```
def response_plot_data(timedata, variable1, variable2, time0, time1):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    time = np.round(np.asarray(timedata.get('data')),3)

    if time0 == 'start':
        startind = 0
    else:
        startind = list(time).index(time0)
    if time1 == 'end':
        endind = -1
    else:
        endind = list(time).index(time1)
    if variable1.get('description') == 'Time':
        x = time[startind:endind]
    else:
        x = np.asarray(list(itertools.chain.from_iterable(variable1.get('data'))))[startind:endind]
    y = np.asarray(list(itertools.chain.from_iterable(variable2.get('data'))))[startind:endind]

    plt.xlabel(str(variable1.get('description')) + ' [' + str(variable1.get('units')) + ']', fontsi
    plt.ylabel(str(variable2.get('description')) + ' [' + str(variable2.get('units')) + ']', fontsi

    plt.plot(x,y)
    return plt.show()

def state_space_plot(motion, data):

    import numpy as np
    import control.matlab as control
    import matplotlib.pyplot as plt
    from state_space_con import state_space_conv

    states = state_space_conv(data)
    symsys = states[0]
    asymsys = states[2]
    if motion == 'phugoid':

        timesim = 125
        t = np.linspace(0, timesim, 1001)
        U = np.zeros(len(t))
        U[0:80]= -0.09
        U[80:-1] = 0
        yout, T, xout = control.lsim(symsys, U, t)

        f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
        f.suptitle('Symmetric Flight - Phugoid Motion')
        T = np.ndarray.tolist(T)
        yout = np.ndarray.tolist(yout)

        plt.tight_layout()
        ax1.plot(T, np.transpose(yout)[0])
        ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Velocity [m/s]')

        ax2.plot(T, np.transpose(yout)[1])
        ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')
```

```
    ax3.plot(T, np.transpose(yout)[2])
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

    ax4.plot(T, np.transpose(yout)[3])
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')

    plt.tight_layout()
    plt.subplots_adjust(top=0.92)

    f.set_figheight(6)
    f.set_figwidth(9)
    plt.savefig("sym-numerical-phugoid", dpi=300)
    period = ((np.pi*2)/np.imag(states[1][2]))
    c = control.damp(symsys, doprint=True)
    print('period = ' + str(period))
    return plt.show()

if motion == 'short':

    timesim = 8
    t = np.linspace(0, timesim, 1001)
    U = np.zeros(len(t))
    U[0:-1]= -0.024
    yout, T, xout = control.lsim(symsys, U, t)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Symmetric Flight - Short Period Motion')
    T = np.ndarray.tolist(T)
    yout = np.ndarray.tolist(yout)

    ax1.plot(T, np.transpose(yout)[0])
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Velocity [m/s]')

    ax2.plot(T, np.transpose(yout)[1])
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')

    ax3.plot(T, np.transpose(yout)[2])
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

    ax4.plot(T, np.transpose(yout)[3])
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')

    plt.tight_layout()
    f.set_figheight(7)
    f.set_figwidth(8)

    plt.tight_layout()
    plt.subplots_adjust(top=0.92)

    f.set_figheight(6)
    f.set_figwidth(9)
    plt.savefig("sym-numerical-short", dpi=300)
    period = ((np.pi*2)/np.imag(states[1][1]))
    c = control.damp(symsys, doprint=True)
    print('period = ' + str(period))
    return plt.show()
```

```
if motion == 'aperiodic':

    timesim = 8
    t = np.linspace(0, timesim, 1001)
    U = np.zeros((len(t),2))
    U[3:-1,:] = -0.0256

    yout, T, xout = control.lsim(asymsys, U, t)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Aperiodic Roll Motion')

    ax1.plot(T, np.rad2deg(np.transpose(yout)[0]))
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Andle of Side Slip [deg]')

    ax2.plot(T, np.rad2deg(np.transpose(yout)[1]))
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

    ax3.plot(T, np.rad2deg(np.transpose(yout)[2]))
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

    ax4.plot(T, np.rad2deg(np.transpose(yout)[3]))
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

    plt.tight_layout()
    plt.subplots_adjust(top=0.92)

    f.set_figheight(6)
    f.set_figwidth(9)
    plt.savefig("asym-numerical-aperiodic", dpi=300)
    c = control.damp(asymsys, doprint=True)
    print('period = ' + str(0))
    return plt.show()

if motion == 'dutch':

    timesim = 20
    t = np.linspace(0, timesim, 1001)
    U = np.zeros((len(t),2))
    U[0+25:95+25,1] = -0.14
    U[0+25:95+25,1] = 0.14
    U[95+25:180+25,1] = 0
    U[250+25:-1,1] = 0

    yout, T, xout = control.lsim(asymsys, U, t)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Dutch Roll Motion')

    ax1.plot(T, np.rad2deg(np.transpose(yout)[0]))
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Andle of Side Slip [deg]')

    ax2.plot(T, np.rad2deg(np.transpose(yout)[1]))
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Yaw Angle [deg]')

    ax3.plot(T, np.rad2deg(np.transpose(yout)[2]))
```

```
        ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

        ax4.plot(T, np.rad2deg(np.transpose(yout)[3]))
        ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

        plt.tight_layout()
        plt.subplots_adjust(top=0.92)

        f.set_figheight(6)
        f.set_figwidth(9)
        plt.savefig("asym-numerical-dutch", dpi=300)
        period = ((np.pi*2)/np.imag(states[3][1]))
        c = control.damp(asymsys, doprint=True)
        print('period = ' + str(period))
        return plt.show()

    if motion == 'spiral':

        timesim = 120
        t = np.linspace(0, timesim, 1001)
        U = np.zeros((len(t),2))
        U[0:30,:] = -0.01
        U[30:-1,:] = 0

        yout, T, xout = control.lsim(asymsys, U, t)

        f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
        f.suptitle('Asymmetric Flight - Spiral Motion')

        ax1.plot(T, np.rad2deg(np.transpose(yout)[0]))
        ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Andle of Side Slip [deg]')

        ax2.plot(T, np.rad2deg(np.transpose(yout)[1]))
        ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Yaw Angle [deg]')

        ax3.plot(T, np.rad2deg(np.transpose(yout)[2]))
        ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

        ax4.plot(T, np.rad2deg(np.transpose(yout)[3]))
        ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

        plt.tight_layout()
        plt.subplots_adjust(top=0.92)

        f.set_figheight(6)
        f.set_figwidth(9)
        plt.savefig("asym-numerical-spiral", dpi=300)
        c = control.damp(asymsys, doprint=True)
        print('period = ' + str(0))
        return plt.show()

def compare_plot(timedata, variable1, variable2, variable3, variable4, variable5, variable6, time0,
    import numpy as np
    import control.matlab as control
    import matplotlib.pyplot as plt
    from state_space_con import state_space_conv
    import itertools
```

```
    time = np.round(np.asarray(timedata.get('data')),3)

    if time0 == 'start':
        startind = 0
    else:
        startind = list(time).index(time0)
    if time1 == 'end':
        endind = -1
    else:
        endind = list(time).index(time1)
    if variable1.get('description') == 'Time':
        x = time[startind:endind] - time0
    else:
        x = np.asarray(list(itertools.chain.from_iterable(variable1.get('data'))))[startind:endind]

    y1 = np.asarray(list(itertools.chain.from_iterable(variable2.get('data'))))[startind:endind]
    y2 = np.asarray(list(itertools.chain.from_iterable(variable3.get('data'))))[startind:endind]
    y3 = np.asarray(list(itertools.chain.from_iterable(variable4.get('data'))))[startind:endind]
    y4 = np.asarray(list(itertools.chain.from_iterable(variable5.get('data'))))[startind:endind]
    y5 = np.asarray(list(itertools.chain.from_iterable(variable6.get('data'))))[startind:endind]

    X0 = np.transpose(np.matrix([data[5], 0, 0, 0]))
    XX0 = np.transpose(np.matrix([0, 0, 0, 0]))
#    X0 = np.transpose(np.matrix([0, 0, 0, 0]))
    states = state_space_conv(data)
    symsys = states[0]
    asymsys = states[2]

    if motion == 'phugoid':
        timesim = 125
        t = np.linspace(0, timesim, 1001)
        U = np.zeros(len(t))
        U[0:80]= -0.7
        U[80:-1] = 0.19
        yout, T, xout = control.lsim(symsys, U, t, X0)

        f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
        f.suptitle('Symmetric Flight - Phugoid Motion')
        T = np.ndarray.tolist(T)
        yout = np.ndarray.tolist(yout)

        ax1.plot(T, U)
        ax1.plot(x, y1)
        ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Elevator Deflection Angle [deg]')

        ax2.plot(T, np.transpose(yout)[1]+data[3])
        ax2.plot(x, y2)
        ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')

        ax3.plot(T, np.transpose(yout)[2]+data[2]-5.5)
        ax3.plot(x, y3)
        ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

        ax4.plot(T, np.transpose(yout)[3])
        ax4.plot(x, y4)
        ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')
```

```
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("sym-compare-phugoid", dpi=300)
    return plt.show()

if motion == 'short':

    timesim = 8
    t = np.linspace(0, timesim, 1001)
    U = np.zeros(len(t))
    U[0:-1]= -0.6
    yout, T, xout = control.lsim(symsys, U, t, X0)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Symmetric Flight - Short Period Motion')
    T = np.ndarray.tolist(T)
    yout = np.ndarray.tolist(yout)

    ax1.plot(T, U)
    ax1.plot(x, y1)
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Elevator Deflection Angle [deg]')

    ax2.plot(T, np.transpose(yout)[1]+data[3])
    ax2.plot(x, y2)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')

    ax3.plot(T, np.transpose(yout)[2]+data[2])
    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

    ax4.plot(T, np.transpose(yout)[3])
    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')

    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("sym-compare-short", dpi=300)
    return plt.show()

if motion == 'dutch':

    timesim = 20
    t = np.linspace(0, timesim, 1001)
    U = np.zeros((len(t),2))
    U[0+25:95+25,1] = (-9.7+0.6)/0.4
    U[0+25:95+25,1] = (-9.7+0.6)/0.4
    U[95+25:180+25,1] = (3.64+0.6)/0.4
    U[250+25:-1,1] = 0

    yout, T, xout = control.lsim(asymsys, U, t, XX0)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
```

```
    f.suptitle('Asymmetric Flight - Dutch Roll Motion')

    ax1.plot(T, (U[:,1]-0.8)*0.4)
    ax1.plot(x, y2)
    ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Rudder [deg]')

    ax2.plot(T, ((np.transpose(yout)[1])+data[6])*0.4)
    ax2.plot(x, y5)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

    ax3.plot(T, -(np.transpose(yout)[2]+data[6])*0.4)
    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

    ax4.plot(T, -(np.transpose(yout)[3]+data[7])*0.4)
    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("asym-compare-dutch", dpi=300)
    return plt.show()

if motion == 'aperiodic':

    timesim = 8
    t = np.linspace(0, timesim, 1001)
    U = np.zeros((len(t),2))
    U[3:-1,:] = -1.2-0.83

    yout, T, xout = control.lsim(asymsys, U, t, XX0)

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Aperiodic Roll Motion')

    ax1.plot(T, (U[:,0]+0.83))
    ax1.plot(x, y1)
    ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Aileron [deg]')

    ax2.plot(T, (np.transpose(yout)[1])+data[6])
    ax2.plot(x, y5)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

    ax3.plot(T, -(np.transpose(yout)[2])+data[7])
    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

    ax4.plot(T, -(np.transpose(yout)[3])+data[8])
    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("asym-compare-aperiodic", dpi=300)
```

```
        return plt.show()

    if motion == 'spiral':

        timesim = 120
        t = np.linspace(0, timesim, 1001)
        U = np.zeros((len(t),2))
        U[0:30,:] = -1.03
        U[30:-1,:] = 0

        yout, T, xout = control.lsim(asymsys, U, t, XX0)

        f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
        f.suptitle('Asymmetric Flight - Spiral Motion')

        ax1.plot(T, ((U[:,0])+(0.65)))
        ax1.plot(x, y1)
        ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Aileron [deg]')

        ax2.plot(T, ((np.transpose(yout)[1])+data[6])+3)
        ax2.plot(x, y5)
        ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

        ax3.plot(T, -((np.transpose(yout)[2])+data[7]))
        ax3.plot(x, y3)
        ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

        ax4.plot(T, -((np.transpose(yout)[3])+data[8])+1.2)
        ax4.plot(x, y4)
        ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')
        plt.tight_layout()
        f.set_figheight(6)
        f.set_figwidth(8)

        plt.savefig("asym-compare-spiral", dpi=300)
        return plt.show()

def actual_plot(timedata, variable1, variable2, variable3, variable4, variable5, variable6, time0,
    import numpy as np
    import control.matlab as control
    import matplotlib.pyplot as plt
    from state_space_con import state_space_conv
    import itertools

    time = np.round(np.asarray(timedata.get('data')),3)

    if time0 == 'start':
        startind = 0
    else:
        startind = list(time).index(time0)
    if time1 == 'end':
        endind = -1
    else:
        endind = list(time).index(time1)
    if variable1.get('description') == 'Time':
        x = time[startind:endind] - time0
    else:
```

```
    x = np.asarray(list(itertools.chain.from_iterable(variable1.get('data'))))[startind:endind]

y1 = np.asarray(list(itertools.chain.from_iterable(variable2.get('data'))))[startind:endind]
y2 = np.asarray(list(itertools.chain.from_iterable(variable3.get('data'))))[startind:endind]
y3 = np.asarray(list(itertools.chain.from_iterable(variable4.get('data'))))[startind:endind]
y4 = np.asarray(list(itertools.chain.from_iterable(variable5.get('data'))))[startind:endind]
y5 = np.asarray(list(itertools.chain.from_iterable(variable6.get('data'))))[startind:endind]

if motion == 'phugoid':
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Symmetric Flight - Phugoid Motion')

    ax1.plot(x, y1)
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Elevator Deflection Angle [deg]')

    ax2.plot(x, y2)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')

    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')

    plt.subplots_adjust(top=0.92)
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("sym-actual-phugoid", dpi=300)
    return plt.show()

if motion == 'short':
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Symmetric Flight - Short Period Motion')

    ax1.plot(x, y1)
    ax1.set(xlabel = 'Simulation time T [s]', ylabel = 'Elevator Deflection Angle [deg]')

    ax2.plot(x, y2)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Angle of Attack [deg]')

    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Angle [deg]')#

    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Pitch Rate [deg/s]')

    plt.subplots_adjust(top=0.92)
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("sym-actual-short", dpi=300)
    return plt.show()

if motion == 'dutch':
```

```
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Dutch Roll Motion')

    ax1.plot(x, y2)
    ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Rudder [deg]')

    ax2.plot(x, y5)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

    plt.subplots_adjust(top=0.85)
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("asym-actual-dutch", dpi=300)
    return plt.show()

if motion == 'aperiodic':
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Aperiodic Roll Motion')

    ax1.plot(x, y1)
    ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Aileron [deg]')

    ax2.plot(x, y5)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')

    ax3.plot(x, y3)
    ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

    ax4.plot(x, y4)
    ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

    plt.subplots_adjust(top=0.85)
    plt.tight_layout()
    f.set_figheight(6)
    f.set_figwidth(8)

    plt.savefig("asym-actual-aperiodic", dpi=300)
    return plt.show()

if motion == 'spiral':
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
    f.suptitle('Asymmetric Flight - Spiral Motion')

    ax1.plot(x, y1)
    ax1.set(Xlabel = 'Simulation time T [s]', ylabel = 'Deflection Aileron [deg]')

    ax2.plot(x, y5)
    ax2.set(Xlabel = 'Simulation time T [s]', ylabel = 'Roll Angle [deg]')
```

```
ax3.plot(x, y3)
ax3.set(xlabel = 'Simulation time T [s]', ylabel = 'Roll Rate [deg/s]')#

ax4.plot(x, y4)
ax4.set(xlabel = 'Simulation time T [s]', ylabel = 'Yaw Rate [deg/s]')

plt.subplots_adjust(top=0.85)
plt.tight_layout()
f.set_figheight(6)
f.set_figwidth(8)

plt.savefig("asym-actual-spiral", dpi=300)
return plt.show()
```