## Table of Contents

# Test Script: Verify Constant Angular Velocity in Torque-Free Environment

Hypothesis: With only gravity (no J2, drag, or torques), omega should remain constant

```
clear; clc; close all;
```

## Setup Physical Parameters

```
params.mass = 100; % kg
params.mu = 3.986004418e14; % m^3/s^2
params.R_e = 6378137; % m

% Moment of inertia tensor (asymmetric for realism)
params.I_CB = [50, 0, 0;
               0, 75, 0;
               0, 0, 100]; % kg*m^2
```

## Initial Conditions for Circular Orbit

```
altitude = 500e3; % 500 km altitude
r0 = params.R_e + altitude; % orbital radius

% Circular orbit velocity
v_circ = sqrt(params.mu / r0);

% Initial position (start on equatorial plane)
r_vec0 = [r0; 0; 0]; % m

% Initial velocity (circular orbit in XY plane)
v_vec0 = [0; v_circ; 0]; % m/s
```

## Random Initial Angular Velocity

Set random seed for reproducibility

```
rng(42);

% Generate random angular velocity vector (rad/s)
omega0 = randn(3, 1) * 0.05; % Random omega with magnitude ~0.05 rad/s

% Initial quaternion (arbitrary orientation)
q0 = [0; 0; 0; 1]; % No rotation initially (can randomize if desired)
```

## Assemble Initial State Vector

```
X0 = [r_vec0; v_vec0; q0; omega0];
```

## Simulation Time

```
orbital_period = 2 * pi * sqrt(r0^3 / params.mu); % seconds
t_span = [0, 3 * orbital_period]; % Simulate for 3 orbits
```

## Control Input (Zero - No Forces or Torques)

```
u = zeros(6, 1);
```

## ODE Options

```
options = odeset('RelTol', 1e-9, 'AbsTol', 1e-12);
```

## Run Simulation with Only Gravity (No J2, No Drag, No Control)

```
fprintf('Running simulation with only gravitational force...\n');
fprintf('Initial omega: [%.6f, %.6f, %.6f] rad/s\n', omega0(1), omega0(2),
omega0(3));

[t, X] = ode45(@(t, X) Sat_template(t, X, u, params, ...
                                   'useJ2', false, ...
                                   'useAtmDrag', false, ...
                                   'useControl', false, ...
                                   'useSRP', false), ...
               t_span, X0, options);
```

```
Running simulation with only gravitational force...
Initial omega: [-0.026912, 0.043362, 0.048799] rad/s
```

## Extract Angular Velocity Components

```
omega_x = X(:, 11);
omega_y = X(:, 12);
omega_z = X(:, 13);
```

## Calculate Deviation from Initial Values

```
delta_omega_x = omega_x - omega0(1);
delta_omega_y = omega_y - omega0(2);
delta_omega_z = omega_z - omega0(3);

% Calculate magnitude of angular velocity
omega_mag = sqrt(omega_x.^2 + omega_y.^2 + omega_z.^2);
```

## Statistical Analysis

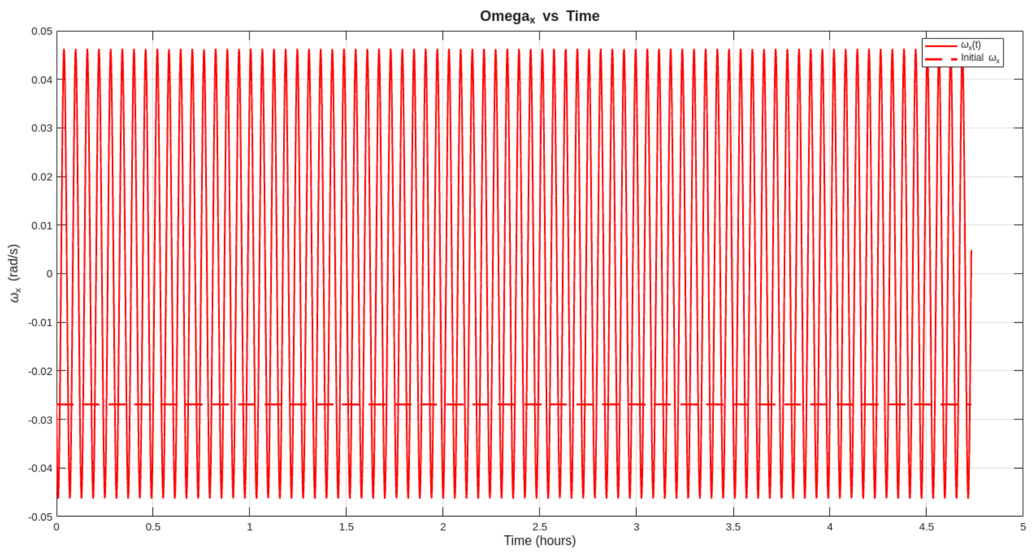
```
fprintf('\n=== Angular Velocity Stability Analysis ===\n');
fprintf('Simulation duration: %.2f hours (%.2f orbits)\n', ...
        t(end)/3600, t(end)/orbital_period);
fprintf('\nInitial omega: [%.8f, %.8f, %.8f] rad/s\n', omega0(1), omega0(2),
omega0(3));
fprintf('Final omega:   [%.8f, %.8f, %.8f] rad/s\n', omega_x(end),
omega_y(end), omega_z(end));
fprintf('\nMaximum deviation from initial:\n');
fprintf('  omega_x: %.3e rad/s\n', max(abs(delta_omega_x)));
fprintf('  omega_y: %.3e rad/s\n', max(abs(delta_omega_y)));
fprintf('  omega_z: %.3e rad/s\n', max(abs(delta_omega_z)));
fprintf('\nStandard deviation:\n');
fprintf('  omega_x: %.3e rad/s\n', std(omega_x));
fprintf('  omega_y: %.3e rad/s\n', std(omega_y));
fprintf('  omega_z: %.3e rad/s\n', std(omega_z));
fprintf('\nInitial omega magnitude: %.8f rad/s\n', norm(omega0));
fprintf('Final omega magnitude:   %.8f rad/s\n', omega_mag(end));
fprintf('Change in magnitude:     %.3e rad/s\n', omega_mag(end) -
norm(omega0));
```

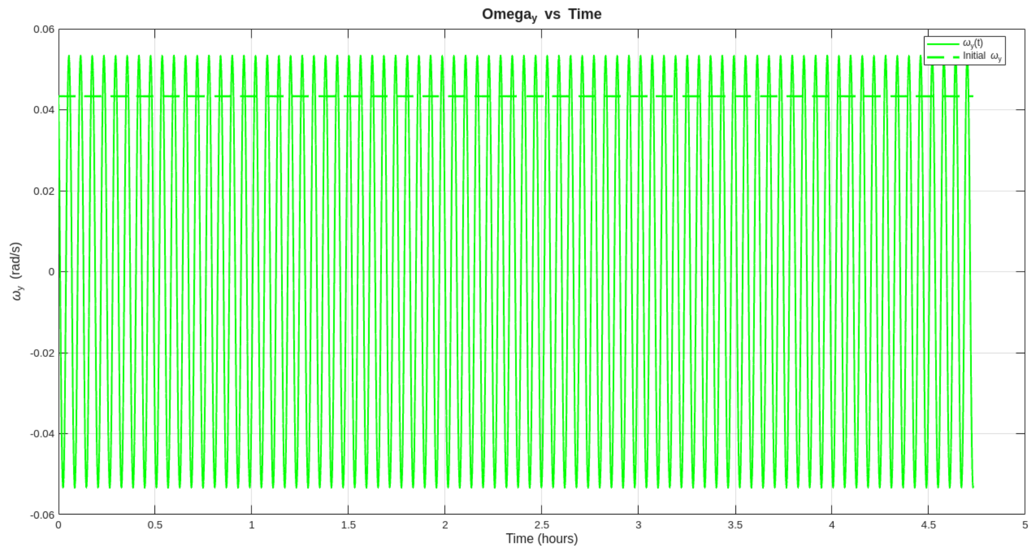
```
=== Angular Velocity Stability Analysis ===
Simulation duration: 4.73 hours (3.00 orbits)

Initial omega: [-0.02691219, 0.04336161, 0.04879932] rad/s
Final omega:   [0.00487658, -0.05304912, 0.04506807] rad/s

Maximum deviation from initial:
  omega_x: 7.311e-02 rad/s
  omega_y: 9.671e-02 rad/s
  omega_z: 6.757e-03 rad/s

Standard deviation:
  omega_x: 3.221e-02 rad/s
  omega_y: 3.824e-02 rad/s
  omega_z: 3.686e-03 rad/s

Initial omega magnitude: 0.07061069 rad/s
Final omega magnitude:   0.06977909 rad/s
Change in magnitude:     -8.316e-04 rad/s
```

## Improved Separate Plots for Each Omega Component

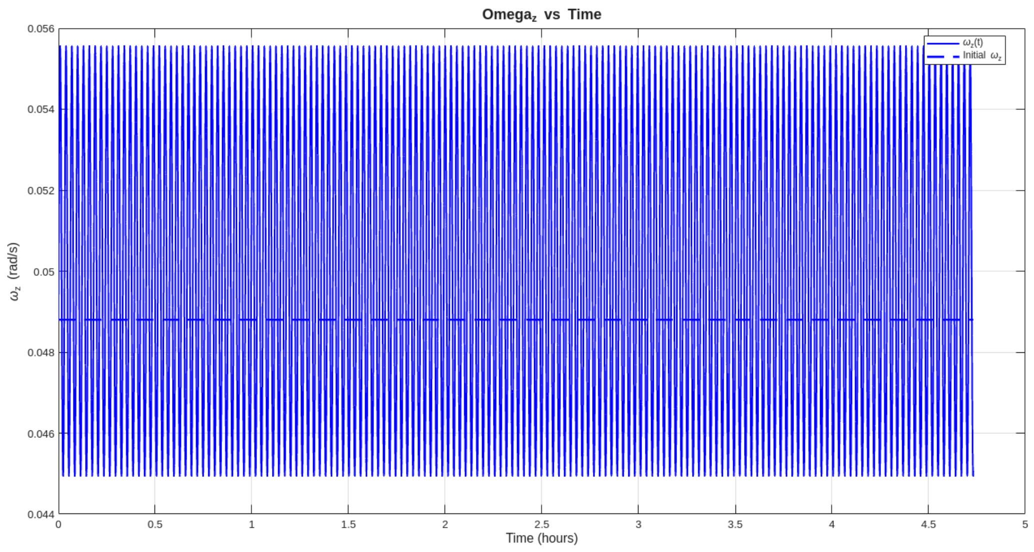
```
% Convert time to hours for better readability
t_hours = t / 3600;

% Figure 1: Omega_x vs Time
figure;
plot(t_hours, omega_x, 'r-', 'LineWidth', 1.5); hold on;
plot(t_hours, omega0(1) * ones(size(t_hours)), 'r--', 'LineWidth', 2);
ylabel('\omega_x (rad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Omega_x vs Time', 'FontSize', 14);
legend('\omega_x(t)', 'Initial \omega_x', 'Location', 'best');
grid on;

% Figure 2: Omega_y vs Time
figure;
plot(t_hours, omega_y, 'g-', 'LineWidth', 1.5); hold on;
plot(t_hours, omega0(2) * ones(size(t_hours)), 'g--', 'LineWidth', 2);
ylabel('\omega_y (rad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Omega_y vs Time', 'FontSize', 14);
legend('\omega_y(t)', 'Initial \omega_y', 'Location', 'best');
grid on;

% Figure 3: Omega_z vs Time
figure;
plot(t_hours, omega_z, 'b-', 'LineWidth', 1.5); hold on;
plot(t_hours, omega0(3) * ones(size(t_hours)), 'b--', 'LineWidth', 2);
ylabel('\omega_z (rad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Omega_z vs Time', 'FontSize', 14);
legend('\omega_z(t)', 'Initial \omega_z', 'Location', 'best');
grid on;

% Figure 4: Delta Omega_x vs Time
figure;
plot(t_hours, delta_omega_x * 1e6, 'r-', 'LineWidth', 1.5); hold on;
plot(t_hours, zeros(size(t_hours)), 'r--', 'LineWidth', 2);
ylabel('\Delta\omega_x (\murad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Deviation in Omega_x vs Time', 'FontSize', 14);
legend('\Delta\omega_x(t)', 'Zero Reference', 'Location', 'best');
grid on;
```

```
% Figure 5: Delta Omega_y vs Time
figure;
plot(t_hours, delta_omega_y * 1e6, 'g-', 'LineWidth', 1.5); hold on;
plot(t_hours, zeros(size(t_hours)), 'g--', 'LineWidth', 2);
ylabel('\Delta\omega_y (\murad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Deviation in Omega_y vs Time', 'FontSize', 14);
legend('\Delta\omega_y(t)', 'Zero Reference', 'Location', 'best');
grid on;

% Figure 6: Delta Omega_z vs Time
figure;
plot(t_hours, delta_omega_z * 1e6, 'b-', 'LineWidth', 1.5); hold on;
plot(t_hours, zeros(size(t_hours)), 'b--', 'LineWidth', 2);
ylabel('\Delta\omega_z (\murad/s)', 'FontSize', 12);
xlabel('Time (hours)', 'FontSize', 12);
title('Deviation in Omega_z vs Time', 'FontSize', 14);
legend('\Delta\omega_z(t)', 'Zero Reference', 'Location', 'best');
grid on;
```

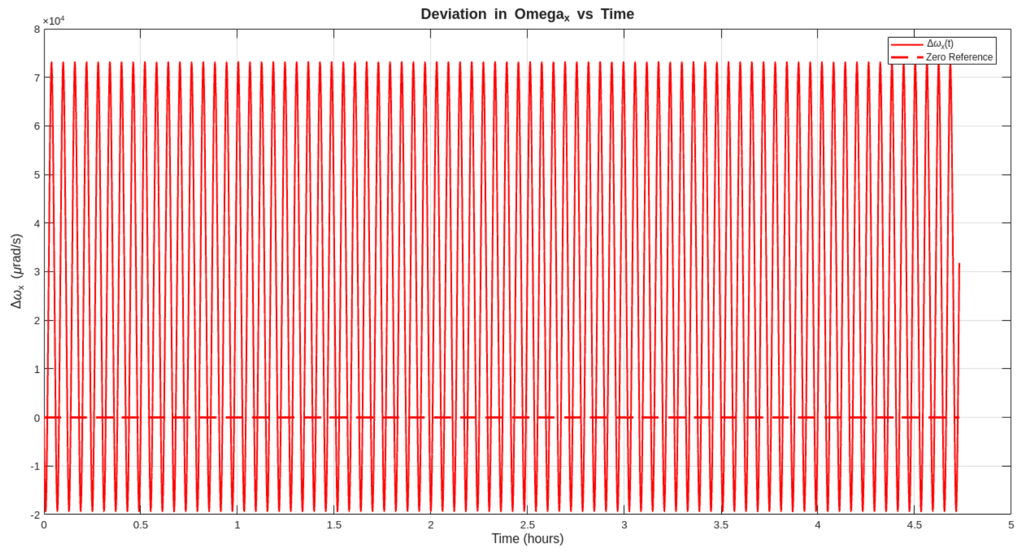


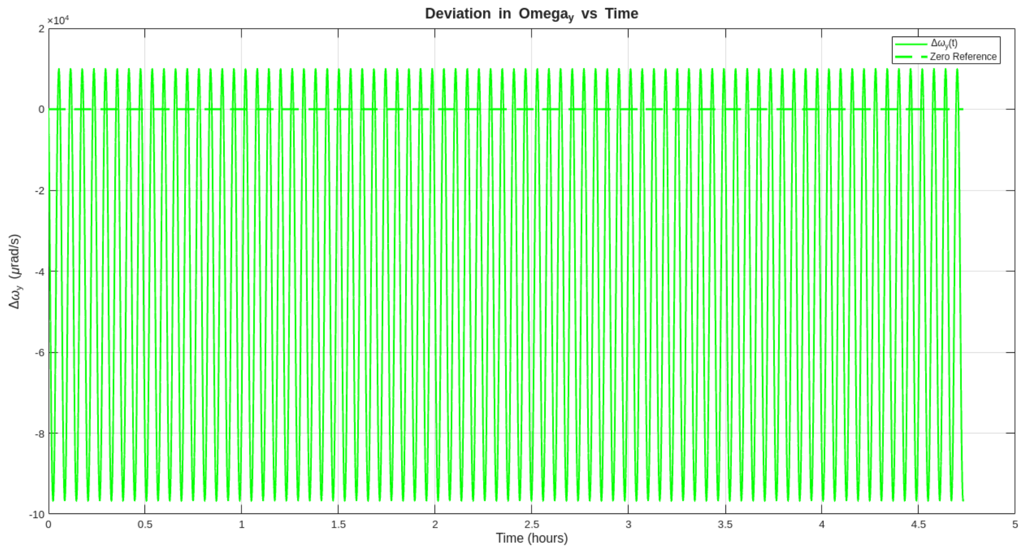


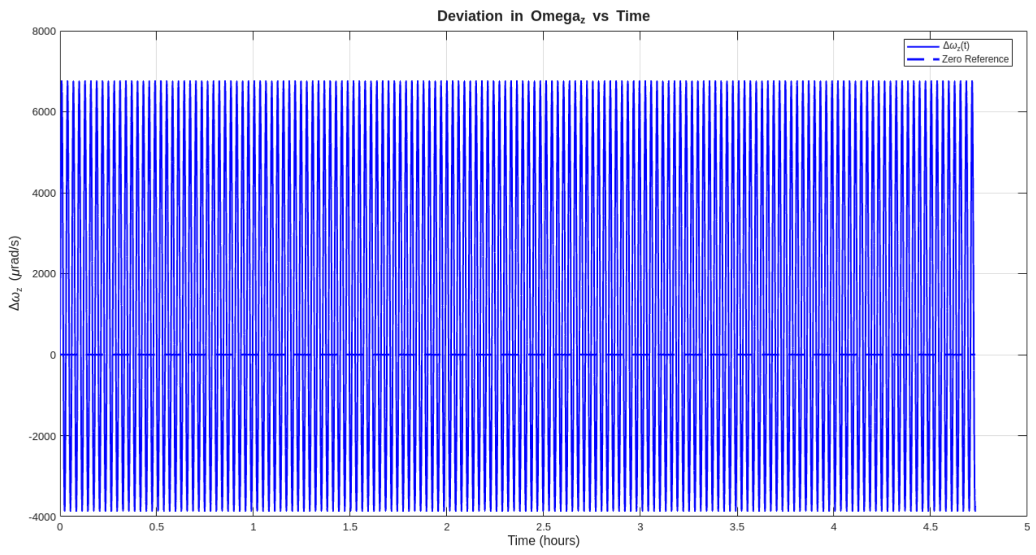
Omega$_y$ vs Time
$\omega_y$ (rad/s)
Time (hours)
$\omega_y$(t)
Initial $\omega_y$



Omega$_z$ vs Time
$\omega_z$ (rad/s)
Time (hours)
$\omega_z$(t)
Initial $\omega_z$


Deviation in Omega$_x$ vs Time
$\times10^4$
8
7
6
5
4
3
2
1
0
-1
-2
$\Delta\omega_x$ ($\mu$rad/s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time (hours)
$\Delta\omega_x(t)$
Zero Reference


Deviation in Omega$_y$ vs Time
$\times10^4$
2
0
-2
-4
-6
-8
-10
$\Delta\omega_y$ ($\mu$rad/s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time (hours)
$\Delta\omega_y(t)$
Zero Reference



*Published with MATLAB® R2025b*