Control of a vehicle with two driving wheels

Adrià Junyent-Ferré

April 21, 2017

Abstract

The purpose of this document is to summarise the basic kinematic equations of a two-wheeled vehicle and describe a control law that can be used to turn the driver's commands into speed references for the actuators attached to each of the wheels.

1 Analysis of the kinematics of the vehicle

This paper analyses the kinematics of a vehicle with two driving wheels that uses a so-called differential drive concept for steering. A top view of the vehicle along with the variables employed to describe the geometry and the kinematics of the vehicle are shown in Figure 1.

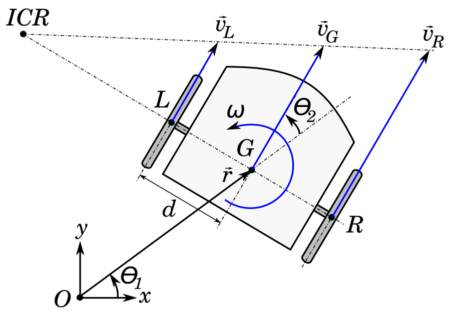


Figure 1: Diagram of the vehicle

Taking a fixed position in the ground plane, O, as the reference, the position of the centre of the vehicle, $\vec{r}$, can be written as:

$$\vec{r} = \begin{bmatrix} r\cos(\theta_1) \\ r\sin(\theta_1) \end{bmatrix} \tag{1}$$

The speed of the centre of the vehicle, $\vec{v}_G$, can be obtained by differentiating this equation against the time:

$$\vec{v}_G = \begin{bmatrix} \dot{r}\cos(\theta_1) - r\dot{\theta}_1\sin(\theta_1) \\ \dot{r}\sin(\theta_1) + r\dot{\theta}_1\cos(\theta_1) \end{bmatrix} \tag{2}$$

On the other hand, the rotational speed of the vehicle, ω, can be expressed as:

$$\omega = \dot{\theta}_1 + \dot{\theta}_2 \tag{3}$$

Many applications require vehicle speed control rather than position control. This means that there isn't an obvious relevant reference position O to use and the speed equations play a more important role than the position equations. Further, as the vehicle is driven and controlled by the actuators in the two wheels of the vehicle, it is useful to describe the speed of the vehicle as a function of the speed of the wheels. The wheels are assumed to not slip, this implies that the point in the wheel that is in contact with the ground has zero absolute speed at all times and the centre of the wheel can only move in a direction that is parallel to the ground and perpendicular to the axis of the wheel. Therefore, speed of the centre of each wheel can be written as:

$$\vec{v}_L = \begin{bmatrix} V_L\cos(\theta_1+\theta_2) \\ V_L\sin(\theta_1+\theta_2) \end{bmatrix} \tag{4}$$

and

$$\vec{v}_R = \begin{bmatrix} V_R\cos(\theta_1+\theta_2) \\ V_R\sin(\theta_1+\theta_2) \end{bmatrix} \tag{5}$$

Given that the motion of the vehicle is confined to a plane, the speed of any point of the vehicle as well as the rotating speed of the vehicle is fully defined once the speeds of two points are known. Therefore the speed of the centre of the vehicle can be written as:

$$\vec{v}_G = \begin{bmatrix} \frac{V_R+V_L}{2}\cos(\theta_1+\theta_2) \\ \frac{V_R+V_L}{2}\sin(\theta_1+\theta_2) \end{bmatrix} \tag{6}$$

Also, the rotational speed can be written as:

$$\omega = \frac{V_R - V_L}{2d} \tag{7}$$

The vehicle will be controlled by an operator that will give commands to the controller using a joystick that will be installed in the vehicle. The operator will point the joystick in the direction that they want the vehicle to move. Therefore, the speed command, $\vec{v}_G^*$, will be of the form:

$$\vec{v}_G^* \triangleq \begin{bmatrix} V_F\cos(\theta_1+\theta_2) + V_S\sin(\theta_1+\theta_2) \\ V_F\sin(\theta_1+\theta_2) - V_S\cos(\theta_1+\theta_2) \end{bmatrix} \tag{8}$$

where V_F and V_S will be the "forward" and the "sideways" speed commands that the controller will be able to read from the two axes of the joystick input. The expressions of $\vec{v}_G$ and $\vec{v}_G^*$ above depend on absolute orientation of the vehicle, $\theta_1+\theta_2$. This orientation is not necessarily known by the controller unless a compass sensor is installed and it doesn't provide any key information for the control. The former equations can be transformer to the reference frame of the vehicle by applying the following transformation:

$$\vec{v}' \triangleq T\vec{v} \tag{9}$$

where

$$T \triangleq \begin{bmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) \\ -\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) \end{bmatrix} \tag{10}$$

This gives:

$$\vec{v}_G' = \begin{bmatrix} \frac{V_R+V_L}{2} \\ 0 \end{bmatrix} \tag{11}$$

and

$$(\vec{v}_G^*)' = \begin{bmatrix} V_F \\ -V_S \end{bmatrix} \tag{12}$$

The condition of no-slip of the wheels impedes the vehicle to move sideways, therefore the vehicle will need to manoeuvre in order to start moving in an arbitrary direction. The derivation of a suitable control law that can be used to translate the operator's commands into the target speeds of the wheels is to approximate the speed equation for small variations of the orientation of the vehicle. This gives:

$$\vec{v}_G \approx \begin{bmatrix} \frac{V_R+V_L}{2}\cos(\hat{\theta}_1+\hat{\theta}_2) - \frac{V_R+V_L}{2}\frac{V_R-V_L}{2d}\sin(\hat{\theta}_1+\hat{\theta}_2)\Delta t \\ \frac{V_R+V_L}{2}\sin(\hat{\theta}_1+\hat{\theta}_2) + \frac{V_R+V_L}{2}\frac{V_R-V_L}{2d}\cos(\hat{\theta}_1+\hat{\theta}_2)\Delta t \end{bmatrix} \tag{13}$$

where θ_1 and θ_2 are assumed to be close to $\hat{\theta}_1$ and $\hat{\theta}_2$ respectively for a time step of Δt using only the linear terms of the Taylor expansion of these angles. If this equation is expressed in the reference frame of the vehicle, the following is obtained:

$$\vec{v}_G' \approx \begin{bmatrix} \frac{V_R+V_L}{2} \\ \frac{V_R+V_L}{2}\frac{V_R-V_L}{2d}\Delta t \end{bmatrix} \tag{14}$$

This can be used to solve for the wheel speeds required in order to obtain the speed requested by the operator, leading to:

$$\begin{cases} V_R = & V_F - d\frac{V_S}{V_F}\frac{1}{\Delta T} \\ V_L = & V_F + d\frac{V_S}{V_F}\frac{1}{\Delta T} \end{cases} \tag{15}$$

This result is intuitive: if the operator commands the vehicle to move towards the forward direction of the vehicle, both wheels must spin at the same speed; whereas if the operator commands the vehicle to move forward but slightly towards one side, the wheel on that side must spin at a lower pace than the wheel on the other side in order for the vehicle to turn towards the side. It is worth pointing out, that once more, as a consequence of the inability of the wheels to move in the direction of their axis, a command of zero forward speed and non-zero sideways speed results in the speed command for the wheels to become infinity. This type of singularity is common in vehicle and manipulator control. One possible way to overcome this issue is to redefine the control law as:

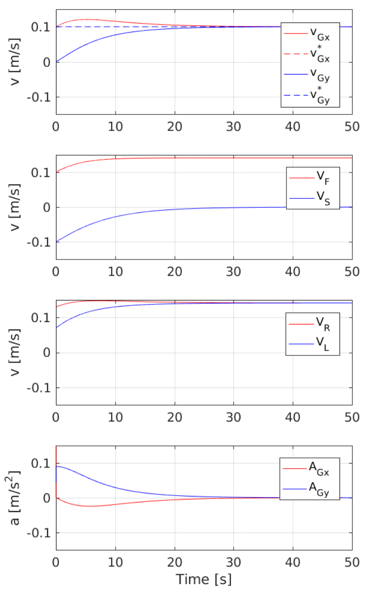
$$\begin{cases} V_R = & V_F - sign(V_F)KV_S \\ V_L = & V_F + sign(V_F)KV_S \end{cases} \tag{16}$$

where K is a controller gain that can be adjusted.

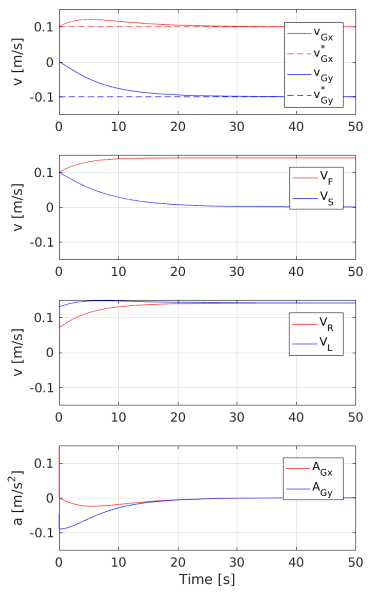
2 Testing of the proposed control law

The control law derived in the previous section can be tested in simulation using the Matlab/GNU Octave scripts included in Appendix A and B. The function *simulate_vehicle.m* integrates the equations of motion of the vehicle using Euler's method. During the simulation, the speed of the two wheels is adjusted every certain time interval the same way as a digital controller would update the speed order of the wheel actuators in real practice. The script generates a table of data that can be used to produce plots of the relevant variables and animations. A detailed description of how the code works can be found in the inline comments in the code in the appendices.

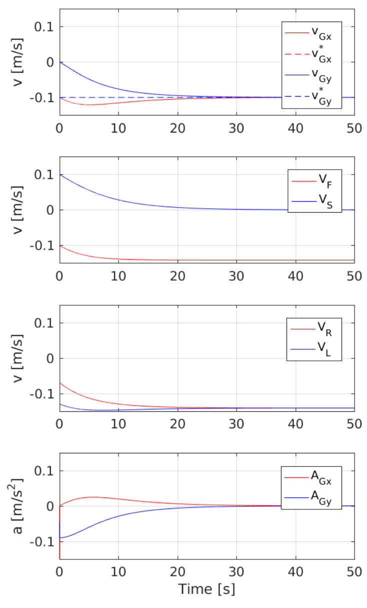
Several tests have been carried out using different speed orders from the operator of the vehicle, some results are shown in Figures 10 and 3. In brief, the controller is able to bring the vehicle to the desired speed and direction of motion. Moreover, the speed profile, once the vehicle starts moving, is smoothly varying.



(a) V_X = 0.1 m/s, V_Y = 0.1 m/s

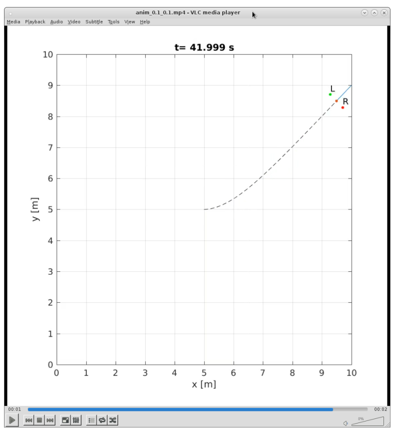


(b) V_X = 0.1 m/s, V_Y = -0.1 m/s

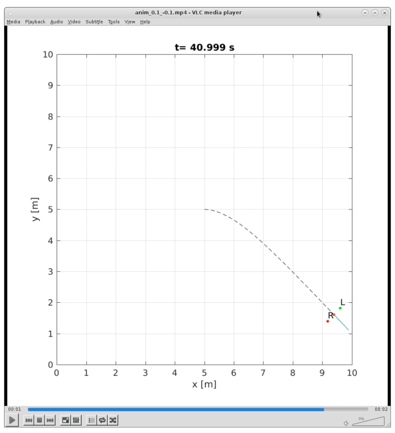


(c) V_X = -0.1 m/s, V_Y = -0.1 m/s

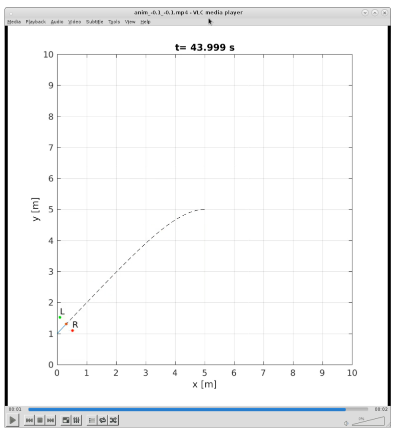
Figure 2: Evolution of the speed for different reference values.



(a) V_X = 0.1 m/s, V_Y = 0.1 m/s



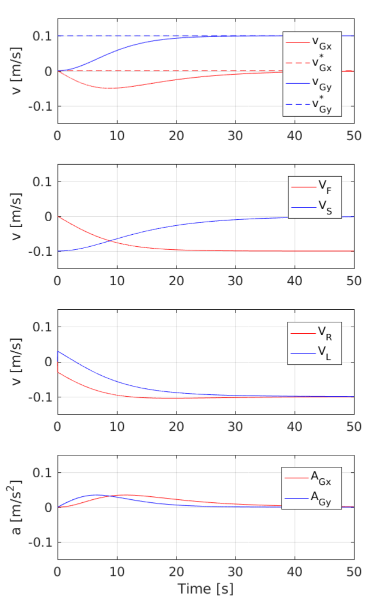
(b) V_X = 0.1 m/s, V_Y = -0.1 m/s



(c) V_X = -0.1 m/s, V_Y = -0.1 m/s

Figure 3: Screenshot of the animation generated by the script for different reference values.

On the other hand, an effect that may not be desirable is observed when V_F is close to zero. If V_F is positive, the vehicle will move forwards, whereas if V_F is negative, the vehicle will move backwards. This situation is illustrated in Figures 4 and 5 where a small variation of V_F is seen to have a great effect on the speed profile of the vehicle. It is not clear at this stage if this effect is a concerning problem or not because the operator could manoeuvre to avoid this situation. If this effect was deemed a problem, the control law could be modified in order to make sure the direction of motion is always forwards (or backwards) and include a switch along with the joystick allowing the operator to choose which direction the vehicle should move.


(a) $V_X = 0$, $V_Y = 0.1$ m/s

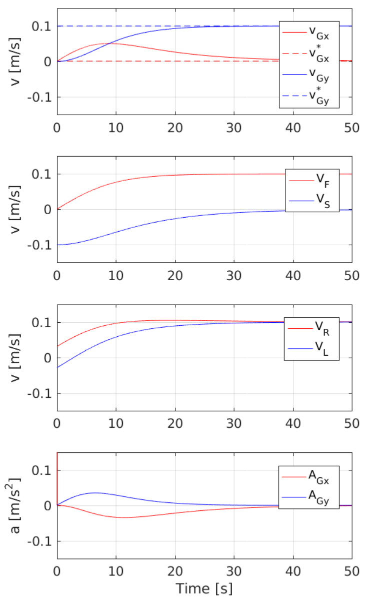
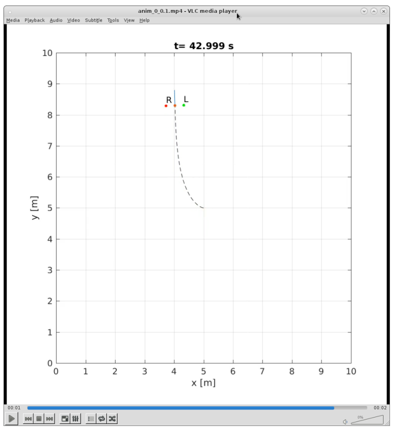
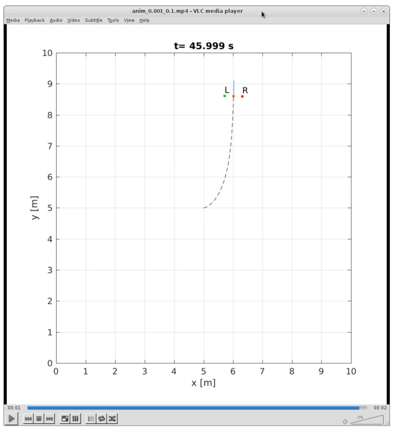

(b) $V_X = 0.001$ m/s, $V_Y = 0.1$ m/s

Figure 4: Evolution of the speed for different reference values.



(a) $V_X = 0$, $V_Y = 0.1$ m/s



(b) $V_X = 0.001$ m/s, $V_Y = 0.1$ m/s

Figure 5: Screenshot of the animation generated by the script for different reference values.

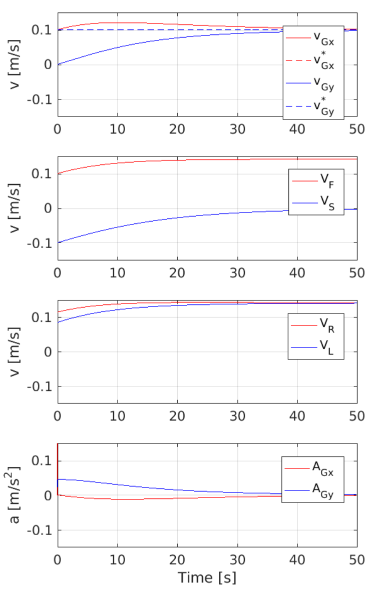
Another design parameter that needs to be chosen is the gain K in (16). From the sizing point of view, the gain has to be chosen to be small enough to not exceed the maximum rated speed of V_L and V_R, $V_{R,L}^{MAX}$, for the maximum expected values of V_F and V_S, V_F^{MAX} and V_S^{MAX} respectively. This can be ensured by choosing a value of K such that:

$$K \leq \frac{V_{R,L}^{MAX} - V_F^{MAX}}{V_S^{MAX}} \tag{17}$$

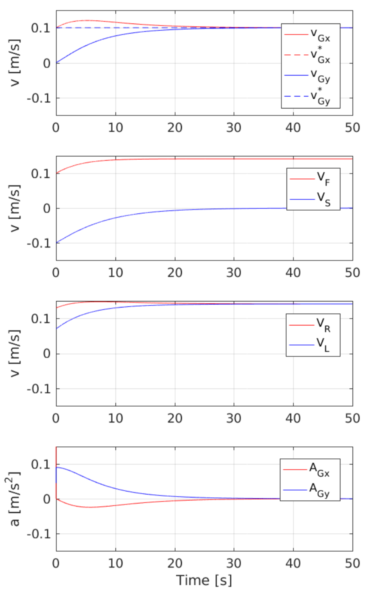
On the other hand, the choice of K has implications on how quick the vehicle achieves the target speed. A comparison of the evolution of the speed for different values of K is shown in Figures 6 and 7.

3 Acceleration and jerk limit

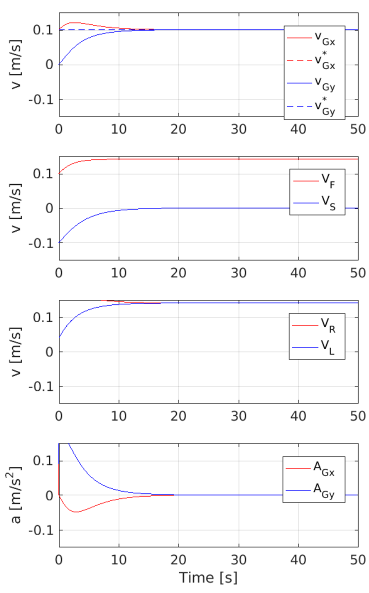
The control law presented in the previous section is designed to make the speed of the vehicle be as close as possible to the speed order given by the operator. If this was used in real practice, abrupt changes of the



(a) V_X, V_Y = 0.1 m/s, $K = 0.15$

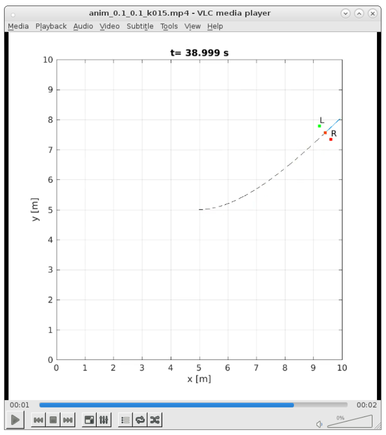


(b) V_X, V_Y = -0.1 m/s, $K = 0.3$

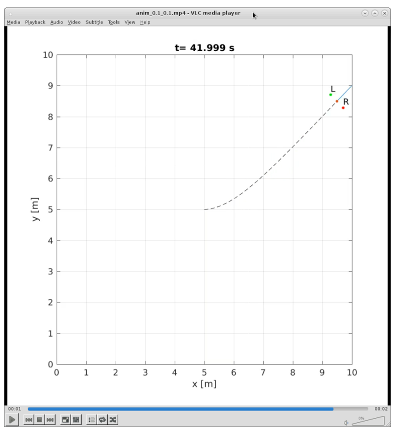


(c) V_X, V_Y = -0.1 m/s, $K = 0.6$

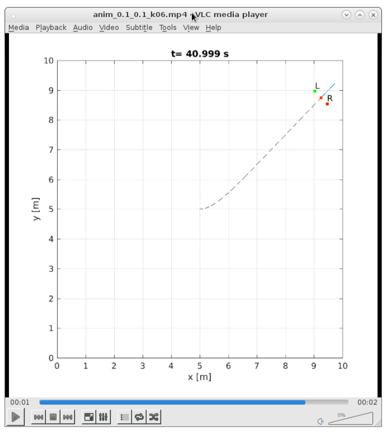
Figure 6: Evolution of the speed for different gain values.


anim_0.1_0.1_k015.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
t= 38.999 s
y [m]
x [m]
L
R
00:01
00:02

(a) V_X, V_Y = 0.1 m/s



(b) V_X, V_Y = -0.1 m/s


anim_0.1_0.1_k06.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
t= 40.999 s
y [m]
x [m]
L
R
00:01
00:02

(c) V_X, V_Y = -0.1 m/s

Figure 7: Screenshot of the animation generated by the script for different gain values.

speed would happen if for example the operator made sudden ”step-wise” changes of the position of the control joystick. Sudden changes of the speed imply large acceleration, which in turn requires large torque from the actuators. In order to avoid this problem, the speed profile is often modified in order to limit the maximum acceleration. Further, the so-called jerk (the derivative of the acceleration) is often limited as well in order to minimise the discomfort caused to the passenger of the vehicle.

Different methods can be used to limit the acceleration and the jerk of a reference speed signal, the signal flow diagram of one such method is shown in Figure 8. The method combines a rate limiter with a low-pass filter. The rate limiter calculates the derivative of the input signal, saturates the result and integrates the signal again. The rate limiter is a non-linear filter which only affects the signal if the derivative of the signal is greater than the limit set by its saturation stage. The rate limiter can be used to limit the acceleration while the low-pass filter can be tuned to limit the jerk of the resulting speed reference signal.

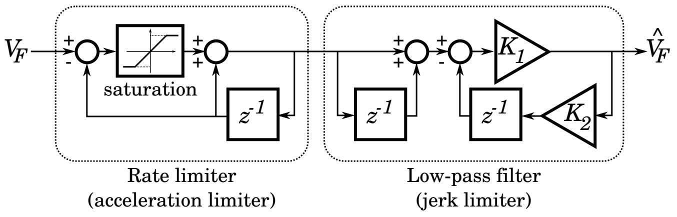
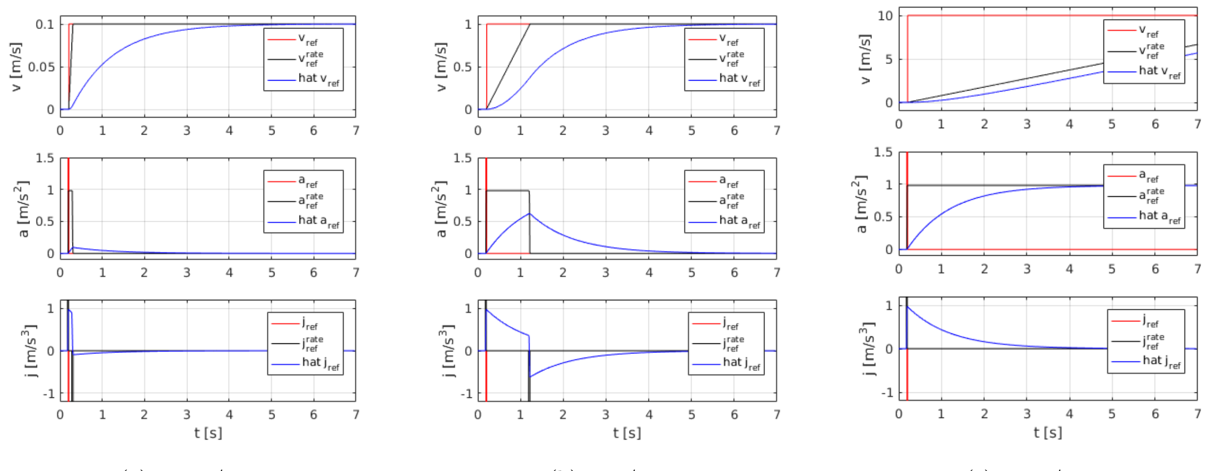


Figure 8: Diagram of the acceleration and jerk limiter

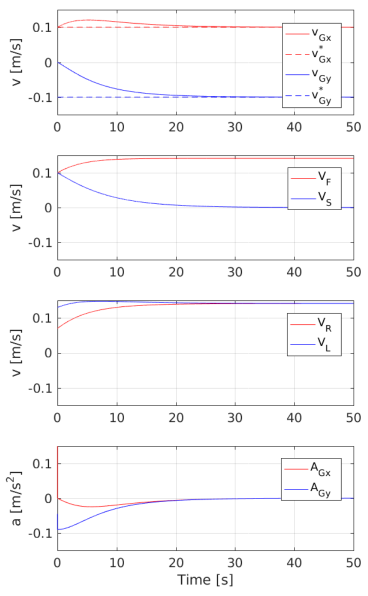
The response of the limiting mechanism to step changes of the speed reference of different magnitudes is shown in Figure 9. The source code of the demonstration can be found in Appendix C. The rate limiter in the example has been adjusted to limit the maximum acceleration to one tenth of the gravity (approximately 1 m/s^2) whereas the low pass limiter has been tuned to limit the jerk to approximately 1 m/s^3. The plot of the acceleration shows that the acceleration required for a step change of the speed can be very large. This is effectively reduced to an acceptable value by using the rate limiter (the black trace in Figure 9). However, the output of the rate limiter still requires abrupt changes of the acceleration, leading to high jerk during the transients. The addition of the low-pass filter at the output of the rate limiter effectively overcomes this problem as shown in the blue trace in Figure 9.



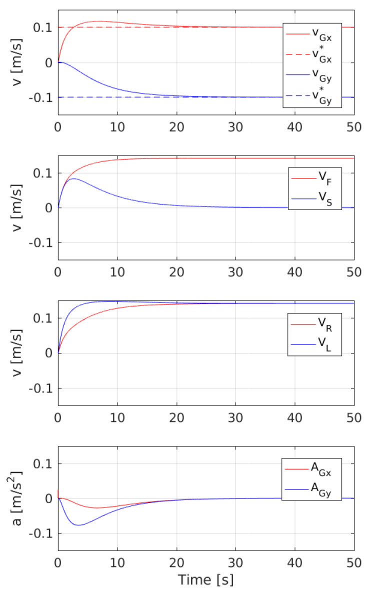
(a) 0.1 m/s step (b) 1 m/s step (c) 10 m/s step

Figure 9: Response of the acceleration and jerk limiter.

The limiting mechanism has been added to the simulation script described earlier in order to validate its operation. The source code of the new script can be found in Appendix D and E. The result of the simulation compared side by side with the result for the same speed reference in the previous section can be seen in Figure 10. The simulation shows that unlike the original controller where the acceleration and the jerk weren’t limited, the new system leads to a much smoother evolution of the speed of the vehicle in the very beginning of the simulation when the speed reference is passed to the controller.



(a) V_X = 0.1 m/s, V_Y = -0.1m/s non limited



(b) V_X = 0.1 m/s, V_Y = -0.1m/s limited

Figure 10: Evolution of the speed with and without acceleration and jerk limit.

Conclusions and remarks

This document has presented a basic analysis of the kinematics of a vehicle with two driving wheels together with a suitable control law that translates the operator commands into the speed order that is given to the actuators of the wheels. In this process, several findings that the designer should take into account have been highlighted. The proposed control law can produce either forward or backward motion when given a speed reference with little or no forward component. Further, several parameters have to be chosen before the implementation of the controller; specifically, a controller gain plus the settings of an acceleration and jerk limiting algorithm. Several software tools that help designing the controller of the vehicle have been presented; however, experimental tests carried on a real platform would be required in order to fully validate the controller design.

Appendix A: Source code of *simulate_vehicle.m*

```
%% We reset all variables and clean the mess
clc
clear
close all

%% We define a few relevant parameters for the simulation
% Characteristics of the vehicle:
d=0.3; %distance between the wheels

% Controller settings:
gain=0.3;
% The control law (as described in the document) is:
control_V=@(VF,VS) [VF-gain*VS*sign(VF);VF+gain*VS*sign(VF)];

% Speed command: we assume that the operator wants to move in a certain
% direction (e.g. to the North) and therefore they will keep aiming the
% joystick in that direction as the vehicle moves
VrefX=0.1;
VrefY=-0.1;
control_Vref=@(theta1,theta2) [VrefX*cos(theta1+theta2)+VrefY*sin(theta1+theta2);...
    VrefX*sin(theta1+theta2)-VrefY*cos(theta1+theta2)];

% Simulation parametres:
Tsim=1e-3; % what is the time step required to perform the simulation (1 ms)
Tcontrol=1e-2; % what is the sampling period of the speed controller (10 ms)
```

```
Ncontrol=Tcontrol/Tsim;

tmax_sim=50; % what is the time horizon of the simulation (100 s)
SimSteps=tmax_sim/Tsim;
ControlSteps=floor(tmax_sim/Tcontrol);

% Two functions that are used to calculate the speed of the vehicle once
% the speed of the two wheels is known and the orientation of the vehicle
% is known:
% speed
calc_VG=@(VR,VL,theta1,theta2) (VR+VL)/2*[cos(theta1+theta2);sin(theta1+theta2)];
% rotational speed
calc_Omega=@(VR,VL) (VR-VL)/(2*d);

% The following variables are used during the simulation
m=1; % this counts the number of simulation steps
n=1; % this counts the number of control cycles
VR=0;
VL=0;
theta1=0;
theta2=0;
theta12=0;
x=5; % this is the initial position of the vehicle
y=5;
vgx=0;
vgy=0;

% The following tables will store the value of different variables during
% the simulation:
tab_t=zeros(1,SimSteps);
tab_x=zeros(1,SimSteps);
tab_y=zeros(1,SimSteps);
tab_vgx=zeros(1,SimSteps);
tab_vgy=zeros(1,SimSteps);
tab_VF=zeros(1,SimSteps);
tab_VS=zeros(1,SimSteps);
tab_VR=zeros(1,SimSteps);
tab_VL=zeros(1,SimSteps);
tab_theta1=zeros(1,SimSteps);
tab_theta2=zeros(1,SimSteps);

% The following loop is the main loop of the simulation that runs from t=0
% until the end of the simulation and updates the control signal of the
% speed controller and calculates the position of the vehicle
for t=0:Tsim:tmax_sim
    % the following code is the code of the controller, which updates the
    % speed of the wheels in order to make the vehicle follow the commands
    % of the operator:
    if(mod(t,Tcontrol)==0)
        temp1=control_Vref(theta1,theta2);
        temp2=control_V(temp1(1),temp1(2));
        VR=temp2(1);
        VL=temp2(2);
        n=n+1;
    end
    % the following lines calculate the speeds and the new position of the
    % vehicle
    temp3=calc_VG(VR,VL,theta1,theta2);
    vgx=temp3(1);
    vgy=temp3(2);
    x=x+vgx*Tsim;
    y=y+vgy*Tsim;
    temp4=calc_Omega(VR,VL);
    theta12=theta12+temp4*Tsim;
    theta1=atan2(y,x);
    theta2=mod(theta12-theta1,2*pi);

    % the following code stores the data point in the tables
    tab_t(m)=t;
    tab_x(m)=x;
    tab_y(m)=y;
    tab_vgx(m)=vgx;
    tab_vgy(m)=vgy;
    tab_VF(m)=temp1(1);
    tab_VS(m)=temp1(2);
    tab_VR(m)=VR;
    tab_VL(m)=VL;
    tab_theta1(m)=theta1;
    tab_theta2(m)=theta2;
    m=m+1;
end

%% Once the simulation is over, we generate some plots to show the results:

% first we produce an animation of the position of the vehicle
Texport=1;
Nexport=tmax_sim/Texport;
p=figure('Position',[100,100,600,600],'Name',['Vrefx= ', num2str(VrefX),...
    ' m/s, Vrefy= ', num2str(VrefY), 'm/s']);
for l=1:Nexport
    plot(tab_x(1:(l*Texport/Tsim)),tab_y(1:(l*Texport/Tsim)),'k--');
    hold on; grid on
    plot_vehicle(tab_x(l*Texport/Tsim),tab_y(l*Texport/Tsim),...
        tab_theta1(l*Texport/Tsim),tab_theta2(l*Texport/Tsim),...
        tab_vgx(l*Texport/Tsim),tab_vgy(l*Texport/Tsim),d);
    p.CurrentAxes.XTick=0:1:10;
    p.CurrentAxes.YTick=0:1:10;
    title(['t= ' num2str(tab_t(l*Texport/Tsim)) ' s']);
    xlabel('x [m]');
    ylabel('y [m]');
%      print(['./frames/f' num2str(l,'%04d') '.png'],'-dpng');
    pause(0.01);
end
%run 'ffmpeg -i ./frames/f%04d.png ./anim.mp4' in order to generate the
%movie

%%
p=figure('Position',[100,100,400,700],'Name',['Vrefx= ', num2str(VrefX),...
    ' m/s, Vrefy= ', num2str(VrefY), 'm/s']);
subplot(4,1,1);
plot(tab_t,tab_vgx,'r');
grid on
hold on
plot(tab_t,VrefX*ones(1,length(tab_t)),'r--');
plot(tab_t,tab_vgy,'b');
plot(tab_t,VrefY*ones(1,length(tab_t)),'b--');
axis([0 tmax_sim -0.15 0.15]);
legend('v_{Gx}','v_{Gx}^*','v_{Gy}','v_{Gy}^*');
ylabel('v [m/s]');

subplot(4,1,2);
plot(tab_t,tab_VF,'r');
grid on
hold on
plot(tab_t,tab_VS,'b');
axis([0 tmax_sim -0.15 0.15]);
legend('V_F','V_S');
ylabel('v [m/s]');

subplot(4,1,3);
plot(tab_t,tab_VR,'r');
grid on
hold on
plot(tab_t,tab_VL,'b');
axis([0 tmax_sim -0.15 0.15]);
legend('V_R','V_L');
ylabel('v [m/s]');

subplot(4,1,4);
plot([-Tcontrol, tab_t(1:Ncontrol:(SimSteps-Ncontrol))],[tab_vgx(1)/Tsim (tab_vgx(Ncontrol:
    Ncontrol:(SimSteps-1))-tab_vgx(1:Ncontrol:(SimSteps-Ncontrol)))/Tsim],'r');
grid on
hold on
plot([-Tcontrol, tab_t(1:Ncontrol:(SimSteps-Ncontrol))],[tab_vgy(1)/Tsim (tab_vgy(Ncontrol:
    Ncontrol:(SimSteps-1))-tab_vgy(1:Ncontrol:(SimSteps-Ncontrol)))/Tsim],'b');
axis([0 tmax_sim -0.15 0.15]);
legend('A_{Gx}','A_{Gy}');
ylabel('a [m/s^2]');
xlabel('Time [s]');

print(['plot_' num2str(VrefX) '_' num2str(VrefY) '.png'],'-dpng');
```

Appendix B: Source code of *plot_vehicle.m*

```matlab
function plot_vehicle(x,y,theta1,theta2,vgx,vgy,d)
    plot([x,x+vgx*5],[y,y+vgy*5],'-');
    hold on;
    plot(x,y,'.','MarkerSize',10);
    plot(x+d*sin(theta1+theta2),y-d*cos(theta1+theta2),'r.','MarkerSize',10);
    text(x+d*sin(theta1+theta2),y-d*cos(theta1+theta2)+0.2,'R');
    plot(x-d*sin(theta1+theta2),y+d*cos(theta1+theta2),'g.','MarkerSize',10);
    text(x-d*sin(theta1+theta2),y+d*cos(theta1+theta2)+0.2,'L');

    hold off;
    axis([0 10 0 10]);
```

Appendix C: Source code of *demo_limiter.m*

```matlab
%% We reset all variables and clean the mess
clc
clear
close all

% Simulation parametres:
Tsim=1e-3; % what is the time step required to perform the simulation (1 ms)
Tcontrol=1e-2; % what is the sampling period of the speed controller (10 ms)
Ncontrol=Tcontrol/Tsim;
tmax_sim=7; % what is the time horizon of the simulation (100 s)
SimSteps=tmax_sim/Tsim;
ControlSteps=floor(tmax_sim/Tcontrol);

% In this demonstration we test the response of the limiting mechanism when
% a step change of the speed reference is requested. The following
% parameter set the amplitude of the step:
TestAmplitude=1;

% Parameters rate limiter plus low pass filter
% Maximum acceleration allowed:
limder=0.1*9.81*Tcontrol; % this corresponds to 10% of the gravity (1 m/s^2)
% Time constant of the low pass filter
tau=1; % 1s for a maximum acceleration of 1 m/s^2 gives a jerk of 1 m/s^3
K1=(Tcontrol+2*tau)/Tcontrol;
K2=(Tcontrol-2*tau)/Tcontrol;

% The following lines initialise the variables required for the simulation
x1=0;
vrefp=0;
vrefrate=0;

tab_t=zeros(1,SimSteps);
tab_vref=zeros(1,SimSteps);
tab_vrefp=zeros(1,SimSteps);
tab_vrefrate=zeros(1,SimSteps);

% the following code simulates the operation of the limiting mechanism
m=1;
for t=0:Tsim:tmax_sim;
    if(mod(t,Tcontrol)==0)
        vref=(t>0.2)*TestAmplitude;
        vrefrate=min(max(vref-x1,-limder),limder)+x1;
        x1n=vrefrate;
        vrefp=(x1n+x1-K2*vrefp)/K1;
        x1=x1n;
    end
    tab_t(m)=t;
    tab_vref(m)=vref;
    tab_vrefp(m)=vrefp;
    tab_vrefrate(m)=vrefrate;

    m=m+1;
end

```

```
%% Next we produce a few plots to show the performance of the limiting mechanism

p=figure('Position',[100,100,400,500]);
subplot(3,1,1);
plot(tab_t,tab_vref,'r');
grid on
hold on
plot(tab_t,tab_vrefrate,'k');
plot(tab_t,tab_vrefp,'b');
axis([0 tmax_sim -TestAmplitude*0.1 TestAmplitude*1.1]);
legend('v_{ref}','v_{ref}^{rate}','hat v_{ref}');
ylabel('v [m/s]');

avref=(tab_vref((Ncontrol+1):Ncontrol:(SimSteps))-tab_vref(1:Ncontrol:...
    (SimSteps-Ncontrol)))/(Tsim*Ncontrol);
avrefrate=(tab_vrefrate((Ncontrol+1):Ncontrol:(SimSteps))...
    -tab_vrefrate(1:Ncontrol:(SimSteps-Ncontrol)))/(Tsim*Ncontrol);
avrefp=(tab_vrefp((Ncontrol+1):Ncontrol:(SimSteps))...
    -tab_vrefp(1:Ncontrol:(SimSteps-Ncontrol)))/(Tsim*Ncontrol);

subplot(3,1,2);
plot(tab_t(1:Ncontrol:(SimSteps-Ncontrol)),avref,'r');
grid on
hold on
plot(tab_t(1:Ncontrol:(SimSteps-Ncontrol)),avrefrate,'k');
plot(tab_t(1:Ncontrol:(SimSteps-Ncontrol)),avrefp,'b');
legend('a_{ref}','a_{ref}^{rate}','hat a_{ref}');
axis([0 tmax_sim -0.1 1.5]);
ylabel('a [m/s^2]');

jvref=(avref(2:(ControlSteps-1))-avref(1:(ControlSteps-2)))/(Tsim*Ncontrol);
jvrefrate=(avrefrate(2:(ControlSteps-1))-avrefrate(1:(ControlSteps-2)))...
    /(Tsim*Ncontrol);
jvrefp=(avrefp(2:(ControlSteps-1))-avrefp(1:(ControlSteps-2)))/(Tsim*Ncontrol);

subplot(3,1,3);
plot(tab_t(1:Ncontrol:(SimSteps-2*Ncontrol)),jvref,'r');
grid on
hold on
plot(tab_t(1:Ncontrol:(SimSteps-2*Ncontrol)),jvrefrate,'k');
plot(tab_t(1:Ncontrol:(SimSteps-2*Ncontrol)),jvrefp,'b');
legend('j_{ref}','j_{ref}^{rate}','hat j_{ref}');
axis([0 tmax_sim -1.2 1.2]);
ylabel('j [m/s^3]');
xlabel('t [s]');
```

Appendix D: Source code of *simulate_vehicle_2.m*

```
%% We reset all variables and clean the mess
clc
clear
% close all

%% We define a few relevant parameters for the simulation
% Characteristics of the vehicle:
d=0.3; %distance between the wheels

% Controller settings:
gain=0.3;
% The control law (as described in the document) is:
control_V=@(VF,VS) [VF-gain*VS*sign(VF);VF+gain*VS*sign(VF)];

% Speed command: we assume that the operator wants to move in a certain
% direction (e.g. to the North) and therefore they will keep aiming the
% joystick in that direction as the vehicle moves
VrefX=0.1;
VrefY=-0.1;
control_Vref=@(theta1,theta2) [VrefX*cos(theta1+theta2)+VrefY*sin(theta1+theta2);VrefX*sin(
    theta1+theta2)-VrefY*cos(theta1+theta2)];

% Simulation parametres:
Tsim=1e-3; % what is the time step required to perform the simulation (1 ms)
```

```
Tcontrol=1e-2; % what is the sampling period of the speed controller (10 ms)
Ncontrol=Tcontrol/Tsim;

tmax_sim=50; % what is the time horizon of the simulation (100 s)
SimSteps=tmax_sim/Tsim;
ControlSteps=floor(tmax_sim/Tcontrol);

% Two functions that are used to calculate the speed of the vehicle once
% the speed of the two wheels is known and the orientation of the vehicle
% is known:
% speed
calc_VG=@(VR,VL,theta1,theta2) (VR+VL)/2*[cos(theta1+theta2);sin(theta1+theta2)];
% rotational speed
calc_Omega=@(VR,VL) (VR-VL)/(2*d);

% The following variables are used during the simulation
m=1; % this counts the number of simulation steps
n=1; % this counts the number of control cycles
VR=0;
VL=0;
theta1=0;
theta2=0;
theta12=0;
x=5; % this is the initial position of the vehicle
y=5;
vgx=0;
vgy=0;
x1F=0;
x1S=0;
VFp=0;
VSp=0;

% The following tables will store the value of different variables during
% the simulation:
tab_t=zeros(1,SimSteps);
tab_x=zeros(1,SimSteps);
tab_y=zeros(1,SimSteps);
tab_vgx=zeros(1,SimSteps);
tab_vgy=zeros(1,SimSteps);
tab_VF=zeros(1,SimSteps);
tab_VS=zeros(1,SimSteps);
tab_VR=zeros(1,SimSteps);
tab_VL=zeros(1,SimSteps);
tab_theta1=zeros(1,SimSteps);
tab_theta2=zeros(1,SimSteps);

% The following loop is the main loop of the simulation that runs from t=0
% until the end of the simulation and updates the control signal of the
% speed controller and calculates the position of the vehicle
for t=0:Tsim:tmax_sim
    % the following code is the code of the controller, which updates the
    % speed of the wheels in order to make the vehicle follow the commands
    % of the operator:
    if(mod(t,Tcontrol)==0)
        temp0=control_Vref(theta1,theta2);
        temp1=calc_ratelimit(temp0(1),x1F,VFp);
        VFp=temp1(1);
        x1F=temp1(2);
        temp1=calc_ratelimit(temp0(2),x1S,VSp);
        VSp=temp1(1);
        x1S=temp1(2);
        temp2=control_V(VFp,VSp);
        VR=temp2(1);
        VL=temp2(2);
        n=n+1;
    end
    % the following lines calculate the speeds and the new position of the
    % vehicle
    temp3=calc_VG(VR,VL,theta1,theta2);
    vgx=temp3(1);
    vgy=temp3(2);
    x=x+vgx*Tsim;
    y=y+vgy*Tsim;
    temp4=calc_Omega(VR,VL);
    theta12=theta12+temp4*Tsim;
    theta1=atan2(y,x);
```

```matlab
    theta2=mod(theta12-theta1,2*pi);

    % the following code stores the data point in the tables
    tab_t(m)=t;
    tab_x(m)=x;
    tab_y(m)=y;
    tab_vgx(m)=vgx;
    tab_vgy(m)=vgy;
    tab_VF(m)=VFp;
    tab_VS(m)=VSp;
    tab_VR(m)=VR;
    tab_VL(m)=VL;
    tab_theta1(m)=theta1;
    tab_theta2(m)=theta2;
    m=m+1;
end

%% Once the simulation is over, we generate some plots to show the results:

% first we produce an animation of the position of the vehicle
Texport=1;
Nexport=tmax_sim/Texport;
p=figure('Position',[100,100,600,600],'Name',['Vrefx= ', num2str(VrefX),...
    ' m/s, Vrefy= ', num2str(VrefY), 'm/s']);
for l=1:Nexport
    plot(tab_x(1:(l*Texport/Tsim)),tab_y(1:(l*Texport/Tsim)),'k--');
    hold on; grid on
    plot_vehicle(tab_x(l*Texport/Tsim),tab_y(l*Texport/Tsim),...
        tab_theta1(l*Texport/Tsim),tab_theta2(l*Texport/Tsim),...
        tab_vgx(l*Texport/Tsim),tab_vgy(l*Texport/Tsim),d);
    p.CurrentAxes.XTick=0:1:10;
    p.CurrentAxes.YTick=0:1:10;
    title(['t= ' num2str(tab_t(l*Texport/Tsim)) ' s']);
    xlabel('x [m]');
    ylabel('y [m]');
%     print(['./frames/f' num2str(l,'%04d') '.png'],'-dpng');
    pause(0.01);
end
%run 'ffmpeg -i ./frames/f%04d.png ./anim.mp4' in order to generate the
%movie

%%
p=figure('Position',[100,100,400,700],'Name',['Vrefx= ', num2str(VrefX),...
    ' m/s, Vrefy= ', num2str(VrefY), 'm/s']);
subplot(4,1,1);
plot(tab_t,tab_vgx,'r');
grid on
hold on
plot(tab_t,VrefX*ones(1,length(tab_t)),'r--');
plot(tab_t,tab_vgy,'b');
plot(tab_t,VrefY*ones(1,length(tab_t)),'b--');
axis([0 tmax_sim -0.15 0.15]);
legend('v_{Gx}','v_{Gx}^*','v_{Gy}','v_{Gy}^*');
ylabel('v [m/s]');

subplot(4,1,2);
plot(tab_t,tab_VF,'r');
grid on
hold on
plot(tab_t,tab_VS,'b');
axis([0 tmax_sim -0.15 0.15]);
legend('V_F','V_S');
ylabel('v [m/s]');

subplot(4,1,3);
plot(tab_t,tab_VR,'r');
grid on
hold on
plot(tab_t,tab_VL,'b');
axis([0 tmax_sim -0.15 0.15]);
legend('V_R','V_L');
ylabel('v [m/s]');

subplot(4,1,4);
plot([-Tcontrol, tab_t(1:Ncontrol:(SimSteps-Ncontrol))],[tab_vgx(1)/Tsim (tab_vgx(Ncontrol:
    Ncontrol:(SimSteps-1))-tab_vgx(1:Ncontrol:(SimSteps-Ncontrol)))/Tsim],'r');
```

```
grid on
hold on
plot([-Tcontrol, tab_t(1:Ncontrol:(SimSteps-Ncontrol))],[tab_vgy(1)/Tsim (tab_vgy(Ncontrol:
    Ncontrol:(SimSteps-1))-tab_vgy(1:Ncontrol:(SimSteps-Ncontrol)))/Tsim],'b');
axis([0 tmax_sim -0.15 0.15]);
legend('A_{Gx}','A_{Gy}');
ylabel('a [m/s^2]');
xlabel('Time [s]');

print(['plot3_' num2str(VrefX) '_' num2str(VrefY) '.png'],'-dpng');
```

Appendix E: Source code of *calc_ratelimit.m*

```
function out=calc_ratelimit(vref,x1,x2)
T=10e-3;
tau=1;
K1=(T+2*tau)/T;
K2=(T-2*tau)/T;
limder=0.1*9.81*T;

x1n=min(max(vref-x1,-limder),limder)+x1;
vrefp=(x1n+x1-K2*x2)/K1;
out=[vrefp,x1n];
end
```

Revision history

This document has gone through the following revisions:

1. (21 April 2017): Original release.
2. (21 April 2017): Added the missing source code of *calc_ratelimit.m* and corrected a few typos.

License