Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προχωρημένες Τεχνικές Συστημάτων Αυτομάτου Ελέγχου

Βέλτιστος Έλεγχος Ηλεκτρικού Τρένου

Νικόλαος Λάππας
Α.Μ.: 03121098

Περιγραφή

Στην άσκηση αυτή, πρόκειται να ασχοληθούμε με τον έλεγχο ελάχιστης ενέργειας ενός ηλεκτρικού τρένου. Θα θεωρήσουμε ότι το τρένο τροφοδοτείται από μια μηχανή συνεχούς ρεύματος με σταθερή διέγερση και ελεγχόμενο ρεύμα τυμπάνου. Η μηχανή έχει την δυνατότητα αναγεννητικής πέδησης, με αποτέλεσμα να μπορεί να φρενάρει το τρένο κερδίζοντας ενέργεια. Τέλος, ασκεί ροπή στους τροχούς μέσω ενός συστήματος μετάδοσης κίνησης με αποτέλεσμα να κινείται το τρένο που θα μελετήσουμε.

Μοντελοποίηση

Για το συγκεκριμένο πρόβλημα, οι ηλεκτρικές εξισώσεις μόνιμης κατάστασης που περιγράφουν την λειτουργία της μηχανής έχουν την ακόλουθη μορφή:

$$V_a = RI_a + \omega(M_{af}I_f),$$

όπου:

- V_a η τάση του τυμπάνου,
- R η αντίσταση του τυλίγματος του τυμπάνου,
- ω η γωνιακή ταχύτητα της μηχανής, και
- M_{af} και I_f μπορούν να θεωρηθούν σταθερές, και αντιπροσωπεύουν την επαγωγή μεταξύ στάση και δρομέα και το ρεύμα του στάτη αντίστοιχα.

Η ροπή της μηχανής δίνεται από τον τύπο:

$$T = (M_{af}I_f)I_a.$$

Στο τρένο ασκείται η δύναμη της τριβής, η οποία είναι ανάλογη της ταχύτητας, καθώς, επίσης, και η δύναη αντίστασης λόγω του αέρα, η οποία είναι ανάλογη του τετραγώνου της ταχύτητας. Ανάγοντας όλες τις ροπές και τις δυνάμεις στον άξονα μιας ρόδας, καταλήγουμε στις ακόλουθες εξισώσεις κατάστασης που περιγράφουν την κίνηση του τρένου.

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -k_1x_2 - k_2x_2^2 + k_3u$$

όπου:

- x_1 είναι η θέση του τρένου,
- x_2 είναι η ταχύτητα του τρένου, και
- u είναι το ρεύμα τυμπάνου της μηχανής.

Βέλτιστος Έλεγχος

4. Καλούμαστε να επιλύσουμε το πρόβλημα βέλτιστου ελέγχου με κριτήριο κόστους

$$J = c_1(x_1(T) - x_{1f})^2 + c_2 x_2^2(T) + \int_0^T k_4 x_2 u + Ru^2 dt$$

θεωρώντας ότι $I_{min} \leq u \leq I_{max}$ και πως αρχικά ισχύει $x_1(0) = x_2(0) = 0$.

Λόγω της μορφής της συνάρτησης κόστους, συμπεραίνουμε ότι

$$\phi(x_1(T), x_2(T)) = c_1(x_1(T) - x_{1f})^2 + c_2 x_2^2(T).$$

Η *Hamiltonian* έχει την μορφή $H = L + p^T f(x, u)$, και άρα είναι:

$$H = k_4 x_2 u + Ru^2 + p_1 x_2 + p_2(-k_1 x_2 - k_2 x_2^2 + k_3 u)$$

Γνωρίζοντας ότι η βέλτιστη λύση u^* δίνεται από την σχέση $u^* = \frac{\partial H}{\partial u}$, προκύπτει:

$$u^* = \frac{-k_3 p_2 + k_4 x_2}{2R}, \text{ με } u \in [I_{min}, I_{max}]$$

Προχωρώντας με τον ορισμό τον διαφορικών εξισώσεων για το costate και τις αντίστοιχες συνοριακές συνθήκες, παίρνουμε τα ακόλουθα:

$$\dot{p} = -\frac{\partial H}{\partial x} \text{ και } \frac{\partial \phi}{\partial x} - p(T) = 0, \text{ δηλαδή,}$$

$$\begin{cases} \dot{p}_1 = 0 \\ \dot{p}_2 = -k_4 u + k_1 x_2 - p_1 + 2k_2 x_2 p_2 \end{cases}$$

Και

$$\begin{cases} p_1(T) = 2c_1(x_1(T) - x_{1f}) \\ p_2(T) = 2c_2 x_2(T) \end{cases}$$

Με αυτόν τον τρόπο καταφέραμε να υπολογίσουμε την βέλτιστη τροχιά που πρέπει να ακολουθήσει το σύστημά μας προκειμένου να ελαχιστοποιήσουμε την συνάρτηση κόστους. Βρήκαμε, λοιπόν, την βέλτιστη είσοδο κλειστού βρόχου.

5. Το παραπάνω πρόβλημα καλούμαστε να το προσομοιώσουμε με την χρήση του λογισμικού Matlab. Για τις σταθερές παραμέτρους δίνονται από την εκφώνηση ότι: $k_1 = 0.5, k_2 = 0.1, k_3 = 1, k_4 = 10, c_1 = c_2 = 1000, x_{1f} = 10, R = 0.3, I_{min} = -1, I_{max} = 2, T = 10$.

Τα αποτελέσματα που παίρνουμε είναι τα ακόλουθα:

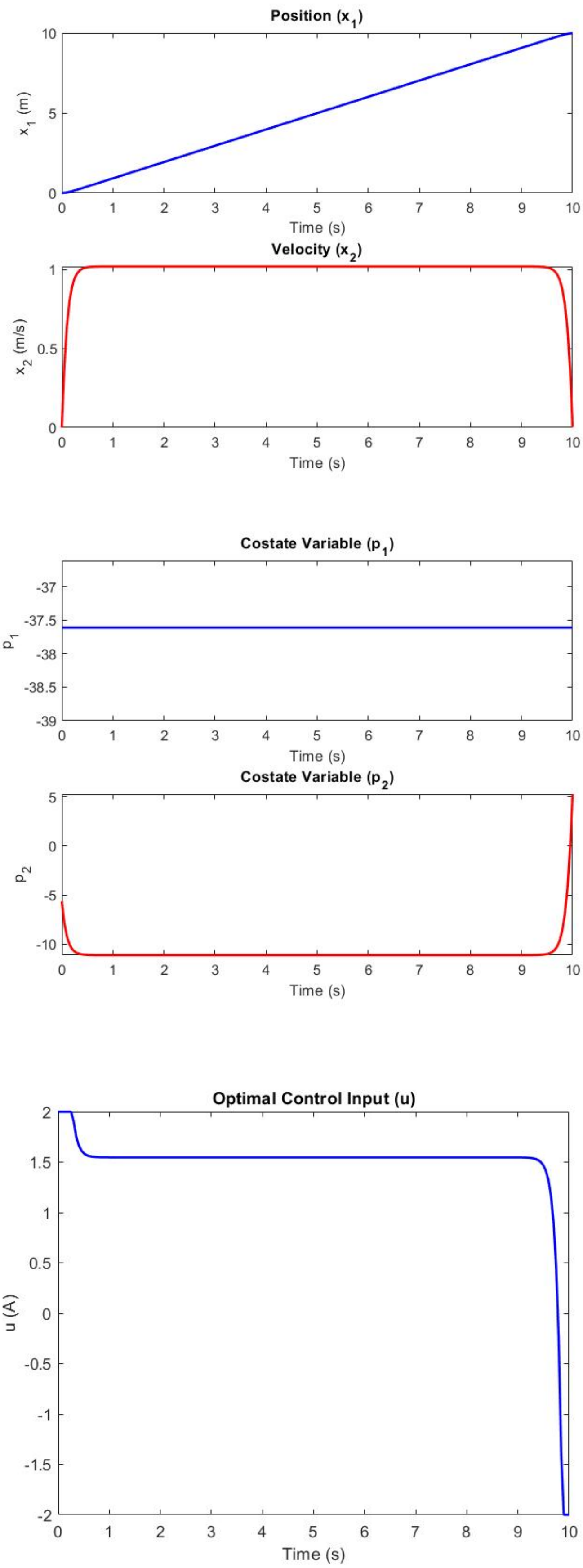


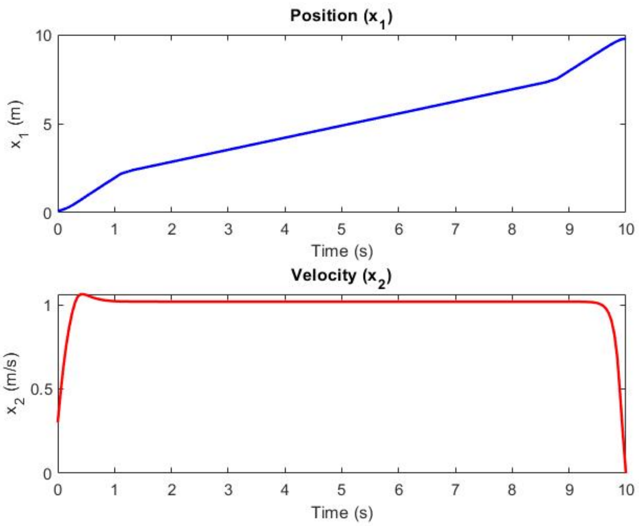
Figure 1: Closed Loop Optimal Controller

Να αναφέρουμε ότι νόμος ελέγχου έγινε με χρήση των συναρτήσεων bvp4c. Θα μπορούσαμε να είχαμε χρησιμοποιήσει και την bvp5c, η οποία είναι 5ου βαθμού collocation method με καλύτερη ακρίβεια λόγω του υψηλότερου βαθμού πολυωνύμων προσέγγισης, απλά είναι υπολογιστικά πιο βαριά συνάρτηση και για το πρόβλημά μας δεν αρκεστήκαμε στην πιο απλή, χωρίς αρνητικό αντίκτηπο στο αποτέλεσμα που πήραμε.

Να παρατηρήσουμε ότι η βέλτιστη είσοδος παίρνει απευθείας την μέγιστη δυνατή τιμή, ώστε να επιταχυνθεί αμέσως το τρένο, και μειώνεται μέχρι να φτάσει το τρένο στην επιθυμητή τελική θέση προκειμένου να μηδενιστή η ταχύτητα την χρονική εκείνη στιγμή. Το διάγραμμα που παίρνουμε μοιάζει με $'Bang - Coast - Bang'$, κάτι λογικό λόγω της ύπαρξης περιορισμών. Αρχικά, $'$Bang phase$'$, λαμβάνει την μέγιστη τιμή το $u = u_{max}$. Στην συνέχεια, $'$Coast phase$'$, η είσοδος u είναι σταθερή (εντός των ορίων $[I_{min}, I_{max}]$) και το σύστημα αφήνεται να κινηθεί με σταθερή ταχύτητα. Αυτή η φάση μειώνει την δαπανώμενη ενέργεια (το κόστος ελέγχου, λόγω του όρου Ru^2). Τέλος, $'$Bang phase$'$, η είσοδος επανέρχεται στην ελάχιστη επιτρεπτή τιμή αυτή τη φορά προκειμένου να φτάσει το τρένο στον τελικό προορισμό με βέλτιστο τρόπο.

6. Ζητείται να προσομοιώσουμε το σύστημα με την είσοδο που προέκυψε στο προηγούμενο ερώτημα, αλλά με ελαφρώς διαφορετικές αρχικές τιμές x_1 και x_2. Θα εισάγουμε, δηλαδή, την υπολογισμένη είσοδο u σαν ελεγκτή ανοιχτού βρόχου στο σύστημά μας. Περιμένουμε να μην έχουμε εξίσου καλά αποτελέσματα με πριν. Χρησιμοποιήσαμε σαν αρχική συνθήκη

$$[x_1(0), x_2(0)]^T = [0.1, 0.3]^T$$




```
Calculated cost for the control problem:
The calculated cost is: 1478.0083

Calculated cost for the new control problem:
The calculated new cost is: 1542.7774
```

Figure 2: Calculated cost for optimal and sub-optimal control

Όπως ακριβώς περιμέναμε, υπάρχει αύξηση στο κόστος για τον νέο έλεγχο που πραγματοποιήσαμε.

Στο συγκεκριμένο ερώτημα, σκοπός μας είναι να γραμμικοποιήσουμε το σύστημα γύρω από την βέλτιστη τροχιά προκειμένου να μειώσουμε το σφάλμα που προκαλούν οι μεταβολές των αρχικών τιμών. Θα θεωρήσουμε μια μικρή απόκλιση $y(t)$ από την βέλτιστη τροχιά και μια μικρή απόκλιση $v(t)$ στην είσοδο. Αυτές συνδέονται μέσω της σχέσης

$$\frac{d(x(t)+y)}{dt} = f(x(t)+y, u(t)+v),$$

η οποία καταλήγει στην γνωστή μας

$$\dot{y} = A(t)y + B(t)v$$

όπου,

$A(t) = \frac{\partial f}{\partial x}\Big|_{(x(t),u(t))}$ και $B(t) = \frac{\partial f}{\partial u}\Big|_{(x(t),u(t))}$.

7. Με στόχο να κρατήσουμε το σύστημα στην βέλτιστη τροχιά, πρέπει να προσδιορίσουμε το $v(y)$ που ελαχιστοποιεί το εκάστοτε τετραγωνικό σφάλμα. Στην περίπτωσή μας, το κριτήριο κόστους είναι

$$J_2 = 20y_1^2(T) + 20y_2^2(T) + \int_0^T 2y_1^2 + 2y_2^2 + v^2 dt$$

Από τις εξισώσεις του συστήματος μπορούμε να υπολογίσουμε τις μήτρες $A(t)$ και $B(t)$,

$$A(t) = \frac{\partial f}{\partial x}\Big|_t = \begin{bmatrix} 0 & 1 \\ 0 & -k_1 - 2k_2x_2(t) \end{bmatrix}$$

Και

$$B(t) = \frac{\partial f}{\partial u}\Big|_t = \begin{bmatrix} 0 \\ k_3 \end{bmatrix}$$

Η συνάρτηση κόστους J_2 έχει τετραγωνική μορφή,

$$J_2 = y^T(T)S_f y(T) + \int_0^T (y^T(t)Qy(t) + v(t)^2)dt$$

όπου,

$$S_f = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}, \text{και } Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Γνωρίζουμε ότι η είσοδος που ελαχιστοποιεί την παραπάνω εξίσωση κόστους είναι η $u(y) = -B^TPy$, όπου το μητρώο P (συμμετρικό) δίνεται από την επίλυση του διαφορικής εξίσωσης *Ricatti*.

$$\dot{P} + PA + A^TP - PBB^TP + Q, \text{και } P(T) = S_f$$

Επομένως, καταλήγουμε στα ακόλουθα:

$$\begin{cases} \dot{P}_{11} = k_3^2P_{12}^2 - 2 \\ \dot{P}_{12} = k_3^2P_{12}P_{22} - P_{11} - P_{12}(-k_1 - 2k_2x_2(t)) \\ \dot{P}_{22} = k_3^2P_{22}^2 - 2 - 2(P_{12} + P_{22}(-k_1 - 2k_2x_2(t))) \end{cases}$$

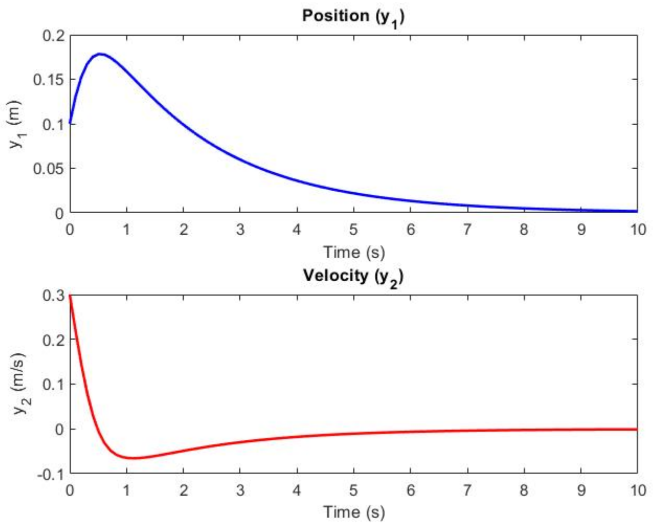
$$\text{και} \begin{cases} P_{11}(T) = 20 \\ P_{12}(T) = 0 \\ P_{22}(T) = 20 \end{cases}$$

Το παραπάνω σύστημα διαφορικών εξισώσεων μπορεί να λυθεί και προσδιορίζει το μητρώο P, και άρα προσδιορίζουμε πλήρως την βέλτιστη είσοδο κλειστού βρόχου.

$$v(y) = -k_3 \begin{bmatrix} P_{12}(t) & P_{22}(t) \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$$

8. Ζητείται να προσομοιωθεί το σύστημα με είσοδο ελέγχου $u(t) + v(y)$ στην περίπτωση που οι αρχικές τιμές είναι ελαφρά διαφορετικές.

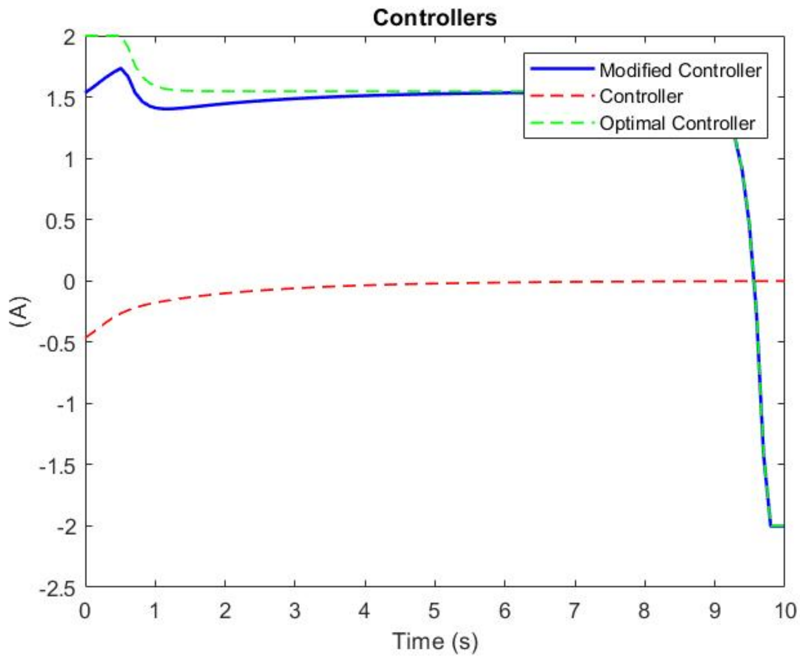
Χρησιμοποιούμε πάλι την συνάρτηση $bvp4c$ και παρόμοια μεθοδολογία όπως στις προηγούμενες περιπτώσεις. Αλλάξαμε, όμως, την συνάρτηση $ode_s ystem$ και την $boundary_c onditions$ προκειμένου αυτές να ανταποκρίνονται στα νέα δεδομένα του συστήματος, την εξίσωση Ricatti.




```
Calculated cost for the linearized control problem:
The calculated new cost is: 1481.4103
```

Παρατηρούμε, όπως και θέλαμε, ότι έχουμε μειωμένο κόστος τόσο σε σχέση με την προηγούμενη προσέγγιση όσο και με την αρχική-βέλτιστη. Η εξίσωση Ricatti μας δίνει μεγαλύτερη ευχέρεια και καλύτερη αποτελεσματικότητα στον έλεγχο του συστήματος, ιδίως τώρα που πρέπει να προσαρμόζεται στις αλλαγές τροχιάς και στους περιορισμούς που θέτουμε.

Ακολουθεί μια συνολική παρουσίαση των τριών ελεγκτών που μελετήσαμε:

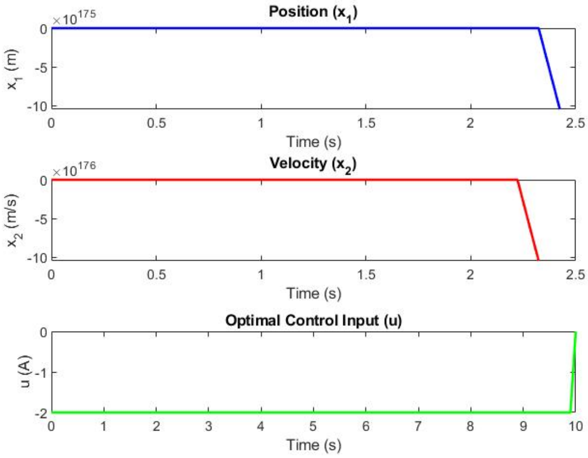


9. Για το συγκεκριμένο ερώτημα θέλουμε να εφαρμόσουμε δυναμικό προγραμματισμό και να επιλύσουμε έτσι το πρόβλημα βέλτιστου ελέγχου του τρένου. Ο δυναμικός προγραμματισμός, εν γένει, διασπά το πρόβλημα σε μικρότερα υποπροβλήματα και χρησιμοποιεί αυτές τις υπο-λύσεις προκειμένου να 'δομήσει' την συνολική (bottom-up approach).

Θα χρησιμοποιήσουμε, λοιπόν, την αρχή Bellman για την εύρεση της βέλτιστης λύσης, και ουσιαστικά θα διακριτοποιήσουμε τον χρόνο, t, σε N βήματα και θα κατασκευάσουμε ένα διακριτό πλέγμα για τις μεταβλητές κατάστασης x_1, x_2. Τα βήματα που εκτελεί ένας τέτοιος αλγόριθμος είναι:

- Διακριτοποίηση χρόνου σε N βήματα.
- Δημιουργία πλέγματος για τις μεταβλητές x_1, x_2.
- Backward iteration για τον υπολογισμό του κόστους.
- Υπολογισμός της τροχιάς.

Η προσομοίωση δεν μας έδωσε ικανοποιητικά αποτελέσματα. Ωστόσο ο κώδικας που υλοποιήσαμε παρατίθεται στο αρχείο .zip. Ακολουθεί μια όχι αντιπροσωπευτική εικόνα του αποτελέσματος που πήραμε.



Ακολουθεί ο κώδικας για τα ερωτήματα 4 έως και 8.

```
function optimal_train_control
% Constants
k1 = 0.5; k2 = 1.0; k3 = 1.0; k4 = 10;
c1 = 1000; c2 = 1000;
x1f = 10;
R = 0.3;
I_min = -2; I_max = 2;
T = 10; % Final time

% Initial guess for Boundary Value Problem (BVP)
solinit = bvpinit(linspace(0, T, 100), @init_guess);

% Solve the BVP with a 4th order collocation method (bvp4c)
sol = bvp4c(@ode_system, @boundary_conditions, solinit);

% Extract solution
t = sol.x;
%y = sol.y;
x1 = sol.y(1, :); % Position
x2 = sol.y(2, :); % Velocity
p1 = sol.y(3, :);
p2 = sol.y(4, :);

% Optimal solution
len = length(t);
u_opt = zeros(1, len);

for i = 1:len
    u_opt(i) = opt_controller(p2(i), x2(i));
end

% Plot results for State Variables
figure;
subplot(2, 1, 1);
plot(t, x1, 'b', 'LineWidth', 1.5);
title('Position (x_1)');
xlabel('Time (s)');
ylabel('x_1 (m)');

```

```
    subplot(2, 1, 2);
    plot(t, x2, 'r', 'LineWidth', 1.5);
    title('Velocity (x_2)');
    xlabel('Time (s)');
    ylabel('x_2 (m/s)');

    % Plot results for Co-State Variables
    figure;
    subplot(2, 1, 1);
    plot(t, p1, 'b', 'LineWidth', 1.5);
    title('Costate Variable (p_1)');
    xlabel('Time (s)');
    ylabel('p_1');

    subplot(2, 1, 2);
    plot(t, p2, 'r', 'LineWidth', 1.5);
    title('Costate Variable (p_2)');
    xlabel('Time (s)');
    ylabel('p_2');

    figure;
    plot(t, u_opt, 'b', 'LineWidth', 1.5);
    title('Optimal Control Input (u)');
    xlabel('Time (s)');
    ylabel('u (A)');

    % Display cost for the output
    disp(' ');
    disp('Calculated cost for the control problem:');

    state_difference = (x1(len) - x1f);
    control_signal = u_opt;
    control_penalty = trapz(k4 * x2 .* control_signal + R * (control_signal.^2));

    J = c1 * state_difference^2 + c2 * x2(len)^2 + control_penalty;
    fprintf('The calculated cost is: %.4f\n', J);


    % ----------------- For question 6 ------------------
    x1_new = zeros(1, len);
    x2_new = zeros(1, len);
    x1_dot_new = zeros(1, len);
    x2_dot_new = zeros(1, len);

    dt = T / len;
    x1_new(1) = 0.1;
    x2_new(1) = 0.3;

    for i = 1:len
        x1_dot_new(i) = x2_new(i);
        x2_dot_new(i) = - k1 * x2_new(i) - k2 * x2_new(i)^2 + k3 * u_opt(i);
        if i < len
            x1_new(i + 1) = x1_new(i) + x1_dot_new(i) * dt;
            x2_new(i + 1) = x2_new(i) + x2_dot_new(i) * dt;
        end
    end
```

```

% Plot results for State Variables
figure;
subplot(2, 1, 1);
plot(t, x1_new, 'b', 'LineWidth', 1.5);
title('Position (x_1)');
xlabel('Time (s)');
ylabel('x_1 (m)');

subplot(2, 1, 2);
plot(t, x2_new, 'r', 'LineWidth', 1.5);
title('Velocity (x_2)');
xlabel('Time (s)');
ylabel('x_2 (m/s)');


% Display new cost for the output
disp(' ');
disp('Calculated cost for the new control problem:');

state_difference = (x1_new(len) - x1f);
control_signal = u_opt;
control_penalty = trapz(k4 * x2_new .* control_signal + R * (control_signal.^2));

J_new = c1 * state_difference^2 + c2 * x2_new(len)^2 + control_penalty;
fprintf('The calculated new cost is: %.4f\n', J_new);
% -------------------------------------------------

% ----------------- For question 8 ------------------
estimation = polyfit(t, x2, 5); % Linearization with polynomials of 5th grade
options = bvpset('RelTol', 1e-6, 'AbsTol', 1e-8); % Increase tolerance
solinit_Ricatti = bvpinit(linspace(0, T, 100), [1;1;1]);
sol_Ricatti = bvp4c(@ode_system_Ricatti, @boundary_conditions_Ricatti, solinit_Ricatti, options);

t = sol_Ricatti.x;
p11 = sol_Ricatti.y(1, :);
p22 = sol_Ricatti.y(2, :);
p12 = sol_Ricatti.y(3, :);

% Optimal solution
v_opt = zeros(1, len);

y1_new = zeros(1, len);
y2_new = zeros(1, len);
y1_dot_new = zeros(1, len);
y2_dot_new = zeros(1, len);

dt = T / len;
y1_new(1) = 0.1;
y2_new(1) = 0.3;

for i = 1:len
    v_opt(i) = v_optimal(y1_new(i),y2_new(i),p12(i),p22(i));
```

```
        A22 = - k1 - 2 * k2 * x2_new(i);

        y1_dot_new(i) = y2_new(i);
        y2_dot_new(i) = A22 * y2_new(i) + k3 * v_opt(i);

        if i < len
            y1_new(i+1) = y1_new(i) + y1_dot_new(i) * dt;
            y2_new(i+1) = y2_new(i) + y2_dot_new(i) * dt;
        end
    end

    % Plot results for State Variables
    figure;
    t = linspace(0, T, length(y1_new));

    subplot(2, 1, 1);
    plot(t, y1_new, 'b', 'LineWidth', 1.5);
    title('Position (y_1)');
    xlabel('Time (s)');
    ylabel('y_1 (m)');

    subplot(2, 1, 2);
    plot(t, y2_new, 'r', 'LineWidth', 1.5);
    title('Velocity (y_2)');
    xlabel('Time (s)');
    ylabel('y_2 (m/s)');

    v_opt_modified = u_opt + v_opt;

    figure;
    plot(t, v_opt_modified, 'b', 'LineWidth', 1.5);
    hold on;
    plot(t, v_opt, 'r--', 'LineWidth', 1);
    plot(t, u_opt, 'g--', 'LineWidth', 1);
    title('Controllers');
    xlabel('Time (s)');
    ylabel('(A)');
    legend('Modified Controller', 'Controller', 'Optimal Controller');

    % Display new cost for the output
    disp(' ');
    disp('Calculated cost for the linearized control problem:');

    state_difference = (x1_new(len) - x1f);
    control_signal = v_opt_modified;
    control_penalty = trapz(k4 * x2_new .* control_signal + R * (control_signal.^2));

    J2 = c1 * state_difference^2 + c2 * x2_new(len)^2 + control_penalty;
    fprintf('The calculated new cost is: %.4f\n', J2);
    % --------------------------------------------------

    % NESTED FUNCTIONS

    % Nested function for optimal controller
```

```
    function u = opt_controller_nobounds(p2, x2)
        u = -(k3 * p2 + k4 * x2)/(2 * R);
    end

    % Nested function for optimal controller (set u bounds)
    function u = opt_controller(p2, x2)
        u = -(k3 * p2 + k4 * x2)/(2 * R);
        if u < I_min
            u = I_min;
        elseif u > I_max
             u = I_max;
        end
    end % With that function, when calling ode_system to solve bvp4c we encountered singular Jacobian

    % Nested function for ODE system
    function state = ode_system(t, y)
        x1 = y(1); x2 = y(2); p1 = y(3); p2 = y(4);

        u = opt_controller_nobounds(p2, x2);

        x1_dot = x2;
        x2_dot = - k1 * x2 - k2 * x2^2 + k3 * u;
        p1_dot = 0;
        p2_dot = - k4 * u + k1 * x2 - p1 + 2 * k2 * x2 * p2;

        state = [x1_dot; x2_dot; p1_dot; p2_dot];
    end

    % Nested function for boundary conditions
    function res = boundary_conditions(y0, yfinal)
        res = [
            y0(1);
            y0(2);
            yfinal(3) - 2 * c1 * (yfinal(1) - x1f);
            yfinal(4) - 2 * c2 * yfinal(2)
        ];
    end

    % Nested function for initial guess
    function guess = init_guess(t)
        guess = [1; 1; 1; 1];
    end

    % --------- Extra Nested Functions for question 8 ----------
    function P = ode_system_Ricatti(t, y)
        p11 = y(1); p22 = y(2); p12 = y(3);

        x2 = polyval(estimation, t);
        A22 = - k1 - 2 * k2 * x2;

        p11_dot = k3^2 * p12^2 - 2;
        p22_dot = k3^2 * p22^2 - 2 - 2 * (p12 + A22 * p22);
        p12_dot = k3^2 * p12 * p22 - p11 - A22 * p12;

        P = [p11_dot; p22_dot; p12_dot];
    end

```

```
    % P(T) = S
    function res = boundary_conditions_Ricatti(y0, yfinal)
        res = [
        yfinal(1) - 20; % Boundary for p11
        yfinal(2) - 20; % Boundary for p22
        yfinal(3)       % Boundary for p12
        ];
    end

    function v = v_optimal(y1, y2, p12, p2)
        v = -k3 * (p12 * y1 + p2 * y2);
    end
    % -------------------------------------------------
end
```

Βιβλιογραφία

1. ΒέλτιστοςΈλεγχοςΤρένου.pdf
2. Bryson and Ho "Applied Optimal Control" Ch. 7
3. optimal_control_problems.pdf