



NOVA SCHOOL OF  
SCIENCE & TECHNOLOGY

# MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

## CONTROLO EM SISTEMAS CIBER-FÍSICOS

---

### Controlling Satellite Dynamics [EN]

---

#### Authors:

António Canteiro (70604)  
Filipe Cavalheiro (70119)  
Rafael Louro (70714)

[a.canteiro@campus.fct.unl.pt](mailto:a.canteiro@campus.fct.unl.pt)  
[fs.cavalheiro@campus.fct.unl.pt](mailto:fs.cavalheiro@campus.fct.unl.pt)  
[rm.louro@campus.fct.unl.pt](mailto:rm.louro@campus.fct.unl.pt)

2025/2026 – 1º Semestre

## Table of contents

<b>1</b>	<b>Objectives of the Project</b>	<b>3</b>
<b>2</b>	<b>Proposed Control Strategy</b>	<b>3</b>
2.1	State and Control Definition	3
2.2	NMPC Formulation	3
<b>3</b>	<b>Task 1: Unconstrained Attitude Control</b>	<b>4</b>
<b>4</b>	<b>Task 2: Constrained Attitude Control</b>	<b>5</b>
<b>5</b>	<b>Discussion and Conclusions</b>	<b>7</b>
<b>6</b>	<b>Anex</b>	<b>8</b>

## List of Images

1	Evolution of the attitude during the maneuver.	5
2	Control effort generated by the NMPC controller.	5
3	Evolution of the attitude during the maneuver.	6
4	Control effort generated by the NMPC controller.	7

# 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

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

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

where  $J$  is the inertia matrix

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

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

$$x = [R \ \omega] \in \mathbb{R}^{12 \times 1}, \quad (4)$$

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

$$u = \tau \in \mathbb{R}^3. \quad (5)$$

## 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:

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

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

The optimization is subject to the discretized nonlinear dynamics

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

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

$$x = [\text{vec}(R)^\top \quad \omega^\top]^\top \in \mathbb{R}^{12},$$

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  $x_{\text{ref}}$ .

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  $R_{\text{ref}} \in SO(3)$ , and the full reference state is constructed as

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

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

$$\omega(0) = [0.5 \quad -0.3 \quad 0.2]^\top \text{ rad/s},$$

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

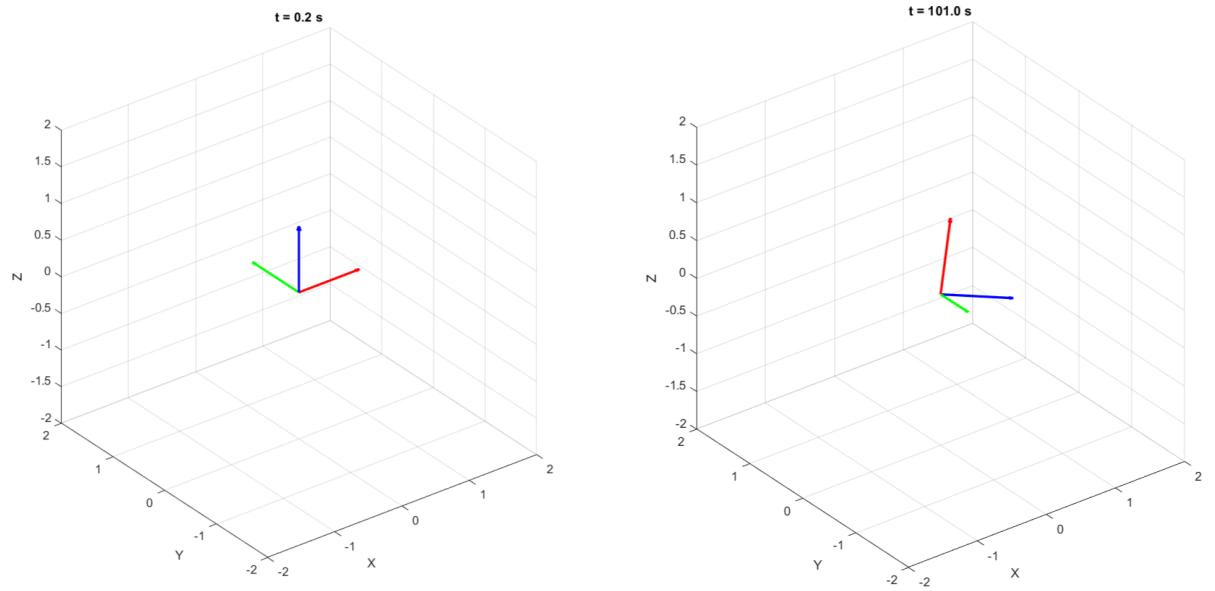
At each sampling instant, a finite-horizon optimal control problem with horizon length  $N = 5$  is formulated. The cost function is quadratic in the state and control variables and penalizes deviations from the reference attitude and angular velocity, as well as control effort. The stage and terminal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

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

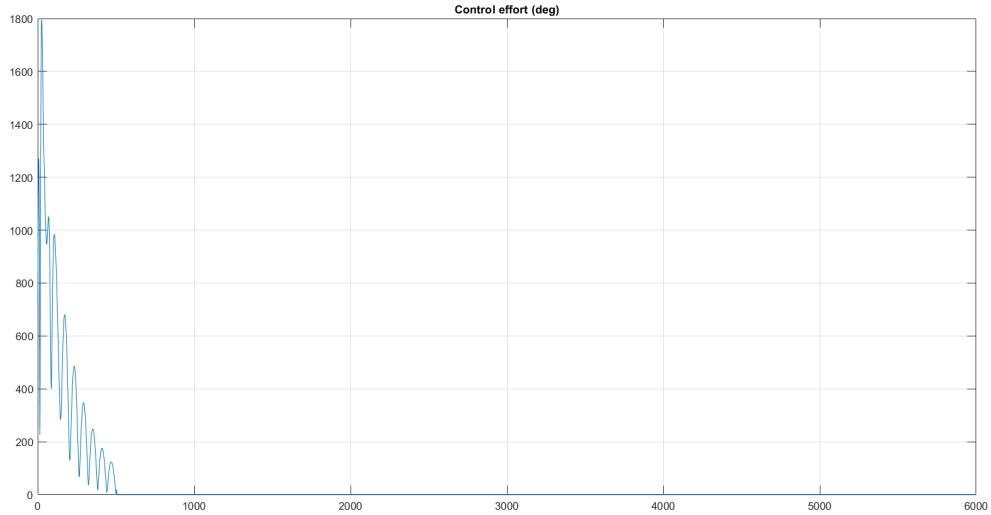
Due to the nonlinear nature of the satellite dynamics, which include rotational kinematics on  $SO(3)$  and nonlinear gyroscopic coupling in the angular velocity dynamics, the resulting NMPC problem is non-convex. Consequently, to the best of our knowledge it cannot be solved analytically or with a globally 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  $0.1^\circ$ , 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  $10^\circ$  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.

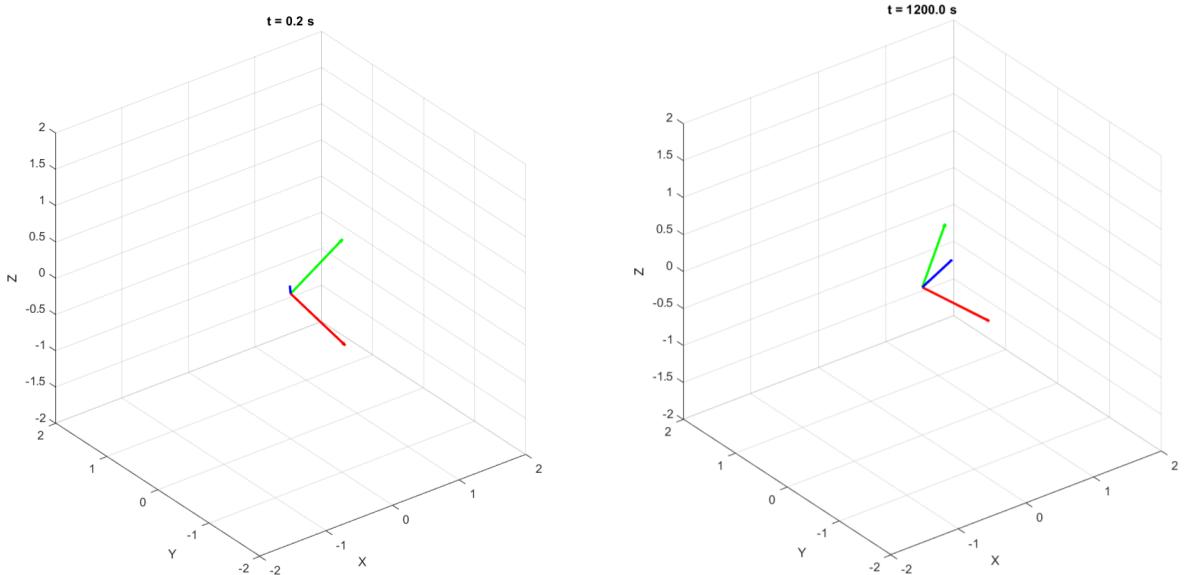
The initial orientation is generated from randomly sampled Euler angles, from which the corresponding rotation matrix  $R_0 \in SO(3)$  is obtained. The angular velocity is initialized randomly. One body-fixed axis of  $R_0$  "i" is then selected and aligned with a reference direction such that the constraint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

This constraint can be expressed as

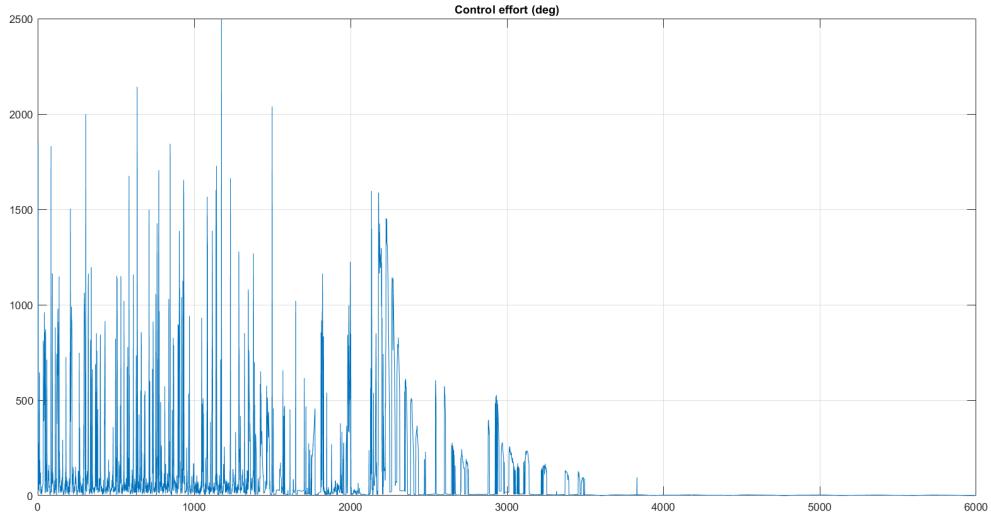
$$b_i^\top v \geq \cos(10^\circ), \quad (8)$$

where  $b_i = R e_i$  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

```

1 % ----- SATELLITE NMPC - EXERCISE 1 -----
2 clear; clc; close all;
3
4 % Parameters
5 Ts = 0.2; % Sampling time (as in statement)
6 Tf = 20*60; % Max simulation time (20 minutes)
7 Nsim = floor(Tf/Ts);
8
9 nx = 12; % [R(:); omega]
10 nu = 3; % torque
11
12 N = 5; % NMPC horizon
13
14 J = diag([125.734 216.211 234.055]);
15
16 % Reference (Euler angles -> Rotation matrix)
17 eul_ref = deg2rad([30; -70; 132]); % desired orientation
18 Rref = eul2rotm(eul_ref', 'ZYX'); % MATLAB has some specific functions
19 % XD
20 xref = [Rref(:); zeros(3,1)];
21
22 % Initial condition
23 R0 = eye(3);
24 w0 = [0.5; -0.3; 0.2];
25 x = [R0(:); w0];
26
27 % Cost matrices
28 Q = diag([50*ones(9,1); 5*ones(3,1)]);
29 P = Q;
30 R = 0.01*eye(nu);
31
32 Qbar = blkdiag(kron(eye(N),Q),P);
33 Rbar = kron(eye(N),R);
34
35 % Optimization options
36 opts = optimoptions('fmincon',...
37     'Display','none',...
38     'Algorithm','sqp',...
39     'MaxIterations',200);
40
41 % Logs
42 xlog = zeros(nx,Nsim);
43 ulog = zeros(nu,Nsim);
44
45 % ----- MAIN SIMULATION LOOP -----
46 for k = 1:Nsim
47     disp("current k: " + k);
48     xlog(:,k) = x;
49
50     % Solve NMPC
51     u = attController(x, xref, N, nx, nu, Ts, Qbar, Rbar, opts, J);
52     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

```

```

108 for k = 1:N
109     xnext = X(:,k) + Ts*satelliteDynamics(X(:,k), U(:,k), J);
110     ceq = [ceq; X(:,k+1) - xnext];
111 end
112 end
113
114 function xdot = satelliteDynamics(x,u,J)
115
116     S = @(w) [ 0 -w(3) w(2);
117                 w(3) 0 -w(1);
118             -w(2) w(1) 0 ];
119
120     R = reshape(x(1:9),3,3);
121     w = x(10:12);
122
123     Rdot = R*S(w);
124     wdot = J \ (-S(w)*J*w + u);
125
126     xdot = [Rdot(:); wdot];
127 end
128
129 function plotSatelliteAxes(R)
130     O = [0 0 0];
131     hold on; grid on; axis equal;
132     quiver3(O(1),O(2),O(3),R(1,1),R(2,1),R(3,1), 'r', 'LineWidth',2);
133     quiver3(O(1),O(2),O(3),R(1,2),R(2,2),R(3,2), 'g', 'LineWidth',2);
134     quiver3(O(1),O(2),O(3),R(1,3),R(2,3),R(3,3), 'b', 'LineWidth',2);
135     xlabel('X'); ylabel('Y'); zlabel('Z');
136     view(3);
137 end

```

**Listing 1:** MATLAB non constrained sattelite controller

```

1 % ===== SATELLITE NMPC – EXERCISE 1 =====
2 clear; clc; close all;
3
4 % Parameters
5 Ts = 0.2; % Sampling time (as in statement)
6 Tf = 20*60; % Max simulation time (20 minutes)
7 Nsim = floor(Tf/Ts);
8
9 nx = 12; % [R(:); omega]
10 nu = 3; % torque
11
12 N = 5; % NMPC horizon
13
14 J = diag([125.734 216.211 234.055]);
15
16 % Initial condition
17 eul_init = deg2rad((rand(3,1)-ones(3,1)*0.5)*360); % desired orientation
18 R0 = eul2rotm(eul_init, 'ZYX'); % MATLAB has some specific functions
19 w0 = rand(3,1);
20 x = [R0(:); w0];
21
22 i = 1;
23 ex = R0(i,:);

```

```

24 ex = ex / norm(ex);
25
26 % choose any vector not parallel
27 v = [0;0;1];
28 if abs(dot(ex,v)) > 0.9
29     v = [0;1;0];
30 end
31
32 ey = cross(v, ex);
33 ey = ey / norm(ey);
34 ez = cross(ex, ey);
35
36 Rref = [ex ey ez]';
37 constraint_vec = R0(i,:);
38 xref = [Rref(:); zeros(3,1)];
39
40 % Cost matrices
41 Q = diag([50*ones(9,1); 5*ones(3,1)]); % care more about position than
42 % velocity
43 P = Q;
44 R = 0.01*eye(nu);
45
46 Qbar = blkdiag(kron(eye(N),Q),P);
47 Rbar = kron(eye(N),R);
48
49 % Optimization options
50 opts = optimoptions('fmincon',...
51     'Display','none',...
52     'Algorithm','sqp',...
53     'MaxIterations',200);
54
55 % Logs
56 xlog = zeros(nx,Nsim);
57 ulog = zeros(nu,Nsim);
58
59 % ===== MAIN SIMULATION LOOP =====
60 for k = 1:Nsim
61     xlog(:,k) = x;
62
63     % Solve NMPC
64     u = constrainedAttController(x, xref, N, nx, nu, Ts, Qbar, Rbar, opts, J
65     , constraint_vec, i);
66
67     ulog(:,k) = u;
68
69     % Integrate dynamics
70     [~,y] = ode45(@(t,y) satelliteDynamics(y,u,J),[0 Ts],x);
71     x = y(end,:)';
72
73     % Stop condition (0.1 deg)
74     Re = reshape(x(1:9),3,3)*Rref;
75     ang_err = acos((trace(Re)-1)/2);
76     disp("current k: " + k + " ang err: " + ang_err);
    if rad2deg(abs(ang_err)) < 0.1
        fprintf('Converged at t = %.2f s\n',k*Ts);
    end
end

```

```

77         break;
78     end
79 end

80
81 %% ===== PLOTS =====
82 figure;
83 plot(rad2deg(vecnorm(ulog)));
84 title('Control effort (deg)');
85 grid on;

86
87 figure;
88 hold on;
89 for i = 1:k
90     clf;
91     plotSatelliteAxes(reshape(xlog(1:9,i),3,3));
92     title(sprintf('t = %.1f s',i*Ts));
93     axis([-2 2 -2 2 -2 2]);
94     drawnow;
95 end

96
97 %% ===== FUNCTIONS =====
98 function u = constrainedAttController(x0, xref, N, nx, nu, Ts, ...
99                                     Qbar, Rbar, opts, J, v, i)
100
101    v = v / norm(v); % ensure unit vector
102    cosMax = cosd(10); % 10 deg constraint
103
104    XU0 = zeros((N+1)*nx + N*nu,1);
105
106    cost = @(XU) (XU - [ repmat(xref,N+1,1); zeros(N*nu,1) ]) ' ...
107          * blkdiag(Qbar,Rbar) ...
108          * (XU - [ repmat(xref,N+1,1); zeros(N*nu,1) ]);
109
110    nonlcon = @(XU) dynamicsAndAngleConstraint( ...
111                  XU, x0, N, nx, nu, Ts, J, v, cosMax, i);
112
113    XU = fmincon(cost, XU0, [], [], [], [], [], nonlcon, opts);
114
115    u = XU((N+1)*nx + (1:nu));
116 end

117 function [c,ceq] = dynamicsAndAngleConstraint( ...
118                                         XU, x0, N, nx, nu, Ts, J, v, cosMax, i)
119
120    c = [];
121    ceq = [];
122
123
124    X = reshape(XU(1:(N+1)*nx), nx, N+1);
125    U = reshape(XU((N+1)*nx+1:end), nu, N);
126
127    % Initial condition
128    ceq = [ceq; X(:,1) - x0];
129
130    for k = 1:N
131        % 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
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
157 end
158
159 function plotSatelliteAxes(R)
160     O = [0 0 0];
161     hold on; grid on; axis equal;
162     quiver3(O(1),O(2),O(3),R(1,1),R(2,1),R(3,1),'r','LineWidth',2);
163     quiver3(O(1),O(2),O(3),R(1,2),R(2,2),R(3,2),'g','LineWidth',2);
164     quiver3(O(1),O(2),O(3),R(1,3),R(2,3),R(3,3),'b','LineWidth',2);
165     xlabel('X'); ylabel('Y'); zlabel('Z');
166     view(3);
167 end

```

**Listing 2:** MATLAB constrained sattelite controller