Market Driven Systems

Laboratory Exercise 2

Linear Programming and MPC

Department of Automatic Control
Lund Institute of Technology

1. Problem Formulation

In this laboratory we will focus on implementing the example used in Exercise 6. We review the problem setup of that exercise.

We will consider a company that makes two products out off plywood. In order to make these products, the material needs to be sawed into the correct sizes. After this, the products are made by gluing together the sawed pieces.

- The company sells the first product for p_1 kSEK per unit and the second for p_2 kSEK per unit.
- Each unit of the first product require usage of a saw 7 times and usage of a gluing machine 16 times.
- Each unit of the second product require usage of a saw 10 times and usage of a gluing machine 12 times.
- There are 3 saws. The cost of using each saw is shown in Figure 1.

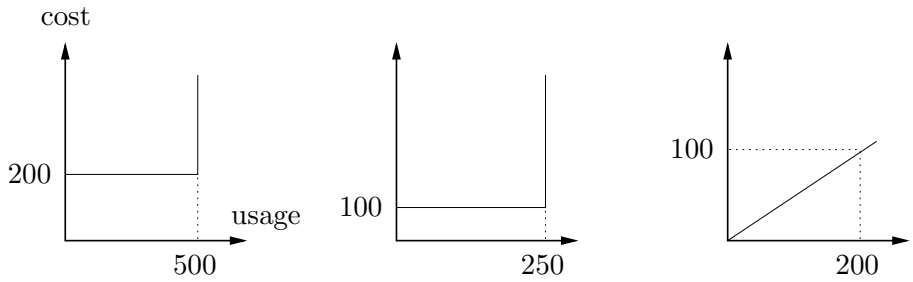

Figure 1 Diagram of the cost of the saws relative to their usage. The costs are in kSEK.

- There are 2 gluing machines. The cost of using each machine is shown in Figure 2.

The objective of the company is of course to make as much money as possible, when satisfying the constraints given by what the machines can handle.

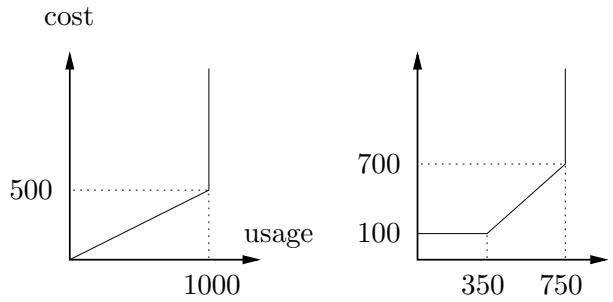

Figure 2 Diagram of the cost of the glue machines relative to their usage. The costs are in kSEK.

2. Implementation

2.1 `linprog` in Matlab

To solve the resulting optimization problem, we will use MATLAB. To solve a linear program in MATLAB we will use the command `linprog`. Assume that we have the problem

$$\begin{aligned} \underset{x}{\text{minimize}}\ & c^T x \\ \text{subject to: } & L_B \leq x \leq U_B \\ & Ax \leq b \\ & A_{\text{eq}} x = b_{\text{eq}} \end{aligned}$$

where L_B and U_B are vectors specifying lower and upper bounds on the variables x, A and b specify the remaining inequality constraints and A_{eq} and b_{eq} specify the equality constraints. To solve this using `linprog` we write

```
x = linprog(c, A, b, Aeq, beq, LB, UB)
```

If we for example don't have any equality constraints, we let `Aeq = []` and `beq = []`. If for example x_i should not be bounded from above, we let `UB(i) = Inf`.

2.2 Implementation of the production problem

Formulate and solve the optimization problem using `linprog` in MATLAB. It is suggested to use the 8 optimization variables in Table 1, which will be used later on in the lab. The necessity of variable x_8 is not immediately obvious. If you think it seems unnecessary, try to solve the problem without it.

Start with $p_1 = 21$ and $p_2 = 18$. Try other prices of the products and explain the reason why certain small changes in prices can lead to drastically different production.

Table 1 Suggested optimization variables for the static problem.

Variable	Quantity
x_1	units produced of product 1
x_2	units produced of product 2
x_3	utilization of saw 1
x_4	utilization of saw 2
x_5	utilization of saw 3
x_6	utilization of gluing machine 1
x_7	utilization of gluing machine 2
x_8	cost of utilization for gluing machine 2

3. Introducing dynamics in the system - MPC

3.1 The dynamics

Uptil now we have only treated a static system. Let's introduce dynamics to some of the machines. We will assume that the first two saws have an upstart and shut down time and that it costs money to start them. We model this by letting the supplied usage x_3 and x_4 of the saws follow the dynamics equations

$$x_3(t+1) = x_3(t) + x_9(t)$$
$$x_4(t+1) = x_4(t) + x_{10}(t)$$

where x_9 and x_{10} represent the effort of starting or shutting down each saw. We impose the constraints that $-100 \leq x_9, x_{10} \leq 100$ and introduce the term $p_3|x_9| + p_4|x_{10}|$ to the cost.

- Download the MATLAB-script `prodDynamic` from the course homepage (under Laboratories). It contains an implementation of the optimization problem that needs to be solved in the dynamic case. Read the code and make sure you understand how it works.
- Try different horizons and explain why the optimization gives different result depending on the length of the horizon.

3.2 MPC

When the prices do not change, the optimization problem can be solved beforehand (off-line) and we can use these optimal production rates and sawing/gluing usages for all times. If on the other hand, the prices changes unexpectedly over time, the optimal rates also changes. In such cases MPC is a good scheme to calculate production rates. In every time step we see what the prices are and redo the optimization to get new production rates.

Assume that the prices change according

$$p_1(t+1) = p_1(t) + w_1(t)$$
$$p_2(t+1) = p_2(t) + w_2(t)$$

where $w_i(t)$ is independent Gaussian noise with variance W_i.

- Use the code in `prodDynamic` to write the code that will control the production system with MPC for a given initial condition (for both the states and the prices) and the variance W_i.
- Use an appropriate horizon and simulate the system for $T = 100$ time steps. Let $x(0) = 0$ and $p_{[1,2]}(0) = [21, 18]^T$ and $W_i = 0.5^2$ for $i = 1, 2$.
- What is the problem of having too short horizon? Try for example a horizon of length 2.

3.3 Extra on MPC

We can also introduce dynamics on changing the amount of products being produced. That is, we do not allow to change the number of products by more than some number. This can be modeled by

$$x_1(t+1) = x_1(t) + x_{13}(t)$$
$$x_2(t+1) = x_2(t) + x_{14}(t)$$

where $-\Delta_1 \leq x_{13} \leq \Delta_1$ and $-\Delta_2 \leq x_{14} \leq \Delta_2$. The states $x_{13}(t)$ and $x_{14}(t)$ represents the change in production of product 1 and 2, respectively.

- Incorporate this dynamics into your model.