Systems of Ordinary Differential Equations Differential Equations X. Du

- Useful for describing sets of dependent differential equations.
  - Example: $x' = x + y$, $y' = x - y$
  - Applications:
    - Multiple things that respond to each other
    - A spring-mass system with multiple springs and masses.
    - A mixing tank system with multiple tanks
    - A circuit with multiple closed loops
    - Economic models
- The most common systems are first-order linear systems of ODEs.
  - Homogeneous version: $\vec{x}' = A\vec{x}$ More next page
  - Non-homogeneous version: $\vec{x}' = A\vec{x} + \vec{r}(t)$ More in two pages
  - Why do only first-order systems suffice? Because higher-order linear systems can be rewritten as a first-order linear system.
  - Example: Consider the second-order linear system $x'' = x' - 2x + 2y' + 5y$, $y'' = 2x' - x + y' - y$. Now, rewrite the system representing $\dot{x} = x'$, $\dot{y} = y'$ (think of these as completely different variables with no connection to the previous variables at all). Then we have $\dot{x}' = -2x + \dot{x} + 5y + 2\dot{y}$, $\dot{y}' = -x + 2\dot{x} - y + \dot{y}$. This can be represented as the homogeneous first-order system of ODEs
    $$\begin{pmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{pmatrix}' = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & 1 & 5 & 2 \\ 0 & 0 & 0 & 1 \\ -1 & 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{pmatrix}.$$
- Non-linear systems are often not solvable by hand. Numerical methods use critical point analysis with Jacobian matrices to approximate an autonomous system near critical points.

Further notes:

- Computer programs (i.e. MATLAB) use a system of first-order equations to numerically solve higher-order differential equations using the Runge-Kutta numerical method.
  - Example: $ay'' + by' + cy = 0$ can be rewritten using $x = y'$:
    - $ax' + bx + cy = 0 \Rightarrow x' = -\frac{b}{a}x - \frac{c}{a}y$ $\quad \begin{pmatrix} x \\ y \end{pmatrix}' = \begin{pmatrix} -\frac{b}{a} & -\frac{c}{a} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
    - In this case where we're solving this numerically, *a*, *b*, and *c* don't have to be constants – they can be functions varying with *y*, so long as *a* is not 0.
- **Euler's Method** for systems of ODEs (not really different from previous)
  - Uses linear approximation. Let $\vec{r}(t)$ be a parametric vector-valued function.
  - Requires a first-order system $\vec{r}' = F(\vec{r}, t)$ and a point $\vec{r}(t_o)$ i.e. IVP
  - Uses very small steps for *dt*. $d\vec{r} = F(\vec{r}, t)dt$
  - Algorithm:
    - Calculate $d\vec{r} = F(\vec{r}, t)dt$. *F* usually involves matrix operations.
    - $t_{n+1} = t_n + dt$, $\vec{r}_{n+1} = \vec{r}_n + d\vec{r}$