Non Linear Programming NLP

Prof. Cesar de Prada
ISA-UVA
prada@autom.uva.es

Outline

- NLP general purpose methods
  - Sequential Quadratic Programming SQP
  - Generalized Reduced Gradient GRG
  - Cutting plane CP
- Software
- Examples

Sequential Quadratic Programming SQP

In order to facilitate the description of the ideas behind the SQP method, we will examine first a simplified case where only equality constraints are considered, and , then, the formulation will be extended to the general NLP case.

The SQP method solves the KKT conditions approximating them linearly around a value (x, λ) (or solving an equivalent QP problem) and iterates in order to improve the estimation, until no noticeable improvement is obtained.

$$\left.\begin{array}{l} \min\limits_{x} J(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{array}\right\} \quad L(\mathbf{x},\boldsymbol{\lambda}) = J(\mathbf{x}) + \boldsymbol{\lambda}'\mathbf{h}(\mathbf{x})$$

SQP

$$\left.\begin{array}{l}\min_x J(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\end{array}\right\} \quad L(\mathbf{x},\boldsymbol{\lambda}) = J(\mathbf{x}) + \boldsymbol{\lambda}'\mathbf{h}(\mathbf{x})$$

KKT Conditions :

$$\begin{cases}\nabla_x L(\mathbf{x},\boldsymbol{\lambda}) = \nabla_x J(\mathbf{x}) + \boldsymbol{\lambda}'\nabla_x \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\end{cases}$$

We can solve the KKT conditions using Newton's steps, linearizing the equations around an initial estimate x_k, λ_k of the solution:

$$\nabla_x L(\mathbf{x},\boldsymbol{\lambda}) \approx \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda_k}) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k}) + \Delta\boldsymbol{\lambda}'\nabla_x \mathbf{h}(x_k) = \mathbf{0}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

Notice that the Newton-Raphson method equates to zero a first order approximation in order to compute Δx, $\Delta\lambda$ such that ∇L=0, h = 0

This is a linear systems of equations that can be solved in order to find Δx y $\Delta\lambda$..

Notation

$$\nabla_x J(\mathbf{x}) = \begin{bmatrix} \frac{\partial J(\mathbf{x})}{\partial x_1} & \frac{\partial J(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial J(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

$$\nabla_x L(\mathbf{x},\boldsymbol{\lambda}) = \begin{bmatrix} \frac{\partial L(\mathbf{x},\boldsymbol{\lambda})}{\partial x_1} & \frac{\partial L(\mathbf{x},\boldsymbol{\lambda})}{\partial x_2} & \dots & \frac{\partial L(\mathbf{x},\boldsymbol{\lambda})}{\partial x_n} \end{bmatrix}$$

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_m(\mathbf{x}) \end{bmatrix} \qquad \nabla_x \mathbf{h}(\mathbf{x}) = \begin{bmatrix} \frac{\partial h_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial h_1(\mathbf{x})}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial h_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial h_m(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla_x h_1(\mathbf{x}) \\ \vdots \\ \nabla_x h_m(\mathbf{x}) \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x},\boldsymbol{\lambda}) = \begin{bmatrix} \frac{\partial^2 L}{\partial x_1^2} & \dots & \frac{\partial^2 L}{\partial x_1 \partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial^2 L}{\partial x_n \partial x_1} & \dots & \frac{\partial^2 L}{\partial x_n^2} \end{bmatrix}$$

$$\boldsymbol{\lambda}' \nabla_x \mathbf{h}(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i \nabla_x h_i(\mathbf{x})$$

$$\nabla_\lambda (\boldsymbol{\lambda}' \nabla_x \mathbf{h}(\mathbf{x})) = \nabla_x \mathbf{h}(\mathbf{x})'$$

SQP

Nevertheless, it is also possible to find the same solution solving the QP problem:

$$\boxed{\begin{aligned}&\min_{\Delta x} \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k})\Delta\mathbf{x}\\ &\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0\end{aligned}}$$

In fact, the Lagrangian L_s of this QP problem is:

$$L_s(\Delta\mathbf{x},\sigma) = \nabla_x L(\mathbf{x}_k,\boldsymbol{\lambda}_k)\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x}_k,\boldsymbol{\lambda}_k)\Delta\mathbf{x} + \sigma'(\mathbf{h}(\mathbf{x}_k) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x})$$

And its corresponding KKT conditions are:

$$\nabla_{\Delta x} L_s(\Delta\mathbf{x},\sigma) = \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \sigma'\nabla_x \mathbf{h}(x_k) = 0$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

Which are exactly the same set of equations that the previous linear ones with $\sigma = \Delta\lambda$

SQP

As the linearized problem is only an approximation, the SQP method iterates, starting again in the point:

$$\begin{cases} \nabla_x L(\mathbf{x},\boldsymbol{\lambda}) = \nabla_x J(\mathbf{x}) + \sum_i \lambda_i \nabla_x h_i(\mathbf{x}) = \mathbf{0} \\ \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{cases}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \Delta\boldsymbol{\lambda}$$

$$\min_{\Delta x} \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k})'\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k})\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

$$\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) = \nabla_x^2 J(\mathbf{x_k}) + \sum_i \lambda_k \nabla_x^2 h_i(\mathbf{x_k})$$

And solving the associated QP problems until there is no sensible changes in x and J

SQP another formulation

As $\nabla_x L(\mathbf{x},\boldsymbol{\lambda}) = \nabla_x J(\mathbf{x}) + \sum_i \lambda_i \nabla_x h_i(\mathbf{x})$ Substituting in:

$$\min_{\Delta x} \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda_k})'\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k})\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0 \qquad (*)$$

$$\min_{\Delta x} \nabla_x J(\mathbf{x_k})'\Delta\mathbf{x} + \boldsymbol{\lambda}_k'\nabla_x h_i(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k})\Delta\mathbf{x} =$$

and using (*)

$$= \min_{\Delta x} \nabla_x J(\mathbf{x_k})'\Delta\mathbf{x} - \boldsymbol{\lambda}_k'\mathbf{h}(\mathbf{x_k}) + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k})\Delta\mathbf{x}$$

Cte.

Which is equivalent to:

$$\boxed{\begin{array}{l} \min_{\Delta x} \nabla_x J(\mathbf{x_k})'\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k})\Delta\mathbf{x} \\ \mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0 \end{array}}$$

SQP another formulation

$$\min_{\Delta x} \nabla_x J(\mathbf{x_k})'\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k})\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

In this case, the KKT conditions are:

$$\nabla_x J(\mathbf{x_k}) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \sigma'\nabla_x \mathbf{h}(x_k) = 0$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

Comparing with the Newton-step equations:

$$\nabla_x L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \Delta\boldsymbol{\lambda}'\nabla_x \mathbf{h}(x_k) = \mathbf{0}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0$$

$$\nabla_x J(\mathbf{x_k}) + \boldsymbol{\lambda}_k'\nabla_x \mathbf{h}(\mathbf{x}_k) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + \Delta\boldsymbol{\lambda}'\nabla_x \mathbf{h}(x_k) = \mathbf{0}$$

$$\nabla_x J(\mathbf{x_k}) + \Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k}) + (\boldsymbol{\lambda}_k' + \Delta\boldsymbol{\lambda}')\nabla_x \mathbf{h}(x_k) = \mathbf{0}$$

And now $\sigma = \lambda + \Delta\lambda$, so that, with this formulation, the updating is:

$$\boxed{\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta\mathbf{x} \\ \boldsymbol{\lambda}_{k+1} &= \sigma \end{aligned}}$$

SQP general case

The NLP problem can be reformulated using slack variables in the following way:

$$\min_{x} J(\mathbf{x}) \qquad \qquad \min_{\mathbf{x},\boldsymbol{\varepsilon}} \mathrm{J}(\mathbf{x})$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \quad \Rightarrow \quad \mathbf{h}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \qquad \qquad \mathbf{g}(\mathbf{x}) + \boldsymbol{\varepsilon} = \mathbf{0}$$

$$\boldsymbol{\varepsilon} \geq \mathbf{0}$$

And the decision vector is extended with the slack variables:

$\mathbf{z} = [\ \mathbf{x}\ ,\ \boldsymbol{\varepsilon}\ ]'$

So that we can consider problems with the format:

$$\left.\begin{array}{l} \min\limits_{\mathbf{z}} \mathrm{J}(\mathbf{z}) \\ \mathbf{c}(\mathbf{z}) = \mathbf{0} \\ \mathbf{m} \leq \mathbf{z} \leq \mathbf{M} \end{array}\right\} \qquad \mathbf{c}(\mathbf{z}) = \begin{bmatrix} \mathbf{h}(\mathbf{x}) \\ \mathbf{g}(\mathbf{x}) + \boldsymbol{\varepsilon} \end{bmatrix}$$

SQP Wilson 1963

$$\left.\begin{aligned}&\min_x J(\mathbf{x})\\&\mathbf{h}(\mathbf{x})=\mathbf{0}\\&\mathbf{m}\le\mathbf{x}\le\mathbf{M}\end{aligned}\right\}\quad L(\mathbf{x},\boldsymbol{\lambda},\mu,\eta)=J(\mathbf{x})+\boldsymbol{\lambda}'\mathbf{h}(\mathbf{x})+\mu'(\mathbf{m}-\mathbf{x})+\eta'(\mathbf{x}-\mathbf{M})$$

KKT conditions:

$$\begin{cases}\nabla_x L(\mathbf{x},\boldsymbol{\lambda},\mu,\eta)=\nabla_x J(\mathbf{x})+\sum_i \lambda_i\nabla_x h_i(\mathbf{x})-\mu'+\eta'=\mathbf{0}\\ \mathbf{h}(\mathbf{x})=\mathbf{0},\quad \mathbf{m}\le\mathbf{x}\le\mathbf{M}\quad \mu'(\mathbf{m}-\mathbf{x})=0\quad \eta'(\mathbf{M}-\mathbf{x})=0\\ \mu\ge 0,\quad \eta\ge 0\end{cases}$$

In order to solve them, a linearization around an initial estimate x_k, λ_k,μ_k,η_k will lead to:

$$\nabla_x L(\mathbf{x},\boldsymbol{\lambda},\mu,\eta)\approx\nabla_x L(\mathbf{x_k},\boldsymbol{\lambda_k},\mu_k,\eta_k)+\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda_k},\mu_k,\eta_k)+\Delta\boldsymbol{\lambda}'\nabla_x\mathbf{h}(x_k)-\Delta\mu'+\Delta\eta'=0$$

$$\mathbf{h}(\mathbf{x_k})+\nabla_x\mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x}=0,\quad \mathbf{m}\le\mathbf{x}_k+\boldsymbol{\Delta}\mathbf{x}\le\mathbf{M},$$

$$\mu_k'(\mathbf{m}-\mathbf{x}_k)-\mu_k'\Delta\mathbf{x}+(\mathbf{m}-\mathbf{x}_k)'\Delta\mu=0\quad \mu_k+\Delta\mu\ge\mathbf{0}$$

$$\eta_k'(\mathbf{x}_k-\mathbf{M})+\eta_k'\Delta\mathbf{x}+(\mathbf{x}_k-\mathbf{M})'\Delta\eta=0\quad \eta_k+\Delta\eta\ge\mathbf{0}$$

SQP

In this case, an equivalent QP problem corresponds to:

$$\min_{\Delta x} \nabla_x L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k},\mu_k,\eta_k)\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k},\boldsymbol{\lambda}_\mathbf{k},\mu_k,\eta_k)\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x} \le \mathbf{M}$$

As can be verified, computing its KKT conditions from its Lagrangian L_s

$$L_s(\Delta\mathbf{x},\Delta\boldsymbol{\lambda},\Delta\mu,\Delta\eta) =$$

$$= \nabla_x L(\mathbf{x}_k,\boldsymbol{\lambda}_k,\mu_k,\eta_k)\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x}_k,\boldsymbol{\lambda}_k,\mu_k,\eta_k)\Delta\mathbf{x} +$$

$$+ \Delta\boldsymbol{\lambda}'(\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)'\Delta\mathbf{x}) - \Delta\mu'(\mathbf{m} - \mathbf{x}_k - \Delta\mathbf{x}) + \Delta\eta'(\mathbf{x}_k + \Delta\mathbf{x} - \mathbf{M}) = 0$$

SQP- (Nash &Sofer Modifications 1996)

When solving each QP subproblem there is no guarantee that $\nabla_x^2 L$ is PD and. In addition, it is required to compute the Hessian of all functions. In order to avoid this difficulties, the QP subproblem is modified as:

$$\min_{\Delta x} \nabla_x L(\mathbf{x_k}, \boldsymbol{\lambda}_\mathbf{k}, \mu_k, \eta_k)\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\nabla_x^2 L(\mathbf{x_k}, \boldsymbol{\lambda}_\mathbf{k}, \mu_k, \eta_k)\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_k + \Delta\mathbf{x} \le \mathbf{M}$$

Substituting $\nabla_x^2 L$ by a PD matrix, B_k that is updated every iteration so that it converges to the Hessian using the BFGS technique, in this way, only L y $\nabla_x L$ are required. Also, as before, $\nabla_x L$ can be replaced by $\nabla_x J$

$$\min_{\Delta \mathrm{x}} \nabla_\mathrm{x} \mathrm{J}(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\mathbf{B}_\mathrm{k}\Delta\mathbf{x}$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_\mathrm{x} \mathbf{h}(\mathbf{x}_\mathrm{k})\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_\mathrm{k} + \Delta\mathbf{x} \le \mathbf{M}$$

SQP- (Nash &Sofer Modifications 1996)

Also, another change in the SQP is incorporated optimizing the step length in every iteration in order to improve the speed of convergence. So, instead of:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$$
$$\boldsymbol{\lambda}_{k+1} = \sigma$$

The correction is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\Delta\mathbf{x}$$
$$\boldsymbol{\lambda}_{k+1} = \sigma$$

Where the step length α is computed in order to minimize J in the direction Δx with an exact penalty added:

$$\min_{\alpha} J((\mathbf{x}_k + \alpha\Delta\mathbf{x})) + \sum_i \omega_i \left| h_i(\mathbf{x}_k + \alpha\Delta\mathbf{x}) \right|$$

where ω_i are the penalty weights

SQP Algorithm

1. $B_k = I, \; x_k = x_0$
2. Solve the QP subproblem obtaining Δx_k and λ_k
3. Test the optimality conditions (KKT and changes in J and x)
4. Choose the weights ω_i and compute α_k minimizing J in the direction Δx_k with exact penalty in h
5. Do $x_{k+1} = x_k + \alpha_k \Delta x_k$
6. Compute $L(x_k)$, $L(x_{k+1})$, $\nabla_x L(x_k, \lambda_k)$, $\nabla_x L(x_{k+1}, \lambda_k)$, and the new estimate B_{k+1} using the BFGS method
7. k=k+1, go to 2

The SQP method is efficient with superlinear convergence up to several thousand variables. For bigger problems is advisable to use SLP.

Codes: NPSOL, NAG, fmincon, SNOPT

Implemented in GAMS, NAG, Matlab, ….

Large-scale SQP

- Practical problems may involve more than 10^5 variables, constraints, equations….
- Two type of problems:
  - Few degrees of freedom (10-100) *(n-m)*
    - RTO, parameter estimation, SS flowsheet optimization,..
  - Many degrees of freedom (> 1000)
    - Distributed parameters, dynamic optimization, data reconciliation, state estimation,…

Reduced space SQP (rSQP)

- Recommended for large scale problems with few degrees of freedom.

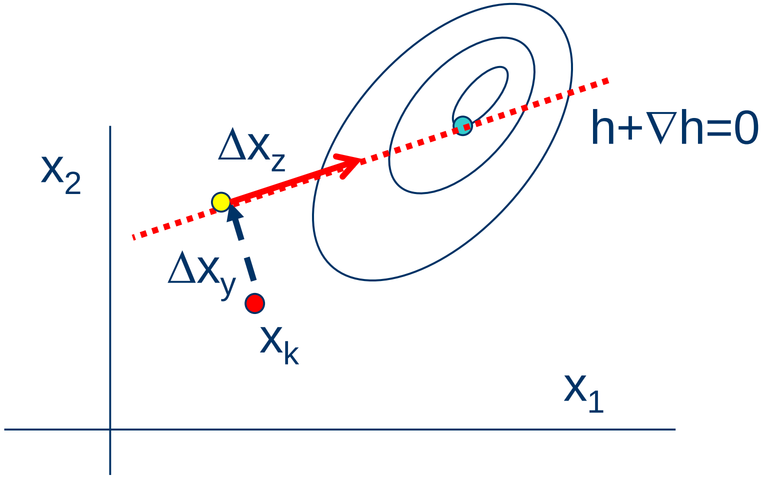
SQP QP problem to be solved at every step

$$\min_{\Delta x} \nabla_x J(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}' H_k \Delta\mathbf{x}$$

$$H_k = \nabla_x^2 L(\mathbf{x_k}, \boldsymbol{\lambda_k}, \mu_k, \eta_k)$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \leq \mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x} \leq \mathbf{M}$$

rSQP moves at every step in two separate directions. One fulfils the linearized equality constraints, the other moves along these constraints improving the cost respecting the inequalities



Codes: SNOPT, MUSCOD-II,..

Reduced space SQP (rSQP)

Let's define a new basis $[Y_k, Z_k]$ for Δx where the last n-m components, Z_k are perpendicular to the gradient of the equality constraints h and Y_k is chosen to make $[Y_k, Z_k]$ non-singular :

$$\nabla_x \mathbf{h}(x_k) Z_k = 0$$
$$\Delta \mathbf{x} = Y_k \Delta \mathbf{x}_y + Z_k \Delta \mathbf{x}_z$$
$$Y_k (n \text{ x } m), \quad Z_k (n \text{ x } (n-m))$$

n size of x
m size of h

If $\Delta \mathbf{x}$ has to fulfill the linearized constraints: $\mathbf{h(x_k)} + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta \mathbf{x} = 0$

$$\nabla_x \mathbf{h}(x_k)(Y_k \Delta \mathbf{x}_Y + Z_k \Delta \mathbf{x}_z) = -\mathbf{h}(\mathbf{x}_k)$$
$$\nabla_x \mathbf{h}(x_k) Y_k \Delta \mathbf{x}_Y = -\mathbf{h}(\mathbf{x}_k)$$
$$\Delta \mathbf{x}_Y = -[\nabla_x \mathbf{h}(x_k) Y_k]^{-1} \mathbf{h}(\mathbf{x}_k)$$
$$\Delta \mathbf{x} = -Y_k [\nabla_x \mathbf{h}(x_k) Y_k]^{-1} \mathbf{h}(\mathbf{x}_k) + Z_k \Delta \mathbf{x}_z$$

Reduced space SQP (rSQP)

And substituting Δx into the original QP problem, results in a new QP in the reduced space Δx_z:

$$\min_{\Delta x} \nabla_x J(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}' H_k \Delta\mathbf{x}$$
$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x} \le \mathbf{M}$$

$$\min_{\Delta \mathbf{x}_z} \frac{1}{2}\Delta\mathbf{x}_z'\mathbf{B}_k\Delta\mathbf{x}_z + \Delta\mathbf{x}_z(Z_k'\mathbf{H}_k'Y_k\Delta\mathbf{x}_y + Z_k'\nabla_x J(\mathbf{x}_k)')$$
$$\mathbf{m} \le \mathbf{x}_k + Y_k\Delta\mathbf{x}_y + Z_k\Delta\mathbf{x}_z \le \mathbf{M}$$

Where the constant terms have been dropped from the cost function and where B_k is BFGS update of $Z_k'H_K'Z_k$

After the reduced QP, Δx can be computed from:

$$\Delta\mathbf{x} = -Y_k\left[\nabla_x \mathbf{h}(x_k)Y_k\right]^{-1}\mathbf{h}(\mathbf{x}_k) + Z_k\Delta\mathbf{x}_z$$

Computing Z_k, Y_k

Apply QR decomposition to $\nabla h(x_k)'$ (n x m)

$$\nabla_x \mathbf{h}(x_k)' = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Y_k & Z_k \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix}$$

With Q unitary matrix

$$Y_k' Y_k = I_m \quad Z_k' Z_k = I_{n-m} \quad Z_k' Y_k = 0$$

$$Z_k' \nabla_x \mathbf{h}(x_k)' = Z_k \begin{bmatrix} Y_k & Z_k \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & I \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix} = 0$$

$$\nabla_x \mathbf{h}(x_k) Z_k = 0$$

$$\nabla_x \mathbf{h}(x_k)' = \begin{bmatrix} K_k & L_k \end{bmatrix} \quad Z_k = \begin{bmatrix} -K_k^{-1} L_k \\ I \end{bmatrix} \quad Y_k = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

Or other methods:

K(m x m)

as $\nabla_x \mathbf{h}(x_k)' Z_k = \begin{bmatrix} K_k & L_k \end{bmatrix}\begin{bmatrix} -K_k^{-1} L_k \\ I \end{bmatrix} = 0$

rSQP algorithm

With inequality constraints on x, the [Y, Z] decomposition of Δx is applied to the QP problem:

1- Choose $x_{0,}$ $k = 0$
2- At every iteration, compute $J(x_k)$, $h(x_k)$, $\nabla J(x_k)$, $\nabla h(x_k)$
3- Compute Y_k, Z_k
4- Compute Δx_y from $\nabla h(x_k) Y_k \Delta x_y = - h(x_k)$
5- Update B_k using BFGS instead of computing $Z_k' H_K' Z_k$
6- Solve

$$\min_{\Delta \mathbf{x}_z} \frac{1}{2} \Delta \mathbf{x}_z ' \mathbf{B}_k \Delta \mathbf{x}_z + \Delta \mathbf{x}_z (Z_k' \mathbf{H}_k' Y_k \Delta \mathbf{x}_y + Z_k' \nabla_x J(\mathbf{x}_k)')$$

$$\mathbf{m} \leq \mathbf{x}_k + Y_k \Delta \mathbf{x}_y + Z_k \Delta \mathbf{x}_z \leq \mathbf{M}$$

Often $Z_k' H_k' Y_k \Delta x_y$ is approximated by zero

7- Check stopping criteria. If satisfied, stop
8- Compute multipliers from $Y_k' \nabla h(x_k)' \lambda_k = - Y_k' \nabla J(x_k)'$
9- Calculate $\Delta x = Y_k \Delta x_y + Z_k \Delta x_z$
10- Compute step size α : $x_{k+1} = x_k + \alpha \Delta x$
11– Make k = k + 1,Go to step 2

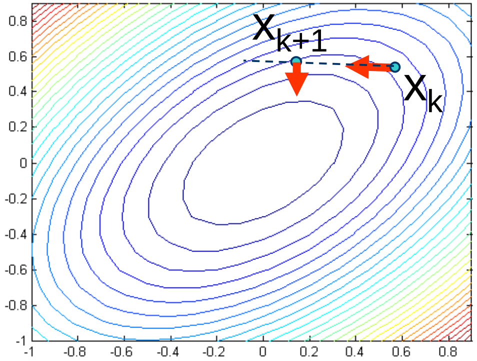
Generalized Reduced Gradient GRG

This method uses the equality constraints to eliminate decision variables, converting the constraint problem in an unconstraint one. Also, it can be seen as an adaptation of the steepest descent method that uses a projected gradient on the constraints

It was developed by Abadie &Carpentier (1969). An improved version due to Lasdon (1992) is known as GRG2

Implemented in the Excel solver, CONOPT,..

Steepest descend method:



$$\mathbf{x}_{k+1} = \mathbf{x}_k - \sigma_k \frac{\partial J(\mathbf{x}_k)'}{\partial \mathbf{x}} =$$

$$= \mathbf{x}_k - \sigma_k \nabla_x J(\mathbf{x}_k)$$

$$\min_{\sigma_k} J(\mathbf{x}_k - \sigma_k \nabla_x J(\mathbf{x}_k))$$

$$\text{parar si } \|\nabla_x J(\mathbf{x}_k)\| \leq \varepsilon$$

GRG How to include constraints?

$$\min_x J(\mathbf{x})$$
$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

In order to facilitate the description of the ideas behind the GRG method, we will examine first a simplified case where only equality constraints are considered, and , then, the formulation will be extended to the general NLP case

If it were possible to work out m variables x_i from h(x) =0, then, after substitution in J(x), the problem would be converted in an unconstraint one in the remaining n-m variables, that could be solved e.g. with the steepest descend method.

In general, as the m equations $h_i(x)=0$ can be non-linear, it won't be possible to work out explicitly the m, x_i. The GRG method provides a way to obtain an equivalent formulation. When GRG was devloped, there were no computing facilities to work out some variables as a function of others from h(x) = 0

GRG

$$\min_x J(\mathbf{x})$$
$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

Be x_k a point that satisfy the equality constraints of the NLP problem. A linear approximation of h(x) at this point is:

$$\mathbf{h}(\mathbf{x}) \approx \mathbf{h}(\mathbf{x}_k) + \nabla_x \mathbf{h}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)$$

And we impose the constraint that the linear approximation be zero. Then, as $h(x_k) = 0$:

$$\nabla_x \mathbf{h}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) = 0$$

What is a linear system in x. By simplicity, let's name x_B to the first m components of x, (basic variables) and x_N to the remaining ones, so that $x' = [x'_B | x'_N]$ and let's try to work out x_B as functions of x_N

GRG

$$\nabla_x \mathbf{h}(\mathbf{x}) = \left[\begin{array}{ccc|ccc} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_m} & \frac{\partial h_1}{\partial x_{m+1}} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_m} & \frac{\partial h_m}{\partial x_{m+1}} & \cdots & \frac{\partial h_m}{\partial x_n} \end{array}\right] = [\mathbf{B}|\mathbf{N}]$$

$$\nabla_x \mathbf{h}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) = \mathbf{0} \quad \Rightarrow \quad [\mathbf{B}(\mathbf{x}_k) \mid \mathbf{N}(\mathbf{x}_k)]\begin{pmatrix} \mathbf{x}_B - \mathbf{x}_{Bk} \\ \mathbf{x}_N - \mathbf{x}_{Nk} \end{pmatrix} = \mathbf{0}$$

$$\mathbf{x}_B - \mathbf{x}_{Bk} = -\mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)(\mathbf{x}_N - \mathbf{x}_{Nk})$$

$$J_h(\mathbf{x}_N) = J\begin{pmatrix} \mathbf{x}_B(\mathbf{x}_N) \\ \mathbf{x}_N \end{pmatrix} = J\begin{pmatrix} \mathbf{x}_{Bk} - \mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)(\mathbf{x}_N - \mathbf{x}_{Nk}) \\ \mathbf{x}_N \end{pmatrix}$$

Which depends only on x_N

GRG

$$\min_x J(\mathbf{x})$$
$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

The problem of minimizing J(x) under h(x)=0 is equivalent to minimizing $J_h(x_B(x_N), x_N)$ with respect to x_N and without constraints. For this purpose, one can use the gradient of J_h with respect to x_N that is:

$$\frac{dJ_h}{d\mathbf{x}_N} = \frac{\partial J}{\partial \mathbf{x}_N} + \frac{\partial J}{\partial \mathbf{x}_B}\frac{\partial \mathbf{x}_B}{\partial \mathbf{x}_N}$$

but:

$$\mathbf{x}_B - \mathbf{x}_{Bk} = -\mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)(\mathbf{x}_N - \mathbf{x}_{Nk}) \Rightarrow \frac{\partial \mathbf{x}_B}{\partial \mathbf{x}_N} = -\mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)$$

so,

$$\boxed{\mathbf{d} = \frac{dJ_h}{d\mathbf{x}_N} = \frac{\partial J}{\partial \mathbf{x}_N} - \frac{\partial J}{\partial \mathbf{x}_B}\mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)}$$

Which receives the name of reduced gradient and can be used, for instance, when applying the steepest descend method in relation to x_N

Stopping criterion : $$\left\| \frac{\partial J_h}{\partial \mathbf{x}_N} \right\| \le \varepsilon$$

Problems if B is singular!

GRG

This strategy leads to points that improve the values of J(x) independently of the linearity of h(x).

$$\mathbf{x}_{N,k+1} = \mathbf{x}_{N,k} - \sigma \frac{\partial J_h(\mathbf{x}_{N,k})}{\partial \mathbf{x}_N} \quad \Rightarrow \quad \mathbf{x}_{N,k+1} - \mathbf{x}_{N,k} = -\sigma \frac{\partial J_h(\mathbf{x}_{N,k})}{\partial \mathbf{x}_N}$$

$$J_h(\mathbf{x}_{N,k+1}) \approx J_h(\mathbf{x}_{N,k}) + \frac{\partial J_h(\mathbf{x}_{N,k})}{\partial \mathbf{x}_N}(\mathbf{x}_{N,k+1} - \mathbf{x}_{N,k})$$

$$J_h(\mathbf{x}_{N,k+1}) - J_h(\mathbf{x}_{N,k}) \approx -\sigma \left\| \frac{\partial J_h(\mathbf{x}_{N,k})}{\partial \mathbf{x}_N} \right\|^2$$

So that, for a σ small enough to guarantee the validity of the linear approximation of J, the reduced gradient gives a descent direction of J

GRG

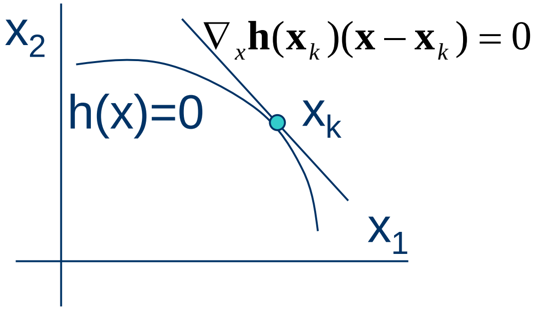
The main problem of this strategy is associated to the fact that the linear approximation of h(x) leads to points that do not satisfy the non-linear constraint h(x) = 0. It is not difficult to see that the points of the hyperplane

$$\nabla_x \mathbf{h}(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k)=0$$

Do not coincide, in general with h(x) = 0. So, if we use

$$\mathbf{x}_B-\mathbf{x}_{Bk}=-\mathbf{B}(\mathbf{x}_k)^{-1}\mathbf{N}(\mathbf{x}_k)(\mathbf{x}_N-\mathbf{x}_{Nk})$$

to compute x_B from x_N, in general they will not satisfy h(x) = 0.



As the relation between the change in x_N and the change in x_B is $-B^{-1}N$, this policy is equivalent to use on x:

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\sigma\begin{bmatrix}-\mathbf{B}^{-1}\mathbf{N}\mathbf{d}\\ \mathbf{d}\end{bmatrix}$$

GRG

The correct strategy is to compute the m components x_B at iteration k+1 , from the non-linear constraints h(x)=0 so that they are satisfied:

$$\mathbf{h}(\mathbf{x}_{B,k+1}, \mathbf{x}_{N,k} - \sigma\mathbf{d}) = 0$$

For this purpose, the Newton's method can be used:

$$\mathbf{x}_{B,k+1}^{j+1} = \mathbf{x}_{B,k+1}^{j} - \left[\frac{\partial \mathbf{h}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d})}{\partial \mathbf{x}_B}\right]_k^{-1} \mathbf{h}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d}) =$$

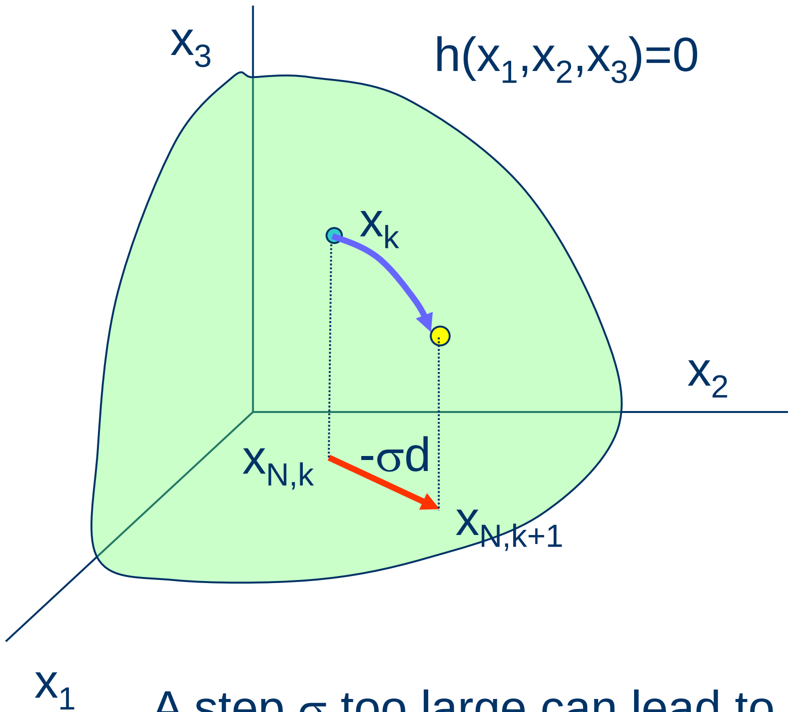
$$= \mathbf{x}_{B,k+1}^{j} - \mathbf{B}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d})^{-1} \mathbf{h}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d})$$

If it does not converge, σ must be reduced and the iterations started again. An initial estimate of $x_{B,k+1}$ can be obtained from:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \sigma \begin{bmatrix} -\mathbf{B}^{-1}\mathbf{N}\mathbf{d} \\ \mathbf{d} \end{bmatrix}$$

Then, one should check that J(x) improves in x_{k+1}

GRG



A step σ too large can lead to points where no x_3 could satisfy $h(x_1,x_2,x_3)=0$

Example: One single constraint, x_3 basic variable, x_1, x_2 non basic variables

The optimization is performed on x_1, x_2 using the reduced gradient d and then x_3 is adjusted so that (x_1, x_2, x_3) is on the surface defined by $h(x_1,x_2,x_3)=0$.

This is equivalent to the use of a gradient vector projected on the constraint surface

GRG Example

$$\left.\begin{array}{l}\min_{\mathbf{x}} J(\mathbf{x}) = 4x_1 - x_2^2 + x_3^2 - 12 \\ 20 - x_1^2 - x_2^2 = 0 \\ x_1 + x_3 - 7 = 0\end{array}\right\} \quad \mathbf{x} \text{ feasible initial} : \mathbf{x}_{(1)} = \begin{pmatrix} 2 \\ 4 \\ 5 \end{pmatrix} \quad \begin{array}{l}\text{non basic} : x_1 \\ \text{basic} : x_2, x_3\end{array}$$

$$\nabla J(\mathbf{x}_{(1)})' = \begin{pmatrix} 4 \\ -2x_2 \\ 2x_3 \end{pmatrix}_{\mathbf{x}_{(1)}} = \begin{pmatrix} 4 \\ \hdashline -8 \\ 10 \end{pmatrix}, \nabla \mathbf{h}(\mathbf{x}_{(1)}) = \begin{pmatrix} -2x_1 & -2x_2 & 0 \\ 1 & 0 & 1 \end{pmatrix}_{\mathbf{x}_{(1)}} = \left(\begin{array}{c:cc} -4 & -8 & 0 \\ 1 & 0 & 1 \end{array}\right)$$

N B

The reduced gradient is:

$$\mathbf{d} = \frac{\partial J_h}{\partial \mathbf{x}_N} = \frac{\partial J}{\partial \mathbf{x}_N} - \frac{\partial J}{\partial \mathbf{x}_B} \mathbf{B}(\mathbf{x}_{(k)})^{-1} \mathbf{N}(\mathbf{x}_{(k)}) = (4) - (-8, 10) \begin{pmatrix} -8 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} -4 \\ 1 \end{pmatrix} = -2$$

GRG Example

As d = -2 and the non-basic variable is x_1, we will optimize $J_h(x_N)=J_h(x_1)$ moving x_1 en la dirección –d a certain amount σ, e.g. $\sigma = 0.4$, so that the new $x_{1(2)}$ would be: $x_{1(1)} - \sigma d = 2 - 0.4.(-2) = 2.8$

The other components of the new $\mathbf{x}_{(2)}$ would be computed so that the constraints h_1 and h_2, are satisfy, by solving:

$$\left.\begin{array}{l} 20 - 2.8^2 - x_2^2 = 0 \\ 2.8 + x_3 - 7 = 0 \end{array}\right\}$$

This simple example can be solved analytically, giving, ($x_2 = \pm 3.487$, $x_3 = 4.2$) but, in general, the Newton's method should be used. A iteration of it would be:

$$\mathbf{x}_{B,k+1}^{j+1} = \mathbf{x}_{B,k+1}^{j} - \mathbf{B}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d})^{-1}\mathbf{h}(\mathbf{x}_{B,k+1}^{j}, \mathbf{x}_{N,k} - \sigma\mathbf{d})$$

Starting from:

$$\mathbf{x}_{\mathrm{B,k+1}}^{0} = \mathbf{x}_{\mathrm{B,k}} - \sigma(-\mathbf{B}^{-1}\mathbf{N}\mathbf{d}) = \begin{pmatrix} 4 \\ 5 \end{pmatrix} - 0.4\begin{pmatrix} -8 & 0 \\ 0 & 1 \end{pmatrix}^{-1}\begin{pmatrix} -4 \\ 1 \end{pmatrix}(-2) = \begin{pmatrix} 3.6 \\ 4.2 \end{pmatrix}$$

Initial estimate

GRG Example

$$\mathbf{x}^1_{B,k+1} = \mathbf{x}^0_{B,k+1} - \mathbf{B}(\mathbf{x}^0_{B,k+1}, 2.8)^{-1}\mathbf{h}(\mathbf{x}^0_{B,k+1}, 2.8) =$$

$$= \begin{pmatrix} 3.6 \\ 4.2 \end{pmatrix} - \begin{pmatrix} -2x_2 & 0 \\ 0 & 1 \end{pmatrix}^{-1}_{\begin{pmatrix} 2.8 \\ 3.6 \\ 4.2 \end{pmatrix}} \begin{pmatrix} 20 - x_1^2 - x_2^2 \\ x_1 + x_3 - 7 \end{pmatrix}_{\begin{pmatrix} 2.8 \\ 3.6 \\ 4.2 \end{pmatrix}} =$$

$$= \begin{pmatrix} 3.6 \\ 4.2 \end{pmatrix} - \begin{pmatrix} -7.2 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} -0.8 \\ 0 \end{pmatrix} = \begin{pmatrix} 3.49 \\ 4.2 \end{pmatrix}$$

And it will continue until the solution (3.487,4.2) is reached.

The new $\mathbf{x}_{(2)}$ would be, then: (2.8,3.487,4.2) and another iteration of the GRG algorithm could be started.

Nevertheless, before this, an improvement of J should be checked:

$$J(\mathbf{x}_{(2)}) = 4\;\;2.8 - 3.487^2 + 4.2^2 - 12 = 4.66 < J(\mathbf{x}_{(1)}) = 4\;\;2 - 4^2 + 5^2 - 12 = 5$$

If there would be no improvement, then σ should be reduced. After several iterations the final solution is: (2.5,3.71,4.2) where $\nabla_h J=0$

GRG - inequalities

The more general case where both equality and inequality constraints are present, is approached in a similar way to SQP, by transforming inequality into equality equations using additional slack variables :

$$\min_{x} J(\mathbf{x}) \qquad\qquad \Rightarrow \qquad\qquad \min_{\mathrm{x},\varepsilon} \mathrm{J}(\mathbf{x})$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \qquad\qquad \mathbf{h}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \qquad\qquad \mathbf{g}(\mathbf{x}) + \boldsymbol{\varepsilon} = \mathbf{0}$$

$$\boldsymbol{\varepsilon} \geq \mathbf{0}$$

And the decision vector is extended with the slack variables:

z = [x , ε]

So that we can consider problems with the format:

$$\left.\begin{array}{l} \min_{\mathrm{z}} \mathrm{J}(\mathbf{z}) \\ \mathrm{c}(\mathbf{z}) = \mathbf{0} \\ \mathbf{m} \leq \mathbf{z} \leq \mathbf{M} \end{array}\right\} \quad \mathrm{c}(\mathbf{z}) = \begin{bmatrix} \mathrm{h}(\mathrm{x}) \\ \mathrm{g}(\mathrm{x}) + \varepsilon \end{bmatrix}$$

The new inequalities generated by the slack variables are considered implicitly in the steps of the GRG

GRG - inequalities

Some aspects to be considered in the implicit treatment of the inequalities associated to the slack variables:

1. Select as basic only those that are not very close to the constraints, so that the non basic variables can be changed within a certain range
2. Modify the search direction d, so that the constraints associated to the slack variables are not violated if x_N is moved in the direction –d
3. Check that the constraints associated to the slack variables are not violated when the step length σ is adjusted as well as when x_B is computed in order to satisfy the equality constraints

GRG is an efficient method up to several hundred of constraints and decision variables

Sequential solution using a simulator

It is based on an idea similar to GRG

Optimizer of $J(x)$ with respect to a subset of n-m variables x_b

x dim n
$h(x) = 0$ dim m < n

n - m variables x_b

Values of $J(x)$, $g(x)$

Numerical solution of $h(x) = 0$ to compute the values of the remaining m x_b
Computation of $J(x)$, $g(x)$

Sequential solution using a simulator

$$\min_{x} J(\mathbf{x}_b, \mathbf{x}_d)$$
$$\mathbf{h}(\mathbf{x}_b, \mathbf{x}_d) = \mathbf{0}$$
$$\mathbf{g}(\mathbf{x}_b, \mathbf{x}_d) \leq \mathbf{0}$$
$$\mathbf{m} \leq \mathbf{x}_b \leq \mathbf{M}$$

$$\min_{x,\varepsilon} J(\mathbf{x}_b, \mathbf{x}_d)$$
$$\mathbf{h}(\mathbf{x}_b, \mathbf{x}_d) = \mathbf{0}$$
$$\mathbf{g}(\mathbf{x}_b, \mathbf{x}_d) + \boldsymbol{\varepsilon} = \mathbf{0}$$
$$\mathbf{m} \leq \mathbf{x}_b \leq \mathbf{M} \quad \boldsymbol{\varepsilon} \geq \mathbf{0}$$

$$\min_{x_b,\varepsilon} J(\mathbf{x}_b, \mathbf{x}_d(\mathbf{x}_b))$$
$$\mathbf{g}(\mathbf{x}_b, \mathbf{x}_d(\mathbf{x}_b)) + \boldsymbol{\varepsilon} = \mathbf{0}$$
$$\mathbf{m} \leq \mathbf{x}_b \leq \mathbf{M} \quad \boldsymbol{\varepsilon} \geq \mathbf{0}$$

$$\min_{x_b,\varepsilon} J(\mathbf{x}_b, \mathbf{x}_d(\mathbf{x}_b))$$
$$\mathbf{g}(\mathbf{x}_b, \mathbf{x}_d(\mathbf{x}_b)) \leq \mathbf{0}$$
$$\mathbf{m} \leq \mathbf{x}_b \leq \mathbf{M}$$

x_b boundary variables, dim n-m
x_d dependent variables dim m
ε Slack variables dim r

GRG

$$\left.\begin{array}{l}\min\limits_{\mathbf{z}} J(\mathbf{z}) \\ c(\mathbf{z}) = \mathbf{0} \\ \mathbf{m} \leq \mathbf{z} \leq \mathbf{M}\end{array}\right\} \quad \begin{cases}\nabla_z J(\mathbf{z}) + \boldsymbol{\lambda}'\nabla_z \mathbf{c}(\mathbf{z}) - \boldsymbol{\mu}' + \boldsymbol{\eta}' = \mathbf{0} \\ \mathbf{c}(\mathbf{z}) = \mathbf{0}, \quad \mathbf{m} \leq \mathbf{z} \leq \mathbf{M} \quad \boldsymbol{\mu}'(\mathbf{m} - \mathbf{z}) = 0 \quad \eta'(\mathbf{M} - \mathbf{z}) = 0 \\ \boldsymbol{\mu} \geq 0, \quad \boldsymbol{\eta} \geq 0\end{cases}$$

The KKT conditions are:

Given a point of the solution z_k of size n, its components are partitioned in two groups:
m dependent variables, z_d
n - m independent or boundaries z_b
$z_k = [z_b, z_d]$

Reduced gradient

$$c(z_b, z_d) = 0$$

This equation can be used to write the z_d components as functions of z_b

$$\frac{\partial c}{\partial z_d} dz_d + \frac{\partial c}{\partial z_b} dz_b = 0 \quad \Rightarrow \quad \frac{dz_d}{dz_b} = \left[\frac{\partial c}{\partial z_d}\right]^{-1} \frac{\partial c}{\partial z_b}$$

$$\frac{dJ(z_b, z_d(z_b))}{dz_b} = \frac{\partial J}{\partial z_b} + \frac{\partial J}{\partial z_d}\frac{dz_d}{dz_b} = \frac{\partial J}{\partial z_b} + \frac{\partial J}{\partial z_d}\left[\frac{\partial c}{\partial z_d}\right]^{-1}\frac{\partial c}{\partial z_d}$$

Reduced gradient

$$\left.\begin{aligned} &\min_{z_s} J(\mathbf{z}_s) \\ &\mathbf{m} \le \mathbf{z} \le \mathbf{M} \end{aligned}\right\}$$

MINOS

$$
\left.\begin{aligned}
&\min_{x} J(\mathbf{z}) \\
&c(\mathbf{z}) = \mathbf{0} \\
&\mathbf{m} \le \mathbf{z} \le \mathbf{M}
\end{aligned}\right\}
$$

The constraints c(z)=0 are not enforced at every step, but are added via the aumented Lagrangian

1. Start from z_0
2. Linearize the active constraints in z_k : D_k z =v_k
3. Construct the aumented Lagrangian:

$$L = J(\mathbf{z}) + \lambda' c(\mathbf{z}) + \beta' \left\| c(\mathbf{z}) \right\|^2$$

4. Solve the linearize problem using GRG

$$
\begin{aligned}
&\min_{z} J(\mathbf{z}) + \lambda' c(\mathbf{z}) + \beta' \left\| c(\mathbf{z}) \right\|^2 \\
&D_k z = v_k \\
&m \le z \le M
\end{aligned}
$$

5. Go to 2, z_{k+1}= z, k = k +1, iterate until convergence

Cutting plane CP

These family of methods follow three main steps:

1 Formulate the problem in the form:

$$\min_{\mathbf{x}} \mathbf{c}'\mathbf{x}$$
$$\text{with } \mathbf{x} \in S \ \text{ convex}$$

2 Find a convex polytope containing S

3 Solve the NLP problem by means of a succession of LP problems

Cutting Plane CP (1)

1 A problem such as:

$$\min_{\mathbf{z}} f(\mathbf{z}) \qquad \text{convex}$$

$$\text{with } \mathbf{z} \in T \;\; \text{convex}$$

Is equivalent to

$$\min_{\mathbf{x}} u$$

with $\mathbf{x} = (\mathbf{z}\ , u)'$

$$\text{with}$$

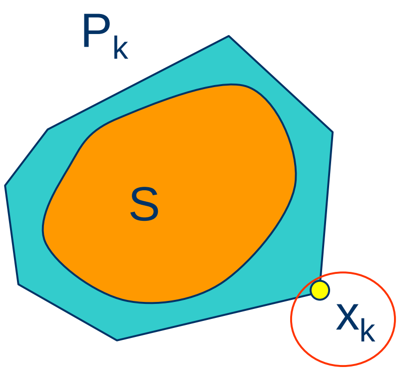
$$\mathbf{z} \in T \;\; \text{convex}$$

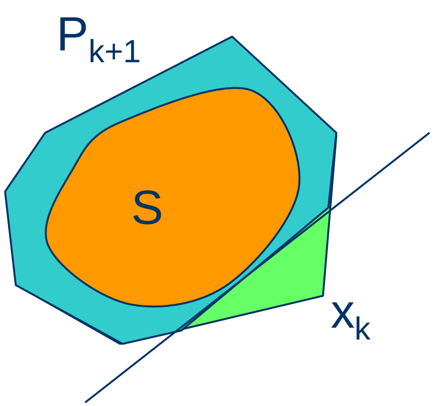
$$f(\mathbf{z}) - u \leq 0$$

That has the desired format

Cutting Plane CP (2)

$$\min_{\mathbf{x}} \mathbf{c}'\mathbf{x}$$
$$\text{with } \mathbf{x} \in S \text{ convex}$$





If a polytope P_k containing S is found, the problem: minimize c'x on P_k is a LP one.

If the solution of this LP is $x_k \in S$, then x_k is also the solution of the the original NLP

If x_k does not belong to S, then, a cutting plane separating x_k from S, will be added which will originate a new polytope P_{k+1} closer to S

The problem of minimizing c'x on P_{k+1} is repeated until a solution, or an adequate approximation, is found.

The different CP algorithms differ in the way the polytope or the cutting planes are generated

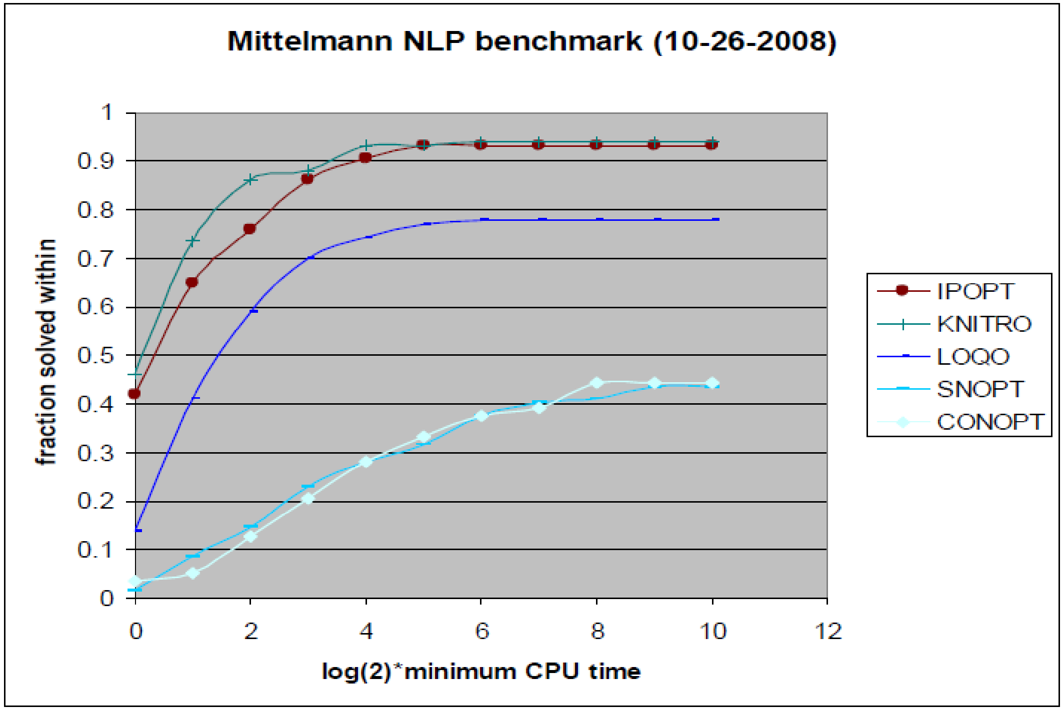
NLP Software

- There are two main types of NLP software :
  - **Solvers** : routines implementing algorithms that can be called from a certain environment or programming language, usually as dll's, providing the solution (MINOS, OSL, Matlab Optimization Toolbox, TOMLAB, NAG, NPSOL, CONOPT, IMSL,...)
  - **Modelling environments**: They are software environment that facilitate the modelling, solution, analysis and management of the NLP problem, formulating it in a particular language (GAMS, XPRESS-MP, AMPL, AIMSS, Gurobi,...) or format (Excel). They call automatically one or several solvers to compute the solution
  - **Free software**: http://www.gams.com/, http://www.gurobi.com/
- Key points when searching for the optimum are the computation of the derivatives, the selection of the initial point and the existence of local minimums

Software NLP

- SQP: NPSOL, Fmincon
- rSQP: SNOPT, MUSCOD, LSSOL,…
- Reduced Gradient: GRG2, SOLVER, CONOPT
- Reduced Gradient (without rest.): MINOS
- Interior point: IPOPT, KNITRO, LOQO

Comparative study of NLP solvers



	Limits	Fail
IPOPT	7	2
KNITRO	7	0
LOQO	23	4
SNOPT	56	11
CONOPT	55	11

117 test problems
500 - 250 000 variables, 0 – 250 000 constraints

NLP Software / Derivatives

Most of these methods require the evaluation of the first derivatives of the cost function J and the constraints with respect to x. If they are not supplied by the user, the solvers may estimate them using finite differences:

$$\frac{J(\mathbf{x}+\Delta\mathbf{x})-J(\mathbf{x})}{\Delta\mathbf{x}} \qquad \frac{J(\mathbf{x}+\Delta\mathbf{x})-J(\mathbf{x}-\Delta\mathbf{x})}{2\Delta\mathbf{x}}$$

Central differences are more precise but they increase the computation time. Usually, relative changes in Δx are in the order of 10^{-6} or 10^{-7} providing good accuracy. Nevertheless, if obtaining J implies the solution of systems of equations, simulations, etc, then, Δx should be increased. As a general rule, the precision of the internal computations should be one or two orders of magnitude higher than the one required in the optimization.

Alternatively, many modelling environments provide automatic differentiation, which increases the accuracy of the results

Software NLP

Once an optimization problem has been stated, it is convenient to re-formulate it in such a way that numerical problems are avoided and the efficiency in the searching of the solution is increased.

Among possible changes we can mention:

✓Scaling the decision variables

✓Changes of variables to avoid computations out of range: log(x), $x^{1/2}$, …

✓Changes of variables to avoid non differentiability, discontinuities,..

✓Changes of variables to improve the convexity of the problem

Formulate the problem avoiding potential numerical problems

$$x^2 + \log(z) \le 3 \quad \Rightarrow \quad \begin{cases} x^2 + y \le 3 \\ \exp(y) = z \end{cases}$$

$$v\sqrt{xy - z^3} = 3 \quad \Rightarrow \quad \begin{cases} vu = 3 \\ u^2 = xy - z^3 \\ u \ge 0 \end{cases}$$

$$\min_x \quad |x| \qquad \boxed{\Rightarrow \begin{cases} \min\limits_u \quad u \\ -u \le x \le u \end{cases}}$$

$$|x| \le a \qquad \Rightarrow \qquad -a \le x \le a$$

Formulate the problem avoiding potential numerical problems

Add constraints in order to avoid non-desirable solutions of equality constraints

$h(x) = 0 \Rightarrow h(x) = 0,\ a \le x \le b$

Exploit problem structure

Min [zx – 3zy]
s.t. xz + y - z y = 2
4x – 5zy + zx = 9
0 ≤ z ≤ 1
Non-convex NLP problem

$\Rightarrow$

It is LP for a fix z

Min zx – 3zy
0 ≤ z ≤ 1

LP in z

$\Downarrow$ z

x,y

Min [zx – 3zy]
s.t. xz + y - z y = 2
4x – 5zy + zx = 9

LP in x,y

Convexification

$$J(x_1, x_2) = x_1 x_2$$

Non convex in x

$$x_1 = e^{v_1} \qquad x_2 = e^{v_2}$$

Change of variables

$$x_1 x_2 = e^{v_1} e^{v_2} = e^{v_1 + v_2}$$

$$\min_{x_1, x_2} J(x_1, x_2) = \min_{v_1, v_2} J(v_1, v_2)$$

Convex function in v

Software NLP

One important problem in NLP is to know if the optimum proposed by the algorithm is a local or global one.

In general, except if the problem is a convex one, we cannot guarantee that the solution is a global one. In order to improve the chances of obtaining a global solution, three kind of approaches are usually used:

✓Multistart: Repeat the problem starting with different initial points spread over the feasible set. If all of them finish in the same point, this gives a certain confidence in the solution found.

✓Convexification: Reformulate the problem so that a new equivalent convex problem is found and then solve this problem.

✓Global optimization: Choose a global optimization algorithm. Deterministic global methods, such as BARON, are very slow while evolutionary algorithms do not provide real guarantee that the global optimum is found.

NLP Software

Finally, another important point to consider when solving the NLP problem is the tuning of the parameters of the algorithms, which appear, either in the evaluation of the optimality conditions, or in the intermediate steps of the algorithm, which themselves are LP, QP, steepest descend, etc. problems.

$$\frac{\left|J(\mathbf{x}_{k+1}) - J(\mathbf{x}_k)\right|}{\varepsilon_0 + \left|J(\mathbf{x}_k)\right|} \leq \varepsilon_3 \qquad \frac{\left\|\mathbf{x}_{k+1} - \mathbf{x}_k\right\|}{\varepsilon_0 + \left\|\mathbf{x}_k\right\|} \leq \varepsilon_2$$

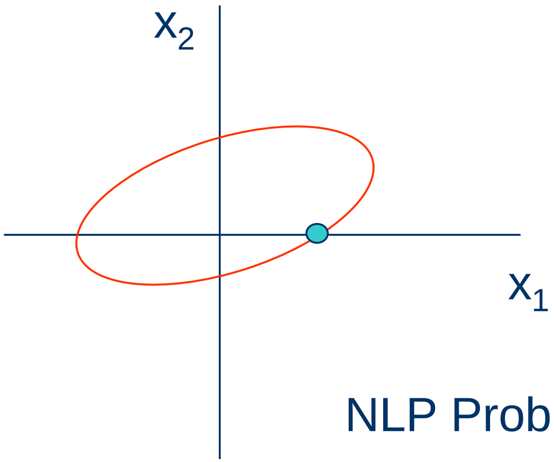
Changes in the function J or the value of x

$$g_j(\mathbf{x}_k) \leq \varepsilon_j \qquad \left\|h_i(\mathbf{x}_k)\right\| \leq \varepsilon_i$$

Tolerances in the constraints

Maximum number of iterations,….

Minimum distance



Find the closest point to the origin of the curve on the first quadrant:

$$5x_1^2 + 6x_1x_2 + 5x_2^2 = 8$$

NLP Problem:

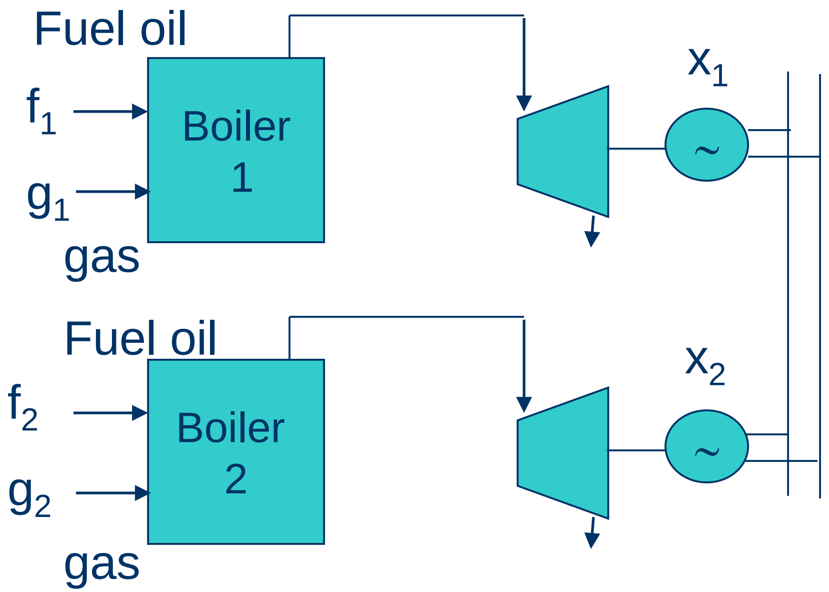
$$\min_{\mathbf{x}} \quad \sqrt{x_1^2 + x_2^2}$$

under:

$$5x_1^2 + 6x_1x_2 + 5x_2^2 = 8$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

Aim: To generate 50 MW with minimum fuel oil consumption



Fuel consumption Kg/min to generate x_{ij} MW in each alternator

$$f_1 = 1.46 + 0.15x_{11} + 0.0014x_{11}^2$$

$$g_1 = 1.57 + 0.16x_{12} + 0.0013x_{12}^2$$

$$f_2 = 0.8 + 0.2x_{21} + 0.0009x_{21}^2$$

$$g_2 = 0.73 + 0.23x_{22} + 0.0008x_{22}^2$$

They can work at the same time with fuel oil and gas (adding powers)

x_{ij} power (MW) generated in alternator i with fuel j

Working range: Alternator 1 between 18 and 30 MW
Alternator 2 between 14 and 25 MW
Total flow of gas less than 10 Kg/min

$$\min_{x_{11},x_{12},x_{21},x_{22}} f_1 + f_2 = \min_{x_{11},x_{12},x_{21},x_{22}} 2.26 + 0.15x_{11} + 0.0014x_{11}^2 + 0.2x_{21} + 0.0009x_{21}^2$$

Power constraints:

$$x_{11} + x_{12} + x_{21} + x_{22} \geq 50$$
$$18 \leq x_{11} + x_{12} \leq 30$$
$$14 \leq x_{21} + x_{22} \leq 25 \qquad x_{11} \geq 0,\ x_{12} \geq 0,\ x_{21} \geq 0,\ x_{22} \geq 0,$$

Availability constraints

$$g_1 + g_2 = 1.8 + 0.16x_{12} + 0.0013x_{12}^2 + 0.23x_{22} + 0.0008x_{22}^2 \leq 10$$

$$\left.\begin{aligned} g_1 &= 1.57 + 0.16x_{12} + 0.0013x_{12}^2 \geq 0 \\ g_2 &= 0.73 + 0.23x_{22} + 0.0008x_{22}^2 \geq 0 \\ f_1 &= 1.46 + 0.15x_{11} + 0.0014x_{11}^2 \geq 0 \\ f_2 &= 0.8 + 0.2x_{21} + 0.0009x_{21}^2 \geq 0 \end{aligned}\right\}$$

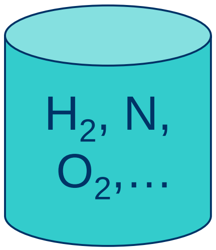
They are redundant, as they are always positive for $x_{ij} \geq 0$

Excel

Chemical equilibrium

A mixture of 10 chemical species (H, H_2, H_2O, N, N_2, NH, NO, O, O_2, OH) is in equilibrium at T=298 ºK and P = 750 mmHg. It is known that the species are made out only of hydrogen, nitrogen and oxygen, and the mixture behaves as an ideal gas.

Which is the composition of the mixture if we know that there are the following amounts of elements: 2 moles of H, 1 mol of N and one mol of O?



T = 298 ºK

P = 750 mmHg

Chemical equilibrium

j	Moles of j	w_j
H	x_1	-10.021
H_2	x_2	-21.096
H_2O	x_3	-37.986
N	x_4	-9.846
N_2	x_5	-28.653
NH	x_6	-18.918
NO	x_7	-28.032
O	x_8	-14.640
O_2	x_9	-30.594
OH	x_{10}	-26.111

At equilibrium, the Gibbs energy of the system must be minimal

Free energy per mol of component j:

$$G_j = RT[w_j + \ln(Py_j)]$$

y_j molar fraction of component j in the mixture

$$y_j = x_j \Big/ \sum_{i=1}^{10} x_i$$

Find the composition that minimizes the total Gibbs energy of the mixture:

$$G = \sum_{j=1}^{10} x_j G_j$$

Chemical equilibrium

$$\min_{\mathbf{x}} G = \sum_{j=1}^{10} x_j G_j = RT \sum_{j=1}^{10} x_j [w_j + \ln\left(Px_j \middle/ \sum_{i=1}^{10} x_i \right)]$$

Mass conservation of element i

a_{ij} moles of element i in one mol of specie j

	H	H_2	H_2O	N	N_2	NH	NO	O	O_2	OH
a_{1j} / H	1	2	2	0	0	1	0	0	0	1
a_{2j} / N	0	0	0	1	2	1	1	0	0	0
a_{3j} / O	0	0	1	0	0	0	1	1	2	1

$$z_i = \sum_{j=1}^{10} a_{ij} x_j \quad z_1 = 2, \quad z_2 = 1, \quad z_3 = 1$$

$$x_j \geq 0$$

NLP problem with linear constraints

GAMS

```
sets  c  compounds / H, H2, H2O, N, N2, NH, NO, O, O2, OH /
      i  atoms     / H  hydrogen, N  nitrogen, O  oxygen /

table  a(i,c)  atoms per compound
    H  H2  H2O  N  N2  NH  NO  O  O2  OH
H   1   2   2          1              1
N                  1   2   1   1
O              1              1  1   2   1

parameters mix(i)   number of moles in the mixture / h=2, n=1, o=1 /
           gibbs(c) Gibbs free energy coef at 3500 k  and 750 psi /
      H  -10.021,  H2  -21.096,  H2O -37.986,  N   -9.846, N2 -28.653
      NH -18.918, NO -28.032,  O  -14.640,   o2 -30.594, OH -26.11 /
           gplus(c) Gibbs free energy plus pressure ;
           gplus(c) = gibbs(c) + log(750*.07031); display gplus;
```

GAMS

```
variables x(c)    number of moles in the mixture
          xb      total number of moles in the mixture
          energy  total free energy of the mixture
positive variables x, xb;
equations cdef(i) compound definition
          edef    energy definition
          xdef    total number of moles definition ;

          cdef(i)..  sum(c, a(i,c)*x(c)) =e= mix(i);
          xdef..     xb =e= sum(c, x(c));
          edef..     energy =e= sum(c, x(c)*(gplus(c) + log(x(c)/xb)));
          x.lo(c) = .001; xb.lo = .01;

model mixer chemical mix for N2H4+O2 / all /;
solve mixer minimizing energy using nlp;
```

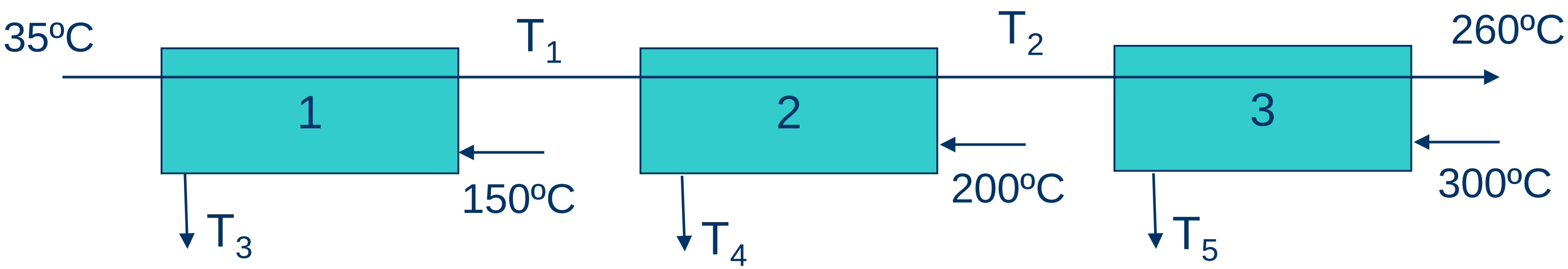
GAMS

---- VAR x number of mols in mixture

	LOWER	LEVEL	UPPER	MARGINAL
H	0.001	0.040	+INF	EPS
H2	0.001	0.146	+INF	.
H2O	0.001	0.785	+INF	EPS
N	0.001	0.001	+INF	EPS
N2	0.001	0.485	+INF	.
NH	0.001	0.001	+INF	0.371
NO	0.001	0.027	+INF	.
O	0.001	0.018	+INF	EPS
O2	0.001	0.037	+INF	EPS
OH	0.001	0.096	+INF	EPS

** Feasible solution. Value of objective = -47.3618693341

Minimum surface heat exchangers

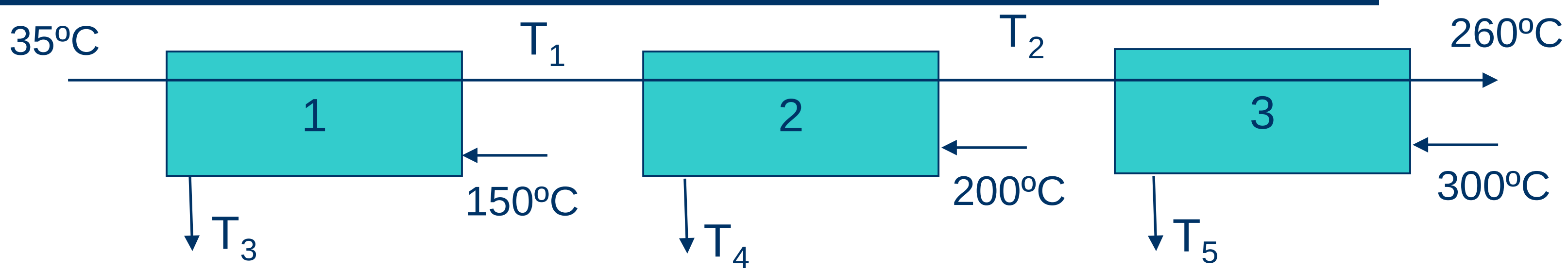


Heat exchanger	U (w/m2ºK)	Area (m2)
1	681	A_1
2	454	A_2
3	227	A_3

Size the heat exchangers so that the especifications can be satisfied and its total surface is minimum

$$q\rho c_p = 50000 Kcal/h^o K$$

Minimum surface heat exchangers



$\min \; A_1 + A_2 + A_3$

Energy balance:

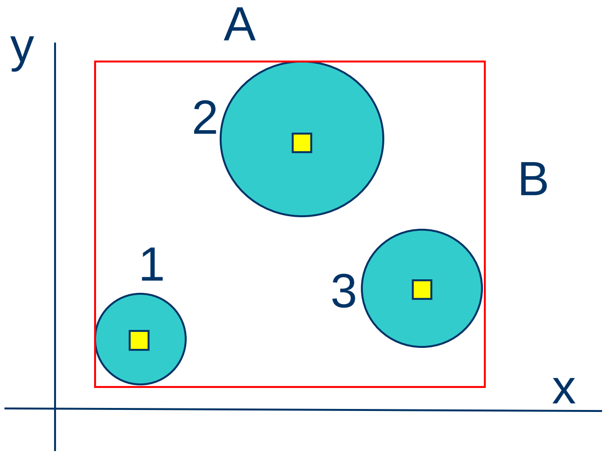
$$q\rho c_e (T_1 - 35) = U_1 A_1 \frac{(T_1 - 150) - (T_3 - 35)}{\ln(T_1 - 150) - \ln(T_3 - 35)} = F_1 \rho_1 c_{e1} (150 - T_3)$$

$$q\rho c_e (T_2 - T_1) = U_2 A_2 \frac{(200 - T_2) - (T_4 - T_1)}{\ln(200 - T_2) - \ln(T_4 - T_1)} = F_2 \rho_2 c_{e2} (200 - T_4)$$

$$q\rho c_e (260 - T_2) = U_3 A_3 \frac{(40) - (T_5 - T_2)}{\ln(40) - \ln(T_5 - T_2)} = F_3 \rho_3 c_{e3} (300 - T_5)$$

$$A_i \ge 0, T_1 \ge 35, T_1 \le 150 - \Delta, T_3 \ge 35, T_2 \ge T_1, T_2 \le 200 - \Delta, T_4 \ge T_1, T_2 \le 260, T_5 \ge T_2$$

Placement



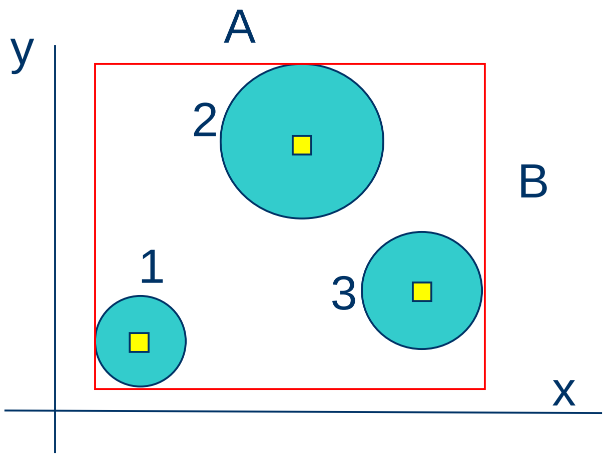
Three cylindrical storage tanks must be placed in a site (first quadrant) and enclosed with a wall, which is the best placement in order to minimize the wall length?

Tank	Radius r (m)
1	5
2	15
3	10

(x_i, y_i) coordinates of the cylinder I centre

A, B sizes of the wall length and width

Placement



Tank	Radius r (m)
1	5
2	15
3	10

(x_i,y_i) coordinates of the cylinder centre

A, B sizes of the wall length and width

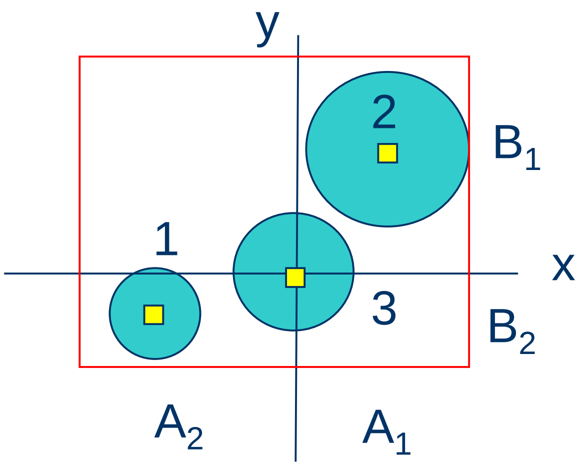
$$\min_{x_i, y_i, A, B} 2(A+B)$$

$$\left.\begin{array}{l} (x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2 \\ |x_i - x_j| + r_i + r_j \leq A \\ |y_i - y_j| + r_i + r_j \leq B \end{array}\right\} \left\{\begin{array}{l} i=1,\ j=2 \\ i=1,\ j=3 \\ i=2,\ j=3 \end{array}\right.$$

$$x_i \geq r_i \quad y_i \geq r_i \quad i = 1,2,3 \quad A \geq 0, B \geq 0$$

Multiplicity of solutions due to the problem symmetry. Discontinuities in the derivatives

Placement: alternative



Tank 3 is placed at the origen, so that there are only two tanks to place

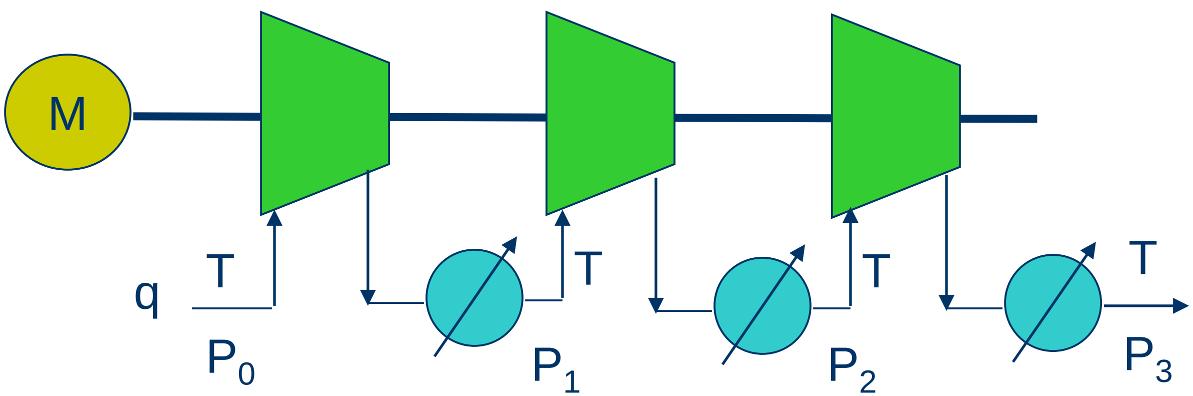
Here all derivatives are continuous

$$\min_{x_i, y_i, A_1, A_2, B_1, B_2} (A_1 + A_2 + B_1 + B_2)$$

$$\left.\begin{array}{l} (x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2 \\ x_i + r_i \leq A_1 \\ -A_2 \leq x_i - r_i \\ y_i + r_i \leq B_1 \\ -B_2 \leq y_i - r_i \end{array}\right\} \left\{\begin{array}{l} i = 1, j = 2,3 \\ i = 2, j = 1.3 \\ i = 1,2,3 \end{array}\right.$$

$$A_1 \geq 0, B_1 \geq 0, A_2 \geq 0, B_2 \geq 0$$

Three stages compressor



The power consumed by a reversible adiabatic compressor which input temperature is T, is given by:

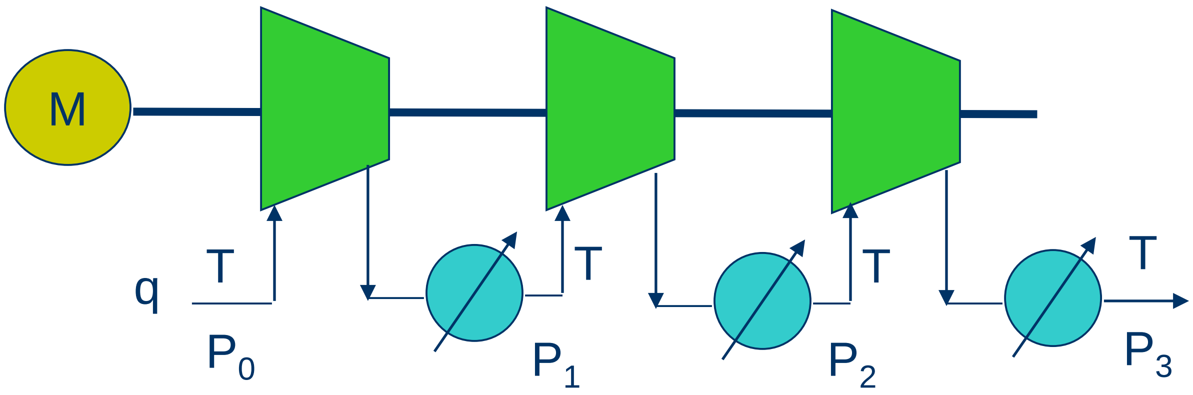
q mol/h   T ºK $\gamma = 4/3$

$$W = qRT\frac{\gamma}{\gamma-1}\left[\left(\frac{P_{sal}}{P_{ent}}\right)^{\frac{\gamma-1}{\gamma}} - 1\right]$$

$$\gamma = \frac{c_p}{c_v} \qquad R = \text{constante gases}$$

If the gas enters at 1bar and must be compressed up to 64 bars maintaining q and T constants, which must be the intermediate working pressures in order to consume the minimum energy?

Three stages compressor



The total power consumed will be the sum of the power consumed by each stage:

$$W_{Total} = qRT4\left[\left(\frac{P_1}{P_0}\right)^{\frac{1}{4}} + \left(\frac{P_2}{P_1}\right)^{\frac{1}{4}} + \left(\frac{P_3}{P_2}\right)^{\frac{1}{4}} - 3\right]$$

$$P_1 \geq P_0, \quad P_1 \leq P_2 \leq P_3$$

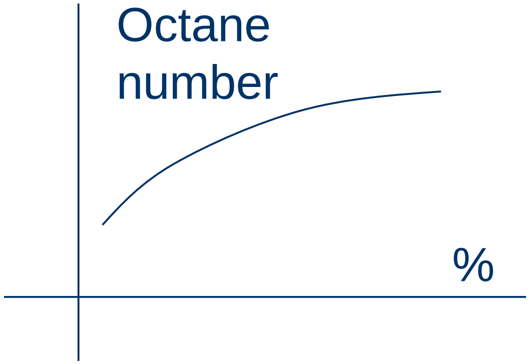
$$\min_{P_1, P_2} P_1^{\frac{1}{4}} + \left(\frac{P_2}{P_1}\right)^{\frac{1}{4}} + \left(\frac{64}{P_2}\right)^{\frac{1}{4}}$$

$$P_1 \geq 1, \quad P_1 \leq P_2 \leq 64$$

Octane number in mixtures

In the blending operation of a refinery, several products with different properties, among them RON (Research Octane Number), are mixed to obtain a certain amont of commercial gasoline. Some of the properties of the mixture can be computed as a linear combination of the corresponding property of the different components. Nevertheless, this is not the case with some others such as RON.

The blending problem consists of determining the flows of the components with minimum cost that guarantees an octane number (and other properties) above a minimum, respecting component's availability and other possible constraints



Octane number in mixtures

Variables:

x_i flow of compound i

p_i price of component i

F desired total flow of the mixture

z_i octane number of component i

z_m octane number of the mixture

ϕ non-linear function

θ Minimum RON in the mixture

M_i maximum availability of component i

$$\min_{x_i} \sum_i p_i x_i$$

$$F = \sum_i x_i$$

$$z_m = \phi\left[\sum_i \frac{x_i}{F} z_i\right]$$

$$x_i \leq M_i$$

$$z_m \geq \theta$$

NLP problem

RON in mixtures

The previous formulation is non-linear. In order to simplify the solution, the Blending Index method can be applied, which transform the problem in a LP one. It consist of a change of variable $w_i = B_i(z_i)$, specific for each property i, such that it verifies:

$$Fw_m = FB(z_m) = \sum_i x_i B_i(z_i) = \sum_i x_i w_i$$

$$\min_{x_i, w_m} \sum_i p_i x_i$$

$$F = \sum_i x_i$$

$$Fw_m = \sum_i x_i w_i$$

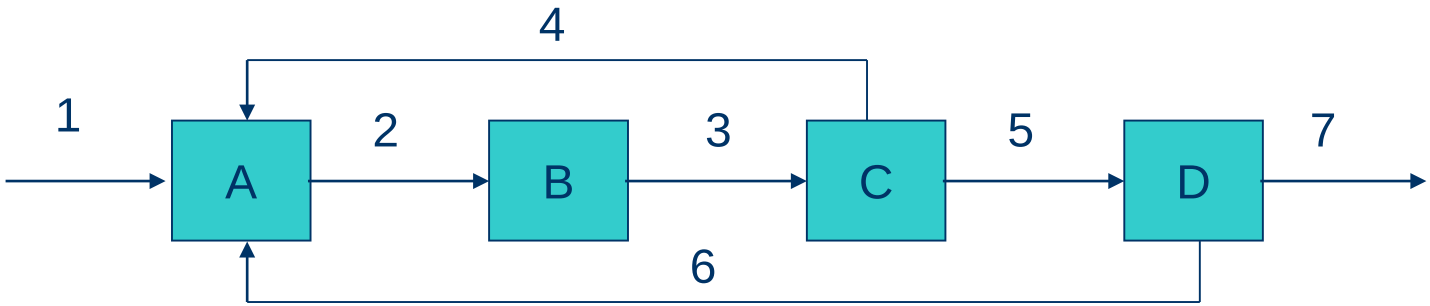
$$x_i \le M_i$$

$$w_m \ge B(\theta)$$

LP problem

Afterwords, the value of z_m can be recovered from $w_m = B(z_m)$

Data reconciliation (Rollins 93)

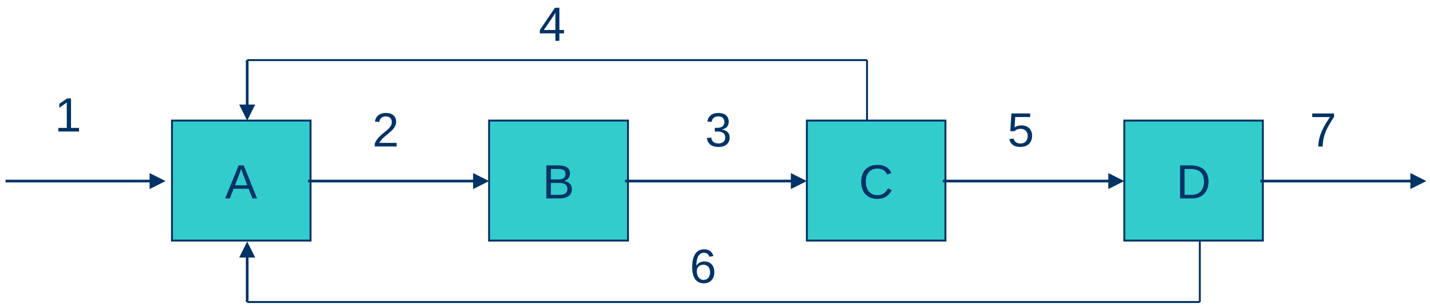


In the process represented in the figure the flows of the different streams (1 to 7) have been measured using transmitters with different accuracies, as in adjoint table

Stream	1	2	3	4	5	6	7
Value	49.5	81.5	85.3	10.1	72.9	25.7	50.7
Variance	1.5625	4.5156	4.5156	0.0625	3.5156	0.3906	0.3906

Which is the best coherent estimation of the real value of the flows?

Data reconciliation



Notice that the measurements are not coherent, e.g. a balance around the C unit gives: $F_3 \neq F_4 + F_5$ $83.5 \neq 10.1 + 72.9 = 83$, due to errors in the transmitters. One wish to correct them as less as possible according to its respective accuracy, so that the mass balances are satisfied.

Stream	1	2	3	4	5	6	7
Value	49.5	81.5	85.3	10.1	72.9	25.7	50.7
Variance	1.5625	4.5156	4.5156	0.0625	3.5156	0.3906	0.3906

Data reconciliation

Variables

F_{im} measured flow in stream i

F_i estimated flow in stream i

Aim

$$\min_{F_i} \sum_{i=1}^{7} \frac{1}{v_i} \left(\frac{F_i - F_{im}}{F_{im}} \right)^2$$

The relative corrections are made proportional to the inverse of the variance of each instrument

Some errors (losses, malfunctions, etc.) can be detected according to the size of the corrections

Constraints: The mass balances around each node must be fulfilled

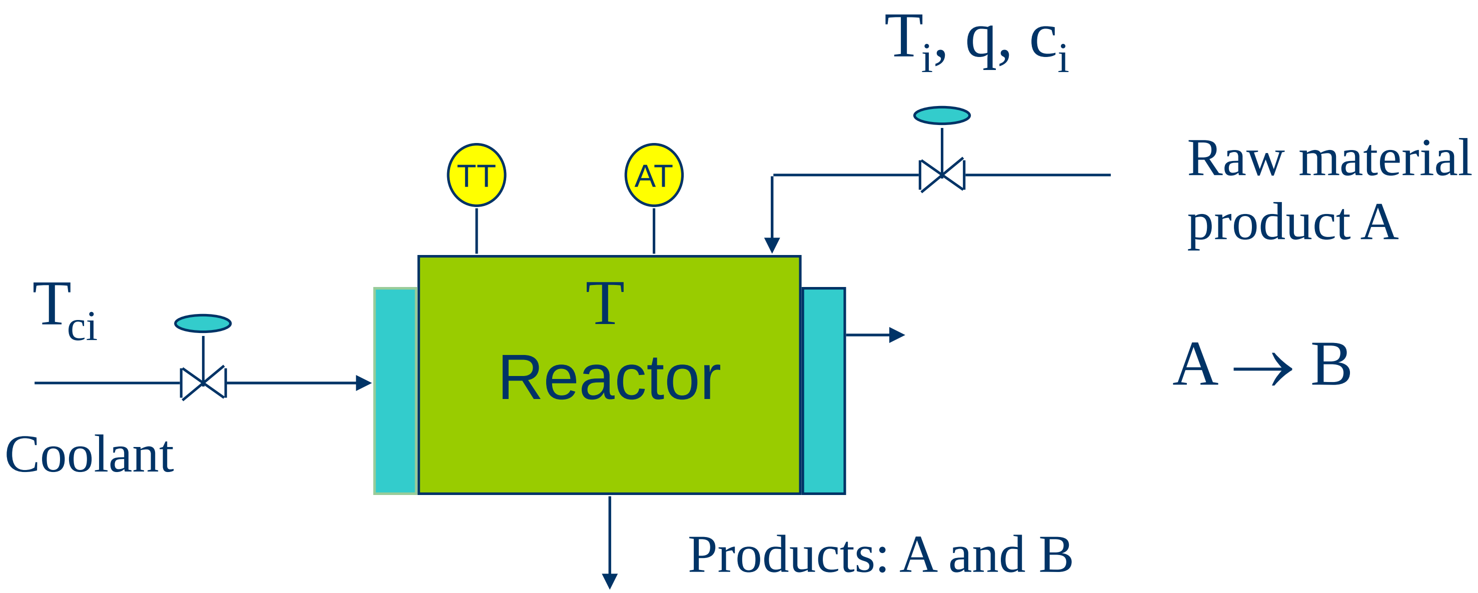
$$F_1 + F_4 + F_6 = F_2 \qquad F_2 = F_3$$

$$F_3 = F_4 + F_5 \qquad F_5 = F_6 + F_7$$

$$F_i \geq 0$$

Chemical reactor

Specifications: T_i, q, c_i, T_{ci}



Conservation of A and B

$$qc_i - qc_A - Vkc_A = 0$$

$$-qc_B + Vkc_A = 0$$

$$k = \beta e^{-E/RT}$$

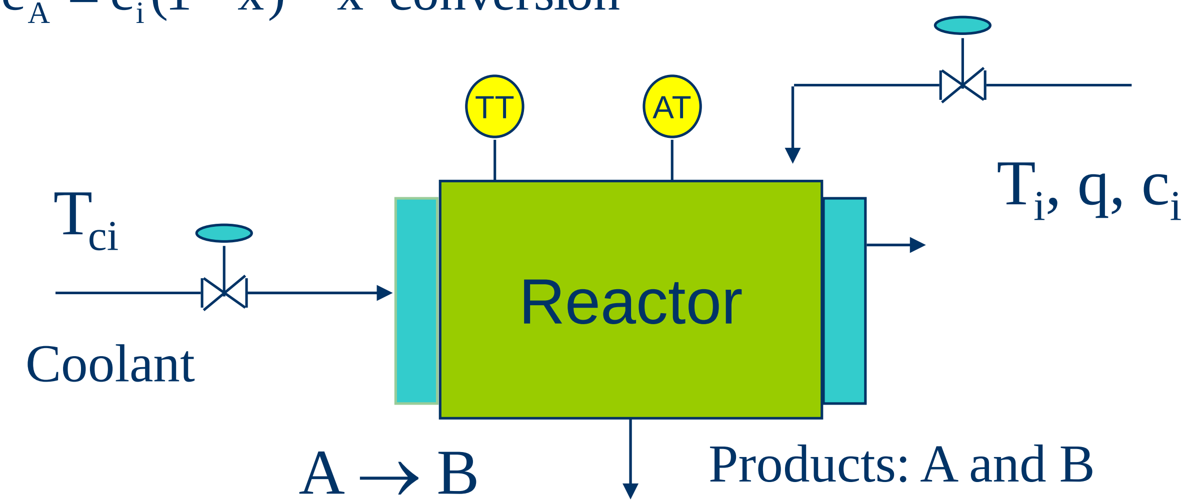
$$c_A = c_i(1 - x) \quad x \text{ conversion}$$

Energy conservation

$$Q = UA(T - T_c)$$

$$q\rho c_p T_i - q\rho c_p T + Vkc_A H - Q = 0$$

$$F\rho_j c_{pj}(T_{ci} - T_c) + Q = 0$$



Row material product A

Geometry:

$$V = \frac{\pi D^2}{4} L$$

$$A = \pi D L$$

Reactor design

$$qc_i - qc_A - V\beta e^{-E/RT} c_A = 0$$

$$-qc_B + V\beta e^{-E/RT} c_A = 0$$ C_B redundant

$$c_A = c_i(1-x) \quad x \text{ conversión}$$

$$q\rho c_p T_i - q\rho c_p T + Vkc_A H - UA(T - T_c) = 0$$

$$F\rho_j c_{pj}(T_{ci} - T_c) + UA(T - T_c) = 0$$

$$V = \frac{\pi D^2}{4} L$$

$$A = \pi D L$$

x, T and L can be selected within a range

Total variables: 14

q, c_i, c_A, c_B, V, T, x, T_i, A, T_c, F, T_{ci}, D, L

Equations: 7

Specifications: 4

q, c_i, T_i, T_{ci}

Degrees of freedom: 3 x, T, L

Manual Design from x,T,L

Given x, T and L :

Compute c_A : $c = (1- x)\, c_i$

Compute the size V: $V = qx / (k(1- x))$

Compute sizes D, A and the building cost

Compute Q $Q = -(c_i - c)qH - c_p q(T - T_i)$

Compute T_c $T_c = T - Q / (U\, A)$

Compute F $F = Q / (\rho_j\, c_{pj}\, (T_c - T_{ci}\,)$

Compute operation costs

If the design is not satisfactory, then, specify another x, T or L and start again

Degrees of freedom and optimization

- The problem can be formulated also as an optimization one where the values of the variables are selected so that, verifying the model equations, a set of constraints are satisfied and a certain cost function is minimized

$$T_{min} \leq T \leq T_{max}$$

$$x_{min} \leq x \leq x_{max}$$

$$L_{min} \leq L \leq L_{max}$$

$$c_A \geq 0 \quad c_B \geq 7$$

$$V \geq 0 \quad 1 \leq L/D \leq 3$$

$$T - T_c \geq 10$$

$$\ldots$$

$$\min \quad \text{construction cost} =$$
$$= 1916.9\, D^{1.066}\, L^{0.802}\ €$$

Notice that if the degrees of freedom are zero, then there is only a single solution and no room for optimization is left.

Two approaches First one: all variables are decision variables

$$\max_{x, T, L, D, F, \ldots} \text{Benefit} = \max \; - 1916.9\, D^{1.066} L^{0.802} + (q c_B price_B - q c_{Ai} price_{Ai} - F\, price_F)*time$$

under:

$$q c_i - q c_A - V\beta e^{-E/RT} c_A = 0$$

$$-q c_B + V\beta e^{-E/RT} c_A = 0$$

$$c_A = c_i(1-x)$$

$$q\rho c_p T_i - q\rho c_p T + V k c_A H - UA(T - T_c) = 0$$

$$F\rho_j c_{pj}(T_{ci} - T_c) + UA(T - T_c) = 0$$

$$V = \frac{\pi D^2}{4} L$$

$$A = \pi D L$$

$$T_{min} \le T \le T_{max}$$

$$x_{min} \le x \le x_{max}$$

$$L_{min} \le L \le L_{max}$$

$$c_A \ge 0 \quad c_B \ge 0$$

$$V \ge 0$$

$$1 \le L/D \le 4.$$

$$10 \le T - T_r \le \ldots$$

Second approach: only the degrees of freedom are decision variables

$$\max_{x,T,L} \text{Benefit} = \max \; - 1916.9\, D^{1.066} L^{0.802} + (qc_B \text{price}_B - qc_{Ai}\text{price}_{Ai} - F\,\text{price}_F)*\text{time}$$

Use only the degrees of freedom x, T and L as decision variables and compute the other variables by means of the equality constraints of the model.

A simulator is needed, there are no equality constraints and the inequality ones are evaluated in the simulator

$$qc_i - qc_A - V\beta e^{-E/RT} c_A = 0$$

$$-qc_B + V\beta e^{-E/RT} c_A = 0$$

$$c_A = c_i(1-x)$$

$$q\rho c_p T_i - q\rho c_p T + Vkc_A H - UA(T-T_c) = 0$$

$$F\rho_j c_{pj}(T_{ci} - T_c) + UA(T-T_c) = 0$$

$$V = \frac{\pi D^2}{4} L$$

$$A = \pi D L$$

Optimal design

$$\text{Max Benefit} = \max_{x,T,L} \; -3 * 1916.9\, D^{1.066} L^{0.802} + (qc_B \text{price}_B - qc_{Ai}\text{price}_{Ai} - F\,\text{price}_F)*\text{time}$$

under:

$$T_{min} \le T \le T_{max}$$
$$x_{min} \le x \le x_{max}$$
$$L_{min} \le L \le L_{max}$$
$$c_A \ge 0 \quad c_B \ge 0$$
$$V \ge 0$$
$$1 \le L/D \le 4.$$
$$10 \le T - T_r \le \ldots$$

$$qc_i - qc_A - V\beta e^{-E/RT} c_A = 0$$
$$-qc_B + V\beta e^{-E/RT} c_A = 0$$
$$c_A = c_i(1-x)$$
$$q\rho c_p T_i - q\rho c_p T + Vkc_A H - UA(T - T_c) = 0$$
$$F\rho_j c_{pj}(T_{ci} - T_c) + UA(T - T_c) = 0$$
$$V = \frac{\pi D^2}{4} L$$
$$A = \pi D L$$

Second approach: only the degrees of freedom are decision variables

NLP software

x, T, L ↓ ↑ J, g

Simulator

$$q c_i - q c_A - V\beta e^{-E/RT} c_A = 0$$

$$-q c_B + V\beta e^{-E/RT} c_A = 0$$

$$c_A = c_i(1-x)$$

$$q\rho c_p T_i - q\rho c_p T + V k c_A H - UA(T - T_c) = 0$$

$$F\rho_j c_{pj}(T_{ci} - T_c) + UA(T - T_c) = 0$$

$$V = \frac{\pi D^2}{4} L$$

$$A = \pi D L$$

Constraints evaluation

$$T_{min} \le T \le T_{max}$$

$$x_{min} \le x \le x_{max}$$

$$L_{min} \le L \le L_{max}$$

$$c_A \ge 0 \quad c_B \ge 0$$

$$V \ge 0$$

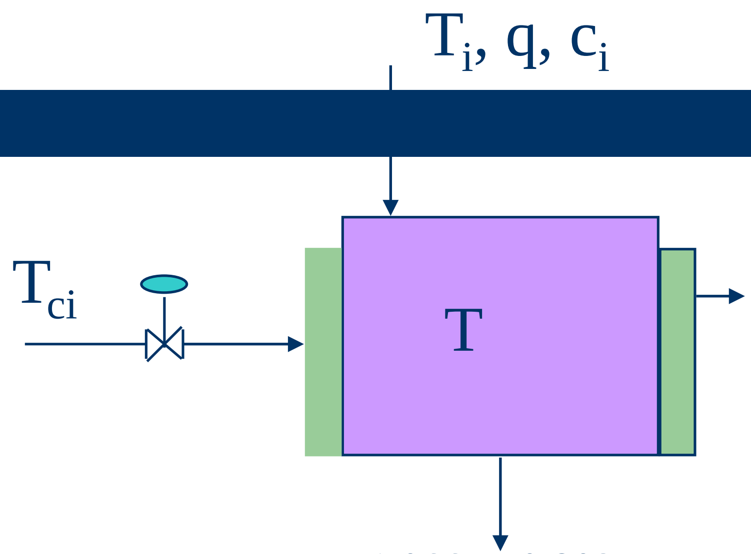
$$1 \le L/D \le 4.$$

$$10 \le T - T_r \le \ldots$$

Cost Function evaluation

J =……

Example



$Cost = 575.7\ D^{1.066}\ L^{0.802}$

Solution: T= 65, D = L = 3.03

$T_r = 51.5$ $F_r = 58.4$ $x = 0.8$

J = 4565.04 €

Specifications:

$q = 2.832\ m^3/h$

$k = 0.5\ h^{-1}$

$H = 69.710,5$ kJ/kmol

$T_i = T_{ri} = 21.11$ ºC

$\rho = 800.8\ Kg/m^3$

c_i=15 $kmol/m^3$

U = 6129 $kJ/h\ m^2 \cdot K$

$c_p = 0,968\ kJ/kg \cdot K$

$c_{pc} = 1,291\ kJ/kg \cdot K$

$\rho_j = 1041.1\ Kg/m^3$

jacket width 0.1 m.

$1 < L / D < 3$ $F_r < 90$

$10 < T - T_r < 30$

$0.8 < x < 0.95$

Reduced space SQP (rSQP)

- Recommended for large scale problems with few degrees of freedom.

QP problem to be solved at every step

$$\min_{\Delta x} \nabla_x J(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}' H_k \Delta\mathbf{x}$$

$$H_k = \nabla_x^2 L(\mathbf{x_k}, \boldsymbol{\lambda}_\mathbf{k}, \mu_k, \eta_k)$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x} \le \mathbf{M}$$

rSQP moves at every step in two separate directions. One fulfils the linearized equality constraints, the other moves along these constraints improving the cost respecting the inequalities

Codes: SNOPT, MUSCOD,..

Associated Lagrangean:

$$L = \nabla_x J(\mathbf{x_k})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}'\mathbf{H}_k\Delta\mathbf{x} + \lambda'(\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x}) +$$
$$+ \mu'(m - \mathbf{x}_k - \boldsymbol{\Delta}\mathbf{x}) + \eta'(\mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x} - \mathbf{M})$$

rSQP KKT conditions

$$\nabla_x J(\mathbf{x_k}) + \Delta\mathbf{x}'\mathbf{H}_k + \boldsymbol{\lambda}'\nabla_x \mathbf{h}(x_k) - \mu' + \eta' = 0$$

$$\mathbf{h}(\mathbf{x_k}) + \nabla_x \mathbf{h}(\mathbf{x}_k)\Delta\mathbf{x} = 0, \quad \mathbf{m} \le \mathbf{x}_k + \mathbf{\Delta x} \le \mathbf{M},$$

$$\mu_k'(\mathbf{m} - \mathbf{x}_k - \Delta\mathbf{x}) = 0 \quad \mu_k \ge \mathbf{0}$$

$$\eta_k'(\mathbf{x}_k + \Delta\mathbf{x} - \mathbf{M}) = 0 \quad \eta_k \ge \mathbf{0}$$

Let's define a new basis $[Y_k, Z_k]$ for Δx where the last n-m components, Z_k are perpendicular to the gradient of the equality constraints h and Y_k is chosen to make $[Y_k, Z_k]$ non-singular :

$$\nabla_x \mathbf{h}(x_k) Z_k = 0$$

$$\Delta\mathbf{x} = Y_k \Delta\mathbf{x}_y + Z_k \Delta\mathbf{x}_z$$

$$Y_k (n \text{ x } m), \quad Z_k (n \text{ x } (n-m))$$

n size of x
m size of h

rSQP

If they were no inequality constraints, then μ, η =0 and the KKT would reduce to:

$$\begin{bmatrix} \mathbf{H}_k' & \nabla_x \mathbf{h}(\mathbf{x}_k)' \\ \nabla_x \mathbf{h}(\mathbf{x}_k) & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \boldsymbol{\lambda}_k \end{bmatrix} = - \begin{bmatrix} \nabla_x J(\mathbf{x}_k)' \\ \mathbf{h}(\mathbf{x}_k) \end{bmatrix}$$

And, in the new basis:

$$\begin{bmatrix} [Y_k, Z_k]' & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{H}_k' & \nabla_x \mathbf{h}(\mathbf{x}_k)' \\ \nabla_x \mathbf{h}(x_k) & 0 \end{bmatrix} \begin{bmatrix} [Y_k, Z_k] & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_y \\ \Delta \mathbf{x}_z \\ \boldsymbol{\lambda}_k \end{bmatrix} = - \begin{bmatrix} [Y_k, Z_k]' & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \nabla_x J(\mathbf{x}_k)' \\ \mathbf{h}(\mathbf{x}_k) \end{bmatrix}$$

$$\begin{bmatrix} Y_k' \mathbf{H}_k' Y_k & Y_k' \mathbf{H}_k' Z_k & Y_k' \nabla_x \mathbf{h}(\mathbf{x}_k)' \\ Z_k' \mathbf{H}_k' Y_k & Z_k' \mathbf{H}_k' Z_k & 0 \\ \nabla_x \mathbf{h}(\mathbf{x}_k) Y_k & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_y \\ \Delta \mathbf{x}_z \\ \boldsymbol{\lambda}_k \end{bmatrix} = - \begin{bmatrix} Y_k' \nabla_x J(\mathbf{x}_k)' \\ Z_k' \nabla_x J(\mathbf{x}_k)' \\ \mathbf{h}(\mathbf{x}_k) \end{bmatrix}$$

rSQP

$$\begin{bmatrix} \cancel{Y_k' \mathbf{H}_k' Y_k} & \cancel{Y_k' \mathbf{H}_k' Z_k} & Y_k' \nabla_x \mathbf{h}(\mathbf{x}_k)' \\ Z_k' \mathbf{H}_k' Y_k & Z_k' \mathbf{H}_k' Z_k & 0 \\ \nabla_x \mathbf{h}(\mathbf{x}_k) Y_k & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_y \\ \Delta \mathbf{x}_z \\ \boldsymbol{\lambda}_k \end{bmatrix} = -\begin{bmatrix} Y_k' \nabla_x J(\mathbf{x}_k)' \\ Z_k' \nabla_x J(\mathbf{x}_k)' \\ \mathbf{h}(\mathbf{x}_k) \end{bmatrix}$$

$$\nabla_x \mathbf{h}(x_k) Y_k \Delta \mathbf{x}_Y = -\mathbf{h}(\mathbf{x}_k)$$

Square, allows computing Δx_Y. This term brings x_k to the linearized constraint h

$$Z_k' \mathbf{H}_k' Z_k \Delta \mathbf{x}_z = -Z_k' \mathbf{H}_k' Y_k \Delta \mathbf{x}_Y - Z_k' \nabla_x J(\mathbf{x}_k)'$$

Will allow computing Δx_z
If $Z_k'H_k'Z_k$ is not PD, add βI terms in the diagonal

Then, $\quad \Delta \mathbf{x} = Y_k \Delta \mathbf{x}_y + Z_k \Delta \mathbf{x}_z$

λ_k can be computed from first row with full expression or approximating the terms $Y_k'H_k'Y_k$ and $Y_k'H_k'Z_k$ by zero::

$$Y_k' \nabla_x \mathbf{h}(x_k)' \boldsymbol{\lambda}_k = -Y_k' \nabla_x J(\mathbf{x}_k)'$$