  
 $\sigma = 10$ ,  $\beta = 8/3$ , and  $\rho = 28$ , as well as  $x(0) = -8$ ,  $y(0) = 8$ , and  $z(0) = 27$
- Define a function as m file:  
    `function xprime = lorenz(t,x);`  
    ...- Evaluate with `ode45`
  - `>>x0=[-8 8 27]; tspan=[0,20];`
  - `>>[t,x]=ode45(@lorenz,tspan,x0)`
- Plot the Lorenz strange attractor, which is a plot of  $z$  versus  $x$ :
  - `>>plot(x(:,1),x(:,3))`
- Plot each component of the solution as a function of  $t$ ,
  - `>>plot(t,x(:,1))`