Control of Mindstorms EV3

Maurice Filo

November 2023

1 Configurations

- Install Simulink/Matlab toolbox for EV3 Lego. https://ch.mathworks.com/hardware-support/lego-mindstorms-ev3-matlab.html.
- Provide an IP address to your EV3 using a router.
- Test connection in matlab with the right IP:

```
mylego = legoev3('10.0.0.99')
```

2 Data Collection

The first step is to obtain a model for the system to be controlled. To do this, we need to feed the system with a known input and measure the output obtained. The choice of the input signal is important, it needs to be "rich enough" to stimulate the dynamics of the system and extract as much information from it as possible. To this end we use a cosine signal as the input with a variable frequency. In simulink, this can be done using the chirp block.

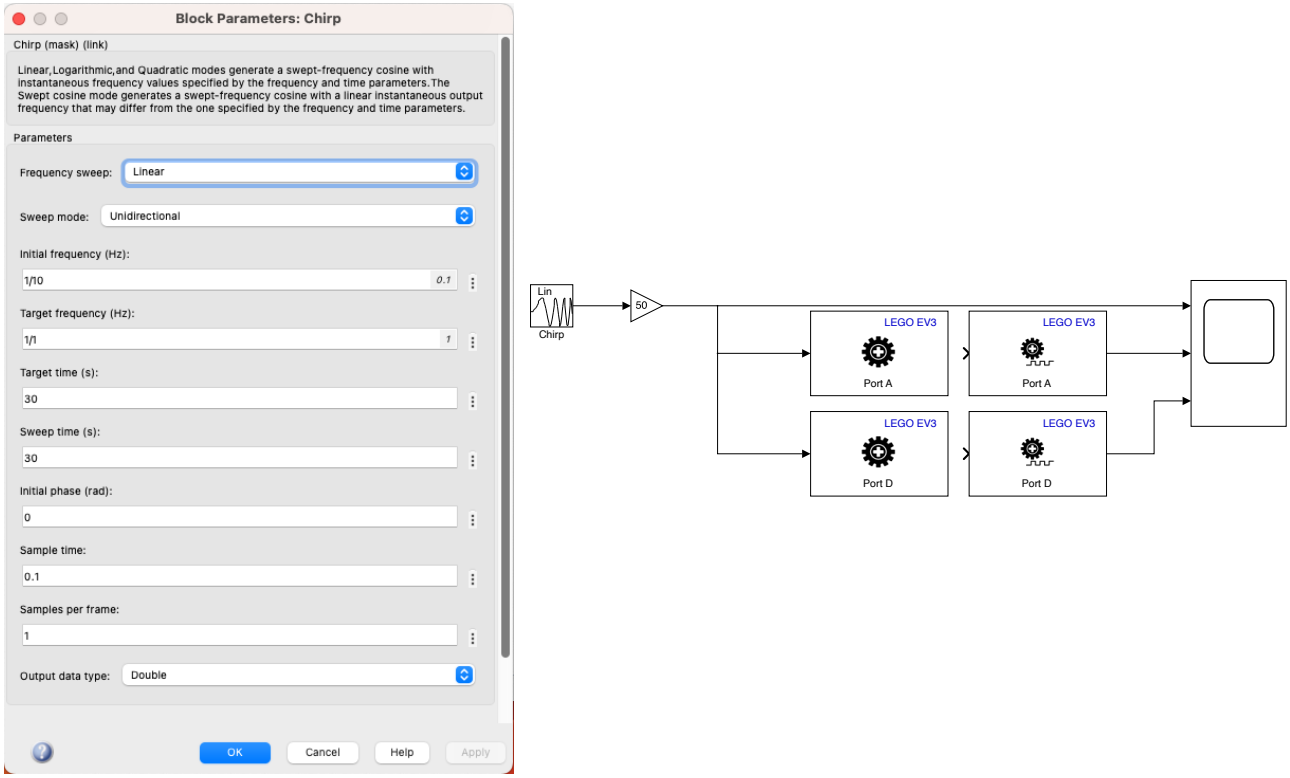

Figure 1: Simulink model to generate input/output data from the system.

Follow the following considerations:

- Build the model depicted in Fig. 1.
- Make sure to select the right ports in the motor and encoder blocks.
- Select the parameters of the chirp block as depicted in Fig. 1.
- The input to the motor blocks are signals that range between -100 and 100, while the chirp signal is a cosine with amplitude 1. A gain of 50 is used to put more power into the motor.
- The scope measures the applied input and the outputs from the two motors.
- Double click on the scope, and go to its settings (gear icon) to open it configuration properties. Under the Logging tab, check the box of Log data to workspace. This will save the data recorded by the scope in the workspace as a variable called ScopeData.
- Once the model is built, go to Modeling → Model Settings → Hardware Implementation → Hardware board → Lego Mindstorms EV3. In the same panel, go to Target Hardware resources → Host to Target Connection → Connection Type → Ethernet, and type in the correct IP address of the EV3 (e.g. 10.0.0.99). Finally click on Apply.
- Go to the Hardware tab, set the Stop Time to 30 and click on Monitor & Tune. The system should run and record the input/output signals in real-time for 30 seconds.
- Once the run is over, go to the command window in MATLAB and save the obtained data using the following command:

```
save Data
```

A file called Data.mat will appear in the current folder.

3 System Identification

Equipped with the Data.mat file, we are ready to perform the system identification. There are many ways to do that; however here we use a simple way to obtain a second order transfer function. The following code loads the data, extract the input/output signals and sampling time, and plot them. Then the command "tfest" is used to estimate a transfer function that fits the data. The transfer function is selected to have a second order denominator and a zeroth order numerator. Finally, the estimated system is simulated to the same chirp signal and plotted to be compared with the collected output measurements.

```
%% Clear Workspace
close all
clear
clc

%% Load Data
load Data.mat

%% Extract Input/Output Signals
t_u = double(logsout{3}.Values.Time);
u = squeeze(double(logsout{3}.Values.Data));
y1 = double(logsout{1}.Values.Data);
y2 = double(logsout{2}.Values.Data);
y = [y1, y2];

%% Sampling Time
```

```
Ts = t_u(2) - t_u(1);

%% Plot the Collected Data
LineWidth = 3;
plot(t_u, u, 'LineWidth', LineWidth);
hold on
plot(t_u, y1, 'LineWidth', LineWidth);
plot(t_u, y2, 'LineWidth', LineWidth);
grid on

%% System ID
ID_Data = iddata(y, u, Ts);
System_TF = tfest(ID_Data, 2, 0);
display(System_TF);

%% Simulate Identified System
y_hat = lsim(System_TF, u, t_u);
plot(t_u, y_hat(:,1), 'LineWidth', LineWidth, 'LineStyle', '--');
plot(t_u, y_hat(:,2), 'LineWidth', LineWidth, 'LineStyle', '--');
legend('Input', 'Measured Output 1', 'Measured Output 2', 'Modelled
   Output 1', 'Modelled Output 2');
```

Fig. 2 example of how the result should look like:

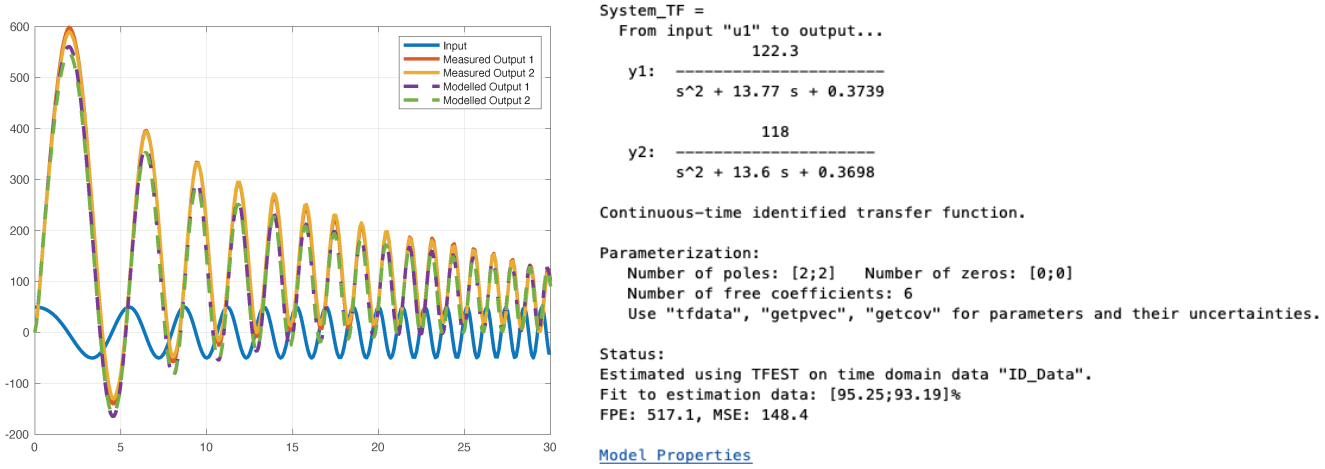

Figure 2: System Identification

4 Model Validation

To test the accuracy of the model, we control the system in simulation and in reality using a PID controller to do setpoint tracking. Build the simulink model depicted in Fig. 3. Here are some considerations:

- Make sure to select the right ports in the motor and encoder blocks.
- Select the parameters of the step block, PID blocks as depicted in Fig. 3.
- Select the parameters of the transfer function block according to your estimated system in the previous section.
- The scope measures the outputs from the real system and the simulated system.

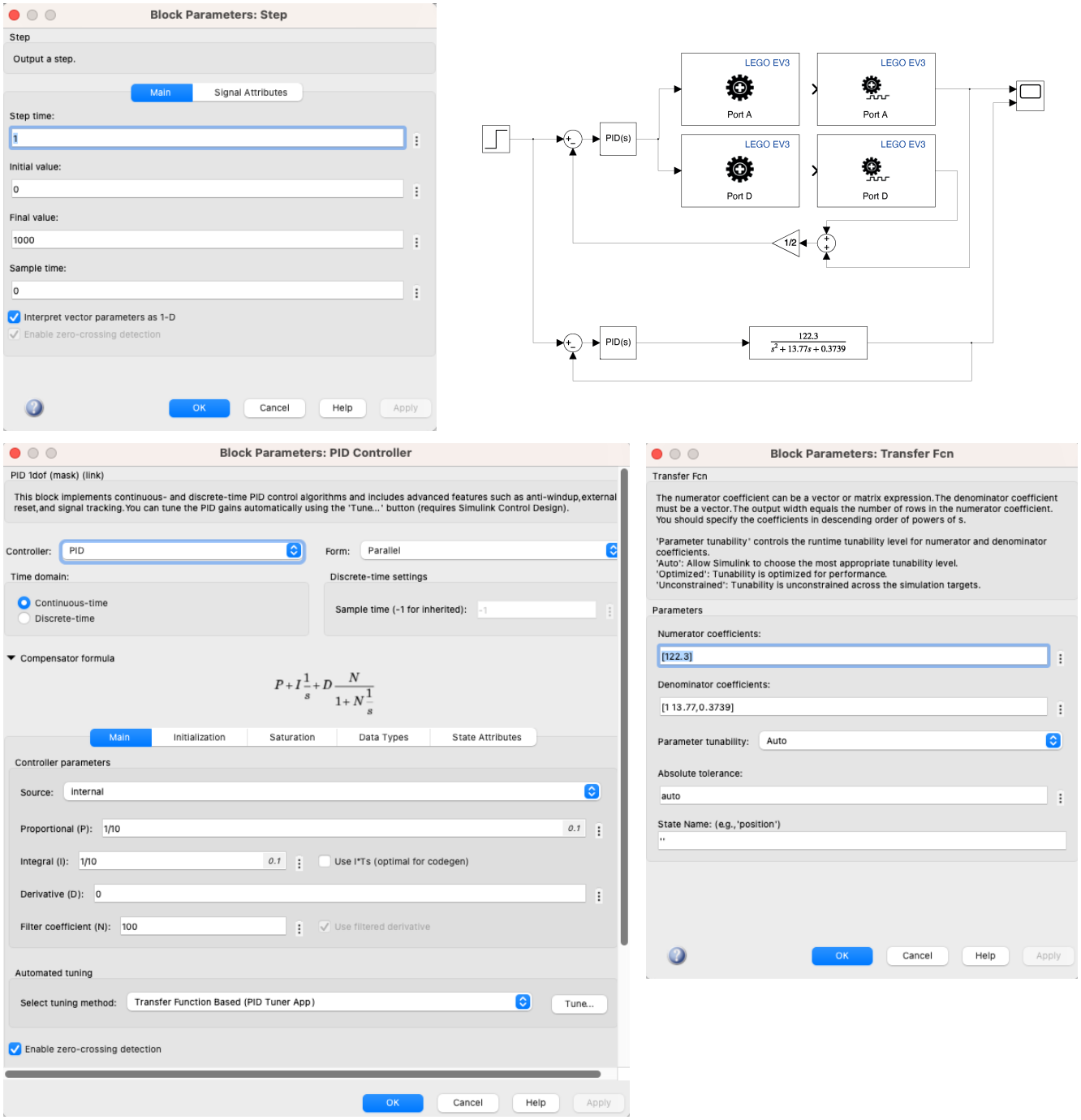

Figure 3: Model Validation

- For the real system, the feedback is taken here as the average between the two encoders. This technique is adopted here for simplicity only.
- Once the model is built, go to Modeling → Model Settings → Hardware Implementation → Hardware board → Lego Mindstorms EV3. In the same panel, go to Target Hardware resources → Host to Target Connection → Connection Type → Ethernet, and type in the correct IP address of the EV3 (e.g. 10.0.0.99). Finally click on Apply.
- Go to the Hardware tab, set the Stop Time to 30 and click on Monitor & Tune. The system should run and record the input/output signals in real-time for 30 seconds.

5 Simulating LQG Control

Having a good enough model, we now use it to simulate LQG control with setpoint tracking. Follow the following steps:

1. Write a MATLAB script (Setpoint_LQR.m) that
   - Obtains a state-space realization from the estimated transfer functions.
   - Computes the optimal LQR gain K and the feedthrough matrices F and N for setpoint tracking. Use $Q = 10$ and $R = 1$.
   - Build the simulink model depicted in Fig. 4. The model includes band-limited white noise to simulate actuator noise (Noise Power = 10^{-4}) and sensor noise (Noise Power = 10^{-2}). You can play with Noise Power values.
   - Run the MATLAB script first to get the gains, then run the simulink model.

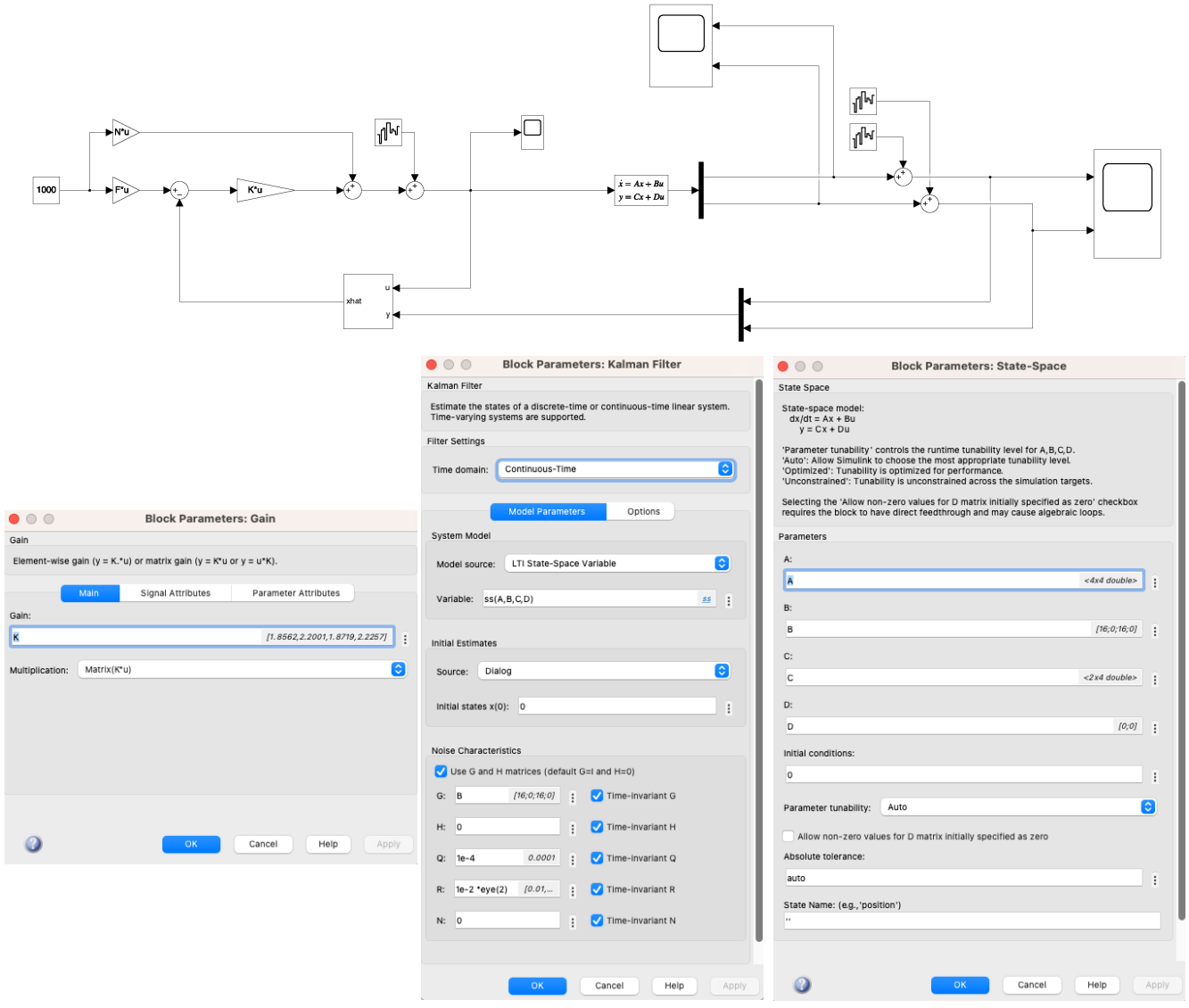

Figure 4: Simulation of LQG Control.

6 LQG Control of EV3

To implement LQG with Hardware, the kalman filter and the LQR gains need to be converted to discrete time. To do so, carry out the following steps:

1. Add to the MATLAB script Setpoint_LQR.m the following:

```
%% Discrete-Time System Matrices
System_d = c2d(System, 0.1);
A_d = System_d.A;
B_d = System_d.B;
C_d = System_d.C;
D_d = System_d.D;

%% LQR in Discrete Time
Q_d = 10;
R_d = 1;
M_d = [A_d - eye(size(A_d,1)), B_d; C_d, D_d];
M_d = M_d \ [zeros(4,1); 1; 1];
F_d = M_d(1:4,:);
N_d = M_d(5,:);
K_d = dlqr(A_d, B_d, Q_d, R_d, 0);
```

   Note the command "c2d", and slightly different structure of the Feedthrough gains.

2. Starting from the previous simulink model, build another model as the one Fig. 5. Note the convert block and the discrete-time Kalman filter. Also note that the LQR gains are now the discrete-time versions. The setpoint is 1000 degrees.

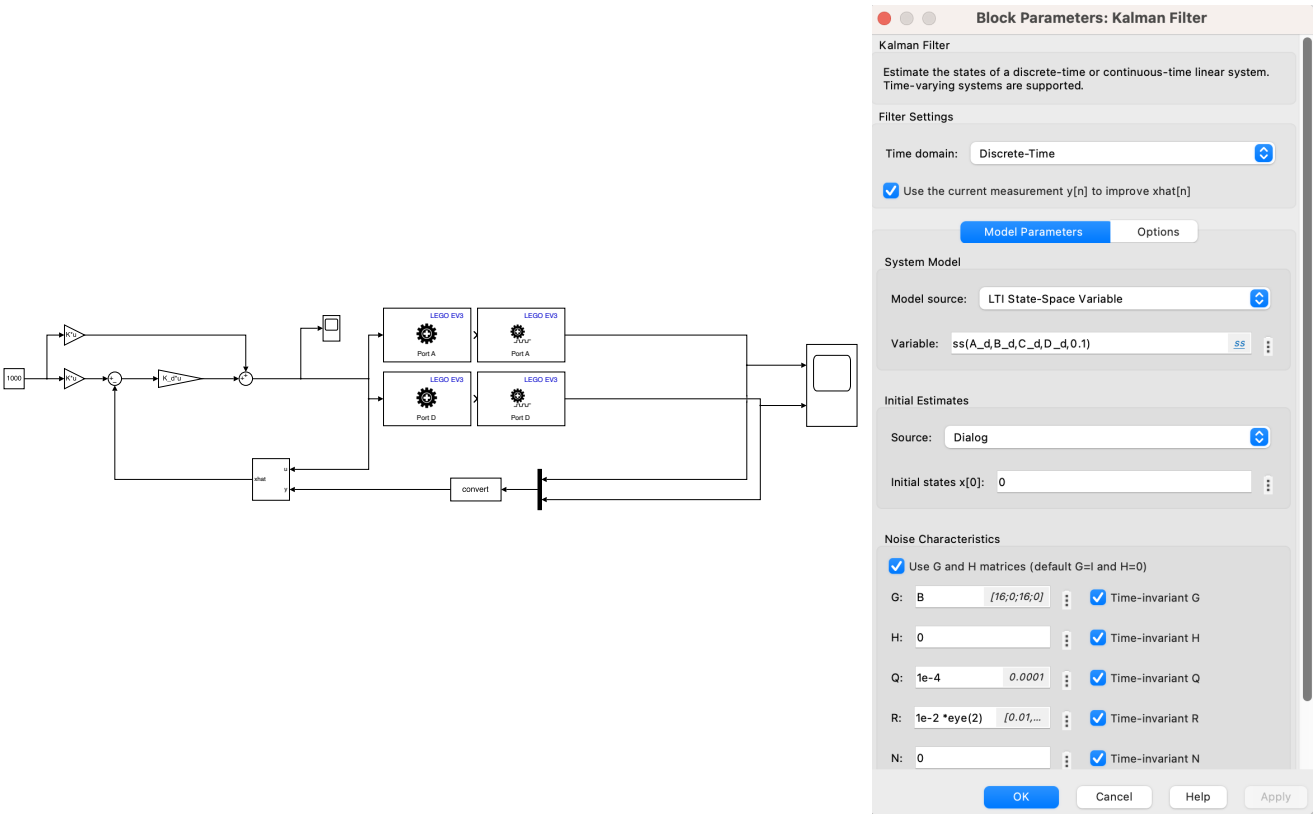

Figure 5: LQG Control.