Chemical Engineering 4G03

Tutorial 10 Practice Activity

Here for your own benefit and practice (best to do it individually)
Recommended completion: Week 11.
Grading: 0% (Practice for assignments and tests)

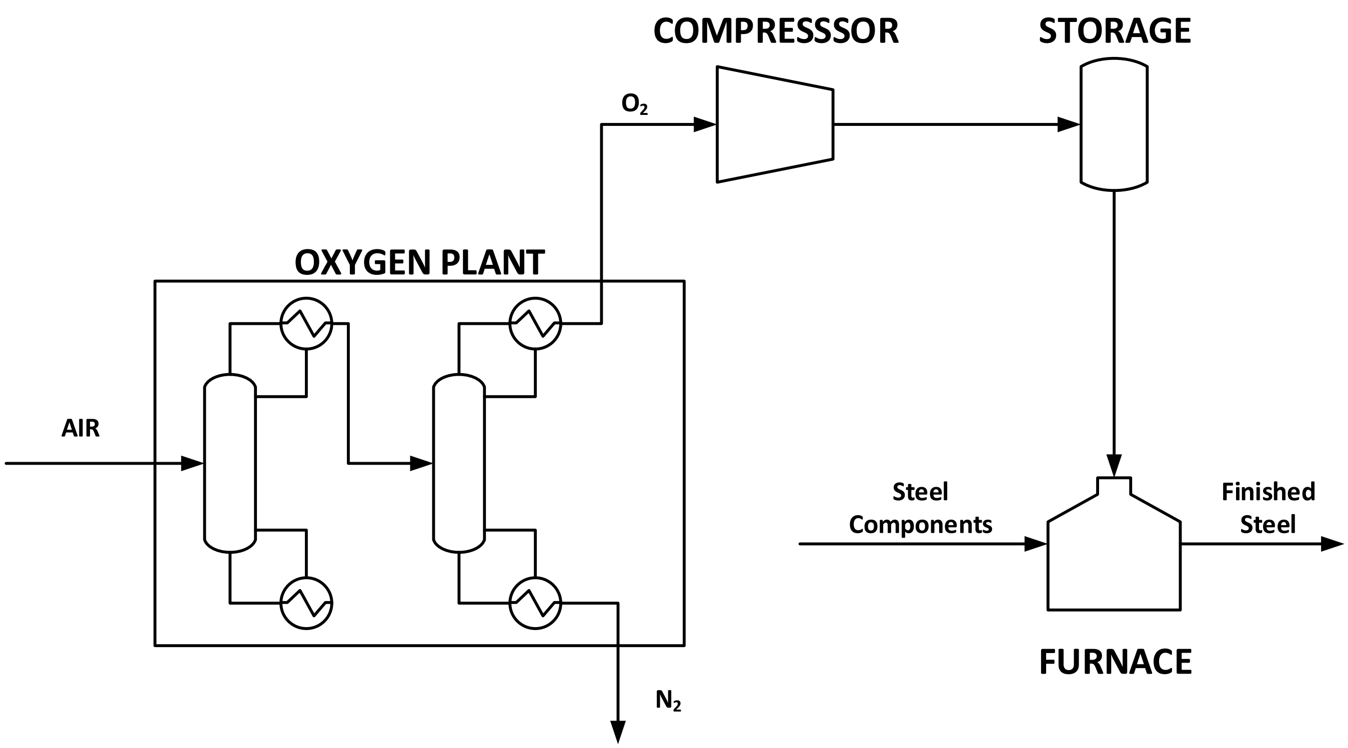
The first problem is based on a problem presented in Rardin: "Optimization in Operations Research" 2nd ed (chapter 17.1). It is a great nonlinear chemical engineering design example, which should give you some perspective on how nonlinear strategies, power-laws for capital investments, and NPV can be optimized in the chemical engineering industry

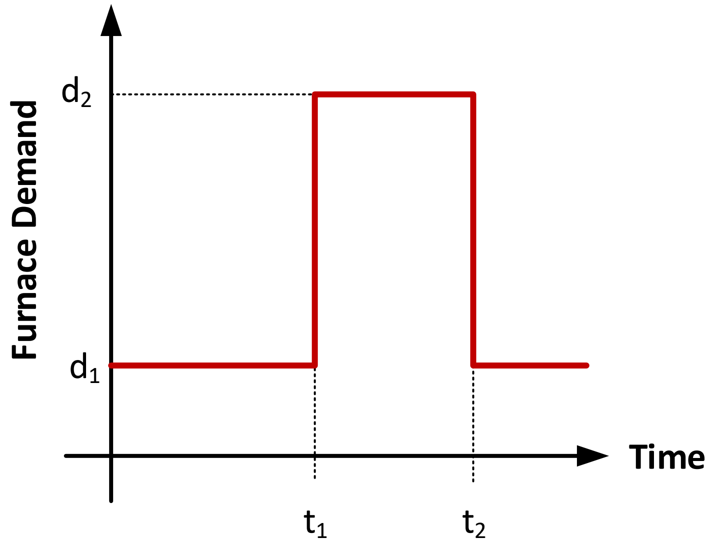
---

Part 1 Background

You work for a company that uses a basic oxygen furnace (BOF) as a step to produce steel products. A basic PFD of your process is in the figure below, in which you take O_2 from an oxygen separation unit, compress it, and store it for use in your BOF. Your company tries to take advantage of low spot-prices of electricity, so during any given optimization period (say one day) you prescribe a relatively low demand d_1 until a certain time t_1, and then a much higher demand d_2 from t_1 to t_2. This demand profile is shown in the second figure below.

We seek the minimum total cost of sizing and operating the main equipment (except the oxygen plant) based on this expected demand profile that will minimize the total normalized cost per unit of steel produced.





Our decision variables are as follows:

- $x_1 \triangleq$ the production rate of O_2 from the oxygen plant.
- $x_2 \triangleq$ the pressure in the storage tank.
- $x_3 \triangleq$ the compressor power consumption.
- $x_4 \triangleq$ the volume of the storage tank.

Our objective function for this problem is to minimize the total cost defined as:

$$(cost) = (oxygen\ plant\ cost) + (compressor\ cost) + (storage\ tank\ cost) + (electrical\ costs)$$

Based on their prior experience with similar systems, your operators and engineers know that the cost of operating the oxygen plant is very close to linear and is described as:

$$(oxygen\ plant\ cost) = 61.8 + 5.72(oxygen\ production\ rate)$$

Normalized compressor costs grow using a nonlinear power-law (Don Woods anyone):

$$(compressor\ cost) = 0.0175(power)^{0.85}$$

Storage vessels follow a similar law, but based on volume:

$$(storage\ tank\ cost) = 0.0094(volume)^{0.75}$$

And finally, the electricity consumed by the compressor is proportional to the time of operation and the cost per unit of electricity:

$$(electrical\ costs) = 0.006(time\ of\ operation)(power)$$

A realistic constraint for this model is that the O_2 produced by the oxygen plant must be sufficient to supply the demand for the entire cycle:

$$t_2 x_1 \geq d_1 t_1 + d_2(t_2 - t_1)$$

Furthermore, the minimum storage pressure in the tank (to prevent compressor stall) is

$$x_2 \geq p_m$$

Our final two constraints are physical relations that require that the compressor must consume enough power to address the maximum stored inventory pressure just before the large demand period between t_1 and t_2. Using a basic compressor correlation, we have:

$$x_3 = 36.25 \frac{(d_2 - x_1)(t_2 - t_1)}{t_1} \ln\left(\frac{x_2}{p_m}\right)$$

And finally, we require that the storage tank volume can hold the maximum required inventory to satisfy the sudden burst of demand. After handling units and using an appropriate gas law to convert the kg of O_2 required to pressure, we have:

$$x_4 = 348{,}300 \frac{(d_2 - x_1)(t_2 - t_1)}{x_2}$$

Part 1 Tasks

Your tasks are as follows:

1. Use the information provided to you by your engineers above to formulate this problem as a constrained NLP. Clearly state the objective function and constraints.

2. Code your NLP in GAMS and solve it for $d_1 = 2.5$, $d_2 = 40$, $t_1 = 0.6$, $t_2 = 1.0$ and $p_m = 200$.

3. Explain how you might convert this to an *unconstrained* nonlinear program, which would allow you to use gradient-based or derivative-free optimization methods without having to explicitly consider the constraint equations.

Hint – The answer should be $x^\star = (x_1 \ldots x_4)$ = $(17.5, 473.7, 468.8, 6618)$ *for a total cost of* $\phi^\star = 173.7$.

GAMS Advice

You will need to manually provide an initial guess to the NLP solver for this problem. If you don't, it will use initial guesses of 0, which is a serious issue for some equation terms, such as $\ln\left(\frac{x_2}{p_m}\right)$. You can set the "level" (as in, the value) of a variable at any time, including before the solve statement, which counts as an initial guess. Consider using these initial guesses before the solve statement:

```
x1.l = 20;
x2.l = 500;
x3.l = 500;
x4.l = 6000;
```

Part 2

In this section we are going to do a little `MATLAB`. This is going to come in very handy for you in the final assignment!

Consider a function of N variables $f(\boldsymbol{x})$ where $x = [x_1\ x_2\ \dots\ x_N]^T$. You can approximate the partial derivative of f at the point $x^* = [x_1^*\ x_2^*\ \dots x_N^*]$ with respect to any individual variable x_i using the **centered finite difference method** like this:

$$\frac{\partial f}{\partial x_1}(x^*) = \frac{f(x_1^* + h, x_2^*, \dots, x_N^*) - f(x_1^* - h, x_2^*, \dots, x_N^*)}{2h}$$
$$\frac{\partial f}{\partial x_2}(x^*) = \frac{f(x_1^*, x_2^* + h, \dots, x_N^*) - f(x_1^*, x_2^* - h, \dots, x_N^*)}{2h}$$
$$\vdots$$
$$\frac{\partial f}{\partial x_N}(x^*) = \frac{f(x_1^*, x_2^*, \dots, x_N^* + h) - f(x_1^*, x_2^*, \dots, x_N^* - h)}{2h}$$

Your task is to write a MATLAB function that accepts *any function in MATLAB* and returns the *numerical gradient* of that function for a certain step size h.

1. Consider the following function:

$$f(x) = 3x\sin(x) + e^x \tan(x)$$

   Write a MATLAB code that computes the numerical gradient $\nabla f(x)$ at the point $x = 2$. Use a step size of $h = 10^{-6}$. In other words, take the numerical derivative of $f(2)$. *ANSWER: 26.75*

2. Consider the following function:

$$f(\boldsymbol{x}) = \sin(x_1^2) + 2\cos(x_1 x_2) - \ln(x_3)\, e^{(\sin(x_1))^3}$$

   Write a MATLAB code that computes the numerical gradient $\nabla f(\boldsymbol{x}^\star)$ at the point $\boldsymbol{x}^\star = [1\ \ 2\ \ 3]^T$. Use a step size of $h = 10^{-6}$. *ANSWER: [-0.602337 -1.68294 -1.81452]*

3. Consider the following function in $N = 20$ dimensions (that is, $x_1 \dots x_{20}$) provided in the **tutorial 10 companion folder**:

$$f(\boldsymbol{x}) = \mathrm{jollycooperation(x)}$$

   Write a MATLAB code that computes the numerical gradient $\nabla f(\boldsymbol{x}^\star)$ at the point $\boldsymbol{x}^\star = [1 \dots 1]^T$. Use a step size of $h = 10^{-6}$. Moreover, you **cannot open this function**. Rather, you will need to make sure your gradient function works *generally* for any function in N dimensions. *ANSWER: the norm of the gradient vector using `norm()` is 226.0885.*

Why am I making you do this? Because your final assignment will require you to use the numerical optimization methods we are covering in Module 09, and the functions you will be assigned are either too tedious to differentiate (and take the Hessian!) by hand, or *are unknown to you*.