Optimization in Systems and Control (SC42055)

Quadratic Programming Assignment

Authors:
Kwabena Ofori
Koen Bakker

Student Number:
4031601
4389018

October 18, 2018

Contents

1 | Quadratic Optimization

In this report a cost effective solution is provided for the heat tanks allocation problem. The provided solution is based on the quadratic programming problem.

1.1 Problem definition

Provided the continuous dynamics of a heat tank which is,

$$\frac{dT}{dt}(t) = a_1(T^{amb}(t) - T(t)) + a_2(\dot{Q}^{in}(t) - \dot{Q}^{out}(t)) \tag{1.1}$$

where a_1 and a_2 are the systems parameters and $\dot{Q}^{in}(t)$, $\dot{Q}(out) > 0$

Before the problem can be put into a standard QP form, equation 1.1 should be discretized using

$$\frac{dT_k}{dt} = \frac{T_{t+1} - T_k}{\Delta t} \tag{1.2}$$

By substituting 1.2 into 1.1 we obtain the following equation

$$T_{k+1} = T_k + a_1(T_k^{amb} - T_k)\Delta t + a_2(\dot{Q}_k^{in} - \dot{Q}_k^{out})\Delta t \tag{1.3}$$

Where $\Delta t = t_{k+1}t_k$ is 3600 seconds.
By rearranging 1.3 we obtain

$$T_{k+1} = (1 - a_1\Delta t)T_k + a_2(\dot{Q}_k^{in} - \dot{Q}_k^{out})\Delta t + a_1 T_k^{amb}\Delta t \tag{1.4}$$

From which A, B and ck can be deduced as

$$A = 1 - a_1\Delta t \tag{1.5}$$
$$B = \begin{bmatrix} a_2\Delta t & -a_2\Delta t \end{bmatrix} \tag{1.6}$$
$$c_k = a_1 T_k^{amb}\Delta t \tag{1.7}$$

1.2 Parameter identification

The parameters a_1 and a_2 are determined by minimizing

$$\min \sum_{k=1}^{N} (\bar{T}_{k+1} - (A\bar{T}_k + B[\bar{\dot{Q}}_k^{in} - \bar{\dot{Q}}_k^{out}] + c_k)^2 \tag{1.8}$$

The parameters obtained using the optimization process are

$$a_1 = 0.9954 \tag{1.9}$$
$$a_2 = -0.0046 \tag{1.10}$$

1.3 Optimizing energy trade

a

There are only linear terms in both the objective function and in the constraints so the problem can be seen as a linear programming problem.

b

The prices should have the unit [J] or [W/s] since this are units used for the other variables (e.g. $\dot{Q}^k_{out}$[W] and Δt[s]). Therefore λ_k^{in} should be divided $1 * 10^6$ to go from Megawatt to Watt and it should also be divided by 3600 to go from Watt per our to Watt per second. The Δt in the objective function will reverse the second operation.

c

To get the form:

$$min\, c^T x \quad s.t.\, Ax = b, \quad x \geq 0 \tag{1.11}$$

Where x, A, b and c need to be of the form:

$$x = \begin{pmatrix} \dot{Q}^1_{in} \\ \vdots \\ \dot{Q}^N_{in} \\ T_2 \\ \vdots \\ T_N + 1 \end{pmatrix} \tag{1.12}$$

$$A = \begin{pmatrix} -\Delta t * a_2 & \cdots & \cdots & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & -\Delta t * a_2 & \cdots & 0 & \Delta t * a_1 - 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -\Delta t * a_2 & 0 & 0 & \Delta t * a_1 - 1 & 1 \end{pmatrix} \tag{1.13}$$

$$b = \begin{pmatrix} -\Delta t * a_2 * \dot{Q}^1_{out} + \Delta t * a_1 * T_{amb} + (1 - \Delta t * a_1) * T_1 \\ -\Delta t * a_2 * \dot{Q}^2_{out} + \Delta t * a_1 * T_{amb} \\ \vdots \\ -\Delta t * a_2 * \dot{Q}^N_{out} + \Delta t * a_1 * T_{amb} \end{pmatrix} \tag{1.14}$$

$$c = \begin{pmatrix} \lambda_1^{in} * 1 * 10^{-6} \\ \vdots \\ \lambda_N^{in} * 1 * 10^{-6} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{1.15}$$

Using the linprog function in Matlab gives the optimal cost of buying the input energy as 127,04 euro, with as exist flag 1.

1.4 Terminal Cost

When implementing the modification of the minimization problem we obtain:

$$\min \sum_{k=1}^{N} {}^{in}_{k} \dot{Q}^{in}_k \Delta t + 1.01(T_{N+1} - T_{ref})^2 \qquad subsection \tag{1.16}$$

$$s.t. T_{k+1} = AT_k + B[\dot{Q}^{in}_k, \dot{Q}^{out}_k]^T + c_k \tag{1.17}$$

$$0 \leq \dot{Q}^{in}_k \leq \dot{Q}^{in}_{max} \tag{1.18}$$

$$T^{min} \leq T_k \leq T^{max} \tag{1.19}$$

the optimization problem is done using quadprog with H matrix described in 1.20 as and the matrices obtained in question 3, with a slight change to the c vector.

$$H = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (0.1 + \frac{E_2}{10})/N) \end{pmatrix} \tag{1.20}$$

$$c_{new} = \begin{pmatrix} \lambda_1^{in} * 1 * 10^{-6} \\ \vdots \\ \lambda_N^{in} * 1 * 10^{-6} \\ 0 \\ \vdots \\ 0 \\ -2 * T_{ref} * (0.1 + \frac{E_2}{10})/N) \end{pmatrix} \tag{1.21}$$

The Hessian has 720 rows and 720 columns, where alone H(720,720) has a nonzero entry, c_{new} has 720 rows and 1 column.
The cost after the modification of the objective function and the extra constraint is

$$Cost_{optimal} = 154,42 \ euro \tag{1.22}$$

From which the terminal cost is

$$Cost_{terminal} = (T_{N+1} - T_{ref}) * (0.1 + \frac{E_2}{10})/N) \tag{1.23}$$

Which is 0.1237 euro, which is a small portion of the total cost.

Appendices

A | Matlab code for assignment 2.2

```matlab


min = fminunc(@FMin,[1;1])

function [K] = FMin(a)
measurements = table2array(readtable('measurements.csv'));
Qin = measurements(:,2);
Qout = measurements(:,3);
T = measurements(:,4);
Tamb = measurements(:,5);
E1 = 6;
delta = 3600;
A = 1 - delta*a(1);
B = [-delta*a(2), delta*a(2)];

tempArray = zeros(1,100+E1);
for k = 1:length(100+E1)
    ck = delta*a(1)*Tamb(k);
    tempArray(k) = (T(k+1) - (A*T(k) + B*[Qout(k);Qin(k)] + ck))^2;
    K = sum(tempArray);
end

end
```

B | Matlab code for assignment 2.3 and 2.4

```
clc
clear all
close all

%Different E-values
E1 = 6;
E2 = 1;
E3 = 9;

%Read in files
heatDemand = table2array(readtable('heatDemand.csv'));
inputPrices = table2array(readtable('inputPrices.csv'));
measurements = table2array(readtable('measurements.csv'));

%Parameters
a1 = 1.96e-7;
a2 = 3.8e-9;
delta = 3600;
T1 = 330 + E3;
Tmin = 315;
Qmax = (100 + E2) * 1000;
Tamb = 275 + E1;

Qout = heatDemand(:,2);
Lambda = inputPrices(1:360,2);
f = [Lambda / 1e6; zeros(360,1)];

%Make matrix A
deel1 = zeros(360,360);
deel2 = deel1;
deel1(1,1) = -delta*a2;
deel2(1,1) = 1;
for i=2:1:360
    deel1(i,i) = -delta*a2;
    deel2(i,i) = 1;
    deel2(i,i-1) = delta*a1-1;
end
A=[deel1 deel2];

%Make vector b
deel1b = zeros(360,1);
deel1b(1) = -delta*a2*Qout(1) + delta*a1*Tamb + (1-delta*a1)*T1;
for i = 2:1:360
    deel1b(i) = -delta*a2*Qout(i) + delta*a1*Tamb;
end
b=[deel1b];

%Make upper and lower bounds
deel1Lb = zeros(360,1);
deel2Lb = deel1Lb;
deel1Ub = deel1Lb;
deel2Ub = deel1Lb;

```

```
deel1Ub = deel1Ub + Qmax;
deel2Lb = deel2Lb + Tmin;
deel2Ub = deel2Ub + inf;

Lb = [deel1Lb ; deel2Lb];
Ub = [deel1Ub ; deel2Ub];

o = optimoptions('linprog','Algorithm','dual-simplex');
[X, Fval, flag] = linprog(f,[],[],A,b,Lb,Ub,[],o);
```

```
%Exercise 4
Tref = 323;
ecost = (0.1+E2/10)/360;
Tmax = 368;
Ub2 = Ub;
Ub2(361:720) = Tmax;
H = zeros(720,720);
H(720,720) = ecost;
f2 = f;
f2(720,1) = -2*Tref*ecost;
[X2, Fval2, flag2] = quadprog(H,f,[],[],A,b,Lb,Ub,[]);
res = Fval2 - Tref*ecost;
```