Posting Date: Monday Sept. 23rd.
Due Date: Monday Sept. 30th.

1. Euler angles can be used to resolve the force components of the point-mass aircraft motion model that was the subject of the early lectures in this course. This analysis employs the rotation matrix from local-level North/East/Down coordinates to aircraft wind-axis coordinates. North/East/Down coordinates have $\hat{i}$ pointing in the local North direction at the coordinate system center, $\hat{j}$ pointing in the local East direction at the coordinate system center, and $\hat{k}$ pointing in the local Down direction at the coordinate system center. This axis system is designated "ned" or "N/E/D" axes. The wind-axis coordinates have $\hat{i}'''$ pointing along the velocity vector, $\hat{j}'''$ pointing along the aircraft's right wing, and $\hat{k}''' = \hat{i}''' \times \hat{j}'''$ pointing roughly downward, except that non-zero flight-path angle or roll/bank angle cause $\hat{k}'''$ not to point directly down. This second axis system is designated "win" or "wind" axes. The general rotation matrix to transform a vector from its representation along N/E/D axes to its representation along wind axes is

   $$R^{win/ned} = R_1(\phi)R_2(\gamma)R_3(\psi)$$

   where ϕ is the roll/bank angle, γ is the flight-path angle, and ψ is the yaw/heading angle, as in lecture. An additional rotation about the $\hat{j}'''$ axis transforms from wind axes to aircraft body axis. The additional rotation angle is the angle-of-attack. Let this third set of axes be designated the "bod" or "body" coordinate axes. They have $\hat{i}''''$ pointing out the nose of the aircraft, $\hat{j}''''$ pointing out the right wing, and $\hat{k}'''' = \hat{i}'''' \times \hat{j}''''$ pointing roughly downward, except that non-zero flight-path angle, roll/bank angle, or angle-of-attack will cause $\hat{k}''''$ not to point directly down. The rotation matrix to transform from the N/E/D axes representation of a vector to its body-axes representation is

   $$R^{bod/ned} = R_2(\alpha)R_1(\phi)R_2(\gamma)R_3(\psi)$$

   where α is the angle-of-attack, as in lecture. The 4 angles α, ϕ, γ, and ψ do not constitute a minimal Euler-angle set for the rotation from N/E/D axis to body axes, but this rotation matrix formula is still valid.

   A third set of axes that are needed for this analysis have $\hat{i}''$ pointing along the velocity vector (like $\hat{i}'''$), $\hat{j}''$ in the horizontal plane and pointing perpendicular to $\hat{i}''$ roughly out the right wing (but not exactly out the right wing due to non-zero roll/bank angle), and $\hat{k}'' = \hat{i}'' \times \hat{j}''$ pointing roughly downward (but not exactly downward due to non-zero flight-path angle). Let this set of axes be called the "nav" or the "navigation" axes. This last set of axes and its designation as the navigation axes are non-standard, but they will be useful to our analysis. The rotation matrix to transform from N/E/D axis to navigation axes takes the form:

   $$R^{nav/ned} = R_2(\gamma)R_3(\psi)$$

It is straightforward to use the given three rotation matrix definitions in order to determine the rotations between all of the frames. This is true because transformations can be concatenated by successive multiplication of their R matrices and because they can be inverted by transposing their R matrices. For example, the transformation from body axes to navigation axes can be deduced as follows:

$$\begin{aligned} R^{nav/bod} &= R^{nav/ned} R^{ned/bod} = R^{nav/ned} [R^{bod/ned}]^{\mathrm{T}} = R_2(\gamma) R_3(\psi) [R_2(\alpha) R_1(\phi) R_2(\gamma) R_3(\psi)]^{\mathrm{T}} \\ &= R_2(\gamma) R_3(\psi) [R_3(\psi)]^{\mathrm{T}} [R_2(\gamma)]^{\mathrm{T}} [R_1(\phi)]^{\mathrm{T}} [R_2(\alpha)]^{\mathrm{T}} \\ &= [R_1(\phi)]^{\mathrm{T}} [R_2(\alpha)]^{\mathrm{T}} = R_1(-\phi) R_2(-\alpha) \end{aligned}$$

This derivation has exploited the following properties of the three single-axis rotation matrices: $R_l(\beta)[R_l(\beta)]^{\mathrm{T}} = I$ and $[R_l(\beta)]^{\mathrm{T}} = R_l(-\beta)$ for any rotation axis l = 1, 2, or 3 and any rotation angle β. This derivation also exploits the fact that for any two matrices A and B for which the product AB is computable, $(AB)^{\mathrm{T}} = B^{\mathrm{T}}A^{\mathrm{T}}$.

The navigation axes are important for the following reason: The three inertial acceleration components that were used in lecture to derive the point-mass equations of motion are given in this reference frame: $\dot{V}$ is the acceleration along the $\hat{i}''$ navigation axis, $V \cos\gamma\, \dot{\psi}$ is the acceleration along the $\hat{j}''$ navigation axis, and $V\dot{\gamma}$ is the acceleration along the $-\hat{k}''$ navigation axis.

The other axes systems are important because they can be used to define the 4 forces that act on the aircraft. The thrust T acts along the $\hat{i}''''$ body axis. The lift L acts along the $-\hat{k}'''$ wind axis, and the drag D acts along the $-\hat{i}'''$ wind axis. The force of gravity mg acts along the $+\hat{k}$ down axis.

Derive the navigation-axes point-mass aircraft equations of motion that were given in lecture:

$$m\dot{V} = T\cos\alpha - D - mg\sin\gamma$$

$$mV\dot{\gamma} = [T\sin\alpha + L]\cos\phi - mg\cos\gamma$$

$$mV\cos\gamma\,\dot{\psi} = [T\sin\alpha + L]\sin\phi$$

Do this by using the information given above about coordinate frame relationships and the axes of the various accelerations and forces.

Hints: Write the representation of each of the 4 forces in the coordinate system in which it is defined above. Afterwards, apply the appropriate rotation matrix from that coordinate system to the navigation-axes coordinate system. The results along the three navigation axes will equal the corresponding accelerations along these three axis multiplied by the aircraft mass.

Note: This derivation resolves Newton's second law $\vec{F} = m\vec{a}$ in the non-inertial rotating navigation-axes coordinate system. This approach may seem wrong because the law holds true only when $\vec{a}$ is the acceleration with respect to inertial coordinates. In fact, the acceleration vector $[\dot{V}, (V\cos\gamma\dot{\psi}), (-V\dot{\gamma})]^T$ is the acceleration with respect to inertial coordinates. It is allowable to express this inertial acceleration vector along any useful axis system in order to apply Newton's 2nd law. Stated differently, one can multiply both sides of the 3-by-1 vector version of Newton's second law by any arbitrary 3-by-3 rotation matrix R without altering the validity of the law. Thus, if $\vec{F} = m\vec{a}$, then $R\vec{F} = R\,m\vec{a} = m(R\,\vec{a})$. It matters not whether R varies with time due to the time-varying nature of its attitude parameterization. The equation is still valid. Only three things matter: $\vec{F}$ must be the total external force, $\vec{a}$ must be the acceleration of the particle with respect to inertial coordinates, and the components of $\vec{F}$ and $\vec{a}$ must both be expressed along the same coordinate system axes.

2. Complete the MATLAB template file `rotmateuler123_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB function `rotmateuler123.m`. This function must compute the Euler-angle rotation matrix from an initial coordinate system to a final system. Test your function using the data provided in the example input/output file `rotmateuler123_data01.mat`. The correct rotation matrix output for your function is called `R_true` in this file. Next, use your function to compute the R matrix for the angles contained in the file `rotmateuler123_data02.mat`. Hand in the latter R matrix as displayed using MATLAB's "`format long`" command. Also hand in your completed template file.

3. Complete the MATLAB template file `rotmatquaternion_temp.m` by completing the parts of the code where ???? appears. The result will be the MATLAB function `rotmatquaternion.m`. This function must compute the quaternion-based rotation matrix from one set of right-hand axes to another set of right-handed axes. Test your function using the data provided in the example input/output file `rotmatquaternion_data05.mat`. The correct rotation matrix output for your function is called `R_true` in this file. It so happens that this is the same rotation matrix, to within machine precision, as has been computed using Euler angles in the file `rotmateuler123_data01.mat`. Next, use your function to compute the R matrix for the quaternion contained in the file `rotmatquaternion_data06.mat`. Hand in the latter R matrix as displayed using MATLAB's "`format long`" command. Also hand in your completed template file.

4. Suppose one is given two orthonormal 3-by-3 direction-cosines matrices R_a and R_b that both have determinant equal to +1. In general, these rotations will not commute. In other words, it is normally not true that $R_a R_b = R_b R_a$. Find an example pair of valid 3-by-3 rotation matrixes R_a and R_b such that $R_a R_b \neq R_b R_a$ and demonstrate this fact by performing the necessary matrix-matrix multiplications.

   Hint: Simple counter examples can be developed in which all of the elements of R_a and R_b are either +1, 0, or -1.