Prediction Model Simulation Using Runge Kutta (RK) Method

Presenter:
Mohamed W. Mehrez, PhD

June, 2020

- **Model Predictive Control for** (Differential drive robots – point stabilization)

system model

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t)) \qquad \mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \xrightarrow[\text{Sampling Time } (\Delta T)]{\text{Euler Discretization}} \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \Delta T \begin{bmatrix} v(k)\cos\theta(k) \\ v(k)\sin\theta(k) \\ \omega(k) \end{bmatrix}$$

MPC controller

Running (stage) Costs: $\ell(\mathbf{x}, \mathbf{u}) = \left\| \mathbf{x}_\mathbf{u} - \mathbf{x}^{ref} \right\|_\mathbf{Q}^2 + \left\| \mathbf{u} - \mathbf{u}^{ref} \right\|_\mathbf{R}^2$

Optimal Control Problem (OCP):

$$\underset{\mathbf{u}\,\text{admissible}}{\text{minimize}}\, J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k))$$

$$\text{subject to}: \mathbf{x}_\mathbf{u}(k+1) = \mathbf{f}(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k)),$$

$$\mathbf{x}_\mathbf{u}(0) = \mathbf{x}_0,$$

$$\mathbf{u}(k) \in U, \quad \forall k \in [0, N-1]$$

$$\mathbf{x}_\mathbf{u}(k) \in X, \quad \forall k \in [0, N]$$

- **OCP and NLP[1]**

OCP

$$\min_{\mathbf{u}} J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k))$$

$$\text{s.t.}: \mathbf{x}_\mathbf{u}(k+1) = \mathbf{f}(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k)),$$

$$\mathbf{x}_\mathbf{u}(0) = \mathbf{x}_0,$$

$$\mathbf{u}(k) \in U, \ \forall k \in [0, N-1]$$

$$\mathbf{x}_\mathbf{u}(k) \in X, \ \forall k \in [0, N]$$

Shooting Methods:
1- Single Shooting
2- Multiple shooting
3- .....

NLP

$$\min_{\mathbf{w}} \Phi(\mathbf{w})$$

$$\text{s.t.} \mathbf{g}_1(\mathbf{w}) \leq 0,$$

$$\mathbf{g}_2(\mathbf{w}) = 0,$$

[1]Joel Andersson (2015)

- **Model simulation (integration)**

system model

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t))$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$$

Euler Discretization

Sampling Time (ΔT)

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$$

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \Delta T \begin{bmatrix} v(k)\cos\theta(k) \\ v(k)\sin\theta(k) \\ \omega(k) \end{bmatrix}$$

Exact Discrete Model

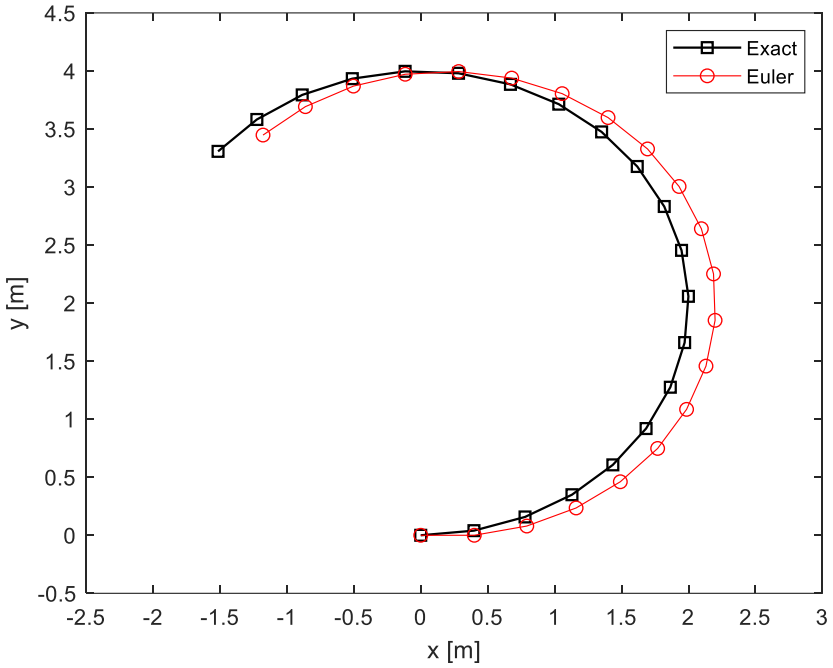
$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$$

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \frac{v(k)}{\omega(k)}\left(\sin(\theta(k) + \Delta T\omega(k)) - \sin(\theta(k))\right) \\ \frac{v(k)}{\omega(k)}\left(\cos(\theta(k)) - \cos(\theta(k) + \Delta T\omega(k))\right) \\ \Delta T\ \omega(k) \end{bmatrix}$$

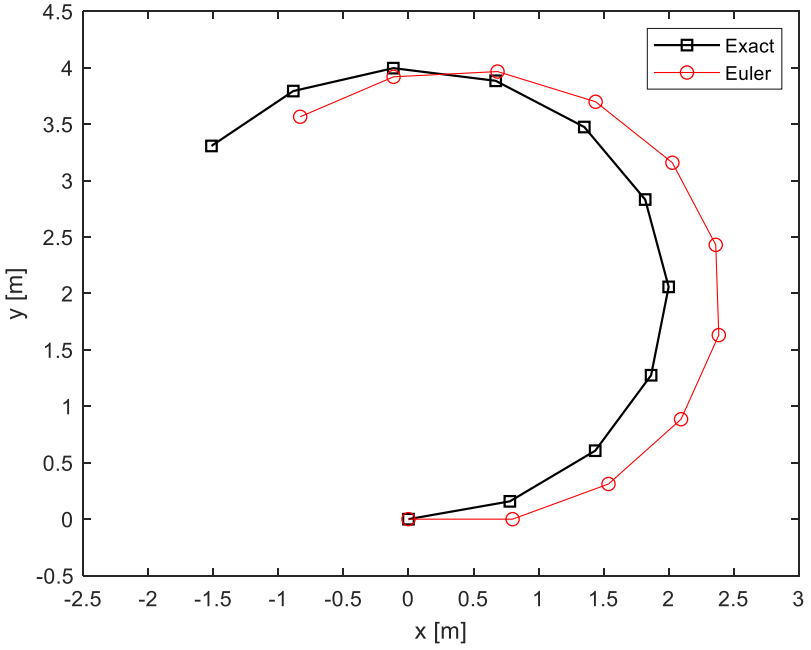
- **Model simulation (integration)**

```
v =2; %[m/s]
omega =1; %[rad/s]
```

```
h =0.2; %[s] sampling time
```




```
h =0.4; %[s] sampling time
```



- **Model simulation (integration)**

Runge-Kutta 4th (RK4) Order Method to Solve Differential Equations

$$\dot{x} = f(x, u(t))$$

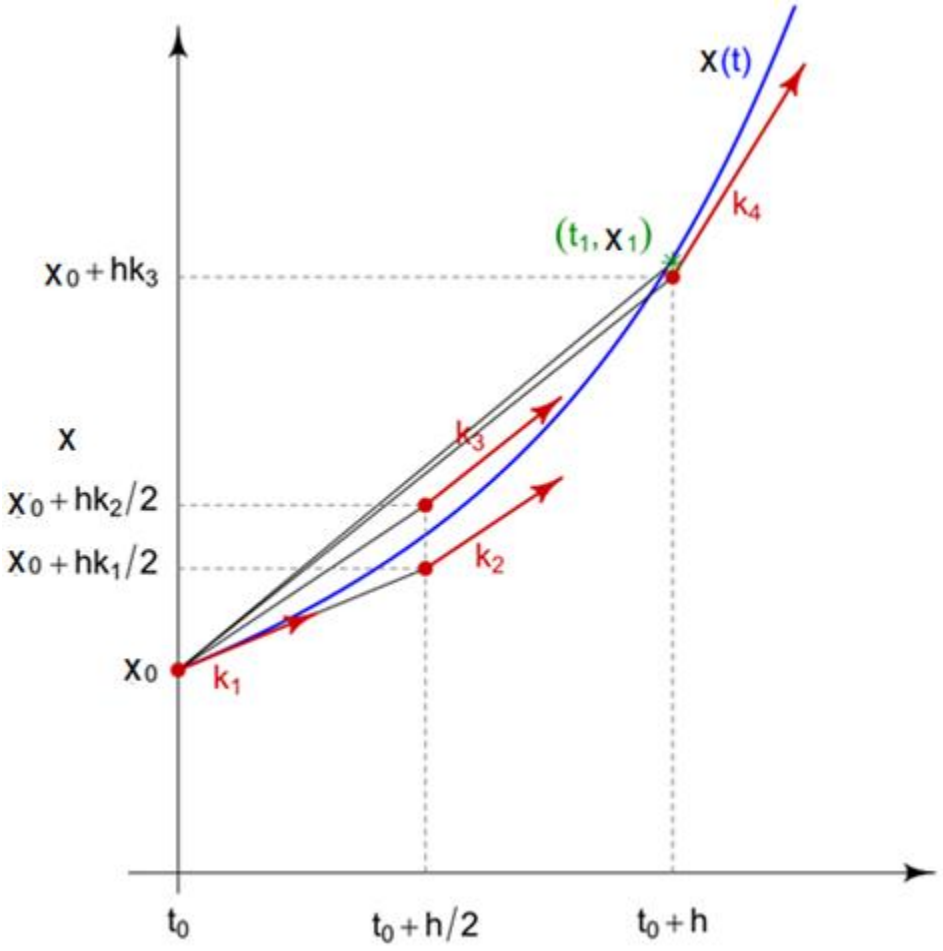
$$k_1 = f(x_{n-1}, u)$$

$$k_2 = f\left(x_{n-1} + \frac{h}{2}k_1, u\right)$$

$$k_3 = f\left(x_{n-1} + \frac{h}{2}k_2, u\right)$$

$$k_4 = f(x_{n-1} + hk_3, u)$$

$$x_n = x_{n-1} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$



$$x_n = x_{n-1} + h * (k_1)$$ **Euler Discretization**

Source: WikiPedia

- **Model simulation (integration)**

Runge-Kutta 4th Order Method to Solve Differential Equations (implementation with Casadi)

```
h =0.4; %[s]
x = SX.sym('x'); y = SX.sym('y'); theta = SX.sym('theta');
states = [x;y;theta]; n_states = length(states);
v = SX.sym('v'); omega = SX.sym('omega');
xdot = [v*cos(theta);v*sin(theta);omega]; % system r.h.s
f = Function('f',{states,controls},{xdot}); % nonlinear mapping function f(x,u)
```

$$\dot{x} = f(x, u(t))$$

```
sim_time = 4; % seconds
time = [0:h:sim_time];

st_exact = [0; 0; 0];  con = [2; 1];
st_RK4 = st_exact;
st_traj_RK4 = [st_RK4];

for k = 1: length(time)-1
    k1 = f(st_RK4, con);
    k2 = f(st_RK4 + h/2*k1, con);
    k3 = f(st_RK4 + h/2*k2, con);
    k4 = f(st_RK4 + h*k3, con);
    st_RK4=st_RK4+h/6*(k1 +2*k2 +2*k3 +k4);
    st_traj_RK4 = [st_traj_RK4, full(st_RK4)];
end
```

$$k_1 = f(x_{n-1}, u)$$

$$k_2 = f\left(x_{n-1} + \frac{h}{2}k_1, u\right)$$

$$k_3 = f\left(x_{n-1} + \frac{h}{2}k_2, u\right)$$

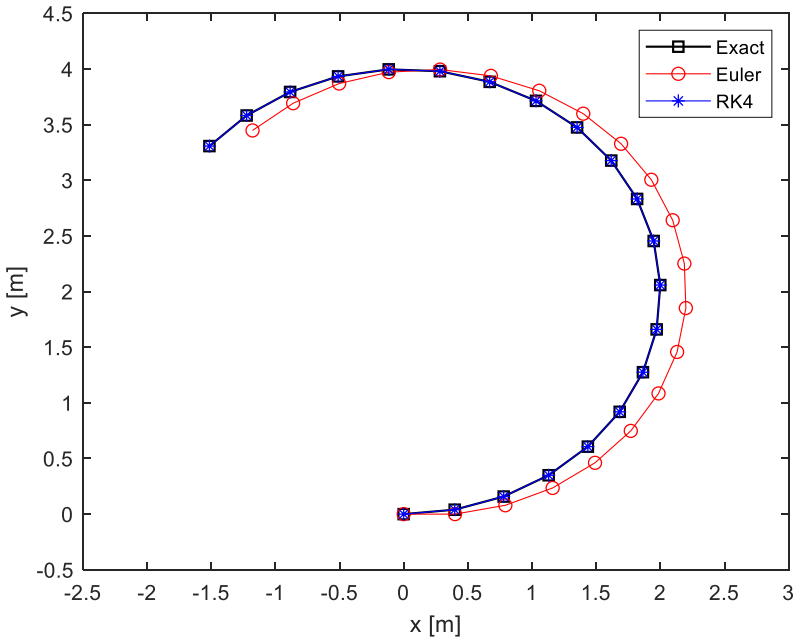
$$k_4 = f(x_{n-1} + hk_3, u)$$

$$x_n = x_{n-1} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

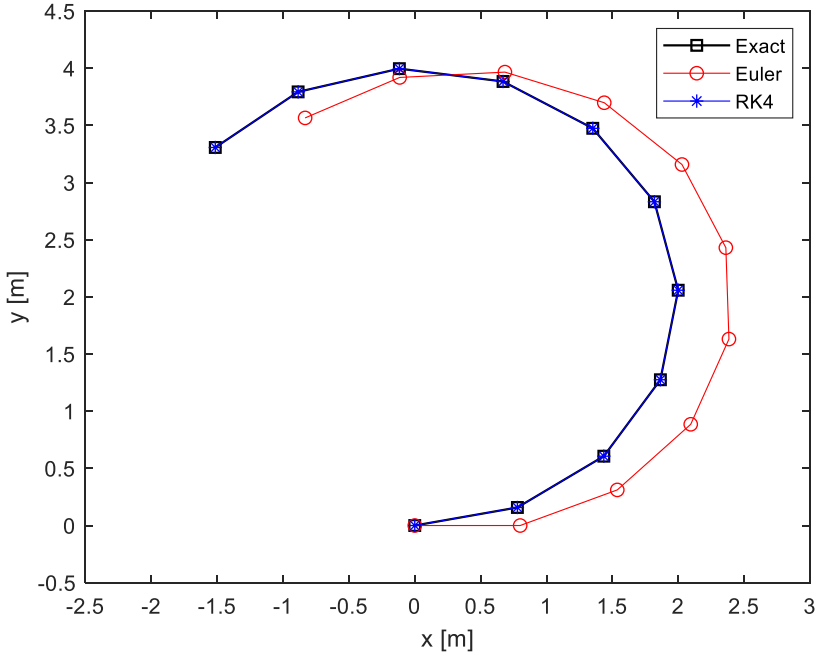
- **Model simulation (integration)**

Runge-Kutta 4th Order Method to Solve Differential Equation (implementation with Casadi)

```
h =0.2; %[s]
```




```
h =0.4; %[s]
```

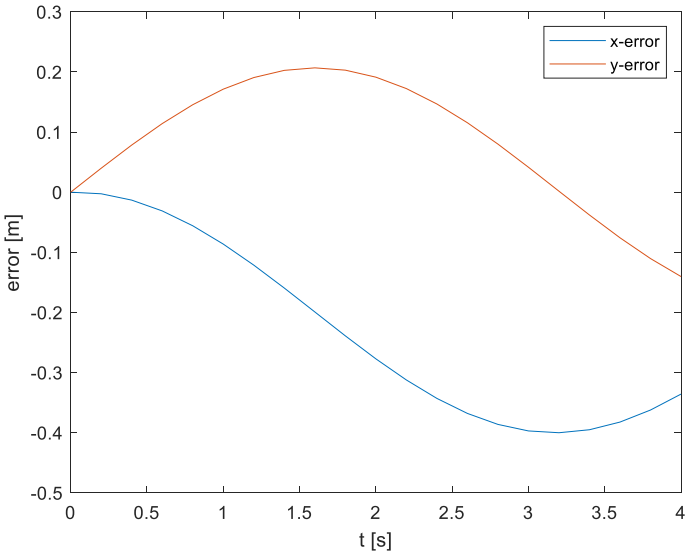


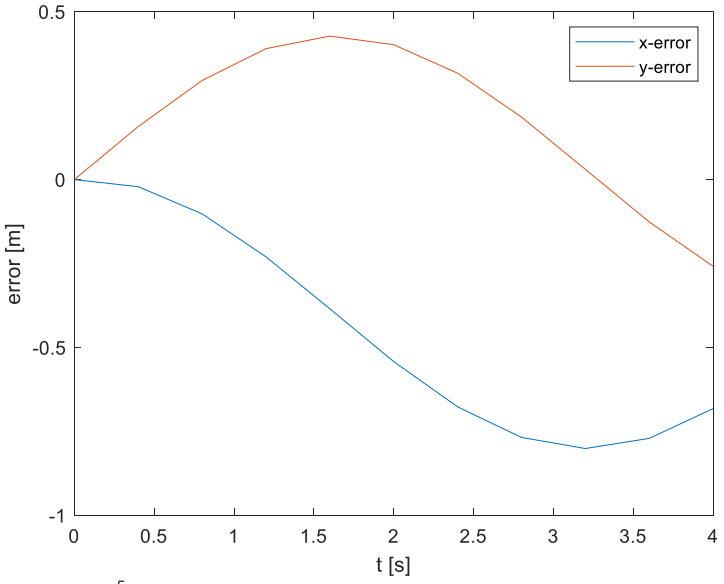
- **Model simulation (integration)**

`h =0.2; %[s]`

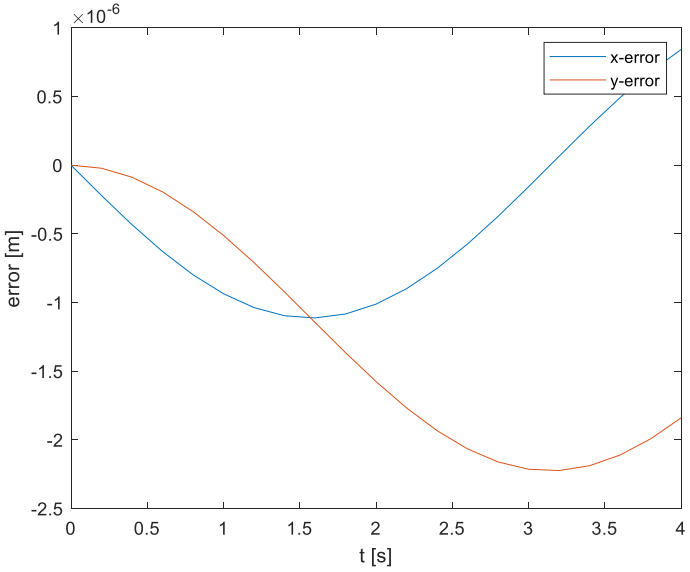
`h =0.4; %[s]`

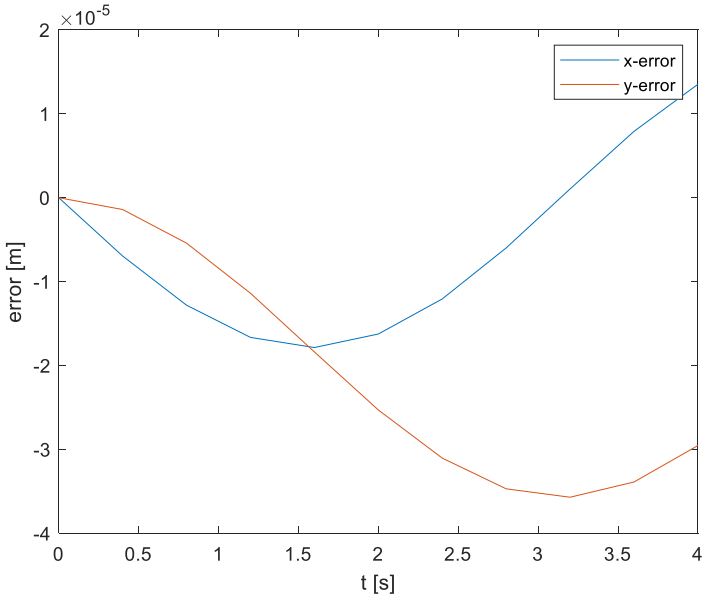
Euler





RK4





Demo

Sim_2_MPC_Robot_PS_mul_shooting_RK.m