Nelder-Mead User's Manual – The Nelder-Mead Method –

Michaël BAUDIN

Version 0.3
September 2009

Contents

Notations

n	number of variables
$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$	the unknown
$\mathbf{x}_0 \in \mathbb{R}^n$	the initial guess
$\mathbf{v} \in \mathbb{R}^n$	a vertex
$S = \{\mathbf{v}_i\}_{i=1,m}$	a complex, where $m \geq n+1$ is the number of vertices
$S = \{\mathbf{v}_i\}_{i=1,n+1}$	a simplex (with $n+1$ vertices)
$(\mathbf{v}_i)_j$	the j-th component of the i-th vertex
S_0	the initial simplex
S_k	the simplex at iteration k
$f : \mathbb{R}^n \to \mathbb{R}$	the cost function

Fig. 1 : Notations used in this document

Chapter 1

Nelder-Mead method

In this chapter, we present Nelder and Mead's [8] algorithm. We begin by the analysis of the algorithm, which is based on a variable shape simplex. Then, we present geometric situations where the various steps of the algorithm are used. In the third part, we present the rate of convergence toward the optimum of the Nelder-Mead algorithm. This part is mainly based on Han and Neumann's paper [3], which makes use of a class of quadratic functions with a special initial simplex. The core of this chapter is the analysis of several numerical experiments which have been performed with the neldermead component. We analyze the behavior of the algorithm on quadratic functions and present several counter examples where the Nelder-Mead algorithm is known to fail.

1.1 Introduction

In this section, we present the Nelder-Mead algorithm for unconstrained optimization. This algorithm is based on the iterative update of a simplex. Then we present various geometric situations which might occur during the algorithm.

1.1.1 Overview

The goal of the Nelder and Mead algorithm is to solve the following unconstrained optimization problem

$$\min f(\mathbf{x}) \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, n is the number of optimization parameters and f is the objective function $f : \mathbb{R}^n \to \mathbb{R}$.

The Nelder-Mead method is an improvement over the Spendley's et al. method with the goal of allowing the simplex to vary in *shape*, and not only in *size*, as in Spendley's et al. algorithm.

This algorithms is based on the iterative update of a *simplex* made of $n+1$ points $S = \{\mathbf{v}_i\}_{i=1,n+1}$. Each point in the simplex is called a *vertex* and is associated with a function value $f_i = f(\mathbf{v}_i)$ for $i = 1, n+1$.

The vertices are sorted by increasing function values so that the *best* vertex has index 1 and the *worst* vertex has index $n+1$

$$f_1 \leq f_2 \leq \ldots \leq f_n \leq f_{n+1}. \tag{1.2}$$

The $\mathbf{v}_1$ vertex (resp. the $\mathbf{v}_{n+1}$ vertex) is called the *best* vertex (resp. *worst*), because it is associated with the lowest (resp. highest) function value.

The centroid of the simplex $\overline{\mathbf{x}}(j)$ is the center of the vertices where the vertex $\mathbf{v}_j$ has been excluded. This centroid is

$$\overline{\mathbf{x}}(j) = \frac{1}{n} \sum_{i=1,n+1,i\neq j} \mathbf{v}_i. \tag{1.3}$$

The algorithm makes use of one coefficient $\rho > 0$, called the reflection factor. The standard value of this coefficient is $\rho = 1$. The algorithm attempts to replace some vertex $\mathbf{v}_j$ by a new vertex $\mathbf{x}(\rho, j)$ on the line from the vertex $\mathbf{v}_j$ to the centroid $\overline{\mathbf{x}}(j)$. The new vertex $\mathbf{x}(\rho, j)$ is defined by

$$\mathbf{x}(\rho, j) = (1+\rho)\overline{\mathbf{x}}(j) - \rho\mathbf{v}_j. \tag{1.4}$$

1.1.2 Algorithm

In this section, we analyze the Nelder-Mead algorithm, which is presented in figure 1.1.

The Nelder-Mead algorithm makes use of four parameters: the coefficient of reflection ρ, expansion χ, contraction γ and shrinkage σ. When the expansion or contraction steps are performed, the shape of the simplex is changed, thus "adapting itself to the local landscape" [8].

These parameters should satisfy the following inequalities [8, 5]

$$\rho > 0, \qquad \chi > 1, \qquad \chi > \rho, \qquad 0 < \gamma < 1 \qquad \text{and} \qquad 0 < \sigma < 1. \tag{1.5}$$

The standard values for these coefficients are

$$\rho = 1, \qquad \chi = 2, \qquad \gamma = \frac{1}{2} \qquad \text{and} \qquad \sigma = \frac{1}{2}. \tag{1.6}$$

In [4], the Nelder-Mead algorithm is presented with other parameter names, that is $\mu_r = \rho$, $\mu_e = \rho\chi$, $\mu_{ic} = -\gamma$ and $\mu_{oc} = \rho\gamma$. These coefficients must satisfy the following inequality

$$-1 < \mu_{ic} < 0 < \mu_{oc} < \mu_r < \mu_e. \tag{1.7}$$

At each iteration, we compute the centroid $\overline{\mathbf{x}}(n+1)$ where the worst vertex $\mathbf{v}_{n+1}$ has been excluded. This centroid is

$$\overline{\mathbf{x}}(n+1) = \frac{1}{n} \sum_{i=1,n} \mathbf{v}_i. \tag{1.8}$$

Compute an initial simplex S_0
Sorts the vertices S_0 with increasing function values
$S \leftarrow S_0$
while $\sigma(S) > tol$ **do**
  $\overline{x} \leftarrow \overline{x}(n+1)$
  $x_r \leftarrow x(\rho, n+1)$ {Reflect}
  $f_r \leftarrow f(x_r)$
  if $f_r < f_1$ **then**
    $x_e \leftarrow x(\rho\chi, n+1)$ {Expand}
    $f_e \leftarrow f(x_e)$
    if $f_e < f_r$ **then**
      Accept x_e
    else
      Accept x_r
    end if
  else if $f_1 \leq f_r < f_n$ **then**
    Accept x_r
  else if $f_n \leq f_r < f_{n+1}$ **then**
    $x_c \leftarrow x(\rho\gamma, n+1)$ {Outside contraction}
    $f_c \leftarrow f(x_c)$
    if $f_c < f_r$ **then**
      Accept x_c
    else
      Compute the points $x_i = x_1 + \sigma(x_i - x_1)$, $i = 2, n+1$ {Shrink}
      Compute $f_i = f(\mathbf{v}_i)$ for $i = 2, n+1$
    end if
  else
    $x_c \leftarrow x(-\gamma, n+1)$ {Inside contraction}
    $f_c \leftarrow f(x_c)$
    if $f_c < f_{n+1}$ **then**
      Accept x_c
    else
      Compute the points $x_i = x_1 + \sigma(x_i - x_1)$, $i = 2, n+1$ {Shrink}
      Compute $f_i = f(\mathbf{v}_i)$ for $i = 2, n+1$
    end if
  end if
  Sort the vertices of S with increasing function values
end while

Fig. 1.1 : Nelder-Mead algorithm – Standard version

We perform a reflection with respect to the worst vertex $\mathbf{v}_{n+1}$, which creates the reflected point $\mathbf{x}_r$ defined by

$$\mathbf{x}_r = \mathbf{x}(\rho, n+1) = (1+\rho)\overline{\mathbf{x}}(n+1) - \rho\mathbf{v}_{n+1} \tag{1.9}$$

We then compute the function value of the reflected point as $f_r = f(\mathbf{x}_r)$.

From that point, there are several possibilities, which are listed below. Most steps try to replace the worst vertex $\mathbf{v}_{n+1}$ by a better point, which is computed depending on the context.

- In the case where $f_r < f_1$, the reflected point $\mathbf{x}_r$ were able to improve (i.e. reduce) the function value. In that case, the algorithm tries to expand the simplex so that the function value is improved even more. The expansion point is computed by

  $$\mathbf{x}_e = \mathbf{x}(\rho\chi, n+1) = (1+\rho\chi)\overline{\mathbf{x}}(n+1) - \rho\chi\mathbf{v}_{n+1} \tag{1.10}$$

  and the function is computed at this point, i.e. we compute $f_e = f(\mathbf{x}_e)$. If the expansion point allows to improve the function value, the worst vertex $\mathbf{v}_{n+1}$ is rejected from the simplex and the expansion point $\mathbf{x}_e$ is accepted. If not, the reflection point $\mathbf{x}_r$ is accepted.

- In the case where $f_1 \leq f_r < f_n$, the worst vertex $\mathbf{v}_{n+1}$ is rejected from the simplex and the reflected point $\mathbf{x}_r$ is accepted.

- In the case where $f_n \leq f_r < f_{n+1}$, we consider the point

  $$\mathbf{x}_c = \mathbf{x}(\rho\gamma, n+1) = (1+\rho\gamma)\overline{\mathbf{x}}(n+1) - \rho\gamma\mathbf{v}_{n+1} \tag{1.11}$$

  is considered. If the point $\mathbf{x}_c$ is better than the reflection point $\mathbf{x}_r$, then it is accepted. If not, a shrink step is performed, where all vertices are moved toward the best vertex $\mathbf{v}_1$.

- In other cases, we consider the point

  $$\mathbf{x}_c = \mathbf{x}(-\gamma, n+1) = (1-\gamma)\overline{\mathbf{x}}(n+1) + \gamma\mathbf{v}_{n+1}. \tag{1.12}$$

  If the point $\mathbf{x}_c$ is better than the worst vertex $\mathbf{x}_{n+1}$, then it is accepted. If not, a shrink step is performed.

The algorithm from figure 1.1 is the most popular variant of the Nelder-Mead algorithm. But the original paper is based on a "greedy" expansion, where the expansion point is accepted if it is better than the best point (and not if it is better than the reflection point). This "greedy" version is implemented in AS47 by O'Neill in [9] and the corresponding algorithm is presented in figure 1.2.

```
[...]
```

$\mathbf{x}_e \leftarrow \mathbf{x}(\rho\chi, n+1)$ {Expand}
$f_e \leftarrow f(\mathbf{x}_e)$
if $f_e < f_1$ **then**
  Accept $\mathbf{x}_e$
else
  Accept $\mathbf{x}_r$
end if

```
[...]
```

Fig. 1.2 : Nelder-Mead algorithm – Greedy version

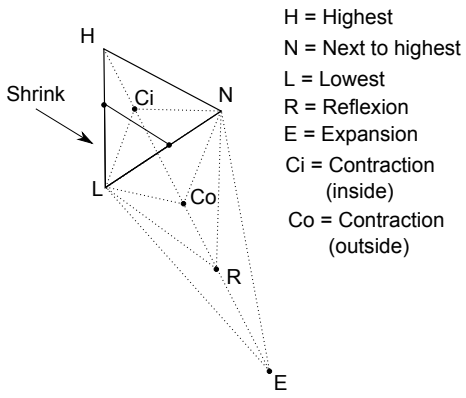

Fig. 1.3 : Nelder-Mead simplex steps

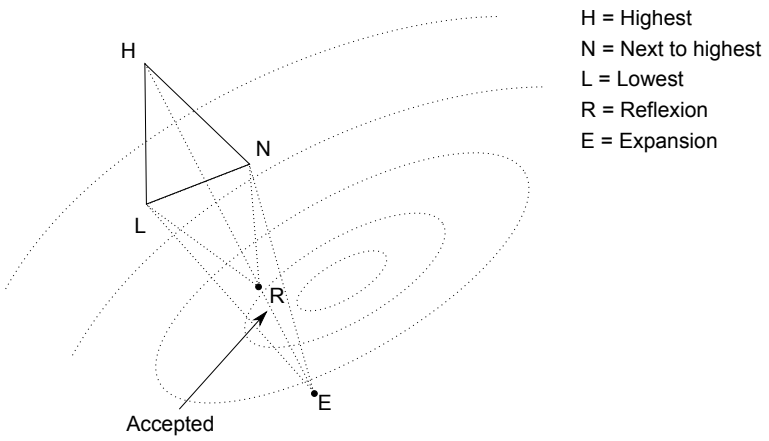

Fig. 1.4 : Nelder-Mead simplex moves – Reflection

1.2 Geometric analysis

The figure 1.3 presents the various moves of the simplex in the Nelder-Mead algorithm.

The figure 1.4 to 1.9 present the detailed situations when each type of step occur. We emphasize that these figures are not the result of numerical experiments. These figures been created in order to illustrate specific points of the algorithm.

- Obviously, the expansion step is performed when the simplex is far away from the optimum. The direction of descent is then followed and the worst vertex is moved into that direction.
- When the reflection step is performed, the simplex is getting close to an valley, since the expansion point does not improve the function value.
- When the simplex is near the optimum, the inside and outside contraction steps may be performed, which allows to decrease the size of the simplex. The figure 1.6, which illustrates the inside contraction step, happens in "good" situations. As presented in section 1.4.4, applying repeatedly the inside contraction step can transform the simplex into a degenerate simplex, which may let the algorithm converge to a non stationnary point.
- The shrink steps (be it after an outside contraction or an inside contraction) occurs only in very special situations. In practical experiments, shrink steps are rare.

1.3 Convergence properties on a quadratic

In this section, we reproduce one result presented by Han and Neumann [3], which states the rate of convergence toward the optimum on a class of quadratic functions with a special initial simplex. Some additional results are also presented in the Phd thesis by Lixing Han [2]. We study

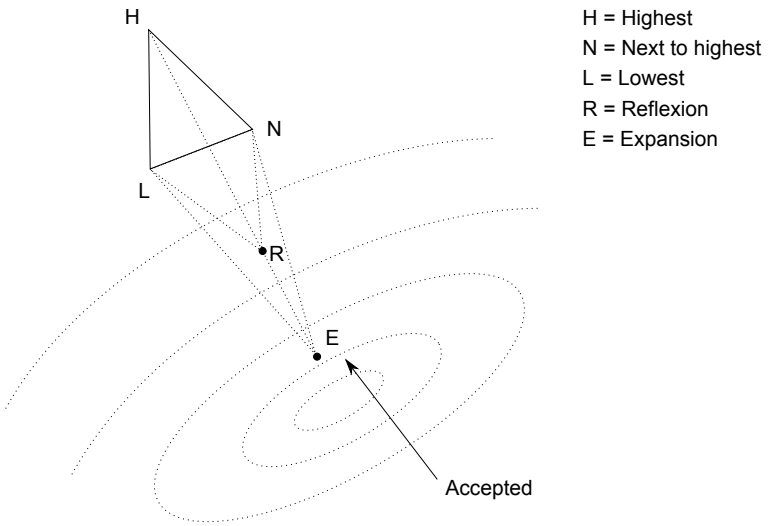

Fig. 1.5 : Nelder-Mead simplex moves – Expansion

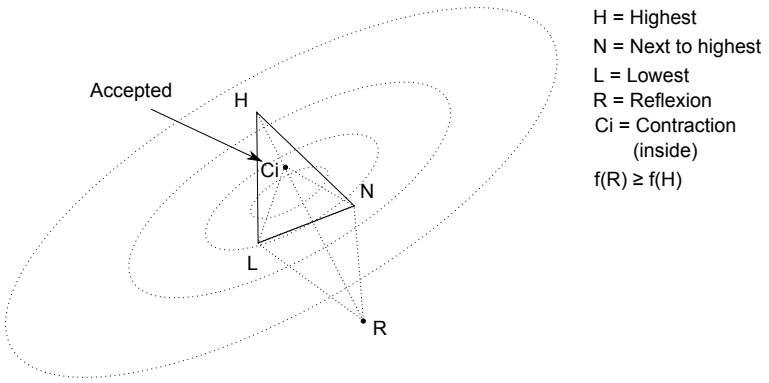

Fig. 1.6 : Nelder-Mead simplex moves - Inside contraction

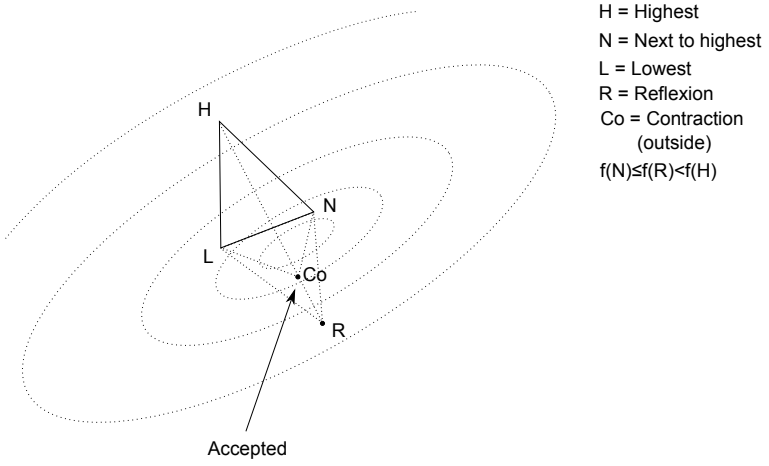

Fig. 1.7 : Nelder-Mead simplex moves – Outside contraction

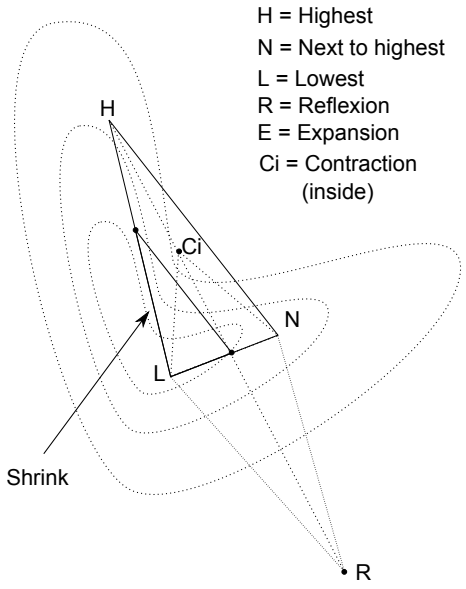

Fig. 1.8 : Nelder-Mead simplex moves – Shrink after inside contraction.

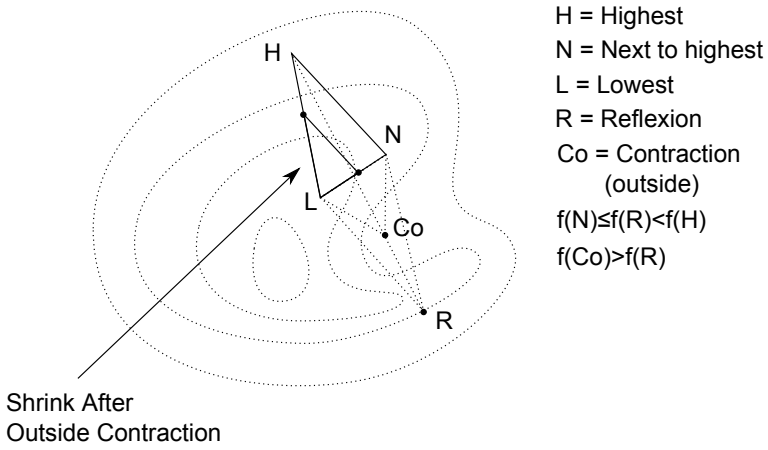

Fig. 1.9 : Nelder-Mead simplex moves – Shrink after outside contraction

a generalized quadratic and use a particular initial simplex. We show that the vertices follow a recurrence equation, which is associated with a characteristic equation. The study of the roots of these characteristic equations give an insight of the behavior of the Nelder-Mead algorithm when the dimension n increases.

Let us suppose than one wants to minimize the function

$$f(\mathbf{x}) = x_1^2 + \ldots + x_n^2 \tag{1.13}$$

with the initial simplex

$$S_0 = \left[\mathbf{0}, \mathbf{v}_1^{(0)}, \ldots, \mathbf{v}_n^{(0)}\right] \tag{1.14}$$

With this choice of the initial simplex, the best vertex remains fixed at $\mathbf{0} = (0, 0, \ldots, 0)^T \in \mathbb{R}^n$. As the cost function 1.13 is strictly convex, the Nelder-Mead method never performs the *shrink* step. Therefore, at each iteration, a new simplex is formed by replacing the worst vertex $\mathbf{v}_n^{(k)}$, by a new, better vertex. Assume that the Nelder-Mead method generates a sequence of simplices $\{S_k\}_{k\geq 0}$ in $\mathbb{R}^n$, where

$$S_k = \left[\mathbf{0}, \mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_n^{(n)}\right] \tag{1.15}$$

We wish that the sequence of simplices $S_k \to \mathbf{0} \in \mathbb{R}^n$ as $k \to \infty$. To measure the progress of convergence, Han and Neumann use the oriented length $\sigma_+(S_k)$ of the simplex S_k, defined by

$$\sigma_+(S) = \max_{i=2,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2. \tag{1.16}$$

We say that a sequence of simplices $\{S_k\}_{k\geq 0}$ converges to the minimizer $\mathbf{0} \in \mathbb{R}^n$ of the function in equation 1.13 if $\lim_{k\to\infty} \sigma_+(S_k) = 0$.

We measure the rate of convergence defined by

$$\rho(S_0, n) = \text{lim sup}_{k\to\infty} \left(\sum_{i=0,k-1} \frac{\sigma(S_{i+1})}{\sigma(S_i)} \right)^{1/k} \tag{1.17}$$

That definition can be viewed as the geometric mean of the ratio of the oriented lengths between successive simplices and the minimizer 0. This definition implies

$$\rho(S_0, n) = \text{lim sup}_{k\to\infty} \left(\frac{\sigma(S_{k+1})}{\sigma(S_0)} \right)^{1/k} \tag{1.18}$$

According to the definition, the algorithm is convergent if $\rho(S_0, n) < 1$. The larger the $\rho(S_0, n)$, the slower the convergence. In particular, the convergence is very slow when $\rho(S_0, n)$ is close to 1. The analysis is based on the fact that the Nelder-Mead method generates a sequence of simplices in $\mathbb{R}^n$ satisfying

$$S_k = \left[\mathbf{0}, \mathbf{v}^{(k+n-1)}, \ldots, \mathbf{v}^{(k+1)}, \mathbf{v}^{(k)}\right], \tag{1.19}$$

where $\mathbf{0}, \mathbf{v}^{(k+n-1)}, \ldots, \mathbf{v}^{(k+1)}, \mathbf{v}^{(k)} \in \mathbb{R}^n$ are the vertices of the $k-th$ simplex, with

$$f(\mathbf{0}) < f\left(\mathbf{v}^{(k+n-1)}\right) < f\left(\mathbf{v}^{(k+1)}\right) < f\left(\mathbf{v}^{(k)}\right), \tag{1.20}$$

for $k \geq 0$.

To simplify the analysis, we consider that only one type of step of the Nelder-Mead method is applied repeatedly. This allows to establish recurrence equations for the successive simplex vertices. As the shrink step is never used, and the expansion steps is never used neither (since the best vertex is already at 0), the analysis focuses on the outside contraction, inside contraction and reflection steps.

The centroid of the n best vertices of S_k is given by

$$\begin{aligned} \overline{\mathbf{v}}^{(k)} &= \frac{1}{n}\left(\mathbf{v}^{(k+1)} + \ldots + \mathbf{v}^{(k+n-1)} + \mathbf{0}\right) && (1.21)\\ &= \frac{1}{n}\left(\mathbf{v}^{(k+1)} + \ldots + \mathbf{v}^{(k+n-1)}\right) && (1.22)\\ &= \frac{1}{n}\sum_{i=1,n-1} \mathbf{v}^{(k+i)} && (1.23) \end{aligned}$$

1.3.1 With default parameters

In this section, we analyze the roots of the characteristic equation with *fixed*, standard inside and outside contraction coefficients.

Outside contraction

If the outside contraction step is repeatedly performed with $\mu_{oc} = \rho\gamma = \frac{1}{2}$, then

$$\mathbf{v}^{(k+n)} = \overline{\mathbf{v}}^{(k)} + \frac{1}{2}\left(\overline{\mathbf{v}}^{(k)} - \mathbf{v}^{(k)}\right). \tag{1.24}$$

By plugging the definition of the centroid **??** into the previous equality, we find the recurrence formula

$$2n\mathbf{v}^{(k+n)} - 3\mathbf{v}^{(k+1)} - \ldots - 3\mathbf{v}^{(k+n-1)} + n\mathbf{v}^{(k)} = 0. \tag{1.25}$$

The associated characteristic equation is

$$2n\mu^n - 3\mu^{n-1} - \ldots - 3\mu + n = 0. \tag{1.26}$$

Inside contraction

If the inside contraction step is repeatedly performed with $\mu_{ic} = -\gamma = -\frac{1}{2}$, then

$$\mathbf{v}^{(k+n)} = \overline{\mathbf{v}}^{(k)} - \frac{1}{2}\left(\overline{\mathbf{v}}^{(k)} - \mathbf{v}^{(k)}\right). \tag{1.27}$$

By plugging the definition of the centroid **??** into the previous equality, we find the recurrence formula

$$2n\mathbf{v}^{(k+n)} - \mathbf{v}^{(k+1)} - \ldots - \mathbf{v}^{(k+n-1)} - n\mathbf{v}^{(k)} = 0. \tag{1.28}$$

The associated characteristic equation is

$$2n\mu^n - \mu^{n-1} - \ldots - \mu - n = 0. \tag{1.29}$$

Reflection

If the reflection step is repeatedly performed with $\mu_r = \rho = 1$, then

$$\mathbf{v}^{(k+n)} = \overline{\mathbf{v}}^{(k)} + \left(\overline{\mathbf{v}}^{(k)} - \mathbf{v}^{(k)}\right). \tag{1.30}$$

By plugging the definition of the centroid **??** into the previous equality, we find the recurrence formula

$$n\mathbf{v}^{(k+n)} - 2\mathbf{v}^{(k+1)} - \ldots - 2\mathbf{v}^{(k+n-1)} + n\mathbf{v}^{(k)} = 0. \tag{1.31}$$

The associated characteristic equation is

$$n\mu^n - 2\mu^{n-1} - \ldots - 2\mu + n = 0. \tag{1.32}$$

The recurrence equations 1.26, 1.29 and 1.32 are linear. Their general solutions are of the form

$$\mathbf{v}^{(k)} = \mu_1^k \mathbf{a}_1 + \ldots + \mu_n^k \mathbf{a}_n, \tag{1.33}$$

where $\{\mu_i\}_{i=1,n}$ are the roots of the characteristic equations and $\{\mathbf{a}_i\}_{i=1,n} \in \mathbb{C}^n$ are independent vectors such that $\mathbf{v}^{(k)} \in \mathbb{R}^n$ for all $k \geq 0$.

The analysis by Han and Neumann [3] gives a deep understanding of the convergence rate for this particular situation. For $n = 1$, they show that the convergence rate is $\frac{1}{2}$. For $n = 2$, the convergence rate is $\frac{\sqrt{2}}{2} \approx 0.7$ with a particular choice for the initial simplex. For $n \geq 3$, Han and Neumann [3] perform a numerical analysis of the roots.

In the following Scilab script, we compute the roots of these 3 characteristic equations.

```
1  //
2  // computeroots1 --
3  //   Compute the roots of the characteristic equations of
4  //   usual Nelder-Mead method.
5  //
6  function computeroots1 ( n )
7    // Polynomial for outside contraction :
8    // n - 3x - ... - 3x^(n-1) + 2n x^(n) = 0
9    mprintf("Polynomial for outside contraction :\n");
10   coeffs = zeros(1,n+1);
11   coeffs(1) = n
12   coeffs(2:n) = -3
13   coeffs(n+1) = 2 * n
14   p=poly(coeffs,"x","coeff")
15   disp(p)
```

```
16    mprintf("Roots :\n");
17    r = roots(p)
18    for i=1:n
19      mprintf("Root #%d/%d |%s|=%f\n", i, length(r),string(r(i)),abs(r(i)))
20    end
21    // Polynomial for inside contraction :
22    // - n - x - ... - x^(n-1) + 2n x^(n)= 0
23    mprintf("Polynomial for inside contraction :\n");
24    coeffs = zeros(1,n+1);
25    coeffs(1) = -n
26    coeffs(2:n) = -1
27    coeffs(n+1) = 2 * n
28    p=poly(coeffs,"x","coeff")
29    disp(p)
30    mprintf("Roots :\n");
31    r = roots(p)
32    for i=1:n
33      mprintf("Root #%d/%d |%s|=%f\n", i, length(r),string(r(i)),abs(r(i)))
34    end
35    // Polynomial for reflection :
36    // n - 2x - ... - 2x^(n-1) + n x^(n) = 0
37    mprintf("Polynomial for reflection :\n");
38    coeffs = zeros(1,n+1);
39    coeffs(1) = n
40    coeffs(2:n) = -2
41    coeffs(n+1) = n
42    p=poly(coeffs,"x","coeff")
43    disp(p)
44    r = roots(p)
45    mprintf("Roots :\n");
46    for i=1:n
47      mprintf("Root #%d/%d |%s|=%f\n", i, length(r),string(r(i)),abs(r(i)))
48    end
49  endfunction
```

If we execute this script with $n = 10$, the following output is produced.

```
-->computeroots1 ( 10 )
Polynomial for outside contraction :


                2    3    4    5    6    7    8    9    10
    10 - 3x - 3x - 3x - 3x - 3x - 3x - 3x - 3x - 3x + 20x
Roots :
Root #1/10 |0.5822700+%i*0.7362568|=0.938676
Root #2/10 |0.5822700-%i*0.7362568|=0.938676
Root #3/10 |-0.5439060+%i*0.7651230|=0.938747
Root #4/10 |-0.5439060-%i*0.7651230|=0.938747
```

```
Root #5/10 |0.9093766+%i*0.0471756|=0.910599
Root #6/10 |0.9093766-%i*0.0471756|=0.910599
Root #7/10 |0.0191306+%i*0.9385387|=0.938734
Root #8/10 |0.0191306-%i*0.9385387|=0.938734
Root #9/10 |-0.8918713+%i*0.2929516|=0.938752
Root #10/10 |-0.8918713-%i*0.2929516|=0.938752
Polynomial for inside contraction :

              2   3   4   5   6   7   8    9    10
  - 10 - x - x - x - x - x - x - x - x - x + 20x
Roots :
Root #1/10 |0.7461586+%i*0.5514088|=0.927795
Root #2/10 |0.7461586-%i*0.5514088|=0.927795
Root #3/10 |-0.2879931+%i*0.8802612|=0.926175
Root #4/10 |-0.2879931-%i*0.8802612|=0.926175
Root #5/10 |-0.9260704|=0.926070
Root #6/10 |0.9933286|=0.993329
Root #7/10 |0.2829249+%i*0.8821821|=0.926440
Root #8/10 |0.2829249-%i*0.8821821|=0.926440
Root #9/10 |-0.7497195+%i*0.5436596|=0.926091
Root #10/10 |-0.7497195-%i*0.5436596|=0.926091
Polynomial for reflection :

                2    3    4    5    6    7    8     9    10
    10 - 2x - 2x - 2x - 2x - 2x - 2x - 2x - 2x - 2x + 10x
Roots :
Root #1/10 |0.6172695+%i*0.7867517|=1.000000
Root #2/10 |0.6172695-%i*0.7867517|=1.000000
Root #3/10 |-0.5801834+%i*0.8144859|=1.000000
Root #4/10 |-0.5801834-%i*0.8144859|=1.000000
Root #5/10 |0.9946011+%i*0.1037722|=1.000000
Root #6/10 |0.9946011-%i*0.1037722|=1.000000
Root #7/10 |0.0184670+%i*0.9998295|=1.000000
Root #8/10 |0.0184670-%i*0.9998295|=1.000000
Root #9/10 |-0.9501543+%i*0.3117800|=1.000000
Root #10/10 |-0.9501543-%i*0.3117800|=1.000000
```

The following Scilab script allows to compute the minimum and the maximum of the modulus of the roots. The "e" option of the "roots" command has been used to force the use of the eigenvalues of the companion matrix as the computational method. The default algorithm, based on the Jenkins-Traub Rpoly method is generating a convergence error and cannot be used in this case.

```
function [rminoc , rmaxoc , rminic , rmaxic] = computeroots1_abstract ( n )
  // Polynomial for outside contraction :
  // n - 3x - ... - 3x^(n-1) + 2n x^(n) = 0
  coeffs = zeros(1,n+1);
  coeffs(1) = n
  coeffs(2:n) = -3
  coeffs(n+1) = 2 * n
  p=poly(coeffs,"x","coeff")
  r = roots(p , "e")
  rminoc = min(abs(r))
  rmaxoc = max(abs(r))
  // Polynomial for inside contraction :
  // - n - x - ... - x^(n-1) + 2n x^(n)= 0
  coeffs = zeros(1,n+1);
  coeffs(1) = -n
  coeffs(2:n) = -1
  coeffs(n+1) = 2 * n
  p=poly(coeffs,"x","coeff")
  r = roots(p , "e")
  rminic = min(abs(r))
  rmaxic = max(abs(r))
  mprintf("%d & %f & %f & %f & %f\\\\\n", n, rminoc, rmaxoc, rminic, rmaxic)
endfunction

function drawfigure1 ( nbmax )
  rminoctable = zeros(1,nbmax)
  rmaxoctable = zeros(1,nbmax)
  rminictable = zeros(1,nbmax)
  rmaxictable = zeros(1,nbmax)
  for n = 1 : nbmax
    [rminoc , rmaxoc , rminic , rmaxic] = computeroots1_abstract ( n )
    rminoctable ( n ) = rminoc
    rmaxoctable ( n ) = rmaxoc
    rminictable ( n ) = rminic
    rmaxictable ( n ) = rmaxic
  end
  plot2d ( 1:nbmax , [ rminoctable' , rmaxoctable' , rminictable' , rmaxictable' ] )
  f = gcf();
  f.children.title.text = "Nelder-Mead characteristic equation roots";
  f.children.x_label.text = "Number of variables (n)";
  f.children.y_label.text = "Roots of the characteristic equation";
  captions(f.children.children.children,["R-max-IC","R-min-IC","R-max-OC","R-min-OC"]);
  f.children.children(1).legend_location="in_lower_right";
  for i = 1:4
  mypoly = f.children.children(2).children(i);
  mypoly.foreground=i;
  mypoly.line_style=i;
```

```
48    end
49    xs2png(0,"neldermead-roots.png");
50  endfunction
```

For the reflection characteristic equation, the roots all have a unity modulus. The minimum and maximum roots of the inside contraction ("ic" in the table) and outside contraction ("oc" in the table) steps are presented in table 1.10. These roots are presented graphically in figure 1.11. One can see that the roots converge toward 1 when $n \to \infty$.

n	$\min_{i=1,n} \mu_i^{oc}$	$\max_{i=1,n} \mu_i^{oc}$	$\min_{i=1,n} \mu_i^{ic}$	$\max_{i=1,n} \mu_i^{ic}$
1	0.500000	0.500000	0.500000	0.500000
2	0.707107	0.707107	0.593070	0.843070
3	0.776392	0.829484	0.734210	0.927534
4	0.817185	0.865296	0.802877	0.958740
5	0.844788	0.888347	0.845192	0.973459
6	0.864910	0.904300	0.872620	0.981522
7	0.880302	0.916187	0.892043	0.986406
8	0.892487	0.925383	0.906346	0.989584
9	0.902388	0.932736	0.917365	0.991766
10	0.910599	0.938752	0.926070	0.993329
11	0.917524	0.943771	0.933138	0.994485
12	0.923446	0.948022	0.938975	0.995366
13	0.917250	0.951672	0.943883	0.996051
14	0.912414	0.954840	0.948062	0.996595
15	0.912203	0.962451	0.951666	0.997034
16	0.913435	0.968356	0.954803	0.997393
17	0.915298	0.972835	0.957559	0.997691
18	0.917450	0.976361	0.959999	0.997940
19	0.919720	0.979207	0.962175	0.998151
20	0.922013	0.981547	0.964127	0.998331
21	0.924279	0.983500	0.965888	0.998487
22	0.926487	0.985150	0.967484	0.998621
23	0.928621	0.986559	0.968938	0.998738
24	0.930674	0.987773	0.970268	0.998841
25	0.932640	0.988826	0.971488	0.998932
26	0.934520	0.989747	0.972613	0.999013
27	0.936316	0.990557	0.973652	0.999085
28	0.938030	0.991274	0.974616	0.999149
29	0.939666	0.991911	0.975511	0.999207
30	0.941226	0.992480	0.976346	0.999259
31	0.942715	0.992991	0.977126	0.999306
32	0.944137	0.993451	0.977856	0.999348
33	0.945495	0.993867	0.978540	0.999387
34	0.946793	0.994244	0.979184	0.999423
35	0.948034	0.994587	0.979791	0.999455
36	0.949222	0.994900	0.980363	0.999485
37	0.950359	0.995187	0.980903	0.999513
38	0.951449	0.995450	0.981415	0.999538
39	0.952494	0.995692	0.981900	0.999561
40	0.953496	0.995915	0.982360	0.999583
45	0.957952	0.996807	0.984350	0.999671
50	0.961645	0.997435	0.985937	0.999733
55	0.964752	0.997894	0.987232	0.999779
60	0.967399	0.998240	0.988308	0.999815
65	0.969679	0.998507	0.989217	0.999842
70	0.971665	0.998718	0.989995	0.999864
75	0.973407	0.998887	0.990669	0.999881
80	0.974949	0.999024	0.991257	0.999896
85	0.976323	0.999138	0.991776	0.999908
90	0.977555	0.999233	0.992236	0.999918
95	0.978665	0.999313	0.992648	0.999926
100	0.979671	0.999381	0.993018	0.999933

Fig. 1.10 : Roots of the characteristic equations of the Nelder-Mead method with standard coefficients. (Some results are not displayed to make the table fit the page).

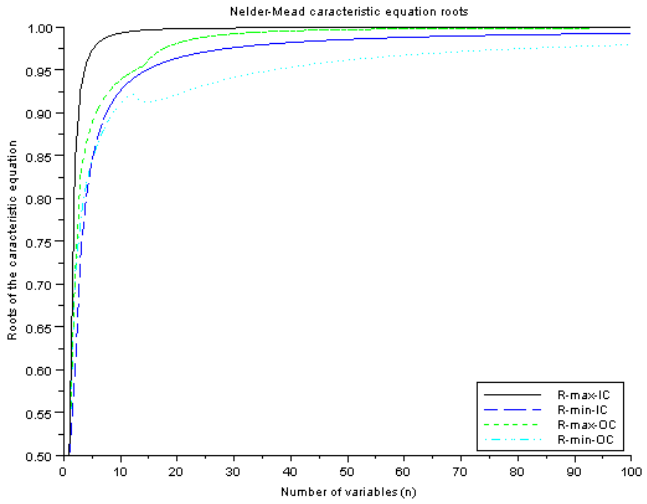

Fig. 1.11 : Modulus of the roots of the characteristic equations of the Nelder-Mead method with standard coefficients – R-max-IC is the maximum of the modulus of the root of the Inside Contraction steps

1.3.2 With variable parameters

In this section, we analyze the roots of the characteristic equation with *variable* inside and outside contraction coefficients.

Outside contraction If the outside contraction step is repeatedly performed with variable $\mu_{oc} > 0$, then

$$\begin{aligned} \mathbf{v}^{(k+n)} &= \overline{\mathbf{v}}^{(k)} + \mu_{oc}\left(\overline{\mathbf{v}}^{(k)} - \mathbf{v}^{(k)}\right) && (1.34)\\ &= (1+\mu_{oc})\overline{\mathbf{v}}^{(k)} - \mu_{oc}\mathbf{v}^{(k)} && (1.35) \end{aligned}$$

By plugging the definition of the centroid into the previous equality, one find the recurrence formula

$$n\mathbf{v}^{(k+n)} - (1+\mu_{oc})\mathbf{v}^{(k+1)} - \ldots - (1+\mu_{oc})\mathbf{v}^{(k+n-1)} + n\mu_{oc}\mathbf{v}^{(k)} = 0 \quad (1.36)$$

The associated characteristic equation is

$$n\mu^n - (1+\mu_{oc})\mu^{n-1} - \ldots - (1+\mu_{oc})\mu + n\mu_{oc} = 0. \quad (1.37)$$

Inside contraction We suppose that the inside contraction step is repeatedly performed with $-1 < \mu_{ic} < 0$. The characteristic equation is the same as 1.37, but it is here studied in the range $\mu_{ic} \in]-1, 0[$.

To study the convergence of the method, one simply have to study the roots of equation 1.37, where the range $]-1,0[$ corresponds to the inside contraction (with $-1/2$ as the standard value) and where the range $]0,\mu_r[$ corresponds to the outside contraction (with $1/2$ as the standard value).

In the following Scilab script, one computes the minimum and maximum root of the characteristic equation, with n fixed.

```
//
// rootsvariable --
//   Compute roots of the characteristic equation
//   of Nelder-Mead with variable coefficient mu.
// Polynomial for outside/inside contraction :
// n mu - (1+mu)x - ... - (1+mu)x^(n-1) + n x^(n) = 0
//
function [rmin , rmax] = rootsvariable ( n , mu )
  coeffs = zeros(1,n+1);
  coeffs(1) = n * mu
  coeffs(2:n) = -(1+mu)
  coeffs(n+1) = n
  p=poly(coeffs,"x","coeff")
  r = roots(p , "e")
  rmin = min(abs(r))
  rmax = max(abs(r))
  mprintf("%f & %f & %f \\\\\n", mu, rmin, rmax)
endfunction

function drawfigure_variable ( n , nmumax )
  rmintable = zeros(1,nmumax)
  rmaxtable = zeros(1,nmumax)
  mutable = linspace ( -1 , 1 , nmumax )
  for index = 1 : nmumax
    mu = mutable ( index )
    [rmin , rmax ] = rootsvariable ( n , mu )
    rmintable ( index ) = rmin
    rmaxtable ( index ) = rmax
  end
  plot2d ( mutable , [ rmintable' , rmaxtable' ] )
  f = gcf();
  pause
  f.children.title.text = "Nelder-Mead characteristic equation roots";
  f.children.x_label.text = "Contraction coefficient";
  f.children.y_label.text = "Roots of the characteristic equation";
  captions(f.children.children.children,["R-max","R-min"]);
  f.children.children(1).legend_location="in_lower_right";
  for i = 1:2
  mypoly = f.children.children(2).children(i);
  mypoly.foreground=i;
```

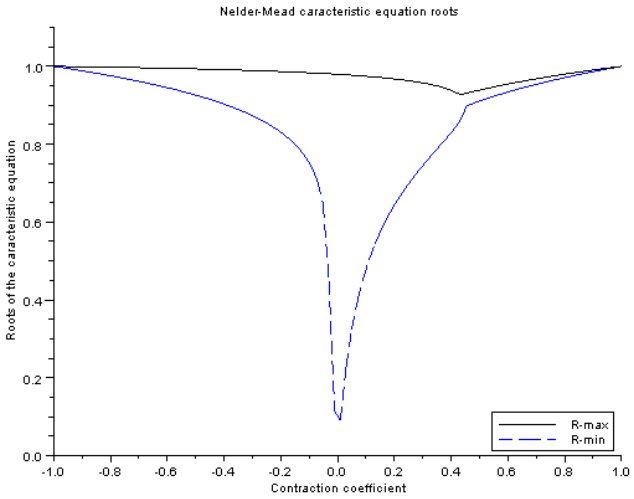

Fig. 1.12 : Modulus of the roots of the characteristic equations of the Nelder-Mead method with variable contraction coefficient and $n = 10$ – R-max is the maximum of the modulus of the root of the characteristic equation

```
41    mypoly.line_style=i;
42    end
43    xs2png(0,"neldermead-roots-variable.png");
44  endfunction
```

The figure 1.12 presents the minimum and maximum modulus of the roots of the characteristic equation with $n = 10$. The result is that when μ_{oc} is close to 0, the minimum root has a modulus close to 0. The maximum root remains close to 1, whatever the value of the contraction coefficient. This result would mean that either modifying the contraction coefficient has no effect (because the maximum modulus of the roots is close to 1) or diminishing the contraction coefficient should improve the convergence speed (because the minimum modulus of the roots gets closer to 0). This is the expected result because the more the contraction coefficient is close to 0, the more the new vertex is close to 0, which is, in our particular situation, the global minimizer. No general conclusion can be drawn from this single experiment.

1.4 Numerical experiments

In this section, we present some numerical experiments with the Nelder-Mead algorithm.

1.4.1 Quadratic function

The function we try to minimize is the following quadratic in 2 dimensions

$$f(x_1, x_2) = x_1^2 + x_2^2 - x_1 x_2 \tag{1.38}$$

The stopping criteria is based on the relative size of the simplex with respect to the size of the initial simplex

$$\sigma(S) < tol \times \sigma(S_0) \tag{1.39}$$

The initial simplex is computed from the coordinate axis and the unit length. The numerical results are presented in table 1.13.

Iterations	65
Function Evaluations	127
x_0	$(2.0, 2.0)$
Relative tolerance on simplex size	10^{-8}
Exact $x^\star$	$(0., 0.)$
Computed $x^\star$	$(7.3e-10, -2.5e-9)$
Computed $f(x^\star)$	$8.7e-18$

Fig. 1.13 : Numerical experiment with Nelder-Mead method on the quadratic function $f(x_1, x_2) = x_1^2 + x_2^2 - x_1 x_2$

The various simplices generated during the iterations are presented in figure 1.14. The method use reflections in the early iterations. Then there is no possible improvement using reflections and shrinking is necessary.

The figure 1.15 presents the history of the oriented length of the simplex. The length is updated at each iteration, which generates a continuous evolution of the length, compared to the step-by-step evolution of the simplex with the Spendley et al. algorithm.

The convergence is quite fast in this case, since less than 60 iterations allow to get a function value lower than 10^{-15}, as shown in figure 1.16.

Badly scaled quadratic function

The function we try to minimize is the following quadratic in 2 dimensions

$$f(x_1, x_2) = a x_1^2 + x_2^2, \tag{1.40}$$

where $a > 0$ is a chosen scaling parameter. The more a is large, the more difficult the problem is to solve with the simplex algorithm.

We set the maximum number of function evaluations to 400. The initial simplex is computed from the coordinate axis and the unit length.

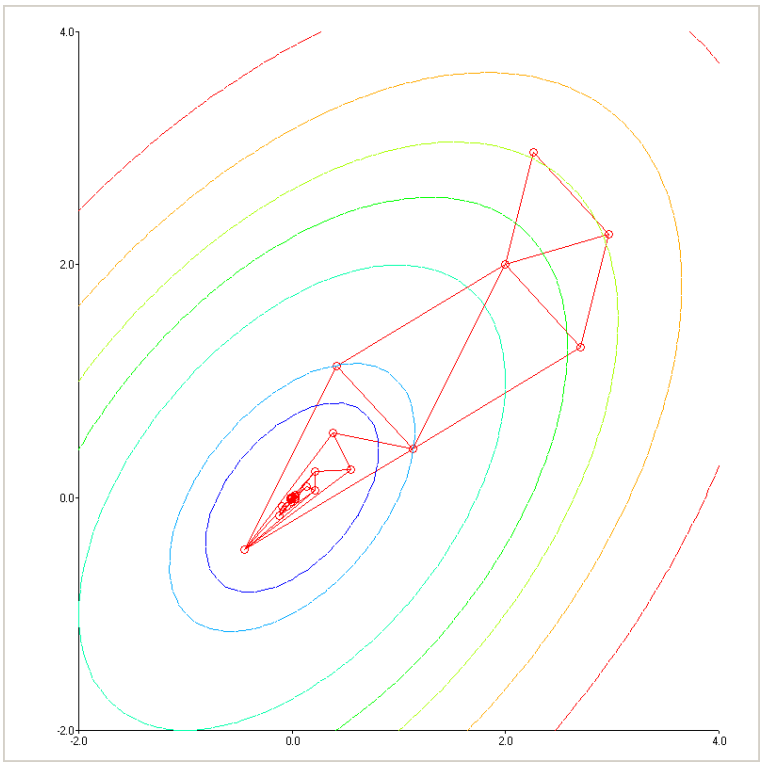

Fig. 1.14 : Nelder-Mead numerical experiment – history of simplex

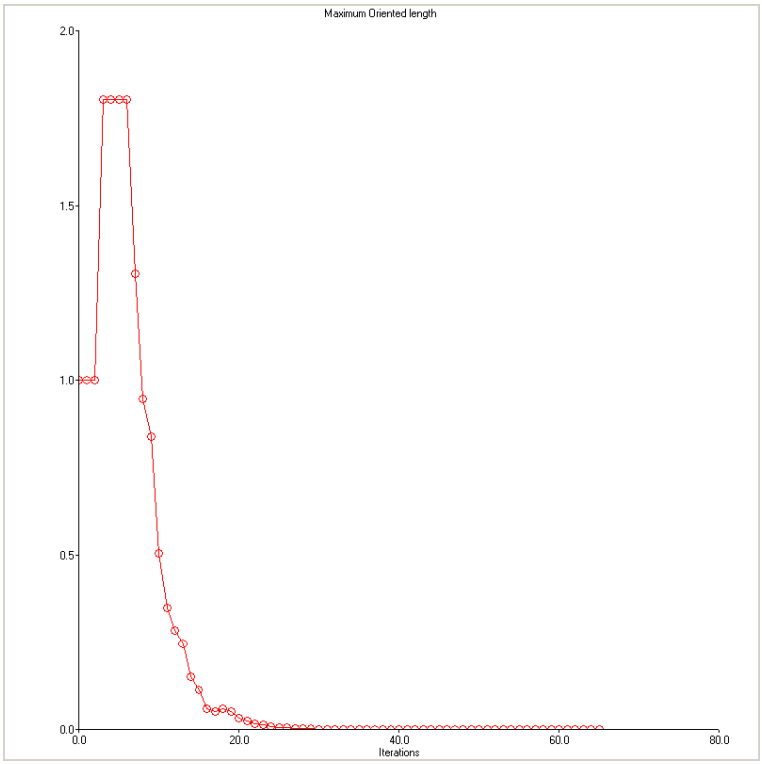

Fig. 1.15 : Nelder-Mead numerical experiment – history of length of simplex

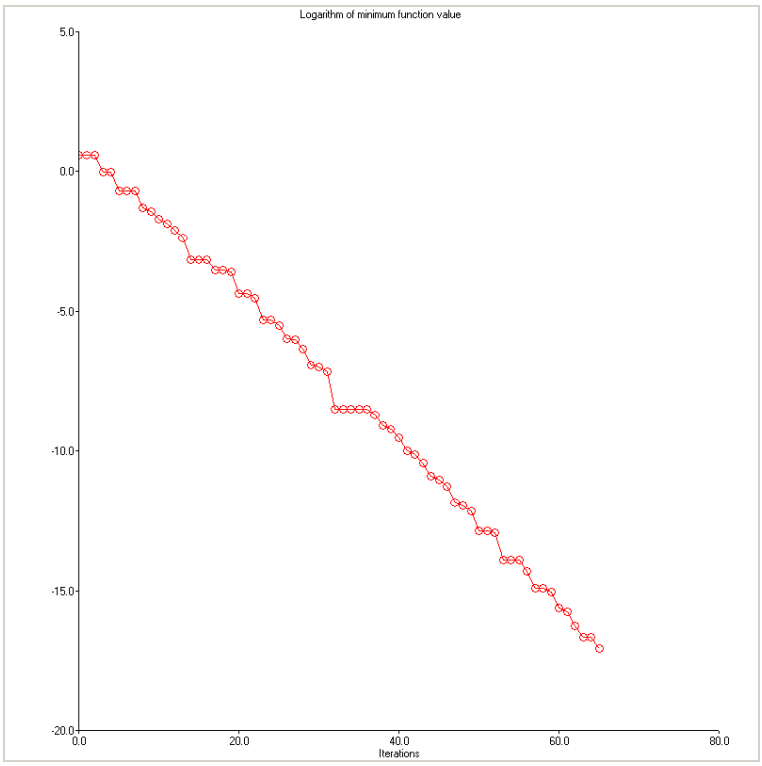

Fig. 1.16 : Nelder-Mead numerical experiment – history of logarithm of function

The numerical results are presented in table 1.17, where the experiment is presented for $a = 100$. One can check that the number of function evaluation (161 function evaluations) is much lower than the number for the fixed shape Spendley et al. method (400 function evaluations) and that the function value at optimum is very accurate ($f(x^\star) \approx 1.e-17$ compared to Spendley's et al. $f(x^\star) \approx 0.08$).

In figure 1.18, we analyze the behavior of the method with respect to scaling. We check that the method behave very smoothly, with a very small number of additional function evaluations when the scaling deteriorates. This shows how much the Nelder-Mead algorithms improves over Spendley's et al. method.

1.4.2 Sensitivity to dimension

In this section, we try to reproduce the result presented by Han and Neumann [3], which shows that the convergence rate of the Nelder-Mead algorithms rapidly deteriorates when the number of variables increases. The function we try to minimize is the following quadratic in n-dimensions

$$f(\mathbf{x}) = \sum_{i=1,n} x_i^2. \tag{1.41}$$

The initial simplex is computed from the coordinate axis and the unit length. The initial guess is at 0 so that the first vertex is the origin ; this vertex is never updated during the iterations.

	Nelder-Mead	Spendley et al.
Iterations	83	340
Function Evaluations	161	400
a	100.0	-
x_0	$(10.0, 10.0)$	-
Relative tolerance on simplex size	-	
Exact $x^\star$	$(0., 0.)$	-
Computed $x^\star$	$(2.e-10, -3.e-9)$	$(0.001, 0.2)$
Computed $f(x^\star)$	$1.e-17$	0.08

Fig. 1.17 : Numerical experiment with Nelder-Mead method on a badly scaled quadratic function. The variable shape Nelder-Mead algorithm improves the accuracy of the result compared to the fixed shaped Spendley et al. method.

a	Function evaluations	Computed $f(x^\star)$	Computed $x^\star$
1.0	139	$8.0e-18$	$(2.e-9-1.e-9)$
10.0	151	$7.0e-17$	$(5.e-102.e-9)$
100.0	161	$1.0e-17$	$(2.e-10-3.e-9)$
1000.0	165	$1.0e-17$	$(-1.e-0109.e-10)$
10000.0	167	$3.0e-17$	$(5.0e-11, -1.0e-10)$

Fig. 1.18 : Numerical experiment with Spendley's et al. method on a badly scaled quadratic function

The figure 1.21 presents the results of this experiment for $n = 1, 19$.

During the iterations, no shrink steps are performed. The algorithm performs reflections, inside and outside contractions. The figure 1.19 shows the detailed sequence of iterations for $n = 10$. One can see that there is no general pattern for the iterations. One can check, however, that there are never no more than n consecutive reflection steps, which is as expected. After one or more contractions, the reflection steps move the worst vertices toward better function values. But there are only $n + 1$ vertices so that the n worst vertices are moved in at most n reflection steps.

The figure 1.20 presents the number and the kind of steps performed during the iterations for $n = 1, 19$. It appears that the number of shrink steps and expansion steps is zero, as expected. More interesting is that the number of reflection is larger than the number of inside contraction when n is large. The number of outside contraction is always the smallest in this case.

One can check that the number of function evaluations increases approximately linearly with the dimension of the problem in figure 1.22. A rough rule of thumb is that, for $n = 1, 19$, the number of function evaluations is equal to $100n$.

The figure 1.23 presents the rate of convergence depending on the number of variables. The

I I I I I I I I I I I I I I I I I I I R R R R R R R R R R R R I O
R R R R R R R R R R R R I R I I R I R O I I I I R I R I I R O I R R
R R I R I R I R R R R R R R I R R R R I R I I R I R I I I R R I I I R
R R I R R I R R R R R R I R I R R R R R I R R O R R I O I O R R R R I
I I O R I R R R R R R I I I R R I I R R R O R I I R R R I R I I O I R
I R R O I I R R R R I R R O I R R O R I R I R I R R I R I R R R I I I
I I O R R R R I I I R R R R I I I R R R I I I I R R R R I I R R R R I R
R R I O I R R I I R R R R O I R I I R R R R R R O R R R O I R R I I I
I O R I I I R I I I I R R I I R R I R R R R R R I R R I I R R O R I I
O R I R O O R O I I R I I I R I I R R R R R R R R R R R R I R R O I R
I O I R I I I I R I I R I I R I R O R I O R I R I R R R O R I R R R I
I R I R R R I R I R R R R I I R R I R R R R I I R R R R I I R I R I I
O I R I I R R R R R R R R I O I R R I I I R I R I I I I R R R R I R R
I R I R R R R I I R R R I I I I R I I I I R I R R I R I O R R R I R I
O I R R I I R R I R R R R O R R R R I O R R R I R I I I I R I R R R R
R I I R I I R R R R R O R R R I R R R R I R I R R I R I I I R R I I R R
I I I I R R R R R R R R R I R R O R R R R R O I I I I R I I R O I I R
R R R R I I I R R

Fig. 1.19 : Numerical experiment with Nelder-Mead method on a generalized quadratic function - steps of the algorithm : I = inside contraction, O = outside contraction, R = reflection, S = shrink

n	# Reflections	# Expansion	# Inside Contractions	# Outside Contractions	#Shrink
1	0	0	27	0	0
2	0	0	5	49	0
3	54	0	45	36	0
4	93	0	74	34	0
5	123	0	101	33	0
6	170	0	122	41	0
7	202	0	155	35	0
8	240	0	178	41	0
9	267	0	205	40	0
10	332	0	234	38	0
11	381	0	267	36	0
12	476	0	299	32	0
13	473	0	316	42	0
14	545	0	332	55	0
15	577	0	372	41	0
16	635	0	396	46	0
17	683	0	419	52	0
18	756	0	445	55	0
19	767	0	480	48	0

Fig. 1.20 : Numerical experiment with Nelder-Mead method on a generalized quadratic function – number and kinds of steps performed

n	Function evaluations	Iterations	$\rho(S_0, n)$
1	56	28	0.5125321059829373
2	111	55	0.71491052830553248
3	220	136	0.87286283470760984
4	314	202	0.91247307800713973
5	397	258	0.93107793607270162
6	503	334	0.94628781077508028
7	590	393	0.95404424343636474
8	687	460	0.96063768057900478
9	767	513	0.96471820169933631
10	887	605	0.97000569588245511
11	999	685	0.97343652480535203
12	1151	808	0.97745310525741003
13	1203	832	0.97803465666405531
14	1334	933	0.98042500139065414
15	1419	991	0.98154526298964495
16	1536	1078	0.98305435726547608
17	1643	1155	0.98416149958157839
18	1775	1257	0.98544909490809807
19	1843	1296	0.98584701106083183

Fig. 1.21 : Numerical experiment with Nelder-Mead method on a generalized quadratic function

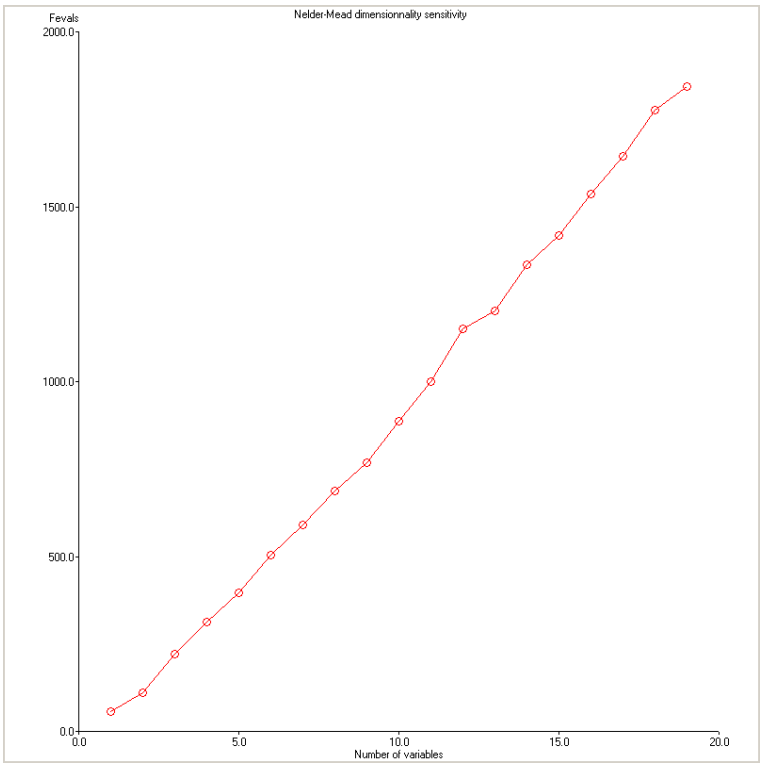

Fig. 1.22 : Nelder-Mead numerical experiment – number of function evaluations depending on the number of variables

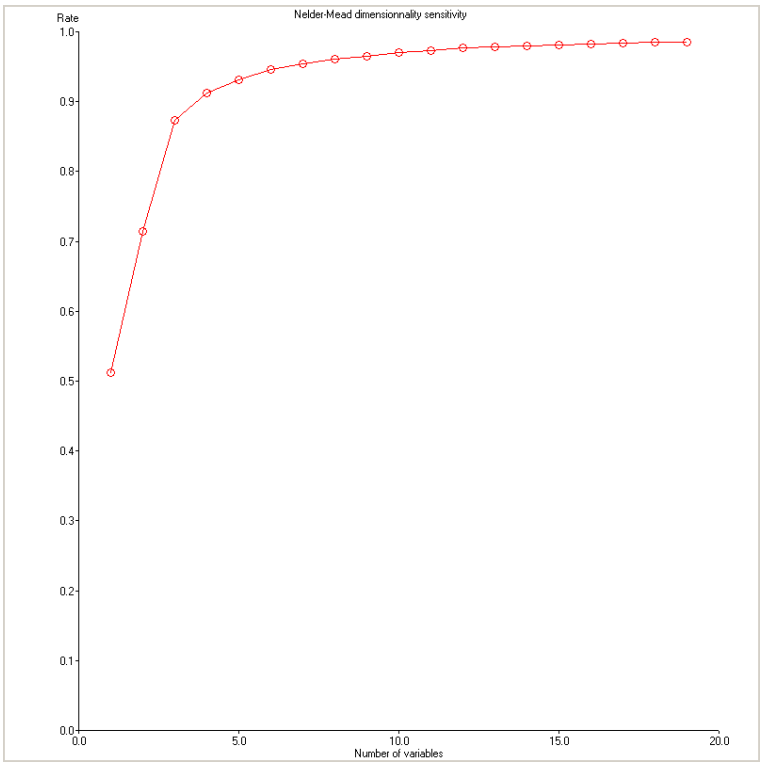

Fig. 1.23 : Nelder-Mead numerical experiment – rate of convergence depending on the number of variables

figure shows that the rate of convergence rapidly gets close to 1 when the number of variables increases. That shows that the rate of convergence is slower and slower as the number of variables increases, as explained by Han & Neumann.

1.4.3 O'Neill test cases

In this section, we present the results by O'Neill, who implemented a fortran 77 version of the Nelder-Mead algorithm [9].

The O'Neill implementation of the Nelder-Mead algorithm has the following particularities

- the initial simplex is computed from the axes and a (single) length,
- the stopping rule is based on variance (not standard deviation) of function value,
- the expansion is greedy, i.e. the expansion point is accepted if it is better than the lower point,
- an automatic restart is performed if a factorial test shows that the computed optimum is greater than a local point computed with a relative epsilon equal to 1.e-3.

The following tests are presented by O'Neill :

- Rosenbrock's parabolic valley [11]

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1.42}$$

  with starting point $(x_1, x_2) = (-1.2, 1.0)$

- Powell's quartic function [10]

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \tag{1.43}$$

  with starting point $(x_1, x_2, x_3, x_4) = (3, -1, 0, 1)$

- Fletcher and Powell's helical valley [1]

$$f(x_1, x_2, x_3) = 100\left(x_3 + 10\theta(x_1, x_2)\right)^2 + \left(\sqrt{x_1^2 + x_2^2} - 1\right)^2 + x_3^2 \tag{1.44}$$

  where

$$\begin{aligned} 2\pi\theta(x_1, x_2) &= \arctan(x_2, x_1), x_1 > 0 && (1.45)\\ &= \pi + \arctan(x_2, x_1), x_1 < 0 && (1.46)\\ & && (1.47) \end{aligned}$$

  with starting point $(x_1, x_2, x_3) = (-1, 0, 0)$. Note that since $\arctan(0/0)$ is not defined neither the function f on the line $(0, 0, x_3)$. This line is excluded by assigning a very large value to the function.

- the sum of powers

$$f(x_1, \ldots, x_{10}) = \sum_{i=1,10} x_i^4 \tag{1.48}$$

  with starting point $(x_1, \ldots, x_{10}) = (1, \ldots, 1)$

The parameters are set to

- $REQMIN = 10^{-16}$, the absolute tolerance on the variance of the function values in the simplex,
- $STEP = 1.0$, the absolute side length of the initial simplex,
- $ICOUNT$, the maximum number of function evaluations.

The table 1.24 presents the results which were computed by O'Neill compared with our software. For most experiments, the results are very close in terms of number of function evaluations. The problem #4 exhibits a completely different behavior than the results presented by O'Neill.

For us, the maximum number of function evaluations is reached (i.e. 1000 function evaluations), whereas for O'Neill, the algorithm is restarted and gives the result with 474 function evaluations. We did not find any explanation for this behavior. A possible cause of difference may be the floating point system which are different and may generate different simplices in the algorithms. Although the CPU times cannot be compared (the article is dated 1972 !), let's mention that the numerical experiment were performed by O'Neill on a ICL 4-50 where the two problem 1 and 2 were solved in 3.34 seconds and the problems 3 and 4 were solved in 22.25 seconds.

Author	Problem	Function Evaluations	No. of Restarts	Function Value	Iterations	CPU Time
O'Neill	1	148	0	3.19 e-9	?	?
Baudin	1	149	0	1.15 e-7	79	0.238579
O'Neill	2	209	0	7.35 e-8	?	?
Baudin	2	224	0	1.07 e-8	126	0.447958
O'Neill	3	250	0	5.29 e-9	?	?
Baudin	3	255	0	4.56 e-8	137	0.627493
O'Neill	4	474	1	3.80 e-7	?	?
Baudin	4	999	0	5.91 e-9	676	-

Fig. 1.24 : Numerical experiment with Nelder-Mead method on O'Neill test cases - O'Neill results and our results

1.4.4 Convergence to a non stationnary point

In this section, we analyze the Mc Kinnon counter example from [6]. We show the behavior of the Nelder-Mead simplex method for a family of examples which cause the method to converge to a non stationnary point.

Consider a simplex in two dimensions with vertices at 0 (i.e. the origin), $\mathbf{v}^{(n+1)}$ and $\mathbf{v}^{(n)}$. Assume that

$$f(0) < f(\mathbf{v}^{(n+1)}) < f(\mathbf{v}^{(n)}). \tag{1.49}$$

The centroid of the simplex is $\overline{\mathbf{v}} = \mathbf{v}^{(n+1)}/2$, the midpoint of the line joining the best and second vertex. The reflected point is then computed as

$$\mathbf{r}^{(n)} = \overline{\mathbf{v}} + \rho(\overline{\mathbf{v}} - \mathbf{v}^{(n)}) = \mathbf{v}^{(n+1)} - \mathbf{v}^{(n)} \tag{1.50}$$

Assume that the reflection point $\mathbf{r}^{(n)}$ is rejected, i.e. that $f(\mathbf{v}^{(n)}) < f(\mathbf{r}^{(n)})$. In this case, the inside contraction step is taken and the point $\mathbf{v}^{(n+2)}$ is computed using the reflection factor $-\gamma = -1/2$ so that

$$\mathbf{v}^{(n+2)} = \overline{\mathbf{v}} - \gamma(\overline{\mathbf{v}} - \mathbf{v}^{(n)}) = \frac{1}{4}\mathbf{v}^{(n+1)} - \frac{1}{2}\mathbf{v}^{(n)} \tag{1.51}$$

Assume then that the inside contraction point is accepted, i.e. $f(\mathbf{v}^{(n+2)}) < f(\mathbf{v}^{(n+1)})$. If this sequence of steps repeats, the simplices are subject to the following linear recurrence formula

$$4\mathbf{v}^{(n+2)} - \mathbf{v}^{(n+1)} + 2\mathbf{v}^{(n)} = 0 \tag{1.52}$$

Their general solutions are of the form

$$\mathbf{v}^{(n)} = \lambda_1^k a_1 + \lambda_2^k a_2 \tag{1.53}$$

where $\lambda_{ii=1,2}$ are the roots of the characteristic equation and $a_{ii=1,2} \in \mathbb{R}^n$. The characteristic equation is

$$4\lambda^2 - \lambda + 2\lambda = 0 \tag{1.54}$$

and has the roots

$$\lambda_1 = \frac{1+\sqrt{33}}{8} \approx 0.84307, \qquad \lambda_2 = \frac{1-\sqrt{33}}{8} \approx -0.59307 \tag{1.55}$$

After Mc Kinnon has presented the computation of the roots of the characteristic equation, he presents a special initial simplex for which the simplices degenerates because of repeated failure by inside contraction (RFIC in his article). Consider the initial simplex with vertices $\mathbf{v}^{(0)} = (1, 1)$ and $\mathbf{v}^{(1)} = (\lambda_1, \lambda_2)$ and 0. If follows that the particular solution for these initial conditions is $\mathbf{v}^{(n)} = (\lambda_1^n, \lambda_2^n)$.

Consider the function $f(x_1, x_2)$ given by

$$\begin{aligned} f(x_1, x_2) &= \theta\phi|x_1|^\tau + x_2 + x_2^2, \qquad x_1 \leq 0, & (1.56)\\ &= \theta x_1^\tau + x_2 + x_2^2, \qquad x_1 \geq 0. & (1.57) \end{aligned}$$

where θ and ϕ are positive constants. Note that $(0, -1)$ is a descent direction from the origin $(0, 0)$ and that f is stricly convex provided $\tau > 1$. f has continuous first derivatives if $\tau > 1$, continuous second derivatives if $\tau > 2$ and continuous third derivatives if $\tau > 3$.

Mc Kinnon computed the conditions on θ, ϕ and τ so that the function values are ordered as expected, i.e. so that the reflection step is rejected and the inside contraction is accepted. Examples of values which makes these equations hold are as follows : for $\tau = 1$, $\theta = 15$ and $\phi = 10$, for $\tau = 2$, $\theta = 6$ and $\phi = 60$ and for $\tau = 3$, $\theta = 6$ and $\phi = 400$.

We consider here the more regular case $\tau = 3$, $\theta = 6$ and $\phi = 400$, i.e. the function is defined by

$$\begin{aligned} f(x_1, x_2) &= -2400x_1^3 + x_2 + x_2^2, \qquad x_1 \leq 0, & (1.58)\\ &= 6x_1^3 + x_2 + x_2^2, \qquad x_1 \geq 0. & (1.59) \end{aligned}$$

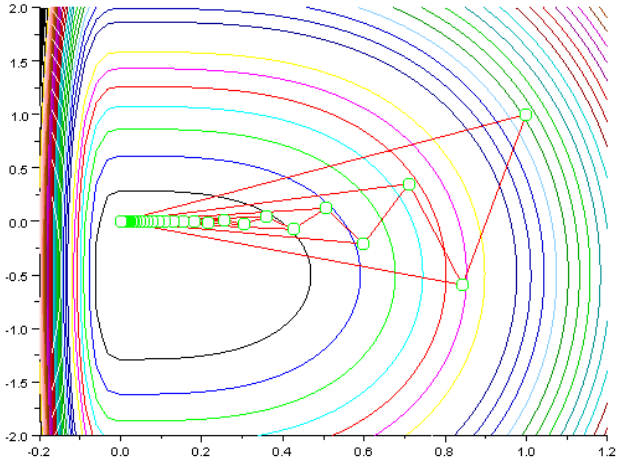

Fig. 1.25 : Nelder-Mead numerical experiment – Mc Kinnon example for convergence toward a non stationnary point

The figure 1.25 shows the contour plot of this function and the first steps of the Nelder-Mead method. The global minimum is located at $(0, -1/2)$. Notice that the simplex degenerates to the point $(0, 0)$, which is a non stationnary point.

The figure 1.26 presents the first steps of the algorithm in this numerical experiment. Because of the particular shape of the contours of the function, the reflected point is always worse that the worst vertex $\mathbf{x}_{n+1}$. This leads to the inside contraction step. The vertices constructed by Mc Kinnon are so that the situation loops without end.

1.4.5 Han counter examples

In his Phd thesis [2], Han presents two counter examples in which the Nelder-Mead algorithm degenerates by applying repeatedly the inside contraction step.

First counter example

The first counter example is based on the function

$$f(x_1, x_2) = x_1^2 + x_2(x_2 + 2)(x_2 - 0.5)(x_2 - 2) \tag{1.60}$$

This function is nonconvex, bounded below and has bounded level sets. The initial simplex is chosen as $S_0 = [(0., -1), (0, 1), (1, 0)]$. Han proves that the Nelder-Mead algorithm generates a sequence of simplices $S_k = [(0., -1), (0, 1), (\frac{1}{2^k}, 0)]$.

The figure 1.27 presents the isovalues and the simplices during the steps of the Nelder-Mead algorithm. Note that the limit simplex contains no minimizer of the function. The failure is

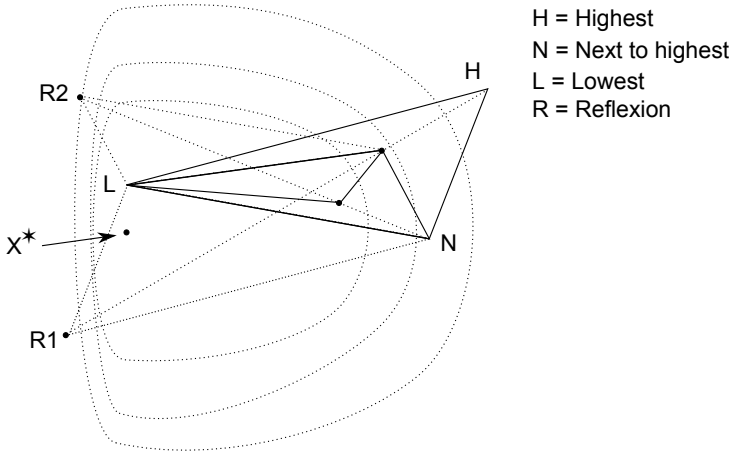

Fig. 1.26 : Nelder-Mead numerical experiment – Detail of the first steps. The simplex converges to a non stationnary point, after repeated inside contractions.

caused by repeated inside contractions.

Second counter example

The second counter example is based on the function

$$f(x_1, x_2) \quad = \quad x_1^2 + \rho(x_2) \tag{1.61}$$

where ρ is a continuous convex function with bounded level sets defined by

$$\begin{cases} \rho(x_2) = 0, & \text{if} \quad |x_2| \leq 1, \\ \rho(x_2) \geq 0, & \text{if} \quad |x_2| > 1. \end{cases} \tag{1.62}$$

The example given by Han for such a ρ function is

$$\rho(x_2) = \begin{cases} 0, & \text{if} \quad |x_2| \leq 1, \\ x_2 - 1, & \text{if} \quad x_2 > 1, \\ -x_2 - 1, & \text{if} \quad x_2 < -1. \end{cases} \tag{1.63}$$

The initial simplex is chosen as $S_0 = [(0., 1/2), (0, -1/2), (1, 0)]$. Han prooves that the Nelder-Mead algorithm generates a sequence of simplices $S_k = [(0., 1/2), (0, -1/2), (\frac{1}{2^k}, 0)]$.

The figure 1.28 presents the isovalues and the simplices during the steps of the Nelder-Mead algorithm. The failure is caused by repeated inside contractions.

These two examples of non convergence show that the Nelder-Mead method may unreliable. They also reveal that the Nelder-Mead method can generate simplices which collapse into a degenerate simplex, by applying repeated inside contractions.

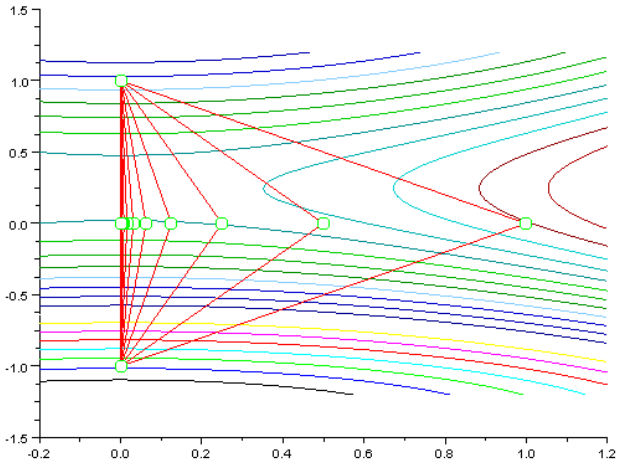

Fig. 1.27 : Nelder-Mead numerical experiment – Han example #1 for convergence toward a non stationnary point

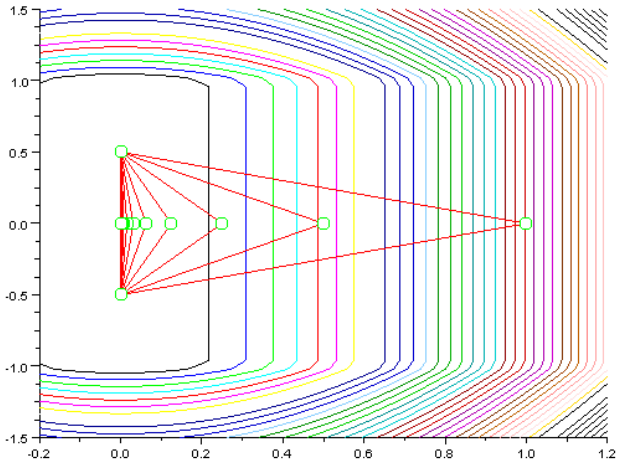

Fig. 1.28 : Nelder-Mead numerical experiment – Han example #2 for convergence toward a non stationnary point

1.4.6 Torczon's numerical experiments

In her Phd Thesis [12], Virginia Torczon presents the multi-directional direct search algorithm. In order to analyze the performances of her new algorithm, she presents some interesting numerical experiments with the Nelder-Mead algorithm. These numerical experiments are based on the collection of test problems [7], published in the ACM by Moré, Garbow and Hillstrom in 1981. These test problems are associated with varying number of variables. In her Phd, Torczon presents numerical experiments with n from 8 to 40. The stopping rule is based on the relative size of the simplex. The angle between the descent direction (given by the worst point and the centroid), and the gradient of the function is computed when the algorithm is stopped. Torczon shows that, when the tolerance on the relative simplex size is decreased, the angle converges toward 90°. This fact is observed even for moderate number of dimensions.

In this section, we try to reproduce Torczon numerical experiments.

All experiments are associated with the following sum of squares cost function

$$f(\mathbf{x}) = \sum_{i=1,m} f_i(\mathbf{x})^2, \tag{1.64}$$

where $m \geq 1$ is the number of functions f_i in the problem.

The stopping criteria is based on the relative size of the simplex and is the following

$$\frac{1}{\Delta} \max_{i=2,n+1} \|\mathbf{v}_i - \mathbf{v}_1\| \leq \epsilon, \tag{1.65}$$

where $\Delta = \max(1, \|\mathbf{v}_1\|)$. Decreasing the value of ϵ allows to get smaller simplex sizes.

Penalty #1

The first test function is the *Penalty #1* function :

$$f_i(\mathbf{x}) = \sqrt{1.e-5}(x_i - 1), \qquad i = 1, n \tag{1.66}$$

$$f_{n+1} = -\frac{1}{4} + \sum_{j=1,n} x_j^2. \tag{1.67}$$

The initial guess is given by $\mathbf{x}_0 = ((\mathbf{x}_0)_1, (\mathbf{x}_0)_2, \ldots, (\mathbf{x}_0)_n)^T$ and $(\mathbf{x}_0)_j = j$ for $j = 1, n$.

The problem given by Moré, Garbow and Hillstrom in [7] is associated with the size $n = 4$. The value of the cost function at the initial guess $\mathbf{x}_0 = (1, 2, 3, 4)^T$ is $f(\mathbf{x}_0) = 885.063$. The value of the function at the optimum is given in [7] as $f(\mathbf{x}^\star) = 2.24997d - 5$.

Torzcon shows an experiment with the Penalty #1 test case and $n = 8$. For this particular case, the initial function value is $f(\mathbf{x}_0) = 4.151406.10^4$. The figure 1.29 presents the results of these experiments. The number of function evaluations is not the same so that we can conclude

Author	Step Tolerance	$f(\mathbf{v}_1^\star)$	Function Evaluations	Angle (°)
Torzcon	1.e-1	7.0355e-5	1605	89.396677792198
Baudin	1.e-1	8.2272e-5	530	87.7654
Torzcon	1.e-2	6.2912e-5	1605	89.935373548613
Baudin	1.e-2	7.4854e-5	1873	89.9253
Torzcon	1.e-3	6.2912e-5	3600	89.994626919197
Baudin	1.e-3	7.4815e-5	2135	90.0001
Torzcon	1.e-4	6.2912e-5	3670	89.999288284747
Baudin	1.e-4	7.481546e-5	2196	89.9991
Torzcon	1.e-5	6.2912e-5	3750	89.999931862232
Baudin	1.e-5	7.427212e-5	4626	89.999990

Fig. 1.29 : Numerical experiment with Nelder-Mead method on Torczon test cases - Torczon results and our results

that the algorithm may be different variants of the Nelder-Mead algorithms. We were not able to explain why the number of function evaluations is so different.

The figure 1.30 presents the angle between the gradient of the function $-\mathbf{g}_k$ and the search direction $\mathbf{x}_c - \mathbf{x}_h$, where $\mathbf{x}_c$ is the centroid of the best vertices and $\mathbf{x}_h$ is the worst (or high) vertex.

The numerical experiment shows that the conditioning of the matrix of simplex direction has an increasing condition number. This corresponds to the fact that the simplex is increasingly distorted.

1.5 Conclusion

The main advantage of the Nelder-Mead algorithm over Spendley et al. algorithm is that the shape of the simplex is dynamically updated. That allows to get a reasonably fast convergence rate on badly scaled quadratics, or more generally when the cost function is made of a sharp valley. Nevertheless, the behavior of the algorithm when the dimension of the problem increases is disappointing : the more there are variables, the more the algorithm is slow. In general, it is expected that the number of function evaluations is roughly equal to $100n$.

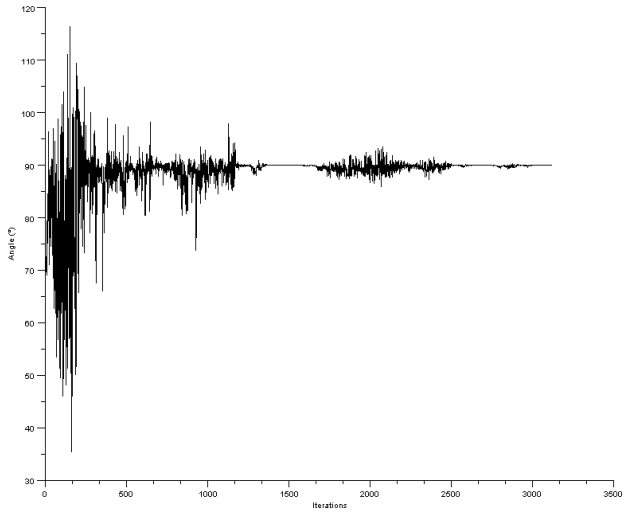

Fig. 1.30 : Nelder-Mead numerical experiment – Penalty #1 function – One can see that the angle between the gradient and the search direction is very close to 90°, especially for large number of iterations.

Bibliography

[1] R. Fletcher and M. J. D. Powell. A Rapidly Convergent Descent Method for Minimization. *The Computer Journal*, 6(2):163–168, 1963.

[2] Lixing Han. *Algorithms in Unconstrained Optimization*. Ph.D., The University of Connecticut, 2000.

[3] Lixing Han and Michael Neumann. Effect of dimensionality on the nelder-mead simplex method. *Optimization Methods and Software*, 21(1):1–16, 2006.

[4] C. T. Kelley. *Iterative Methods for Optimization*, volume 19. SIAM Frontiers in Applied Mathematics, 1999. http://www.siam.org/books/textbooks/fr18_book.pdf.

[5] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998. http://link.aip.org/link/?SJE/9/112/1.

[6] K. I. M. McKinnon. Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM J. on Optimization*, 9(1):148–158, 1998.

[7] J. J. Moré, Burton S. Garbow, and Kenneth E. Hillstrom. Algorithm 566: Fortran subroutines for testing unconstrained optimization software [c5], [e4]. *ACM Trans. Math. Softw.*, 7(1):136–140, 1981.

[8] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[9] R. O'Neill. Algorithm AS47 - Function minimization using a simplex procedure. *Applied Statistics*, 20(3):338–346, 1971.

[10] M. J. D. Powell. An Iterative Method for Finding Stationary Values of a Function of Several Variables. *The Computer Journal*, 5(2):147–151, 1962.

[11] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, March 1960.

[12] Virginia Joanne Torczon. Multi-directional search: A direct search algorithm for parallel machines. Technical report, Rice University, 1989.