Steering Control

SEED Lab **Alex Curtis**

Upload the stepResponse code to the Arduino before running.

Summary

This live script does the following:

1. Communicates with the Arduino over serial
2. Runs two experiments that show the characteristics of how the robot drives in a straight path and how it rotates
3. Validates the experimental step response data from the Arduino
4. Filters noise from the data
5. Analyzes the data to determine the transfer functions of the robot
6. Validates the transfer functions graphically
7. Creates a controller for each transfer function using the transfer functions
8. Outputs the Kp, Ki, and Kd constants for each controller in a format that can be easily implemented in the Arduino

Important Variables

$\dot{\rho}_d$ **(rho dot)**

The forward velocity of the robot (inches/sec)

ρ_d **(rho)**

The forward position of the robot (inches)

$\dot{\phi}$ **(phi dot)**

The angular velocity of the robot (radians/sec)

ϕ **(phi)**

The angular position of the robot (radians)

Important Equations

$$\overline{V}_a = V_{a,1} + V_{a,2}$$

$\overline{V}_a$ is the sum of the left and right motor voltage commands.

In the range [-400, 400] with 400 representing full forward speed

$$\Delta V_a = V_{a,1} - V_{a,2}$$

ΔV_a is the difference between the left and right motor voltage commands.

In the range [-400,400] with 400 representing spinning clockwise at full speed

Transfer functions to find:

$$G_\rho(s) = \frac{\rho(s)}{V_a(s)}$$

Forward position to the combined voltage commands

$$G_\phi(s) = \frac{\phi(s)}{\Delta V_a(s)}$$

Angular position to the difference between the voltage commands

Clear previous workspace data

```
clear
```

Automatically find the port the Arduino is connected to

```
allPorts = serialportlist;  % list of all available ports
pat = "/dev/cu.usbmodem";   % text to look for

% logical array of matching ports
matchingPorts = startsWith(allPorts, pat);

if any(matchingPorts)                    % If any matches were found,
    numMatches = sum(matchingPorts);     % Count the number of matches.
    if size(numMatches) > 1              % If multiple matching ports were found,
        error("Multiple ports found")    % throw an error.
    else                                 % If only one match was found,
        port = allPorts(matchingPorts); % that string is the arduino port.
    end
else                                     % If no ports were found,
    error("Port not found")              % throw an error.
end
```

Run Experiment 1 (forward velocity)

Create a serialport object to talk to the Arduino for the first experiment

```
ard = serialport(port, 115200);   % creates a device object (port, baud)
configureTerminator(ard, "CR/LF") % sets the terminator
```

Wait for the Arduino

When the Arduino is ready, it sends "Start!"

```
readline(ard)
```

```
ans =
"Start!"
```

Tell the Arduino to start the forward velocity experiment (#1)

```
disp('Starting forward velocity experiment in Arduino')
```

```
Starting forward velocity experiment in Arduino
```

```
writeline(ard,"1"); % sends the start command to the Arduino
```

Read data from Arduino

```
k = 0; % array index
% stringArray = zeros(1001,5); % preallocate

% Read the first line of output from the Arduino
newData = readline(ard);

% Read new data until the Arduino signals it's done
while (~strncmp(newData,'Finished1',9))
    k = k + 1; % k++

    % save the line of text to stringArray(k)
    stringArray(k,:) = split(newData)';

    % Read a new line of data from Arduino
    newData = readline(ard);
end
clear newData ard; % Disconect from the Arduino

% change string data to cell array using tab delimiter
dataArray = str2double(stringArray);

% Make time start at 0 and convert from ms to s
dataArray(:,1) = (dataArray(:,1)-1000)/1000;

% Save experiment 1 data to the exp1 data table
exp1 = array2table(dataArray,"VariableNames",{'time', 'posL', 'posR', 'fVel', 'rVel'})
save steerData.mat exp1
clear dataArray stringArray newData
```

Run Experiment 2 (angular velocity)

Create a serialport object to talk to the Arduino for the second experiment

```
ard = serialport(port, 115200);   % creates a device object (port, baud)
configureTerminator(ard, "CR/LF") % sets the terminator
```

Wait for the Arduino

When the Arduino is ready, it sends "Start!"

```
readline(ard)
```

```
ans =
"Start!"
```

Tell the Arduino to start the rotational velocity experiment (#2)

```
disp('Starting rotational velocity experiment in Arduino')
```

```
Starting rotational velocity experiment in Arduino
```

```
writeline(ard,"2") % sends the start command to the Arduino
```

Read data from Arduino

```
k2 = 0; % array index

% Read the first line of output from the Arduino
newData = readline(ard);

% Read new data until the Arduino signals it's done
while (~strncmp(newData,'Finished2',9))
    k2 = k2 + 1; % k++

    % save the line of text to stringArray(k)
    stringArray(k2,:) = split(newData)';

    % Read a new line of data from Arduino
    newData = readline(ard);
end

clear ard; % Disconect from the Arduino

% change string data to cell array using tab delimiter
dataArray = str2double(stringArray);

% Make time start at 0 and convert from ms to s
dataArray(:,1) = (dataArray(:,1)-1000)/1000;

% Save experiment 2 data to the exp2 data table
exp2 = array2table(dataArray,"VariableNames",{'time', 'posL', 'posR', 'fVel', 'rVel'})
```

Verify the velocity data from the Arduino

The values sent from the Arduino aren't as consistent as what Matlab can do.

```
WHEEL_RADIUS = 2.95;              % Radius of wheel in inches
WHEELBASE = 13.974;
rho_dot1(k,1) = 0;
phi_dot1(k,1) = 0;
rho_dot2(k2,1) = 0;
phi_dot2(k2,1) = 0;
CONVERSION = (WHEEL_RADIUS*2.0*pi)/3200;
```

```
for i = 2:k
    rho_dot1(i,1) = CONVERSION*(exp1.posR(i) - exp1.posR(i-1) + exp1.posL(i) - exp1.pos
    phi_dot1(i,1) = CONVERSION*(exp1.posL(i) - exp1.posL(i-1) - exp1.posR(i) + exp1.pos
end

for i = 2:k2
    rho_dot2(i,1) = CONVERSION*(exp2.posR(i) - exp2.posR(i-1) + exp2.posL(i) - exp2.pos
    phi_dot2(i,1) = CONVERSION*(exp2.posL(i) - exp2.posL(i-1) - exp2.posR(i) + exp2.pos
end

exp1.newRho = rho_dot1;
exp1.newPhi = phi_dot1
```

exp1 = 101×7 table

	time	posL	posR	fVel	rVel	newRho	newPhi
1	0	0	0	0	0	0	0
2	0.0100	3	3	1.8817	0	1.7377	0
3	0.0200	12	11	5.3314	0.0456	4.9235	0.0415
4	0.0300	21	22	6.2723	-0.0912	5.7923	-0.0829
5	0.0400	34	37	8.7812	-0.0912	8.1092	-0.0829
6	0.0500	50	53	10.0356	0	9.2677	0
7	0.0600	69	73	12.2309	-0.0456	11.2950	-0.0415
8	0.0700	88	92	11.9173	0	11.0054	0
9	0.0800	110	115	14.1126	-0.0456	13.0327	-0.0415
10	0.0900	134	140	15.3671	-0.0456	14.1912	-0.0415
11	0.1000	158	167	15.9943	-0.1368	14.7704	-0.1244
12	0.1100	182	191	15.0535	0	13.9015	0
13	0.1200	209	220	17.5624	-0.0912	16.2185	-0.0829
14	0.1300	237	249	17.8760	-0.0456	16.5081	-0.0415
15	0.1400	266	278	18.1896	0	16.7977	0
16	0.1500	294	309	18.5032	-0.1368	17.0873	-0.1244
17	0.1600	321	337	19.4049	-0.0513	15.9289	-0.0415
18	0.1700	351	369	19.4441	-0.0912	17.9562	-0.0829
19	0.1800	382	401	19.7577	-0.0456	18.2458	-0.0415
20	0.1900	413	433	19.7577	-0.0456	18.2458	-0.0415
21	0.2000	441	463	18.1896	-0.0912	16.7977	-0.0829
22	0.2100	473	496	20.3849	-0.0456	18.8250	-0.0415
23	0.2200	505	529	20.3849	-0.0456	18.8250	-0.0415
24	0.2300	537	563	20.6985	-0.0912	19.1146	-0.0829
25	0.2400	566	593	18.5032	-0.0456	17.0873	-0.0415

	time	posL	posR	fVel	rVel	newRho	newPhi
26	0.2500	599	627	21.0121	-0.0456	19.4042	-0.0415
27	0.2600	631	661	20.6985	-0.0912	19.1146	-0.0829
28	0.2700	664	695	21.0121	-0.0456	19.4042	-0.0415
29	0.2800	694	726	19.1305	-0.0456	17.6665	-0.0415
30	0.2900	727	760	21.0121	-0.0456	19.4042	-0.0415
31	0.3000	760	795	21.3257	-0.0912	19.6939	-0.0829
32	0.3100	794	829	21.3257	0	19.6939	0
33	0.3200	827	864	21.3257	-0.0912	19.6939	-0.0829
34	0.3300	857	895	19.1305	-0.0456	17.6665	-0.0415
35	0.3400	890	929	21.0121	-0.0456	19.4042	-0.0415
36	0.3500	924	964	21.6393	-0.0456	19.9835	-0.0415
37	0.3600	958	999	21.6393	-0.0456	19.9835	-0.0415
38	0.3700	988	1030	19.1305	-0.0456	17.6665	-0.0415
39	0.3800	1022	1065	21.6393	-0.0456	19.9835	-0.0415
40	0.3900	1055	1099	21.0121	-0.0456	19.4042	-0.0415
41	0.4000	1089	1134	21.6394	-0.0456	19.9835	-0.0415
42	0.4100	1119	1165	19.1304	-0.0456	17.6665	-0.0415
43	0.4200	1153	1200	21.6393	-0.0456	19.9835	-0.0415
44	0.4300	1187	1235	21.6394	-0.0456	19.9835	-0.0415
45	0.4400	1221	1270	21.6393	-0.0456	19.9835	-0.0415
46	0.4510	1254	1304	18.9109	-0.0411	17.6402	-0.0377
47	0.4600	1285	1336	22.2274	-0.0513	20.2731	-0.0461
48	0.4700	1318	1371	21.3257	-0.0912	19.6939	-0.0829
49	0.4800	1352	1405	21.3257	0	19.6939	0
50	0.4900	1386	1440	21.6394	-0.0456	19.9835	-0.0415
51	0.5000	1417	1471	19.4440	0	17.9562	0
52	0.5100	1450	1506	21.3258	-0.0912	19.6939	-0.0829
53	0.5200	1484	1541	21.6393	-0.0456	19.9835	-0.0415
54	0.5300	1518	1575	21.3257	0	19.6939	0
55	0.5400	1549	1607	19.7577	-0.0456	18.2458	-0.0415
56	0.5500	1582	1641	21.0121	-0.0456	19.4042	-0.0415
57	0.5600	1617	1676	21.9530	0	20.2731	0
58	0.5700	1650	1711	21.3257	-0.0912	19.6939	-0.0829

	time	posL	posR	fVel	rVel	newRho	newPhi
59	0.5800	1681	1742	19.4441	0	17.9562	0
60	0.5900	1714	1777	21.3257	-0.0912	19.6939	-0.0829
61	0.6000	1749	1812	21.9530	0	20.2731	0
62	0.6100	1782	1846	21.0122	-0.0456	19.4042	-0.0415
63	0.6200	1816	1881	21.6393	-0.0456	19.9835	-0.0415
64	0.6300	1846	1912	21.5217	-0.0513	17.6665	-0.0415
65	0.6400	1880	1947	21.6393	-0.0456	19.9835	-0.0415
66	0.6500	1914	1982	21.6393	-0.0456	19.9835	-0.0415
67	0.6600	1948	2017	21.6394	-0.0456	19.9835	-0.0415
68	0.6700	1978	2049	19.4441	-0.0912	17.9562	-0.0829
69	0.6800	2012	2084	21.6393	-0.0456	19.9835	-0.0415
70	0.6900	2046	2118	21.3257	0	19.6939	0
71	0.7000	2080	2154	21.9529	-0.0912	20.2731	-0.0829
72	0.7100	2111	2185	19.4441	0	17.9562	0
73	0.7200	2144	2220	21.3257	-0.0912	19.6939	-0.0829
74	0.7300	2178	2255	21.6393	-0.0456	19.9835	-0.0415
75	0.7400	2212	2290	21.6394	-0.0456	19.9835	-0.0415
76	0.7500	2243	2322	19.7577	-0.0456	18.2458	-0.0415
77	0.7600	2277	2357	21.6393	-0.0456	19.9835	-0.0415
78	0.7700	2311	2392	21.6394	-0.0456	19.9835	-0.0415
79	0.7800	2345	2427	21.6394	-0.0456	19.9835	-0.0415
80	0.7900	2379	2462	21.6394	-0.0456	19.9835	-0.0415
81	0.8000	2409	2493	21.5217	-0.0513	17.6665	-0.0415
82	0.8100	2444	2528	21.9529	0	20.2731	0
83	0.8200	2477	2563	21.3257	-0.0912	19.6939	-0.0829
84	0.8300	2512	2598	21.9530	0	20.2731	0
85	0.8400	2542	2630	19.4441	-0.0912	17.9562	-0.0829
86	0.8500	2577	2665	21.9529	0	20.2731	0
87	0.8600	2610	2700	21.3257	-0.0912	19.6939	-0.0829
88	0.8700	2645	2736	22.2666	-0.0456	20.5627	-0.0415
89	0.8800	2676	2767	19.4440	0	17.9562	0
90	0.8900	2710	2803	21.9530	-0.0912	20.2731	-0.0829
91	0.9000	2744	2838	21.6393	-0.0456	19.9835	-0.0415

	time	posL	posR	fVel	rVel	newRho	newPhi
92	0.9100	2779	2873	21.9530	0	20.2731	0
93	0.9200	2809	2905	19.4440	-0.0912	17.9562	-0.0829
94	0.9300	2844	2940	21.9530	0	20.2731	0
95	0.9400	2878	2976	21.9529	-0.0912	20.2731	-0.0829
96	0.9500	2912	3011	21.6394	-0.0456	19.9835	-0.0415
97	0.9600	2946	3046	21.6393	-0.0456	19.9835	-0.0415
98	0.9700	2977	3078	19.7577	-0.0456	18.2458	-0.0415
99	0.9800	3011	3113	21.6394	-0.0456	19.9835	-0.0415
100	0.9900	3045	3149	21.9529	-0.0912	20.2731	-0.0829

⋮

```
exp2.newRho = rho_dot2;
exp2.newPhi = phi_dot2
```

exp2 = 101×7 table

	time	posL	posR	fVel	rVel	newRho	newPhi
1	0	0	0	0	0	0	0
2	0.0100	3	-2	0.3136	0.2281	0.2896	0.2073
3	0.0200	12	-10	0.3136	0.7755	0.2896	0.7047
4	0.0300	23	-21	0	1.0036	0	0.9119
5	0.0400	36	-34	0	1.1860	0	1.0777
6	0.0500	52	-51	-0.3136	1.5053	-0.2896	1.3679
7	0.0600	71	-71	-0.3136	1.7790	-0.2896	1.6166
8	0.0700	93	-93	0	2.0071	0	1.8238
9	0.0800	113	-114	-0.3136	1.8703	-0.2896	1.6995
10	0.0900	137	-139	-0.3136	2.2352	-0.2896	2.0311
11	0.1000	163	-165	0	2.3721	0	2.1554
12	0.1100	189	-193	-0.6272	2.4633	-0.5792	2.2383
13	0.1200	217	-221	0	2.5545	0	2.3212
14	0.1300	242	-248	-0.7056	2.6686	-0.5792	2.1554
15	0.1400	271	-278	-0.3136	2.6914	-0.2896	2.4456
16	0.1500	300	-309	-0.6272	2.7370	-0.5792	2.4870
17	0.1600	330	-340	-0.3136	2.7826	-0.2896	2.5285
18	0.1700	358	-368	0	2.5545	0	2.3212
19	0.1800	388	-400	-0.6272	2.8282	-0.5792	2.5699

	time	posL	posR	fVel	rVel	newRho	newPhi
20	0.1900	420	-433	-0.3136	2.9651	-0.2896	2.6943
21	0.2000	451	-466	-0.6272	2.9195	-0.5792	2.6528
22	0.2100	479	-496	-0.6272	2.6458	-0.5792	2.4041
23	0.2200	511	-529	-0.3136	2.9651	-0.2896	2.6943
24	0.2300	544	-563	-0.3136	3.0563	-0.2896	2.7772
25	0.2400	576	-597	-0.6272	3.0107	-0.5792	2.7357
26	0.2500	606	-627	0	2.7370	0	2.4870
27	0.2600	638	-661	-0.6272	3.0107	-0.5792	2.7357
28	0.2700	671	-696	-0.6272	3.1019	-0.5792	2.8186
29	0.2800	704	-730	-0.3136	3.0563	-0.2896	2.7772
30	0.2900	738	-765	-0.3136	3.1475	-0.2896	2.8601
31	0.3000	767	-796	-0.6272	2.7370	-0.5792	2.4870
32	0.3100	801	-831	-0.3136	3.1475	-0.2896	2.8601
33	0.3200	834	-866	-0.6272	3.1019	-0.5792	2.8186
34	0.3300	868	-901	-0.3136	3.1475	-0.2896	2.8601
35	0.3400	897	-933	-0.9408	2.7826	-0.8688	2.5285
36	0.3500	931	-968	-0.3136	3.1475	-0.2896	2.8601
37	0.3600	965	-1003	-0.3136	3.1475	-0.2896	2.8601
38	0.3700	999	-1039	-0.6272	3.1932	-0.5792	2.9015
39	0.3800	1028	-1070	-0.6272	2.7370	-0.5792	2.4870
40	0.3900	1063	-1106	-0.3136	3.2388	-0.2896	2.9430
41	0.4000	1096	-1141	-0.6272	3.1019	-0.5792	2.8186
42	0.4100	1130	-1177	-0.6272	3.1932	-0.5792	2.9015
43	0.4210	1164	-1212	-0.2822	2.8328	-0.2633	2.6001
44	0.4300	1194	-1244	-0.7056	3.1818	-0.6436	2.8555
45	0.4400	1228	-1280	-0.6273	3.1932	-0.5792	2.9015
46	0.4500	1262	-1315	-0.3136	3.1475	-0.2896	2.8601
47	0.4600	1295	-1351	-0.9408	3.1475	-0.8688	2.8601
48	0.4700	1326	-1383	-0.3136	2.8738	-0.2896	2.6114
49	0.4800	1360	-1418	-0.3136	3.1475	-0.2896	2.8601
50	0.4900	1394	-1454	-0.6272	3.1932	-0.5792	2.9015
51	0.5000	1428	-1490	-0.6272	3.1932	-0.5792	2.9015
52	0.5100	1458	-1522	-0.6272	2.8282	-0.5792	2.5699

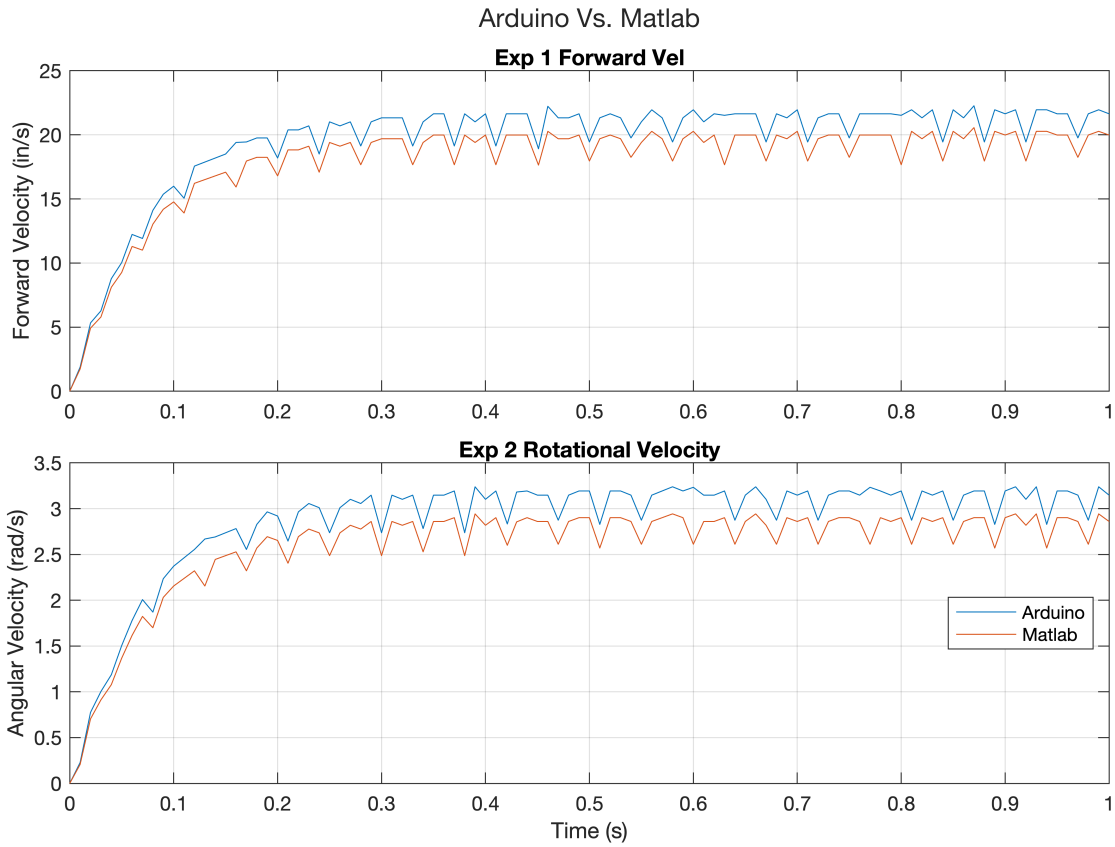
	time	posL	posR	fVel	rVel	newRho	newPhi
53	0.5200	1492	-1558	-0.6272	3.1932	-0.5792	2.9015
54	0.5300	1526	-1594	-0.6272	3.1932	-0.5792	2.9015
55	0.5400	1560	-1629	-0.3137	3.1475	-0.2896	2.8601
56	0.5500	1590	-1662	-0.9408	2.8738	-0.8688	2.6114
57	0.5600	1624	-1697	-0.3136	3.1475	-0.2896	2.8601
58	0.5700	1658	-1733	-0.6272	3.1932	-0.5792	2.9015
59	0.5800	1693	-1769	-0.3136	3.2388	-0.2896	2.9430
60	0.5900	1727	-1805	-0.6273	3.1932	-0.5792	2.9015
61	0.6000	1758	-1837	-0.3528	3.2331	-0.2896	2.6114
62	0.6100	1791	-1873	-0.9409	3.1475	-0.8688	2.8601
63	0.6200	1825	-1908	-0.3136	3.1475	-0.2896	2.8601
64	0.6300	1859	-1944	-0.6272	3.1932	-0.5792	2.9015
65	0.6400	1890	-1976	-0.3136	2.8738	-0.2896	2.6114
66	0.6500	1923	-2012	-0.9408	3.1475	-0.8688	2.8601
67	0.6600	1958	-2048	-0.3136	3.2388	-0.2896	2.9430
68	0.6700	1991	-2083	-0.6272	3.1019	-0.5792	2.8186
69	0.6800	2022	-2115	-0.3136	2.8738	-0.2896	2.6114
70	0.6900	2056	-2151	-0.6272	3.1932	-0.5792	2.9015
71	0.7000	2090	-2186	-0.3136	3.1475	-0.2896	2.8601
72	0.7100	2124	-2222	-0.6272	3.1932	-0.5792	2.9015
73	0.7200	2155	-2254	-0.3136	2.8738	-0.2896	2.6114
74	0.7300	2189	-2289	-0.3136	3.1475	-0.2896	2.8601
75	0.7400	2223	-2325	-0.6273	3.1932	-0.5792	2.9015
76	0.7500	2257	-2361	-0.6273	3.1932	-0.5792	2.9015
77	0.7600	2291	-2396	-0.3136	3.1475	-0.2896	2.8601
78	0.7700	2322	-2428	-0.3528	3.2331	-0.2896	2.6114
79	0.7800	2356	-2464	-0.6272	3.1932	-0.5792	2.9015
80	0.7900	2390	-2499	-0.3137	3.1475	-0.2896	2.8601
81	0.8000	2424	-2535	-0.6273	3.1932	-0.5792	2.9015
82	0.8100	2455	-2567	-0.3135	2.8738	-0.2896	2.6114
83	0.8200	2489	-2603	-0.6273	3.1932	-0.5792	2.9015
84	0.8300	2523	-2638	-0.3136	3.1476	-0.2896	2.8601
85	0.8400	2557	-2674	-0.6273	3.1932	-0.5792	2.9015

	time	posL	posR	fVel	rVel	newRho	newPhi
86	0.8500	2588	-2706	-0.3136	2.8738	-0.2896	2.6114
87	0.8600	2622	-2741	-0.3136	3.1475	-0.2896	2.8601
88	0.8700	2656	-2777	-0.6273	3.1932	-0.5792	2.9015
89	0.8800	2690	-2813	-0.6272	3.1932	-0.5792	2.9015
90	0.8900	2721	-2844	0	2.8282	0	2.5699
91	0.9000	2755	-2880	-0.6273	3.1932	-0.5792	2.9015
92	0.9100	2790	-2916	-0.3136	3.2388	-0.2896	2.9430
93	0.9200	2823	-2951	-0.6272	3.1019	-0.5792	2.8186
94	0.9300	2858	-2987	-0.3136	3.2388	-0.2896	2.9430
95	0.9400	2888	-3019	-0.6272	2.8282	-0.5792	2.5699
96	0.9500	2923	-3054	0	3.1932	0	2.9015
97	0.9600	2957	-3090	-0.6273	3.1932	-0.5792	2.9015
98	0.9700	2991	-3125	-0.3136	3.1475	-0.2896	2.8601
99	0.9800	3022	-3157	-0.3136	2.8738	-0.2896	2.6114
100	0.9900	3057	-3193	-0.3136	3.2388	-0.2896	2.9430

⋮

Compare the data Matlab calculated with what the Arduino sent

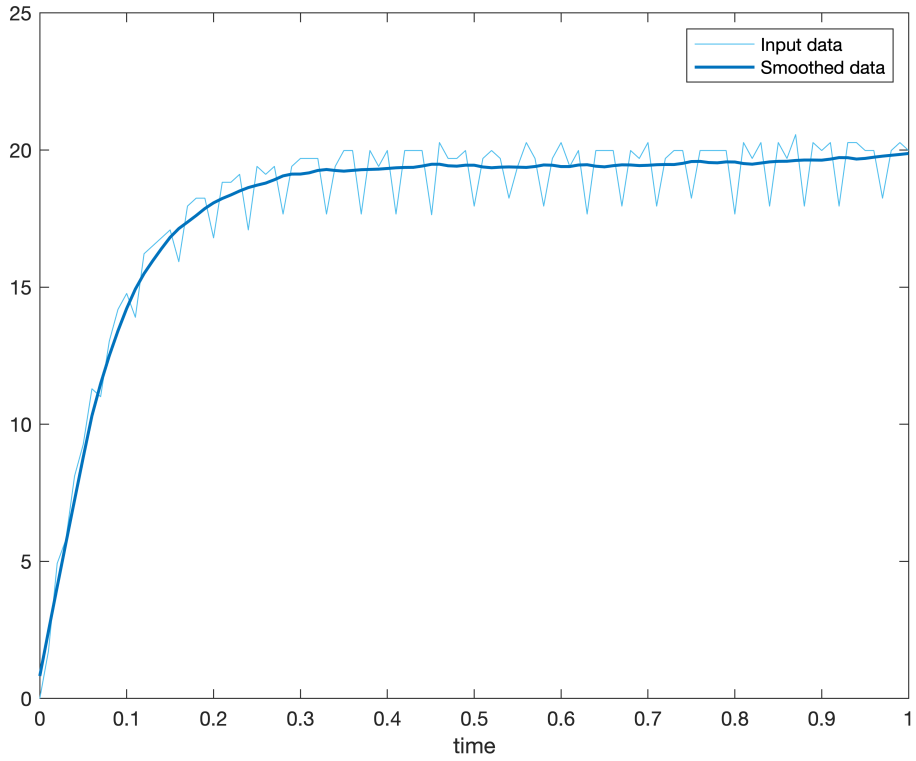
```
t = tiledlayout(2,1,'TileSpacing','Compact','Padding','tight');
title(t,'Arduino Vs. Matlab')

nexttile;
plot(exp1.time,exp1.fVel,exp1.time,exp1.newRho);
title('Exp 1 Forward Vel');
ylabel('Forward Velocity (in/s)');
grid on;

nexttile;
plot(exp2.time, exp2.rVel,exp2.time,exp2.newPhi);
title('Exp 2 Rotational Velocity');
legend('Arduino','Matlab','Location','east');
ylabel('Angular Velocity (rad/s)');
grid on

xlabel('Time (s)')
```



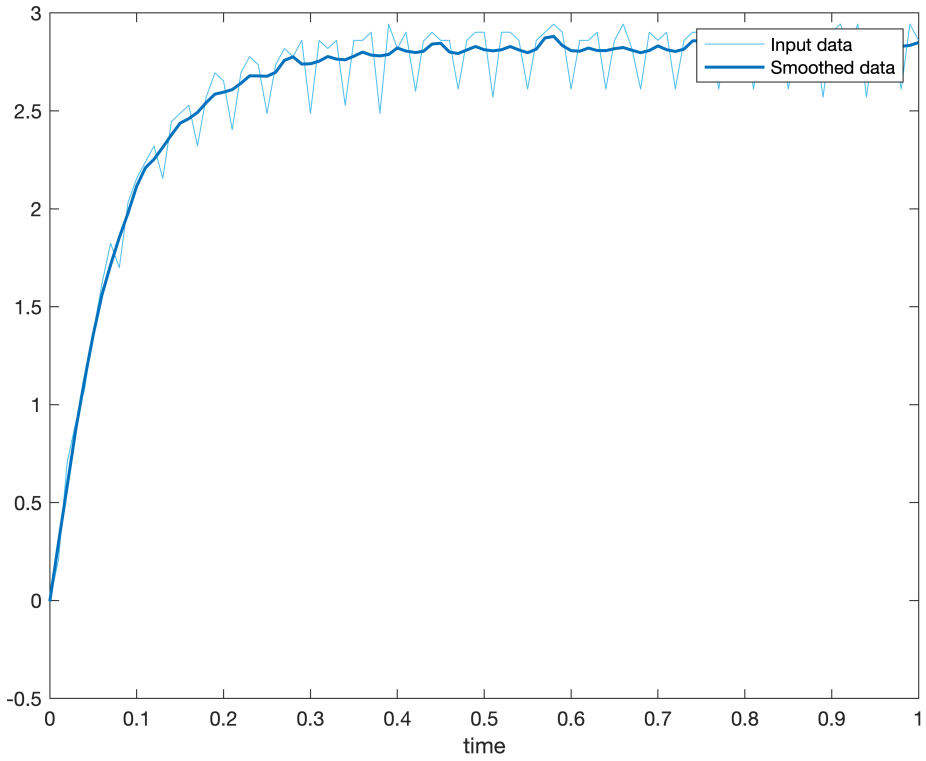
Clean and Scale the velocity data

Smoothing the velocity data gets rid of any noise that might mess up the resulting transfer functions.

```
new1 = exp1;
new2 = exp2;
% Smooth input data
smoothedRho = smoothdata(exp1.newRho,"lowess","SmoothingFactor",0.4,...
    "SamplePoints",exp1.time);

% Display results
clf
plot(exp1.time,exp1.newRho,"Color",[77 190 238]/255,...
    "DisplayName","Input data")
hold on
plot(exp1.time,smoothedRho,"Color",[0 114 189]/255,"LineWidth",1.5,...
    "DisplayName","Smoothed data")
hold off
legend
xlabel("time")
```




```
new1.newRho = smoothedRho;
% Smooth input data
smoothedPhi = smoothdata(exp2.newPhi,"lowess","SmoothingFactor",0.25,...
    "SamplePoints",exp2.time);

% Display results
clf
plot(exp2.time,exp2.newPhi,"Color",[77 190 238]/255,...
    "DisplayName","Input data")
hold on
plot(exp2.time,smoothedPhi,"Color",[0 114 189]/255,"LineWidth",1.5,...
    "DisplayName","Smoothed data")
hold off
legend
xlabel("time")
```




```
new2.newPhi = smoothedPhi;
```

max motor.setM1Speed() command = 400

Since the motor command sum is a number out of 400, I need to divide the forward velocity by 400 to get a unit step response.

The maximum motor command difference is 400, corresponding to -400 and 400 on each motor. The rotational velocity needs to be divided by 400

```
new1.newRho = new1.newRho/400; new2.newPhi = new2.newPhi/400;
```

Determine K and sigma for each experiment

Find the steady state velocity of the motor (K)

```
K_rho = new1.newRho(new1.time > 0.4); K_phi = new2.newPhi(new2.time > 0.4);
K_rho = mean(K_rho), K_phi = mean(K_phi)
```

```
K_rho = 0.0488
K_phi = 0.0071
```

Find 0.64 * K and the time constant to find the values of sigma.

```
% Find 0.64K
K_rho64 = 0.64*K_rho;
K_phi64 = 0.64*K_phi;

% Find the closest match to 0.64K in each set of data
```

```
[~,idxRho] = min( abs(new1.newRho-K_rho64) );
magAt64K_rho = new1.newRho(idxRho)
```

```
magAt64K_rho = 0.0313
```

```
[~,idxPhi]=min(abs(new2.newPhi-K_phi64));
magAt64K_phi=new2.newPhi(idxPhi)
```

```
magAt64K_phi = 0.0046
```

```
% 0.64K corresponds to the location of the time constant (1/sigma)
TC_rho = new1.time(new1.newRho == magAt64K_rho);
TC_phi = new2.time(new2.newPhi == magAt64K_phi);

% We found sigma!
sigmaRho = 1/TC_rho
```

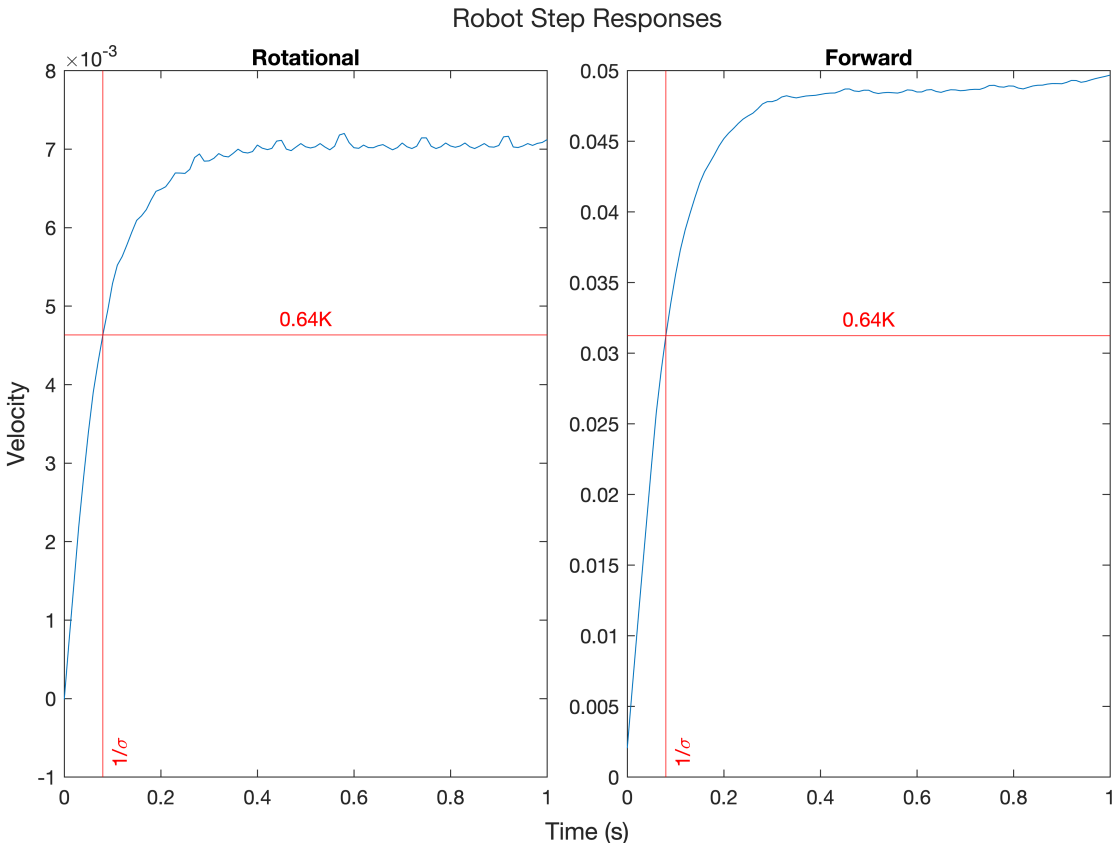
```
sigmaRho = 12.5000
```

```
sigmaPhi = 1/TC_phi
```

```
sigmaPhi = 12.5000
```

Plot the processed data (motor unit step response)

```
t2 = tiledlayout(1,2,"TileSpacing","compact",'Padding','tight');
xlabel(t2, 'Time (s)')
ylabel(t2, 'Velocity')
title(t2, 'Robot Step Responses')

nexttile
plot(new2.time,new2.newPhi)
title('Rotational')
yline(magAt64K_phi,'Label','0.64K','Color','r',LabelHorizontalAlignment='center')
xline(TC_phi,'Label','1/\sigma','Color','r','LabelVerticalAlignment','bottom')

nexttile
plot(new1.time,new1.newRho)
title('Forward')
yline(magAt64K_rho,'Label','0.64K','Color','r',LabelHorizontalAlignment='center')
xline(TC_rho,'Label','1/\sigma','Color','r','LabelVerticalAlignment','bottom')
```



Create the velocity and position transfer functions

With the sigma and K values for each experiment, I can create first order transfer functions that represent the response of the robot

```
s = tf('s');

% Find the velocity transfer functions
rho_velTF = K_rho*(sigmaRho)/(s+sigmaRho);
rho_velTF.InputName = "Command Sum";
rho_velTF.OutputName = "Forward Velocity";
rho_velTF.OutputUnit = "inches/s"
```

```
rho_velTF =

  From input "Command Sum" to output "Forward Velocity":
    0.61
  --------
  s + 12.5

Continuous-time transfer function.
```

```
phi_velTF = K_phi*(sigmaPhi)/(s+sigmaPhi);
phi_velTF.InputName = "Command Difference";
phi_velTF.OutputName = "Rotational Velocity";
phi_velTF.OutputUnit = "rad/s"
```

```
phi_velTF =

  From input "Command Difference" to output "Rotational Velocity":
  0.08815
  --------
  s + 12.5

Continuous-time transfer function.
```

```
% Position is the integral of velocity, and 1/s represents an integral.
rho_posTF = rho_velTF*(1/s);
rho_posTF.InputName = "Command Sum";
rho_posTF.OutputName = "Forward Position";
rho_posTF.OutputUnit = "inches"
```

```
rho_posTF =

  From input "Command Sum" to output "Forward Position":
      0.61
  ------------
  s^2 + 12.5 s

Continuous-time transfer function.
```

```
phi_posTF = phi_velTF*(1/s);
phi_posTF.InputName = "Command Difference";
phi_posTF.OutputName = "Rotational Position";
phi_posTF.OutputUnit = "radians"
```

```
phi_posTF =

  From input "Command Difference" to output "Rotational Position":
    0.08815
  ------------
  s^2 + 12.5 s

Continuous-time transfer function.
```

Turn the generated transfer functions into coefficient arrays I can use in Simulink

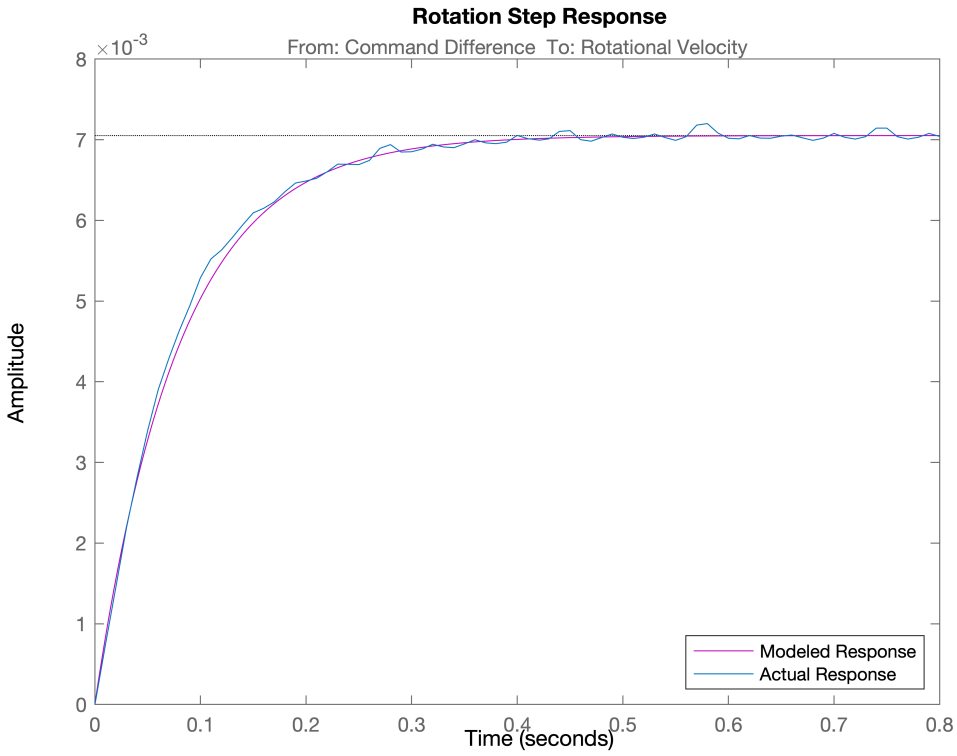
```
posNum_rho = rho_posTF.Numerator;       % Find the numerator coefficients
posNum_rho = posNum_rho{1};             % Convert the numerator to a the right form
posDen_rho = rho_posTF.Denominator;     % Find the denominator coefficients
posDen_rho = posDen_rho{1};             % Convert the denominator to a the right form

velNum_rho = rho_velTF.Numerator;       % Find the numerator coefficients
velNum_rho = velNum_rho{1};             % Convert the numerator to a the right form
velDen_rho = rho_velTF.Denominator;     % Find the denominator coefficients
velDen_rho = velDen_rho{1};             % Convert the denominator to a the right form

posNum_phi = phi_posTF.Numerator;       % Find the numerator coefficients
posNum_phi = posNum_phi{1};             % Convert the numerator to a the right form
posDen_phi = phi_posTF.Denominator;     % Find the denominator coefficients
posDen_phi = posDen_phi{1};             % Convert the denominator to a the right form

velNum_phi = phi_velTF.Numerator;       % Find the numerator coefficients
```

```
velNum_phi = velNum_phi{1};              % Convert the numerator to a the right form
velDen_phi = phi_velTF.Denominator;      % Find the denominator coefficients
velDen_phi = velDen_phi{1};              % Convert the denominator to a the right form
```

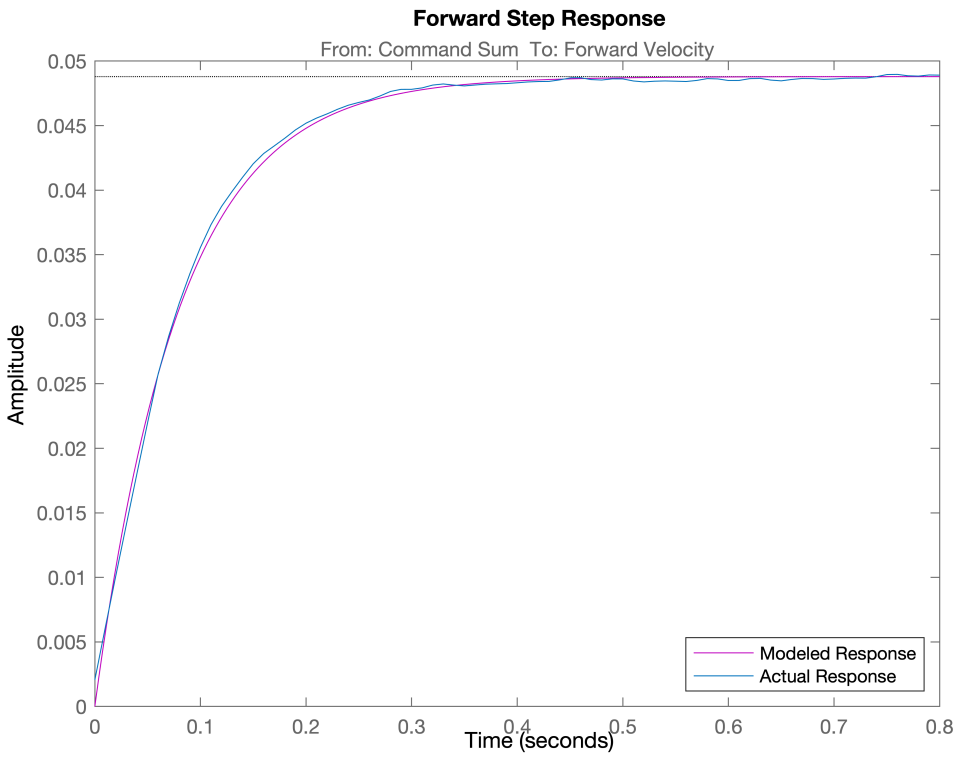
Compare the velocity transfer functions with the actual velocity of the robot

The closer they match, the more accurately the transfer functions represent the robot

```
figure
stepplot(phi_velTF,'m')
hold on
plot(new2.time,new2.newPhi)
title("Rotation Step Response")
legend('Modeled Response','Actual Response','Location','southeast')
hold off
```




```
stepplot(rho_velTF,'m')
hold on
plot(new1.time,new1.newRho)
title("Forward Step Response")
legend('Modeled Response','Actual Response','Location','southeast')
hold off
```

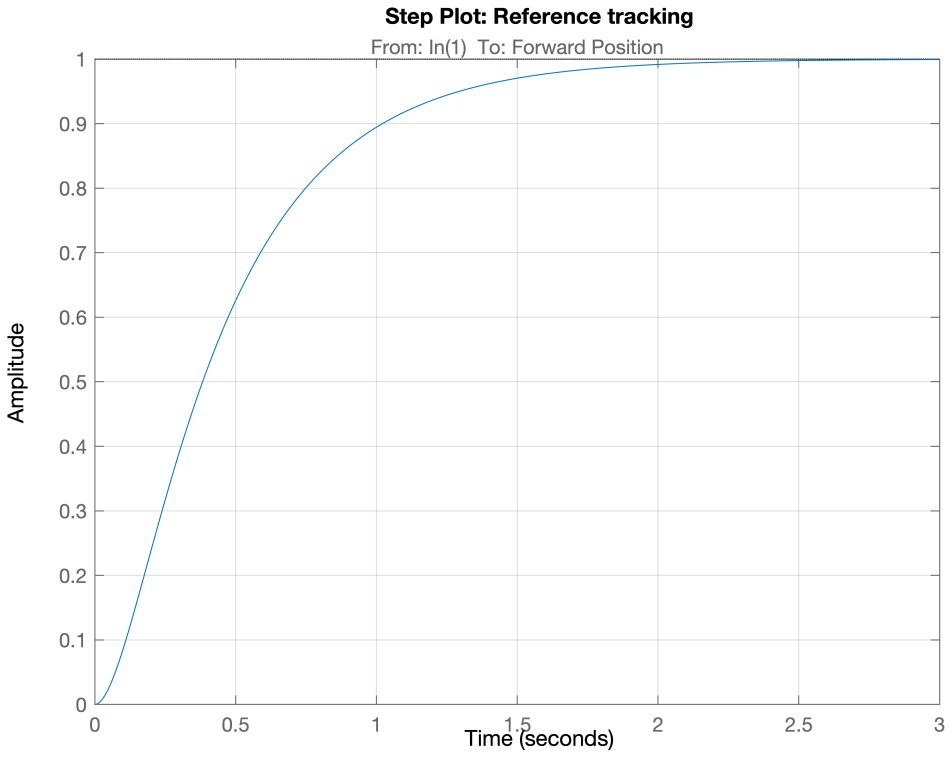



```
figure
```

Design controllers to regulate position

Make a controller for forward position (rho)

```
% Convert Response Time to Bandwidth
% Bandwidth is equivalent to 2 divided by the Response Time
wc2 = 2/1;

% PID tuning algorithm for linear plant model
[C_rho,pidInfo] = pidtune(posTF_rho,'P',wc2);

% Clear Temporary Variables
clear wc2

% Get desired loop response
Response = getPIDLoopResponse(C_rho,posTF_rho,'closed-loop');

% Plot the result
stepplot(Response)
title('Step Plot: Reference tracking')
grid on
```




```
% Display system response characteristics
disp(stepinfo(Response))
```

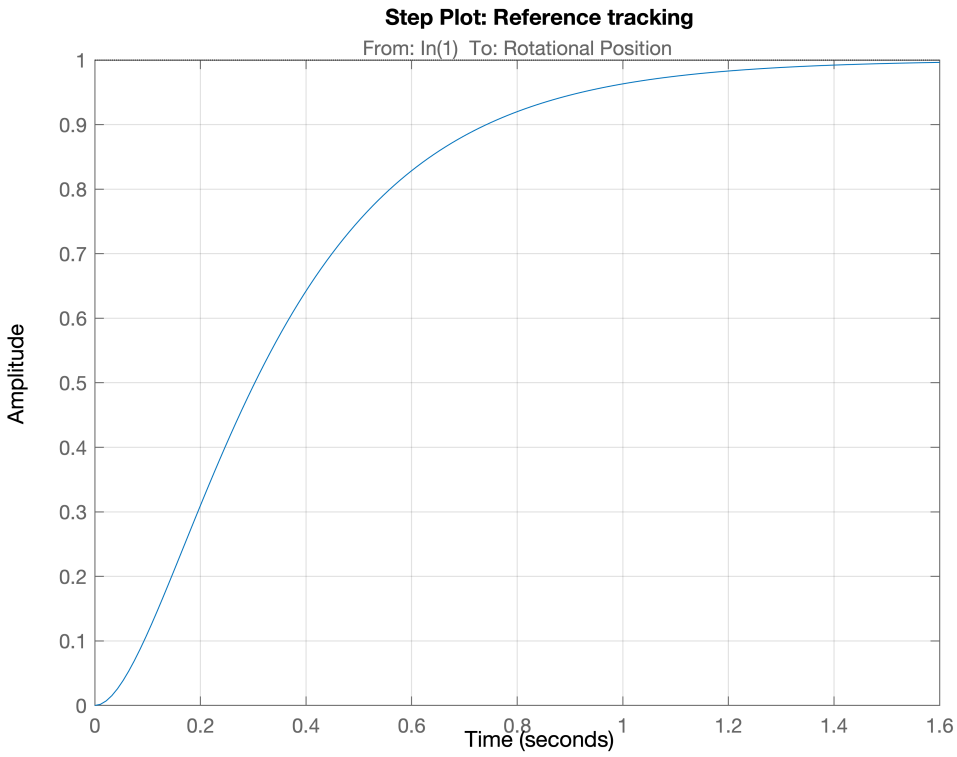
```
      RiseTime: 0.9111
  TransientTime: 1.6545
   SettlingTime: 1.6545
    SettlingMin: 0.9013
    SettlingMax: 0.9997
      Overshoot: 0
     Undershoot: 0
           Peak: 0.9997
       PeakTime: 3.3207
```

```
% Clear Temporary Variables
clear Response
```

Make a controller for angular position (phi)

```
% Convert Response Time to Bandwidth
% Bandwidth is equivalent to 2 divided by the Response Time
wc = 2/0.761527;

% Define options for pidtune command
opts = pidtuneOptions('DesignFocus','reference-tracking');

% PID tuning algorithm for linear plant model
[C_phi,pidInfo2] = pidtune(posTF_phi,'P',wc,opts);
```

```
% Clear Temporary Variables
clear wc opts

% Get desired loop response
Response2 = getPIDLoopResponse(C_phi,posTF_phi,'closed-loop');

% Plot the result
stepplot(Response2)
title('Step Plot: Reference tracking')
grid on
```




```
% Display system response characteristics
disp(stepinfo(Response2))
```

```
        RiseTime: 0.6486
    TransientTime: 1.1573
     SettlingTime: 1.1573
      SettlingMin: 0.9031
      SettlingMax: 0.9990
        Overshoot: 0
       Undershoot: 0
             Peak: 0.9990
         PeakTime: 1.9173
```

```
% Clear Temporary Variables
clear Response2
```

Display the Kp, Ki, and Kd values for each new controller

```
KP_rho = C_rho.Kp; KI_rho = C_rho.Ki; KD_rho = C_rho.Kd;
KP_phi = C_phi.Kp; KI_phi = C_phi.Ki; KD_phi = C_phi.Kd;
fprintf('const float KP_RHO = %f, KI_RHO = %f, KD_RHO = %f;\n',KP_rho, KI_rho,KD_rho)
```

```
const float KP_RHO = 41.507628, KI_RHO = 0.000000, KD_RHO = 0.000000;
```

```
fprintf('const float KP_PHI = %f, KI_PHI = %f, KD_PHI = %f;\n',KP_phi, KI_phi,KD_phi)
```

```
const float KP_PHI = 380.556400, KI_PHI = 0.000000, KD_PHI = 0.000000;
```