Robótica Móvel

Exercises with Generic Kinematics of Wheeled Robots

Main purpose

- Simulate wheeled robot locomotion after their kinematics equations
- Robots to use as examples
    - Differential drive
    - Tricycle
    - Omnidirectional

Principle

- Use the robot kinematics expression in the world (global) frame, whatever they may be in each case.
- Simulate time by discrete iteration (e.g. Δt=1 or any convenient factor, like Δt=0.01 for much finer control!)
- Calculate each new position during the simulation:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix}$$

$$x(n+1) = x(n) + \dot{x}\Delta t$$
$$y(n+1) = y(n) + \dot{y}\Delta t$$
$$\theta(n+1) = \theta(n) + \dot{\theta}\Delta t$$

Reference frames

- To convert local robot velocities into global frame velocities you need to apply the Orthogonal Rotation Matrix : R(θ)

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- V_X, V_Y, ω - are the robot local velocity parameters
- To calculate the global velocities, apply the following:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & -\sin\theta(t) & 0 \\ \sin\theta(t) & \cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_X(t) \\ V_Y(t) \\ \omega(t) \end{bmatrix}$$

Differential Drive Robot Specification

- r - Wheel radius
- L - Wheel separation
- ω_L - left wheel angular velocity
- ω_R - right wheel angular velocity

$$\begin{bmatrix} V_X(t) \\ V_Y(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix}$$

Tricycle Robot Specification

- r - radius of steering wheel
- ω_S - steering wheel angular velocity
    - $V_s = \omega_S r$ - linear velocity of steering wheel (front wheel)
- L - Distance from steering wheel to back wheels
- α - Steering angle of front wheel

$$V_X(t) = V_S(t)\cos\alpha(t)$$
$$V_Y(t) = 0$$
$$\omega(t) = \frac{V_S(t)}{L}\sin\alpha(t)$$

Omnidirectional Drive Robot Specification

- r - wheel radius
- L - distance of wheels to robot center
- ω_i - angular velocity of wheel number i (i=1,2,3)

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & \frac{r}{\sqrt{3}} & -\frac{r}{\sqrt{3}} \\ -\frac{2r}{3} & \frac{r}{3} & \frac{r}{3} \\ \frac{r}{3L} & \frac{r}{3L} & \frac{r}{3L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

1a - Create basic functions in Matlab - orm()

function R=orm(theta)

Returns the orthogonal rotation matrix for angle theta

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1b-Create basic functions in Matlab - localvels()

```
function [Vx,Vy,w]=localvels(t,r,L,w1,aw2,w3)
% Vx, Vy, w - velocities in local frame
% t   - type of robot:
%     1 - DD
%     2 - TRI
%     3 - OMNI
% r   - traction wheel radius
% L   - meaning depending on type (wheel sep/1, wheel dist/2, wheel dist/3)
% w1 - angular vel of wheel 1 (Right wheel/1, steering/2)
% aw2- angular vel of wheel 2 (left/1) or alpha/2
% w3 - angular vel of wheel 3 (OMNI)
```

Valid for all three types of kinematics, where the parameter **t** determines what type of robot is asked and parameter 5 (**aw2**) will mean an angular velocity **w2** or a steering value **a** , and also determine the precise meaning of **r** and **L** in each case.

Returns the 3 velocities of the robot in its local frame.

2-Generate and plot paths for some local velocities laws

- Try several wheel angular velocities combinations as suggested ahead
- To build a path you must create a core loop for all simulation steps (ST)
- Inside this loop, you must perform some operations:
  - Calculate local velocities (Local Kinematics) - use the **localvels**() function
  - Convert local velocities to global Frame - using the **orm**() function
  - Successive global positions of path are the accumulation of previous positions with the product of instantaneous velocities by the time interval (Dt)
- Before entering the loop, you must define:
  - The number of simulation steps (ST) and the simulation time interval (Dt)
  - A starting point and orientation (x0, y0, th0)
  - The robot parameters (r,L, type of robot)
  - The wheel velocity laws (w1,aw2,w3) that can be fixed or change with time

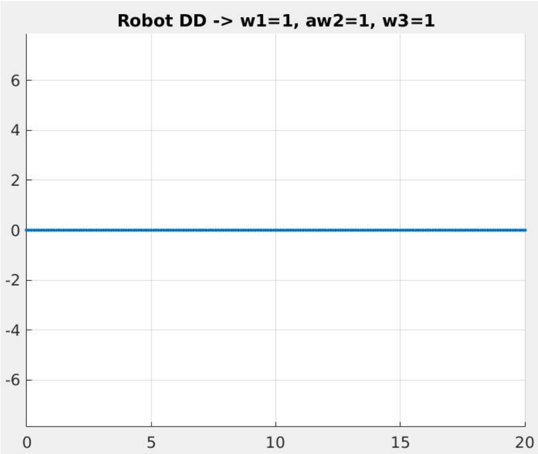
2a) - Path example 1 - similar wheel parameters

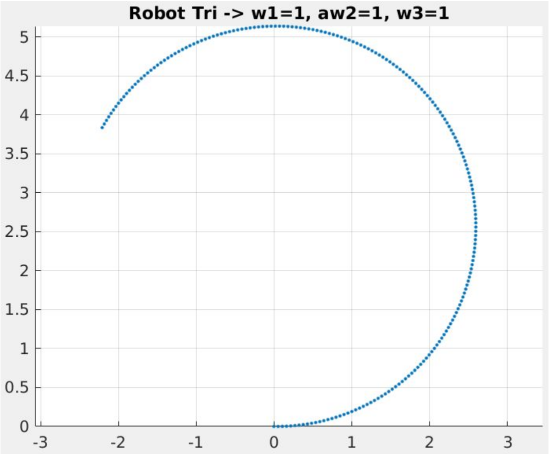
```
ST=200;
r=1;
L=4;
P0=[0;0];
th0=0;
Dt=0.1;
```

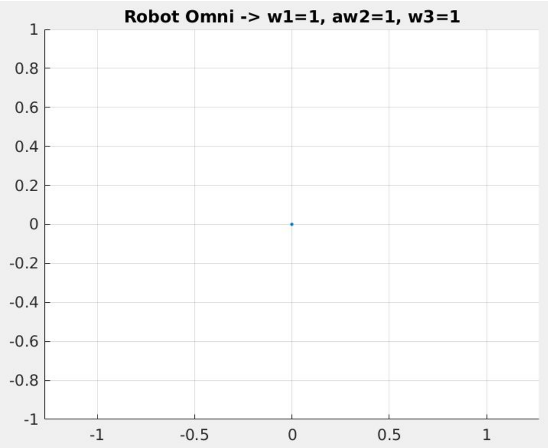
All wheel “velocities” are constant for the three robot models:

w1=1; aw2=1; w3=1

Remind the meaning of aw2 for the tricycle!

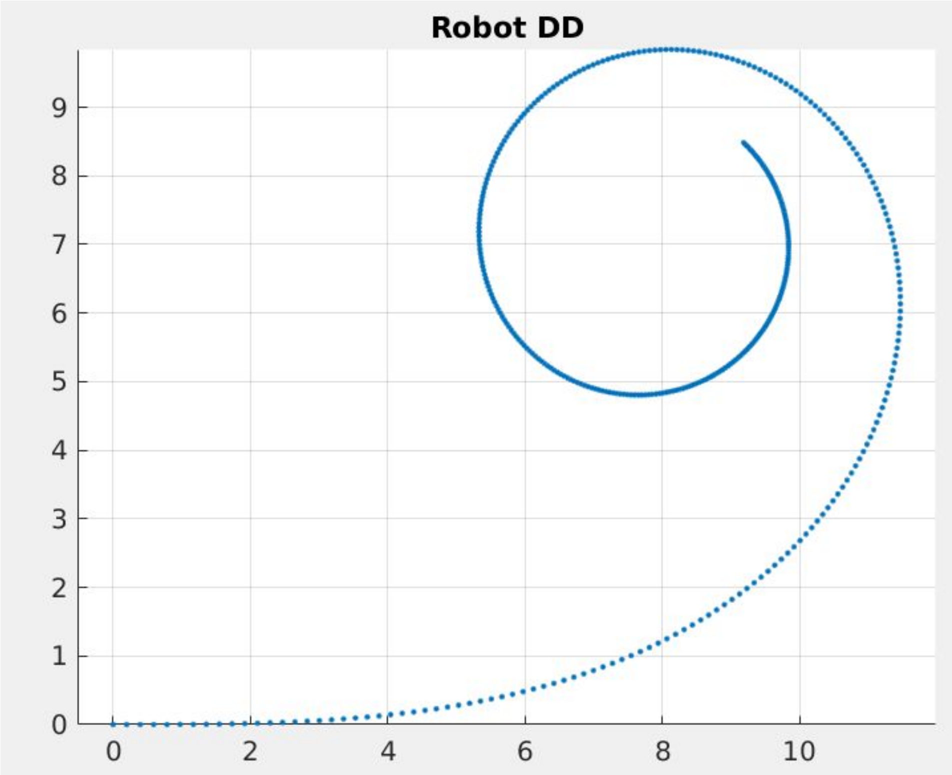






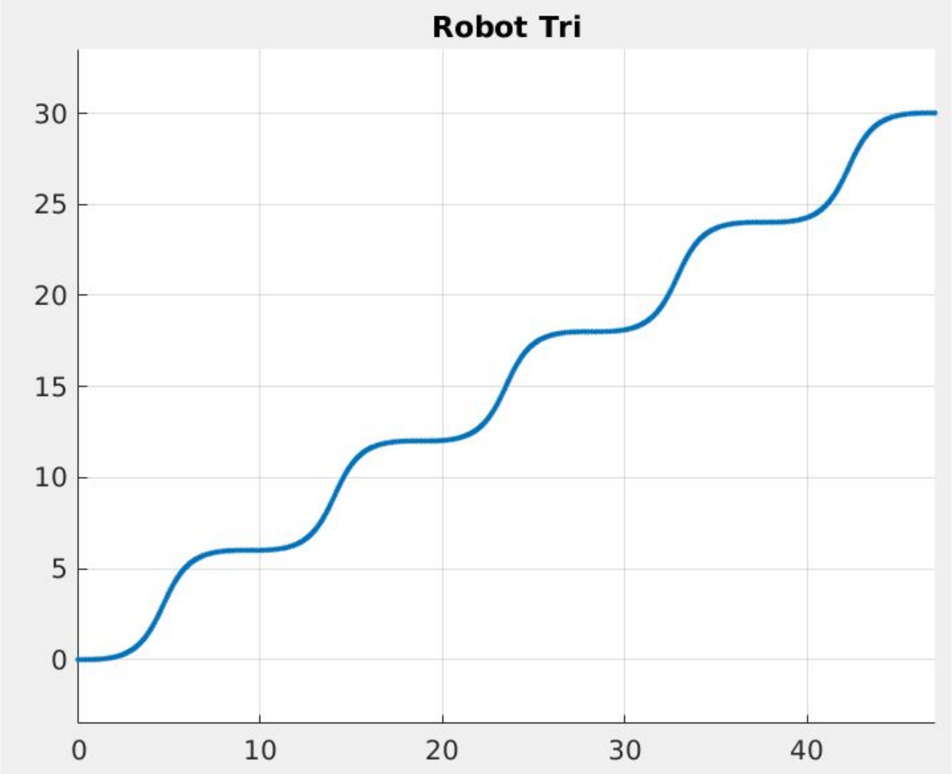
2b) - Path example 2 - DD with exponential decaying

```
ST=400;
r=1;
L=4;
P0=[0;0];
th0=0;
Dt=0.1;


w1 =2*exp(-n/ST);
aw2=2*exp(-5*n/ST);
```



2c) - Path example 3 - tricycle with shaking steering

```
ST=400;
r=1;
L=4;
P0=[0;0];
th0=0;
Dt=0.1;


w1=2;
aw2=sin(10*n*pi/ST)
```



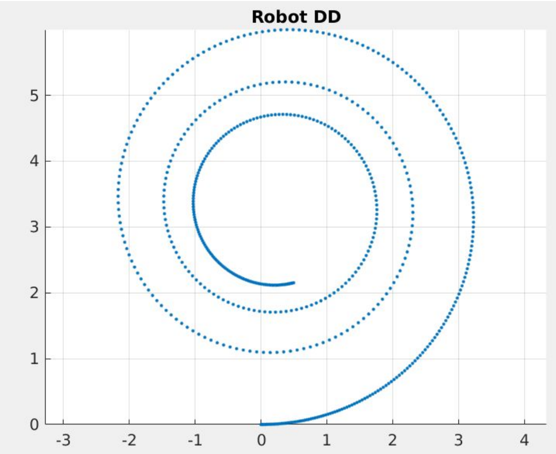
2d) - Paths example 4 - sinusoidal wheel velocities

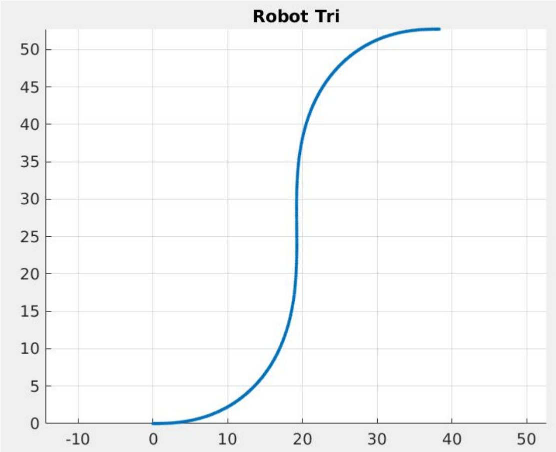
```
ST=600;
r=1;
L=4;
P0=[0;0];
th0=0;
Dt=0.1;
```

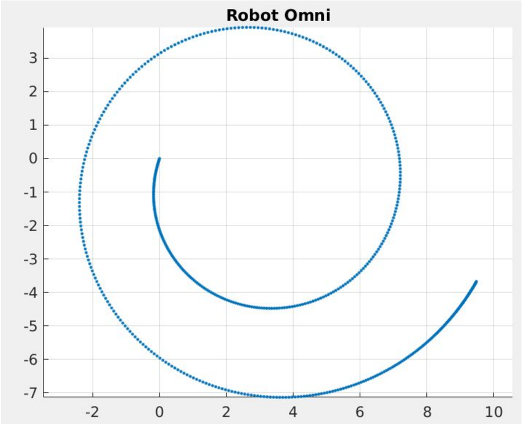
Wheel “velocities” are variable like sinus curves :

- `w1  = 2*sin(n*pi/ST);`
- `aw2= 0.25*sin(2*n*pi/ST);`
- `w3  = sin(n*pi/ST);`

Remind the meaning of aw2 for the tricycle!

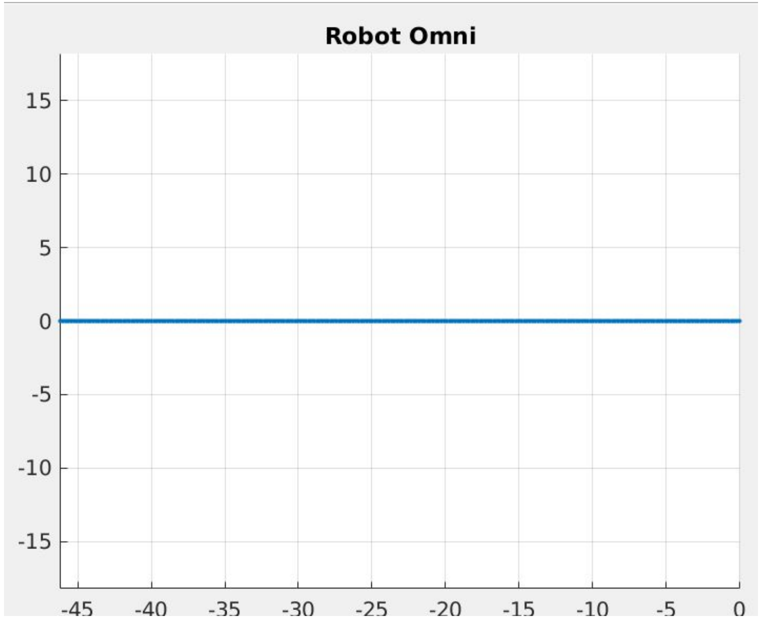


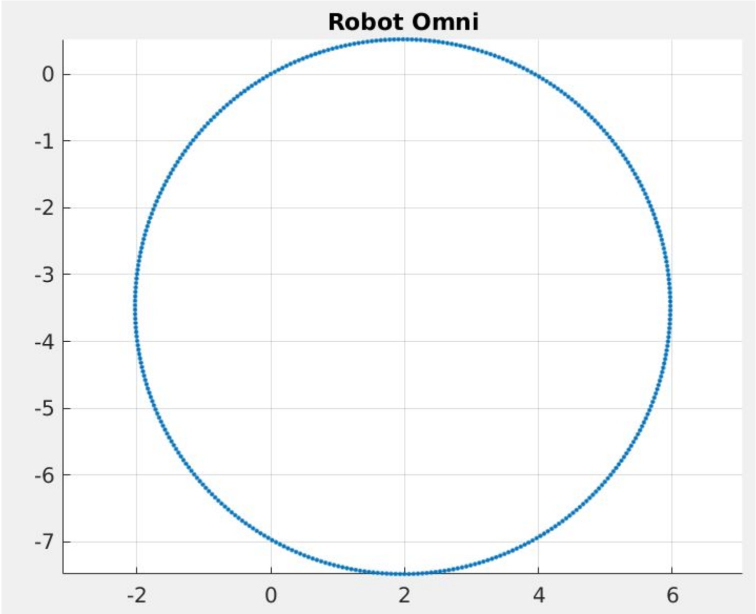




3) - Challenge with Omni kinematics

Which combinations of the 3 wheels velocities of the Omni robot can produce the paths below?

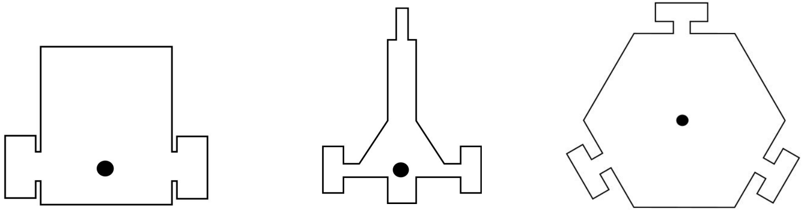




4 - Create graphic representations of robots

To better illustrate the simulated motion, suggestive representations of each of the 3 types of robots can be done.

- DD - Differential Drive
- TRI - Tricycle
- OMNI - Omnidirectional



A set of points to form a polygon is enough for each case.

Suggestions are given for the 3 illustrated polygons where the robot center is always at point (0,0):

```
20 20 20 22 22 33 33 33 22 22 20  20 -20 -20 -22 -22 -33 -33 -22 -22 -22 -20 -20
40 40  7  7 11 11 11 -7 -7 -3 -3 -10 -10  -3  -3  -7  -7  11  11  11   7   7  40

2   2  5  5 15 20 20 27 27 20 20  5   5  -5 -5 -20 -20 -27 -27 -20 -20 -14 -5 -5 -2 -2
54 44 44 16  2  2  8  8 -8 -8 -3 -3 -12 -12 -3  -3  -8  -8   8   8   2   2 16 44 44 54

12 12  4  4 21 44 35  40  44  51  39  32  36  31  21 -24 -33 -37 -34 -41 -52 -45 -41 -36 -46 -24 -4 -4 -12 -12
52 44 44 39 39 0 -15 -18 -11 -15 -35 -31 -24 -21 -39 -39 -23 -26 -33 -37 -17 -13 -20 -17   0  39 39 44  44  52
```

4a) - Create function to draw robot - suggestion

```
function [P,h]=DrawRobot(t,scale)
% P - points of robot to calculate its new coordinate
% h - graphic handle of robot
% t - type of robot
% scale - of robot to draw- (default 0.01)
if nargin < 2
   scale = 0.01;
end
switch t
   case 1 % DD
P=[
20 20 20 22 22 33 33 33 22 22 20  20 -20 -20 -22 -22 -33 -33 -22 -22 -22 -20 -20
40 40  7  7 11 11 11 -7 -7 -3 -3 -10 -10  -3  -3  -7  -7  11  11  11   7   7  40
];
   case 2 % tri
P=[
2   2  5  5 15 20 20 27 27 20 20   5   5  -5 -5 -20 -20 -27 -27 -20 -20 -14 -5 -5 -2 -2
54 44 44 16  2  2  8  8 -8 -8 -3 -3 -12 -12 -3  -3  -8  -8   8   8   2   2 16 44 44 54
];
   case 3 %omni
P=[
12 12  4  4 21 44 35  40  44  51  39  32  36  31  21 -24 -33 -37 -34 -41 -52 -45 -41 -36 -46 -24 -4 -4 -12 -12
52 44 44 39 39 0 -15 -18 -11 -15 -35 -31 -24 -21 -39 -39 -23 -26 -33 -37 -17 -13 -20 -17   0  39 39 44  44  52
];
end
P=scale*P;
h=fill(P(1,:), P(2,:), 'y');
```

4b) - Animate robot along plotted path (partial code)

```
[Rob,h]=DrawRobot(t);
Robh=[Rob; ones(1,size(Rob,2))];
...% Correct initial orientation of robot
for i=1:size(P,2)-1
    dr=P(:,i+1)-P(:,i);
    th=... % obtain correct orientation
    T=... %define the geometric transf.
    nRob=... %obtain current robot position
    h.XData=nRob(1,:);
    h.YData=nRob(2,:);
    pause(1/ST/Dt);
end
```