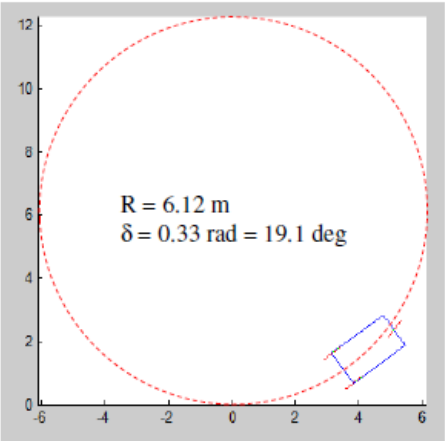
MATLAB PROGRAMMING

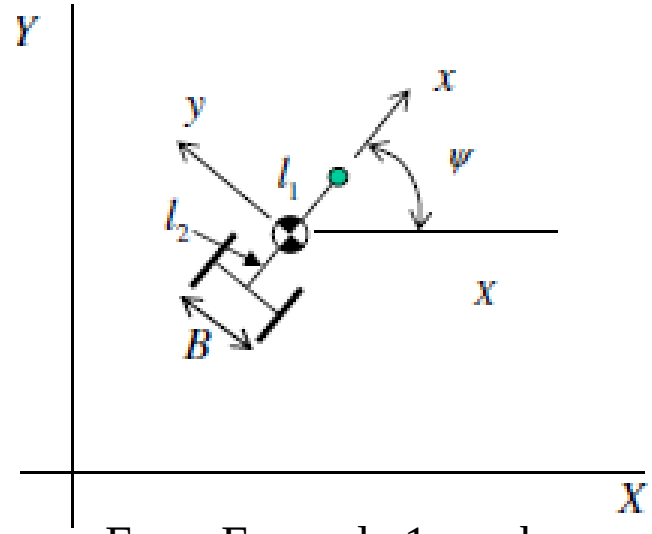
ME5670



Kinematics: Example 1

Position and velocity in inertial frame

- Vehicle kinematic state in the inertial frame . $\mathbf{q}_I = \begin{bmatrix} X \\ Y \\ \psi \end{bmatrix}$



- Velocities in the local reference frame are related with the inertial frame by the rotation matrix .$\mathbf{R}(\psi)$

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \dot{\mathbf{q}} = \mathbf{R}(\psi)\cdot\dot{\mathbf{q}}_I$$

- Velocities in the global reference frame

$$\dot{\mathbf{q}}_I = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} U \\ V \\ \Omega \end{bmatrix} = \mathbf{\Psi}(\psi)\cdot\dot{\mathbf{q}} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$$

From Example 1, we have

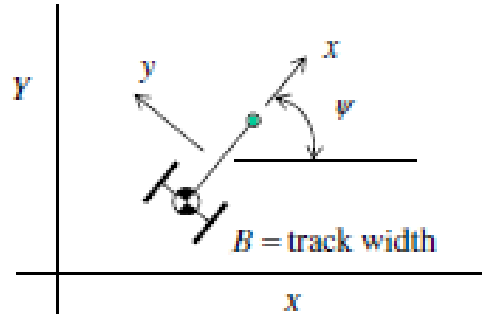
$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{R_w}{2}(\omega_1+\omega_2) \\ \frac{l_2 R_w}{B}(\omega_1-\omega_2) \\ \frac{R_w}{B}(\omega_1-\omega_2) \end{bmatrix}$$

- Velocities in the global reference frame in terms of wheel velocities are

$$\dot{\mathbf{q}}_I = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{R_w}{2}(\omega_1+\omega_2)\cos\psi - \frac{l_2 R_w}{B}(\omega_1-\omega_2)\sin\psi \\ \frac{R_w}{2}(\omega_1+\omega_2)\sin\psi + \frac{l_2 R_w}{B}(\omega_1-\omega_2)\cos\psi \\ \frac{R_w}{B}(\omega_1-\omega_2) \end{bmatrix}$$

Kinematics: Example 2

Differentially-driven single axle vehicle with CG on axle

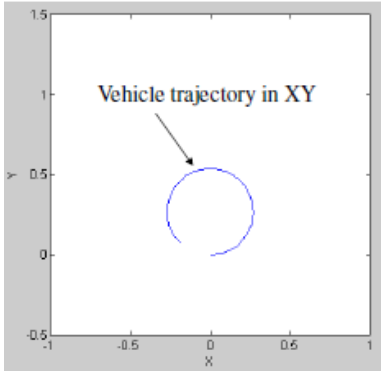


- For a kinematic model for a vehicle with CG on axle

$$l_1 = L \quad \text{and} \quad l_2 = 0$$

- Velocities in the global reference frame

$$\dot{\mathbf{q}}_I = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{R_w}{2}(\omega_1+\omega_2)\cos\psi - \frac{l_2 R_w}{B}(\omega_1-\omega_2)\sin\psi \\ \frac{R_w}{2}(\omega_1+\omega_2)\sin\psi + \frac{l_2 R_w}{B}(\omega_1-\omega_2)\cos\psi \\ \frac{R_w}{B}(\omega_1-\omega_2) \end{bmatrix} = \begin{bmatrix} \frac{R_w}{2}(\omega_1+\omega_2)\cos\psi \\ \frac{R_w}{2}(\omega_1+\omega_2)\sin\psi \\ \frac{R_w}{B}(\omega_1-\omega_2) \end{bmatrix}$$



- MATLAB programming

```
function Xidot = DS_vehicle(t,Xi)
global R_w B omegaw1 omegaw2
X = Xi(1); Y = Xi(2); psi = Xi(3);
Xdot = 0.5*cos(psi)*R_w*(omegaw1+omegaw2);
Ydot = 0.5*sin(psi)*R_w*(omegaw1+omegaw2);
psidot = R_w*(omegaw1-omegaw2)/B;
Xidot=[Xdot;Ydot;psidot];
```

```
clear all
global R_w B omegaw1 omegaw2
R_w = 0.05; B = 0.18;
omegaw1 = 4; omegaw2 = 2;
Xi0=[0,0,0];
[t,Xi] = ode45(@DS_vehicle,[0 10],Xi0);
N = length(t);
figure(1)
plot(Xi(:,1),Xi(:,2)), axis([-1.0 1.0 -0.5 1.5]), axis('square')
xlabel('X'), ylabel('Y')
```

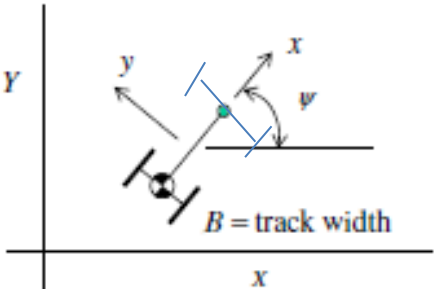
Courtesy: Prof. R.G. Longoria

Kinematics: 2D Animation

Differentially-driven single axle vehicle with CG on axle

Vehicle State

```
function [xb, yb, xfc, yfc, xrlw, yrlw, xrrw, yrrw] = vehicle_state(q,u)
global L B R_w
% x and y are the coordinates at the rear axle center - CG location
% u is a control input
x = q(1);  y = q(2); psi = q(3);

% xfc and yfc are coordinates of a center pivot at front
% then the pivot point  is located w.r.t CG at a distance L

 xfc = x + 1*L*cos(psi); yfc = y + 1*L*sin(psi);

% Find coordinates of vehicle base

xfl = xfc - 0.5*B*sin(psi); yfl = yfc + 0.5*B*cos(psi);
xfr = xfc + 0.5*B*sin(psi); yfr = yfc - 0.5*B*cos(psi);
xrl = x - 0.5*B*sin(psi); yrl = y + 0.5*B*cos(psi);
xrr = x + 0.5*B*sin(psi); yrr = y - 0.5*B*cos(psi);
xb = [xfl, xfr, xrr, xrl, xfl]; % x coordinates for vehicle base
yb = [yfl, yfr, yrr, yrl, yfl]; % y coordinates for vehicle base
```




```
% Find coordinates to draw wheels
% rear left wheel
xrlwf = xrl + R_w*cos(psi);
yrlwf = yrl + R_w*sin(psi);
xrlwr = xrl - R_w*cos(psi);
yrlwr = yrl - R_w*sin(psi);
% rear right wheel
xrrwf = xrr + R_w*cos(psi);
yrrwf = yrr + R_w*sin(psi);
xrrwr = xrr - R_w*cos(psi);
yrrwr = yrr - R_w*sin(psi);

xrlw = [xrlwf, xrlwr];
yrlw = [yrlwf, yrlwr];

xrrw = [xrrwf, xrrwr];
yrrw = [yrrwf, yrrwr];
```

Courtesy: Prof. R.G. Longoria

Kinematics: 2D Animation

Differentially-driven single axle vehicle with CG on axle

Rk4 Solver

```
function [T,X]=rk4fixed(Fcn,Tspan,X0,N)
h = (Tspan(2)-Tspan(1))/N;
halfh = 0.5*h;

neqs=size(X0);
X=zeros(neqs(1),N);
T=zeros(1,N);
X(:,1)=X0;
T(1)=Tspan(1);
Td = Tspan(1);
Xd = X0;
for i=2:N,
 RK1 = feval(Fcn,Td,Xd);
 Thalf = Td + halfh;
 Xtemp = Xd + halfh*RK1;
 RK2 = feval(Fcn,Thalf,Xtemp);
 Xtemp = Xd + halfh*RK2;
 RK3 = feval(Fcn,Thalf,Xtemp);
 Tfull = Td + h;
 Xtemp = Xd + h*RK3;
 RK4 = feval(Fcn,Tfull,Xtemp);
 X(:,i) = Xd + h*(RK1+2.0*(RK2+RK3)+RK4)/6;
 T(i) = Tfull;
 Xd = X(:,i);
 Td = T(i);
end
X=X';T=T';
```

Courtesy: Prof. R.G. Longoria

2D Animation

Sim_2Danim.m

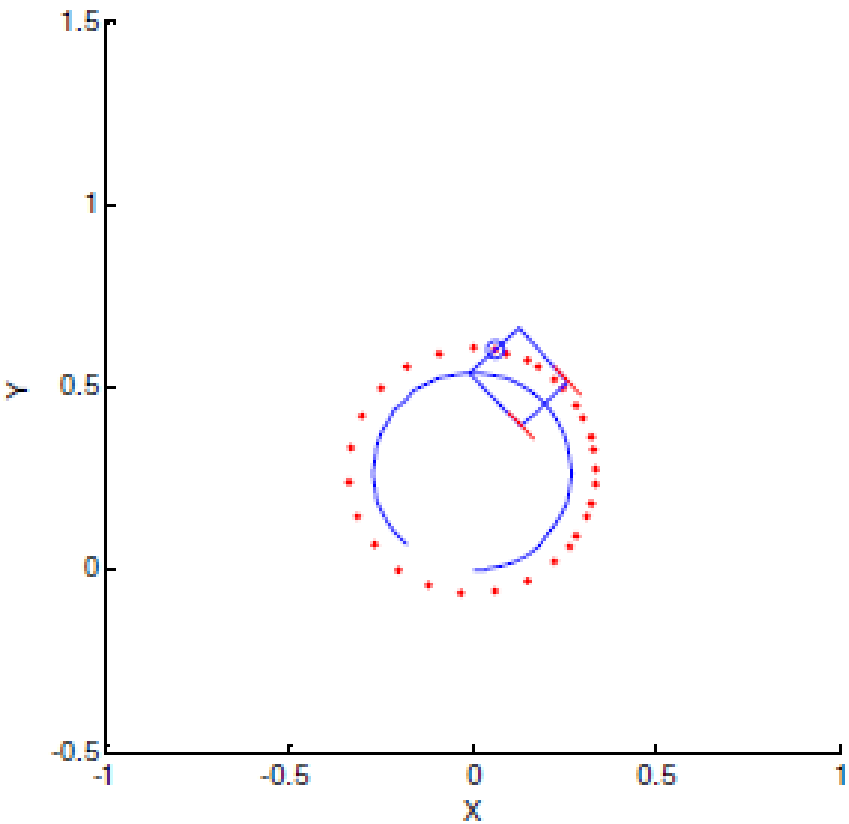
```
clear all;  % Clear all variables
close all;  % Close all figures

global L B R_w omegaw1 omegaw2

% Geometric vehicle parameters
L = 0.20;  % wheel base
B = 0.18;  % rear axle track width
R_w = 0.05;  % wheel radius

% Initial location and orientation of the vehicle CG
x0 = 0; y0 = 0; psi0 = 0*pi/180; % psi = yaw angle in radians

fig1 = figure(1);
axis([-1.0 1.0 -0.5 1.5]); axis('square')
xlabel('X'), ylabel('Y')
hold on;

q0=[x0,y0,psi0];
```

```
% Vehicle_State provides spatial state information for the vehicle

[xb, yb, xfc, yfc, xrlw, yrlw, xrrw, yrrw] = vehicle_state(q0,0);

% Plot vehicle and define component plots
plotzb = plot(xb, yb);  % Plot robot base
plotzfc = plot(xfc, yfc, 'o');  % Plot front pivot
plotzrlw = plot(xrlw, yrlw, 'r');  % Plot rear left wheel
plotzrrw = plot(xrrw, yrrw, 'r');  % Plot rear right wheel

% Set handle graphics parameters and plotting modes
set(gca, 'drawmode','fast');
set(plotzb, 'erasemode', 'xor'); % use 'xor' rather than 'none' to redraw
set(plotzfc, 'erasemode', 'xor');
set(plotzrlw, 'erasemode', 'xor');
set(plotzrrw, 'erasemode', 'xor');

q1 = q0;  % Set initial state to q1 for simulation

% Fixed wheel speed command - should make a circle!
omegaw1 = 2; omegaw2 = 1;
```

Courtesy: Prof. R.G. Longoria

Kinematics: 2D Animation

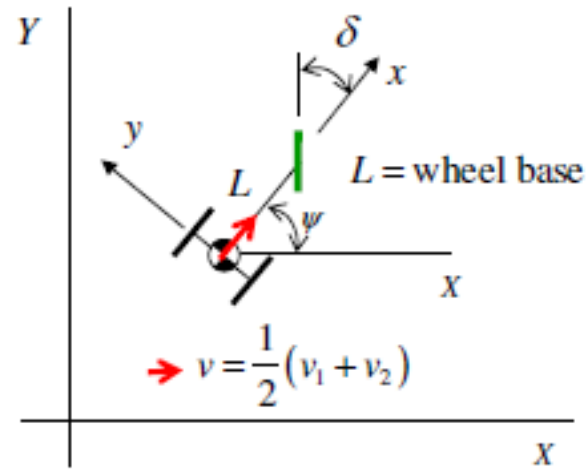
```
% Parameters related to simulations
tfinal = 100;
N = 30;  % Number of iterations
dt = tfinal/N;  % Time step interval
t = [0:dt:tfinal];

% Beginning of simulation and animation
for i = 1:N+1
    to = t(i); tf = t(i)+dt;
    % integrate from to to tf
    [t2,q2]=rk4fixed('dssakv',[to tf],q1',2);
     t1 = t2(2); % keep only the last point
    q1 = q2(2,:);  % store q2 in q1 for next step

    % capture the state of the vehicle for animation
    [xb, yb, xfc, yfc, xrlw, yrlw, xrrw, yrrw] = vehicle_state(q1, 0);
    plot(xfc,yfc,'r.')
    % Plot vehicle - updates data in each plot
    set(plotzb,'xdata',xb);
    set(plotzb,'ydata',yb);
    set(plotzfc,'xdata',xfc);
    set(plotzfc,'ydata',yfc);
    set(plotzrlw,'xdata',xrlw);
    set(plotzrlw,'ydata',yrlw);
    set(plotzrrw,'xdata',xrrw);
    set(plotzrrw,'ydata',yrrw);
    pause(0.2);  % Pause by X seconds for slower animation
end
```



Courtesy: Prof. R.G. Longoria

Ackerman Steering : A Tricycle

Single-axle vehicle with front-steered wheel; rolling rear wheels



- For a given steer angle δ and C.G. velocity along x as v.

- Velocities in the inertial frame is given by

$$\dot{X} = v\cos\psi = R_w\omega\cos\psi$$

$$\dot{Y} = v\sin\psi = R_w\omega\sin\psi$$

$$\dot{\psi} = \frac{v}{L}\tan\delta$$

- The input control variables are $v = R_w\omega$ and steer angle δ

- Here, C.G. is located at the rear axle, its velocity is given by

$$v = \frac{1}{2}(v_1 + v_2), \text{ where } v_1 = R_w\omega_1 \text{ and } v_2 = R_w\omega_2$$

- Forward velocty at the front wheel is v along x.

- . Because of steering, the velocity along the path of the wheel is $v_\delta = \frac{v}{\cos(\delta)}$
- The velocity lateral to the wheel is $v_t = v_\delta \sin(\delta) = v\tan(\delta)$
- Therefore, $\tan(\delta) = {}^{v_t}/_{v} \longrightarrow \omega_z = \dot{\psi} = \frac{v}{L}\tan(\delta) = {}^{v_t}/_{L}$

Courtesy: Prof. R.G. Longoria

Kinematics: 2D Animation

Differentially-driven single axle tricycle

Tricycle State

```
function [xb, yb, xfw, yfw, xrlw, yrlw, xrrw, yrrw] = tricycle_state(q, u)
global L B R_w
% x and y are the coordinates at the rear axle center - CG location
% u is a control input
x = q(1);  y = q(2); psi = q(3);
 v = u(1); delta = u(2);
% xfc and yfc are coordinates of a center pivot at front
% then the pivot point  is located w.r.t CG at a distance L

 xfc = x + L*cos(psi); yfc = y + L*sin(psi);

% Find coordinates of vehicle base

xfl = xfc - 0.5*B*sin(psi); yfl = yfc + 0.5*B*cos(psi);
xfr = xfc + 0.5*B*sin(psi); yfr = yfc - 0.5*B*cos(psi);
xrl = x - 0.5*B*sin(psi); yrl = y + 0.5*B*cos(psi);
xrr = x + 0.5*B*sin(psi); yrr = y - 0.5*B*cos(psi);
% end points of the front-steered wheel
xfwf = xfc + R_w*cos(psi+delta);
yfwf = yfc + R_w*sin(psi+delta);
xfwr = xfc - R_w*cos(psi+delta);
yfwr = yfc - R_w*sin(psi+delta);
```

Courtesy: Prof. R.G. Longoria

```
% Find coordinates to draw wheels
% rear left wheel
xrlwf = xrl + R_w*cos(psi);
yrlwf = yrl + R_w*sin(psi);
xrlwr = xrl - R_w*cos(psi);
yrlwr = yrl - R_w*sin(psi);
% rear right wheel
xrrwf = xrr + R_w*cos(psi);
yrrwf = yrr + R_w*sin(psi);
xrrwr = xrr - R_w*cos(psi);
yrrwr = yrr - R_w*sin(psi);
 % define the states
% front center point (not returned)
qfc = [xfc, yfc];
% body x-y points
xb = [xfl, xfr, xrr, xrl, xfl];
yb = [yfl, yfr, yrr, yrl, yfl];
% front wheel x-y points
 xfw = [xfwf, xfwr];
yfw = [yfwf, yfwr];
% rear-left wheel x-y points
xrlw = [xrlwf, xrlwr];
yrlw = [yrlwf, yrlwr];
% rear-right wheel x-y points
xrrw = [xrrwf, xrrwr];
yrrw = [yrrwf, yrrwr];
```

2D Animation …

```
function qdot = ks_tricycle_kv(t,q)

global L vc delta_radc delta_max_deg R_w

% L is length between the front wheel axis and rear wheel
%axis [m]
% vc is speed command
% delta_radc is the steering angle command

% State variables
x = q(1); y = q(2); psi = q(3);

% Control variables
v = vc;
delta = delta_radc;

% kinematic model
xdot = v*cos(psi);
ydot = v*sin(psi);
psidot = v*tan(delta)/L;
qdot = [xdot;ydot;psidot];
```

```
% sim_tricycle_model.m
clear all;  % Clear all variables
close all;  % Close all figures
global L B R_w vc delta_radc

% Physical parameters of the tricycle
L = 2.040; %0.25;  % [m]
B = 1.164; %0.18;  % Distance between the rear wheels [m]
m_max_rpm = 8000;  % Motor max speed [rpm]
gratio = 20;  % Gear ratio
R_w = 13/39.37;  % Radius of wheel [m]

% Parameters related to vehicle
m_max_rads = m_max_rpm*2*pi/60;  % Motor max speed [rad/s]
w_max_rads = m_max_rads/gratio;  % Wheel max speed [rad/s]
v_max = w_max_rads*R_w;  % Max robot speed [m/s]

% Initial values
x0 = 0;  % Initial x coodinate [m]
y0 = 0;  % Initial y coodinate [m]
psi_deg0 = 0;  % Initial orientation of the robot (theta [deg])

% desired turn radius
R_turn = 3*L;
delta_max_rad = L/R_turn;  % Maximum steering angle [deg]

% Parameters related to simulations
t_max = 10;  % Simulation time [s]
n = 100;  % Number of iterations
dt = t_max/n;  % Time step interval
t = [0:dt:t_max];  % Time vector (n+1 components)
```

Courtesy: Prof. R.G. Longoria

2D Animation…

```
% velocity and steering commands (open loop)
v = v_max*ones(1,n+1);  % Velocity vector (n+1 components)
delta_rad = delta_max_rad*ones(1,n+1); % Steering angle vector (n+1
%components) [rad]

psi_rad0 = psi_deg0*pi/180;  % Initial orientation [rad]
v0 = v(1);  % Initial velocity [m/s]
delta_rad0 = delta_rad(1);  % Initial steering angle [rad]

q0 = [x0, y0, psi_rad0];  % Initial state vector
u0 = [v0, delta_rad0];  % Initial control vector

fig1 = figure(1);  % Figure set-up (fig1)
axis([-R_turn R_turn -0*R_turn 2*R_turn]); axis('square')
hold on;

% Acquire the configuration of robot for plot
[xb, yb, xfw, yfw, xrlw, yrlw, xrrw, yrrw] = tricycle_state(q0, u0);
plotqb = plot(xb, yb);  % Plot vehicle base
plotqfw = plot(xfw, yfw, 'r');  % Plot front wheel
plotqrlw = plot(xrlw, yrlw, 'r');  % Plot rear left wheel
plotqrrw = plot(xrrw, yrrw, 'r');  % Plot rear right wheel
% Draw fast and erase fast
set(gca, 'drawmode','fast');
set(plotqb, 'erasemode', 'xor');
set(plotqfw, 'erasemode', 'xor');
set(plotqrlw, 'erasemode', 'xor');
set(plotqrrw, 'erasemode', 'xor');

q1 = q0;  % Set initial state to z1 for simulation
```

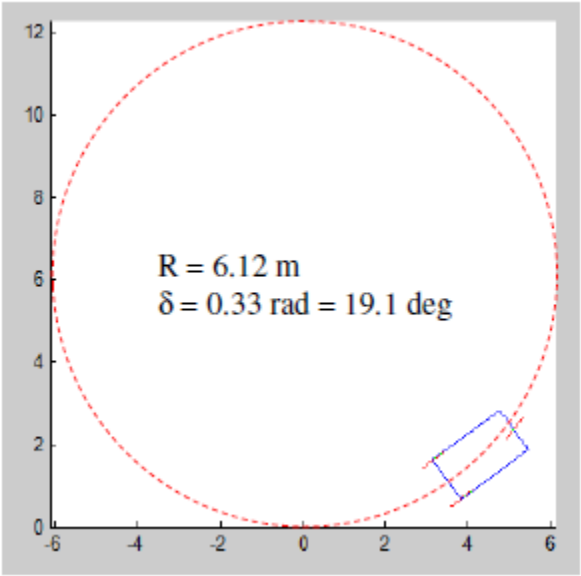
```
% Beginning of simulation
for i = 1:n+1
   v(i) = v_max*cos(2*delta_rad(i));
   u = [v(i), delta_rad(i)];  % Set control input
   vc = u(1); delta_radc = u(2);
   to = t(i); tf = t(i)+dt;
   [t2,q2]=rk4fixed('ks_tricycle_kv',[to tf],q1',2);
   t1 = t2(2);
   q1 = q2(2,:);

% Acquire the configuration of vehicle for plot
   [xb, yb, xfw, yfw, xrlw, yrlw, xrrw, yrrw] = tricycle_state(q1, u);

   % Plot vehicle
   set(plotqb,'xdata',xb);
   set(plotqb,'ydata',yb);
   set(plotqfw,'xdata',xfw);
   set(plotqfw,'ydata',yfw);
   set(plotqrlw,'xdata',xrlw);
   set(plotqrlw,'ydata',yrlw);
   set(plotqrrw,'xdata',xrrw);
   set(plotqrrw,'ydata',yrrw);
   % drawnow
   pause(0.1);  % Pause by 0.2s for slower simulation
end
```

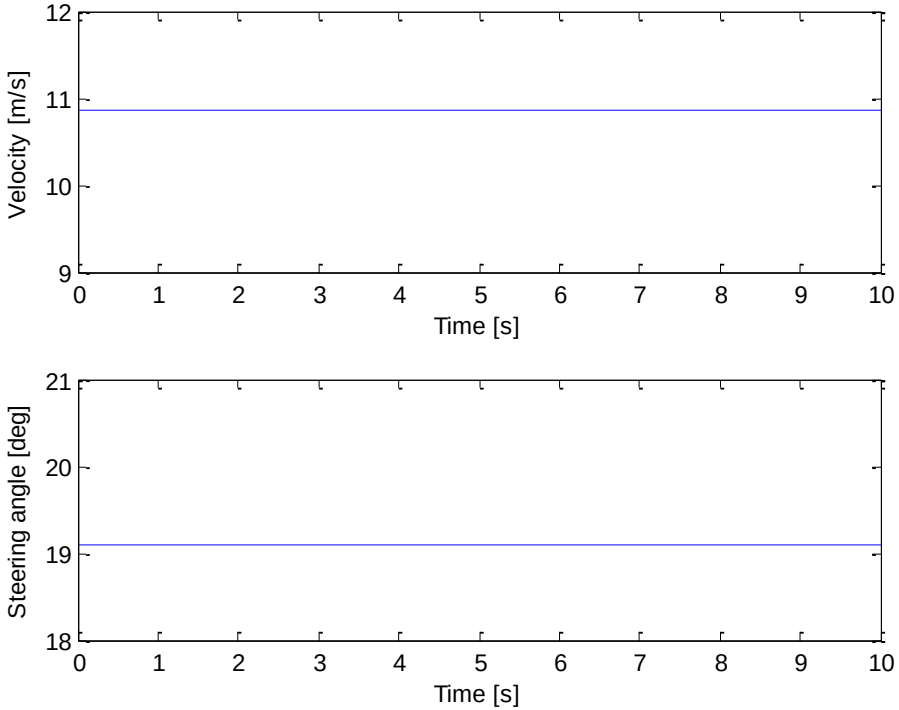
Courtesy: Prof. R.G. Longoria

2D Animation…

```
% Plot the resultant velocity and steering angle configurations
fig2 = figure(2);  % Figure set-up (fig2)
subplot(2,1,1);  % Upper half of fig1
plot(t, v);  % Plot velocity-time curve
xlabel('Time [s]');
ylabel('Velocity [m/s]');
```

```
subplot(2,1,2);  % Lower half of fig1
deltad = delta_rad*180/pi;  % Steering angle vector (n+1 comp.) [deg]
plot(t,deltad);  % Plot steering angle-time curve
xlabel('Time [s]');
ylabel('Steering angle [deg]');
```





Courtesy: Prof. R.G. Longoria

Kinematics: Lane Change Problem

Differentially-driven Double axle Four Wheel Vehicle

Four Wheel Vehicle State

```
function [xb, yb, xfc, yfc, xfwr, yfwr,xfwl, yfwl, xrlw, yrlw,
xrrw, yrrw] = Fourwheel_state(q, u)

global L B R_w

x = q(1); y = q(2); psi = q(3);
v = u(1); delta = u(2);

% locates front-center point
xfc = x + L*cos(psi);
yfc = y + L*sin(psi);

% locates four corners
xfl = xfc - 0.5*B*sin(psi);
yfl = yfc + 0.5*B*cos(psi);
xfr = xfc + 0.5*B*sin(psi);
yfr = yfc - 0.5*B*cos(psi);
xrl = x - 0.5*B*sin(psi);
yrl = y + 0.5*B*cos(psi);
xrr = x + 0.5*B*sin(psi);
yrr = y - 0.5*B*cos(psi);
```

```
% end points of the front-steered wheel
xfwfr = xfl + R_w*cos(psi+delta);
yfwfr = yfl + R_w*sin(psi+delta);
xfwrr = xfl - R_w*cos(psi+delta);
yfwrr = yfl - R_w*sin(psi+delta);

% end points of the front-steered wheel
xfwfl = xfr + R_w*cos(psi+delta);
yfwfl = yfr + R_w*sin(psi+delta);
xfwrl = xfr - R_w*cos(psi+delta);
yfwrl = yfr - R_w*sin(psi+delta);

% end points of the rear-left wheel
xrlwf = xrl + R_w*cos(psi);
yrlwf = yrl + R_w*sin(psi);
xrlwr = xrl - R_w*cos(psi);
yrlwr = yrl - R_w*sin(psi);

% end points of the rear-right wheel
xrrwf = xrr + R_w*cos(psi);
yrrwf = yrr + R_w*sin(psi);
xrrwr = xrr - R_w*cos(psi);
yrrwr = yrr - R_w*sin(psi);
];
```

2D Animation …

```
% define the states

% front center point (not returned)
qfc = [xfc, yfc];

% body x-y points
xb = [xfl, xfr, xrr, xrl, xfl];
yb = [yfl, yfr, yrr, yrl, yfl];

% left front wheel x-y points
xfwl = [xfwfl, xfwrl];
yfwl = [yfwfl, yfwrl];

% Right front wheel x-y points
xfwr = [xfwfr, xfwrr];
yfwr = [yfwfr, yfwrr];

% rear-left wheel x-y points
xrlw = [xrlwf, xrlwr];
yrlw = [yrlwf, yrlwr];

% rear-right wheel x-y points
xrrw = [xrrwf, xrrwr];
yrrw = [yrrwf, yrrwr];
```

```
% sim_tricycle_model.m
clear all;  % Clear all variables
close all;  % Close all figures
global L B R_w vc delta_radc

% Physical parameters of the tricycle
L = 2.040; %0.25;  % [m]
B = 1.164; %0.18;  % Distance between the rear wheels [m]
m_max_rpm = 8000;  % Motor max speed [rpm]
gratio = 20;  % Gear ratio
R_w = 13/39.37;  % Radius of wheel [m]

% Parameters related to vehicle
m_max_rads = m_max_rpm*2*pi/60;  % Motor max speed [rad/s]
w_max_rads = m_max_rads/gratio;  % Wheel max speed [rad/s]
v_max = w_max_rads*R_w;  % Max robot speed [m/s]

% Initial values
x0 = 0;  % Initial x coodinate [m]
y0 = 0;  % Initial y coodinate [m]
psi_deg0 = 0;  % Initial orientation of the robot (theta [deg])

% desired turn radius
R_turn = 3*L;
delta_max_rad = L/R_turn;  % Maximum steering angle [deg]

% Parameters related to simulations
t_max = 10;  % Simulation time [s]
n = 100;  % Number of iterations
dt = t_max/n;  % Time step interval
t = [0:dt:t_max];  % Time vector (n+1 components)
```

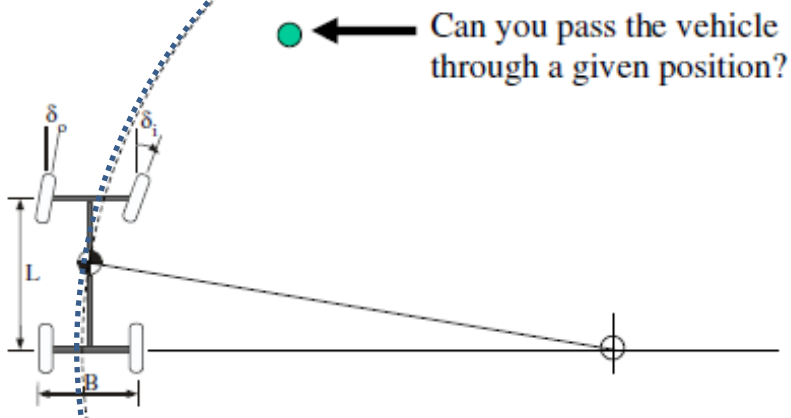
2D Animation…

```
% velocity and steering commands (open loop)
v = v_max*ones(1,n+1);  % Velocity vector (n+1 components)
delta_rad = delta_max_rad*ones(1,n+1); % Steering angle vector (n+1
%components) [rad]

psi_rad0 = psi_deg0*pi/180;  % Initial orientation [rad]
v0 = v(1);  % Initial velocity [m/s]
delta_rad0 = delta_rad(1);  % Initial steering angle [rad]

q0 = [x0, y0, psi_rad0];  % Initial state vector
u0 = [v0, delta_rad0];  % Initial control vector

fig1 = figure(1);  % Figure set-up (fig1)
axis([-R_turn R_turn -0*R_turn 2*R_turn]); axis('square')
hold on;

% Acquire the configuration of robot for plot
[xb, yb, xfw, yfw, xrlw, yrlw, xrrw, yrrw] = tricycle_state(q0, u0);
plotqb = plot(xb, yb);  % Plot vehicle base
plotqfw = plot(xfw, yfw, 'r');  % Plot front wheel
plotqrlw = plot(xrlw, yrlw, 'r');  % Plot rear left wheel
plotqrrw = plot(xrrw, yrrw, 'r');  % Plot rear right wheel
% Draw fast and erase fast
set(gca, 'drawmode','fast');
set(plotqb, 'erasemode', 'xor');
set(plotqfw, 'erasemode', 'xor');
set(plotqrlw, 'erasemode', 'xor');
set(plotqrrw, 'erasemode', 'xor');

q1 = q0;  % Set initial state to z1 for simulation
```

```
% Beginning of simulation
for i = 1:n+1
   v(i) = v_max*cos(2*delta_rad(i));
   u = [v(i), delta_rad(i)];  % Set control input
   vc = u(1); delta_radc = u(2);
   to = t(i); tf = t(i)+dt;
   [t2,q2]=rk4fixed('ks_tricycle_kv',[to tf],q1',2);
   t1 = t2(2);
   q1 = q2(2,:);

% Acquire the configuration of vehicle for plot
   [xb, yb, xfw, yfw, xrlw, yrlw, xrrw, yrrw] = tricycle_state(q1, u);

   % Plot vehicle
   set(plotqb,'xdata',xb);
   set(plotqb,'ydata',yb);
   set(plotqfw,'xdata',xfw);
   set(plotqfw,'ydata',yfw);
   set(plotqrlw,'xdata',xrlw);
   set(plotqrlw,'ydata',yrlw);
   set(plotqrrw,'xdata',xrrw);
   set(plotqrrw,'ydata',yrrw);
   % drawnow
   pause(0.1);  % Pause by 0.2s for slower simulation
end
```

Practice Problem

Using the basic geometry of Ackermann steering



1. Assume low-speed turning
2. Project along rear-axle
3. Define $R = L/\delta_{max}$
4. Project from CG
5. Project ideal turning path