Silesian University
of Technology

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS
AND COMPUTER SCIENCE

Advanced Optimization Methods

Constrained dynamic optimization (penalty methods)

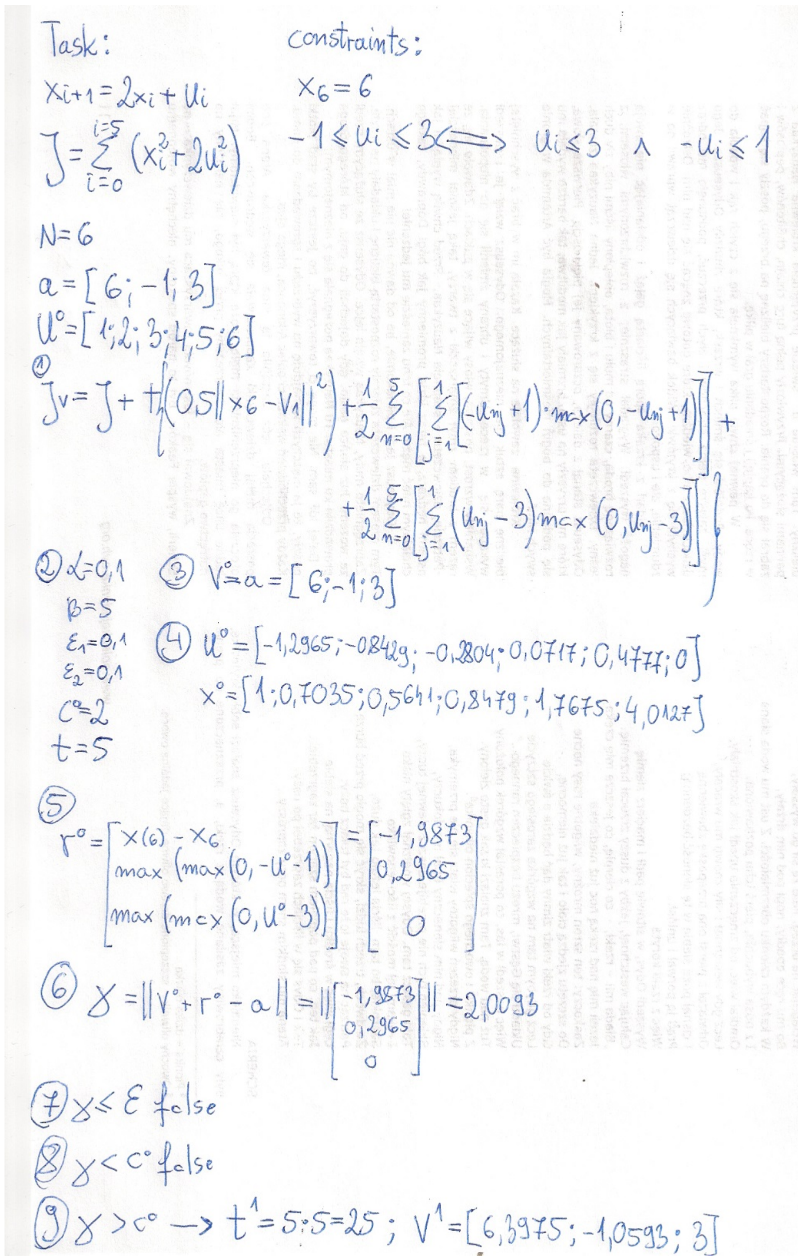
Łukasz Kania, Szymon Zosgórnik
Year 1, semester 1, Macro Data Science

June 1, 2020

1 Task

The task was to determine optimal control for a system described by state equation, with given x_0 and constraints. The equations are specified in the instruction in problem number 3. Selected x_6 parameter was $x_6 = 6$. In accordance with Professor's e-mail, the constraint $x_3 = 5$ was removed.

2 Hand-written part of solution


Task:

$x_{i+1} = 2x_i + u_i$

$J = \sum_{i=0}^{i=5} (x_i^2 + 2u_i^2)$

constraints:

$x_6 = 6$

$-1 \leq u_i \leq 3 \iff u_i \leq 3 \wedge -u_i \leq 1$

$N = 6$

$a = [6; -1; 3]$

$U^\circ = [1; 2; 3; 4; 5; 6]$

(1) $J_v = J + t_h \Big(0{,}5 \|x_6 - V_1\|^2 \Big) + \frac{1}{2} \sum_{n=0}^{5} \Big[\sum_{j=1}^{1} \big[(-u_{nj} + 1) \cdot \max(0, -u_{nj} + 1) \big] \Big] + \frac{1}{2} \sum_{n=0}^{5} \Big[\sum_{j=1}^{1} (u_{nj} - 3) \max(0, u_{nj} - 3) \Big]$

(2) $\alpha = 0{,}1$, $\beta = 5$, $\varepsilon_1 = 0{,}1$, $\varepsilon_2 = 0{,}1$, $c^\circ = 2$, $t = 5$

(3) $V^\circ = a = [6; -1; 3]$

(4) $U^\circ = [-1{,}2965; -0{,}8429; -0{,}2804; 0{,}0717; 0{,}4777; 0]$

$x^\circ = [1; 0{,}7035; 0{,}5641; 0{,}8479; 1{,}7675; 4{,}0127]$

(5) $r^\circ = \begin{bmatrix} x(6) - x_6 \\ \max(\max(0, -U^\circ - 1)) \\ \max(\max(0, U^\circ - 3)) \end{bmatrix} = \begin{bmatrix} -1{,}9873 \\ 0{,}2965 \\ 0 \end{bmatrix}$

(6) $\gamma = \|V^\circ + r^\circ - a\| = \left\| \begin{bmatrix} -1{,}9873 \\ 0{,}2965 \\ 0 \end{bmatrix} \right\| = 2{,}0093$

(7) $\gamma < \varepsilon$ false

(8) $\gamma < c^\circ$ false

(9) $\gamma > c^\circ \rightarrow t^1 = 5 \cdot 5 = 25$; $V^1 = [6{,}3975; -1{,}0593; 3]$

Figure 1: Hand written part of solutions with selected parameters

Selected initial values of control vector: $u_0 = 1, u_1 = 2, u_2 = 3, u_3 = 4, u_4 = 5, u_5 = 6$

3 Results

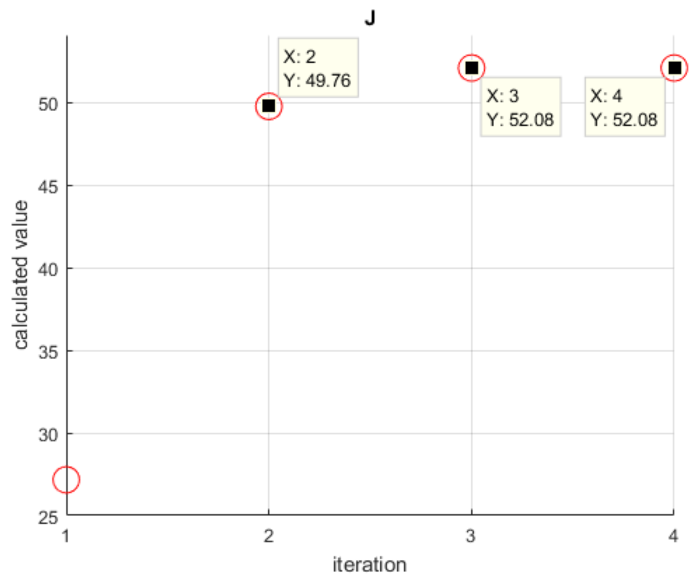


Figure 2: Results obtained for performance index J

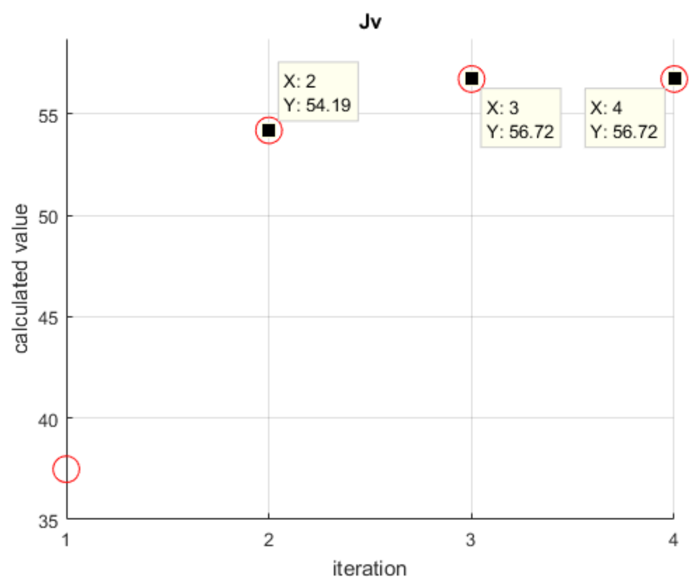


Figure 3: Results obtained for penalty functional

```
X values, x(1): 1, x(2): 0.82318, x(3): 0.74425, x(4): 1.2235, x(5): 2.6204, x(6): 5.9826
U values, u(1): -1.1768, u(2): -0.90211, u(3): -0.26497, u(4): 0.17336, u(5): 0.74178, ...
    u(6): -1.9861e-05
```

4 Conclusions

The algorithm as usual needed small amount of iterations to achieve the solution. As can be seen the constraints are not fully satisfied, the u_1 still violates them therefore the value of penalty functional is still greater than the function being minimized. The algorithm has stopped after meeting both stop criteria - achieving absolute value of consecutive penalty functionals being subtracted smaller than given accuracy and calculated gamma also being smaller than given accuracy. As expected, the value of found x_6 is within given precision, which was set to 0.1.

5 Source code

main.m

```
clc
clear all
%%%%%%
x_1 = 1;
x_6 = 6;
n=6;
%%%%%% defining parameters and initial values%%%%%%
alfa = 0.1;
beta = 5;
epsilon = 0.1;
epsilon_2 = 0.1;
c = 2;
t = 5;
a = [x_6, -1, 3];                                           %values of constraints, x_6=7, ...
    -1<u_i<3
v = a;                                                      %step 3
u0 = [1,2,3,4,5,6]                                          %random values of initial control
u=u0
i = 1;

while true
    Jv_function = @(u)calculate_penalty(x_1, u, n, v, t);   %step 1
    u = fminsearch(Jv_function, u);                         %finding un, step 4
    [Jv(i), x, J(i)] = calculate_penalty(x_1, u, n, v, t);  %finding xn, step 4
    r = calculate_r(x(6), v, u);                            %step 5
    gamma = norm(v+r-a);                                    %step 6

    if gamma > epsilon                                      %step 7
        if gamma < c                                        %step 8
            v= a-r;
            c = alfa*beta;
        else                                                %step 9
            t = beta*t;
            v = a - r / beta;
        end
        gamma = norm(v+r-a);
    end

    if (i >1)                                               %step 10
        j_jv = abs(prev_Jv - Jv(i))
        if gamma ≤ epsilon && j_jv < epsilon_2
            break
        end
    end

    prev_Jv=Jv(i);
    i=i+1;
```

```
end

visualize_results(x,u,J,Jv)
```

calculate_penalty.m

```
function [Jv, x, J] = calculate_penalty(x0 , u, n, v, t)
x = zeros (1, n);
x(1) = x0;
for i = 2:n
    x(i) = 2*x(i-1) + u(i-1);
end
J = sum(x.^2 + 2*u.^2);
R_1 = 0.5 * norm(x(6) - v(1))^2; %v(1) corresponds to x_6, etc
R_2 = sum(sum((-u + v(2)).* max(0, -u + v(2)))) ;
R_3 = sum(sum((u - v(3)).* max(0, u - v(3)))) ;
Jv = J + t*(R_1 + R_2 + R_3);
end
```

calculate_r.m

```
function r = calculate_r(x6,v,u)
    r = [0,0,0];
    r(1) = x6 - v(1);                      %for constraint x6 = 1
    r(2) = max(max(0, -u + v(2)));    %for u_i ≤ 14.5
    r(3) = max(max(0, u - v(3)));     %for u_i ≥ 14.5
end
```

visualize_results.m

```
function r = visualize_results(x, u, J, Jv)
display_x = ['X values, x(1): ', num2str(x(1)), ...
                     ', x(2): ', num2str(x(2)), ...
                     ', x(3): ', num2str(x(3)), ...
                     ', x(4): ', num2str(x(4)), ...
                     ', x(5): ', num2str(x(5)), ...
                     ', x(6): ', num2str(x(6))];

display_u = ['U values, u(1): ', num2str(u(1)), ...
                     ', u(2): ', num2str(u(2)), ...
                     ', u(3): ', num2str(u(3)), ...
                     ', u(4): ', num2str(u(4)), ...
                     ', u(5): ', num2str(u(5)), ...
                     ', u(6): ', num2str(u(6))];
disp(display_x)
disp(display_u)
x_vector = [1:length(Jv)];
figure
scatter(x_vector, Jv, 200, 'red')
grid on
title("Jv")
xlabel("iteration")
ylabel("calculated value")
ax = gca;
ax.XTick = x_vector;
ax.YLim(2) = max(Jv)+2;

figure
scatter (x_vector, J, 200, 'red')
grid on
title("J")
```

```
xlabel("iteration")
ylabel("calculated value")
ax = gca;
ax.XTick = x_vector;
ax.YLim(2) = max(J)+2;
end
```