NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Mestrado em Engenharia Eletrotécnica e de Computadores

Controlo em Sistemas Ciber-físicos

Controlling Satellite Dynamics
[EN]

Authors:

Filipe Cavalheiro (70119) a.canteiro@campus.fct.unl.pt
Rafael Louro (70714) fs.cavalheiro@campus.fct.unl.pt
António Canteiro (70604) rm.louro@campus.fct.unl.pt

2025/2026 – 1º Semestre

Table of contents

List of Images

1 Objectives of the Project

Attitude control is a fundamental requirement in spacecraft missions, as it enables the correct pointing of onboard instruments, sensors and communication systems. In this project, the objective is to design and evaluate control strategies for the attitude regulation of a rigid-body satellite described by a nonlinear rotational dynamic model.

The spacecraft orientation is represented by a rotation matrix $R \in SO(3)$, relating the body-fixed frame F_b to an inertial reference frame F_0. The angular velocity is denoted by $\omega \in \mathbb{R}^3$, and the applied control input is the torque vector $\tau \in \mathbb{R}^3$.

The rotational dynamics of the satellite are given by

$$\dot{R} = RS(\omega) \tag{1}$$

$$\dot{\omega} = J^{-1}\big(S(J\omega)\omega + \tau\big) \tag{2}$$

where J is the inertia matrix

$$J = \begin{bmatrix} 125.734 & 0 & 0 \\ 0 & 216.211 & 0 \\ 0 & 0 & 234.055 \end{bmatrix}; (\mathrm{kg} \cdot \mathrm{m}^2), \tag{3}$$

and $S(\cdot)$ denotes the skew-symmetric matrix such that $S(a)b = a \times b$.

The goal of this second project is divided into two tasks. The first task consists of designing an unconstrained attitude controller capable of steering the satellite from an initial orientation to a desired orientation expressed in Euler angles. The second task extends this controller by introducing directional constraints, requiring that the angle between any body axis and a given reference vector remains below a specified bound.

2 Proposed Control Strategy

To address the nonlinear nature of the spacecraft dynamics and the performance criteria imposed by the problem statement, a Nonlinear Model Predictive Control (NMPC) strategy was adopted. NMPC is well suited for this application, as it allows the explicit handling of nonlinear dynamics while optimizing a performance index over a finite prediction horizon.

2.1 State and Control Definition

The state vector is defined as

$$x = \begin{bmatrix} R\ \omega \end{bmatrix} \in \mathbb{R}^{12\times 1}, \tag{4}$$

where R stacks the elements of the rotation matrix column-wise. The control input is the torque vector

$$u = \tau \in \mathbb{R}^3. \tag{5}$$

2.2 NMPC Formulation

At each sampling instant, the controller solves a finite-horizon optimal control problem with horizon length N. The cost function penalizes deviations from the reference state and excessive control effort:

$$J = \sum_{k=0}^{N-1} \big((x_k - x_{\mathrm{ref}})^T Q (x_k - x_{\mathrm{ref}}) + u_k^T R u_k\big) \ + (x_N - x_{\mathrm{ref}})^T P (x_N - x_{\mathrm{ref}}), \tag{6}$$

where Q, R, and P are positive definite weighting matrices.

The optimization is subject to the discretized nonlinear dynamics

$$x_{k+1} = x_k + T_s f(x_k, u_k), \tag{7}$$

with $T_s = 0.2$ s being the sampling period.

The resulting nonlinear program is solved using `fmincon` with a Sequential Quadratic Programming (SQP) algorithm. Only the first control input of the optimal sequence is applied to the system, following the receding horizon principle.

3 Task 1: Unconstrained Attitude Control

In the first task, an NMPC controller is implemented to regulate the satellite attitude in the absence of actuator constraints. The controller is implemented as the MATLAB function `attController`, which receives the current state vector

$$x = \begin{bmatrix} \mathrm{vec}(R)^\top & \omega^\top \end{bmatrix}^\top \in \mathbb{R}^{12},$$

where $R \in SO(3)$ is the body-to-inertial rotation matrix and $\omega \in \mathbb{R}^3$ is the angular velocity, as well as a desired reference state x_{ref}.

The reference attitude is specified in terms of ZYX Euler angles and converted to a rotation matrix using standard MATLAB routines. Specifically, for this example the desired Euler angles

$$(\phi, \theta, \psi) = (30^\circ,\ -70^\circ,\ 132^\circ)$$

are converted to a rotation matrix $R_{\mathrm{ref}} \in SO(3)$, and the full reference state is constructed as

$$x_{\mathrm{ref}} = \begin{bmatrix} \mathrm{vec}(R_{\mathrm{ref}}) \\ 0_{3\times 1} \end{bmatrix},$$

corresponding to the desired orientation with zero angular velocity.

The initial condition is defined similarly. The satellite is initialized with an identity rotation matrix,

$$R(0) = I_3,$$

and a non-zero angular velocity

$$\omega(0) = \begin{bmatrix} 0.5 & -0.3 & 0.2 \end{bmatrix}^\top \text{ rad/s},$$

resulting in the initial state vector $x(0) = [\mathrm{vec}(R(0))^\top\ \omega(0)^\top]^\top$.

At each sampling instant, a finite-horizon optimal control problem with horizon length $N = 5$ is formulated. The cost function is quadratic in the state and control variables and penalizes deviations from the reference attitude and angular velocity, as well as control effort. The stage and terminal weighting matrices are given by

$$Q = \mathrm{diag}(50\,\mathbf{1}_9,\ 5\,\mathbf{1}_3), \qquad R = 0.01\,I_3, \qquad P = Q.$$

Due to the nonlinear nature of the satellite dynamics, which include rotational kinematics on $SO(3)$ and nonlinear gyroscopic coupling in the angular velocity dynamics, the resulting NMPC problem is non-convex. Consequently, to the best of our knowledge it cannot be solved analytically or with a globally optimal solver. Instead, the nonlinear constrained optimization problem is solved numerically using MATLAB's `fmincon` function with a Sequential Quadratic Programming (SQP) algorithm. This approach provides a locally optimal solution at each time step, which is sufficient for stabilizing the system in a receding-horizon fashion.

The satellite dynamics are enforced through nonlinear equality constraints using a forward Euler discretization of the continuous-time equations of motion. After solving the optimization problem, only the first control input of the optimal sequence is applied to the system, and the true continuous dynamics are propagated using an ODE solver.

The simulation is terminated when the attitude error, computed from the relative rotation matrix, falls below 0.1°, or when the maximum allowed simulation time of 20 minutes is reached. The NMPC controller successfully steers the satellite from its initial orientation to the desired attitude while producing smooth and bounded control torque profiles.

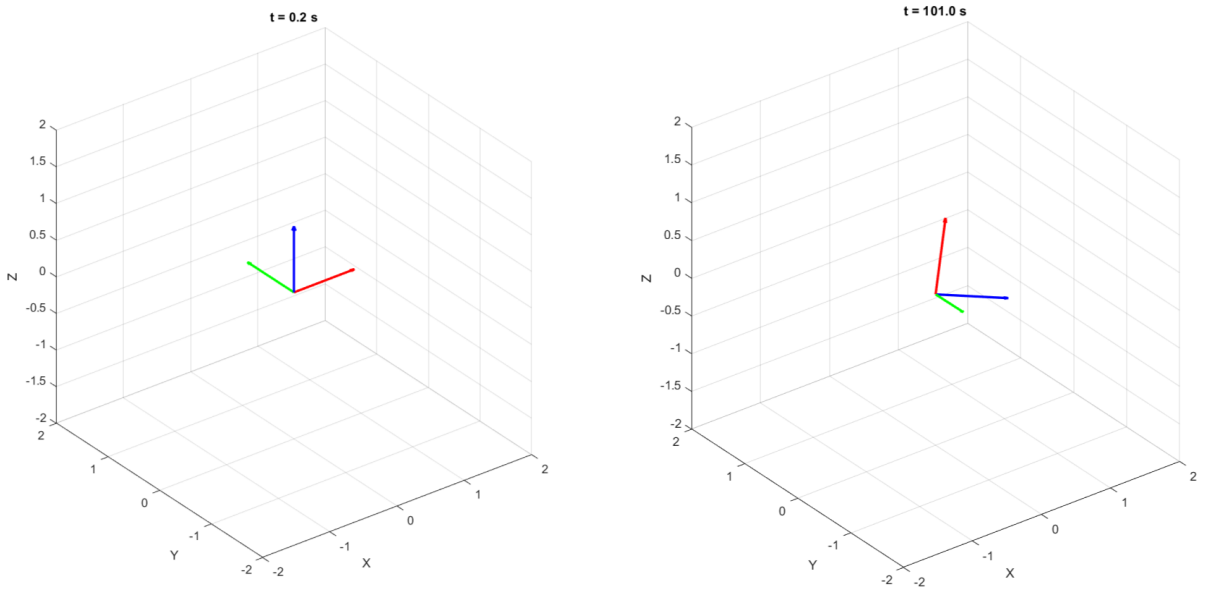

Figure 1: Evolution of the attitude during the maneuver.

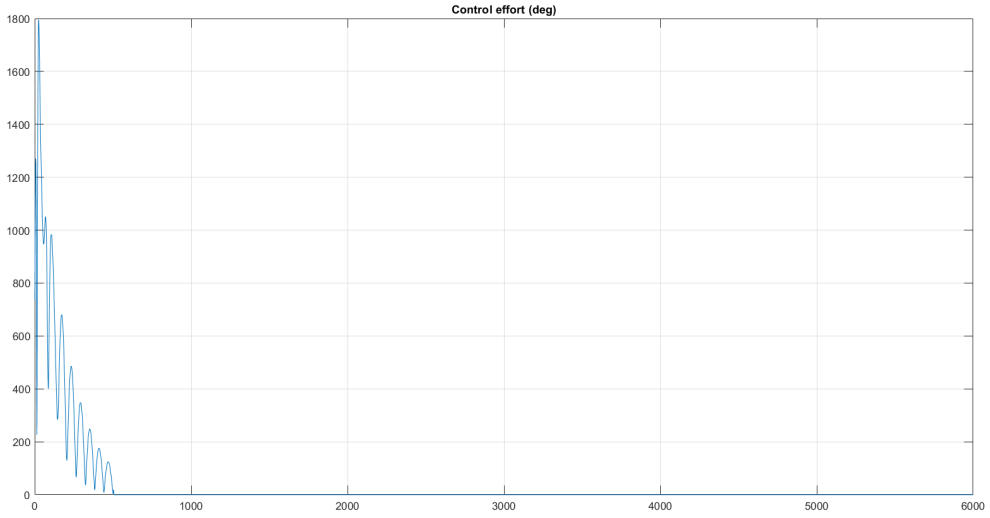

Figure 2: Control effort generated by the NMPC controller.

As shown in the fig. 2, the actuation does not reduce exponentially as in linear systems. Instead, it follows an overall decreasing trend with the presence of local maxima. With the given setup the system seems to always follow the reference to the desired attitude. The code can be seen in listing 1.

4 Task 2: Constrained Attitude Control

The second task extends the previous attitude control problem by introducing an additional geometric constraint. Given a fixed reference vector $v \in \mathbb{R}^3$, the angle between this vector and a selected body-fixed axis must remain below 10° throughout the maneuver.

To ensure feasibility, the system must be initialized in a configuration that already satisfies the constraint. Therefore, both the reference attitude and the initial condition are constructed explicitly to

enforce the angular constraint at the start of the maneuver.

The initial orientation is generated from randomly sampled Euler angles, from which the corresponding rotation matrix $R_0 \in SO(3)$ is obtained. The angular velocity is initialized randomly. One body-fixed axis of R_0 "i" is then selected and aligned with a reference direction such that the constraint is satisfied. Using this axis, an orthonormal reference frame is constructed, defining a constraint-consistent reference attitude R_{ref}.

This constraint can be expressed as

$$b_i^\top v \geq \cos(10^\circ), \tag{8}$$

where $b_i = Re_i$ denotes the i-th body-fixed axis expressed in the inertial frame, and $v \in \mathbb{R}^3$ is a unit reference vector. The inequality ensures that the angle between the selected body axis and the reference direction remains below the prescribed bound.

These nonlinear inequality constraints are directly incorporated into the NMPC optimization problem using `fmincon`. Although the controller retains the same NMPC structure as in Task 1, the additional geometric constraints reduce the feasible set of orientations, resulting in more conservative maneuvers and potentially increased convergence times.

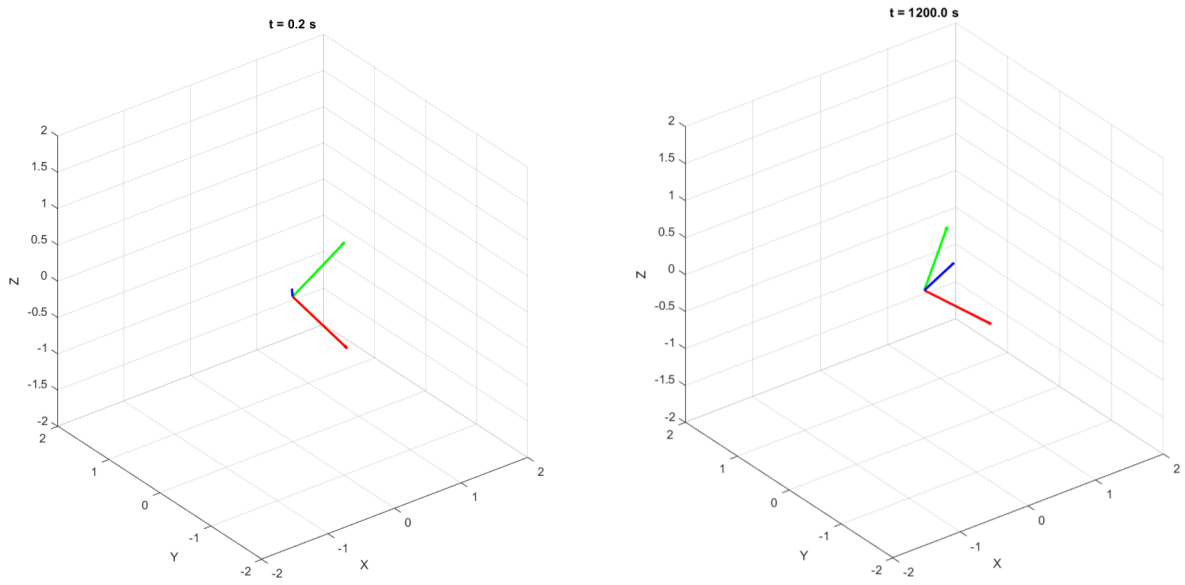

Figure 3: Evolution of the attitude during the maneuver.

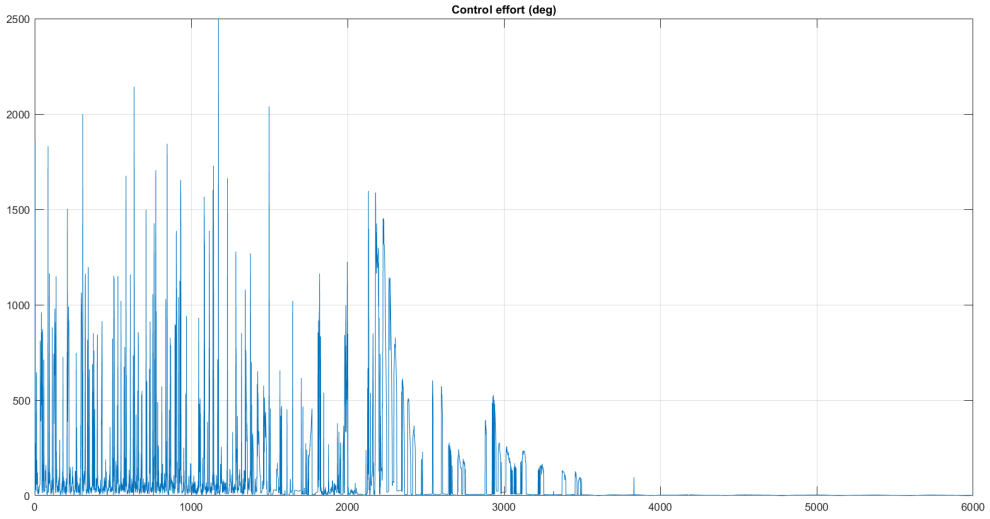

Figure 4: Control effort generated by the NMPC controller.

As shown in fig. 4, the presence of the geometric constraints leads to slower convergence when compared to the unconstrained case. This behavior is expected, as the constraints reduce the set of admissible orientations and force the optimizer to follow more conservative trajectories.

It is also observed that the angle constraints are not strictly satisfied at every prediction step. This behavior is attributed to the use of nonlinear inequality constraints within `fmincon`, which enforces constraint satisfaction only up to a specified numerical tolerance. Unlike simple bound constraints, which are handled explicitly by the solver, nonlinear constraints may exhibit small violations due to solver tolerances, discretization effects, and the iterative nature of the optimization process. Once again all code can be found in listing 2.

5 Discussion and Conclusions

This project demonstrated the effectiveness of NMPC for satellite attitude control under both unconstrained and constrained scenarios. The nonlinear formulation allows direct use of the rotation matrix representation, avoiding singularities associated with Euler angles.

The unconstrained controller achieves fast convergence with moderate control effort, while the constrained version highlights the trade-off between performance and safety requirements.

6 Anex

```m
% ========== SATELLITE NMPC - EXERCISE 1 ==========
clear; clc; close all;

% Parameters
Ts   = 0.2;                  % Sampling time (as in statement)
Tf   = 20*60;                % Max simulation time (20 minutes)
Nsim = floor(Tf/Ts);

nx = 12;                     % [R(:); omega]
nu = 3;                      % torque

N  = 5;                     % NMPC horizon

J = diag([125.734  216.211  234.055]);

% Reference (Euler angles -> Rotation matrix)
eul_ref = deg2rad([30; -70; 132]);   % desired orientation
Rref = eul2rotm(eul_ref', 'ZYX');    % MATLAB has some specific functions XD
xref = [Rref(:); zeros(3,1)];

% Initial condition
R0 = eye(3);
w0 = [0.5; -0.3; 0.2];
x  = [R0(:); w0];

% Cost matrices
Q = diag([50*ones(9,1); 5*ones(3,1)]);
P = Q;
R = 0.01*eye(nu);

Qbar = blkdiag(kron(eye(N),Q),P);
Rbar = kron(eye(N),R);

% Optimization options
opts = optimoptions('fmincon',...
    'Display','none',...
    'Algorithm','sqp',...
    'MaxIterations',200);

% Logs
xlog = zeros(nx,Nsim);
ulog = zeros(nu,Nsim);

% ========== MAIN SIMULATION LOOP ==========
for k = 1:Nsim
    disp("current k: " + k);
    xlog(:,k) = x;

    % Solve NMPC
    u = attController(x, xref, N, nx, nu,Ts, Qbar, Rbar, opts, J);

    ulog(:,k) = u;
```

```
53
54      % Integrate dynamics
55      [~,y] = ode45(@(t,y) satelliteDynamics(y,u,J),[0 Ts],x);
56      x = y(end,:)';
57
58      % Stop condition (0.1 deg)
59      Re = reshape(x(1:9),3,3)'*Rref;
60      ang_err = acos((trace(Re)-1)/2);
61      if rad2deg(abs(ang_err)) < 0.1
62          fprintf('Converged at t = %.2f s\n',k*Ts);
63          break;
64      end
65  end
66
67  %% ========== PLOTS ==========
68  figure;
69  plot(rad2deg(vecnorm(ulog)));
70  title('Control effort (deg)');
71  grid on;
72
73  figure;
74  for i = 1:1:k
75      clf;
76      plotSatelliteAxes(reshape(xlog(1:9,i),3,3));
77      title(sprintf('t = %.1f s',i*Ts));
78      axis([-2 2 -2 2 -2 2]);
79      drawnow;
80  end
81
82  %% ========== FUNCTIONS ==========
83  function u = attController(x0, xref, N, nx, nu, Ts, Qbar, Rbar, opts, J)
84      % Decision variables: [X; U]
85      XU0 = zeros((N+1)*nx + N*nu,1);
86
87      cost = @(XU) (XU - [repmat(xref,N+1,1); zeros(N*nu,1)])' ...
88                   * blkdiag(Qbar,Rbar) ...
89                   * (XU - [repmat(xref,N+1,1); zeros(N*nu,1)]);
90
91      nonlcon = @(XU) dynamicsConstraint(XU, x0, N, nx, nu, Ts, J);
92
93      XU = fmincon(cost,XU0,[],[],[],[],[],[],nonlcon,opts);
94
95      u = XU((N+1)*nx + (1:nu));
96  end
97
98  function [c,ceq] = dynamicsConstraint(XU, x0, N, nx, nu, Ts, J)
99
100     c = [];
101     ceq = [];
102
103     X = reshape(XU(1:(N+1)*nx),nx,N+1);
104     U = reshape(XU((N+1)*nx+1:end),nu,N);
105
106     ceq = [ceq; X(:,1) - x0];
107
```

```
    for k = 1:N
        xnext = X(:,k) + Ts*satelliteDynamics(X(:,k), U(:,k), J);
        ceq = [ceq; X(:,k+1) - xnext];
    end
end

function xdot = satelliteDynamics(x,u,J)

    S = @(w)[  0   -w(3)  w(2);
             w(3)   0   -w(1);
            -w(2) w(1)   0  ];

    R = reshape(x(1:9),3,3);
    w = x(10:12);

    Rdot = R*S(w);
    wdot = J \ (-S(w)*J*w + u);

    xdot = [Rdot(:); wdot];
end

function plotSatelliteAxes(R)
    O = [0 0 0];
    hold on; grid on; axis equal;
    quiver3(O(1),O(2),O(3),R(1,1),R(2,1),R(3,1),'r','LineWidth',2);
    quiver3(O(1),O(2),O(3),R(1,2),R(2,2),R(3,2),'g','LineWidth',2);
    quiver3(O(1),O(2),O(3),R(1,3),R(2,3),R(3,3),'b','LineWidth',2);
    xlabel('X'); ylabel('Y'); zlabel('Z');
    view(3);
end
```

Listing 1: MATLAB non constrained sattelite controller

```
    % ========== SATELLITE NMPC - EXERCISE 1 ==========
clear; clc; close all;

% Parameters
Ts   = 0.2;                 % Sampling time (as in statement)
Tf   = 20*60;               % Max simulation time (20 minutes)
Nsim = floor(Tf/Ts);

nx = 12;                    % [R(:); omega]
nu = 3;                     % torque

N  = 5;                     % NMPC horizon

J = diag([125.734 216.211 234.055]);

% Initial condition
eul_init = deg2rad((rand(3,1)-ones(3,1)*0.5)*360);   % desired orientation
R0 = eul2rotm(eul_init', 'ZYX');    % MATLAB has some specific functions
w0 = rand(3,1);
x  = [R0(:); w0];

i = 1;
ex = R0(i,:)';
```

```
ex = ex / norm(ex);

% choose any vector not parallel
v = [0;0;1];
if abs(dot(ex,v)) > 0.9
    v = [0;1;0];
end

ey = cross(v, ex);
ey = ey / norm(ey);
ez = cross(ex, ey);

Rref = [ex ey ez]';
constraint_vec = R0(i,:);
xref = [Rref(:); zeros(3,1)];

% Cost matrices
Q = diag([50*ones(9,1); 5*ones(3,1)]); % care more about position than
    velocity
P = Q;
R = 0.01*eye(nu);

Qbar = blkdiag(kron(eye(N),Q),P);
Rbar = kron(eye(N),R);

% Optimization options
opts = optimoptions('fmincon',...
    'Display','none',...
    'Algorithm','sqp',...
    'MaxIterations',200);

% Logs
xlog = zeros(nx,Nsim);
ulog = zeros(nu,Nsim);

% ========== MAIN SIMULATION LOOP ==========
for k = 1:Nsim
    xlog(:,k) = x;

    % Solve NMPC
    u = constrainedAttController(x, xref, N, nx, nu,Ts, Qbar, Rbar, opts, J
        , constraint_vec, i);

    ulog(:,k) = u;

    % Integrate dynamics
    [~,y] = ode45(@(t,y) satelliteDynamics(y,u,J),[0 Ts],x);
    x = y(end,:)';

    % Stop condition (0.1 deg)
    Re = reshape(x(1:9),3,3)'*Rref;
    ang_err = acos((trace(Re)-1)/2);
    disp("current k: " + k + " ang err: " + ang_err);
    if rad2deg(abs(ang_err)) < 0.1
        fprintf('Converged at t = %.2f s\n',k*Ts);
```

```
        break;
    end
end

%% ========== PLOTS ==========
figure;
plot(rad2deg(vecnorm(ulog)));
title('Control effort (deg)');
grid on;

figure;
hold on;
for i = 1:1:k
    clf;
    plotSatelliteAxes(reshape(xlog(1:9,i),3,3));
    title(sprintf('t = %.1f s',i*Ts));
    axis([-2 2 -2 2 -2 2]);
    drawnow;
end

%% ========== FUNCTIONS ==========
function u = constrainedAttController(x0, xref, N, nx, nu, Ts, ...
                                      Qbar, Rbar, opts, J, v, i)

    v = v / norm(v);                % ensure unit vector
    cosMax = cosd(10);              % 10 deg constraint

    XU0 = zeros((N+1)*nx + N*nu,1);

    cost = @(XU) (XU - [repmat(xref,N+1,1); zeros(N*nu,1)])' ...
                 * blkdiag(Qbar,Rbar) ...
                 * (XU - [repmat(xref,N+1,1); zeros(N*nu,1)]);

    nonlcon = @(XU) dynamicsAndAngleConstraint( ...
                    XU, x0, N, nx, nu, Ts, J, v, cosMax, i);

    XU = fmincon(cost, XU0, [], [], [], [], [], [], nonlcon, opts);

    u = XU((N+1)*nx + (1:nu));
end

function [c,ceq] = dynamicsAndAngleConstraint( ...
                    XU, x0, N, nx, nu, Ts, J, v, cosMax, i)

    c   = [];
    ceq = [];

    X = reshape(XU(1:(N+1)*nx), nx, N+1);
    U = reshape(XU((N+1)*nx+1:end), nu, N);

    % Initial condition
    ceq = [ceq; X(:,1) - x0];

    for k = 1:N
        % Dynamics constraint
```

```
132         xnext = X(:,k) + Ts*satelliteDynamics(X(:,k), U(:,k), J);
133         ceq   = [ceq; X(:,k+1) - xnext];
134
135         % ------ Angle constraints ------
136         R = reshape(X(1:9,k),3,3);
137         bi = R(:,i);                    % body axis in inertial frame
138         c  = [c; cosMax - dot(bi,v)];
139     end
140 end
141
142
143 function xdot = satelliteDynamics(x,u,J)
144
145     S = @(w)[  0   -w(3)  w(2);
146               w(3)   0   -w(1);
147              -w(2)  w(1)   0 ];
148
149     R = reshape(x(1:9),3,3);
150     w = x(10:12);
151
152     Rdot = R*S(w);
153     wdot = J \ (-S(w)*J*w + u);
154
155     xdot = [Rdot(:); wdot];
156 end
157
158 function plotSatelliteAxes(R)
159     O = [0 0 0];
160     hold on; grid on; axis equal;
161     quiver3(O(1),O(2),O(3),R(1,1),R(2,1),R(3,1),'r','LineWidth',2);
162     quiver3(O(1),O(2),O(3),R(1,2),R(2,2),R(3,2),'g','LineWidth',2);
163     quiver3(O(1),O(2),O(3),R(1,3),R(2,3),R(3,3),'b','LineWidth',2);
164     xlabel('X'); ylabel('Y'); zlabel('Z');
165     view(3);
166 end
```

Listing 2: MATLAB constrained sattelite controller