EXERCISES

16.06/16.07: MATLAB & Simulink Tutorials

I. Class Materials

1. Download matlab_simulink_tutorial.tar

From a web browser:

Download **matlab_simulink_tutorial.tar** from **http://web.mit.edu/acmath/matlab/course16** to a local directory. On Windows, if you do not have WinZip, download the ZIP file instead.

Alternatively, on Athena:

athena% add acmath
athena% cp /mit/acmath/matlab/course16/matlab_simulink_tutorial.tar **.**

2. Extract this session's sub-directories and files

On Athena** (or the UNIX shell on Mac OS X)**:

tar –xvf matlab_simulink_tutorial.tar

On laptops:

Use your computer's utilities, such as double click or WinZip on Windows or StuffIt on Mac.

Your local work directory **matlab_simulink_tutorial** should now contain the following directories and files:

matlab_simulink_tutorial

Exercise1		**Exercise2**
odeLanderVelocity.m	longTimeResponse.m	f8.mdl
MarsLander.m	f8long.m	

II. Start MATLAB

On Athena:

athena% cd matlab_simulink_tutorial
athena% add matlab
athena% matlab &
>> desktop

On laptops:

Launch MATLAB and navigate to the work directory **matlab_simulink_tutorial**.

III. Exercise 1: Matrices & Differential Equations

Purpose

To practice the following in MATLAB:

- Creating and referencing the elements of matrices and vectors.
- Using MATLAB's differential equation solvers and other built-in functions.
- Understanding MATLAB programs with script and function M-files.

To prepare for the Simulink tutorial and exercise.

1-A Mars Lander Velocity

Background

The velocity **v** of a Mars Lander during EDL (Entry, Descent, Landing) can be described with the differential equation for velocity of a body in free fall:

$$\frac{dv(t)}{dt} = g - \frac{k}{m}v^2$$

Problem

Compute the velocity, **v**, and acceleration**, dv/dt**, for the last stage of landing (after the retro-rockets are launched) for a Mars Lander with mass m=150 kg and drag coefficient k=1.2. The gravity constant on Mars is 3.688 m/s^2. Plot v and dv/dt vs. time on the same plot and show that the acceleration approaches zero as the velocity approaches a terminal velocity value.

1. Open M-files odeLanderVelocity.m and MarsLander.m

```
>> cd Exercise1
>> edit odeLanderVelocity.m
>> edit MarsLander.m
```

Lines that start with % are comments. The rest are MATLAB commands.

2. Follow the instructions in MarsLander.m

- M-file odeLanderVelocity.m defines a differential equation for velocity in free fall.
- Follow instructions in the file MarsLander.m and add the missing code.
- In A1, define global variables M and K for the mass and drag coefficient.
- In A3, define a time vector tspan for the numerical solution.
- In A4, use the built-in MATLAB ODE solver ode45 to compute the numerical solution, which uses function odeLanderVelocity, tspan, and initial velocity V0 as arguments. The solver should return two output arguments: a velocity vector V and a time vector t.

3. Execute the M-file in the Command Window

```
>> MarsLander
```

Look at the plot that is created and interpret the results[1].

[1] For another *Dynamics* example for 16.07, see files getF.m and Central_Force_Motion.m.

1-B F-8 Longitudinal Time Response

Background

The following system of first-order differential equations describes the longitudinal time response of an airplane due to elevator deflection:

$$\begin{bmatrix} \dot{q} \\ \dot{u} \\ \dot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} q \\ u \\ \alpha \\ \theta \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{31} \end{bmatrix} \begin{bmatrix} \delta_e \end{bmatrix}$$

In matrix form the system can be represented as:

$$\dot{x} = Ax + Bv$$

where the variables in the **x** vector are called **state** variables and define the state of the aircraft at any time, while the **v** vector contains the **control** variables.

The following state variables represent perturbations from an equilibrium condition:

q(t) is the pitch rate (deg/s)
u(t) is the perturbation of horizontal velocity (ft/s)
α(t) is the perturbation of the angle of attack from trim (deg)
θ(t) is the perturbation of the pitch angle from trim (deg)

The control variable is δ_e: the elevator deflection (deg) from equilibrium (trimmed) condition.

When the equations are written in the following form, they are said to be in **State Space** form:

$$\dot{x} = Ax + Bv$$
$$y = Cx + Dv$$

where **y** is the output vector (which could be the same as or different from **x**). **A**, **B**, **C**, and **D** are called **state**, **input**, **output**, and **feedthrough** matrices, respectively.

The free longitudinal response of an aircraft can be studied by setting the control input to zero:

$$\dot{x} = Ax$$

Two natural modes of the system, called **Short Period** mode and **Phugoid** mode, are related to the *eigenvalues* of the matrix A.

Problem

The following linear differential equations model the longitudinal motions of the F-8 aircraft:

$$\dot{q}(t) = -0.7q(t) - 0.0344u(t) - 12\alpha(t) - 19\delta_e(t)$$
$$\dot{u}(t) = -0.014u(t) - 0.290\alpha(t) - 0.562\theta(t) - 0.0115\delta_e(t)$$
$$\dot{\alpha}(t) = q(t) - 0.0057u(t) - 1.3\alpha(t) - 0.16\delta_e(t)$$
$$\dot{\theta}(t) = q(t)$$

about the following equilibrium flight conditions:

Altitude:	20,000 ft = 6094 m
Speed:	Mach 0.9 = 916 ft/s = 281.6 m/s
Dynamic pressure:	550 lbs/ft2 = 26,439 N/m2
Trim pitch angle:	2.25 deg
Trim angle of attack:	2.25 deg
Trim elevator angle:	-2.65 deg

Define the state and input matrices **A** and **B** for the State Space form of the system. Compute the **transfer functions** relating the state variables to the control input. Compute and plot the change over time of the elements of the state vector **x** due to a perturbation in the elevator angle, δ_e.

1. Open M-files longTimeResponse.m and f8long.m

```
>> edit longTiimeResponse.m
>> edit f8long.m
```

Lines that start with % are comments. The rest are MATLAB commands.

2. Follow the instructions in f8long.m

- M-file longTimeReponse.m defines the differential equations that describe the longitudinal time response of an aircraft due to input of elevator deflection angle.
- Follow instructions in the file f8long.m and add the missing code.
- In B3, define the state and input matrices A and B.
- In B4, compute the eigenvalues and eigenvectors of A.
- In B5, compute the numerator and denominator of the transfer functions relating the state variables to the control input. Hint: Use the built-in function ss2tf.
- In B8, add the code for plotting the perturbation of the pitch angle θ(t) over time.

3. Execute the M-file in the Command Window

```
>> f8long
```

What is the transfer function relating the horizontal velocity to the elevator deflection?

Write it here: $G(s) = \frac{u(s)}{\delta_e(s)} = ...$

V. Exercise 2: F-8 Controller Design

Purpose

To practice the following in Simulink:

- Creating and saving models.
- Using blocks to create block diagrams.
- Configuring parameters and running simulations.

Background

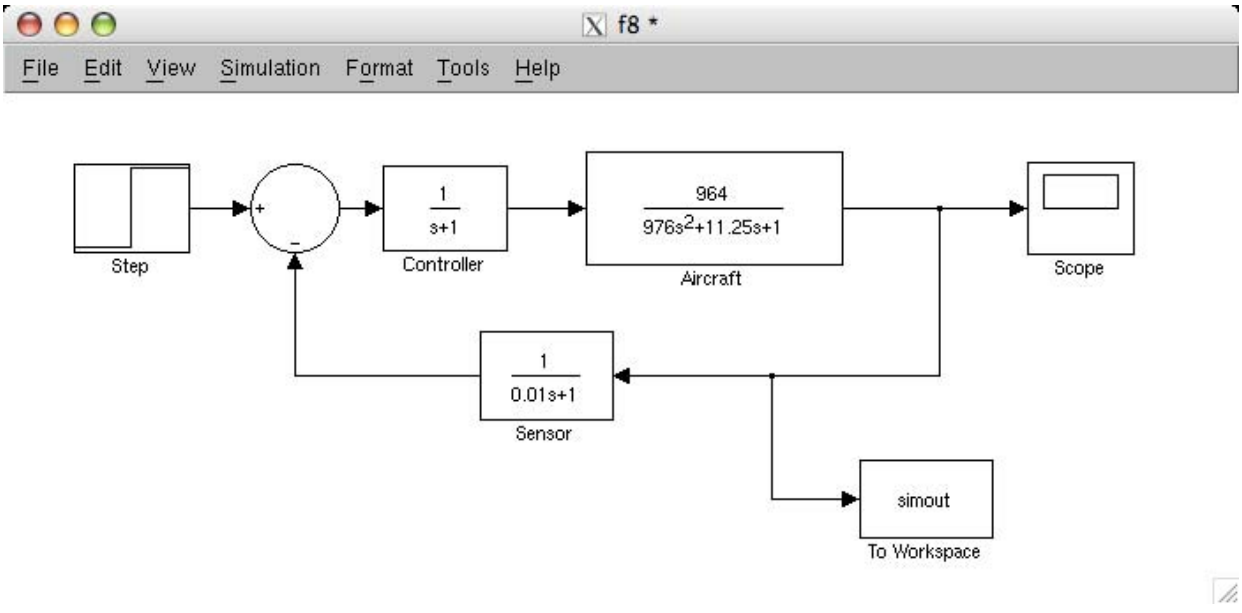
In this exercise, we will be designing a controller for the longitudinal dynamics of the F-8. The transfer function of the horizontal speed of the F-8 to the elevator angle is approximately:

$$G(s) = \frac{u(s)}{\delta_e(s)} = \frac{964}{976s^2 + 11.25s + 1}$$

The controller takes the error in the speed as input and outputs the elevator angle. The F-8 dynamics then converts this elevator angle to airspeed. We will also model a sensor, which will have a small amount of lag. We are modeling the behavior of the system when the desired speed suddenly changes, which will be represented with a step function as input.

Problem

Create the Simulink model shown in the figure below (also saved in file f8.mdl). Run the simulation. Change the transfer function of the controller and re-run the simulation. Repeat until the output has improved.



1. Create the model's block diagram

- Create a new model in Simulink.

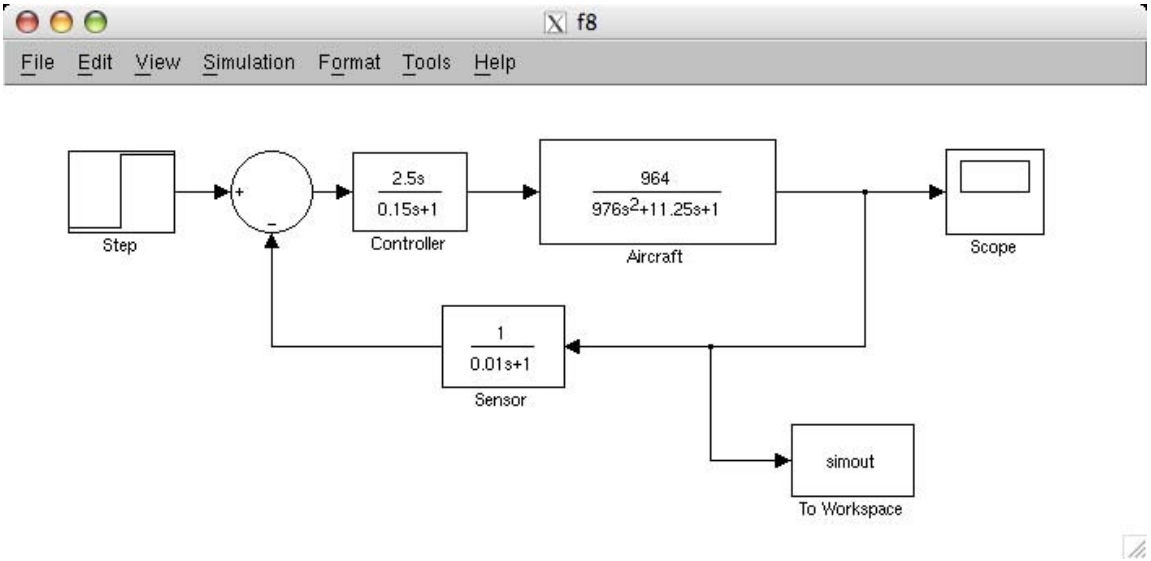
- Add a step function input to the model: use Step Function from the Sources library.
- Add a scope to the model: use Scope from the Sinks library.
- Add three transfer functions to the model: Transfer Fcn from the Continuous library.
- Change the names of the transfer functions to *Controller*, *Aircraft*, and *Sensor*.
- Add a sum operator to the model: Sum from the Math Operations library.
- Change the sum operator to perform subtraction rather than addition. (Double click on the operator and change the list of signs to "|+ -" in the "Block Parameters: Sum" window).
- Change the transfer function of the aircraft. (Double click on the aircraft icon and change the *nominator* and *denominator* using MATLAB matrix notation in the "Function Block Parameters: Aircraft" window. In this exercise both the nominator and denominator are row vectors that represent the coefficients of a polynomial in decreasing exponent of *s*.)
- Change the transfer function of the sensor to $\frac{1}{0.01s+1}$ (i. e. a sensor with 0.01 sec lag).
- Add an output to the MATLAB plot interface: To Workspace from the Sinks library.
- Connect the blocks of the model as shown in the figure.

2. Run the simulation

- Set the simulation to run for 30 seconds: Simulation->Configuration Parameters.
- Run the simulation: Simulation->Start.
- Examine the output of the simulation in the Scope window. (Click on the binocular icon to show the most appropriate scale.) Is this an appropriate way for an aircraft to respond?

3. Change the controller

- Change the transfer function of the controller to $\frac{2.5s}{0.15s+1}$.
- Run the simulation again and observe the output of the scope. Has the new controller improved the output? Your model should now look like this:



- In the MATLAB command line window, type the command who to see the variables in the workplace. Then plot simout vs. tout. What does this plot look like?