1 Minimization

1.1 Introduction

Minimization (maximization) is the problem of finding the minimum (maximum) of a given—generally non-linear—real valued function $\phi(\mathbf{x})$ of an n-dimensional argument $\mathbf{x} \doteq \{x_1, \ldots, x_n\}$. The function is often called the *objective function* or the *cost function*.

Minimization is a simpler case of a more general poblem—*optimization*—which includes finding the best available values of the objective function within a given domain and/or subject to given constrains.

Minimization is not unrelated to root-finding: at the minimum all partial derivatives of the objective function vanish,

$$\frac{\partial \phi}{\partial x_i} = 0 \Big|_{i=1\ldots n} , \tag{1}$$

and one can alternatively solve this system of (non-linear) equations.

1.2 Local minimization

Local minimization refers to a group of algorithms that move from one candidate solution to another candidate solution by applying local changes and moving "downhill" until a solution deemed optimal is found (or the alotted time is elapsed).

1.2.1 Newton's method

Newton's method is based on the quadratic approximation of the objective function $\phi(\mathbf{x})$ in the vicinity of the suspected minimum,

$$\phi(\mathbf{x} + \Delta\mathbf{x}) \approx \phi(\mathbf{x}) + \nabla\phi(\mathbf{x})^{\mathsf{T}} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^{\mathsf{T}} \mathrm{H}(\mathbf{x}) \Delta\mathbf{x} , \tag{2}$$

where the vector $\nabla\phi(\mathbf{x})$ is the gradient of the objective function at the point $\mathbf{x}$,

$$\nabla\phi(\mathbf{x}) \doteq \left\{ \frac{\partial \phi(\mathbf{x})}{\partial x_i} \right\}_{i=1\ldots n} , \tag{3}$$

and $\mathrm{H}(\mathbf{x})$ is the *Hessian matrix* – a square matrix of second-order partial derivatives of the objective function at the point $\mathbf{x}$,

$$\mathrm{H}(\mathbf{x}) \doteq \left\{ \frac{\partial^2 \phi(\mathbf{x})}{\partial x_i \partial x_j} \right\}_{i,j \in 1\ldots n} . \tag{4}$$

The minimum of the quadratic form (2), as function of $\Delta\mathbf{x}$, is found at the point where its gradient with respect to $\Delta\mathbf{x}$ vanishes,

$$\nabla\phi(\mathbf{x}) + \mathrm{H}(\mathbf{x}) \Delta\mathbf{x} = 0 . \tag{5}$$

This gives an approximate step towards the minimum, called the *Newton's step*,

$$\Delta\mathbf{x} = -\mathrm{H}(\mathbf{x})^{-1}\nabla\phi(\mathbf{x})\,. \tag{6}$$

The original Newton's method is simply the iteration,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathrm{H}(\mathbf{x}_k)^{-1}\nabla\phi(\mathbf{x}_k)\,, \tag{7}$$

where at each iteration the full Newton's step is taken and the Hessian matrix is recalculated. In practice, instead of calculating H^{-1} one rather solves the linear equation (5).

Usually the Newton's method is modified to take a smaller step $\mathbf{s}$,

$$\mathbf{s} = \lambda\Delta\mathbf{x}, \tag{8}$$

with $0 < \lambda < 1$. The factor λ can be found by a backtracking algoritm similar to that in the Newton's method for root-finding. One starts with $\lambda = 1$ and than backtracks, $\lambda \leftarrow \lambda/2$, until the *Armijo condition*,

$$\phi(\mathbf{x}+\mathbf{s}) < \phi(\mathbf{x}) + \alpha\mathbf{s}^{\mathsf{T}}\nabla\phi(\mathbf{x})\,, \tag{9}$$

is satisfied (or the minimal λ (say, 1/1024) is reached, in which case the step is taken unconditionally). The parameter α can be chosen as small as 10^{-4}.

1.2.2 Numerical calculation of gradient and Hessian matrix

The *forward* finite-difference approximation to the derivative $f'(x)$ of a function $f(x)$ is given as

$$f'(x) \approx \frac{f(x+\delta x) - f(x)}{\delta x}\,, \tag{10}$$

where δx is usually chosen as $|x|\sqrt{\epsilon}$ where ϵ is the machine precision (for "double" numbers $\sqrt{\epsilon} = 2^{-26}$). This gives the following approximation for the gradient,

$$\nabla\phi(x)_i = \frac{\partial\phi(x)}{\partial x_i} = \frac{\phi(x_1,\ldots,x_i+\delta x_i,\ldots,x_n) - \phi(x_1,\ldots,x_n)}{\delta x_i}\,, \tag{11}$$

where $\delta x_i = |x_i|\sqrt{\epsilon}$.

The Hessian matrix in the same approximation is given as

$$\frac{\partial^2\phi}{\partial x_i\partial x_j} = \frac{\partial}{\partial x_i}\nabla\phi_j = \frac{\nabla\phi(x_1,\ldots,x_i+\delta x_i,\ldots,x_n)_j - \nabla\phi(x_1,\ldots,x_n)_j}{\delta x_i} \tag{12}$$

The *central* (possibly better) finite difference formula for the first derivative is given as,

$$f'(x) \approx \frac{f(x+\delta x) - f(x-\delta x)}{2\delta x}\,. \tag{13}$$

This gives the following approximation for the gradient,

$$\nabla\phi(x)_i = \frac{\partial\phi(x)}{\partial x_i} = \frac{\phi(x_1,\ldots,x_i+\delta x_i,\ldots,x_n)-\phi(x_1,\ldots,x_i-\delta x_i,\ldots,x_n)}{2\delta x_i}\,, \tag{14}$$

and the following approximation for the Hessian matrix,

$$\mathrm{H}_{jk} = \frac{\partial^2\phi}{\partial x_j\partial x_k} \approx \frac{\phi(\mathbf{x}+\delta\mathbf{x}_k+\delta\mathbf{x}_j)-\phi(\mathbf{x}+\delta\mathbf{x}_k-\delta\mathbf{x}_j)-\phi(\mathbf{x}-\delta\mathbf{x}_k+\delta\mathbf{x}_j)+\phi(\mathbf{x}-\delta\mathbf{x}_k-\delta\mathbf{x}_j)}{4\delta x_k\delta x_j}\,. \tag{15}$$

where $\delta\mathbf{x}_k$ is a vector in the direction k with the length $\delta x_k = |x_k|\sqrt{\epsilon}$ where ϵ is the *machine epsilon*. In practice the calculation of the gradient should reuse the ϕ-values from the calculation of the Hessian matrix.

1.2.3 Quasi-Newton methods

Quasi-Newton methods are variations of the Newton's method which attempt to avoid recalculation of the Hessian matrix at each iteration, trying instead certain updates based on the analysis of the gradient vectors. The update $\delta\mathrm{H}$ is usually chosen to satisfy the condition

$$\nabla\phi(\mathbf{x}+\mathbf{s}) = \nabla\phi(\mathbf{x}) + (\mathrm{H}+\delta\mathrm{H})\mathbf{s}\,, \tag{16}$$

called *secant equation*, which is the Taylor expansion of the gradient.

The secant equation is under-determined in more than one dimension as it consists of only n equations for the n^2 unknown elements of the update $\delta\mathrm{H}$. Various quasi-Newton methods use different choices for the form of the solution of the secant equation.

In practice one typically uses the inverse Hessian matrix (often—but not always—denoted as B) and applies the updates directly to the inverse matrix thus avoiding the need to solve the linear equation (5) at each iteration.

For the inverse Hessian matrix the secant equation (16) reads

$$(\mathrm{B}+\delta\mathrm{B})\mathbf{y} = \mathbf{s}\,, \tag{17}$$

or, in short,

$$\delta\mathrm{B}\mathbf{y} = \mathbf{u}\,, \tag{18}$$

where $\mathrm{B} \doteq \mathrm{H}^{-1}$, $\mathbf{y} \doteq \nabla\phi(\mathbf{x}+\mathbf{s}) - \nabla\phi(\mathbf{x})$, and $\mathbf{u} \doteq \mathbf{s} - \mathrm{B}\mathbf{y}$.

One usually starts with the identity matrix as the zeroth approximation for the inverse Hessian matrix and then applies the updates.

If the minimal λ (say, 1/1024) is reached during the bactracking line-search—which might be a signal of lost precision in the approximate (inverse) Hessian matrix—it is advisable to reset the current inverse Hessian matrix to identity matrix.

Table 1.2.3 lists one possible algorithm of the quasi-newton method with updates.

Table 1: Quasi-newton minimisation algorithm with updates.

```
set the inverse Hessian matrix to unity, B = 1
repeat until converged (e.g. ‖∇φ‖ < tolerance) :
     calculate the Newton's step Δx = −B∇φ
     do linesearch starting with λ = 1 :
         if φ(x + λΔx) < φ(x) accept the step and update B:
             x = x + λΔx
             update B = B + δB
             break linesearch
         λ = λ/2
         if λ < 1/1024 accept the step and reset B:
             x = x + λΔx
             B = 1
             break linesearch
         continue linesearch
```

Broyden's update The Broyden's update is chosen in the form

$$\delta \mathrm{B} = \mathbf{c}\mathbf{s}^{\mathsf{T}} \, . \tag{19}$$

where the vector $\mathbf{c}$ is found from the condition (18),

$$\mathbf{c} = \frac{\mathbf{u}}{\mathbf{s}^{\mathsf{T}}\mathbf{y}} \, . \tag{20}$$

Sometimes the dot-product $\mathbf{s}^{\mathsf{T}}\mathbf{y}$ becomes very small or even zero which results in serious numerical difficulties. One can avoid this by only performing update if the condition $|\mathbf{s}^{\mathsf{T}}\mathbf{y}| > \epsilon$ is satisfied where ϵ is a small number, say 10^{-6}.

Symmetric Broyden's update The Broyden's update (19) is not symmetric (while the Hessian matrix should be) which is an obvious drawback. Therefore a beter approximation might be the symmetric Broyden's update,

$$\delta \mathrm{B} = \mathbf{a}\mathbf{s}^{\mathsf{T}} + \mathbf{s}\mathbf{a}^{\mathsf{T}} \, . \tag{21}$$

The vector $\mathbf{a}$ is again found from the condition (18),

$$\mathbf{a} = \frac{\mathbf{u} - \gamma\mathbf{s}}{\mathbf{s}^{\mathsf{T}}\mathbf{y}} \, , \tag{22}$$

where $\gamma = (\mathbf{u}^{T}\mathbf{y})/(2\mathbf{s}^{\mathsf{T}}\mathbf{y})$.

Again one only performs the update if $|\mathbf{s}^{\mathsf{T}}\mathbf{y}| > \epsilon$.

SR1 update The symmetric-rank-1 update (SR1) in chosen in the form

$$\delta \mathrm{B} = \mathbf{v}\mathbf{v}^{\mathsf{T}} \,, \tag{23}$$

where the vector $\mathbf{v}$ is again found from the condition (16), which gives

$$\delta \mathrm{B} = \frac{\mathbf{u}\mathbf{u}^{T}}{\mathbf{u}^{T}\mathbf{y}} \,. \tag{24}$$

Again, one only performs the update if denominator is not too small, that is, $|\mathbf{u}^{\mathsf{T}}\mathbf{y}| > \epsilon$.

Other popular updates The wikipedia article "Quasi-Newton method" list several other popular updates.

1.2.4 Downhill simplex method

The *downhill simplex method* [1] (also called "Nelder-Mead" or "amoeba") is a commonnly used minimization algorithm where the minimum of a function in an n-dimensional space is found by transforming a simplex—a polytope with n+1 vertexes—according to the function values at the vertexes, moving it downhill until it converges towards the minimum.

The advantages of the downhill simplex method is its stability and the lack of use of derivatives. However, the convergence is realtively slow as compared to Newton's methods.

In order to introduce the algorithm we need the following definitions:

- Simplex: a figure (polytope) represented by n+1 points, called vertexes, $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}\}$ (where each point $\mathbf{p}_k$ is an n-dimensional vector).
- Highest point: the vertex, $\mathbf{p}_{\mathrm{hi}}$, with the highest value of the function: $\phi(\mathbf{p}_{\mathrm{hi}}) = \max_k \phi(\mathbf{p}_k)$.
- Lowest point: the vertex, $\mathbf{p}_{\mathrm{lo}}$, with the lowest value of the function: $\phi(\mathbf{p}_{\mathrm{lo}}) = \min_k \phi(\mathbf{p}_k)$.
- Centroid: the center of gravity of all points, except for the highest: $\mathbf{p}_{\mathrm{ce}} = \frac{1}{n}\sum_{(k\neq \mathrm{hi})} \mathbf{p}_k$

The simplex is moved downhill by a combination of the following elementary operations:

1. Reflection: the highest point is reflected against the centroid, $\mathbf{p}_{\mathrm{hi}} \to \mathbf{p}_{\mathrm{re}} = \mathbf{p}_{\mathrm{ce}} + (\mathbf{p}_{\mathrm{ce}} - \mathbf{p}_{\mathrm{hi}})$.
2. Expansion: the highest point reflects and then doubles its distance from the centroid, $\mathbf{p}_{\mathrm{hi}} \to \mathbf{p}_{\mathrm{ex}} = \mathbf{p}_{\mathrm{ce}} + 2(\mathbf{p}_{\mathrm{ce}} - \mathbf{p}_{\mathrm{hi}})$.

Table 2: Downhill simplex (Nelder-Mead) algorithm

```
REPEAT :
  find highest, lowest, and centroid points of the simplex
  try reflection
  IF φ(reflected) < φ(lowest) :
    try expansion
    IF φ(expanded) < φ(reflected) :
      accept expansion
    ELSE :
      accept reflection
  ELSE :
    IF φ(reflected) < φ(highest) :
      accept reflection
    ELSE :
      try contraction
      IF φ(contracted) < φ(highest) :
        accept contraction
      ELSE :
        do reduction
UNTIL converged (e.g. size(simplex)<tolerance)
```

3. Contraction: the highest point halves its distance from the centroid, $\mathbf{p}_{\rm hi} \rightarrow \mathbf{p}_{\rm co} = \mathbf{p}_{\rm ce} + \frac{1}{2}(\mathbf{p}_{\rm hi} - \mathbf{p}_{\rm ce})$.

4. Reduction: all points, except for the lowest, move towards the lowest points halving the distance. $\mathbf{p}_{k\neq{\rm lo}} \rightarrow \frac{1}{2}(\mathbf{p}_k + \mathbf{p}_{\rm lo})$.

Table 2 shows one possible algorithm for the downhill simplex algorithm.

1.2.5 Gauss-Newton algorithm

The Gauss-Newton algorithm is designed to minimize an objective function $\phi(\mathbf{c})$ that is given as a sum of squares of several (non-linear) functions $r_i(\mathbf{c})$,

$$\phi(\mathbf{c}) = \sum_{i=1}^{n} r_i^2(\mathbf{c}) \ , \tag{25}$$

where $\{c_{k=1\ldots m}\}$ is the set of parameters of the objective function. In particular, the algorithm can be used to solve a non-linear least squares curve fitting problem where the function to minimize is given as

$$\chi^2(\mathbf{c}) = \sum_{i=1}^{n} \left(\frac{f(\mathbf{c}, x_i) - y_i}{\delta y_i} \right)^2 \ , \tag{26}$$

where $\{x_i, y_i \pm \delta y_i\}$ is the set of data to fit and $f(\mathbf{c}, x)$ is the fitting function that depends on a set of parameters $\mathbf{c}$.

The algorithm can also be used to find an approximate solution to an overdetermined (if $n > m$) system of non-linear equations

$$\mathbf{r}(\mathbf{c}) = 0 \; . \tag{27}$$

Just like the Newton's method the algorithm relies on the Taylor expansion of the objective function in the vicinity of the suspected minimum,

$$\phi(\mathbf{c} + \Delta\mathbf{c}) \approx \phi(\mathbf{c}) + \mathbf{g}^{\mathsf{T}}\Delta\mathbf{c} + \frac{1}{2}\Delta\mathbf{c}^{\mathsf{T}}\mathrm{H}\mathbf{c} \; , \tag{28}$$

where the (size-m) gradient $\mathbf{g}$ is given as

$$g_k = 2\sum_{i=1}^{n} r_i \frac{\partial r_i}{\partial c_k} \tag{29}$$

and the (square $m \times m$) Hessian matrix H is given as

$$\mathrm{H}_{jk} = 2\sum_{i=1}^{n}\left(\frac{\partial r_i}{\partial c_j}\frac{\partial r_i}{\partial c_k} + r_i\frac{\partial^2 r_i}{\partial c_j \partial c_k}\right) \; . \tag{30}$$

Now, in the Gauss-Newton method one ignores the second-derivative term in (30) which results in the following approximation for the Hessian matrix,

$$\mathrm{H}_{jk} \approx 2\mathrm{J}^{\mathsf{T}}\mathrm{J} \; , \tag{31}$$

where J is the (tall $n \times m$) Jacobian matrix of the $\{r_i\}$ functions,

$$\mathrm{J}_{ik} = \frac{\partial r_i}{\partial c_k} \; . \tag{32}$$

The approximation (31) may be valid in two cases,

1. The functions r_i are small in the vicinity of the minimum;
2. The functions r_i are only slightly non-linear such that the second derivatives are small in magnitude.

Using the Jacobian matrix the gradient of ϕ can be written as

$$\mathbf{g} = 2\mathrm{J}^{\mathsf{T}}\mathbf{r} \; . \tag{33}$$

From here the algorithm proceeds as in the usual Newton's method: one finds the Newton's step,

$$\Delta\mathbf{c} = -\mathrm{H}^{-1}\mathbf{g} \approx -\left(\mathrm{J}^{\mathsf{T}}\mathrm{J}\right)^{-1}\left(\mathrm{J}^{\mathsf{T}}\mathbf{r}\right) \; , \tag{34}$$

and then does the backtracking line-search.

Note that $\left(\mathrm{J}^{\mathsf{T}}\mathrm{J}\right)^{-1}\mathrm{J}^{\mathsf{T}}$ is the left pseudo-inverse of the matrix J. Therefore the Newton's step of the Gauss-Newton method for the objective function $\sum_{i=1}^{n} r_i^2$ is equivalent to the step of the root-finding Newton's method for the system of equations $\mathbf{r}(\mathbf{c}) = 0$.

1.3 Uncertainties of nonlinear least squares fit parameters

The *non-linear least squares* fit is the process of fitting a curve (a mathematical function, $f(\mathbf{c}, x)$, where $\mathbf{c}$ is the set of fitting parameters) to a set of data points with uncertainties, $\{x_i, y_i \pm \delta y_i\}$. However, unlike the ordinary least squares fit, where the fitting parameters enter linearly, in the non-linear least squares fit the parameters enter essentially non-linearly.

The fit is achieved by minimizing the sum of squares (hence the name) of the deviations of the curve from the data (called χ^2 in physics),

$$\chi^2(\mathbf{c}) = \sum_{i=1}^{n} \left(\frac{f(\mathbf{c}, x_i) - y_i}{\delta y_i} \right)^2 \equiv \sum_{i=1}^{n} r_i^2(\mathbf{c}) \,, \tag{35}$$

where

$$r_i(\mathbf{c}) \doteq \frac{f(\mathbf{c}, x_i) - y_i}{\delta y_i} \tag{36}$$

are the (weighted) residuals.

The χ^2 can be minimized in the space of the fitting parameters either using any of the general minimization algorithms or using the Gauss-Newton algortithm which is specifically designed for an objective function in the form of the sum of squares of some residuals.

The uncertainties of the fitting parameters can be estimated by i) Taylor expansion of χ^2 around the minimum; ii) linearizing the problem; iii) calculating the uncertainties using the same technique as for the ordinary least squares fit. In other words, we apply the Newton's method to find the solution of the minimization problem and then determine the uncertainties of the fitting parameters from the last Newton's step (the one that brings us to the minimum).

1.3.1 Linearization of nonlinear problem at minimum

The Newton's step $\Delta\mathbf{c}$ toward the minimum is found from the second order Taylor expansion of χ^2,

$$\chi^2(\mathbf{c} + \Delta\mathbf{c}) \approx \chi^2(\mathbf{c}) + \mathbf{g}^\mathsf{T}\Delta\mathbf{c} + \frac{1}{2}\Delta\mathbf{c}^\mathsf{T}\mathrm{H}\Delta\mathbf{c} \,, \tag{37}$$

where the gradient $\mathbf{g}$ is given as

$$g_k = \frac{\partial\chi^2}{\partial c_k} = \sum_{i=1}^{n} 2r_i \frac{\partial r_i}{\partial c_k} \,. \tag{38}$$

and the Hessian matrix H is given as

$$\mathrm{H}_{jk} = \frac{\partial^2\chi^2}{\partial c_j \partial c_k} = \sum_{i=1}^{n} \left(2\frac{\partial r_i}{\partial c_j}\frac{\partial r_i}{\partial c_k} + 2r_i \frac{\partial^2 r_i}{\partial c_j \partial c_k} \right) \,. \tag{39}$$

Close to the minimum the term with the second derivative can be (hopefully) neglected (since at the minimum $f(\mathbf{c}, x_i) \approx y_i$).

Introducing the (tall $n \times m$) Jacobian matrix J of the residuals,

$$\mathrm{J}_{ij} = \frac{\partial r_i}{\partial c_j} = \frac{1}{\delta y_i} \frac{\partial f(\mathbf{c}, x_i)}{\partial c_j} \ , \tag{40}$$

one can rewrite the gradient as

$$\mathbf{g} = 2\mathrm{J}^{\mathsf{T}}\mathbf{r} \tag{41}$$

and the Hessian matrix as

$$\mathrm{H} = 2\mathrm{J}^{\mathsf{T}}\mathrm{J} \ . \tag{42}$$

The corresponding Newton's step is determined by the equation

$$\mathrm{H}\Delta\mathbf{c} = -\mathbf{g} \ , \tag{43}$$

or

$$\mathrm{J}^{\mathsf{T}}\mathrm{J}\Delta\mathbf{c} = -\mathrm{J}^{\mathsf{T}}\mathbf{r} \ , \tag{44}$$

which gives the Newton's step to the minimum as

$$\Delta\mathbf{c} = -\mathrm{J}^{-1}\mathbf{r} \tag{45}$$

where

$$\mathrm{J}^{-1} = (\mathrm{J}^{\mathsf{T}}\mathrm{J})^{-1}\mathrm{J}^{\mathsf{T}} \tag{46}$$

is the pseudo-inverse of the matrix J.

1.3.2 Uncertainties of the fit parameters

Equation (45) defines $\Delta c_{k=1\ldots m}$ as function of $y_{i=1\ldots n}$. The question is, if y_i are determined with uncertainties δy_i, what are the uncertainties of Δc_k?

The answer is given by the *propagation of uncertainty* rule which says that the (co)variances $\delta c_k \delta c_j$ are given as

$$\delta c_k \delta c_j = \sum_i \frac{\partial \Delta c_k}{\partial y_i} \frac{\partial \Delta c_j}{\partial y_i} \delta y_i \delta y_i = \sum_i (\mathrm{J}^{-1})_{ki} (\mathrm{J}^{-1})_{ji} \ . \tag{47}$$

In matrix notation the covariance matrix $\Sigma_{kj} = \delta c_k \delta c_j$ is given as

$$\Sigma = \mathrm{J}^{-1}\mathrm{J}^{-\mathsf{T}} = (\mathrm{J}^{\mathsf{T}}\mathrm{J})^{-1} \ . \tag{48}$$

The uncertainties of the fitting parameters are then given as the square roots of the diagonal elements of the covariance matrix,

$$\delta c_k = \sqrt{\Sigma_{kk}} \ . \tag{49}$$

Notice that within the approximation (42) (that should work well at the minimum) the covariance matrix is given via the inverse of the Hessian matrix, H^{-1}, at the minimum,

$$\Sigma = (\mathrm{J}^{\mathsf{T}}\mathrm{J})^{-1} = 2\mathrm{H}^{-1} \ , \tag{50}$$

which is the canonical texbook result. It can also be obtained from the Taylor expansion of the variation of χ^2 with respect to fit parameters at the minimum, where the gradient is zero,

$$\delta\chi^2 = \frac{1}{2}\delta\mathbf{c}^{\mathsf{T}}\mathrm{H}\delta\mathbf{c} = \mathrm{trace}\left(\frac{1}{2}\Sigma\mathrm{H}\right) \ . \tag{51}$$

The uncertainties of fit parameters are determined by a unit variation of χ^2 per degree of freedom (that is, per fit parameter). That is, the matrix inside the trace operator must be the unit $m \times m$ matrix. This gives

$$\Sigma = 2\mathrm{H}^{-1} \ . \tag{52}$$

References

[1] J.A. Nelder and R. Mead, "A simplex method for function minimization", Computer Journal 7 (1965) 308