$$\dot{x} = f(x), \quad x(0) = x_0$$

x - may be vector of states
f - may be nonlinear function.

$\dot{x} = Ax, \; x(0) = x_0$ is much simpler for matrix A.
system of first-order linear differential Eq's.

$x(t) = e^{At} x_0$ ... different class.

We are interested in numerically solving this, by starting with x_0 and iterating to get $x_0 \to x_1 \to x_2 \to \dots \to x_N$. (trajectory)

Forward Euler:

$$\frac{x_{k+1} - x_k}{\Delta t} \approx \dot{x} = f(x_k) \implies \boxed{x_{k+1} = x_k + \Delta t \, f(x_k)}$$

$$\text{If } \dot{x} = Ax \implies \boxed{x_{k+1} = (I + \Delta t A) x_k}$$

(I: identity matrix.)

(not very stable)

Backward (Implicit) Euler:

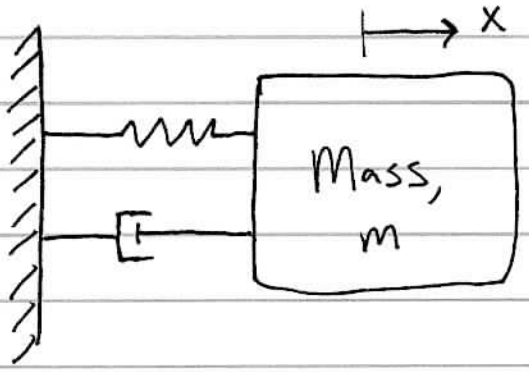
$$\frac{x_{k+1} - x_k}{\Delta t} \approx f(x_{k+1}) \implies x_{k+1} = x_k + \Delta t \, f(x_{k+1})$$

$$\text{if } \dot{x} = Ax \implies x_{k+1} = x_k + A \Delta t \, x_{k+1}$$

$$\implies \boxed{x_{k+1} = (I - A\Delta t)^{-1} x_k}$$

better stability.

Spring - Mass - Damper



$$m\ddot{x} = -Kx - c\dot{x}$$

$$m\ddot{x} + Kx + c\dot{x} = 0$$

$$\ddot{x} + \frac{K}{m}x + \frac{c}{m}\dot{x} = 0$$

If $\omega_0 = \sqrt{\frac{K}{m}}$ natural frequency

$\zeta = \frac{c}{2\sqrt{Km}}$ damping ratio.

$$\boxed{\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0}$$

Second order <u>linear</u> differential equation.

$$\left.\begin{aligned}\dot{x} &= v \\ \dot{v} &= -2\zeta\omega_0 v - \omega_0^2 x\end{aligned}\right\} \quad \frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix}}\begin{bmatrix} x \\ v \end{bmatrix}$$

ω_0 & ζ determine eigenvalues of A, hence, the behavior of the system.

Cases: ① Under-damped $\zeta < 1$

system oscillates w/ freq $\omega_d = \omega_0 \sqrt{1-\zeta^2}$

② Over-damped $\zeta > 1$

③ Critically Damped $\zeta = 1$

Lets code up forward Euler

$$x_{k+1} = (I + A\Delta t)\, x_k$$

... try $dt = .01$ $T = 10$

... compare w/ RK4

... try $dt = 0.1$, $d = 0.5$, $d = 1$, $d = 2$.

What went wrong?..

Look at $eig(I + A\Delta t)$.

Next time: • error analysis

• global stability properties

• why is ODE45 so good?

```
clear all

w = 2*pi;
d = .5;  % will break for d=20

A = [0 1; -w^2 -2*d*w];

dt = .01;  % time step
T = 10;    % amount of time to integrate

x0 = [2; 0];  % initial condition

% iterate forward euler
x(:,1) = x0;
t1(1) = 0;
for i=1:T/dt
    t1(i+1) = i*dt;
    x(:,i+1) = (eye(2) + A*dt)*x(:,i);
end

plot(t1,x(1,:))

% compute better integral using build-in Matlab code
[t,y] = ode45(@(t,y) A*y, 0:dt:T,x0);
hold on
plot(t,y(:,1),'r')
```

```
clear all

w = 2*pi;
d = 1.75;  % will break for d=20

A = [0 1; -w^2 -2*d*w];

dt = .1;  % time step
T = 10;    % amount of time to integrate

x0 = [2; 0];  % initial condition

% iterate forward euler
xF(:,1) = x0;
tF(1) = 0;
for i=1:T/dt
    tF(i+1) = i*dt;
    xF(:,i+1) = (eye(2) + A*dt)*xF(:,i);
end
plot(tF,xF(1,:),'k')
hold on

% iterate backward euler
xB(:,1) = x0;
tB(1) = 0;
for i=1:T/dt
    tB(i+1) = i*dt;
    xB(:,i+1) = inv(eye(2)-A*dt)*xB(:,i);
end
plot(tB,xB(1,:),'b')

% compute better integral using build-in Matlab code
[t,y] = ode45(@(t,y) A*y, 0:dt:T,x0);
hold on
plot(t,y(:,1),'r')
```