Table of Contents

# Validation

The engineering flight simulation was tested comprehensively.

## Unit Testing

It was ensured that individual Matlab functions provided the output expected, and that Simulink blocks employing the Matlab functions worked on a unit level - Reference Models were employed for this purpose.

Sample unit testing code is provided below:

actual = COPFinSet([1,1,1,1]);  
expected = 1.75;  
assert( ...  
 actual == expected, ...  
 'COPFinSet() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = COPNoseConeHaack([1]);  
expected = 0.437;  
assert( ...  
 actual == expected, ...  
 'COPNoseConeHaack() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = COPNoseConeVonKarman([1]);  
expected = 0.5;  
assert( ...  
 actual == expected, ...  
 'COPNoseConVonKarman() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefFinBodyInterference([1,1]);  
expected = 4/3;  
assert( ...  
 actual == expected, ...  
 'CoefFinBodyInterference() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefMomentCorrective([1,1,1,1,1,0]);  
expected = 0.5;  
assert( ...  
 actual == expected, ...  
 'CoefMomentCorrective() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefMomentDampingAero([1,1,1,1,1,1,1,1,1,1,1]);  
expected = 0;  
assert( ...  
 actual == expected, ...  
 'CoefMomentDampingAero() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefMomentDampingJet([1,1,0,1,0]);  
expected = 1;  
assert( ...  
 actual == expected, ...  
 'CoefMomentDampingJet() Test 1 \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefMomentDampingJet([1,1,0,1,1]);  
expected = 0;  
assert( ...  
 actual == expected, ...  
 'CoefMomentDampingJet() Test 2 \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = DynamicCenterGravity([1,1,1,1]);  
expected = 1;  
assert( ...  
 actual == expected, ...  
 'DampingCenterGravity() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = DynamicMomentInertia([1,1,1,1,1,1,1]);  
expected = 13/12;  
assert( ...  
 actual == expected, ...  
 'DynamicMomentInertia() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = StabilityDerivativeFinSet([1,1,1,1,1,1,1]);  
expected = 1.656854;  
assert( ...  
 actual ~= expected, ...  
 'StabilityDerivativeFinSet() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = StaticStabilityMarginLong([1,1,1,1]);  
expected = 0;  
assert( ...  
 actual == expected, ...  
 'StaticStabilityMarginLong() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefNormalForceBodyLift([1,1,1,2]);  
expected = 4;  
assert( ...  
 actual == expected, ...  
 'CoefNormalForceBodyLift() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefNormalForceBodyLift([1,1,1,0]);  
expected = 0;  
assert( ...  
 actual == expected, ...  
 'CoefNormalForceBodyLift() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = ForceNormal([1,1,1,1]);  
expected = 0.5;  
assert( ...  
 actual == expected, ...  
 'ForceNormal() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = COPRocket([1,1,1,1,1,1,1,1]);  
expected = 1.620907;  
assert( ...  
 actual == expected, ...  
 'COPRocket() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
% make sure that coef\_normal\_force == 0 is captured and handled  
actual = COPRocket([1,1,1,1,1,1,1,0]);  
expected = 1;  
assert( ...  
 actual == expected, ...  
 'COPRocket() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
% make sure that coef\_normal\_force == 0 is captured and handled  
actual = COPRocket([1,1,1,1,1,1,1,0]);  
expected = 1;  
assert( ...  
 actual == expected, ...  
 'COPRocket() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = StabilityDerivativeFinSet([1.131536,3,0.095,0.118004,0.1,0.28,0.103]);  
expected = 6.3371;  
assert( ...  
 actual == expected, ...  
 'COPFinSet() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
actual = CoefFinBodyInterference([0.103,0.095]);  
expected = 1.351536;  
assert( ...  
 actual == expected, ...  
 'COPFinSet() \n actual = %d \n expected = %d ', actual, expected ...  
)  
  
disp('ALL ANGULAR\_FLIGHT\_UNIT\_TESTING.m TESTS PASSED!');

And the output is as follows

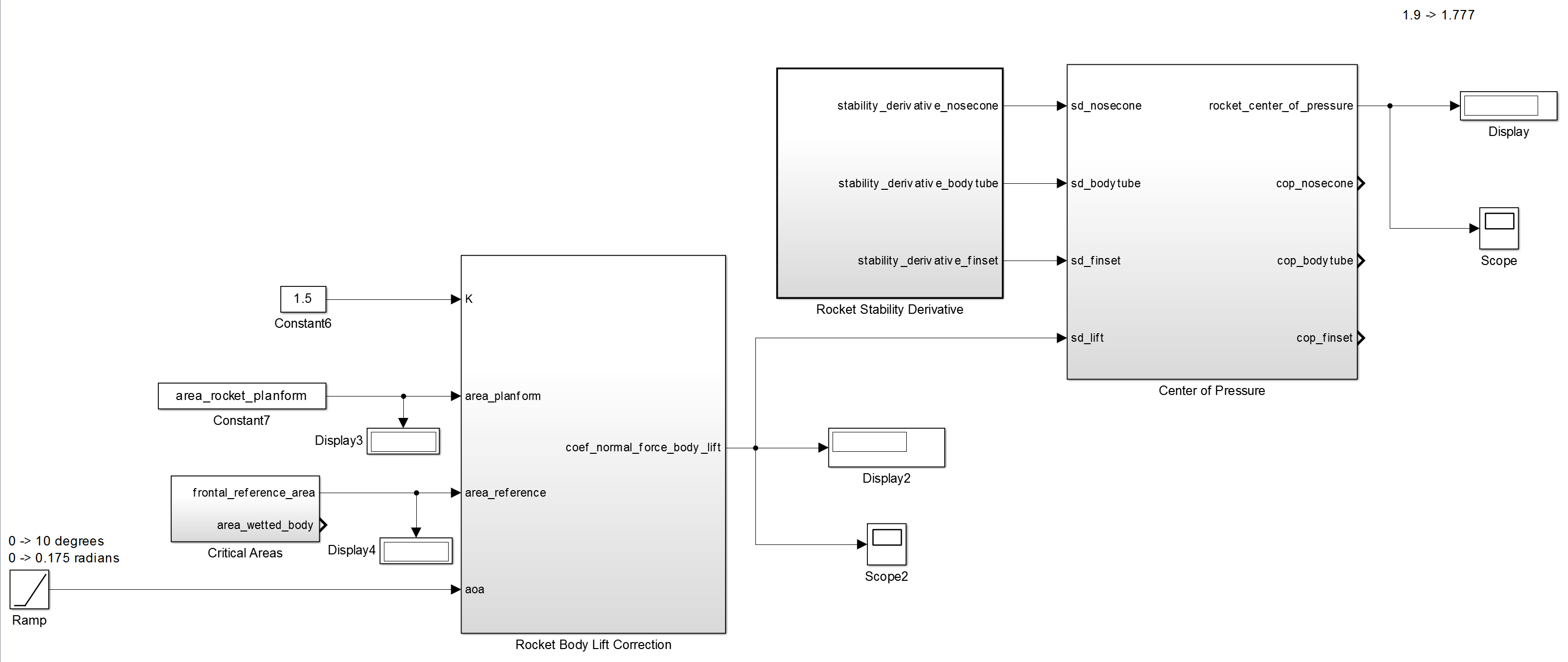
ALL ANGULAR\_FLIGHT\_UNIT\_TESTING.m TESTS PASSED!

## Integration Testing

Where many Simulink blocks were combined fro a higher-level function, they were tested with existing data from other simulators to verify functional correctness - Reference Models were again employed for this purpose.

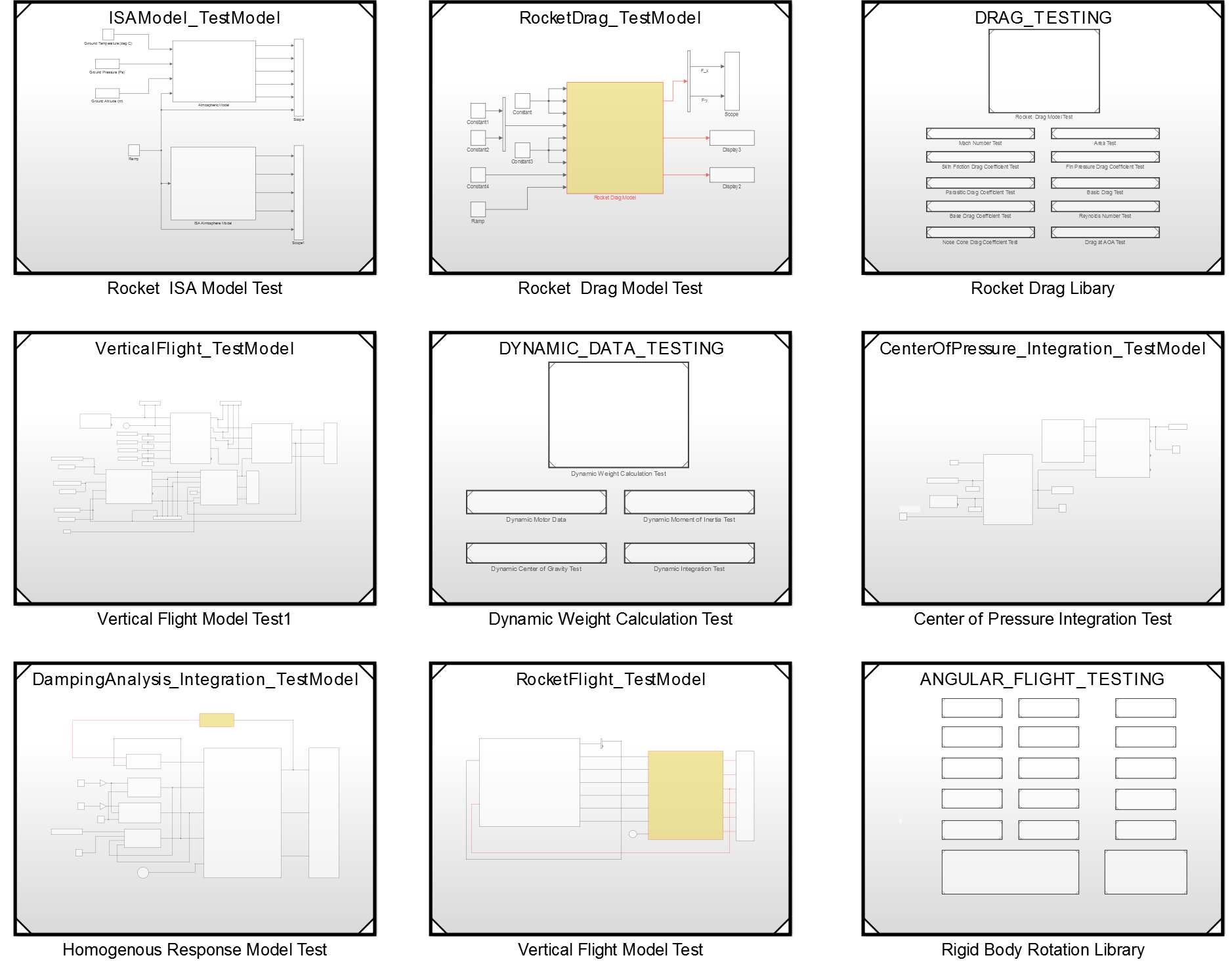
At this point, the code and the models were tested against values known to be correct to confirm their functionality.

Figure shows an example of an integration test model, used to verify the Center of Pressure calculation with the input *Stability Derivative* and *Rocket Body Lift Correction*.



Center of Pressure - Integration Test

Figure shows the high level Model Reference which contains all unit, integration, and system test models.



All Integration Tests

## System Testing

The simulator was tested on a system level by simulating the CR\_2-4G rocket flight on all available simulators and comparing all possible results. These results are discussed in detail in the *Simulation Execution* section.