|  |  |  |
| --- | --- | --- |
| Case | Case Details | Analysis |
| CASE\_1 | 1. Without any limits on Euler angles 2. Pitch Torque **m = 4.849** | * When Pitch Torque m was maintained low as m = 0.1 (as in the test case folder ‘..\q\_version\testcases\ 1\_without\_euler\_limits\_u0=0’), pitch angle evaluated by the expression     is defined. Because in this case, P\_exp is within the range [-1,1] for which asin function is defined.   * But when m is bumped up to say m = 4.849, perhaps P\_exp is going out of bounds [-1,1] and that is the reason there is error during the run (at around 70 seconds of simulation) —      * This error vanishes when we do not use asin function at all and use a dummy test case of always setting y(8) = 0. That means the culprit for the above error is neither the expression for phi nor the expression for psi. Only culprit is the asin function that appears in the pitch expression. * Check case1\_code for code details. |
| CASE\_2 | 1. Without any limits on Euler angles 2. Pitch Torque **m = 30.0** | * Case code for this case is the same as CASE\_1 except a change in the input value of m. * While in the CASE\_1 of m=4.849, error occurs at t=70s, in this CASE\_2 of m=30, error occurs much before at around 12s itself. * Hence, it is definitely the case of asin going undefined.  * Remedy has to lie in setting bounds to the P\_exp, the argument of asin. * Check case2\_code for code details. |
| CASE\_3 | 1. With **theta** limits 2. Pitch Torque m = 30.0 | * Unlike CASE\_2, CASE\_3 never throws errors because whenever P\_exp crosses the range [-1,1], the value of theta is set to either –pi/2 or pi/2 based on the sign of P\_exp. * Since we have tested it for a much higher pitch torque (m=30) than that is usual, and we’ve seen that the code never crashes, it means this fix for pitch angle is working. * Check case3\_code for code details. |
| CASE\_4 | 1. With **phi, theta, psi** limits 2. Pitch Torque m = 30.0 | * One thing that is unusual in all the CASE\_1,2,3 is that there is also non-zero phi angle. This should not be the case because m is always applied about y-axis and this doesn’t have a coupling with x-axis and z-axis. So there should always be zero phi, zero psi when torque is applied about y-axis. This error is occurring because for computation of phi and psi, atan2(y,x) function is used which becomes undefined for values of both y and x being very close to zero. * Once we take care of that zone when y and x both are very close to zero at the same time, phi and psi will never attain non-zero values for any other values of y and x. * Hence in this case, phi is always zero when m is applied— |
| CASE\_5 | 1. With **phi, theta, psi** limits 2. Pitch Torque m = 0.1 | * Case code being the same as CASE\_4, only change in CASE\_5 is that ‘m’ is set to more practical value i.e., m = 0.1 which is the same value that we tested euler version of the code with. * But note that in case of theta, there is a difference between euler code and quaternion code. There is no way we can get pitch angle beyond the range [-pi/2, pi/2] because we have to rely on asin() function. * But in case if euler angle code we keep getting values that get integrated over time and can keep on increasing without any hurdles. So it is a trade off. If we only expect our aircraft pitch to stay between [-pi/2, pi/2], we can rely on quaternions which avoid singularities and are much faster computationally. On the other hand euler angles are physically more intuitive and depict the theta values over complete range of [0, 2pi]. |
| Case still not fully created | Shifting up the phi & psi curves by 2\*pi | * With the case code CASE\_4, we get phi and psi curves in the range of [-pi,pi] as they are obtained by using atan2 function. * But if we want to view them visually in the range of [0,2pi] we should add 2pi to the –ve values of the phi and psi. But this leads to no –ve values when yaw torque n is applied. So though phi curves compares well with e\_version for roll torque, when yaw torque is applied, it doesn’t match. So we are not shifting the curve up. Rather we have to interpret the results because both in e\_version and q\_version they are one and the same physical position but are represented with different angle values: in case of e\_version the range is [-2pi to 2pi] whereas in q\_version it is [-pi to pi]. |
|  |  |  |