Posting Date: Monday Dec. 2nd.
Due Date: Wednesday Dec. 11th.

1. Complete the MATLAB template file `linearizedmodelaircraft01_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB function `linearizedmodelaircraft01.m`. This function must compute the linearized state dynamics model for perturbations of the 3D point-mass aircraft motion model from steady, level, straight-line motion. Recall that the formulas for this model's *A* and *B* matrices have been given in a previous assignment. Hand in the code for your completed function.

2. Complete the MATLAB template file `script_analandsimaircraft01_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB script `script_analandsimaircraft01.m`. This script uses the function `linearizedmodelaircraft01.m` to develop a linearized model near steady-motion state and control vectors. The steady-motion state and control are defined in the file `linearizedmodelaircraft01_data03.mat`. It then checks the stability of the linearized model by checking its eigenvalues. Next, it checks the controllability of the linearized model. Next, it designs a full-state feedback controller by using pole placement techniques on the linearized model. Finally, it simulates the closed-loop and open-loop response of the system and makes various plots.

   You can test your code by running it for the alternate steady-motion conditions in the alternate input file `linearizedmodelaircraft01_data04.mat`. In that case, the displayed outputs that you should expect of your code are given in comment statements at the end of `script_analandsimaircraft01_temp.m`. These outputs will allow you to check your code.

   Run your `script_analandsimaircraft01.m` code using the steady-motion input file `linearizedmodelaircraft01_data03.mat`. Hand in the parts of your code that had ???? marks that you filled in. Hand in all of the displayed numerical outputs of your code for this run (which must be the same set of outputs as are given at the end of `script_analandsimaircraft01_temp.m` for a different input file), hand in the plot that shows the closed-loop and the open-loop ground tracks, hand in the plot that shows the altitude, airspeed, flight-path angle, and heading angle time histories, and hand in the plot that shows the thrust, angle-of-attack, and roll angle time histories. How well does the closed-loop system track the target steady-motion trajectory after initial transients have died out?

   Re-run the script with the alternate initial condition `x0` that scales down the perturbation from the steady-motion condition by a factor of 20. For this second case, hand in just the plot that shows the altitude, airspeed, flight-path angle, and heading angle time histories. Compare this plot with the similar plot from the previous case. In which case does the nonlinear response more closely match the linear response? Is this what you would expect?

3. Complete the MATLAB template file `linearizedmodelgravgradsc01_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB function

`linearizedmodelgravgradsc01.m`. This function must compute the linearized state dynamics model for perturbations from equilibrium of a rigid-body spacecraft's attitude dynamics model acting under the influence of gravity-gradient torques and an additional external control torque. Recall that the formulas for this model's *A* and *B* matrices have been given in a previous assignment. Hand in the code for your completed function.

4. Complete the MATLAB template file `script_analandsimgravgradsc01_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB script `script_analandsimgravgradsc01.m`. This script uses the function `linearizedmodelgravgradsc01.m` to develop a linearized model about an equilibrium state vector and the corresponding control vector. The spacecraft model and orbit parameters are defined in the file `linearizedmodelgravgradsc13_data.mat`. The equilibrium that is used for linearization has the roll, pitch, and yaw axes aligned with their local-level orbit-following coordinate system counterparts. The script then checks the stability of the linearized model by checking its eigenvalues. Next, it defines a measurement model state coefficient matrix *C*. This matrix models the outputs of 3 sensors: a roll angle sensor, a pitch angle sensor, and a yaw rate sensor. Next, the script checks the observability of the linearized model. Next, it designs a full-state observer by using pole placement techniques on the linearized model. Finally, it simulates the system and observer response and makes various plots.

   You can test your code by running it for the spacecraft model and orbit parameters in the alternate input file `linearizedmodelgravgradsc16_data.mat`. In that case, the displayed outputs that you should expect of your code are given in comment statements at the end of `script_analandsimgravgradsc01_temp.m`. These outputs will allow you to check your code.

   Run your `script_analandsimgravgradsc01.m` code using the spacecraft model and orbit that are defined in the file `linearizedmodelgravgradsc13_data.mat`. Hand in the parts of your code that had ???? marks that you filled in. Hand in all of the displayed numerical outputs of your code for this run (which must be the same set of outputs as are given at the end of `script_analandsimgravgradsc01_temp.m` for a different input file), hand in the plot that shows the true system's roll, pitch, and yaw angle time histories, and hand in the plot that shows the observer's roll, pitch, and yaw attitude estimation error time histories (both nonlinear and linearized versions). How well do the observer errors converge to zero?

   Re-run the script with the alternate initial condition `x0` that scales down the perturbation from the steady-motion condition by a factor of 10. For this second case, hand in just the plot that shows the observer's roll, pitch, and yaw attitude estimation error time histories (both nonlinear and linearized versions). Compare this plot with the similar plot from the previous case. In which case does the nonlinear observer error response more closely match the linear response? Is this what you would expect?