Introduction to Numerical Integration

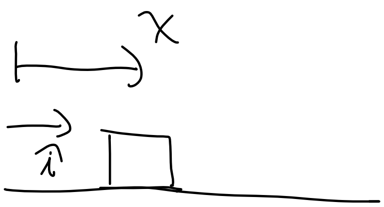
Tuesday, September 8, 2020 2:50 PM

- Euler's method
- ode45
- animation

EOM

analytical: $Z(t) \rightarrow \vec{r}(t), \vec{v}(t)$

Numerical: $Z(t) \rightarrow$ plots/animation



$$Z = [x]$$

$x_i \quad t_i$

$\dot{x}_i$

$$x_{i+1} = x_i + \dot{x}_i (t_{i+1} - t_i)$$

$$\boxed{Z_{i+1} = Z_i + \dot{Z}_i \Delta t}$$ Euler's Method

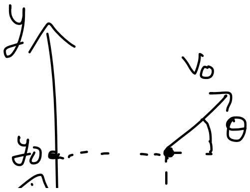
Z_i (initial state)

$\dot{Z}(t)$ rate of change of state

$Z(t)$

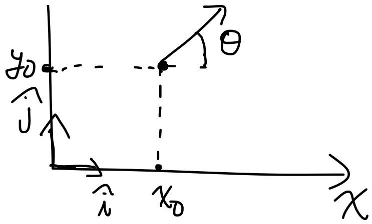
---

e.g.



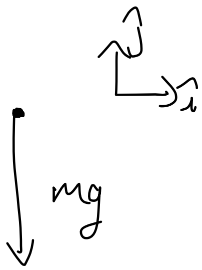
$\downarrow g$

goal: find position & velocity over time & "do interesting things" with that info



time & "do interesting things" with that information

FBD



$$\sum \vec{F} = m\vec{a}$$

$$-mg\hat{j} = m(\ddot{x}\hat{i} + \ddot{y}\hat{j})$$

$$\boxed{\begin{aligned}\ddot{x} &= 0 \\ \ddot{y} &= -g\end{aligned}}$$

Analytical Solution

$$x(t) = V_0\cos(\theta)t + x_0$$

$$y(t) = -\frac{gt^2}{2} + V_0\sin(\theta)t + y_0$$

Numerical Solution

$$z_{i+1} = z_i + \dot{z}_i \Delta t \qquad z = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \qquad \dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix}$$

$$z_{next} = z_{now} + \dot{z}\Delta t \quad ?$$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \dot{x}_{i+1} \\ \dot{y}_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \ddot{x}_i \\ \ddot{y}_i \end{bmatrix} \Delta t \qquad z_0 = \begin{bmatrix} x_0 \\ y_0 \\ V_0\cos(\theta) \\ V_0\sin(\theta) \end{bmatrix}$$

$$\ddot{x} = 0$$

$$\ddot{y} = -g$$

$$\ddot{y} = -g$$

code setup

Master file

- Define parameters (g, m, L, etc)
- Define time array
- Define initial conditions
- call integrator (myEulerSolver / ode45)
- post-processing (plotting & animating)

myrhs (function)

- Unpack parameters
- calculate $\dot{z}$ using EOM

myEulerSolver

- create z array
- Begin loop
- Define z for each time step
- call myrhs function to get $\dot{z}$
- calculate new z from $\dot{z}$ & old z
- end loop

anan-fcns

anon-fcns

p = struct(. .)

myfun

in: (t, z, p)

% does stuff

out: $\dot{z}$

f = @(t,z) myfun(t,z,p)

f

in: (t, z)

$\dot{z}$ = myfun(t, z, p)

out: $\dot{z}$