```
%% 6-DOF Rocket Simulation Workspace Setup
clear; clc;

%% === 0. PHYSICAL ROCKET CONSTANTS ===
% Define these first so all sections can use them safely
rocket_dry_mass = 15.0;                % Mass of rocket without motor casing/fuel (kg)
rocket_ref_area = pi * (0.1524 / 2)^2; % Example: 6-inch diameter rocket (m^2)
r_rocket        = 0.0762;              % Radius of rocket (meters) -> 6-inch diameter / 2
L_rocket        = 2.5;                 % Total length of rocket (meters)


%% === 1. PARSE AEROTECH MOTOR DATA (.CSV FILE) ===
motorFile = 'myData.xlsx.csv'; % <--- Your filename

% Read the numerical data table, skipping the first row of column text labels
try
    motorDataMatrix = readmatrix(motorFile, 'NumHeaderLines', 1);
catch
    error('Could not read %s. Make sure it is in your active MATLAB directory.', motorFile);
end

% Create clean vectors for Simulink 1-D Lookup Tables
sim_motor_time   = motorDataMatrix(:,1);
sim_motor_thrust = motorDataMatrix(:,2);

% === MANUAL MOTOR MASS SPECIFICATIONS ===
% Aerotech J250-14 Data (Change these numbers if using a different motor)
total_mass = 0.612;   % <--- Enter your motor's total launch weight (in kg)
prop_mass  = 0.286;   % <--- Enter your motor's propellant weight (in kg)
dry_mass   = total_mass - prop_mass;

% Calculate dynamic mass over burn time (linear depletion approximation)
total_burn_time = max(sim_motor_time);
sim_motor_mass = zeros(size(sim_motor_time));

for i = 1:length(sim_motor_time)
    t = sim_motor_time(i);
    sim_motor_mass(i) = total_mass - (prop_mass * (t / total_burn_time));
end

% === DYNAMIC INERTIA TENSOR CALCULATION ===
% Pre-allocate the 3D matrix for Simulink (3x3 matrix for every time point)
sim_motor_I = zeros(3, 3, length(sim_motor_time));

for i = 1:length(sim_motor_time)
    % Current total mass at this specific split second
    current_mass = sim_motor_mass(i) + rocket_dry_mass;

    % Solid cylinder approximations for moments of inertia
    I_xx = 0.5 * current_mass * r_rocket^2;                    % Roll axis
    I_yy = (1/12) * current_mass * L_rocket^2;                 % Pitch axis
    I_zz = I_yy;                                               % Yaw axis

    % Populate the 3x3 matrix diagonal for time-step 'i'
    sim_motor_I(:,:,i) = diag([I_xx, I_yy, I_zz]);
end


%% === 2. PARSE OPENROCKET AERODYNAMICS (.CSV EXPORT) ===
aeroFile = 'J250-14 Drag data.csv'; % <--- Your filename

```

```
% Detect data layout properties
opts = detectImportOptions(aeroFile, 'FileType', 'text', 'CommentStyle', '#');

% Force the layout options to start reading data right after the headers
opts.DataLines = [5, Inf];

% Manually hardcode clean, friendly column names to override OpenRocket's text
opts.VariableNames = {'Time', 'AngleOfAttack', 'MachNumber', 'DragCoefficient'};

% Load the table using our overridden names
aeroData = readtable(aeroFile, opts);

% Clean the data: Remove rows where Mach or AoA are NaN to prevent interpolator crashes
cleanRows = ~isnan(aeroData.MachNumber) & ~isnan(aeroData.AngleOfAttack);
aeroData = aeroData(cleanRows, :);

% Extract unique grid coordinates for the 2-D Lookup Table
raw_mach = aeroData.MachNumber;
raw_aoa  = deg2rad(aeroData.AngleOfAttack); % Convert to radians for physics equations
raw_cd   = aeroData.DragCoefficient;

% Unique sorting forces a clean grid for Simulink's 2-D lookup breakpoints
sim_breakpoints_mach = unique(raw_mach);
sim_breakpoints_aoa  = unique(raw_aoa);

% Build a 2-D Grid Matrix for Cd(Mach, AoA) using scattered interpolation
F_interp = scatteredInterpolant(raw_mach, raw_aoa, raw_cd, 'linear', 'nearest');
[Mesh_Mach, Mesh_AoA] = meshgrid(sim_breakpoints_mach, sim_breakpoints_aoa);
sim_table_cd = F_interp(Mesh_Mach, Mesh_AoA)'; % Matched matrix for 2-D lookup block

% Add these to the very end of your script:
sim_block_mass = mean(sim_motor_mass + rocket_dry_mass); % Average total flight mass
sim_block_I    = mean(sim_motor_I, 3);                   % Average 3x3 inertia matrix


disp('✅ Workspace successfully initialized! Variables are ready for Simulink.');

```