WARSAW UNIVERSITY OF TECHNOLOGY
Faculty of Mathematics
and Information Science

Optimization in Data Analysis

Project

“CONSTRAINED DYNAMIC OPTIMIZATION”
Moving penalty methods’

By

Amir Ali

Summer Semester 2021

INTRODUCTION

The aim of this Project is to introduce methods 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, as usually in practical applications control is bounded. Similarly, large values of state variables are also not feasible, as they might represent e.g. number of items that should be stored in a storage system of a limited capacity or value of electrical current that would result in burning the circuit etc. The framework introduced in this exercise facilitates dealing with various types of the constraints by adding appropriate penalty functions to the original functional that is to be minimized.

Problem statement

Assuming we have considered a process described by the following:

Difference state equation: $\mathbf{x_{n+1} = f_n (x_n; u_n)}$

Performance index given: $\mathbf{J = \Sigma X^{N-1}_{n=0} L_n(x_n; u_n)}$

To be minimized is where
$\mathbf{x_n}$ – k {dimensional state vector in a time moment n,
$\mathbf{u_n}$ – r {dimensional control vector in a time moment n,
N - optimization horizon,
$\mathbf{L_n}$ - partial cost function.

Assume the initial state $\mathbf{x_0}$ and the optimization horizon N are given. Additional constraints may be imposed on control and state variables, concerning their values at particular steps **n**. They can take either an equality or inequality forms such as $\mathbf{h_n(x_n; u_n)}$ **= / <= a**

Moving penalty functional

Having defined the standard forms of the penalty functionals, we introduce the algorithm that will be used in the project. It involves changing the particular penalty functionals with each iteration and modifying the values of parameters, in those steps that necessitate such changes. Change of the penalty functional is be done by changing the penalty shift v and its weight t.

In each iteration, after choosing appropriate v and t values, the first step is to find the solution of the optimization problem as unconstrained problem (with state equation as only constraint) which can be solved by means of any direct method. The found solution is denoted by u and its corresponding state trajectory by x. Next, we calculate how much constraints are violated by the found solution. Taking into account the above considerations, the algorithm may be summarized in the following steps:

- Define the modified penalty functional J.
- Choose the values for the following coefficients:
- Substitution and initialization
- Find initial the solution minimizing state equation.
- Apply stopping criteria followed by subsequent calculation

PROBLEM

Here it is asked to solve this constrained optimization problem using moving penalty function

$$J = \Sigma_{i=0}{}^{6}\ (x_i^2 + 3u_i^2)$$

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

$$x_0 = 120$$

$$x_7 = 10$$

$$|u_i| \leq 5$$

$$x_3 = 40$$

$$u_3 = 4$$

$$\varepsilon = 0.1$$

Modified J function using penalty functions

$$J = \Sigma_{i=0}{}^{6}\ (x_i^2 + 3u_i^2) + \Big[\ t * (|x_7 - 10|2 + \Sigma_{i=0}{}^{6}\ (u_i - 5) * \max\ (0, u_i - 5) + (-ui - 5) * \max\ (0, -u_i - 5) + |u_3 - 4|^2 + |x_3 - 40|^2\ \Big]$$

RESULT

A-

Parameterization implemented in the script

```
clear all;
close all;
clc;

%% Parameterisation

% Rate of conv % ? > 1       % Shift    % Weight        % Constraint
alfa = 0.1;    beta = 2;     c = 1;     t = 1;          xn = 10;

% Gamma's                    % Treshold (e1/e2)         % Fit initial
Gama1 = 0; Gama2 = 10;       e1 = 0.1; e2 = 0.1;        Jbest = 0;

%%  Subtitution
a = [20 30 40 5 5 5 5 5 5 5];
v1= a(1); v2 = a(2); v3 = a(3);
v4= a(4:10); v = [v1 v2 v3 v4];

%% Initialisation
u = [5 5 5 5 5 5 5];                      % Initial Solution
```

```matlab
x = [120 0 0 40 0 0 0];                       % State trajectory
it = 0;
%% Calculation and computation
while abs(Gama1-Gama2) > e1                      % First Condition check
    Gama1 = Gama2;
    it = it+1
                                         % Performance index J fitness function
    Fit = @(u)    (sum(x.^2+ 3*u.^2)...
               + (t*((((x(1) + sum(u))-v(1))^ 2)...
               + sum((u-v(4:10)).* max(0, u-v(4:10))...
               + (-u-v(4:10)).* max(0, -u-v(4:10)))...
               + ((u(4)-v(3))^2) + (((x(1) + sum(u(1:3))-v(2))^2)))));

    [u,Jx] = fminsearch(Fit,u);                  % Local minimum search of J

    for j = 1:7
         if j < 7                                % Computing x values up to x6
            x(j+1) = x(j)+u(j);
         end
         if  j == 7
             xn = x(j)+u(j);                     % Assigning given value of x7
         end
    end

    for j = 1:7
        b =  u(j)-5;  c = -u(j)-5;
        r4(j) = max([0 (u(j)-5) (-u(j)-5)]);     % Condition of Ui <= 5
    end
    r = [xn-a(1),x(4)-a(2),u(4)-a(3),r4];       % Next violation values

    for j = 1:7
        a4(j) = v(j+3)+r4(j);
    end
    ai= [v(1)+r(1),v(2)+r(2),v(3)+r(3),a4]; % Next matrix of constraints

    if Gama2 > e1 && Gama2 < c                   % Stopping critiria check
        for j = 1:10
            v(j) = a(j)-r(j);                    % Update of v matrix
        end
            c = alfa * Gama2;                    % Update of the penalty shift
    end

    if Gama2 > e1 && Gama2 >= c                  % Stopping critiria check
        t = beta*t;                              % Weight coefficient
        for j = 1:10
            v(j) = a(j)-(1/beta)*r(j);           % Update of v matrix
        end
    end

    if abs(Jbest-Jx) < e1 || Gama2 < e1          % Stopping criteria to satisfy
      break                                      % constraints.
    else
        Jbest = Jx;                              % Update of Performance index
    end

    Gama2 = norm(ai-a);                          % First stopping criteria's
                                               % Check (While loop)
    [x;u]'
    Jx
end
```

According to the results obtained from the original given data, we observed that this optimization problem has got something not right with the constraints mentioned. We see that the stopping criteria ends the algorithm without some constraints not being satisfied (Which is to obtain the value of $X_3 = 40$ in third state and its control value at this stage $U_3 = 4$) because of minimizing of the cost function J (Performance index) and weighting the violation of constraints. Which may be due to the fact of strong equality constraint for this stage of the control.

Iterations	**Gamma**	**Index J**	**States(x_i)**	**Control (u_i)**
1	1	2.3332e+04	120.0000	-17.4265
			102.5735	-17.4265
			85.1470	-17.4266
			67.7205	3.2537
			70.9741	-7.9963
			62.9779	-7.9962
			54.9817	-7.9963
2	63.1785	6.7579e+04	120.0000	-23.6918
			96.3082	-23.6919
			72.6163	-23.6918
			48.9245	11.6329
			60.5574	-10.4639
			50.0935	-10.4640
			39.6295	-10.4640
3	26.0179	6.7276e+04	120.0000	-23.4828
			96.5172	-23.4829
			73.0343	-23.4828
			49.5515	14.3418
			63.8933	-10.6830
			53.2103	-10.6830
			42.5273	-10.6830
4	24.4759	8.9368e+04	120.0000	-24.9322
			95.0678	-24.9322
			70.1355	-24.9323
			45.2033	16.6796
			61.8828	-11.4202
			50.4626	-11.4203
			39.0423	-11.4202
5	23.9341	1.2812e+05	120.0000	-24.9752
			95.0248	-24.9753
			70.0495	-24.9752
			45.0743	17.7701
			62.8444	-11.5690
			51.2754	-11.5690
			39.7064	-11.5690

B- The same problem with weak constraints is considered here

```
clear all;
close all;
clc;

%% Parameterisation

% Rate of conv % ? > 1       % Shift    % Weight        % Constraint
alfa = 0.1;    beta = 4;     c = 1;     t = 1;          xn = 10;
% Gamma's                    % Treshold (e1/e2)         % Fit initial
Gama1 = 0; Gama2 = 1;        e1 = 0.1; e2 = 0.1;        Jbest = 0;

%%  Subtitution
a = [20 30 4 5 5 5 5 5 5 5];
v1= a(1); v2 = a(2); v3 = a(3);
v4= a(4:10); v = [v1 v2 v3 v4];

%% Initialisation
u = [5 5 5 5 5 5 5];                        % Initial Solution
x = [20 0 0 50 0 0 0];                    % State trajectory
it = 0;

%% Calculation and computation
while abs(Gama1-Gama2) > e1                     % First Condition check
    Gama1 = Gama2
    it = it+1
                                  % Performance index J fitness function
    Fit = @(u)    (sum(x.^2+ 3*u.^2)...
                + (t*((((x(1) + sum(u))-v(1))^ 2)...
                + sum((u-v(4:10)).* max(0, u-v(4:10))...
                + (-u-v(4:10)).* max(0, -u-v(4:10)))...
                + ((u(4)-v(3))^2) + (((x(1) + sum(u(1:3))-v(2))^2)))));

    [u,Jx] = fminsearch(Fit,u);                 % Local minimum search of J

    for j = 1:7
         if j < 7                               % Computing x values up to x6
            x(j+1) = x(j)+u(j);
         end
         if  j == 7
             xn = x(j)+u(j);                    % Assigning given value of x7
         end
    end

    for j = 1:7
        b =  u(j)-5;   c = -u(j)-5;
        r4(j) = max([0 (u(j)-5) (-u(j)-5)]);      % Condition of Ui <= 5
    end
    r = [xn-a(1),x(4)-a(2),u(4)-a(3),r4];     % Next violation values

    for j = 1:7
        a4(j) = v(j+3)+r4(j);
    end
    ai= [v(1)+r(1),v(2)+r(2),v(3)+r(3),a4]; % Next matrix of constraints

    if Gama2 > e1 && Gama2 < c                  % Stopping critiria check
        for j = 1:10
            v(j) = a(j)-r(j);                   % Update of v matrix
        end
            c = alfa * Gama2;                   % Update of the penalty shift
```

```
    end

    if Gama2 > e1 && Gama2 >= c                 % Stopping critiria check
        t = beta*t;                              % Weight coefficient
        for j = 1:10
            v(j) = a(j)-(1/beta)*r(j);           % Update of v matrix
        end
    end

    if abs(Jbest-Jx) < e1 || Gama2 < e1          % Stopping criteria to satisfy
      break                                      % constraints.
    else
        Jbest = Jx;                              % Update of Performance index
    end

    Gama2 = norm(ai-a);      % First stopping criteria's check (While loop)
    [x;u]'
    Jx
end
```

Iterations	Gamma	Index J	States(x_i)	Control (u_i)
1	1	2.9751e+03	20.0000	1.3030
			21.3030	1.3031
			22.6060	1.3030
			23.9091	0.4546
			24.3636	-0.7273
			23.6364	-0.7273
			22.9091	-0.7273
2	7.3777	3.8597e+03	20.0000	2.5472
			22.5472	2.5472
			25.0944	2.5472
			27.6416	1.6661
			29.3077	-2.6276
			26.6802	-2.6275
			24.0527	-2.6276
3	1.8888	4.7814e+03	20.0000	3.0853
			23.0853	3.0854
			26.1707	3.0854
			29.2560	3.2239
			32.4799	-4.0270
			28.4530	-4.0271
			24.4259	-4.0271
4	**0.2505**	**5.2785e+03**	20.0000	3.2738
			23.2738	3.2738
			26.5476	3.2738
			29.8214	**3.8045**
			33.6259	-4.5048
			29.1211	-4.5047
			24.6163	-4.504

In this case the algorithm achieved and meets almost all constraints without violation. It works out to give better results when the problem is set to be without strong equality constraint. We see that the stopping criteria ends the algorithm with all constraints being satisfied (Obtain the value of X_3 set to 30 in 3rd state and corresponding control value U_3 set to 4).

This may be explained by the fact of weak or easy satisfiable equality constraint for states and control values while minimizing the cost function J (Performance index) and weighting the violation of constraints.

Eventually with this change the optimization ends at 4th iteration with γ = **0.2505** and performance index J = **5.2785e+03** as results

N	States (Xi)	Control (Ui)
0	20.0000	3.2738
1	23.2738	3.2738
2	26.5476	3.2738
3	**29.8214**	**3.8045**
4	33.6259	-4.5048
5	29.1211	-4.5047
6	24.6163	-4.504

CONCLUSION

The aim of this project is to implement constrained dynamic optimization algorithm with usage of penalty methods as the goal is to achieve a global minimum of a performance index by satisfying the constraint. The Penalty function keeps the algorithm checking every possible solution but not to get stuck in local minimum. This algorithm here is not fully correct, however, can be improved to cope the given problem, as well in a more complicated situation where precision is required, some adjustment must be performed to cope the challenge.