GAUSS-NEWTON and OTHER METHODS FOR ALGEBRAIC MODELS[1]

Given the MODEL $\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{k})$ and a set of data, $[\hat{\mathbf{y}}_i, \mathbf{x}_i]$, i=1,…,N, we aim to match these data with the values calculated by the model. in some optimal fashion

$\mathbf{k}=[k_1,k_2,\ldots,k_p]^T$ is a *p-dimensional* vector of parameters

$\mathbf{x}=[x_1,x_2,\ldots,x_n]^T$ is an *n-dimensional* vector of independent variables (which are often set by the experimentalist and their numerical values are either known precisely or have been measured)

f is a *m-dimensional* vector function of known form (the algebraic equations)

$\mathbf{y}=[y_1,y_2,\ldots,y_m]^T$ is the *m-dimensional* vector of depended variables which are measured experimentally (output vector).

Objective function: $$S(\mathbf{k})=\sum_{i=1}^{N}[\hat{\mathbf{y}}_i-\mathbf{f}(\mathbf{x}_i,\mathbf{k})]^T\mathbf{Q}_i[\hat{\mathbf{y}}_i-\mathbf{f}(\mathbf{x}_i,\mathbf{k})] \qquad (4.2b)$$

If we recall the definition of the residuals[2]

$$(\textit{residual}\ \mathbf{e}_i)\quad \mathbf{e}_i=[\hat{\mathbf{y}}_i-\mathbf{f}(\mathbf{x}_i,\mathbf{k})] \qquad (2.14)$$

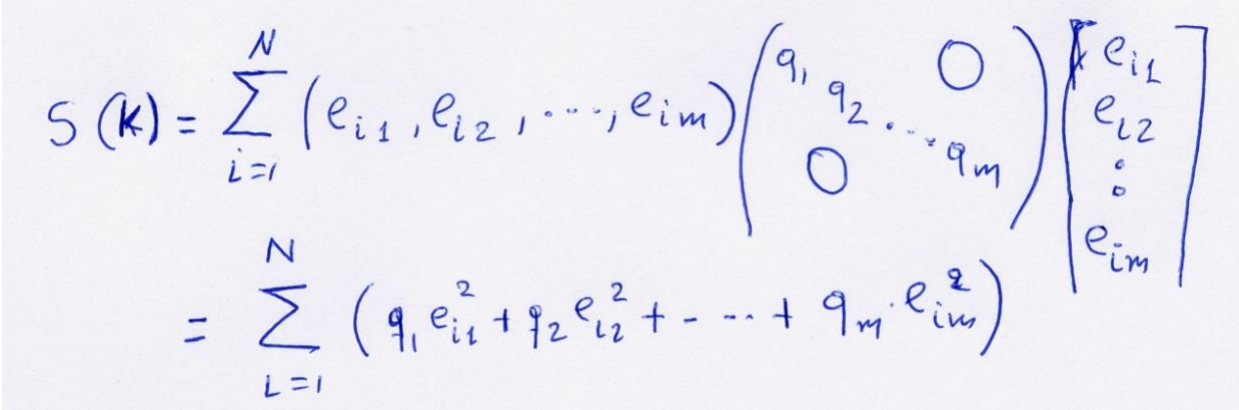
then $$S(\mathbf{k})=\sum_{i=1}^{N}\left(\sum_{k=1}^{m}\sum_{l=1}^{m}e_k Q_{kl} e_l\right)_i \qquad (4.2c)$$

or

---

[1] Englezos, P. and N. Kalogerakis, "*Applied Parameter Estimation for Chemical Engineers*", Marcel-Dekker, New York, 2001

[2] NOTE: the model based value $\mathbf{f}(\mathbf{x}_i,\mathbf{k})$ is calculated using the estimated parameter values.



Minimization of S(k) can be accomplished by using almost any technique available from optimization theory[3]. Next we shall present the **Gauss-Newton** method as we have found it to be overall the best one (Bard, 1970; 1974)[4].

THE GAUSS-NEWTON METHOD

Assume an estimate $\mathbf{k}^{(j)}$ is available at the j^{th} iteration. Linearization of the model equations around $\mathbf{k}^{(j)}$ yields,

---

[3] Scales, L.E., *Introduction to Non-linear Optimization*, Springer-Verlag, New York, NY, 1985.

[3] Gill, P.E., W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, UK, 1981.

[3] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, 1988.

[3] Englezos, P., and N.E. Kalogerakis, Applied Parameter Estimation for Chemical Engineers, Marcel-Dekker, 2001

[4] Bard, Y., "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems", *SIAM J. Numer. Anal.*, 7, 157-186 (1970); Bard, Y., *Nonlinear Parameter Estimation*, Academic Press, New York, NY, 1974.

$$\mathbf{f}(\mathbf{x}_i, \mathbf{k}^{(j+1)}) = \mathbf{f}(\mathbf{x}_i, \mathbf{k}^{(j)}) + \left(\frac{\partial \mathbf{f}^T}{\partial \mathbf{k}}\right)_i^T \Delta\mathbf{k}^{(j+1)} + \text{H.O.T.} \quad ; \; i=1,\ldots,N \qquad (4.3)$$

Neglecting all H.O.T., the model output at $\mathbf{k}^{(j+1)}$ can be approximated by

$$\mathbf{y}(\mathbf{x}_i,\mathbf{k}^{(j+1)}) = \mathbf{y}(\mathbf{x}_i,\mathbf{k}^{(j)}) + \mathbf{G}_i\Delta\mathbf{k}^{(j+1)} \quad ; \quad i=1,\ldots,N \qquad (4.4)$$

where $\mathbf{G}_i$ is the *(m×p)*-sensitivity matrix $\left(\partial \mathbf{f}^T / \partial \mathbf{k}\right)_i^T \equiv \left(\nabla \mathbf{f}^T\right)_i^T$. It is evaluated at $\mathbf{x}_i$ and $\mathbf{k}^{(j)}$.

Note that $\mathbf{G}$ is also the Jacobean matrix of the vector function $\mathbf{f}(\mathbf{x},\mathbf{k})$.

Substitution of $\mathbf{y}(\mathbf{x}_i,\mathbf{k}^{(j+1)})$ from Equation 4.4, into the LS objective function and taking

$$\frac{\partial S(\mathbf{k}^{(j+1)})}{\partial \mathbf{k}^{(j+1)}} = \mathbf{0} \qquad (4.5)$$

yields a linear equation of the form

$$\mathbf{A}\,\Delta\mathbf{k}^{(j+1)} = \mathbf{b} \qquad (4.6)$$

where

$$\mathbf{A} = \sum_{i=1}^{N} \mathbf{G}_i^T \mathbf{Q}_i \mathbf{G}_i \qquad (4.7)$$

and

$$\mathbf{b} = \sum_{i=1}^{N} \mathbf{G}_i^T \mathbf{Q}_i [\ \hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k}^{(j)})] \qquad (4.8)$$

Solution of the above equation yields $\Delta\mathbf{k}^{(j+1)}$. Then $\mathbf{k}^{(j+1)}$, is obtained as

$$\mathbf{k}^{(j+1)} = \mathbf{k}^{(j)} + \mu\Delta\mathbf{k}^{(j+1)} \qquad (4.9$$

where a *stepping parameter*, μ ($0<\mu\leq 1$), is introduced to avoid overstepping.

HOW TO CHOOSE μ: Use the *bisection rule* to obtain μ: start with μ=1 and keep on halving μ until the objective function becomes less than that obtained in the previous iteration

$$S(\mathbf{k}^{(j)} + \mu\Delta\mathbf{k}^{(j+1)}) < S(\mathbf{k}^{(j)}) \qquad (4.10)$$

Algorithm - Implementation Steps:

1. Input the initial guess for the parameters, $\mathbf{k}^{(0)}$ and NSIG
2. For j=0,1, 2,…, repeat the following
3. Compute $\mathbf{y}(\mathbf{x}_i,\mathbf{k}^{(j)})$ and $\mathbf{G}_i$ for each i=1,…,N, and set up matrix **A** & vector **b**
4. Solve the linear equation $\mathbf{A}\Delta\mathbf{k}^{(j+1)}=\mathbf{b}$ and obtain $\Delta\mathbf{k}^{(j+1)}$.
5. Determine μ using the bisection rule and obtain $\mathbf{k}^{(j+1)}=\mathbf{k}^{(j)}+\mu\Delta\mathbf{k}^{(j+1)}$
6. Continue until the maximum number of iterations is reached or convergence is achieved (i.e., $\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta k_i^{(j+1)}}{k_i^{(j)}}\right| \le 10^{-NSIG}$).
7. Compute statistical properties of parameter estimates

Convergence Criteria ("WHEN TO STOP")

A suitable test for convergence is $$\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta k_i^{(j+1)}}{k_i^{(j)}}\right| \le 10^{-NSIG} \quad (4.11)$$

where *p* is the number of parameters and NSIG is the number of significant digits desired in the parameter estimates.

SOLUTION OF $A \cdot \Delta k = b$

$$A = U \cdot S \cdot V^T \Rightarrow A^{-1} = V \cdot S^{-1} U^T$$

$$\boxed{\Delta k = V \cdot S^{-1} \cdot U^T \cdot b}$$

see Singular Value Decomposition

where $S = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$

↑ eigenvalues of A

note: A is a $(p \times p)$ matrix

note

$$\text{Condition Number} = \frac{\lambda_1}{\lambda_p}$$

Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$

Example. ***Consecutive Chemical Reactions .*** Consider a batch reactor where the following reactions take place (Smith, 1981)

$$A \xrightarrow{k_1} B \xrightarrow{k_2} D \qquad (4.12)$$

Taking into account the concentration invariant $C_A+C_B+C_D= C_{A0}$, i.e. that there is no change in the total number of moles, the integrated forms of the isothermal rate equations are

$$C_A(t) = C_{A0}e^{-k_1 t} \qquad (4.13a)$$

$$C_B(t) = C_{A0}k_1\left(\frac{e^{-k_1 t}}{k_2 - k_1} + \frac{e^{-k_2 t}}{k_2 - k_1}\right) \qquad (4.13b)$$

$$C_D(t) = C_{A0} - C_A(t) - C_B(t) \qquad (4.13c)$$

where C_A, C_B and C_D are the concentrations of A, B and D respectively, t is the reaction time, and k_1, k_2 are the unknown rate constants. The concentrations of A and B are measured as a function of time and the data are of the form [t_i, $\hat{C}_{Ai}$, $\hat{C}_{Bi}$], i=1,…,N. Using our standard notation,

Parameter vector:	$\mathbf{k} = [k_1, k_2]^T$	
Vector of independent variables:	$\mathbf{x} = [x_1]$	where $x_1 = t$
Output vector (dependent variables):	$\mathbf{y} = [y_1, y_2]^T$	where $y_1= C_A$, $y_2= C_B$
Model equations:	$\mathbf{f} = [f_1, f_2]^T$	

where

$$f_1(x_1,k_1,k_2) = C_{A0}e^{-k_1x_1} \tag{4.14a}$$

$$f_2(x_1,k_1,k_2) = C_{A0}k_1\left(\frac{e^{-k_1x_1}}{k_2-k_1}+\frac{e^{-k_2x_1}}{k_2-k_1}\right) \tag{4.14b}$$

We have two equations (m=2) and two parameters (p=2).

Thus, the matrix G is a (m × p) or (*2×2*) matrix.

The elements of the *(2×2)-sensitivity coefficient matrix* **G** are

$$G_{11} = \left(\frac{\partial f_1}{\partial k_1}\right) = -x_1f_1 \tag{4.15a}$$

$$G_{12} = \left(\frac{\partial f_1}{\partial k_2}\right) = 0 \tag{4.15b}$$

$$G_{21} = \left(\frac{\partial f_2}{\partial k_1}\right) = \frac{f_1\left(k_2\left(1+e^{(k_2-k_1)x_1}\right)+k_1x_1(k_1-k_2)\right)}{(k_2-k_1)^2} \tag{4.15c}$$

$$G_{22} = \left(\frac{\partial f_2}{\partial k_2}\right) = -\frac{k_1f_1\left(1+(1+(k_2-k_1)x_1)e^{(k_2-k_1)x_1}\right)}{(k_2-k_1)^2} \tag{4.15d}$$

OTHER METHODS FOR ALGEBRAIC MODELS[7]

Two categories: *gradient methods* (require derivatives of the objective functions) and *direct search methods* (derivative-free).

It is assumed that the objective function has continuous second derivatives, whether or not these are explicitly available. *Gradient methods* are still efficient if there are some discontinuities in the derivatives. *Direct search techniques* use function values and are more efficient for highly discontinuous functions.

5.1 GRADIENT MINIMIZATION METHODS

The basic problem is: *Minimize* $S(\mathbf{k})=\sum_{i=1}^{N}\mathbf{e}_i^T\mathbf{Q}_i\mathbf{e}_i$ (5.1)

where

$\mathbf{k}=[k_1, k_2,\ldots,k_p]^T$ the *p-dimensional* vector of parameters

$\mathbf{e}=[e_1, e_2,\ldots,e_m]^T$ the *m-dimensional* vector of residuals $\mathbf{e}_i=\left[\hat{\mathbf{y}}_i-\mathbf{f}(\mathbf{x}_i,\mathbf{k})\right]$ and $\mathbf{Q}_i$ is a user specified positive definite weighting matrix.

We need an *iterative scheme* because it is not possible to find the exact solutions of the equation that gives the stationary points of $S(\mathbf{k})$

$$\nabla S(\mathbf{k})\equiv\frac{\partial S(\mathbf{k})}{\partial \mathbf{k}}=\mathbf{0} \qquad (5.2a)$$

where the operator $\nabla\equiv\left(\frac{\partial}{\partial k_1},\frac{\partial}{\partial k_2},\ldots,\frac{\partial}{\partial k_p}\right)^T$ is applied to the scalar function $S(\mathbf{k})$ yielding the column vector, $\nabla S(\mathbf{k})$, known as the *gradient vector* $\mathbf{g}(\mathbf{k})$. It contains the first partial derivatives of the objective function $S(\mathbf{k})$ with respect to $\mathbf{k}$ and

$$\nabla S(\mathbf{k})\equiv\begin{bmatrix}\frac{\partial S(\mathbf{k})}{\partial k_1}\\ \frac{\partial S(\mathbf{k})}{\partial k_2}\\ \vdots\\ \frac{\partial S(\mathbf{k})}{\partial k_p}\end{bmatrix} \qquad (5.2b)$$

[7] Englezos, P., CHBE 552-*Optimization Methods*, University of British Columbia, 2005

We start from an initial guess for the parameters, $\mathbf{k}^{(0)}=[\,k_1^{(0)}, k_2^{(0)},..., k_p^{(0)}\,]^T$. Some heuristic rules can be used in order to obtain an initial guess (chapter 8).

At the start of the j^{th} iteration we denote by $\mathbf{k}^{(j)}$ the current estimate of the parameters. The j^{th} iteration consists of the computation of a search vector $\Delta\mathbf{k}^{(j+1)}$ from which we obtain the new estimate $\mathbf{k}^{(j+1)}$ according to the following equation

$$\mathbf{k}^{(j+1)} = \mathbf{k}^{(j)} + \mu^{(j)}\Delta\mathbf{k}^{(j+1)} \tag{5.3}$$

where $\mu^{(j)}$ is the *stepping parameter* also known as *dumping* or *relaxation factor*.

Based on the method to calculate the search vector, $\Delta k^{(i+1)}$, different solution methods to the minimization problem arise.

5.1.1 Steepest Descent Method

Search vector: $$\Delta\mathbf{k}^{(j+1)} = -\nabla S(\mathbf{k}^{(j)}) \equiv -\mathbf{g}(\mathbf{k}^{(j)}) \tag{5.6a}$$

Based on Equation 5.1 the search vector is related to the residuals $\mathbf{e}_i = [\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i,\mathbf{k})]$ as follows

$$\Delta\mathbf{k}^{(j+1)} = -2\sum_{i=1}^{N}\left(\nabla\mathbf{e}_i^T\right)\mathbf{Q}_i\,\mathbf{e}_i = 2\sum_{i=1}^{N}\left(\nabla\mathbf{f}_i^T\right)\mathbf{Q}_i\,[\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i,\mathbf{k})] \tag{5.6b}$$

where

$$\nabla\mathbf{e}_i^T = \begin{bmatrix} \frac{\partial}{\partial k_1} \\ \frac{\partial}{\partial k_2} \\ . \\ . \\ \frac{\partial}{\partial k_p} \end{bmatrix}\begin{bmatrix} e_1 & e_2 & . & e_m \end{bmatrix}_i = \begin{bmatrix} \frac{\partial e_1}{\partial k_1} & \frac{\partial e_2}{\partial k_1} & . & . & \frac{\partial e_m}{\partial k_1} \\ \frac{\partial e_1}{\partial k_2} & \frac{\partial e_2}{\partial k_2} & . & . & \frac{\partial e_m}{\partial k_2} \\ . & . & . & . & . \\ . & . & . & . & . \\ \frac{\partial e_1}{\partial k_p} & \frac{\partial e_2}{\partial k_p} & . & . & \frac{\partial e_m}{\partial k_p} \end{bmatrix}_i \tag{5.7}$$

and

$$\nabla \mathbf{f}_i^T = \begin{bmatrix} \frac{\partial}{\partial k_1} \\ \frac{\partial}{\partial k_2} \\ . \\ . \\ \frac{\partial}{\partial k_p} \end{bmatrix} \begin{bmatrix} f_1 & f_2 & . & . & f_m \end{bmatrix}_i = \begin{bmatrix} \frac{\partial f_1}{\partial k_1} & \frac{\partial f_2}{\partial k_1} & . & . & \frac{\partial f_m}{\partial k_1} \\ \frac{\partial f_1}{\partial k_2} & \frac{\partial f_2}{\partial k_2} & . & . & \frac{\partial f_m}{\partial k_2} \\ . & . & . & . & . \\ . & . & . & . & . \\ \frac{\partial f_1}{\partial k_p} & \frac{\partial f_2}{\partial k_p} & . & . & \frac{\partial f_m}{\partial k_p} \end{bmatrix}_i \qquad (5.8)$$

As seen from the above equations

- the *(m×p)* matrix $(\nabla \mathbf{e}^T)^T$ is the Jacobean matrix, **J**, of the vector function **e**
- the *(m×p)* matrix$(\nabla \mathbf{f}^T)^T$ is the Jacobean matrix, **G**, of the vector function **f**(**x**,**k**).

- The srth element of the Jacobean matrix **J**, is given by

$$J_{sr} = \left(\frac{\partial e_s}{\partial k_r} \right) \quad ; \qquad s=1,2,\ldots,m \,, \quad r=1,2,\ldots,p \qquad (5.9a)$$

Similarly, the srth element of the Jacobean matrix **G**, is given by

$$G_{sr} = \left(\frac{\partial f_s}{\partial k_r} \right) \quad ; \quad s=1,2,\ldots,m \,, \quad r=1,2,\ldots,p. \qquad (5.9b)$$

The Steepest Descent method is of interest only for historical and theoretical reasons because its convergence is very slow and oscillations in the **k**-space can easily occur.

<u>It is not a viable method for the general purpose minimization of nonlinear functions</u>.

Algorithm - Implementation Steps

8. Input the initial guess for the parameters, $\mathbf{k}^{(0)}$ and NSIG
9. Specify weighting matrix $\mathbf{Q}_i$ for i=1,2,…N.
10. For j=0,1, 2,…, repeat
11. Compute $\Delta\mathbf{k}^{(j+1)}$ using Equation 5.6
12. Determine μ using the bisection rule and obtain $\mathbf{k}^{(j+1)}=\mathbf{k}^{(j)}+\mu^{(j)}\Delta\mathbf{k}^{(j+1)}$
13. Continue until the maximum number of iterations is reached or convergence is achieved i.e., $\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta k_i^{(j+1)}}{k_i^{(j)}}\right| \le 10^{-NSIG}$

5.1.2 Newton's Method

The step-size parameter $\mu^{(j)}$ is taken equal to 1 and the search vector is

$$\Delta\mathbf{k}^{(j+1)} = -\left[\nabla^2 S(\mathbf{k})\right]^{-1}\nabla S(\mathbf{k}^{(j+1)}) \qquad (5.10)$$

where $\nabla^2 S(\mathbf{k})$ is the Hessian matrix of $S(\mathbf{k})$ evaluated at $\mathbf{k}^{(j)}$ denoted as $\mathbf{H}(\mathbf{k})$. Hence,

$$\Delta\mathbf{k}^{(j+1)} = -\left[\mathbf{H}(\mathbf{k}^{(j)})\right]^{-1}\mathbf{g}(\mathbf{k}^{(j)}) \qquad (5.11)$$

There is no need to obtain the inverse of the Hessian matrix because it is better to solve the following linear system of equations

$$\left[\nabla^2 S(\mathbf{k})\right]\Delta\mathbf{k}^{(j+1)} = -\nabla S(\mathbf{k}) \qquad (5.12a)$$

or equivalently

$$\mathbf{H}(\mathbf{k}^{(j+1)})\,\Delta\mathbf{k}^{(j+1)} = -\,\mathbf{g}(\mathbf{k}^{(j)}) \qquad (5.12b)$$

As seen the steepest-descent method arises from Newton's method if we assume that the *Hessian* matrix of S(**k**) is approximated by the *identity* matrix.

Newton's method:

(i) It is the most rapidly convergent method when the Hessian matrix of S(**k**) is available.

(ii) There is no guarantee that it will converge to a minimum from an arbitrary starting point.

(iii) Problems arise when the Hessian matrix is indefinite or singular.

(iv) The method requires analytical first and second order derivatives which may not be practical to obtain. In that case, finite difference techniques may be employed.

(v) Newton's method is not a satisfactory general-purpose algorithm for function minimization, even when a stepping parameter μ is introduced. Fortunately, it can be modified to provide extremely reliable algorithms with the same asymptotic rate of convergence. *The ratio of the largest to the smallest eigenvalue of the Hessian matrix at the minimum is defined as the* condition number. *The larger the condition number, the more difficult it is for the minimization to converge.*

Solution of the linear Equations 5.12: One approach is the Cholesky factorization of **H** as follows[8] : $\mathbf{H} = \mathbf{L}\,\mathbf{D}\,\mathbf{L}^T$. Matrix **D** is a diagonal one and **L** is a lower triagonal matrix with diagonal elements of unity. We prefer to perform an *eigenvalue decomposition* (discussed later).

$$\text{Also} \quad \mathbf{g}(\mathbf{k}) \equiv \nabla S(\mathbf{k}) = 2\sum_{i=1}^{N} \left(\nabla \mathbf{e}_i^T\right) \mathbf{Q}_i\, \mathbf{e}_i \qquad (5.14)$$

where $\mathbf{e}_i = [\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k})]$ and $\nabla \mathbf{e}_i^T = \left(\mathbf{J}_e^T\right)_i$ is the transpose of the Jacobean matrix of the vector function **e** .

The sr[th] element of this Jacobean was defined by Equation 5.9a. Equation 5.14 can now be written in an expanded form as follows

[8] Gill and Marruy, 1974

$$\nabla S(\mathbf{k}) = 2\sum_{i=1}^{N} \begin{bmatrix} \frac{\partial e_1}{\partial k_1} & \frac{\partial e_2}{\partial k_1} & \cdot & \cdot & \frac{\partial e_m}{\partial k_1} \\ \frac{\partial e_1}{\partial k_2} & \frac{\partial e_2}{\partial k_2} & \cdot & \cdot & \frac{\partial e_m}{\partial k_2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{\partial e_1}{\partial k_p} & \frac{\partial e_2}{\partial k_p} & \cdot & \cdot & \frac{\partial e_m}{\partial k_p} \end{bmatrix}_i \begin{bmatrix} Q_{11} & Q_{12} & \cdot & \cdot & Q_{1m} \\ Q_{21} & Q_{22} & \cdot & \cdot & Q_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ Q_{m1} & Q_{m2} & \cdot & \cdot & Q_{mm} \end{bmatrix}_i \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ e_m \end{bmatrix}_i \tag{5.15}$$

After completing the matrix multiplication operations, we obtain

$$\mathbf{g}(\mathbf{k}) \equiv \frac{\partial S(\mathbf{k})}{\partial \mathbf{k}} = \nabla S(\mathbf{k}) = 2\sum_{i=1}^{N} \begin{bmatrix} \sum_{l=1}^{m}\sum_{r=1}^{m} \frac{\partial e_l}{\partial k_1} Q_{lr} e_r \\ \sum_{l=1}^{m}\sum_{r=1}^{m} \frac{\partial e_l}{\partial k_2} Q_{lr} e_r \\ \cdot \\ \cdot \\ \sum_{l=1}^{m}\sum_{r=1}^{m} \frac{\partial e_l}{\partial k_p} Q_{lr} e_r \end{bmatrix}_i \tag{5.16}$$

Thus, the s^{th} element of the gradient vector $\mathbf{g}(\mathbf{k})$ of the objective function $S(\mathbf{k})$ is given by the following equation

$$g_s = 2\sum_{i=1}^{N}\sum_{l=1}^{m}\sum_{r=1}^{m} \frac{\partial e_l}{\partial k_s} Q_{lr} e_r \quad ; \quad s = 1,2,...,p \tag{5.17}$$

We are now able to obtain the Hessian matrix of the objective function $S(\mathbf{k})$ which is denoted by $\mathbf{H}$ and is given by the following equation

$$\nabla^2 S(\mathbf{k}) \equiv \nabla \nabla^T S(\mathbf{k}) = \begin{bmatrix} \frac{\partial}{\partial k_1} \\ \frac{\partial}{\partial k_2} \\ \cdot \\ \cdot \\ \frac{\partial}{\partial k_p} \end{bmatrix} \begin{bmatrix} \frac{\partial S(\mathbf{k})}{\partial k_1} & \frac{\partial S(\mathbf{k})}{\partial k_2} & \cdot & \cdot & \frac{\partial S(\mathbf{k})}{\partial k_p} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial^2 S(\mathbf{k})}{\partial k_1^2} & \frac{\partial^2 S(\mathbf{k})}{\partial k_1 \partial k_2} & \cdots & \frac{\partial^2 S(\mathbf{k})}{\partial k_1 \partial k_p} \\ \frac{\partial^2 S(\mathbf{k})}{\partial k_2 \partial k_1} & \frac{\partial^2 S(\mathbf{k})}{\partial k_2^2} & \cdots & \frac{\partial^2 S(\mathbf{k})}{\partial k_2 \partial k_p} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 S(\mathbf{k})}{\partial k_p \partial k_1} & \frac{\partial^2 S(\mathbf{k})}{\partial k_p \partial k_2} & \cdots & \frac{\partial^2 S(\mathbf{k})}{\partial k_p^2} \end{bmatrix} \equiv \begin{bmatrix} \frac{\partial g_1}{\partial k_1} & \frac{\partial g_2}{\partial k_1} & \cdots & \frac{\partial g_m}{\partial k_1} \\ \frac{\partial g_1}{\partial k_2} & \frac{\partial g_2}{\partial k_2} & \cdots & \frac{\partial g_m}{\partial k_2} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_1}{\partial k_p} & \frac{\partial g_2}{\partial k_p} & \cdots & \frac{\partial g_m}{\partial k_p} \end{bmatrix} \qquad (5.18)$$

where we use the notation $g_s \equiv \frac{\partial S(\mathbf{k})}{\partial k_s}$.

Thus, the $\xi\mu^{th}$ element of the Hessian matrix is defined by

$$G_{\xi\mu} = \frac{\partial g_\xi}{\partial k_\mu} \quad \xi = 1,2,\ldots,p \quad \text{and} \quad \mu = 1,2,\ldots,p \qquad (5.19)$$

This element can be calculated by taking into account Equation 5.16 as follows

$$G_{\xi\mu} = 2\sum_{i=1}^{N}\sum_{l=1}^{m}\sum_{r=1}^{m}\left(\frac{\partial e_l}{\partial k_\xi} Q_{lr} \frac{\partial e_r}{\partial k_\mu}\right)_i + 2\sum_{i=1}^{N}\sum_{l=1}^{m}\sum_{r=1}^{m}\left(\frac{\partial^2 e_l}{\partial k_\xi \partial k_\mu} Q_{lr} e_r\right)_i \qquad (5.20)$$

where ξ=1,2,…,p and μ=1,2,…,p.

The *Gauss-Newton method* arises when the second order terms on the right-hand side of Equation 5.20 are ignored. Leaving out the second derivative containing terms may be justified by the fact that these terms contain the residuals e_r as factors. These residuals are expected to be small quantities.

Algorithm - Implementation Steps

1. Input the initial guess for the parameters, $\mathbf{k}^{(0)}$ and NSIG
2. Specify weighting matrix $\mathbf{Q}_i$ for I=1,2,…N.
3. For j=0,1, 2,…, repeat.
4. Compute $\Delta\mathbf{k}^{(j+1)}$ by solving Equation 5.12b.
5. Determine μ using the bisection rule and obtain $\mathbf{k}^{(j+1)} = \mathbf{k}^{(j)} + \mu\Delta\mathbf{k}^{(j+1)}$.
6. Continue until the maximum number of iterations is reached or convergence is achieved (i.e., $\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta k_i^{(j+1)}}{k_i^{(j)}}\right| \leq 10^{-NSIG}$

5.1.3 Modified Newton's Methods

- Modified Newton methods attempt to alleviate the deficiencies of Newton's method. The problem arises if the Hessian matrix, **G**, is not positive definite.
- If any of the eigenvalues are not positive then a procedure proposed by Marquardt (1963) based on earlier work by Levenberg (1944) should be followed[9][10]. A positive value γ can be added to all the eigenvalues such that the resulting positive quantities, $\lambda_i+\gamma$, i=1,2,…,m are the eigenvalues of a positive matrix $\mathbf{H}_{LM}$, given by

$$\mathbf{H}_{LM} = \mathbf{G} + \gamma \mathbf{I} \qquad (5.21)$$

where **I** is the identity matrix.

Algorithm – Implementation Steps

1. Input the initial guess for the parameters, $\mathbf{k}^{(0)}$, γ and NSIG.
2. Specify weighting matrix $\mathbf{Q}_i$ for i=1,2,…N.
3. For j=0,1, 2,…, repeat.
4. Compute $\Delta\mathbf{k}^{(j+1)}$ by solving Equation 5.12b but in this case the Hessian matrix $\mathbf{H}(\mathbf{k})$ has been replaced by that given by Equation 5.22.
5. Determine μ using the bisection rule and obtain $\mathbf{k}^{(j+1)} = \mathbf{k}^{(j)} + \mu\Delta\mathbf{k}^{(j+1)}$.
6. Continue until the maximum number of iterations is reached or convergence is achieved (i.e., $\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta k_i^{(j+1)}}{k_i^{(j)}}\right| \le 10^{-NSIG}$).

Modified Newton methods require calculation of second derivatives. If derivatives are not available analytically we calculate them by finite differences (considerable number of gradient evaluations if the number of parameters, p, is large). Finite difference approximations of derivatives are prone to truncation and round-off errors and require a considerable number of gradient evaluations if the number of parameters, p, is large.

[9] Marquardt (1963)

[10] Levenberg (1944)

5.1.4 Conjugate Gradient Methods

Conjugate gradient-type methods form a class of minimization procedures that accomplish two objectives:

(a) There is no need for calculation of second order derivatives;

(b) they have relatively small computer storage requirements.

These methods are suitable for problems with a very large number of parameters. Two versions of the method have been formulated: (a) *Fletcher-Reeves* (FR) version; and (b) *Polak-Ribiere* (PR) version (recommended for its slightly better convergence properties)

5.1.5 Quasi-Newton or Variable Metric or Secant Methods[11]

These methods avoid calculation of the elements of the ($p \times p$) Hessian matrix but they rely on formulas that approximate the Hessian and its inverse.

Two algorithms have been developed:

(a) The Davidon-Fletcher-Powell Formula (DFP)

(b) The Broyden-Fletcher-Goldfard-Shanno Formula (BFGS) .

The BFGS method is considered to be superior to DFP because it is less prone to loss of positive definiteness or to singularity problems through round off errors

[11] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 1988; Scales, L.E., *Introduction to Non-linear Optimization*, Springer-Verlag, New York, NY, 1985;.

5.2 DIRECT SEARCH OR DERIVATIVE FREE METHODS

- They search for the minimum of an objective function without calculating derivatives analytically or numerically (they use only function evaluations).
- They are based upon heuristic rules which make no *a priori* assumptions about the objective function.
- They tend to have much poorer convergence rates than gradient methods and in general are considered not as efficient and robust as the indirect or gradient search methods
- They are robust and reliable for *large scale optimization problems* whose objective functions exhibit many local minima

EXAMPLES OF DIRECT SEARCH METHODS

- The ***simulated annealing*** technique[12] is probably the most popular one. It tries to mimic the physical process of annealing whereby a material starts in a melted state and its temperature is gradually lowered until it reaches its minimum energy state. In the physical system the temperature should not be rapidly lowered because a sub-optimal structure may be formed in the crystallized system and lead to quenching. In practice, simulated annealing is implemented using the Metropolis et al. (1953) algorithm. For parameter estimation purposes, simulated annealing can be implemented by discretizing the parameter space. Alternatively, we can specify minimum and maximum values for each unknown parameter, and by using a random number uniformly distributed in the range [0,1], we can specify randomly the potential parameter values as

$$k_i = k_{min,i} + R[k_{max,i} - k_{min,i}] \quad ; \quad i=1,\ldots,p \qquad (5.23)$$

where R is a random number.

For a good discussion on MC and Metropolis see Scott Shell's notes[13]

[12] Cerny, V., "Thermodynamic Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm", *J. Opt. Theory Applic.*, 45, 41-51 (1985).; Kirkpatrick S., C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by Simulated Annnealing", *Science*, 220, 671-680 (1983).; Metropolis, N., A.W. Rosenbluthh, M.N. Rosenbluth, A.H. Teller and E. Teller, "Equations of State Calculations by Fast Computing Machines", *J. Chem. Physics*, 21, 1087-1092 (1953); Otten R.H.J.M. and L.P.P.P. Van Ginneken, *The annealing algorithm*, Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, NL, 1989.; Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1992.; Ingber, L., Very Fast Simulated Re-annealing, Math. Comput. Modelling, 12(8), 967-973, 1989.

[13] Shell, M. S., Thermodynamics and Statistical Mechanics, Cambridge Univ Press, 2015 p.459-464

- The ***LJ Optimization Procedure*** is one of the most reliable direct search methods[14]. The method is easy to program and handles the problem of multiple optima with high reliability. A important advantage of the method is its ability to handle multiple nonlinear constraints. The adaptation of the original LJ optimization procedure to parameter estimation problems for algebraic equation models is given next.

(i) Choose an initial guess for the p-dimensional unknown parameter vector, $\mathbf{k}^{(0)}$; the *region contraction coefficient*, δ (typically δ=0.95 is used); the number of random evaluations of the objective function, N_R (typically N_R=100 is used) within an iteration; the maximum number of iterations, j_{max} (typically j_{max}=200 is used) and an initial search region, $\mathbf{r}^{(0)}$ (a typical choice is $\mathbf{r}^{(0)} = \mathbf{k}_{max} - \mathbf{k}_{min}$).

(ii) Set the iteration index j=1 and $\mathbf{k}^{(j-1)} = \mathbf{k}^{(0)}$ and $\mathbf{r}^{(j-1)} = \mathbf{r}^{(0)}$.

(iii) Generate or read from a file, $N_R \times p$ random numbers (R_{ni}) uniformly distributed in [-0.5, 0.5]

(iv) For n=1,2,…,N_R, generate the corresponding random trial parameter vectors from

$$\mathbf{k}_n = \mathbf{k}^{(j-1)} + \boldsymbol{R}_n \mathbf{r}^{(j-1)} \qquad (5.24)$$

where $\boldsymbol{R}_n = diag(R_{n1}, R_{n2}, \ldots, R_{np})$.

(v) Find the parameter vector among the N_R trial ones that minimizes the LS Objective function

[14] Luus, R., "Determination of the Region Sizes for LJ Optimization", *Hung. J. Ind. Chem.*, 26, 281-286 (1998).; Luus, R., and T.H.I. Jaakola, "Optimization by Direct Search and Systematic Reduction of the Search Region", *AIChE J*, 19, 760 (1973); Wang, B.C. and R. Luus, "Increasing the Size of the Region of Convergence for Parameter Estimation Through the Use of Shorter Data-Length", *Int. J. Control*, 31, 947-957 (1980); Wang, B.C. and R. Luus, "Optimization of Non-Unimodal Systems", *Int. J. Numer. Meth. Eng.*, 11, 1235-1250 (1977); Wang, B.C. and R. Luus, "Reliability of Optimization Procedures for Obtaining the Global Optimum", *AIChE J.*, 19, 619-626 (1978); Luus, R., "Application of Direct Search Optimization for Parameter Estimation", in *Scientific Computing in Chemical Engineering II*, Keil, F., W. Mackens, H. Vob, J.Werther (Eds), Springer-Verlag, Berlin Heidelberg, 1999.

$$S_{LS}(\mathbf{k}) = \sum_{i=1}^{N} \left[\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k})\right]^T \mathbf{Q}_i \left[\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k})\right] \qquad (5.25)$$

(vi) Keep the best trial parameter vector, $\mathbf{k}^*$, up to now and the corresponding minimum value of the objective function, S*.

(vii) Set $\mathbf{k}^{(j)} = \mathbf{k}^*$ and compute the search region for the next iteration as

$$\mathbf{r}^{(j)} = \delta \times \mathbf{r}^{(j-1)} \qquad (5.26)$$

(viii) If $j < j_{max}$, increment j by 1 go to Step (iii); else STOP.

The algorithm can be more *efficient* if we choose a value for N_R which is a function of the number of unknown parameters e.g.

$$N_R = 50 + 10 \times p \qquad (5.27)$$

We may also use a slower contraction of the search region as p increases e.g.

$$\delta = (0.90)^{1/p} \qquad (5.28)$$

Since we have a minimization problem, significant computational savings can be realized by noting in the implementation of the LJ optimization procedure that for each trial parameter vector, we do not need to complete the summation in Equation 5.23. Once the LS Objective function exceeds the smallest value found up to that point (S*), a new trial parameter vector can be selected.