Advanced Optimization Methods

Report

Constrained Dynamic Optimization

Authors:
Piotr Pawełko
Krzysztof Miler

Laboratory date:
09.06.2022

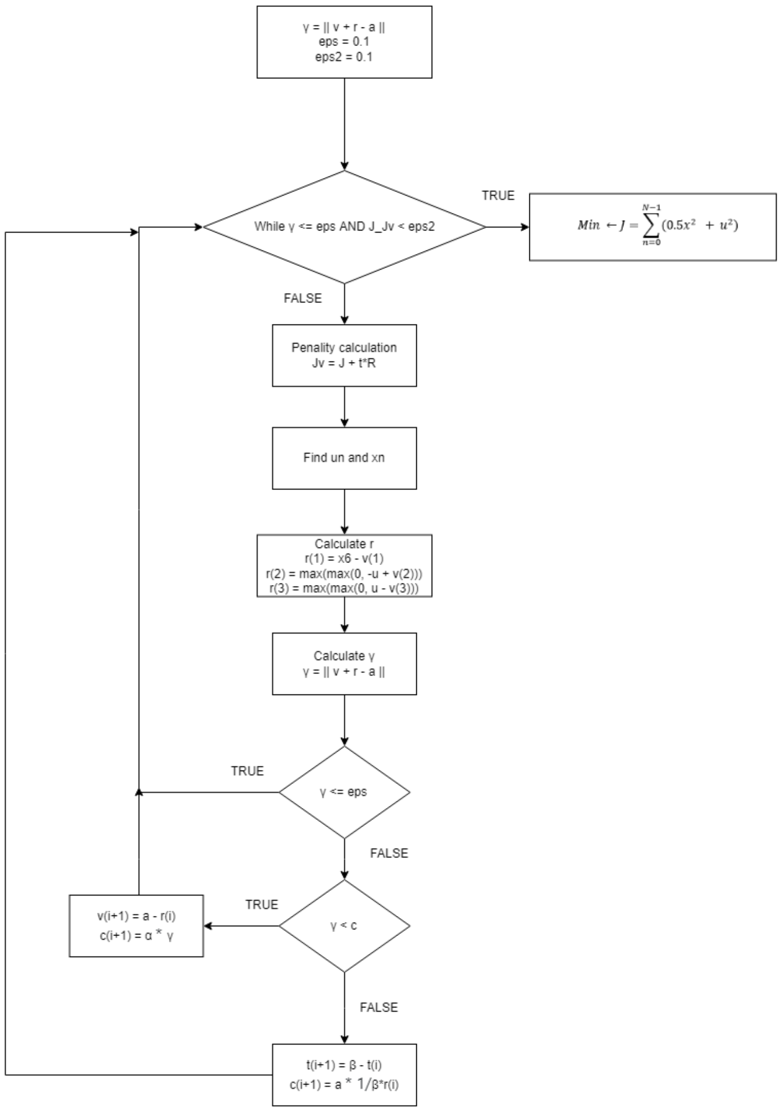
1. Problem statement

$$Min \leftarrow J = \sum_{n=0}^{N-1} (0.5 * x^2 \ + \ u^2)$$

$$x_{n+1} = x_n \ + u_n$$

Parameters:
N = 6
X_0 = 1
u = [3 ,2, 5, 4, 1, -1]
α = 0.1
β = 5
c = 2
t = 2
u_{min} = -2
u_{max} = 2

2. Calculation method



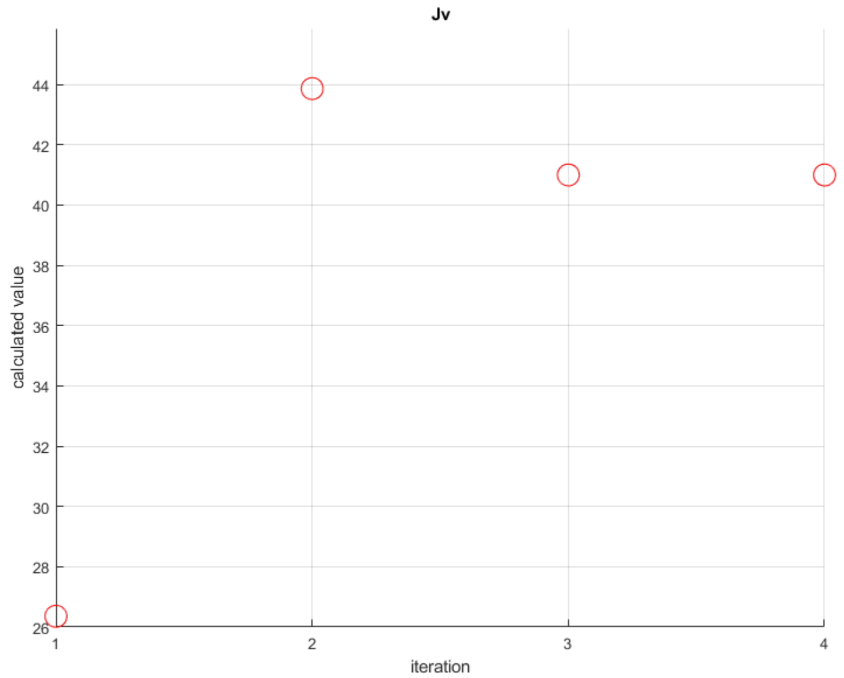
3. Test

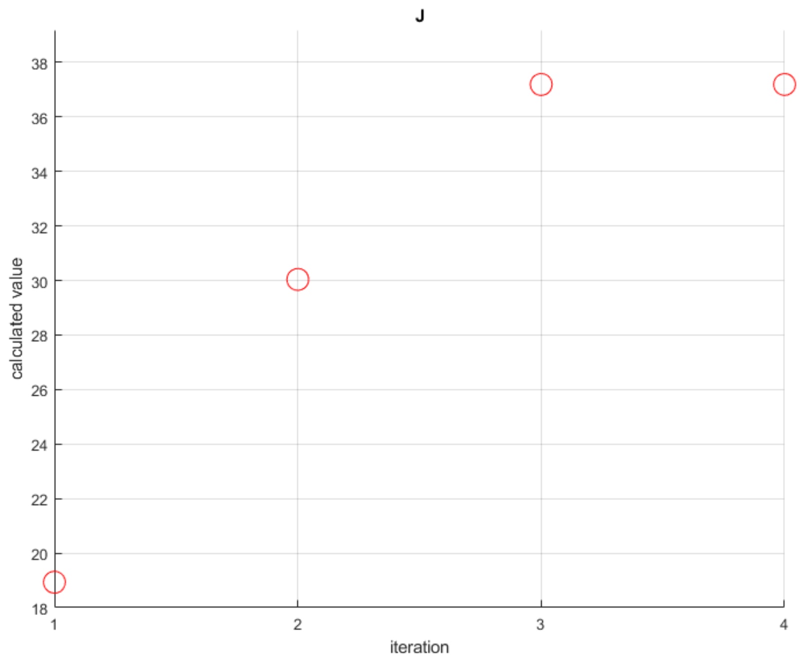
i	x(i)
1	1
2	0.8605
3	1.1511
4	2.0172
5	3.8920
6	5.9244

i	u(i)
1	-0.1939
2	0.29062
3	0.8662
4	1.8748
5	2
6	0

Iteration	J
1	18.9313
2	30.0386
3	37.1905
4	37.1905

Iteration	Jv
1	26.3552
2	43.8649
3	40.9990
4	40.9990





4. Conclusion

The aim of this exercise was to introduce method of dynamical optimization for problems in which additional constraints either on control or state variables are imposed. The most important of these constraints are those concerning control variables. The algorithm used in this exercise involves changing the particular penalty functionals with each iteration and modifying the values of parameters of the algorithm. The change of the penalty functional was done by means of changing the penalty shift *v* and the weight coefficient *t*. The algorithm requires relatively small number of iterations before finding solution.

5. Code

```
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, -2, 2];
v = a;                                                      %step 3
u0 = [3 ,2, 5, 4, 1, -1];                                          %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)

function [Jv, x, J] = calculate_penalty(x0 , u, n, v, t)
x = zeros (1, n);
x(1) = x0;
for i = 2:n
    x(i) = x(i-1) + u(i-1);
end
J = sum(0.5*x.^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

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

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
```