Exercise 3.4. Simulation of a car[1]

Let us consider the car with the following state equations:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ \frac{v\sin\delta}{L} \\ u_1 \\ u_2 \end{pmatrix}.$$

The state vector is given by

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \theta \\ v \\ \delta \end{pmatrix},$$

where x, y, θ corresponds to the pose of the car (in other words, its position and orientation), v is the speed and δ is the angle of the front wheels. The parameter $L = 3$ [m] is the distance between the two axles of the car.

1) Using homogeneous coordinates, design a function which draws a car in a state $\mathbf{x} = (x, y, \theta, v, \delta)^T$.

2) Propose a program which simulates the dynamic evolution of this car during 5 seconds with Euler's method and a sampling step of 0.01 [s]. Take the initial state for the car as $\mathbf{x}(0) = (0, 0, 0, 50, 0)^T$, which means that at time $t = 0$, the car is centered around the origin, with a nil heading angle, a speed of 50 [m s^{-1}] and the front wheels parallel to the axis of the car. We assume that the vectorial control $u(t)$ remains constant and equal to $(0, 0.05)$. Which means that the car does not accelerate (since $u_1 = 0$) and that the steering wheel is turning at a constant speed of 0.05 [rad s^{-1}].

Solution of Exercise 3.4.

1) Using homogeneous coordinates, design a function which draws a car in a state $\mathbf{x} = (x, y, \theta, v, \delta)^T$.

   Consider the sketch of the car represented in Figure 1.

[1] Adapted from https://www.ensta-bretagne.fr/jaulin/automooc.pdf

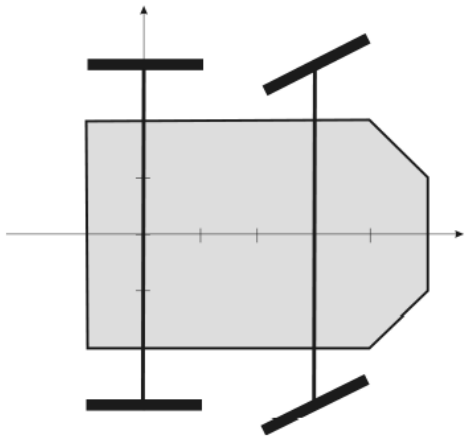

Figure 1: Sketch of a car.

To draw the sketch of a car we have to build first the sketch of the chassis, which is the sketch of the car without the front wheels represented in Figure 2, and then the sketch of the front wheels. In this way, the front wheels can be rotated by the angle δ with respect to the chassis and the whole car will be translated and rotated to assume the pose x, y, θ in the plane.

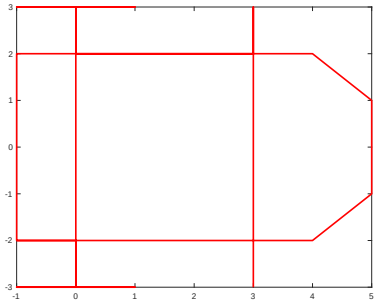

Figure 2: Sketch of the chassis of the car.

The sketch of the chassis can be represented by a rigid polygon whose vertices, in homogeneous coordinates, are defined by the column vectors of the following matrix

$$M_{\text{chassis}} = \begin{pmatrix} -1 & 4 & 5 & 5 & 4 & -1 & -1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 3 & 3 & 3 \\ -2 & -2 & -1 & 1 & 2 & 2 & -2 & -2 & -2 & -3 & -3 & -3 & -3 & 3 & 3 & 3 & 3 & 2 & 2 & 3 & -3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

expressed with respect to a frame whose origin is the center of the rear axl as in Figure 1.

To rotate the sketch of the chassis by θ counterclockwise about the origin of the chassis frame we can use the homogeneous rotation matrix

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

To translate the chassis in such a way the origin of the chassis frame is located at the point (x, y) of the fixed frame we can use the homogeneous translation matrix

$$T_{x,y} = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}.$$

To place the chassis in the pose x, y, θ we first rotate the sketch represent by matrix M and then translate it, that is,

$$M'_{\text{chassis}} = T_{x,y} R_\theta M.$$

The vertices of the segment that represents a wheel are defined in homogeneous coordinates by the column vectors of the following matrix

$$M_{\text{wheel}} = \begin{pmatrix} -1 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

This segment must be rotated counterclockwise by an angle δ, duplicated, and each copy translated in the correct position with respect to the chassis frame.

To rotate the wheel by δ counterclockwise about the origin we can use the homogeneous rotation matrix

$$R_\delta = \begin{pmatrix} \cos\delta & -\sin\delta & 0 \\ \sin\delta & \cos\delta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

To translate the wheel in such a way its midpoint coincides with the right endpoint of the front axis $(3, -3)$ we can use the following translation matrix

$$T_{3,-3} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}.$$

Likewise, to translate the wheel in such a way its midpoint coincides with the left endpoint of the front axis $(3, 3)$ we can use the following translation matrix

$$T_{3,3} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}.$$

Thus,

$$M_{\text{rightwheel}} = T_{3,-3} R_\delta M_{\text{wheel}},$$

and

$$M_{\text{leftwheel}} = T_{3,3} R_\delta M_{\text{wheel}}.$$

Once the sketch of the car is modelled with the front wheels orientation δ, the chassis and the front wheels will be rotated and translated together in such a way that the car to assumes a given pose (x, y, θ).

`Matlab` code

The `Matlab` code that implements the graphical representation of the sketch of the car is contained in the script `car_draw.m`

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

% For this system, the state is x =(x,y,theta,v,delta)
% The v state variable, since it is a speed, it is not used in this
% graphical representation

function car_draw(x)

   % Extraction of the state variables
   px=x(1);
   py=x(2);
   theta=x(3);
   v=x(4);
   delta=x(5);

   % Model of the chassis of the car without front wheels (in homogeneous coordinates)
   M_chassis=[-1  4 5  5 4 -1 -1 -1 0  0  -1  1  0 0 -1 1 0 0 3 3 3;
              -2 -2 -1 1 2 2  -2 -2 -2 -3 -3 -3 -3 3  3 3 3 2 2 3 -3;
              ones(1,21)];

   % Model of a front wheel (in homogeneous coordinates)
   M_wheel=[-1 1;
            0 0;
            1 1];

   % Translation and rotation of the whole car (chassis and front wheels)
   % with respect to the fixed frame
   TR_px_py_theta=[cos(theta),-sin(theta), px;
                   sin(theta),cos(theta), py;
                   0 0 1];

   % Chassis of the car translated and rotated
   M_chassis_transformed=TR_px_py_theta*M_chassis;

   % Translation and rotation matrix for the right wheel with respect to the chassis frame
```

```
    TR_right_wheel_delta = [cos(delta),-sin(delta), 3;
                            sin(delta),cos(delta), 3 ;
                            0 0 1];

    % Translation and rotation matrix for the left wheel with respect to the chassis frame
    TR_left_wheel_delta = [cos(delta),-sin(delta), 3;
                           sin(delta),cos(delta), -3;
                           0 0 1];

    % Right front wheel translated and rotated (first with respect to the chassis frame
    % and then with respect to the fixed frame)
    M_right_wheel_transformed=TR_px_py_theta*TR_right_wheel_delta*M_wheel;

    % Reft front wheel wheel translated and rotated (first with respect to the chassis frame
    % and then with respect to the fixed frame)
    M_left_Wheel_transformed=TR_px_py_theta*TR_left_wheel_delta*M_wheel;

    plot(M_chassis_transformed(1,:),M_chassis_transformed(2,:),'red','LineWidth',1);
    plot(M_right_wheel_transformed(1,:),M_right_wheel_transformed(2,:),'black','LineWidth',1);
    plot(M_left_Wheel_transformed(1,:),M_left_Wheel_transformed(2,:),'black','LineWidth',1);

end
```

2) Propose a program which simulates the dynamic evolution of this car during 5 seconds with Euler's method and a sampling step of 0.01 [s].

The Matlab code that implements the simulator of the car for the given input, is contained in the script car_main.m

```
%Adapted from https://www.ensta-bretagne.fr/jaulin/

init;

%For this system, the state is x =(x,y,theta,v,delta)

x=[0;0;0;50;0]; % Initial state

dt=0.01;

frame_counter=0;

car_draw(x);

for t=0:dt:5

    u1=0;
    u2=0.05;
    u=[u1;u2];
    %x=x+car_f(x,u)*dt; % Euler
    x=x+dt*(0.25*car_f(x,u)+0.75*(car_f(x+dt*(2/3)*car_f(x,u),u))); % Runge-Kutta

    pause(dt);

    frame_counter =frame_counter+1;

    % Frame sampling
    if frame_counter == 15
       car_draw(x);
       frame_counter =0;
    end
end;
```

The Matlab code that implements the kinematic model of the car is contained in the script car_f.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

function  xdot  = f(x,u)
% state x = (x,y,theta,v,delta)
% control u=(u1 u2)
L=3;

px=x(1);
py=x(2);
theta=x(3);
v=x(4);
delta=x(5);

xdot=[v*cos(delta)*cos(theta); v*cos(delta)*sin(theta); v*sin(delta)/L; u(1); u(2)];
end
```

Finally, the script `init.m` contains some initialization parameters.

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

close all;
clear all;
clc;

%size = get(0,'ScreenSize'); % full screen
figure%('Position',[0 0 size(3)/2 size(4)/2]);
hold
% set(gca,'FontSize',12);
xmin=-10;
xmax=100;
ymin=-10;
ymax=100;

axis([xmin xmax ymin ymax]);
axis ('square');
```