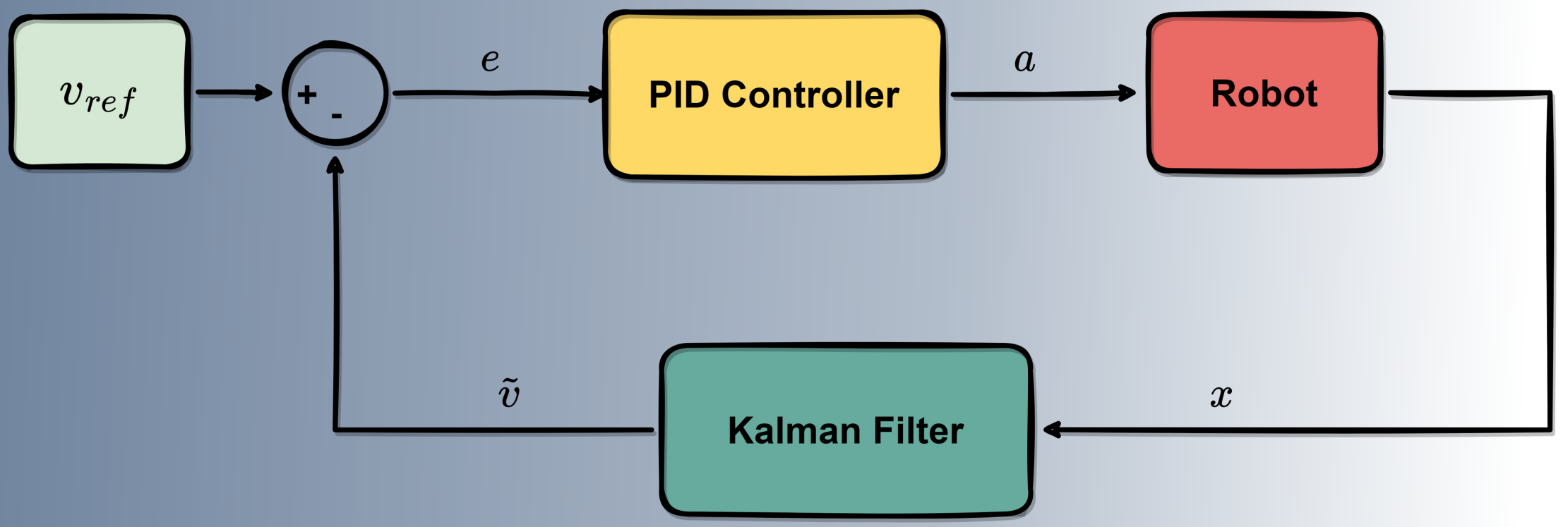
Robot Control using PID Controller and Kalman Filter

The objective is to estimate the Velocity using a Linear Kalman Filter and to control a robot using a PID Controller.



In this example, the design, tuning and implementation of a Kalman Filter and PID Controller applied to a Robot will be summarized.

Code Access

https://github.com/SaysWis/Robot_Control

Simone Bertoni
Wissam Sayssouk

Kalman filter - Model

The Kalman filter is one of the most important and common estimation approach that estimates an unknown state of a dynamic system from a series of noisy measurements.

The Kalman filter model assumes a linear dynamic system discretized:

$$x_k = Fx_{k-1} + Bu_k + w_k$$
$$z_k = Hx_k + v_k$$

where

$$w_k \sim \mathcal{N}(0, Q)$$
$$v_k \sim \mathcal{N}(0, R)$$

are the process and measurement noise, both drawn from a zero mean Gaussian noise, with covariance respectively Q and R.

Simone Bertoni
Wissam Sayssouk

Kalman filter - Implementation

The Kalman filter implementation calculates the predicted state and covariance estimate:

$$\hat{x}_{k|k-1} = Fx_{k-1|k-1} + Bu_k$$
$$\hat{P}_{k|k-1} = FP_{k-1|k-1}F^T + Q$$

the innovation and its covariance:

$$\tilde{y}_k = z_k - H\hat{x}_{k|k-1}$$
$$S_k = H\hat{P}_{k|k-1}H^T + R$$

the optimal Kalman gain:

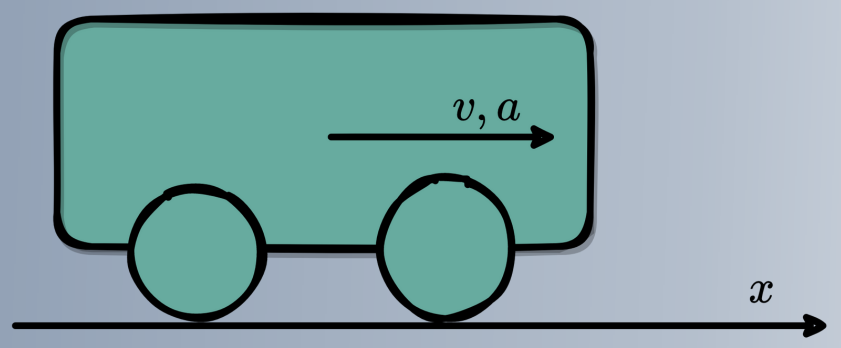
$$K_k = \hat{P}_{k|k-1}H^T S_k^{-1}$$

the updated state and covariance estimate:

$$x_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k$$
$$P_{k|k} = (I - K_kH)\hat{P}_{k|k-1}$$

Simone Bertoni
Wissam Sayssouk

Plant - Robot (Kinematic model) - 1



Consider a robot on a frictionless, straight lane.

We measure the position of the robot every Δt seconds, and we want to maintain a model of the robot's position and velocity.

The position and velocity of the robot are described by the linear state space: $x_k = \begin{bmatrix} x \\ v \end{bmatrix}$, where x: position and v: velocity.

We assume that a(k) is unknown and normally distributed with mean 0 and standard deviation σ_a. Then:

$$x_k = Fx_{k-1} + Ga_k$$

where $F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$ and $G = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}$

Simone Bertoni
Wissam Sayssouk

Plant - Robot (Kinematic model) - 2

Going back to our discretized nominal dynamic system:

$$x_k = Fx_{k-1} + Bu_k + w_k$$

where $Bu_k = 0$ under the assumption of no known control inputs.

Therefore

$$w_k \sim \mathcal{N}(0, Q)$$

where

$$Q = GG^T\sigma_a^2 = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 \end{bmatrix} \sigma_a^2$$

The noisy measurement of the position of the robot is

$$z_k = Hx_k + v_k$$

where $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$. v_k is normally distributed with mean 0 and standard deviation σ_z. Hence $R = \sigma_z^2$.

Simone Bertoni
Wissam Sayssouk

Matlab code - Introduction

The example is implemented using Matlab.

The control architecture is explained on the first slide.

The simulation is implemented discrete time using a "for loop", which simulates the plant, the PI controller and the Kalman filter for 100 seconds. The output of the PI controller is assumed to be the acceleration a_k provided to the robot.

The parameters chosen for the example are:

- $\Delta t = 0.1$
- $\sigma_a = 0.1$
- $\sigma_z = 0.01$
- $P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
- $k_p = 0.8$
- $k_i = 0.001$

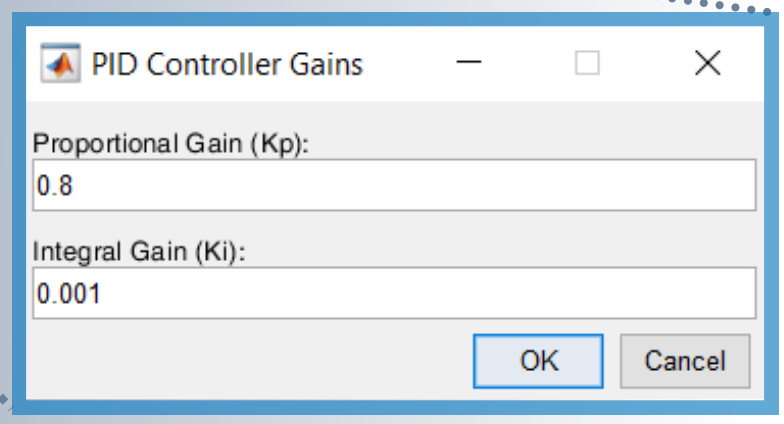
Simone Bertoni
Wissam Sayssouk

Matlab code (1/3) - Initialization

```
%% Initialization
Ts = 0.1;  % Sampling Time
time=0:Ts:100;
N = length(time)-1;

% For Plant
x0 = [0;0];     % Position, Velocity
x(:,1) = x0;

% For Kalman Filter
sigma_a = 0.1;
sigma_y = 0.01;
mu=0;
x_hat(:,1) = x0;
P_k{1} = 0*eye(2,2);
P_k_norm(1) = norm(P_k{1});

% Velocity Profile
Velocity(1:20) = 0; % m/s
Velocity(20:100) = 0.5;
Velocity(100:300) = 1.2;
Velocity(300:500) = 1;
Velocity(500:700) = 0;
Velocity(700:800) = 0.7;
Velocity(800:1001) = 0;
```

```
%% Plant Model
F = [1 Ts; 0 1];
G = [Ts^2/2; Ts];
H = [1 0];          % we have only position's measurement

% Check System Controllability
if(rank(ctrb(F,G))==2)
   disp('System is controllable !')
end
% Check System Observability
if(rank(obsv(F,H))==2)
   disp('System is observable !')
end
```

```
%% PID Controller
Kp = 0.8;
Ki = 1e-3;

[Kp,Ki]= mydialog;
```



When you run the example, a pop-up message will appear. It allows you to choose different values of Kp and Ki to see their impacts on the closed loop system.

Simone Bertoni
Wissam Sayssouk

Matlab code (2/3) - Simulation

```
%% Velocity Estimation using Kalman Filter
Q = G*G'*sigma_a^2;
R = sigma_y^2;
noise = sigma_y*randn(1,N+1)+mu; % Generate White Noise
```

```
%% Simulation
Error = zeros(1,length(N+1));
Prop = zeros(1,length(N+1));
Integral = zeros(1,length(N+1));

for t=1:N
    % PID Controller
    Error(t+1) = Velocity(t)-x_hat(2,t);

    Prop(t+1) = Error(t+1);
    Int(t+1)=Error(t+1)*Ts;
    Integral(t+1)= sum(Int);

    a(t+1) = Kp*Prop(t+1)+ Ki*Integral(t+1);

    % Plant Open Loop Simulation
    x(:,t+1) = F*x(:,t) + G*a(t+1);
    y(t+1) = H*x(:,t+1)+ noise(t+1); % Add noise to the measurement

    % Estimate the Velocity using Kalman Filter
    [x_up, P_k{t+1}] = Kalman_Filter(x_hat(:,t), P_k{t}, y(t), F, H, Q, R);
    x_hat(:,t+1) = x_up;
end
```

```
function [x_up, P_up] = Kalman_Filter(x_k_1, P_k_1, y, F, H, Q, R)

x_k = F*x_k_1;
P_k = F*P_k_1*F'+Q;                 % Project the State Covariance Ahead

Kk = P_k*H'*inv(H*P_k*H'+R);        % Kalman Gain
x_up = x_k+Kk*(y-H*x_k);            % Update Measurement with measurement
P_up = P_k-Kk*H*P_k;                % Update the Error Covariance

end
```

Simone Bertoni
Wissam Sayssouk

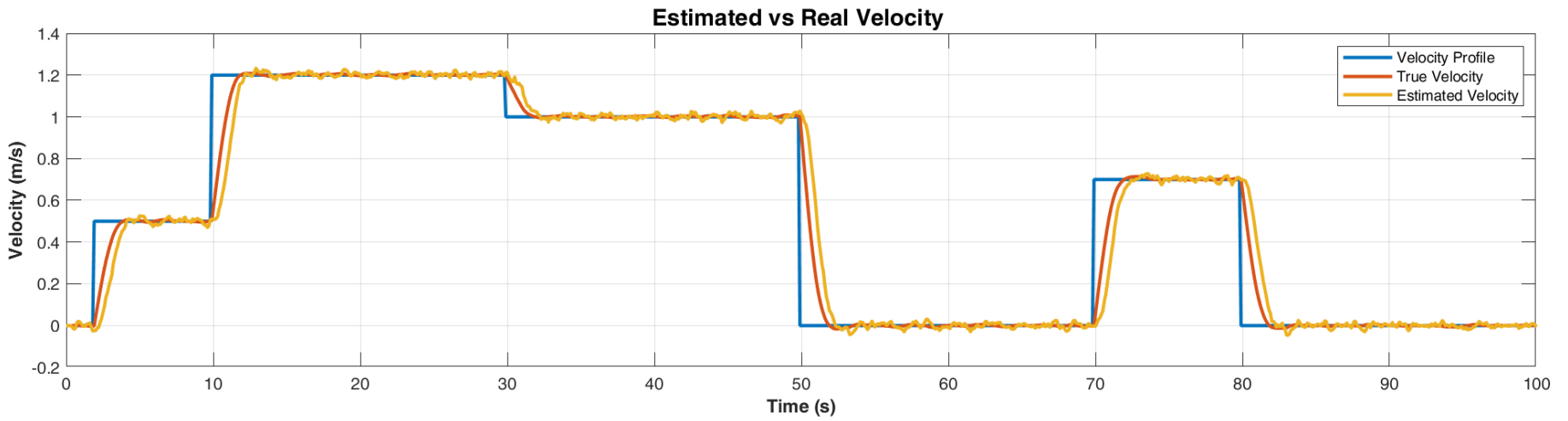
Matlab code (3/3) - Kalman filter - mydialog

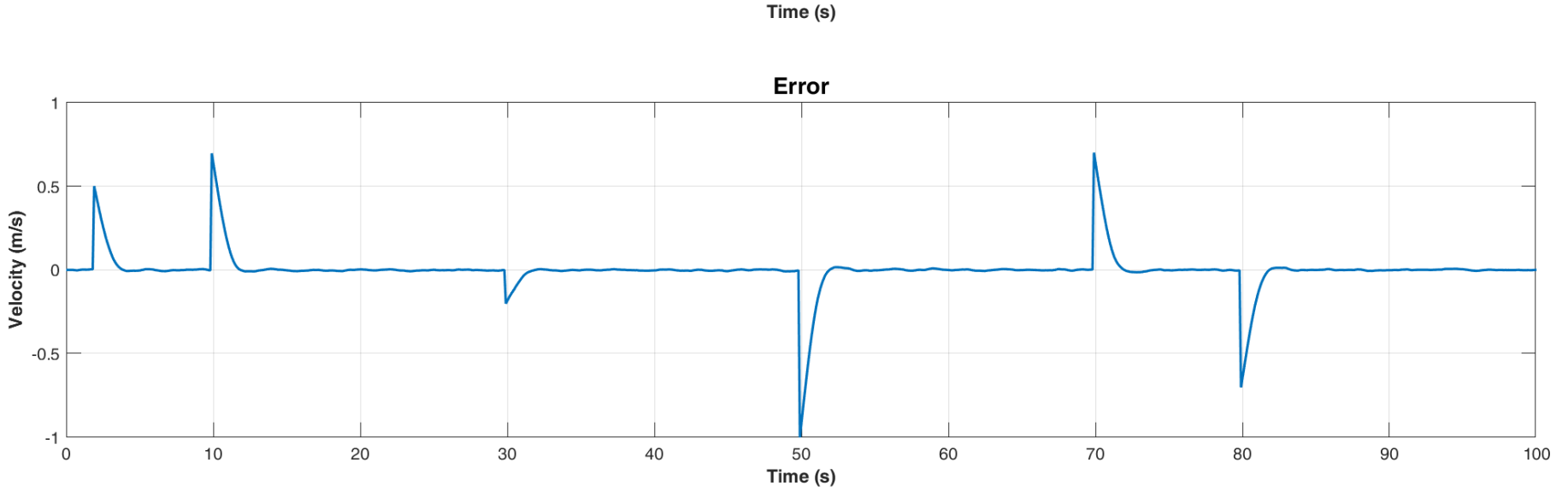
```
function [x_up, P_up] = Kalman_Filter(x_k_1, P_k_1, y, F, H, Q, R)

x_k = F*x_k_1;
P_k = F*P_k_1*F'+Q;                 % Project the State Covariance Ahead

Kk = P_k*H'*inv(H*P_k*H'+R);        % Kalman Gain
x_up = x_k+Kk*(y-H*x_k);            % Update Measurement with measurement
P_up = P_k-Kk*H*P_k;                % Update the Error Covariance

end
```

```
function[Kp,Ki]= mydialog
prompt = {'Proportional Gain (Kp):','Integral Gain (Ki):'};
dlgtitle = 'PID Controller Gains';
dims = [1 50; 1 50];
definput = {'0.8','0.001'};
answer = inputdlg(prompt,dlgtitle,dims,definput);
values = str2double(answer);


Kp = values(1);
Ki = values(2);
end
```

Simone Bertoni
Wissam Sayssouk

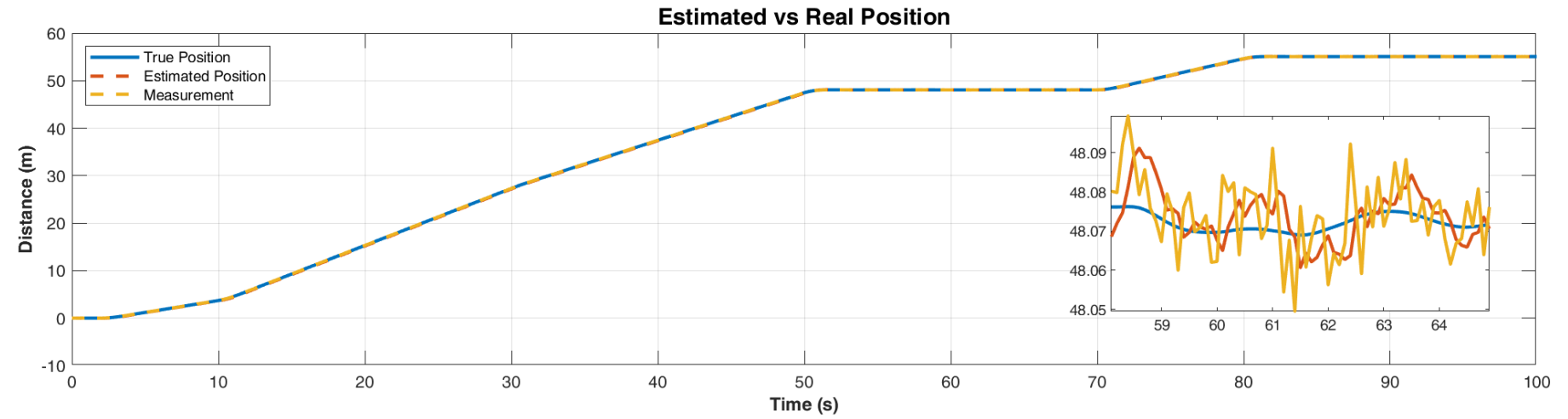
Simulation - Velocity

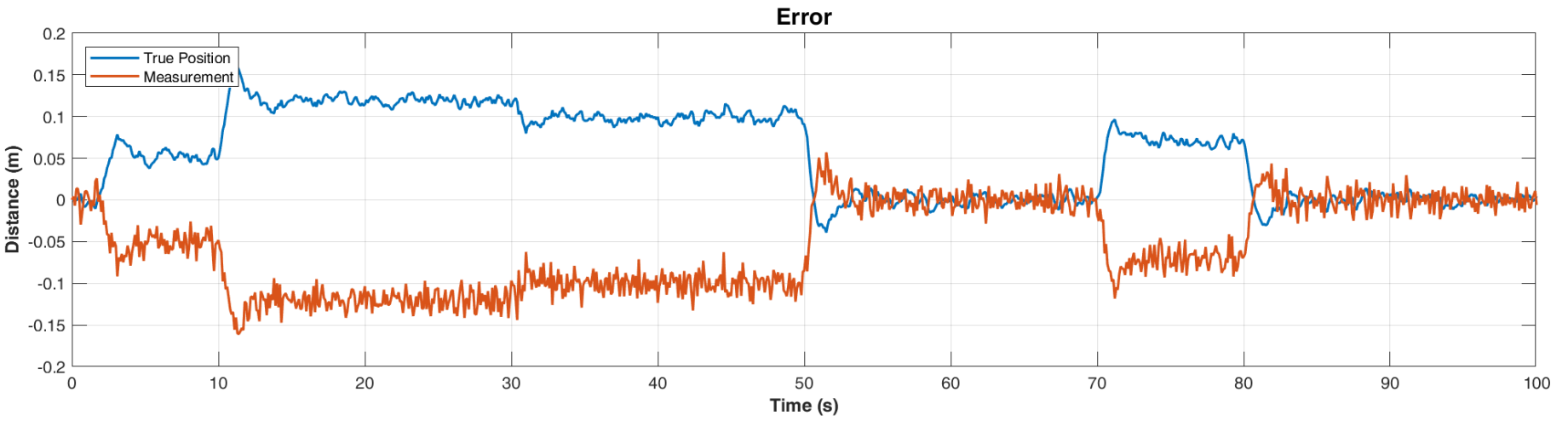




Simone Bertoni
Wissam Sayssouk

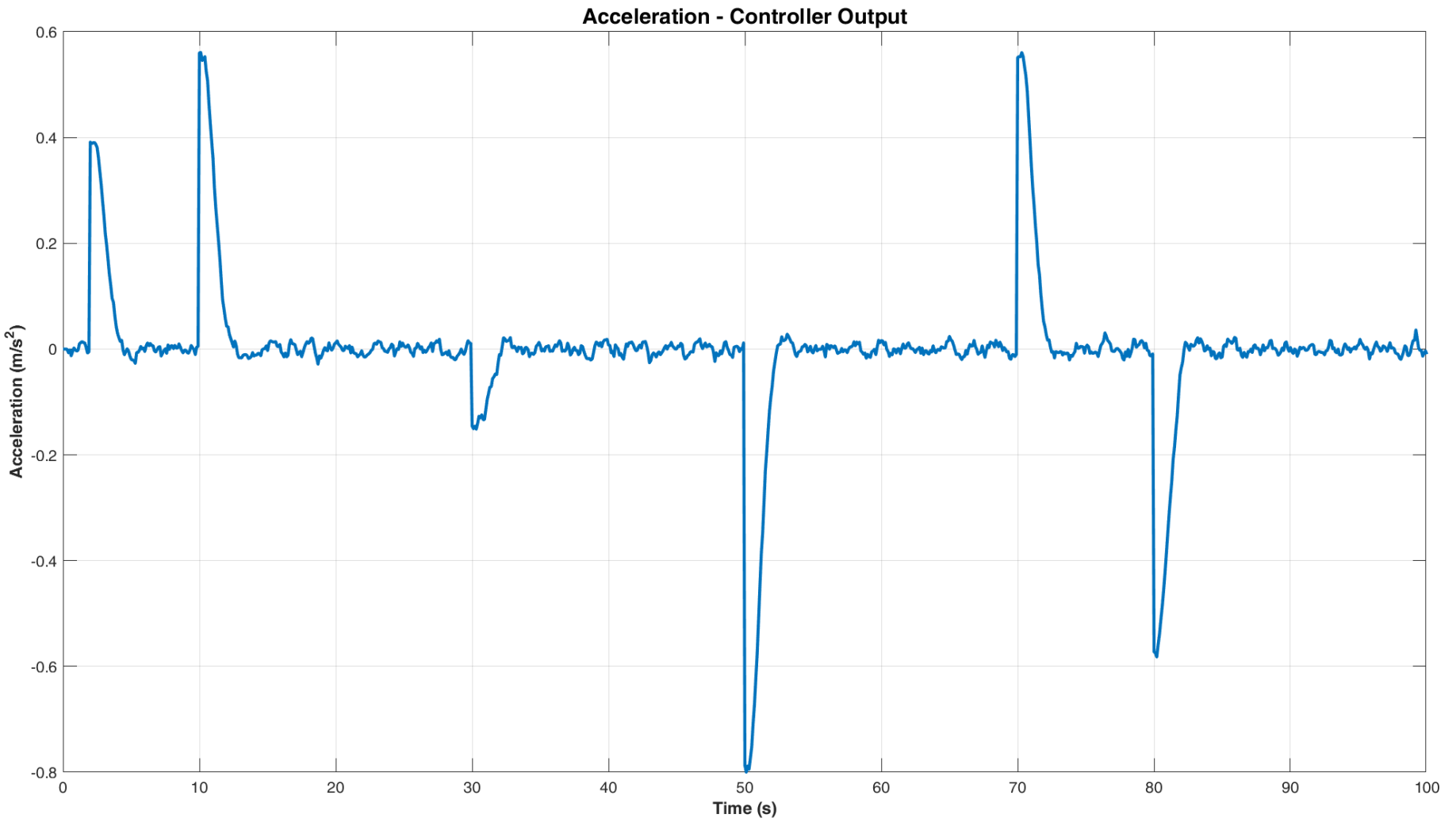
Simulation - Position





Simone Bertoni
Wissam Sayssouk

Simulation - Acceleration



Simone Bertoni
Wissam Sayssouk