Graph Generation

In [2]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Raw

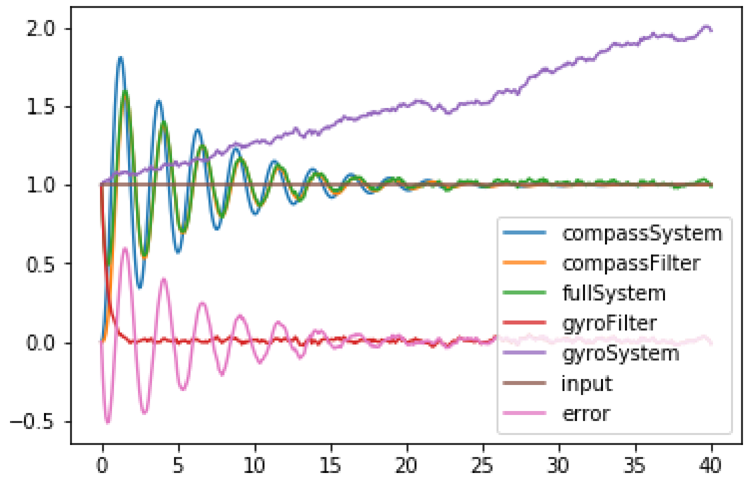
Let's consider how the raw signals look like:

In [3]:

```
stepRaw = pd.read_csv("stepRaw.csv")
[plt.plot(stepRaw.time, stepRaw[i], label=i) for i in stepRaw.columns[stepRaw.columns != "t
plt.legend()
```

Out[3]:

```
<matplotlib.legend.Legend at 0x17f6a5a7dd8>
```

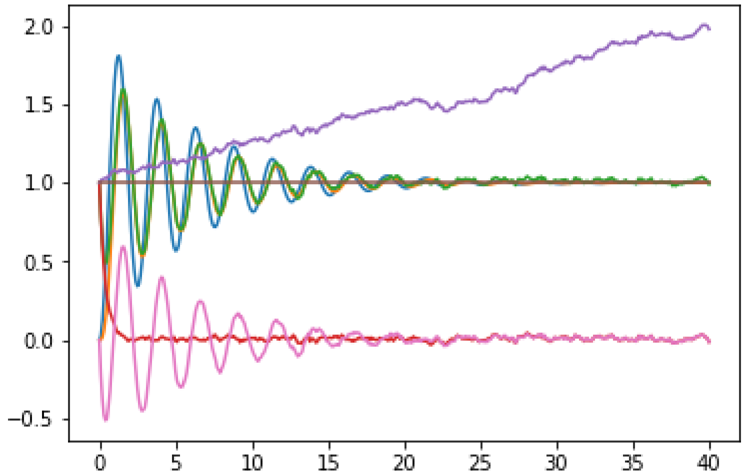


In [4]:

```
plt.plot(stepRaw.time, stepRaw[stepRaw.columns[stepRaw.columns != "time"]])
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x17f6db43198>,
 <matplotlib.lines.Line2D at 0x17f6d5eb940>,
 <matplotlib.lines.Line2D at 0x17f6d5eba90>,
 <matplotlib.lines.Line2D at 0x17f6d5ebbe0>,
 <matplotlib.lines.Line2D at 0x17f6d5ebd30>,
 <matplotlib.lines.Line2D at 0x17f6d5ebe80>,
 <matplotlib.lines.Line2D at 0x17f6d5ebfd0>]
```



Analytics

Note that if we want to model how the system error changes per

In [5]:

```
# Input signal types modelled
signal_types = ["ramp", "step", "sine"]
full_model_data_raw = pd.DataFrame([])
for i in signal_types:
    for j in os.listdir('cutoffVariations'):
        if j.startswith(i) and j.endswith("Analytics.csv"):
            # Select iteration cutoff frequency from file name
            iteration_cutoff_frequency = j.replace(i, "").replace("Analytics.csv", "").repl
            iteration_read_data = pd.read_csv("./cutoffVariations/" + j)
            # Create cutoff frequency and input signalType columns accordingly
            iteration_read_data.loc[:, 'cutoffFrequency'] = np.array([iteration_cutoff_freq
            iteration_read_data.loc[:, 'signalType'] = np.array([i] * len(iteration_read_da
            # Append into model
            full_model_data_raw = full_model_data_raw.append(iteration_read_data, ignore_ir
full_model_data_raw
```

Out[5]:

	signalsNamesOrdered	maxSignals	minSignals	meanSignals	standardDeviationSignals
0	compassSystem	39.947062	0.000000	19.794748	11.680382
1	compassFilter	32.091048	0.000000	13.504995	9.99530:
2	fullSystem	40.126723	0.000000	19.961436	11.711481
3	gyroFilter	8.084963	0.000000	6.456441	2.067357
4	gyroSystem	40.976671	0.000000	20.340501	11.956389
5	input	40.000000	0.000000	19.851624	11.674118
6	error	0.219177	0.000000	0.109812	0.050364
7	compassSystem	39.947062	0.000000	19.794748	11.680382
8	compassFilter	33.706573	0.000000	14.572057	10.482884
9	fullSystem	40.084214	0.000000	19.933131	11.70108(
10	gyroFilter	6.453096	0.000000	5.361074	1.558665
11	gyroSystem	40.976671	0.000000	20.340501	11.956389
12	input	40.000000	0.000000	19.851624	11.674118
13	error	0.178231	-0.006875	0.081508	0.04229(
14	compassSystem	39.947062	0.000000	19.794748	11.680382
15	compassFilter	30.130522	0.000000	12.324548	9.379886
16	fullSystem	40.177773	0.000000	19.992916	11.724958
17	gyroFilter	10.077234	0.000000	7.668368	2.69115:
18	gyroSystem	40.976671	0.000000	20.340501	11.956389
19	input	40.000000	0.000000	19.851624	11.674118
20	error	0.266214	0.000000	0.141293	0.06161:
21	compassSystem	39.947062	0.000000	19.794748	11.680382
22	compassFilter	35.006495	0.000000	15.506652	10.849544
23	fullSystem	40.049865	0.000000	19.908377	11.693423
24	gyroFilter	5.132453	0.000000	4.401726	1.159376

	signalsNamesOrdered	maxSignals	minSignals	meanSignals	standardDeviationSignals
25	gyroSystem	40.976671	0.000000	20.340501	11.956389
26	input	40.000000	0.000000	19.851624	11.674118
27	error	0.142887	-0.033133	0.056754	0.036775
28	compassSystem	39.947062	0.000000	19.794748	11.680382
29	compassFilter	36.039901	0.000000	16.304622	11.113689
...	...	...	...	...	..
22692	input	0.999994	-1.000000	0.046251	0.705648
22693	error	0.530177	-0.375446	-0.002056	0.172378
22694	compassSystem	1.486787	-1.227125	0.055433	0.846847
22695	compassFilter	1.486786	-1.227128	0.055387	0.846825
22696	fullSystem	1.486438	-1.227130	0.055458	0.846784
22697	gyroFilter	0.001296	-0.001217	0.000072	0.000713
22698	gyroSystem	1.974429	-0.875123	0.522334	0.757027
22699	input	0.999997	-1.000000	0.057529	0.710508
22700	error	0.530888	-0.375446	-0.002071	0.172637
22701	compassSystem	1.486803	-1.226332	0.053673	0.842332
22702	compassFilter	1.486801	-1.226332	0.053628	0.842309
22703	fullSystem	1.486453	-1.226314	0.053701	0.842268
22704	gyroFilter	0.001295	-0.001228	0.000073	0.000713
22705	gyroSystem	1.974367	-0.875123	0.525803	0.755088
22706	input	0.999994	-0.999990	0.055731	0.707920
22707	error	0.530908	-0.375446	-0.002031	0.170889
22708	compassSystem	1.499832	-1.250467	0.043888	0.847792
22709	compassFilter	1.499998	-1.250382	0.043699	0.847553
22710	fullSystem	1.496546	-1.250636	0.044103	0.847336
22711	gyroFilter	0.012776	-0.012080	0.000404	0.007157
22712	gyroSystem	1.974424	-0.875123	0.529292	0.760711
22713	input	0.999994	-1.000000	0.042053	0.710333
22714	error	0.545940	-0.375591	0.002050	0.170442
22715	compassSystem	1.499320	-1.249857	0.043987	0.844189
22716	compassFilter	1.479671	-1.240722	0.041346	0.837590
22717	fullSystem	1.445953	-1.252552	0.046478	0.837041
22718	gyroFilter	0.128638	-0.118223	0.005132	0.076661
22719	gyroSystem	1.974163	-0.875123	0.530335	0.759015
22720	input	0.999994	-0.999990	0.042615	0.707197
22721	error	0.533413	-0.355339	0.003863	0.166786

22722 rows × 14 columns

In [6]:

```
# Create multiIndex for all model signals signalsNamesOrdered and input signalType
full_model_multiIndex = pd.MultiIndex.from_frame(full_model_data_raw.loc[:, ["signalsNamesO
# Remove index columns from DataFrame
full_model_data = full_model_data_raw.loc[:, (full_model_data_raw.columns != "signalsNamesO
                                                  & (full_model_data_raw.columns != "s
full_model_data
```

Out[6]:

		maxSignals	minSignals	meanSignals	standardDeviationSi
signalsNamesOrdered	**signalType**				
compassSystem	**ramp**	39.947062	0.000000	19.794748	11.68
compassFilter	**ramp**	32.091048	0.000000	13.504995	9.9
fullSystem	**ramp**	40.126723	0.000000	19.961436	11.7
gyroFilter	**ramp**	8.084963	0.000000	6.456441	2.06
gyroSystem	**ramp**	40.976671	0.000000	20.340501	11.9
input	**ramp**	40.000000	0.000000	19.851624	11.6
error	**ramp**	0.219177	0.000000	0.109812	0.0
compassSystem	**ramp**	39.947062	0.000000	19.794748	11.68
compassFilter	**ramp**	33.706573	0.000000	14.572057	10.48
fullSystem	**ramp**	40.084214	0.000000	19.933131	11.70
gyroFilter	**ramp**	6.453096	0.000000	5.361074	1.5
gyroSystem	**ramp**	40.976671	0.000000	20.340501	11.9
input	**ramp**	40.000000	0.000000	19.851624	11.6
error	**ramp**	0.178231	-0.006875	0.081508	0.04
compassSystem	**ramp**	39.947062	0.000000	19.794748	11.68
compassFilter	**ramp**	30.130522	0.000000	12.324548	9.37
fullSystem	**ramp**	40.177773	0.000000	19.992916	11.72
gyroFilter	**ramp**	10.077234	0.000000	7.668368	2.6
gyroSystem	**ramp**	40.976671	0.000000	20.340501	11.9
input	**ramp**	40.000000	0.000000	19.851624	11.6
error	**ramp**	0.266214	0.000000	0.141293	0.06
compassSystem	**ramp**	39.947062	0.000000	19.794748	11.68
compassFilter	**ramp**	35.006495	0.000000	15.506652	10.84
fullSystem	**ramp**	40.049865	0.000000	19.908377	11.6
gyroFilter	**ramp**	5.132453	0.000000	4.401726	1.1
gyroSystem	**ramp**	40.976671	0.000000	20.340501	11.9
input	**ramp**	40.000000	0.000000	19.851624	11.6
error	**ramp**	0.142887	-0.033133	0.056754	0.0
compassSystem	**ramp**	39.947062	0.000000	19.794748	11.68

		maxSignals	minSignals	meanSignals	standardDeviationSi
signalsNamesOrdered	signalType				
compassFilter	**ramp**	36.039901	0.000000	16.304622	11.1
...	...	...	...	...	
input	**sine**	0.999994	-1.000000	0.046251	0.70
error	**sine**	0.530177	-0.375446	-0.002056	0.17
compassSystem	**sine**	1.486787	-1.227125	0.055433	0.84
compassFilter	**sine**	1.486786	-1.227128	0.055387	0.84
fullSystem	**sine**	1.486438	-1.227130	0.055458	0.84
gyroFilter	**sine**	0.001296	-0.001217	0.000072	0.00
gyroSystem	**sine**	1.974429	-0.875123	0.522334	0.75
input	**sine**	0.999997	-1.000000	0.057529	0.71
error	**sine**	0.530888	-0.375446	-0.002071	0.17
compassSystem	**sine**	1.486803	-1.226332	0.053673	0.84
compassFilter	**sine**	1.486801	-1.226332	0.053628	0.84
fullSystem	**sine**	1.486453	-1.226314	0.053701	0.84
gyroFilter	**sine**	0.001295	-0.001228	0.000073	0.00
gyroSystem	**sine**	1.974367	-0.875123	0.525803	0.75
input	**sine**	0.999994	-0.999990	0.055731	0.70
error	**sine**	0.530908	-0.375446	-0.002031	0.17
compassSystem	**sine**	1.499832	-1.250467	0.043888	0.84
compassFilter	**sine**	1.499998	-1.250382	0.043699	0.84
fullSystem	**sine**	1.496546	-1.250636	0.044103	0.84
gyroFilter	**sine**	0.012776	-0.012080	0.000404	0.00
gyroSystem	**sine**	1.974424	-0.875123	0.529292	0.76
input	**sine**	0.999994	-1.000000	0.042053	0.71
error	**sine**	0.545940	-0.375591	0.002050	0.17
compassSystem	**sine**	1.499320	-1.249857	0.043987	0.84
compassFilter	**sine**	1.479671	-1.240722	0.041346	0.83
fullSystem	**sine**	1.445953	-1.252552	0.046478	0.83
gyroFilter	**sine**	0.128638	-0.118223	0.005132	0.07
gyroSystem	**sine**	1.974163	-0.875123	0.530335	0.75
input	**sine**	0.999994	-0.999990	0.042615	0.70
error	**sine**	0.533413	-0.355339	0.003863	0.16

22722 rows × 12 columns

In [7]:

```
full_model_data.index.levels[0]
```

Out[7]:

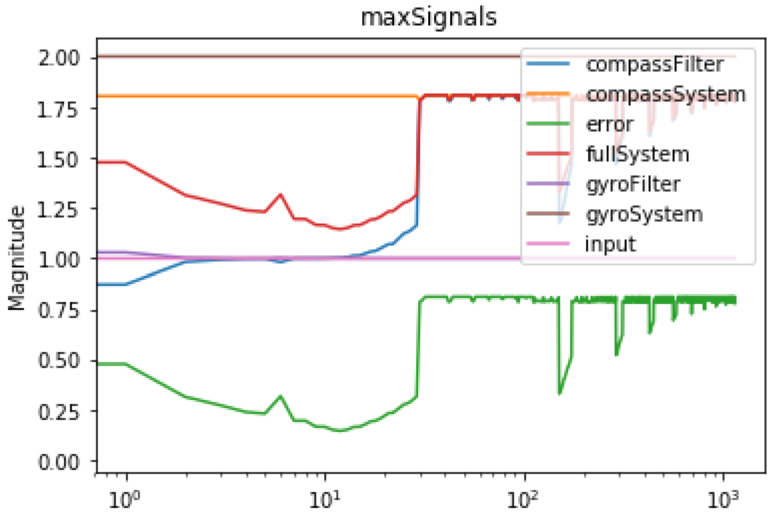
```
Index(['compassFilter', 'compassSystem', 'error', 'fullSystem', 'gyroFilte
r',
       'gyroSystem', 'input'],
      dtype='object', name='signalsNamesOrdered')
```

In [8]:

```
signal = "step"
i = 0
# Select all analytic metrics except cutoffFrequency
for analytic_metric in full_model_data.columns[full_model_data.columns != "cutoffFrequency"
    for model_signal in full_model_data.index.levels[0]: # model signals signalsNamesOrdere
        # Index per model signals signalsNamesOrdered for only for one input signalType
        model_signal_data = full_model_data.loc[pd.IndexSlice[model_signal, signal], :] # S

        plt.figure(i)
        plt.plot(model_signal_data.cutoffFrequency, model_signal_data[analytic_metric], lab
        plt.title(analytic_metric)
        plt.ylabel('Magnitude')
        plt.xlabel('Frequency rad/s')
        plt.xscale('log')
    i+=1
    plt.legend(loc="upper right")
```

```
C:\Program Files\Anaconda3\lib\site-packages\pandas\core\indexing.py:1494:
PerformanceWarning: indexing past lexsort depth may impact performance.
  return self._getitem_tuple(key)
```



Analytics for all signals

So how will we determine the error from each signal accordingly? Let's assume it's propotional. We know the cutoff frequency determines how much of the compass dominates the response, and because its complementary, this also means how much of the gyro is not present in the response. So we want to know what is the error proportionality. We could in theory see the frequency response of the signals that we have, and compare as that changes. Do we have to create our own fast fourier transform response? Let's do it to see what happens.

How do we determine the error proportionality. We can consider a simple step signal case.

It would be inaccurate to linearly substract the value between the error and the

In [ ]:

```

```

In [ ]:

```

```