Non-linear F-16 Simulation using Simulink and Matlab

Richard S. Russell
University of Minnesota

Version 1.0 June 22, 2003

1 What have you just downloaded?

Is this a video game? No, this a non-linear F-16 model that simulates the dynamics of the real aircraft. This plant can simulate the response of an actual F-16 using one of two models as described by Stevens and Lewis [1] and the NASA Report [2]. From this point forward the model described by Stevens and Lewis will be referred to as the low fidelity model and the model described in the NASA report will be referred to as the high fidelity model. The uses of these models are limitless. But these models will most probably be used by both novice and professionals for simulation and controller design for the F-16 and related aircraft in an educational setting.

2 Anatomy of the Non-linear F-16 Model

2.1 Body axis

The body axes of F-16 model are conventional. The x-axis is positive out the nose of the aircraft. The y-axis is positive out the right wing as you sit in the cockpit and face out the front of the aircraft. The z-axis is positive normal to the x and y axis and points vertically downward when the aircraft is in level flight. The moment axes obey the right hand rule about each axis. Moments about the x-, y- and z-axis are labeled M, L, and N, respectively. The body rates (p, q, and r) and Euler angles (ϕ, θ, and ψ) are also measured positively using the right-hand rule about each axis. Verify your understanding of the body axes description by referring to Figure 1.

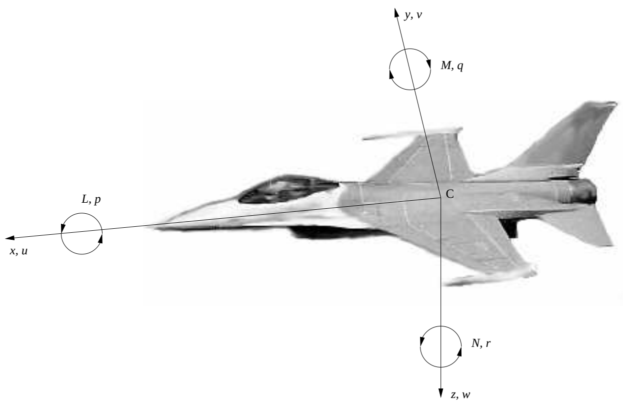


Figure 1: F-16 Body axes, (x, y and z). L = rolling moment, M = pitching moment, N = yawing moment, p = roll rate, q = pitch rate, r = yaw rate.

2.2 Plant

The non-linear F16 model has been constructed using Simulink. The functional Simulnk model is shown in Figure 2 and the actual Simulink model is shown in Figure 3. As can be seen from Figure 2, the non-linear plant of the F-16 requires the four controls, thirteen states, the leading edge flap deflection and a model flag as inputs. The plant will output the twelve state derivatives and six other states of flight. The controls and leading edge flap are discussed in Section 2.4. The input/output states and state derivatives are discussed in Section 2.3.

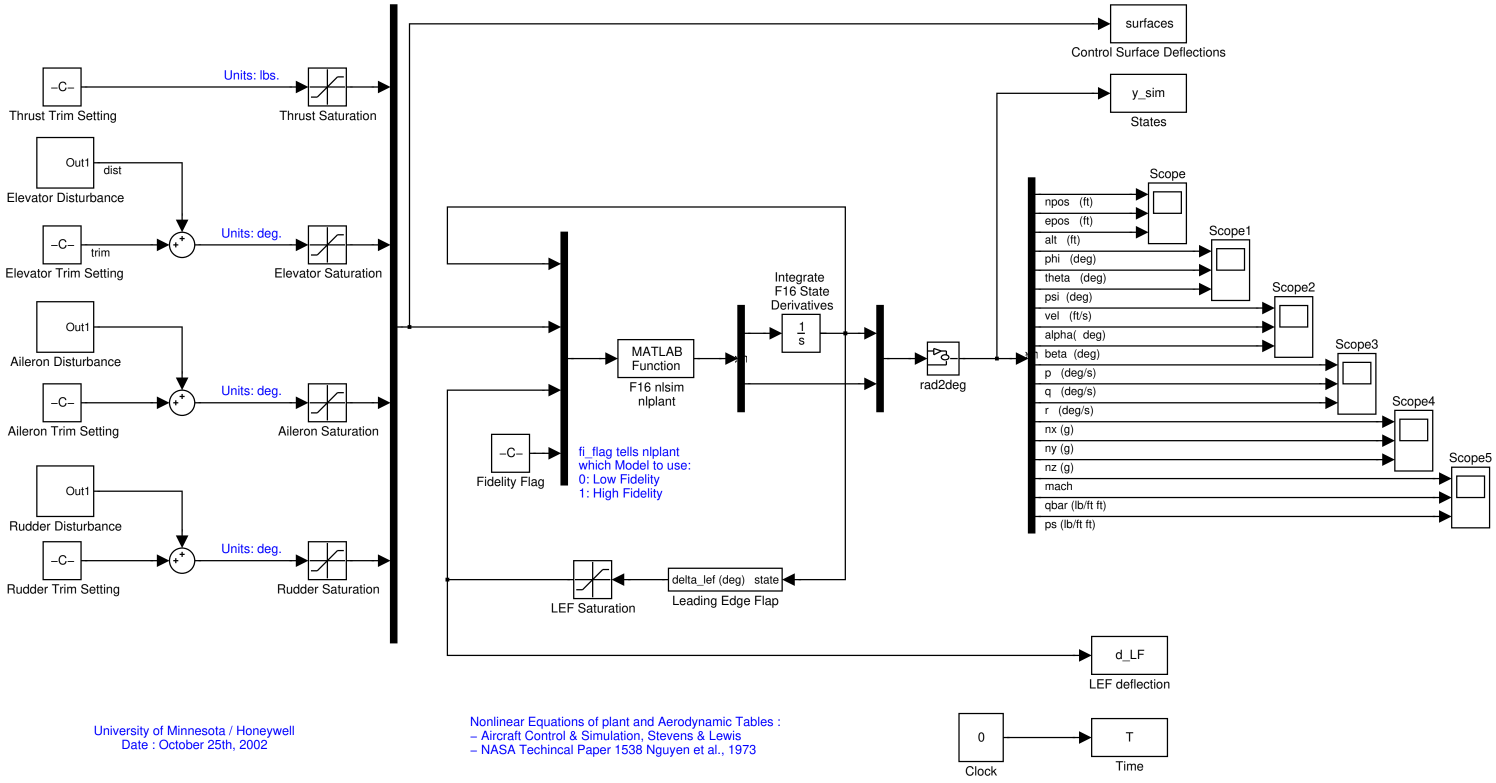


Figure 2: The functional layout of the Simulink model of the F-16 plant.

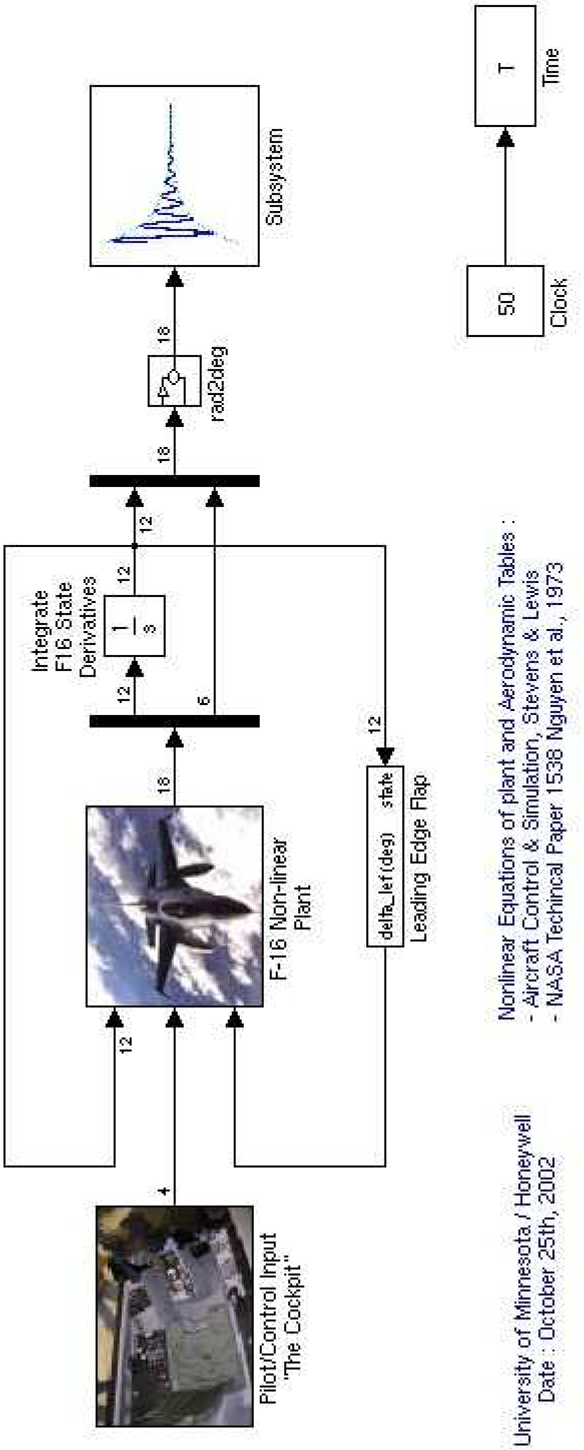


Figure 3: Actual layout of the Simulink Model of the F-16 plant.

2.3 States

The non-linear F-16 plant requires thirteen state inputs. The thirteen states inputs are: north position ($npos$), east position ($epos$), altitude (h), roll angle (ϕ), pitch angle (θ), yaw angle (ψ), total velocity (V_t), angle of attack (α), angle of side-slip (β), roll rate (p), pitch rate (q), yaw rate(r), and finally the deflection of the leading edge flap(δ_{lef}). The leading edge flap deflection is only implemented in the high fidelity model. The leading edge flap is described in more detail in Section 2.4.

The output of the plant are the 12 state derivatives and normalized accelerations in the x, y and z directions (a_{nx}, a_{ny} and a_{nz}, respectively), Mach number (M), free-stream dynamic pressure ($\bar{q}$) and static pressure (P_s).

The initial states of the F-16 constitute the initial conditions of the F-16. And the user can input them arbitrarily but, more likely, the initial states will be supplied from a trimming routine. The trimming routine packaged with these non-linear models trims to steady-state flight for a given altitude and velocity. The F-16 can be initially trimmed in steady wings-level, turning, pull-up, or roll flight. After the plant has taken one discrete time step the twelve input states are found from integrating the twelve output state derivatives.

The units for each states are shown in Table 2.3.

	UNITS		
State	Passed to `nlplant`.	Used by `nlplant`	Passed from `nlplant`
$npos$	ft	ft	ft
$epos$	ft	ft	ft
h	ft	ft	ft
ϕ	rad	rad	rad
θ	rad	rad	rad
ψ	rad	rad	rad
V_t	ft/s	ft/s	ft/s
α	rad	deg	rad
β	rad	deg	rad
p	rad/s	rad/s	rad/s
q	rad/s	rad/s	rad/s
r	rad/s	rad/s	rad/s
a_{nx}	N/A	g	g
a_{ny}	N/A	g	g
a_{nz}	N/A	g	g
M	N/A	–	–
$\bar{q}$	N/A	lb/ft^2	lb/ft^2
P_s	N/A	lb/ft^2	lb/ft^2

Table 2.3 The units of the states used in the F-16 model.

2.4 Controls

The F-16 model allows for control over thrust, elevator, aileron and rudder. The thrust is measured in pounds. Thrust acts positively along the positive body x-axis. Positive thrust causes an increase in acceleration along the body x-axis. For the other control surfaces a positive deflection gives a decrease in the body rates. A positive aileron deflection gives a decrease in the roll rate, p, this requires that the right aileron deflect downward and the left aileron deflect upward. A positive elevator deflection results in a decrease in pitch rate, q, thus elevator is deflected downward. And a positive deflection of the rudder decreases the yaw rate, r, and can be described as a deflection to right. The positive orientations for each control surface is shown in Figure 4. The maximum values and units for each control input is shown in Table 2.4.

Figure 4: The positive control deflections for the control surfaces.

Control	Input	UNITS Used by `nlplant`	Min	Max.
Thrust	lbs.	lbs.	1000	19000 lbs.
Elevator	deg.	deg.	-25	25 deg.
Aileron	deg.	deg.	-21.5	21.5 deg.
Rudder	deg.	deg.	-30	30 deg.
Leading Edge Flap	deg	deg	0	25 deg.

Table 2.4 The control input units and maximum values.

The high fidelity model has an additional control surface that allows for the F-16 to fly at higher angles of attack. This control surface cannot be directly changed by the pilot. Instead its deflection is governed by the angle of attack, and the static and dynamic pressures for which the F-16 is flying. The transfer function that governs the leading edge flap deflection is:

$$\delta_{lef} = 1.38\frac{2s+7.25}{s+7.25}\alpha - 9.05\frac{\bar{q}}{P_s} + 1.45$$

To see the effect of the leading edge flap it helps to look at the linearized plant for the F-16 both with and without the leading edge flap. The following equation, for $\dot{v}$ was taken from the linearized high fidelity model using the leading edge flap, at an altitude of 15,000 ft. and a velocity of 500 ft/s:

$$\dot{v} = 0.0001h - 3.17\theta - 0.0133v + 4.837\alpha - 0.4401\beta - 0.707q$$

The same equation for the linearized high fidelity model, at the same flight condition, but without the leading edge flap gives:

$$\dot{v} = 0.0001h - 3.17\theta - 0.0131v - 10.2070\alpha - 0.1337\beta - 1.5837q$$

This same equation taken from the linearized low fidelity model, at the same flight condition, gives:

$$\dot{v} = 0.0001h - 3.17\theta - 0.0133v - 7.3259\alpha - 1.1965q$$

The important thing to notice from these equations is that when the leading edge flap is used the rate of change of velocity actually increases with increasing angle of attack ($\dot{v} = \ldots + 4.837\alpha + \ldots$). Where as without the leading edge flap the velocity decreases with increasing angle of attack ($\dot{v} = \ldots - 10.207\alpha + \ldots$). Which is common in most conventional aircraft. This reaffirms that the leading edge flap is doing what is supposed to do. That is, allowing the F-16 to fly a high angles of attack.

2.5 Actuators

All of the actuators were modeled as first-order lags with a gain (K) and limits on deflection and rates. The thrust had a unity gain and a rate limit of $\pm 10,000 lbs/s$. The elevator had a gain of 1/0.0495 and rate limits of $\pm 60 deg/s$. The aileron had a gain of 1/0.0495 and rate limits of $\pm 80 deg/s$. The rudder had a gain of 1/0.0495 and rate limits of $\pm 120 deg/s$. Finally, the leading edge flap had a gain of 1/0.136 and a rate limit of $\pm 25 deg/s$. The deflection limits can be found in Section 2.4.

2.6 Differences between the low fidelity and high fidelity model:

Both models use the same navigation equations and equations of motion. The navigation equations are taken from page 81 of Stevens and Lewis [1]. The equations that determine the force and moment coefficients are taken from pg. 37 - 40 of the NASA Report [2].

The file that contains these equations is called `nlplant.c`. The file can be found in the Appendix. The file is well commented and can easily be understood by a novice C programmer. When looking through `nlplant.c` points of confusion can come on line 214. On this line, the difference between the low and high fidelity models become apparent. Each model uses different tables to compute the force and moment coefficients. The aerodynamic data used to find the force and moment coefficients is tabulated as a function of angle of attack, side slip and in some cases elevator deflection. The force and moment coefficients are then found by interpolating the entires for a given angle of attack, side-slip and elevator deflection. The values in these tables come from experimental data, performed in a NASA Langley wind tunnel.

The low fidelity aerodynamic data tables come from the appendix of Stevens and Lewis and the high fidelity aerodynamic tables from Table III of the NASA report. The aerodynamic tables for the low and high fidelity models can be found in the files `lofi_F16_AeroData.c` and `hifi_F16_AeroData.c`, respectively.

The low fidelity model represents a simpler model that does not include the effects of the leading edge flap. There is a complete decoupling between longitudinal and lateral directions. The angle of attack range for the low fidelity model is -10 to 45 degrees and the angle of side-slip ranges from -30 to 30 degrees.

The high fidelity model includes the effect of the leading edge flap and there is a noticeable coupling between the longitudinal and lateral directions much like in a real aircraft. The angle of attack range for the high fidelity model is -20 to 90 degrees and the angle of side-slip ranges from -30 to 30 degrees.

3 Installation Instructions:

1. Unzip the file F16Simulation.tar.gz. Using the command:

   ```
   >> tar xvfz F16Simulation.tar.gz
   ```

   This will create the directory called F16Sim and in it will contain all the files needed to perform non-linear F-16 simulations.

2. Change in to the F16Sim directory.

3. Open Matlab.

4. Mex the file `nlplant.c` using:

   ```
   >> mex nlplant.c
   ```

   This will produce the file `nlplant.mexglx` for Linux distributions, `nlplant.mexsg` for silicon graphics machines, etc. `nlplant.c` is a script file written in C and must be mexed so that it can be used in the Matlab environment. Check to be sure that these files exists.

5. Begin using the F-16 simulation. (See: Using the F-16 Non-Linear Model.)

4 Non-linear F-16 Simulation Files

4.1 F-16 Simulation Files (by filename):

`F16Block.mdl`

This file is the Simulink model of the F-16. At the heart of this model is the function block containing the function `nlplant.c`. This function block will execute the file created by Matlab's mex function called `nlplant.*` The resulting file (nlplant.*) has a platform-dependent extension, as shown in the table below:

sol2, SunOS 5.x	.mexsol
hpux	.mexhpux
hp700	.mexhp7
ibm_rs	.mexrs6
sgi	.mexsg
alpha	.mexaxp
glnx86	.mexglx
Windows	.dll

You can run this model directly from Simulink if all of the proper parameters are supplied or you can use the command `runF16Sim` in Matlab. It is recommended that you use the latter method because it will trim the F-16 at a given altitude and angle of attack, run the model and plot the results.

`runF16Sim.m`

To run, type runF16Sim at the Matlab command line.

```
>> runF16Sim
```

This file contains a program that will used F-16 plant in a non-linear simulation. The user can input disturbances on any of the control surfaces. The program runs the simulation using the Simulink model block named `F16Block.mdl`.

What the program will do:

When executed the program will prompt the user which model to run. It will then ask the user at what conditions (altitude and velocity) the F-16 should be trimmed. The user also has the option to create disturbances on the any of the control surfaces. The simulation will be carried out and the results will be plotted and saved to a text file, of the form:

`ele_*ail_*rud_*[hifi or lofi]model_alt*_vel*.txt`.

`trim_F16.m`

The function used to trim the F-16 at a given steady-state flight condition with specified initial altitude and velocity. This function minimizes the cost function created by `trimfun.m` by varying the free parameters from the initial condition supplied by the user. The free parameters are thrust, elevator, rudder and aileron deflections and angle of attack. The initial guess to the trim flight condition is critical to finding an equilibrium point at the specified altitude and velocity. All of the programs supplied with the download use the same initial guess. This initial guess is 5000 lbs. of thrust, an elevator setting of -0.09 degree, an angle of attack 8.49 degrees, rudder deflection of -0.01 degree and aileron deflection of 0.01 degree. These values offer a good starting point and it allows `trim_F16` to converge to a trim point relatively quickly in most situations. In fact, all of the situations tried by the author of this manual using these initial guesses, an equilibrium point was found. If you are trying to trim the F-16 at extreme flight conditions, these values may or may not need to be changed. Note: If the values for any of the initial conditions need to be changed, they can in runF16Sim.m or runLINF16sim.m.

The steady-state flight conditions and definitions are:

Steady Wings-Level Flight.	$\phi, \dot{\phi}, \dot{\theta}, \dot{\psi} \equiv 0$	
Steady Turning Flight	$\dot{\phi}, \dot{\theta} \equiv 0$	$\dot{\psi}$ = turn rate
Steady Pull-Up	$\phi, \dot{\phi}, \dot{\psi} \equiv 0$	$\dot{\theta}$ = pull-up rate
Steady Roll	$\dot{\theta}, \dot{\psi} \equiv 0$	$\dot{\phi}$ = roll rate

For all flight conditions: $\dot{P}, \dot{Q}, \dot{R}, \dot{U}, \dot{V}, \dot{W}$ (or $\dot{V_T}, \dot{\alpha}, \dot{\beta} \equiv 0$)

`trimfun.m`

A function called by `trim_F16`, that creates the cost function that is to be minimized by Matlab's function `fminsearch`. `fminsearch` tries to find values for the free parameters given by `trim_F16` using an iterative technique that perturbs each free parameter until all of the state derivatives are minimized. The cost function is given in the following equation:

$$cost = 5\dot{h}^2 + W_\phi \dot{\phi}^2 + W_\theta \dot{\theta}^2 + W_\psi \dot{\psi}^2 + 2\dot{v_{tot}}^2 + 10\dot{\alpha}^2 + 10\dot{\beta}^2 + 10\dot{p}^2 + 10\dot{q}^2 + 10\dot{r}^2 \tag{1}$$

where:

W_ϕ = weight of ϕ in cost function.
W_θ = weight of θ in cost function.
W_ψ = weight of ψ in cost function.

W_ϕ, W_θ, W_ψ will have values of 10 for steady level flight. But,
$W_\phi = 0$, when trimming for steady roll flight.
$W_\theta = 0$, when trimming steady for pull-up flight.
$W_\psi = 0$, when trim-min for steady turning flight.

`nlplant.c`

This script C-file, the heart of the F-16 simulation, contains all of the equations that govern the dynamics of the F-16 (ie., the equations of motion, navigation equations, etc).

`lofi_F16_AeroData.c`

Contains all of the functions called by `nlplant.c` to determine the aerodynamic data from it tabular form for the low fidelity model.

`hifi_F16_AeroData.c`

Contains all of the functions called by `nlplant.c` to determine the aerodynamic data from it tabular form for the high fidelity model.

`mexndinterp.c`

Creates the C functions such as `interpn` and `linearInterpolate` that allows the script file `nlplant.c` to quickly interpolate the low and high fidelity aerodynamic tables.

`LIN_F16Block.mdl`

Simulink model for the low and high fidelity model used by `runLINF16` to linearize the model. This model contains saturation blocks that may affect the linearization process.

`runLINF16sim.m`

To run, type `runLINF16sim` at Matlab's command line.

```
>> runLINF16sim
```

What the program will do:

When executed `runLINF16sim.m` will linearize the high fidelity and low fidelity models using the Simulink model, `LIN_F16Block.mdl`. Then in a similar fashion as `runF16sim` it will then prompt the user to enter the trim conditions (altitude and velocity) for the F-16 simulation. The user can create disturbances on the any of the control surfaces. To see the trimmed behavior of the answer no ('n') otherwise enter a deflection, in degrees, at the appropriate prompt for each control surface. The simulation will be carried out for the linear model and the results will be saved to a text file. The function will look for any files that have been created at this flight condition and plot all of the available results, both linear and nonlinear, on a single plot.

`FindF16Dynamics.m`

When executed `FindF16Dynamics.m` will linearize both the high fidelity and low fidelity models, creating a linear time-invariant (LTI) model for the F-16 aircraft. It will then break up the LTI model into the longitudinal and lateral directional modes, find the poles and corresponding damping ratio and natural frequency for each mode, create pole-zero map, and create a Bode plot of each of the states with respect to a given control surface and thrust. Requires: μ-Synthesis Toolbox.

If after you have used the F-16 Simulation, you develop a program that you find to be particularly useful and you think it should be included with the download. Please send the program with a description to `F16Sim@aem.umn.edu` and we will try to include it.

5 Using the Non-linear F-16 Model:

The F-16 Model can have a lot of uses. It can be used for educational purposes to teach basic concepts of aircraft simulation and controller design.

Current uses for the non-linear F-16 plant included with this download include:

1. Use of the non-linear model directly to see the response of the F-16 to different control inputs.
2. Linearizing the non-linear F-16 plant so that classical and modern methods of control design can be applied. These methods include: root-locus, Bode plots, etc.
3. Linearizing the non-linear F-16 plant and use it to analyze the longitudinal and lateral directional dynamics of the F-16.

Please let us know what you have used the F-16 Model for by sending an email to `F16Sim@aem.umn.edu`.

5.1 Using the Non-linear F-16 model directly:

You can run the F-16 Model directly in Simulink or use runF16Sim in Matlab. We will cover how to run the simulation through use of the runF16Sim command. But once you are familiar with how runF16Sim works, it will not be too hard to run the model from Simulink.

Using runF16Sim:

1. At Matlab's command-line type, runF16Sim.

   ```
   >> runF16Sim
   ```

2. The program will prompt you enter which model you would like to use in the simulation. Please select the model by entering a 1 or 2 at the appropriate prompt. If you need help selecting which one to choose please see the Section 2.6.

   For your first time: Please run the low fidelity simulation by entering 1. This will help you to feel more comfortable with the program interface and it results. You can also use it to verify that the program works correctly on you computer.

3. The program will prompt you to choose an altitude and velocity. The program will then trim the F-16 at this flight condition. Before you enter the trim condition the program will display acceptable values for all of the simulation parameters. The program give the following table of acceptable parameters as shown below:

   ```
   Accpetable values for flight condition parameters are:


                                   Model
   Variable              LOFI              HIFI
               Units   Min     Max      Min     Max
   Altitude:   ft       5000    40000    5000    40000
   AOA         deg    -10      45      -10      90
   Thrust      lbs      1000    19000    1000    19000
   Elevator    deg    -25.0    25.0    -25.0    25.0
   ```

```
Aileron     deg    -21.5    21.5   -21.5    21.5
Rudder      deg    -30      30     -30      30
Velocity    ft/s    300     900     300     900

The flight condition you choose will be used to trim the F16.
Note:  The trim routine will trim to the desired
altitude and velocity.  All other parameters
will be varied until level flight is achieved.
You may need to view the results of the simulation
and retrim accordingly.
```

For your first time: Use values of 15,000 ft for the altitude and 500 ft/s for the velocity.

```
Enter the altitude for the simulation (ft):  15000
Enter the velocity for the simulation (ft/s):  500
```

4. The program will ask you if you would like to create a disturbance on any of the control surfaces. The control surfaces include: the elevator, ailerons and rudder. If you would like to see the trimmed behavior of the F-16 at the flight condition you have chosen simply type 'n'. Otherwise, answer 'y' to create a disturbance on any or all of the control surfaces. Deflection of the control surfaces are measured in degrees. For a sign convention of the control surfaces please see Section 2.4. The default disturbance is a doublet added to the trim deflection of the control surface. Please see the Simulink model, `F16Block` for further details or to change the type of disturbance.

   For your first time: Enter a 5 degree elevator deflection as shown below.

```
Would you like to create a disturbance on a surface (y/n): y
Enter the elevator disturbance deflection (deg)  : 5
Enter the aileron  disturbance deflection (deg)  : 0
Enter the rudder  disturbance deflection  (deg)  : 0
```

5. The program will then execute the function `trim_F16`. You will be prompted by `trim_F16` to enter the desired steady-state flight condition.

```
At what flight condition would you like to trim the F-16?
1.  Steady Wings-Level Flight.
2.  Steady Turning Flight.
3.  Steady Pull-Up Flight.
4.  Steady Roll Flight.
Your Selection:
```

   For your first time: Trim the F-16 at steady wings-level flight by entering 1.

   `trim_F16` tries to find an equilibrium point by minimizing the cost function described by Equation 1. It does this by varying the thrust and the different control surfaces until the derivatives are zero. After iterating a few times the trim routine will exit because the maximum number of iterations has been reached or the cost has converged. Upon exiting, it will show you its current values for the cost, controls and angle of attack. The results will be displayed as shown below:

```
Trim Values and Cost:
cost   = XXXe-XX
thrust = XX.XX lb
elev   = X.XXX deg
ail    = X.XXX deg
rud    = X.XXX deg
alpha  = X.XXX deg
dLEF   = X.XXX deg
Vel.   = XXX  ft/s
Continue trim rountine iterations? (y/n):
```

It is at this point you must decide whether or not to continue iterating. What are you looking to do? You would like to minimize the cost function. If you achieve values for the cost on the order of `10e-29` and `10e-6` for the low and high fidelity model, respectively, you can consider the F-16 trim. Otherwise you must continue iterating until the cost converges to a tolerable value. If the cost does not converge then the initial conditions need to be adjusted. It usually takes two iterations for an appropriate trim setting to be found. Enter 'y' to continue iterating or 'n' when the results are satisfactory.

For your first time: Confirm that your results are similar to those shown below then enter 'y' Note: These results were obtained using Matlab 6.5 running Linux. Results may vary from platform to platform, but they should still be very similar.

```
Trim Values and Cost:
cost   = 5.5386e-29
thrust = 2120.6214 lb
elev   = -2.4607 deg
ail    = 1.8644e-15 deg
rud    = 3.6479e-14 deg
alpha  = 4.4655 deg
dLEF   = 0 deg
Vel.   = 500ft/s
Continue trim rountine iterations? (y/n):  y
```

You should now see:

```
Trim Values and Cost:
cost   = 5.444e-29
thrust = 2120.6214 lb
elev   = -2.4607 deg
ail    = 1.9576e-15 deg
rud    = 3.6479e-14 deg
alpha  = 4.4655 deg
dLEF   = 0 deg
Vel.   = 500ft/s
Continue trim rountine iterations? (y/n):
```

Notice that the cost function did not change, this means that the trim settings have been found.

Note: When using the low fidelity model the cost function will continue to decrease until the aileron and rudder settings have reach zero. This occurs due to the lack of coupling between the longitudinal and lateral modes of this model. Thus, if a reasonable value of the cost function has been reached simply answer 'n'. The results will not be affected.

So, enter 'n' at the prompt.

```
cost   = 5.444e-29
thrust = 2120.6214 lb
elev   = -2.4607 deg
ail    = 1.9576e-15 deg
rud    = 3.6479e-14 deg
alpha  = 4.4655 deg
dLEF   = 0 deg
Vel.   = 500ft/s
Continue trim rountine iterations? (y/n): n
```

6. The simulation is now running.

7. The simulation will ask if you would like the results to be plotted. **For your first time:** Answer yes by typing y at the prompt.

8. The simulation should produce the results shown in the Figures 5 - 8 All plots can be found in the Appendix in a much larger format. To get the results for both the lofi and the hifi model follows steps 1-7 using the hifi model. This means at step 2 choose the hifi model by entering 2 instead of 1.

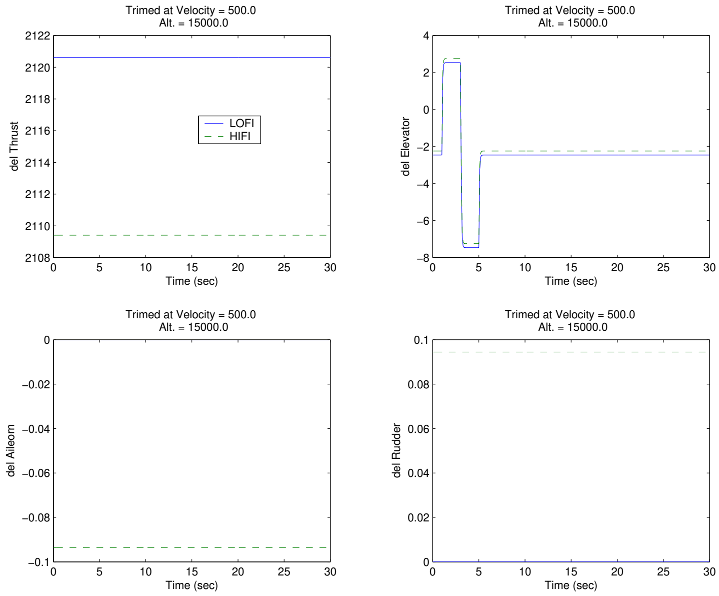


Figure 5: The control input, a 5 degree elevator doublet, to the low fidelity model (solid blue line) and the high fidelity (dashed green line) F-16 model at trim altitude of 15,000 ft and trim velocity of 500 ft/s.

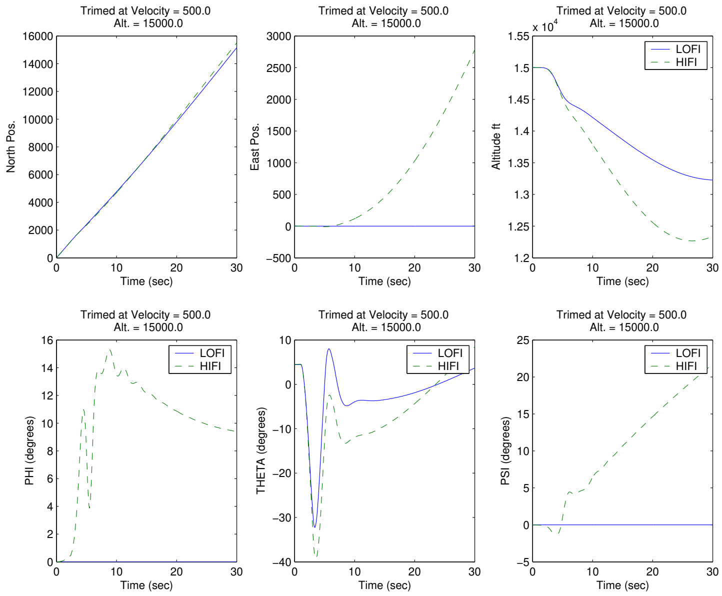


Figure 6: The low fidelity model (solid blue line) and the high fidelity (dashed green line) F-16 model response to a 5 degree elevator doublet at 15,000 ft and velocity of 500 ft/s.

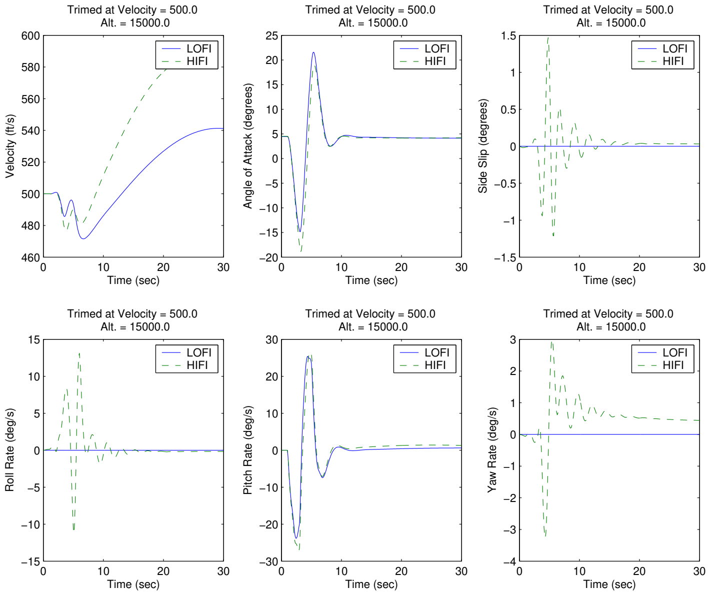


Figure 7: The low fidelity model (solid blue line) and the high fidelity (dashed green line) F-16 model response to a 5 degree elevator doublet at 15,000 ft and velocity of 500 ft/s. Continued.

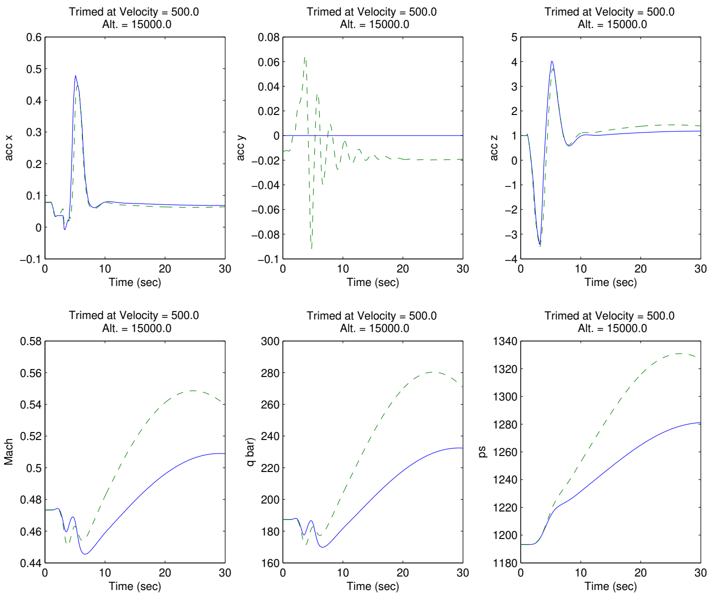


Figure 8: The low fidelity model (solid blue line) and the high fidelity (dashed green line) F-16 model response to a 5 degree elevator doublet at 15,000 ft and velocity of 500 ft/s. Continued.

5.2 Linearizing the non-linear F-16 plant

The non-linear F-16 plant can be linearized. The linearized model can be used in simulation or be used for control design. A linearized model allows classical and modern methods of control design to be applied. `runLINF16Sim` will linearize the F-16 model and write the state-space matrices to a file for later use. It will also run a simulation using the linearized model. If non-linear simulation files exist then `runLINF16Sim` will graph all of the results on one plot.

Using `runLINF16Sim`:

1. At Matlab's command-line type, runF16Sim.

   ```
   >> runLINF16Sim
   ```

2. The program will prompt you to choose an altitude and velocity that the simulation to begin. It will then trim the F-16 at this condition.

   ```
   Accpetable values for flight condition parameters are:


                                       Model
   Variable                  LOFI             HIFI
               Units   Min      Max      Min      Max
   Altitude:   ft       5000    40000    5000     40000
   AOA         deg     -10      45      -10       90
   Thrust      lbs      1000    19000    1000     19000
   Elevator    deg     -25.0    25.0    -25.0     25.0
   Aileron     deg     -21.5    21.5    -21.5     21.5
   Rudder      deg     -30      30      -30       30
   Velocity    ft/s     300     900      300      900
   ```

   For your first time: Use values of 15,000 ft for the altitude and 500 ft/s for the velocity.

   ```
   Enter the altitude for the simulation (ft)  :  15000
   Enter the velocity for the simulation (ft/s):  500
   ```

3. As with `runF16Sim`, `runLINF16Sim` will then execute the function `trim_F16` for both the low fidelity and high fidelity models. First, you will be prompted by `trim_F16` to enter the desired steady-state flight condition for the high fidelity model.

   ```
   Trimming High Fidelity Model:
   At what flight condition would you like to trim the F-16?
   1.  Steady Wings-Level Flight.
   2.  Steady Turning Flight.
   3.  Steady Pull-Up Flight.
   4.  Steady Roll Flight.
   Your Selection:
   ```

For your first time: Trim the F-16 at steady wings-level flight by entering 1 for both the low and high fidelity trim conditions.

This program tries to find an equilibrium point by minimizing the cost function. The cost function is minimized by varying the thrust and the different control surfaces until the derivatives have gone to zero. After iterating a few times the trim routine will show you where it is at by displaying the following results followed by a prompt of whether or not to continue:

```
Trim Values and Cost:
cost   = XXXe-XX
thrust = XX.XX lb
elev   = X.XXX deg
ail    = X.XXX deg
rud    = X.XXX deg
alpha  = X.XXX deg
dLEF   = X.XXX deg
Vel.   = XXX   ft/s
Continue trim rountine iterations? (y/n):
```

What are you looking for at this point? You would like to minimize the cost function. Values on the order of `10e-29` and `10e-6` are common for the low and high fidelity model. It usually takes tow iterations for an appropriate trim setting to be found. Enter 'y' to continue iterating or 'n' when the results are satisfactory.

For your first time: Follow the same instructions of `runF16Sim` instruction set 5 for both the high fidelity and low fidelity model.

4. The program will ask you if you would like to create a disturbance on any of the control surfaces. The control surfaces include: the elevator, ailerons and rudder. If you like to see the trimmed behavior of the F-16 at the flight condition you have chosen simply type 'n'. Otherwise, answer 'y' to create a disturbance on any or all of the control surfaces. The disturbance is a doublet added to the trim deflection of the control surface. Please see the Simulink model, `SS_F16Block`. For your first run through enter the following:

```
Would you like to create a disturbance on a surface (y/n): y
Enter the elevator disturbance deflection     (deg): 5
Enter the aileron disturbance deflection       (deg): 0
Enter the rudder disturbance deflection        (deg): 0
```

5. The simulation is now running.

6. The simulation will ask if you would like the results to be plotted. **For your first time:** Answer yes by typing y at the prompt.

7. The simulation should produce the results shown in Figure 9 - Figure 12.

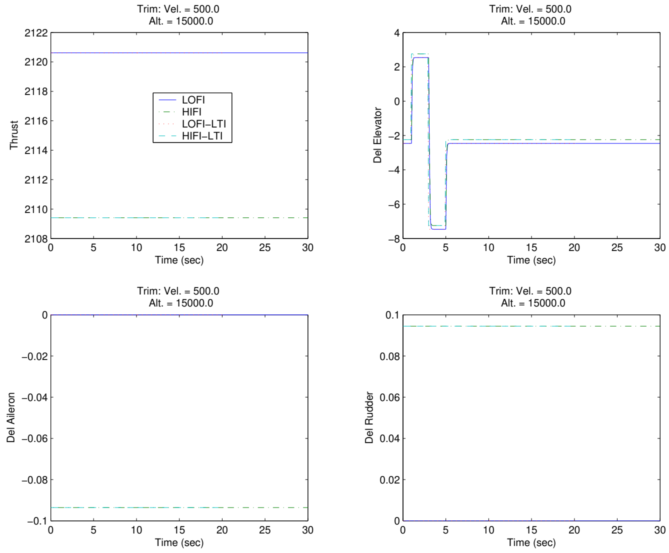


Figure 9: The control input, a 5 degree elevator doublet, to the linearized F-16 model at 15,000 ft and velocity of 500 ft/s.

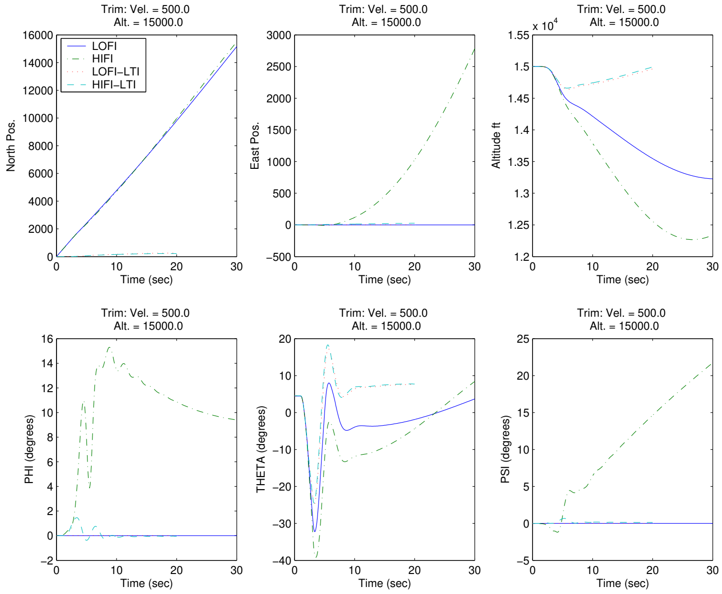


Figure 10: Comparison of linearized F-16 model response (low and high fidelity) to low fidelity model response when a 5 degree doublet is applied to the elevator at 15,000 ft and velocity of 500 ft/s.

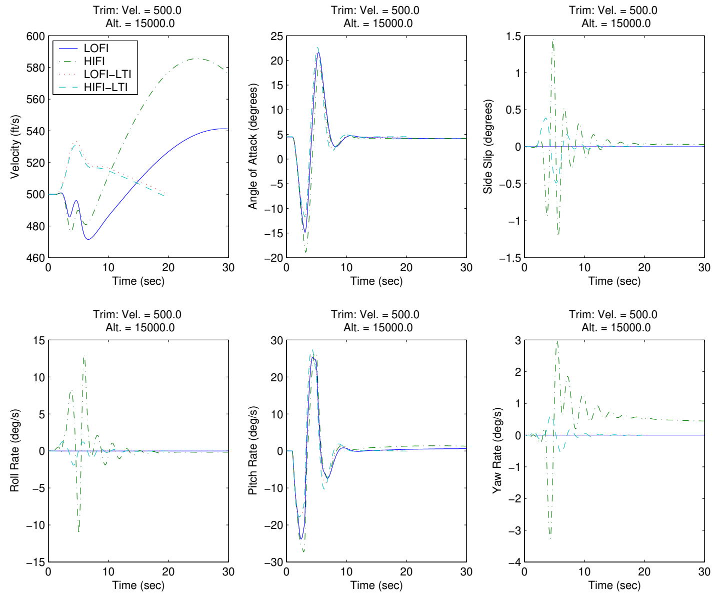


Figure 11: Comparison of linearized F-16 model response (low and high fidelity) to low fidelity model response when a 5 degree doublet is applied to the elevator at 15,000 ft and velocity of 500 ft/s. Continued.

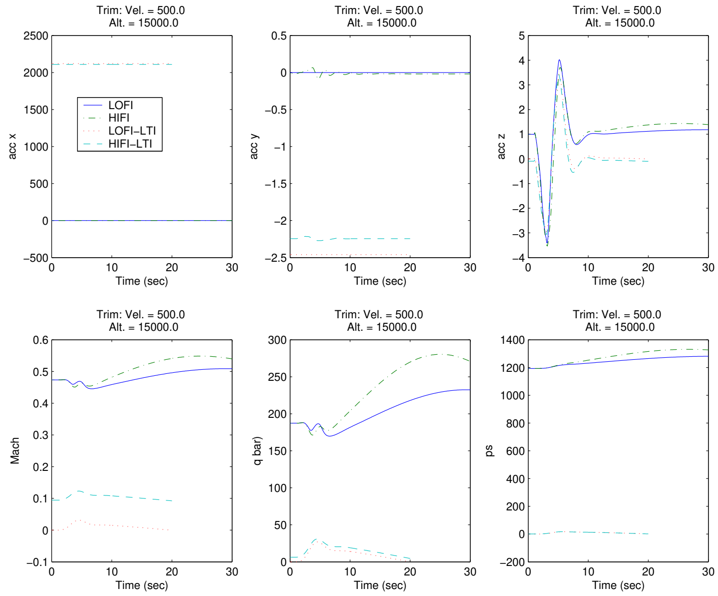


Figure 12: Comparison of linearized F-16 model response (low and high fidelity) to low fidelity model response when a 5 degree doublet is applied to the elevator at 15,000 ft and velocity of 500 ft/s. Continued.

5.3 Analyzing the longitudinal and lateral directional modes of the F16.

`FindF16Dynamics` will linearize the entire F-16 model, it will then extract the longitudinal and lateral directional modes. `FindF16Dynamics` will then give a pole-zero mapping of the longitudinal and lateral directional state space models. Bode plots of each state to a control surface will also be displayed.

Using FindF16Dynamics:

1. At the command-line type:

```
>> FindF16Dynamics
```

2. You will be asked to enter an altitude and velocity for which you would like the non-linear model to be linearized about.

   For your first time: Use an altitude of 15,000 ft and a velocity of 500 ft/s.

```
Enter the altitude for the simulation (ft)  :  15000
Enter the velocity for the simulation (ft/s):  500
```

3. You will then have to trim the aircraft in a similar fashion as for `runF16Sim` and `runLINF16`, for both the low and high fidelity models. Please consult instruction set 5 of `runF16Sim`.

4. `FindF16Dynamics` will then output the linear time invariant state-space matrices for the high fidelity model.

   `FindF16Dynamics` will give the state-space matrices (**A**,**B**, **C** and **D**) for both the longitudinal and lateral directions. The states for the longitudinal modes are altitude, pitch angle, magnitude of the total velocity, angle of attack and pitch rate. The control inputs are thrust and elevator deflection.

   Units: For the following state-space matrices, all angles and rotations are given in degrees. All measures for distance are given in feet and time is given in seconds.

   The longitudinal state-space matrices, given by `FindF16Dynamics`, will be of the form:

$$\begin{bmatrix} \dot{h} \\ \dot{\theta} \\ \dot{v} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\delta}_t \\ \dot{\delta}_e \end{bmatrix} = \mathbf{A} \begin{bmatrix} h \\ \theta \\ v \\ \alpha \\ q \\ \delta_t \\ \delta_e \end{bmatrix} + \mathbf{B} \begin{bmatrix} \delta_t \\ \delta_e \end{bmatrix}$$

$$\begin{bmatrix} h \\ \theta \\ v \\ \alpha \\ q \end{bmatrix} = \mathbf{C} \begin{bmatrix} h \\ \theta \\ v \\ \alpha \\ q \end{bmatrix} + \mathbf{D} \begin{bmatrix} \delta_t \\ \delta_e \end{bmatrix}$$

The lateral directional states are roll angle, yaw angle, magnitude of the total velocity, side-slip angle, roll rate and yaw rate. The inputs for this direction are thrust, aileron deflection and rudder deflection.

The lateral state-space matrices (**A**,**B**, **C** and **D**) will be of the form:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\psi} \\ \dot{v} \\ \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\delta}_t \\ \dot{\delta}_a \\ \dot{\delta}_r \end{bmatrix} = \mathbf{A} \begin{bmatrix} \phi \\ \psi \\ v \\ \beta \\ p \\ r \\ \delta_t \\ \delta_a \\ \delta_r \end{bmatrix} + \mathbf{B} \begin{bmatrix} \delta_t \\ \delta_a \\ \delta_r \end{bmatrix}$$

$$\begin{bmatrix} \phi \\ \psi \\ v \\ \beta \\ p \\ r \end{bmatrix} = \mathbf{C} \begin{bmatrix} \phi \\ \psi \\ v \\ \beta \\ p \\ r \end{bmatrix} + \mathbf{D} \begin{bmatrix} \delta_t \\ \delta_a \\ \delta_r \end{bmatrix}$$

The results shown below are for an altitude of 15000 ft and a velocity of 500 ft/s:

```
For HIFI Model:
Longitudal Direction:


A =
   0.000e+00    5.000e+02    3.553e-10   -5.000e+02    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00
   1.074e-04   -3.217e+01   -1.321e-02   -2.669e+00   -1.186e+00    1.565e-03    3.870e-02
   2.076e-06   -3.681e-13   -2.552e-04   -6.761e-01    9.392e-01   -2.480e-07   -1.437e-03
   9.632e-12    0.000e+00   -1.184e-09   -5.757e-01   -8.741e-01    0.000e+00   -1.188e-01
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -1.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01
B =
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   1.000e-00    0.000e+00
   0.000e+00    2.020e+01
C =
   1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00
D =
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00

       real       imaginary     frequency       damping
```

```
  1.4544e-07    0.0000e+00    1.4544e-07   -1.0000e+00
 -3.5809e-03   -6.8349e-02    6.8442e-02    5.2320e-02
 -3.5809e-03    6.8349e-02    6.8442e-02    5.2320e-02
 -1.0000e+00    0.0000e+00    1.0000e+00    1.0000e+00
 -7.7811e-01   -7.2880e-01    1.0661e+00    7.2986e-01
 -7.7811e-01    7.2880e-01    1.0661e+00    7.2986e-01
 -2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00
```

Lateral Direaction:

```
A =
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.000e-00    7.924e-02    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.003e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00   -1.321e-02   -2.872e-01    0.000e+00    0.000e+00    1.565e-03    0.000e+00    0.000e+00
   6.414e-02    0.000e+00   -1.695e-06   -1.868e-01    7.946e-02   -9.918e-01    0.000e+00    2.026e-04    5.394e-04
   0.000e+00    0.000e+00    2.722e-10   -2.277e+01   -2.236e+00    5.459e-01    0.000e+00   -4.310e-01    8.115e-02
   0.000e+00    0.000e+00   -4.475e-08    4.854e+00   -4.181e-02   -3.146e-01    0.000e+00   -1.797e-02   -4.036e-02
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -1.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01
B =
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   1.000e-00    0.000e+00    0.000e+00
   0.000e+00    2.020e+01    0.000e+00
   0.000e+00    0.000e+00    2.020e+01
C =
   5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00
D =
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00

      real       imaginary     frequency      damping

 -3.9248e-16    0.0000e+00    3.9248e-16    1.0000e+00
 -1.3206e-02    0.0000e+00    1.3206e-02    1.0000e+00
 -1.6981e-02    0.0000e+00    1.6981e-02    1.0000e+00
 -1.0000e+00    0.0000e+00    1.0000e+00    1.0000e+00
 -2.0905e+00    0.0000e+00    2.0905e+00    1.0000e+00
 -3.1526e-01   -2.5225e+00    2.5422e+00    1.2401e-01
 -3.1526e-01    2.5225e+00    2.5422e+00    1.2401e-01
 -2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00
 -2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00
```

For LOFI Model:
Longitudal Direction:

```
A =
   0.000e+00    5.000e+02    3.553e-10   -5.000e+02    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00
   1.080e-04   -3.217e+01   -1.328e-02   -7.326e+00   -1.196e+00    1.565e-03    7.397e-02
   2.076e-06    2.547e-13   -2.553e-04   -6.398e-01    9.378e-01   -2.445e-07   -1.357e-03
   2.573e-20    0.000e+00   -3.163e-18   -1.568e+00   -8.791e-01    0.000e+00   -1.137e-01
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -1.000e+00    0.000e+00
```

```
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01
B =
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   1.000e-00    0.000e+00
   0.000e+00    2.020e+01
C =
   1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00
D =
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00
   0.000e+00    0.000e+00

      real       imaginary     frequency      damping

  -3.1073e-10    0.0000e+00    3.1073e-10    1.0000e+00
  -3.9011e-03   -8.4374e-02    8.4464e-02    4.6187e-02
  -3.9011e-03    8.4374e-02    8.4464e-02    4.6187e-02
  -1.0000e+00    0.0000e+00    1.0000e+00    1.0000e+00
  -7.6215e-01    1.2051e+00    1.4259e+00    5.3451e-01
  -7.6215e-01   -1.2051e+00    1.4259e+00    5.3451e-01
  -2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00


Lateral Direaction:


A =
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.000e-00    7.810e-02    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    1.003e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00   -1.328e-02    9.403e-15    0.000e+00    0.000e+00    1.565e-03    0.000e+00    0.000e+00
   6.414e-02    0.000e+00    3.761e-20   -2.022e-01    7.827e-02   -9.919e-01    0.000e+00    1.724e-04    5.058e-04
   0.000e+00    0.000e+00    4.496e-18   -2.292e+01   -2.254e+00    5.408e-01    0.000e+00   -4.623e-01    5.686e-02
   0.000e+00    0.000e+00   -7.040e-18    6.005e+00   -4.040e-02   -3.146e-01    0.000e+00   -2.437e-02   -4.687e-02
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -1.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00   -2.020e+01
B =
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   1.000e-00    0.000e+00    0.000e+00
   0.000e+00    2.020e+01    0.000e+00
   0.000e+00    0.000e+00    2.020e+01
C =
   5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    1.000e-00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00    0.000e+00    0.000e+00    5.730e+01    0.000e+00    0.000e+00    0.000e+00
D =
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00
   0.000e+00    0.000e+00    0.000e+00

      real       imaginary     frequency      damping
```

```
 1.7768e-15    0.0000e+00    1.7768e-15   -1.0000e+00
-1.1264e-02    0.0000e+00    1.1264e-02    1.0000e+00
-1.3277e-02    0.0000e+00    1.3277e-02    1.0000e+00
-1.0000e+00    0.0000e+00    1.0000e+00    1.0000e+00
-2.1202e+00    0.0000e+00    2.1202e+00    1.0000e+00
-3.1981e-01   -2.7408e+00    2.7594e+00    1.1590e-01
-3.1981e-01    2.7408e+00    2.7594e+00    1.1590e-01
-2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00
-2.0200e+01    0.0000e+00    2.0200e+01    1.0000e+00
```

5. A pole-zero mapping will then be displayed.

   The bode plots displayed by `FindF16Dynamics` will displayed in the following order. It will display the longitudinal Bode plots first beginning with the first state, h, with input to thrust, δ_t. Then it will show the Bode plot for pitch angle, velocity, and so on for thrust input. It will then use the same order for the states to show the Bode plots with the elevator deflection as the input to the system. The lateral directional Bode plot will be displayed in a similar fashion beginning the roll angle with thrust input and ending with roll rate with rudder input. Consult the Table 5 for clarification.

Plot #	State	Input
1	h	δ_t
2	h	δ_e
3	θ	δ_t
4	θ	δ_e
5	v	δ_t
6	v	δ_e
7	α	δ_t
8	α	δ_e
9	q	δ_t
10	q	δ_e

Table 5. Longitudinal Bode Plots.

Plot #	State	Input
11	ϕ	δ_t
12	ϕ	δ_a
13	ϕ	δ_r
14	ψ	δ_t
15	ψ	δ_a
16	ψ	δ_r
17	v	δ_t
18	v	δ_a
19	v	δ_r
20	β	δ_t
21	β	δ_a
22	β	δ_r
23	p	δ_t
24	p	δ_a
25	p	δ_r
26	r	δ_t
27	r	δ_a
28	r	δ_r

Table 5. Lateral Bode Plots.

The pole-zero mapping for an altitude of 15000 ft and a velocity of 500 ft/s is shown in Figure 13 - Figure 15.

6. Finally, the Bode plots will be displayed, as shown in Figure 6. After you have analyzed the results hit enter and a new Bode plot will be displayed. The Bode plots will be displayed in order by state versus the control input.

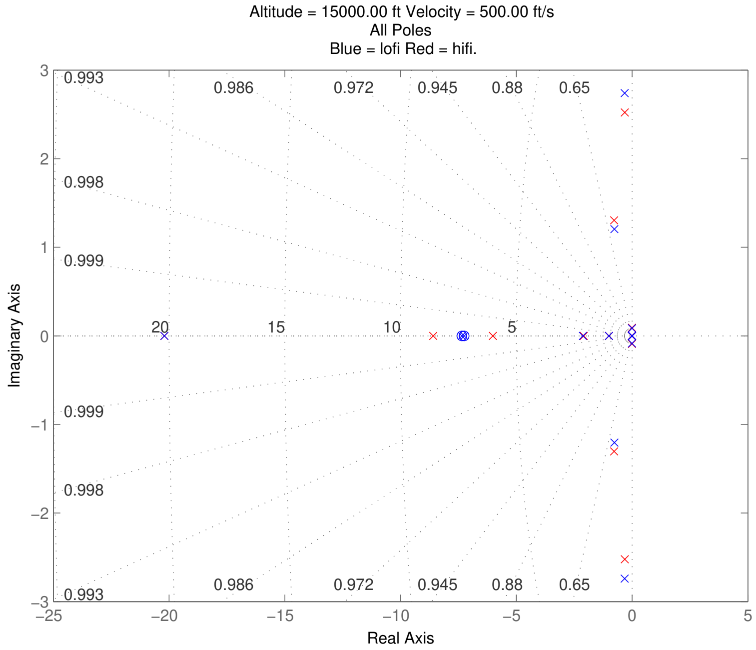


Figure 13: Pole-zero mapping of the entire F-16.

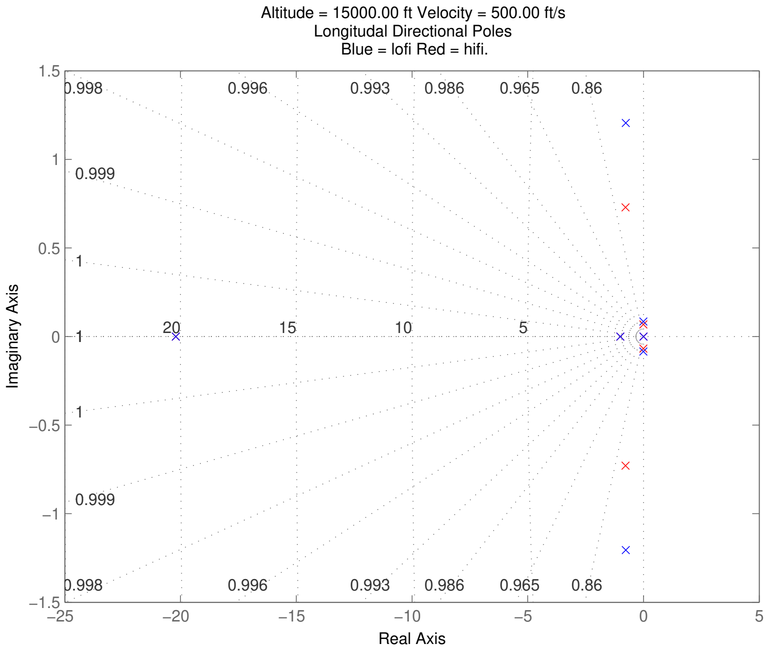


Figure 14: Pole-zero mapping of the longitudinal direction of the F-16.

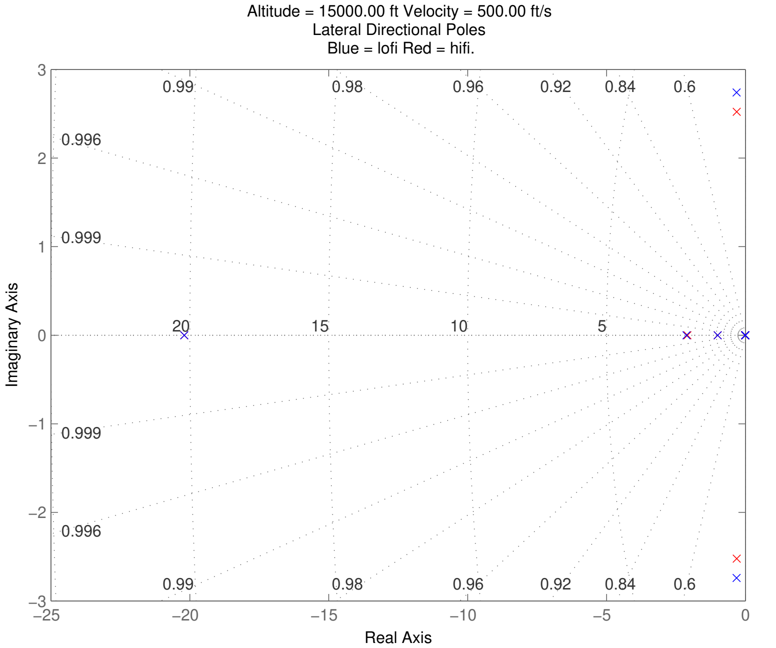


Figure 15: Pole-zero mapping of the lateral direction of the F-16.

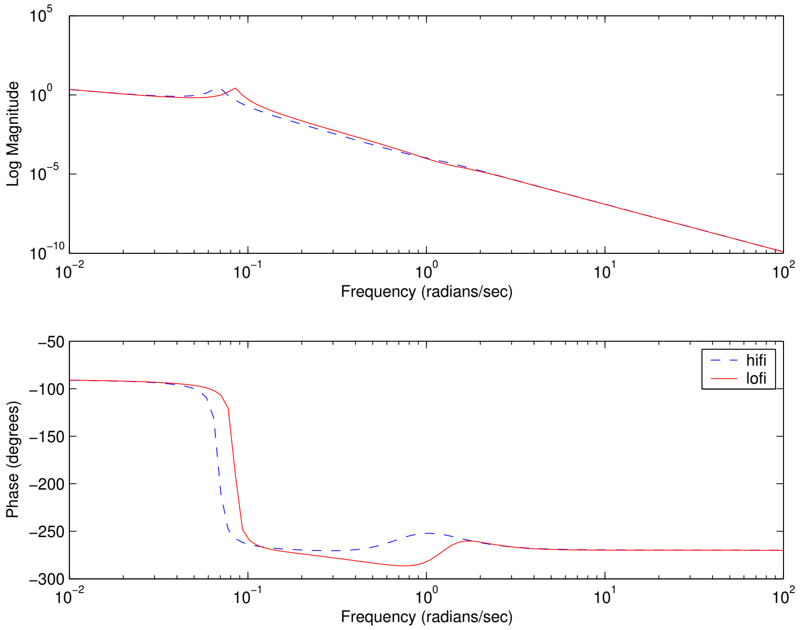


Figure 16: The first Bode plot displayed when FindF16Dynamics is run at an altitude of 15,000 ft. and a velocity of 500 ft/s. This shows the frequency response of the the altitude of the F-16 to thrust.

6 Contributors

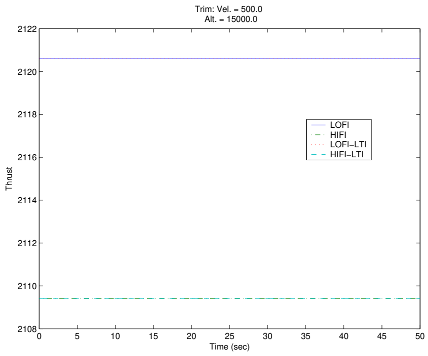
This model has been in the works for quite some time. Not because of its difficulty to construct but because this has been a project under taken by students in the Aerospace Engineering and Control Science and Dynamical Systems Departments at the University of Minnesota in their spare time. Each part was constructed under the supervision of Dr. Gary Balas. The models have been tested for there consistency and we are very pleased with the results.

Dr. Gary Balas	balas@aem.umn.edu
Zachary William Jarvis-Wloszek	zachary@newton.Berkeley.edu
Raktim Bhattacharya	raktim@aem.umn.edu
Aaron Cooper	coop0172@umn.edu
Subhabrata Ganguli	suvo@aem.umn.edu
Tamas Keviczky	keviczky@aem.umn.edu
Andres Marcos	marcosa@aem.umn.edu
Brett Otterson	otteson@princeton.edu OR botteson@aem.umn.edu
Richard Russell	russ1110@aem.umn.edu

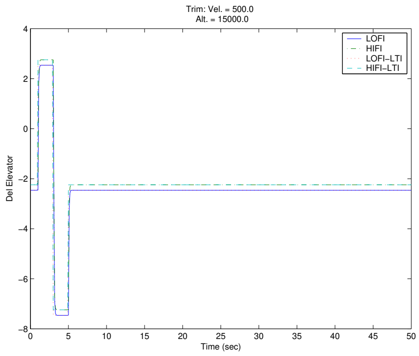
A Appendix

A.1 Response of F-16 Models to 5-deg. elevator doublet.

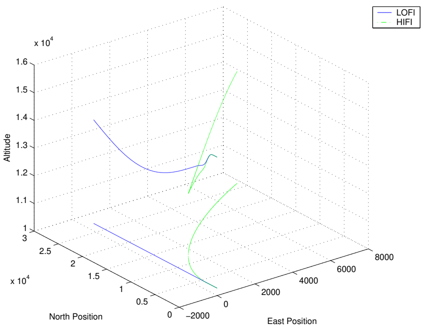
The response of the non-linear and linearized low and high fidelity F-16 models to an 5-deg elevator doublet. The results have been enlarged so that you can compare validity of your plots and reaffirm that the programs are working correctly.



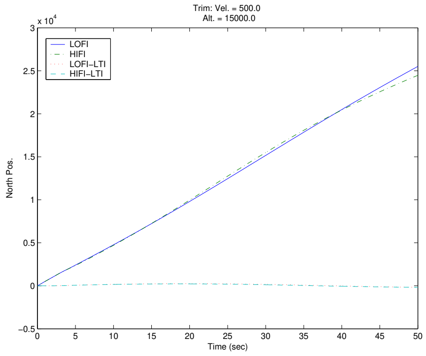
(a) Thrust input.



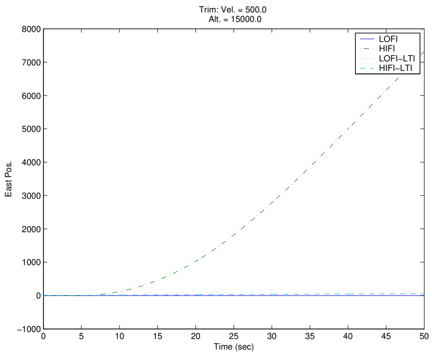
(b) 5-deg. Elevator doublet



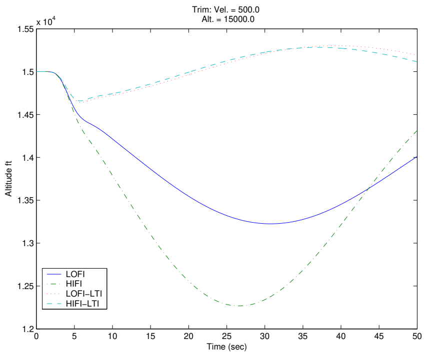
(c) F-16 Trajectory



(d) North Position

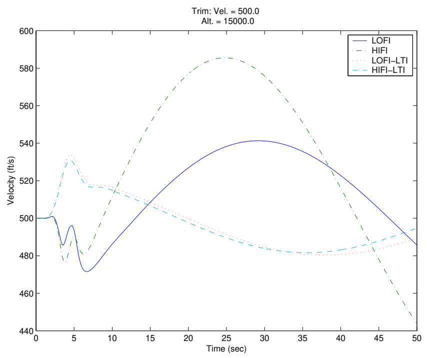


(e) East Position

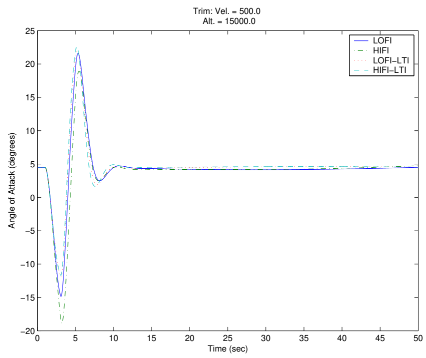


(f) Altitude

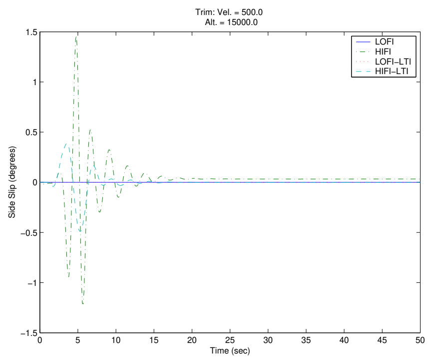
Figure 17: Response of non-linear and linearized low fidelity and high fidelity F-16 Models to an 5-deg elevator doublet.



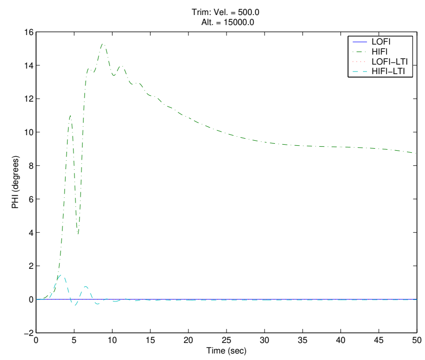
(a) Velocity.



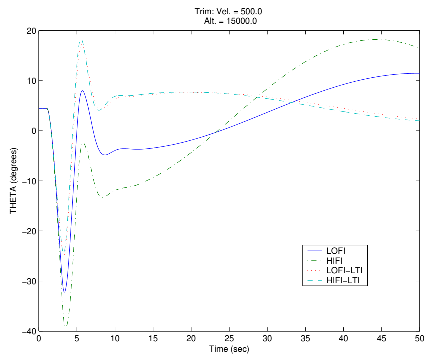
(b) Angle of attack



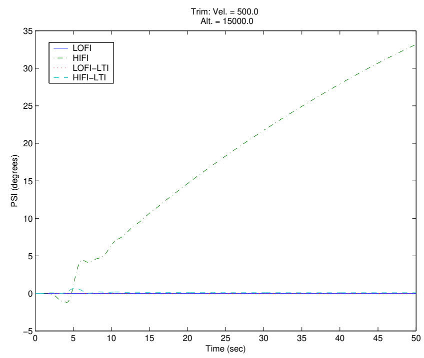
(c) Side-slip



(d) Roll Angle, Φ.

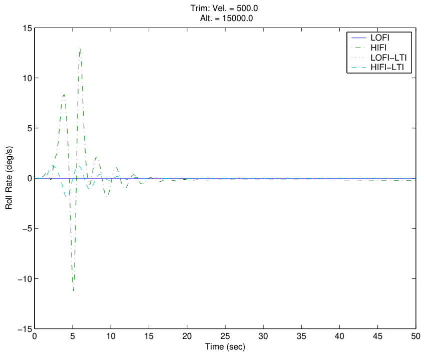


(e) Pitch Angle, θ.

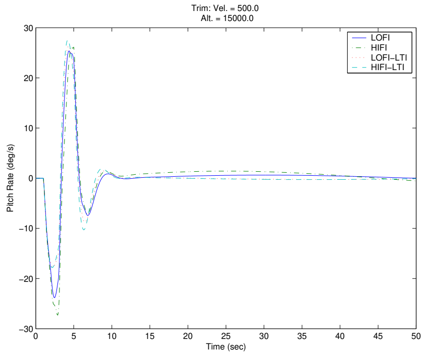


(f) Yaw Angle, ψ.

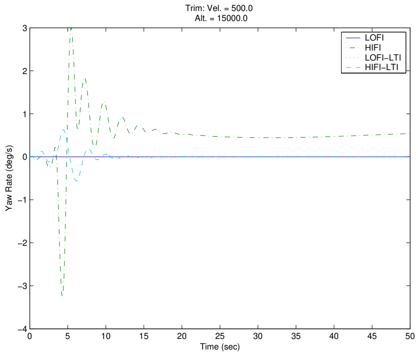
Figure 18: Response of non-linear and linearized low fidelity and high fidelity F-16 Models to an 5-deg elevator doublet. Continued.



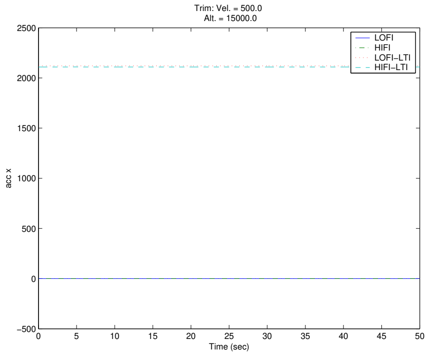
(a) Roll Rate, P.



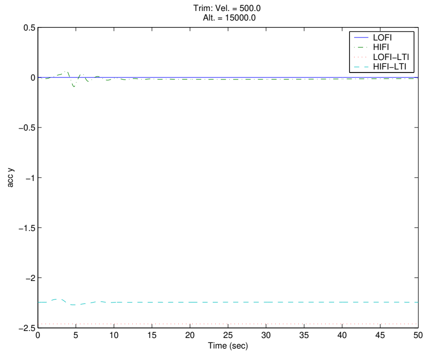
(b) Pitch Rate, Q.



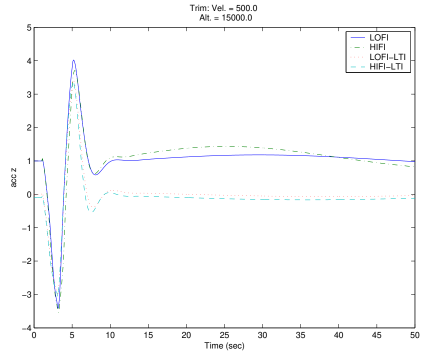
(c) Yaw Rate, R.



(d) X-axis Normal Acceleration

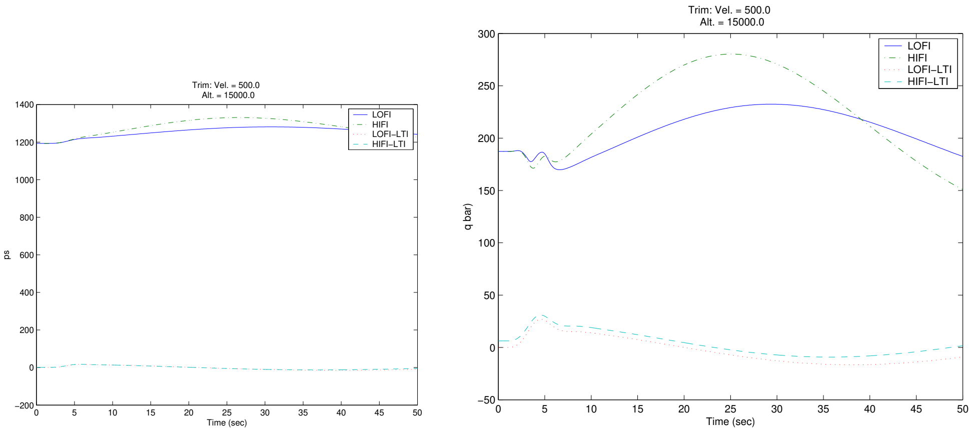


(e) Y-axis Normal Acceleration



(f) Z-axis Normal Acceleration

Figure 19: Response of non-linear and linearized low fidelity and high fidelity F-16 Models to an 5-deg elevator doublet. Continued



(a) Static Pressure

(b) Dynamic Pressure

Figure 20: Response of non-linear and linearized low fidelity and high fidelity F-16 Models to an 5-deg elevator doublet. Continued.

A.2 nlplant.c

```
#include <math.h>

/*  Merging the nlplant.c (lofi) and nlplant_hifi.c to use
    same equations of motion, navigation equations and use
    own look-up tables decided by a flag.                   */

void atmos(double,double,double*);          /* Used by both */
void accels(double*,double*,double*);       /* Used by both */

#include "lofi_F16_AeroData.c"              /* LOFI Look-up header file*/
#include "hifi_F16_AeroData.c"        /* HIFI Look-up header file*/

void nlplant(double*,double*);

/*########################################*/
/*### Added for mex function in matlab ###*/
/*########################################*/

#include "mex.h"

int fix(double);
int sign(double);

void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{

#define XU prhs[0]
#define XDOTY plhs[0]

int i;
double *xup, *xdotp;

if (mxGetM(XU)==18 && mxGetN(XU)==1){

      /* Calling Program */
      xup = mxGetPr(XU);
      XDOTY = mxCreateDoubleMatrix(18, 1, mxREAL);
      xdotp = mxGetPr(XDOTY);

      nlplant(xup,xdotp);

      /* debug
      for (i=0;i<=14;i++){
        printf("xdotp(%d) = %e\n",i+1,xdotp[i]);
      }
      end debug */

} /* End if */
else{
      mexErrMsgTxt("Input and/or output is wrong size.");
} /* End else */

} /* end mexFunction */

/*########################################*/
/*########################################*/

void nlplant(double *xu, double *xdot){

  int fi_flag;
```

```
  /* #include f16_constants */
  double g    = 32.17;          /* gravity, ft/s^2 */
  double m    = 636.94;         /* mass, slugs */
  double B    = 30.0;             /* span, ft */
  double S    = 300.0;            /* planform area, ft^2 */
  double cbar = 11.32;          /* mean aero chord, ft */
  double xcgr = 0.35;      /* reference center of gravity as a fraction of cbar */
  double xcg  = 0.30;      /* center of gravity as a fraction of cbar. */

  double Heng = 0.0;              /* turbine momentum along roll axis. */
  double pi   = acos(-1);
  double r2d;                   /* radians to degrees */


/*NasaData        %translated via eq. 2.4-6 on pg 80 of Stevens and Lewis*/

  double Jy  = 55814.0;           /* slug-ft^2 */
  double Jxz = 982.0;             /* slug-ft^2 */
  double Jz  = 63100.0;           /* slug-ft^2 */
  double Jx  = 9496.0;            /* slug-ft^2 */

  double *temp;

  double npos, epos, alt, phi, theta, psi, vt, alpha, beta, P, Q, R;
  double sa, ca, sb, cb, tb, st, ct, tt, sphi, cphi, spsi, cpsi;
  double T, el, ail, rud, dail, drud, lef, dlef;
  double qbar, mach, ps;
  double U, V, W, Udot,Vdot,Wdot;
  double L_tot, M_tot, N_tot, denom;

  double Cx_tot, Cx, delta_Cx_lef, dXdQ, Cxq, delta_Cxq_lef;
  double Cz_tot, Cz, delta_Cz_lef, dZdQ, Czq, delta_Czq_lef;
  double Cm_tot, Cm, eta_el, delta_Cm_lef, dMdQ, Cmq, delta_Cmq_lef, delta_Cm, delta_Cm_ds;
  double Cy_tot, Cy, delta_Cy_lef, dYdail, delta_Cy_r30, dYdR, dYdP;
  double delta_Cy_a20, delta_Cy_a20_lef, Cyr, delta_Cyr_lef, Cyp, delta_Cyp_lef;
  double Cn_tot, Cn, delta_Cn_lef, dNdail, delta_Cn_r30, dNdR, dNdP, delta_Cnbeta;
  double delta_Cn_a20, delta_Cn_a20_lef, Cnr, delta_Cnr_lef, Cnp, delta_Cnp_lef;
  double Cl_tot, Cl, delta_Cl_lef, dLdail, delta_Cl_r30, dLdR, dLdP, delta_Clbeta;
  double delta_Cl_a20, delta_Cl_a20_lef, Clr, delta_Clr_lef, Clp, delta_Clp_lef;

  temp = (double *)malloc(9*sizeof(double));  /*size of 9.1 array*/

  r2d  = 180.0/pi;     /* radians to degrees */

  /* %%%%%%%%%%%%%%%%%%
          States
     %%%%%%%%%%%%%%%%%% */

  npos  = xu[0];   /* north position */
  epos  = xu[1];   /* east position */
  alt   = xu[2];   /* altitude */
  phi   = xu[3];   /* orientation angles in rad. */
  theta = xu[4];
  psi   = xu[5];

  vt    = xu[6];     /* total velocity */
  alpha = xu[7]*r2d; /* angle of attack in degrees */
  beta  = xu[8]*r2d; /* sideslip angle in degrees */
  P     = xu[9];     /* Roll Rate --- rolling  moment is Lbar */
  Q     = xu[10];    /* Pitch Rate--- pitching moment is M */
```

```
  R     = xu[11];    /* Yaw Rate  --- yawing   moment is N */

  sa    = sin(xu[7]); /* sin(alpha) */
  ca    = cos(xu[7]); /* cos(alpha) */
  sb    = sin(xu[8]); /* sin(beta)  */
  cb    = cos(xu[8]); /* cos(beta)  */
  tb    = tan(xu[8]); /* tan(beta)  */

  st    = sin(theta);
  ct    = cos(theta);
  tt    = tan(theta);
  sphi  = sin(phi);
  cphi  = cos(phi);
  spsi  = sin(psi);
  cpsi  = cos(psi);

  if (vt <= 0.01) {vt = 0.01;}

  /* %%%%%%%%%%%%%%%%%%%
     Control inputs
     %%%%%%%%%%%%%%%%%%% */

  T     = xu[12];   /* thrust */
  el    = xu[13];   /* Elevator setting in degrees. */
  ail   = xu[14];   /* Ailerons setting in degrees. */
  rud   = xu[15];   /* Rudder setting in degrees. */
  lef   = xu[16];   /* Leading edge flap setting in degrees */

  fi_flag = xu[17]/1;       /* fi_flag */

  dail  = ail/20.0;  /* aileron normalized against max angle */
  drud  = rud/30.0;  /* rudder normalized against max angle */
  dlef  = (1 - lef/25.0);  /* leading edge flap normalized against max angle */


  /* %%%%%%%%%%%%%%%%%%
     Atmospheric effects
     sets dynamic pressure and mach number
     %%%%%%%%%%%%%%%%%% */

atmos(alt,vt,temp);
   mach = temp[0];
   qbar = temp[1];
   ps   = temp[2];

/*
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%Dynamics%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*/

  /* %%%%%%%%%%%%%%%%%%
     Navigation Equations
     %%%%%%%%%%%%%%%%%% */

   U = vt*ca*cb;  /* directional velocities. */
   V = vt*sb;
   W = vt*sa*cb;

/* npdosdot */
xdot[0] = U*(ct*cpsi) +
```

```
            V*(sphi*cpsi*st - cphi*spsi) +
            W*(cphi*st*cpsi + sphi*spsi);

/* eposdot */
xdot[1] = U*(ct*spsi) +
            V*(sphi*spsi*st + cphi*cpsi) +
            W*(cphi*st*spsi - sphi*cpsi);

/* altdot */
xdot[2] = U*st - V*(sphi*ct) - W*(cphi*ct);

  /* %%%%%%%%%%%%%%%%%%%
     Kinematic equations
     %%%%%%%%%%%%%%%%%%% */
/* phidot */
xdot[3] = P + tt*(Q*sphi + R*cphi);

/* theta dot */
xdot[4] = Q*cphi - R*sphi;

/* psidot */
xdot[5] = (Q*sphi + R*cphi)/ct;


/* %%%%%%%%%%%%%%%%%%
        Table lookup
     %%%%%%%%%%%%%%%%%% */

if (fi_flag == 1)          /* HIFI Table */
{
    hifi_C(alpha,beta,el,temp);
        Cx = temp[0];
        Cz = temp[1];
        Cm = temp[2];
        Cy = temp[3];
        Cn = temp[4];
        Cl = temp[5];

    hifi_damping(alpha,temp);
        Cxq = temp[0];
        Cyr = temp[1];
        Cyp = temp[2];
        Czq = temp[3];
        Clr = temp[4];
        Clp = temp[5];
        Cmq = temp[6];
        Cnr = temp[7];
        Cnp = temp[8];

    hifi_C_lef(alpha,beta,temp);
        delta_Cx_lef = temp[0];
        delta_Cz_lef = temp[1];
        delta_Cm_lef = temp[2];
        delta_Cy_lef = temp[3];
        delta_Cn_lef = temp[4];
        delta_Cl_lef = temp[5];

    hifi_damping_lef(alpha,temp);
        delta_Cxq_lef = temp[0];
        delta_Cyr_lef = temp[1];
        delta_Cyp_lef = temp[2];
```

```
        delta_Czq_lef = temp[3];
        delta_Clr_lef = temp[4];
        delta_Clp_lef = temp[5];
        delta_Cmq_lef = temp[6];
        delta_Cnr_lef = temp[7];
        delta_Cnp_lef = temp[8];

    hifi_rudder(alpha,beta,temp);
        delta_Cy_r30 = temp[0];
        delta_Cn_r30 = temp[1];
        delta_Cl_r30 = temp[2];

    hifi_ailerons(alpha,beta,temp);
        delta_Cy_a20     = temp[0];
        delta_Cy_a20_lef = temp[1];
        delta_Cn_a20     = temp[2];
        delta_Cn_a20_lef = temp[3];
        delta_Cl_a20     = temp[4];
        delta_Cl_a20_lef = temp[5];

    hifi_other_coeffs(alpha,el,temp);
        delta_Cnbeta = temp[0];
        delta_Clbeta = temp[1];
        delta_Cm     = temp[2];
        eta_el       = temp[3];
        delta_Cm_ds  = 0;        /* ignore deep-stall effect */

}

else if (fi_flag == 0)
{
/* ##############################################
   ##########LOFI Table Look-up #################
   #############################################*/

/* The lofi model does not include the
   leading edge flap.  All terms multiplied
   dlef have been set to zero but just to
   be sure we will set it to zero. */

    dlef = 0.0;

    damping(alpha,temp);
        Cxq = temp[0];
        Cyr = temp[1];
        Cyp = temp[2];
        Czq = temp[3];
        Clr = temp[4];
        Clp = temp[5];
        Cmq = temp[6];
        Cnr = temp[7];
        Cnp = temp[8];

    dmomdcon(alpha,beta, temp);
        delta_Cl_a20 = temp[0];     /* Formerly dLda in nlplant.c */
        delta_Cl_r30 = temp[1];     /* Formerly dLdr in nlplant.c */
        delta_Cn_a20 = temp[2];     /* Formerly dNda in nlplant.c */
        delta_Cn_r30 = temp[3];     /* Formerly dNdr in nlplant.c */

    clcn(alpha,beta,temp);
        Cl = temp[0];
```

```
        Cn = temp[1];

    cxcm(alpha,el,temp);
        Cx = temp[0];
        Cm = temp[1];

    Cy = -.02*beta + .021*dail + .086*drud;

    cz(alpha,beta,el,temp);
        Cz = temp[0];
/*##################################################


/*##################################################
  ##  Set all higher order terms of hifi that are ##
  ##  not applicable to lofi equal to zero. ########
  ##################################################*/

        delta_Cx_lef    = 0.0;
        delta_Cz_lef    = 0.0;
        delta_Cm_lef    = 0.0;
        delta_Cy_lef    = 0.0;
        delta_Cn_lef    = 0.0;
        delta_Cl_lef    = 0.0;
        delta_Cxq_lef   = 0.0;
        delta_Cyr_lef   = 0.0;
        delta_Cyp_lef   = 0.0;
        delta_Czq_lef   = 0.0;
        delta_Clr_lef   = 0.0;
        delta_Clp_lef   = 0.0;
        delta_Cmq_lef   = 0.0;
        delta_Cnr_lef   = 0.0;
        delta_Cnp_lef   = 0.0;
        delta_Cy_r30    = 0.0;
        delta_Cy_a20    = 0.0;
        delta_Cy_a20_lef= 0.0;
        delta_Cn_a20_lef= 0.0;
        delta_Cl_a20_lef= 0.0;
        delta_Cnbeta    = 0.0;
        delta_Clbeta    = 0.0;
        delta_Cm        = 0.0;
        eta_el          = 1.0;     /* Needs to be one. See equation for Cm_tot*/
        delta_Cm_ds     = 0.0;

/*##################################################
  ##################################################*/
}


/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
compute Cx_tot, Cz_tot, Cm_tot, Cy_tot, Cn_tot, and Cl_tot
(as on NASA report p37-40)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* XXXXXXXX Cx_tot XXXXXXXX */

dXdQ = (cbar/(2*vt))*(Cxq + delta_Cxq_lef*dlef);

Cx_tot = Cx + delta_Cx_lef*dlef + dXdQ*Q;

   /* ZZZZZZZZ Cz_tot ZZZZZZZZ */
```

```
dZdQ = (cbar/(2*vt))*(Czq + delta_Cz_lef*dlef);

Cz_tot = Cz + delta_Cz_lef*dlef + dZdQ*Q;

   /* MMMMMMMM Cm_tot MMMMMMMM */

dMdQ = (cbar/(2*vt))*(Cmq + delta_Cmq_lef*dlef);

Cm_tot = Cm*eta_el + Cz_tot*(xcgr-xcg) + delta_Cm_lef*dlef + dMdQ*Q + delta_Cm + delta_Cm_ds;

   /* YYYYYYYY Cy_tot YYYYYYYY */

dYdail = delta_Cy_a20 + delta_Cy_a20_lef*dlef;

dYdR = (B/(2*vt))*(Cyr + delta_Cyr_lef*dlef);

dYdP = (B/(2*vt))*(Cyp + delta_Cyp_lef*dlef);

Cy_tot = Cy + delta_Cy_lef*dlef + dYdail*dail + delta_Cy_r30*drud + dYdR*R + dYdP*P;

   /* NNNNNNNN Cn_tot NNNNNNNN */

dNdail = delta_Cn_a20 + delta_Cn_a20_lef*dlef;

dNdR = (B/(2*vt))*(Cnr + delta_Cnr_lef*dlef);

dNdP = (B/(2*vt))*(Cnp + delta_Cnp_lef*dlef);

Cn_tot = Cn + delta_Cn_lef*dlef - Cy_tot*(xcgr-xcg)*(cbar/B) + dNdail*dail +
delta_Cn_r30*drud + dNdR*R + dNdP*P + delta_Cnbeta*beta;

   /* LLLLLLLL Cl_tot LLLLLLLL */

dLdail = delta_Cl_a20 + delta_Cl_a20_lef*dlef;

dLdR = (B/(2*vt))*(Clr + delta_Clr_lef*dlef);

dLdP = (B/(2*vt))*(Clp + delta_Clp_lef*dlef);

Cl_tot = Cl + delta_Cl_lef*dlef + dLdail*dail + delta_Cl_r30*drud + dLdR*R + dLdP*P + delta_Clbeta*beta;

   /* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      compute Udot,Vdot, Wdot,(as on NASA report p36)
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

Udot = R*V - Q*W - g*st + qbar*S*Cx_tot/m + T/m;

Vdot = P*W - R*U + g*ct*sphi + qbar*S*Cy_tot/m;

Wdot = Q*U - P*V + g*ct*cphi + qbar*S*Cz_tot/m;

   /* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      vt_dot equation (from S&L, p82)
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

xdot[6] = (U*Udot + V*Vdot + W*Wdot)/vt;

   /* %%%%%%%%%%%%%%%%%%%%
      alpha_dot equation
      %%%%%%%%%%%%%%%%%%%% */
```

```
xdot[7] = (U*Wdot - W*Udot)/(U*U + W*W);

  /* %%%%%%%%%%%%%%%%%
     beta_dot equation
     %%%%%%%%%%%%%%%%% */

xdot[8] = (Vdot*vt - V*xdot[6])/(vt*vt*cb);



  /* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     compute Pdot, Qdot, and Rdot (as in Stevens and Lewis p32)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

L_tot = Cl_tot*qbar*S*B;       /* get moments from coefficients */
M_tot = Cm_tot*qbar*S*cbar;
N_tot = Cn_tot*qbar*S*B;

denom = Jx*Jz - Jxz*Jxz;

  /* %%%%%%%%%%%%%%%%%%%%%%%
     Pdot
     %%%%%%%%%%%%%%%%%%%%%%% */

xdot[9] =  (Jz*L_tot + Jxz*N_tot - (Jz*(Jz-Jy)+Jxz*Jxz)*Q*R + Jxz*(Jx-Jy+Jz)*P*Q + Jxz*Q*Heng)/denom;

  /* %%%%%%%%%%%%%%%%%%%%%%%
     Qdot
     %%%%%%%%%%%%%%%%%%%%%%% */

xdot[10] = (M_tot + (Jz-Jx)*P*R - Jxz*(P*P-R*R) - R*Heng)/Jy;

  /* %%%%%%%%%%%%%%%%%%%%%%%
     Rdot
     %%%%%%%%%%%%%%%%%%%%%%% */

xdot[11] = (Jx*N_tot + Jxz*L_tot + (Jx*(Jx-Jy)+Jxz*Jxz)*P*Q - Jxz*(Jx-Jy+Jz)*Q*R +  Jx*Q*Heng)/denom;



/*########################################*/
/*### Create accelerations anx_cg, any_cg */
/*### ans anz_cg as outputs ##############*/
/*########################################*/

accels(xu,xdot,temp);

xdot[12]  = temp[0]; /* anx_cg */
xdot[13]  = temp[1]; /* any_cg */
xdot[14]  = temp[2]; /* anz_cg */
xdot[15]  = mach;
xdot[16]  = qbar;
xdot[17]  = ps;

/*########################################*/
/*########################################*/

free(temp);

}; /*##### END of nlplant() ####*/
```

```
/*########################################*/
/*### Called Sub-Functions  ##############*/
/*########################################*/

/*########################################*/
/* Function for mach and qbar */
/*########################################*/

void atmos(double alt, double vt, double *coeff ){

    double rho0 = 2.377e-3;
    double tfac, temp, rho, mach, qbar, ps;

    tfac =1 - .703e-5*(alt);
    temp = 519.0*tfac;
    if (alt >= 35000.0) {
       temp=390;
    }

    rho=rho0*pow(tfac,4.14);
    mach = (vt)/sqrt(1.4*1716.3*temp);
    qbar = .5*rho*pow(vt,2);
    ps   = 1715.0*rho*temp;

    if (ps == 0){
      ps = 1715;
      }

    coeff[0] = mach;
    coeff[1] = qbar;
    coeff[2] = ps;
}

/*########################################*/
/*########################################*/


/*########################################*/
/*### Port from matlab fix() function ####*/
/*########################################*/

int fix(double in){
    int out;

    if (in >= 0.0){
       out = (int)floor(in);
    }
    else if (in < 0.0){
       out = (int)ceil(in);
    }

    return out;
}

/* port from matlab sign() function */
int sign(double in){
    int out;

    if (in > 0.0){
```

```
        out = 1;
    }
    else if (in < 0.0){
        out = -1;
    }
    else if (in == 0.0){
        out = 0;
    }
    return out;
}

/*##########################################*/
/*##########################################*/


/*##########################################*/
/*#### Calculate accelerations from states */
/*#### and state derivatives. ############ */
/*##########################################*/

void accels(double *state,
            double *xdot,
            double *y)
{


#define grav 32.174

double sina, cosa, sinb, cosb ;
double vel_u, vel_v, vel_w ;
double u_dot, v_dot, w_dot ;
double nx_cg, ny_cg, nz_cg ;

sina = sin(state[7]) ;
cosa = cos(state[7]) ;
sinb = sin(state[8]) ;
cosb = cos(state[8]) ;
vel_u = state[6]*cosb*cosa ;
vel_v = state[6]*sinb ;
vel_w = state[6]*cosb*sina ;
u_dot =          cosb*cosa*xdot[6]
      - state[6]*sinb*cosa*xdot[8]
      - state[6]*cosb*sina*xdot[7] ;
v_dot =          sinb*xdot[6]
      + state[6]*cosb*xdot[8] ;
w_dot =          cosb*sina*xdot[6]
      - state[6]*sinb*sina*xdot[8]
      + state[6]*cosb*cosa*xdot[7] ;
nx_cg = 1.0/grav*(u_dot + state[10]*vel_w - state[11]*vel_v)
      + sin(state[4]) ;
ny_cg = 1.0/grav*(v_dot + state[11]*vel_u - state[9]*vel_w)
      - cos(state[4])*sin(state[3]) ;
nz_cg = -1.0/grav*(w_dot + state[9]*vel_v - state[10]*vel_u)
      + cos(state[4])*cos(state[3]) ;

y[0] = nx_cg ;
y[1] = ny_cg ;
y[2] = nz_cg ;


}
```

```
/*#######################################*/
/*#######################################*/

/*#######################################*/
/*#######################################*/
```

References

[1] Brian Stevens & Frank Lewis, *Aircraft Control and Simulation*, Wiley Inter-science, New York, 1992.

[2] Nguyen, Ogburn, Gilbert, Kibler, Brown and Deal, *NASA Technical Paper 1538 - Simulator Study of Stall / Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability*, Dec 1979.