Faculty of Power and Aeronautical Engineering

Department of Automatic control & Robotics

Optimization Techniques

EOPT, Project II #OC 01 (Optimal Control)

Prepared by – Beidemaryam Awoke Bishaw

Supervised by Prof. Nwachukwu Anthony

Warsaw **April 2022**

Table of Contents

1. Instructions

In this project we have three different AMPL files. These files are project2.mod, project2.dat and project2.run. The project2.mod file contain decision variables, objective functions and constraints. The project2.data file contains all the values of the parameters that we declared on the project2.mod files. In order to run the AMPL files we need to copy and paste the three AMPL files on the AMPL folder and write "`ampl: include project2.run;`" on the AMPL command window. The AMPL-file project2.run contains basic commands to run AMPL. Do not forget to write ";" after each command.

To load the model,

Write model project2.mod;

To load the data,

Write model project2.dat;

To choose the solver,

Write option solver; where . . . has to be replaced by the path to the folder containing the AMPL package.

To obtain the optimal solution,

Write solve;

You may now take a closer look at the solution.

To see the value of a variable use the command display.

As an example to see the value of u[1] write display u [1];

You may obtain the reduced costs for these variables by writing display x [1].rc;

In the same fashion, you may get the dual variables corresponding to the constraint M1 by writing

display M1.dual;

You may get the slack in the constraints by writing

display M1.slack;

2. Introduction / Optimal Control

All physical systems will have a dynamic with associated parameters. So a ubiquitous problem is to pick those parameters to maximize some objective function.

Formally the dynamics can be described by:-

$$\dot{x} = F(x, u)$$

Which starts at an initial state given by:

$$x(0) = x_0$$

And has controllable parameters u:

$$u = U$$

The objective function consists of a function of the final state [x (T)] and a cost function (or loss function) ‘that is integrated over time T.

$$J = \psi[x(T)] + \int_0^T \ell(u, x)dt$$

3. Problem Definition

The dynamic object is described by:-

$$\dot{x_1}(t) = x_2(t)$$

$$\dot{x_2}(t) = -9.8\sin\big(x_1(t)\big) - 3x_2(t) + 0.5u(t)$$

With initial conditions:- $x_1(0) = 0,\ \ x_2(0) = 0$

Our main aim here is that finding of the control $u(t),\ \ 0 \le t \le 2$, by minimizing the cost functional.

$$Jx, u) = \int_0^2 [\,(x_1(t) - 0.4)^2 + (u(t) - 7.63)^2]dt + 50(x_1(2) - 0.4)^2$$

With constraints:-

$$x_2(t) \le 0.4 \qquad 0 \le t \le 2$$

$$u(t) \le 10 \qquad 0 \le t \le 2$$

And use piecewise constant (stair) representation of functions $u(t)$ and $x_1(t), x_2(t)$, dividing time horizon into N = 50 subintervals.

4. Methodology

4.1. Euler method

Use Euler method to pass from ODE to the set of equality constraints that is for the state equation.

$$\dot{x}(t) = f(x(t), u(t), t), \qquad t \epsilon [t_0, t_f]$$

Take

$$\dot{x}(t) = \frac{dx(t)}{dt} \approx \frac{x(t+\Delta) - x(t)}{\Delta}$$

Method of solving the optimal control problem is a direct method that works best for discrete systems. The first step is to convert the formulation of the problem to a discrete.

And transform it into a form of:

$$x(t_{k+1}) - x(t_k) = \Delta. f(x(t_k), u(t_k), t_k), k = 0,1,..,N-1, \Delta = \frac{t_f - t_0}{N}, \quad t_k = t_0 + k.\Delta$$

Or in the simpler notation:-

$$J(x,u) = \Delta \sum_{k=0}^{N-1} g(x(t_k), u(t_k), t_k) + g_f(x_N)$$

Based on this assumption our differential equation is transformed into simpler difference equation as presented below. And we treat this difference equation as a set of constraints.

$$x(t + \Delta) = x(t) + \Delta. f(x(t), u(t), t)$$

$$x(t + \Delta) = x(t) + \Delta \left[\begin{matrix} x_2(t) \\ -9.8 \sin\left(x_1(t)\right) - 3x_2(t) + 0.5u(t) \end{matrix} \right.$$

Equality constraints:

$$x_1[t+1] = x_1[t] + \Delta x_2[t] \quad \forall t \in 0 \ldots N-1$$
$$x_2[t+1] = x_2[t] + \Delta(-9.8 \sin\left(x_1(t)\right) - 3x_2(t) + 0.5u(t)) \quad \forall t \in 0 \ldots N-1$$

4.2. Bolza Type Performance Index

Transform the Bolza type performance index.

$$Jx,u) = \int_{t_0}^{t_f} g(x(t),u(t),t)dt + g_f(x(t_f))$$

Into the summation form:

$$J(x,u) = \Delta \sum_{k=0}^{N-1} g(x(t_k),u(t_k),t_k) + g_f(x_N)$$

Again, based on this assumption our objective function is transformed into summation form as presented below.

$$J(x,u) = \sum_{k=1}^{N-1} (x_1(k) - 0.4)^2 + (u(k) - 7.63)^2 + \ 50(x_1[N] - 0.4)^2$$

Treating all $x_k, u_k \; k = 0, \ldots, N-1$ as independent decision variables we formulate the optimal control problem as a static nonlinear programming problem. This problem is solved on AMPL and 3 nonlinear solvers (minos, loqo, konopt) are used.

5. Assessment of the Solution

An AMPL translator starts by reading, parsing and interpreting a model. The translator then reads some representation of particular data. The model and data are then processed to determine the linear program that they represent, and the linear program is written out in some appropriate form.

To write our AMPL program, I use the following two steps:

1. Formulation of algebraic model.
2. Read data.

5.1. Formulation of Algebraic Model

The formulation begins with a description of the index sets and numerical parameters that the model requires. Next, the decision variables are defined. Finally the objective and constraints are specified as expressions in the sets, parameters and variables.

```
#Initial Parameters

param N;

param tf;

param t0;

param x1_0;

param x2_0;

param delta := (tf - t0) / N;

#Time Sets

set TIME = 0..N;

set TIME_N_1 = 0..(N-1);

#Optimization Variables

var u {i  in TIME_N_1};

var x1 {i in TIME_N_1};

var x2 {i in TIME_N_1};

# Objective

minimize J: delta * sum {t in TIME_N_1} ((x1[t]-0.4)^2 + (u[t]-
7.63)^2) + 50*(x1[2]-0.4)^2 ;

# Constraints

subject to state_1 {t in TIME_N_1} : x2[t] <= 0.4;

subject to state_2 {t in TIME_N_1} : u[t] <= 10;

#Initial Conditions

subject to InitialConditionX1 : x1[0] == x1_0;
```

```
subject to InitialConditionX2 : x2[0] == x2_0;

# Equality Constraints

subject to ModelConstraintsX1 {t in 1..(N-1)} : x1[t] == x1[t-1] +
delta * x2[t];

subject to ModelConstraintsX2 {t in 1..(N-1)} : x2[t] == x2[t-1] +
delta * (-9.8 * sin(x1[t])-3*x2[t] + 0.5 * u[t]);
```

5.2. Data

```
# Number of steps
param N := 50;
# Initial and final time
param tf := 2;
param t0 := 0;
# Initial conditions
param x1_0 := 0;
param x2_0 := 0;
```

5.3. Run File

```
Reset;
# load the model
model control.mod;
#load the data
data control.dat;
#change the solver
#option solver minos;
#option solver loqo;
option solver conopt;
# solve LP
solve;
# display result
display x1;
display x2;
display u;
```

6. Result and Interpretation

6.1. Result

After feeding the AMPL with given data and the model objective function subjected to constraints, the result is obtained as presented below. Note that three nonlinear solver are used as explained above (minos, loqo, conopt). And the result is presented respectively.

```
CONOPT 3.17A: Locally optimal; objective 10.56355924
21 iterations; evals: nf = 23, ng = 18, nc = 39, nJ = 18, nH = 5, nHv = 7
x1 [*] :=
 0 0            10 0.138999     20 0.298296     30 0.409696     40 0.434517
 1 0.00560633   11 0.154999     21 0.313243     31 0.415635     41 0.433814
 2 0.0154858    12 0.170999     22 0.327533     32 0.420669     42 0.432764
 3 0.0283316    13 0.186999     23 0.341059     33 0.424843     43 0.431421
 4 0.04315      14 0.202999     24 0.353736     34 0.428205     44 0.429839
 5 0.0589992    15 0.218999     25 0.365499     35 0.43081      45 0.428064
 6 0.0749992    16 0.234999     26 0.376306     36 0.432714     46 0.426142
 7 0.0909992    17 0.250999     27 0.386131     37 0.433977     47 0.424115
 8 0.106999     18 0.266999     28 0.394966     38 0.434658     48 0.422021
 9 0.122999     19 0.282828     29 0.402815     39 0.434818     49 0.419894
;

x2 [*] :=
 0  0            13  0.4          26  0.270176     39  0.00400775
 1  0.140158     14  0.4          27  0.245639     40 -0.00752396
 2  0.246986     15  0.4          28  0.220871     41 -0.0175948
 3  0.321147     16  0.4          29  0.196224     42 -0.0262512
 4  0.370459     17  0.4          30  0.172007     43 -0.0335519
 5  0.39623      18  0.4          31  0.148481     44 -0.0395655
 6  0.4          19  0.39572      32  0.125866     45 -0.0443686
 7  0.4          20  0.386706     33  0.104343     46 -0.0480437
 8  0.4          21  0.373663     34  0.0840566    47 -0.0506777
 9  0.4          22  0.357265     35  0.0651165    48 -0.0523604
10  0.4          23  0.338149     36  0.0476023    49 -0.0531827
11  0.4          24  0.316905     37  0.0315654
12  0.4          25  0.294082     38  0.0170325
;

u [*] :=
 0 7.63       9 4.80471   18 7.57123   27 7.62847   36 7.62889   45 7.6298
 1 7.95875   10 5.11562   19 7.63016   28 7.62844   37 7.629     46 7.62986
 2 7.1268    11 5.42584   20 7.62979   29 7.62843   38 7.62911   47 7.62992
 3 6.19017   12 5.73527   21 7.62947   30 7.62844   39 7.62923   48 7.62996
 4 5.53385   13 6.04386   22 7.6292    31 7.62848   40 7.62934   49 7.62999
 5 4.82164   14 6.35151   23 7.62898   32 7.62853   41 7.62944
 6 4.05709   15 6.65816   24 7.62879   33 7.62861   42 7.62954
 7 4.18112   16 6.96371   25 7.62865   34 7.62869   43 7.62964
 8 4.49319   17 7.26809   26 7.62855   35 7.62879   44 7.62972
;
```

```
LOQO 7.03: optimal solution (21 iterations, 21 evaluations)
primal objective 10.56355924
  dual objective 10.56355922
x1 [*] :=
 0 0            10 0.138999     20 0.298296     30 0.409696     40 0.434517
 1 0.00560633   11 0.154999     21 0.313243     31 0.415635     41 0.433814
 2 0.0154858    12 0.170999     22 0.327533     32 0.420669     42 0.432764
 3 0.0283316    13 0.186999     23 0.341059     33 0.424843     43 0.431421
 4 0.04315      14 0.202999     24 0.353736     34 0.428205     44 0.429839
 5 0.0589992    15 0.218999     25 0.365499     35 0.43081      45 0.428064
 6 0.0749992    16 0.234999     26 0.376306     36 0.432714     46 0.426142
 7 0.0909992    17 0.250999     27 0.386131     37 0.433977     47 0.424115
 8 0.106999     18 0.266999     28 0.394966     38 0.434658     48 0.422021
 9 0.122999     19 0.282828     29 0.402815     39 0.434818     49 0.419894
;

x2 [*] :=
 0  0            13  0.4          26  0.270176     39  0.00400776
 1  0.140158     14  0.4          27  0.245639     40 -0.00752396
 2  0.246986     15  0.4          28  0.220871     41 -0.0175948
 3  0.321147     16  0.4          29  0.196224     42 -0.0262512
 4  0.370459     17  0.4          30  0.172007     43 -0.0335519
 5  0.39623      18  0.4          31  0.148481     44 -0.0395655
 6  0.4          19  0.39572      32  0.125866     45 -0.0443686
 7  0.4          20  0.386706     33  0.104343     46 -0.0480437
 8  0.4          21  0.373663     34  0.0840566    47 -0.0506777
 9  0.4          22  0.357265     35  0.0651165    48 -0.0523604
10  0.4          23  0.338149     36  0.0476023    49 -0.0531827
11  0.4          24  0.316905     37  0.0315654
12  0.4          25  0.294082     38  0.0170325
;

u [*] :=
 0 7.63       9 4.80471   18 7.57123   27 7.62847   36 7.62889   45 7.6298
 1 7.95875   10 5.11562   19 7.63016   28 7.62844   37 7.629     46 7.62986
 2 7.1268    11 5.42584   20 7.62979   29 7.62843   38 7.62911   47 7.62992
 3 6.19017   12 5.73527   21 7.62947   30 7.62844   39 7.62923   48 7.62996
 4 5.53385   13 6.04386   22 7.6292    31 7.62848   40 7.62934   49 7.62999
 5 4.82164   14 6.35151   23 7.62898   32 7.62853   41 7.62944
 6 4.05709   15 6.65816   24 7.62879   33 7.62861   42 7.62954
 7 4.18112   16 6.96371   25 7.62865   34 7.62869   43 7.62964
 8 4.49319   17 7.26809   26 7.62855   35 7.62879   44 7.62972
;
```

```
MINOS 5.51: optimal solution found.
128 iterations, objective 10.56355924
Nonlin evals: obj = 248, grad = 247, constrs = 248, Jac = 247.
x1 [*] :=
 0 0            10 0.138999     20 0.298296     30 0.409696     40 0.434517
 1 0.00560633   11 0.154999     21 0.313243     31 0.415635     41 0.433814
 2 0.0154858    12 0.170999     22 0.327533     32 0.420669     42 0.432764
 3 0.0283316    13 0.186999     23 0.341059     33 0.424843     43 0.431421
 4 0.04315      14 0.202999     24 0.353736     34 0.428205     44 0.429839
 5 0.0589992    15 0.218999     25 0.365499     35 0.43081      45 0.428064
 6 0.0749992    16 0.234999     26 0.376306     36 0.432714     46 0.426142
 7 0.0909992    17 0.250999     27 0.386131     37 0.433977     47 0.424115
 8 0.106999     18 0.266999     28 0.394966     38 0.434658     48 0.422021
 9 0.122999     19 0.282828     29 0.402815     39 0.434818     49 0.419894
;

x2 [*] :=
 0  0            13  0.4          26  0.270176     39  0.00400775
 1  0.140158     14  0.4          27  0.245639     40 -0.00752396
 2  0.246986     15  0.4          28  0.220871     41 -0.0175948
 3  0.321147     16  0.4          29  0.196224     42 -0.0262512
 4  0.370459     17  0.4          30  0.172007     43 -0.0335519
 5  0.39623      18  0.4          31  0.148481     44 -0.0395655
 6  0.4          19  0.39572      32  0.125866     45 -0.0443686
 7  0.4          20  0.386706     33  0.104343     46 -0.0480437
 8  0.4          21  0.373663     34  0.0840566    47 -0.0506777
 9  0.4          22  0.357265     35  0.0651165    48 -0.0523604
10  0.4          23  0.338149     36  0.0476023    49 -0.0531827
11  0.4          24  0.316905     37  0.0315654
12  0.4          25  0.294082     38  0.0170325
;

u [*] :=
 0 7.63       9 4.80471   18 7.57123   27 7.62847   36 7.62889   45 7.6298
 1 7.95875   10 5.11562   19 7.63016   28 7.62844   37 7.629     46 7.62986
 2 7.1268    11 5.42584   20 7.62979   29 7.62843   38 7.62911   47 7.62992
 3 6.19017   12 5.73527   21 7.62947   30 7.62844   39 7.62923   48 7.62996
 4 5.53385   13 6.04386   22 7.6292    31 7.62848   40 7.62934   49 7.62999
 5 4.82164   14 6.35151   23 7.62898   32 7.62853   41 7.62944
 6 4.05709   15 6.65816   24 7.62879   33 7.62861   42 7.62954
 7 4.18112   16 6.96371   25 7.62865   34 7.62869   43 7.62964
 8 4.49319   17 7.26809   26 7.62855   35 7.62879   44 7.62972
;
```

6.2. Interpretation

To assess the solution the result from minos nonlinear solver is used. And to give analysis for the result the detailed output result from it is presented above.

It is possible to say that, in this case, the controller is bottlenecked from its inferior limiting, allowing it to go less than 10, would surely help in reducing the cost function. Now the major term in the cost function is the final value of x_1 squared, which is multiplied by 10. Making u be able to be smaller would allow for a better way to slow down the increase of x_2, which in turn can make x_1, which is the integral of x_2 achieve smaller results. But aside from this, the result is expected. u <= 10 is the best solution to make x_2 slowdown, which makes its final value smaller and, at the same time, makes x_1 smaller (for long horizons)changing the weights in the cost function could yield different results, where the cost of the control input could be more weighted making the system choose smaller values of u. x_1, x_2 in the current situation seems like the most plausible course of action for minimizing the cost function.

7. Conclusion

x_1, x_2 are the states of the system, that span through time x_1 [0], x_1 [1], ... x_1 [N]. u Is the control input, that also spans through time but only up to N-1 (u[0], u[1], ..., u[N-1]). The constraints are the model constraints, that relate x [t+1] as a function of x[t] and u[t] for all t in 0...N-1, there is also the constraint to make u be in $\{0, 10\}$. And the cost function is expanded as explained in part one and two of this report.

Since the minimum value of u is 0, it will only be able to stop the x_2 growth at x_2 < 0, so u_equilibrium = 0/(1) = 0, which is achievable also, making u <= 10 makes x_2 slow down on its increasing. So the controller converges to using u <=10, which is expected. Especially since the minimization of x_2 and x_1 occurs only at the last value of each, minimizing the increasing value of x_2 will minimizes x_1, since x_1 is the integral of x_2, but since x_1 starts at 0, increasing x_2 at the beginning could help making x_1 (t_f) be small. It is possibly to get a lower objective function solution if we allow u to be smaller than 0, which would allow the controller to "brake" the x_2 rapidly increase dynamic.

The solvers do not solve the problem in its integer form. Since they are nonlinear solver, but not integer nonlinear solvers. So, they relax the value of u, instead of treating it as an integer.

Bibliography

1. Dantzig, G.B., Thapa, M.N., Linear Programming:
2. Introduction, Springer, 1997. Nocedal, J., Wright, S.J., Numerical Optimization, Springer, 1999.
3. H.P. Williams, Model Building in Mathematical Programming, 5th Ed, Wiley 2013. M.S. Bazaraa,
4. J.J. Jarvis, H.D. Sherali, Linear Programming and Network Flows, 4th Ed, Wiley, 2010. M.S. Bazaraa,
5. Fourer R., Gay D.M., Kernighan B.W, A Modeling Language for Mathematical Programming, 2nd Ed, Duxbury Press, Year: 2002