Nelder-Mead Toolbox Manual – Simplex Theory –

Version 0.2
September 2009

Michaël BAUDIN

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

Simplex theory

In this section, we present the various definitions connected to simplex algorithms. We introduce several methods to measure the size of a simplex, including the oriented length. We present several methods to compute an initial simplex, that is, the regular simplex used by Spendley et al., the axis-by-axis simplex, Pfeffer's simplex and the randomized bounds simplex.

1.1 The simplex

Definition 1.1.1 *(*Simplex*) A* simplex *S in $\mathbb{R}^n$ is the convex hull of $n+1$ vertices, that is, a simplex $S = \{\mathbf{v}_i\}_{i=1,n+1}$ is defined by its $n+1$ vertices $\mathbf{v}_i \in \mathbb{R}^n$ for $i = 1, n+1$.*

The j-th coordinate of the i-th vertex $\mathbf{v}_i \in \mathbb{R}^n$ is denoted by $(\mathbf{v}_i)_j \in \mathbb{R}$.

Box extended the Nelder-Mead algorithm to handle bound and non linear constraints [1]. To be able to manage difficult cases, he uses a *complex* made of $m \geq n+1$ vertices.

Definition 1.1.2 *(*Complex*) A* complex *S in $\mathbb{R}^n$ is a set of $m \geq n+1$ vertices, that is, a simplex $S = \{\mathbf{v}_i\}_{i=1,m}$ is defined by its m vertices $\mathbf{v}_i \in \mathbb{R}^n$ for $i = 1, m$.*

In this chapter, we will state clearly when the definition and results can only be applied to a simplex or to a more general a complex.

We assume that we are given a cost function $f : \mathbb{R}^n \to \mathbb{R}$. Each vertex $\mathbf{v}_i$ is associated with a function value

$$f_i = f(\mathbf{v}_i) \text{ for } i = 1, m. \qquad (1.1)$$

For any complex, the vertices can be sorted by increasing function values

$$f_1 \leq f_2 \leq \ldots \leq f_n \leq f_m. \qquad (1.2)$$

The sorting order is not precisely defined neither in Spendley's et al paper [11] nor in Nelder and Mead's [8]. In [6], the sorting rules are defined precisely to be able to state a theoretical convergence result. In practical implementations, though, the ordering rules have no measurable influence.

1.2 The size of the complex

Several methods are available to compute the size of a complex.

In this section, we use the euclidian norm $\|.\|_2$ the defined by

$$\|\mathbf{v}\|_2 = \sum_{j=1,n} (v_j)^2. \tag{1.3}$$

Definition 1.2.1 *(*Diameter*) The simplex diameter $diam(S)$ is defined by*

$$diam(S) = \max_{i,j=1,m} \|\mathbf{v}_i - \mathbf{v}_j\|_2. \tag{1.4}$$

In practical implementations, computing the diameter requires two nested loops over the vertices of the simplex, i.e. requires m^2 operations. This is why authors generally prefer to use lengths which are less expensive to compute.

Definition 1.2.2 *(*Oriented length*) The two oriented lengths $\sigma_-(S)$ and $\sigma_+(S)$ are defined by*

$$\sigma_+(S) = \max_{i=2,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2 \qquad \textit{and} \qquad \sigma_-(S) = \min_{i=2,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2. \tag{1.5}$$

Proposition 1.2.3 *The diameter and the maximum oriented length satisfy the following inequalities*

$$\sigma_+(S) \leq diam(S) \leq 2\sigma_+(S). \tag{1.6}$$

Proof We begin by proving that

$$\sigma_+(S) \leq diam(S). \tag{1.7}$$

This is directly implied by the inequality

$$\begin{aligned} \max_{i=2,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2 &\leq \max_{i=1,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2 && (1.8)\\ &\leq \max_{i,j=1,m} \|\mathbf{v}_i - \mathbf{v}_j\|_2, && (1.9)\end{aligned}$$

which concludes the first part of the proof. We shall now proove the inequality

$$diam(S) \leq 2\sigma_+(S). \tag{1.10}$$

We decompose the difference $\mathbf{v}_i - \mathbf{v}_j$ into

$$\mathbf{v}_i - \mathbf{v}_j = (\mathbf{v}_i - \mathbf{v}_1) + (\mathbf{v}_1 - \mathbf{v}_j). \tag{1.11}$$

Hence,

$$\|\mathbf{v}_i - \mathbf{v}_j\|_2 \leq \|\mathbf{v}_i - \mathbf{v}_1\|_2 + \|\mathbf{v}_1 - \mathbf{v}_j\|_2. \tag{1.12}$$

We take the maximum over i and j, which leads to

$$\max_{i,j=1,m} \|\mathbf{v}_i - \mathbf{v}_j\|_2 \leq \max_{i=1,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2 + \max_{j=1,m} \|\mathbf{v}_1 - \mathbf{v}_j\|_2 \quad (1.13)$$

$$\leq 2 \max_{i=1,m} \|\mathbf{v}_i - \mathbf{v}_1\|_2. \quad (1.14)$$

With the definitions of the diameter and the oriented length, this immediately prooves the inequality 2.10. ∎

In Nash's book [7], the size of the simplex $s_N(S)$ is measured based on the 1-norm and is defined by

$$s_N(S) = \sum_{i=2,m} \|\mathbf{v}_i - \mathbf{v}_1\|_1 \quad (1.15)$$

where the 1-norm is defined by

$$\|\mathbf{v}_i\|_1 = \sum_{j=1,n} |(\mathbf{v}_i)_j|. \quad (1.16)$$

The *optimsimplex_size* function provides all these size algorithms. In the following example, we create an axis-by-axis simplex with length unity and compute its length by several methods.

```
xx0 = [0.0 0.0];
si = optimsimplex_new ( "axes" , x0 );
methodlist = [
"sigmaplus"
"sigmaminus"
"Nash"
"diameter"
];
for i = 1:size(methodlist,"*")
  m = methodlist ( i );
  ss = optimsimplex_size ( si , m );
  mprintf ( "%s: %f\n", m , ss );
end
optimsimplex_destroy(si)
```

The previous script produces the following output.

```
sigmaplus: 1.000000
sigmaminus: 1.000000
Nash: 2.000000
diameter: 1.414214
```

We check that the diameter is equal to $diam(S) = \sqrt{2}$. We see that inequality 2.6 is satisfied since $\sigma_+(S) = 1 \leq \sqrt{2} \leq 2 = 2\sigma_+(S)$.

1.3 The initial simplex

While most of the theory can be developed without being very specific about the initial simplex, it plays a very important role in practice. All approaches are based on the initial guess $\mathbf{x}_0 \in \mathbb{R}^n$ and create a geometric shape based on this point.

In this section, we present the various approach to design the initial simplex. In the first part, we emphasize the importance of the initial simplex in optimization algorithms. Then we present the regular simplex by Spendley et al., the axis-by-axis simplex, the randomized bounds approach by Box and Pfeffer's simplex.

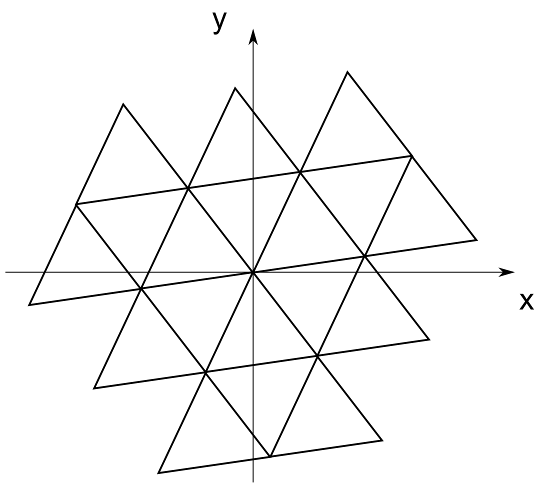

Fig. 1.1 : Typical pattern with fixed-shape Spendley's et al algorithm

1.3.1 Importance of the initial simplex

The initial simplex is particularily important in the case of Spendley's et al method, where the shape of the simplex is fixed during the iterations. Therefore, the algorithm can only go through points which are on the pattern defined by the initial simplex. The pattern presented in figure 2.1 is typical a fixed-shape simplex algorithm (see [12], chapter 3, for other patterns of a direct search method). If, by chance, the pattern is so that the optimum is close to one point defined by the pattern, the number of iteration may be small. On the contrary, the number of iterations may be large if the pattern does not come close to the optimum.

The variable-shape simplex algorithm designed by Nelder and Mead is also very sensitive to the initial simplex. One of the problems is that the initial simplex should be consistently scaled with respect to the unknown **x**. In "An investigation into the efficiency of variants on the simplex method" [9], Parkinson and Hutchinson explored several improvements of Nelder and Mead's algorithm. First, they investigate the sensitivity of the algorithm to the initial simplex. Two parameters were investigated, that is, the initial length and the orientation of the simplex. The conclusion of their study with respect to the initial simplex is the following. "The orientation of the initial simplex has a significant effect on efficiency, but the relationship can be too sensitive for an automatic predictor to provide sufficient accuracy at this time."

Since no initial simplex clearly improves on the others, in practice, it may be convenient to try different approaches.

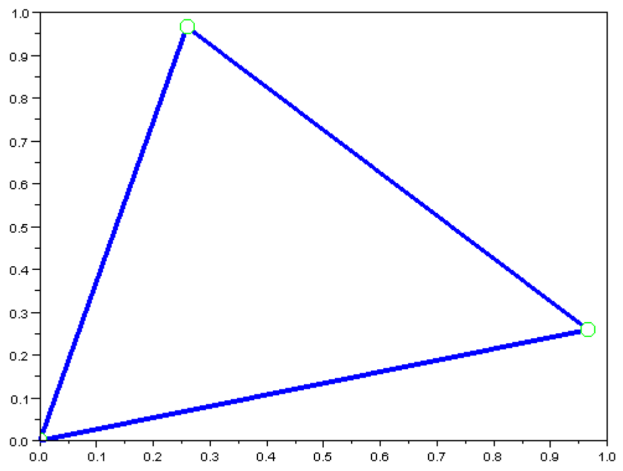

Fig. 1.2 : Regular simplex in 2 dimensions

1.3.2 Spendley's et al regular simplex

In their paper [11], Spendley et al. use a regular simplex with given size $\ell > 0$. We define the parameters $p, q > 0$ as

$$p = \frac{1}{n\sqrt{2}}\left(n - 1 + \sqrt{n+1}\right), \tag{1.17}$$

$$q = \frac{1}{n\sqrt{2}}\left(\sqrt{n+1} - 1\right). \tag{1.18}$$

We can now define the vertices of the simplex $S = \{\mathbf{x}_i\}_{i=1,n+1}$. The first vertex of the simplex is the initial guess

$$\mathbf{v}_1 = \mathbf{x}_0. \tag{1.19}$$

The other vertices are defined by

$$(\mathbf{v}_i)_j = \begin{cases} (\mathbf{x}_0)_j + \ell p, & \text{if } j = i-1, \\ (\mathbf{x}_0)_j + \ell q, & \text{if } j \neq i-1, \end{cases} \tag{1.20}$$

for vertices $i = 2, n+1$ and components $j = 1, n$, where $\ell \in \mathbb{R}$ is the length of the simplex and satisfies $\ell > 0$. Notice that this length is the same for all the edges which keeps the simplex regular.

The regular simplex is presented in figure 2.2.

In the following Scilab session, we define a regular simplex with the *optimsimplex_new* function.

```
x0 = [0.0 0.0];
si = optimsimplex_new ( "spendley" , x0 );
methodlist = [
"sigmaplus"
"sigmaminus"
"diameter"
];
for i = 1:size(methodlist ,"*")
  m = methodlist ( i );
  ss = optimsimplex_size ( si , m );
  mprintf ( "%s:␣%f\n", m , ss );
end
optimsimplex_destroy(si);
```

The previous script produces the following output.

```
sigmaplus: 1.000000
sigmaminus: 1.000000
diameter: 1.000000
```

We check that the three sizes $diam(S)$, $\sigma_+(S)$ and $\sigma_-(S)$ are equal, as expected from a regular simplex.

1.3.3 Axis-by-axis simplex

A very efficient and simple approach leads to an axis-by-axis simplex. This simplex depends on a vector of positive lengths $\mathbf{l} \in \mathbb{R}^n$. The first vertex of the simplex is the initial guess

$$\mathbf{v}_1 = \mathbf{x}_0. \tag{1.21}$$

The other vertices are defined by

$$(\mathbf{v}_i)_j = \begin{cases} (\mathbf{x}_0)_j + \ell_j, & \text{if } j = i-1, \\ (\mathbf{x}_0)_j, & \text{if } j \neq i-1, \end{cases} \tag{1.22}$$

for vertices $i = 2, n+1$ and components $j = 1, n$.

This type of simplex is presented in figure 2.3, where $\ell_1 = 1$ and $\ell_2 = 2$. The axis-by-axis simplex is used in the Nelder-Mead algorithm provided in Numerical Recipes in C [10]. As stated in [10], the length vector $\mathbf{l}$ can be used as a guess for the characteristic length scale of the problem.

1.3.4 Randomized bounds

Assume that the variable $\mathbf{x} \in \mathbb{R}^n$ is bounded so that

$$m_j \leq x_j \leq M_j, \tag{1.23}$$

for $j = 1, n$, where $m_j, M_j \in \mathbb{R}$ are minimum and maximum bounds and $m_j \leq M_j$. A method suggested by Box in [1] is based on the use of pseudo-random numbers. Let $\{\theta_{i,j}\}_{i=1,n+1,j=1,n} \in [0,1]$ be a sequence of random numbers uniform in the interval $[0,1]$. The first vertex of the simplex is the initial guess

$$\mathbf{v}_1 = \mathbf{x}_0. \tag{1.24}$$

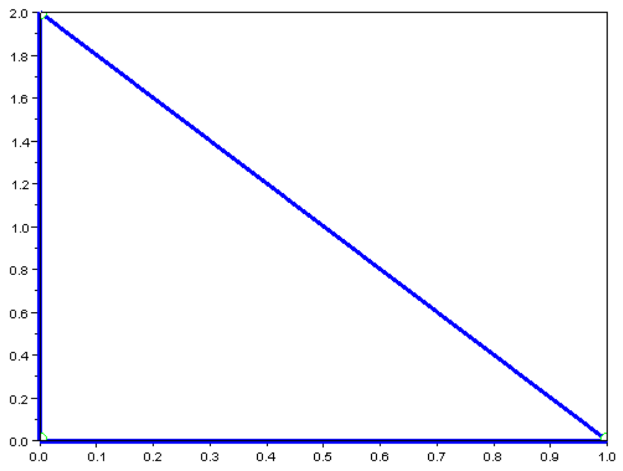

Fig. 1.3 : Axis-based simplex in 2 dimensions – Notice that the length along the x axis is 1 while the length along the y axis is 2.

The other vertices are defined by

$$(\mathbf{v}_i)_j \quad = \quad m_j + \theta_{i,j}(M_j - m_j), \tag{1.25}$$

for vertices $i = 2, n+1$ and components $j = 1, n$.

1.3.5 Pfeffer's method

This initial simplex is used in the function *fminsearch* and presented in [2]. According to [2], this simplex is due to L. Pfeffer at Stanford. The goal of this method is to scale the initial simplex with respect to the characteristic lengths of the problem. This allows, for example, to manage cases where $x_1 \approx 1$ and $x_2 \approx 10^5$. As we are going to see, the scaling is defined with respect to the initial guess $\mathbf{x}_0$, with an axis-by-axis method.

The method proceeds by defining $\delta_u, \delta_z > 0$, where δ_u is used for usual components of $\mathbf{x}_0$ and δ_z is used for the case where one component of $\mathbf{x}_0$ is zero. The default values for δ_u and δ_z are

$$\delta_u = 0.05 \qquad \text{and} \qquad \delta_z = 0.0075. \tag{1.26}$$

The first vertex of the simplex is the initial guess

$$\mathbf{v}_1 \quad = \quad \mathbf{x}_0. \tag{1.27}$$

The other vertices are defined by

$$(\mathbf{v}_i)_j \quad = \quad \begin{cases} (\mathbf{x}_0)_j + \delta_u(\mathbf{x}_0)_j, & \text{if } j = i-1 \text{ and } (\mathbf{x}_0)_{j-1} \neq 0, \\ \delta_z, & \text{if } j = i-1 \text{ and } (\mathbf{x}_0)_{j-1} = 0, \\ (\mathbf{x}_0)_j, & \text{if } j \neq i-1, \end{cases} \tag{1.28}$$

for vertices $i = 2, n + 1$ and components $j = 1, n$.

1.4 The simplex gradient

In this section, we present the simplex gradient and proove that this gradient is an approximation of the gradient of the objective function, provided that the condition of the matrix of simplex directions. We derive the forward simplex gradient.

1.4.1 Matrix of simplex directions

We consider here simplices made of $m = n + 1$ vertices only. This allows to define the matrix of simplex directions as presented in the following definition.

Definition 1.4.1 *(*Matrix of simplex directions*) Assume that S is a set of $m = n + 1$ vertices. The $n \times n$ matrix of simplex directions $D(S)$ is defined by*

$$D(S) = (\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_2 - \mathbf{v}_1, \ldots, \mathbf{v}_{n+1} - \mathbf{v}_1). \tag{1.29}$$

We define by $\{\mathbf{d}_i\}_{i=1,n}$ the columns of the $n \times n$ matrix $D(S)$, i.e.

$$D(S) = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n). \tag{1.30}$$

We say that the simplex S is nonsingular *if the matrix $D(S)$ is* nonsingular*. We define the* simplex condition *as the l^2 condition number of the matrix of simplex directions $\kappa(D(S))$.*

The directions $\mathbf{d}_i$ can be seen as *offsets*, leading from the first vertex to each vertex $\mathbf{v}_i$, i.e.

$$\mathbf{v}_i = \mathbf{v}_1 + \mathbf{d}_1, \text{ for } i = 1, n. \tag{1.31}$$

Example (*A non degenerate simplex*) Consider the axis-by-axis simplex, with first vertex at origin and lengths unity. The vertices are defined by

$$\mathbf{v}_1 = (0,0)^T, \qquad \mathbf{v}_2 = (1,0)^T, \qquad \mathbf{v}_3 = (0,1)^T, \tag{1.32}$$

so that the matrix of simplex directions is given by

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.33}$$

Such a matrix has a unity condition number.

The following Scilab session uses the *optimsimplex* component to generate a axis-by-axis simplex and computes the matrix of directions with the *optimsimplex_dirmat* function.

```
x0 = [0.0 0.0];
si = optimsimplex_new ( "axes" , x0 );
D = optimsimplex_dirmat ( si )
k = cond(D)
optimsimplex_destroy(si)
```

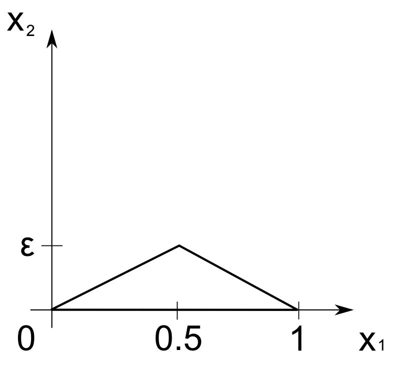

Fig. 1.4 : A ”flat” simplex in 2 dimensions

The previous script produces the following output.

```
-->D = optimsimplex_dirmat ( si )
 D  =
     1.     0.
     0.     1.
-->k = cond(D)
 k  =
     1.
```

We check that an axis-by-axis simplex has a very low condition number. □

Example (*A degenerate simplex*) In this example, we show that a flat simplex is associated with a high condition number. Consider a flat simplex, defined by its vertices:

$$\mathbf{v}_1 = (0,0)^T, \qquad \mathbf{v}_2 = (1,0)^T, \qquad \mathbf{v}_3 = (1/2,\epsilon)^T, \tag{1.34}$$

with $\epsilon = 10^{-10}$. This simplex is presented in figure 2.4.

```
coords = [
0.0  0.0
1.0  0.0
0.5  1.e-10
];
si = optimsimplex_new ( coords );
D = optimsimplex_dirmat ( si )
k = cond(D)
optimsimplex_destroy(si);
```

The previous script produces the following output.

```
-->D = optimsimplex_dirmat ( si )
 D  =
     1.     0.5
     0.     1.000D-10
-->k = cond(D)
 k  =
     1.250D+10
```

We see that a flat simplex is associated with a high condition number. Indeed, a low condition number of the matrix of directions is an indication of the non-degeneracy of the simplex. □

There is a close relationship between the oriented length $\sigma_+(S)$ and the l^2 norm of the matrix of directions $D(S)$ as prooved in the following proposition.

Proposition 1.4.2 *Let S be a simplex and consider the euclidian norm $\|.\|$. Therefore,*

$$\|\mathbf{d}_i\| \leq \sigma_+(S) \leq \|D\|, \tag{1.35}$$

for all $i = 1, \ldots, n$.

Proof It is easy to prove that

$$\|\mathbf{d}_i\| \leq \sigma_+(S). \tag{1.36}$$

Indeed, the definition of the oriented length $\sigma_+(S)$ in the case where there are $n+1$ vertices is

$$\begin{aligned} \sigma_+(S) &= \max_{i=2,n+1} \|\mathbf{v}_i - \mathbf{v}_1\|_2 && (1.37) \\ &= \max_{i=1,n} \|\mathbf{d}_i\|_2, && (1.38) \end{aligned}$$

which concludes the first part of the proof.

We shall now proove that

$$\sigma_+(S) \leq \|D\|. \tag{1.39}$$

The euclidian norm is so that ([3], section 2.3.1, "Definitions"),

$$\|D\mathbf{x}\| \leq \|D\|\|\mathbf{x}\|, \tag{1.40}$$

for any vector $\mathbf{x} \in \mathbb{R}^n$. We choose the specific vector $\mathbf{x}$ which has zeros components, except for the i-th row, which is equal to 1, i.e. $\mathbf{x} = (0, \ldots, 0, 1, 0, \ldots, 0)^T$. With this particular choice of $\mathbf{x}$ we have the properties $D\mathbf{x} = \mathbf{d}_i$ and $\|\mathbf{x}\| = 1$, so that the previous inequality becomes

$$\|\mathbf{d}_i\| \leq \|D\|, \tag{1.41}$$

for all $i = 1, \ldots, n$. We can now take the maximum of the left hand-size of 2.41 and get the oriented length $\sigma_+(S)$, which concludes the proof. ■

Example In the following Scilab session, we define a new simplex by its coordinates, so that the matrix of directions is not symetric and that the edges do not have unit lengths.

```
coords = [
0.0 0.0
1.0 0.5
1.0 2.0
];
si = optimsimplex_new ( coords );
D = optimsimplex_dirmat ( si )
for i=1:2
  nd = norm(D(1:2,i),2);
  mprintf( "||d_%d||=%f\n",i,nd)
end
ss = optimsimplex_size ( si , "sigmaplus" );
mprintf( "sigma_+(S)=%f\n",ss);
normmatrix = norm(D);
mprintf( "||D||=%f\n",normmatrix);
optimsimplex_destroy(si);
```

The previous script produces the following output.

```
||d_1||=1.118034
||d_2||=2.236068
sigma_+(S)=2.236068
||D||=2.422078
```

This result is consistent with the inequality 2.35. □

1.4.2 Taylor's formula

The simplex gradient proposition that we shall proove in the next section assumes that the gradient $\mathbf{g}$ of the function f satisfies a Lipshitz condition. The following proposition presents a result satisfied by such functions. In order to simplify the notations, we denote by $\|.\|$ the euclidian norm.

Proposition 1.4.3 *Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable and assume that its gradient $\mathbf{g}$ is defined and continuous. Let $\mathbf{x} \in \mathbb{R}^n$ be a given point and $\mathbf{p} \in \mathbb{R}^n$ a vector. Assume that the gradient $\mathbf{g}$ is Lipshitz continuous in a neighbourhood of $\mathbf{x}$ and $\mathbf{x}+\mathbf{p}$ with Lipshitz constant L. Then*

$$|f(\mathbf{x}+\mathbf{p}) - f(\mathbf{x}) - \mathbf{p}^T\mathbf{g}(\mathbf{x})| \leq \frac{1}{2}L\|\mathbf{p}\|^2. \tag{1.42}$$

Proof We can write Taylor's expansion of f in a neighbourhood of $\mathbf{x}$

$$f(\mathbf{x}+\mathbf{p}) = f(\mathbf{x}) + \int_0^1 \mathbf{p}^T\mathbf{g}(\mathbf{x}+t\mathbf{p})dt. \tag{1.43}$$

By definition of the Lipshitz condition on $\mathbf{g}$, we have

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \tag{1.44}$$

for $\mathbf{x}$ and $\mathbf{y}$ in that neighbourhood. Assume that $t \in [0,1]$ and use the particular point $\mathbf{y} = \mathbf{x}+t\mathbf{p}$. We get

$$\|\mathbf{g}(\mathbf{x}+t\mathbf{p}) - \mathbf{g}(\mathbf{x})\| \leq tL\|\mathbf{p}\|. \tag{1.45}$$

We now use equality 2.43, substract the term $\mathbf{p}^T\mathbf{g}(\mathbf{x})$ and get

$$f(\mathbf{x}+\mathbf{p}) - f(\mathbf{x}) - \mathbf{p}^T\mathbf{g}(\mathbf{x}) = \int_0^1 \mathbf{p}^T\left(\mathbf{g}(\mathbf{x}+t\mathbf{p}) - \mathbf{g}(\mathbf{x})\right)dt. \tag{1.46}$$

Therefore,

$$\begin{aligned} \left|f(\mathbf{x}+\mathbf{p}) - f(\mathbf{x}) - \mathbf{p}^T\mathbf{g}(\mathbf{x})\right| &= \left|\int_0^1 \mathbf{p}^T\left(\mathbf{g}(\mathbf{x}+t\mathbf{p}) - \mathbf{g}(\mathbf{x})\right)dt\right| && (1.47) \\ &\leq \int_0^1 \|\mathbf{p}\|\,\|\mathbf{g}(\mathbf{x}+t\mathbf{p}) - \mathbf{g}(\mathbf{x})\|\,dt && (1.48) \end{aligned}$$

We plug 2.45 into the previous equality and get

$$\begin{aligned} \left|f(\mathbf{x}+\mathbf{p}) - f(\mathbf{x}) - \mathbf{p}^T\mathbf{g}(\mathbf{x})\right| &\leq \int_0^1 Lt\|\mathbf{p}\|^2dt && (1.49) \\ &\leq \frac{1}{2}L\|\mathbf{p}\|^2, && (1.50) \end{aligned}$$

which concludes the proof. ∎

1.4.3 Forward difference simplex gradient

Finite difference formulas are a common tool to compute the numerical derivative of a function. In this section, we introduce the simplex gradient, which allows to compute an approximation of the gradient of the cost function. As we are going to see, this approximation is more accurate when the simplex has a low condition number.

We denote by $\delta(S)$ the vector of objective function differences

$$\delta(S) = (f(\mathbf{v}_2) - f(\mathbf{v}_1), f(\mathbf{v}_3) - f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{n+1}) - f(\mathbf{v}_1))^T. \tag{1.51}$$

As with classical finite difference formulas, the vector of function can be used to compute the simplex gradient.

Definition 1.4.4 *(*Simplex gradient*) Let S be a non singular simplex. The* simplex gradient *$\overline{\mathbf{g}}(S)$ is the unique solution of the linear system of equations*

$$D(S)^T \overline{\mathbf{g}}(S) = \delta(S). \tag{1.52}$$

By hypothesis, the simplex S is nonsingular so that the linear system of equations has a unique solution, which is equal to

$$\overline{\mathbf{g}}(S) = (D(S)^T)^{-1}\delta(S). \tag{1.53}$$

By hypothesis, the matrix $D(S)$ is non singular, therefore the transpose of the inverse is equal to the inverse of the transpose ([3], section 2.1.3, "Matrix Inverse"), i.e. $(D(S)^T)^{-1} = (D(S)^{-1})^T$. We denote by $D(S)^{-T}$ the inverse of the transpose so that the previous equality becomes

$$\overline{\mathbf{g}}(S) = D(S)^{-T}\delta(S). \tag{1.54}$$

In practice, the matrix of simplex direction is not inverted and the solution of 2.52 is computed directly, using classical linear algebra libraries, like Lapack for example.

The simplex gradient is an approximation of the gradient $\mathbf{g}$ of the function f, as presented in the following proposition.

Proposition 1.4.5 *Let S be a simplex. Let the gradient $\mathbf{g}$ be Lipshitz continuous in a neighbourhood of S with Lipshitz constant L. Consider the euclidian norm $\|.\|$. Then, there is $K > 0$, depending only on L such that*

$$\|\mathbf{g}(\mathbf{v}_1) - \overline{\mathbf{g}}(S)\|_2 \leq K\kappa(S)\sigma_+(S). \tag{1.55}$$

Proof We can write the difference between the simplex gradient and the gradient in the following form

$$\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1) = D(S)^{-T}\left(D(S)^T\overline{\mathbf{g}}(S) - D(S)^T\mathbf{g}(\mathbf{v}_1)\right). \tag{1.56}$$

We now plug the simplex gradient definition 2.52 into the previous equality and get

$$\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1) = D(S)^{-T} \left(\delta(S) - D(S)^T \mathbf{g}(\mathbf{v}_1) \right). \tag{1.57}$$

The fact that the euclidian norm $\|.\|$ satisfies the inequality

$$\|AB\| \leq \|A\| \|B\|, \tag{1.58}$$

for any matrices A and B with suitable number of rows and columns ([3], section 2.3, ”Matrix Norms”) plays an important role in the results that we are going to derive. Indeed, we can compute the euclidian norm of both sides of equation 2.57 and get

$$\|\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1)\| = \left\| D(S)^{-T} \left(\delta(S) - D(S)^T \mathbf{g}(\mathbf{v}_1) \right) \right\|. \tag{1.59}$$

Therefore,

$$\|\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1)\| \leq \left\| D(S)^{-T} \right\| \left\| \delta(S) - D(S)^T \mathbf{g}(\mathbf{v}_1) \right\|. \tag{1.60}$$

The suite of the proof is based on the computation of the right-hand side of equation 2.60, that is, the computation of the norm of the vector $\delta(S) - D(S)^T \mathbf{g}(\mathbf{v}_1)$.

By hypothesis, the gradient $\mathbf{g}$ is Lipshitz continuous in a neighbourhood of S. By proposition 2.4.3, we have

$$\left| f(\mathbf{v}_1 + \mathbf{d}_i) - f(\mathbf{v}_1) - \mathbf{d}_i^T \mathbf{g}(\mathbf{v}_1) \right| \leq \frac{1}{2} L \|\mathbf{d}_i\|^2, \tag{1.61}$$

for $i = 1, n$. By definition of the direction $\mathbf{d}_i$, we have $\mathbf{v}_1 + \mathbf{d}_i = \mathbf{v}_i$ for $i = 1, n$. By proposition 2.4.2, we have $\|\mathbf{d}_j\| \leq \sigma_+(S)$ for all $j = 1, n$. Hence,

$$\left| f(\mathbf{v}_i) - f(\mathbf{v}_1) - \mathbf{d}_i^T \mathbf{g}(\mathbf{v}_1) \right| \leq \frac{1}{2} L \sigma_+(S)^2, \tag{1.62}$$

We can use this to compute the euclidian norm of the vector $\delta(S) - D^T \mathbf{g}(\mathbf{v}_1)$. Using ineguality 2.62, the square of the norm of this vector is

$$\begin{aligned} \left\| \delta(S) - D^T \mathbf{g}(\mathbf{v}_1) \right\|^2 &= \sum_{i=1,n} \left(f(\mathbf{v}_i) - f(\mathbf{v}_1) - \mathbf{d}_i^T \mathbf{g}(\mathbf{v}_1) \right)^2 && (1.63) \\ &\leq \sum_{i=1,n} \left(\frac{1}{2} L \sigma_+(S)^2 \right)^2 && (1.64) \\ &\leq n \left(\frac{1}{2} L \sigma_+(S)^2 \right)^2 && (1.65) \end{aligned}$$

which finally implies

$$\left\| \delta(S) - D^T \mathbf{g}(\mathbf{v}_1) \right\|^2 \leq \frac{1}{2} \sqrt{n} L \sigma_+(S)^2. \tag{1.66}$$

Let us define the constant $K = \frac{1}{2}\sqrt{n}L$. The previous inequality becomes

$$\left\|\delta(S) - D^T\mathbf{g}(\mathbf{v}_1)\right\|^2 \leq K\sigma_+(S)^2. \tag{1.67}$$

We can now plug the previous equality into inequality 2.60 and get

$$\|\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1)\| \quad \leq \quad K\left\|D(S)^{-T}\right\|\sigma_+(S)^2. \tag{1.68}$$

By proposition 2.4.2, we have $\sigma_+(S) \leq \|D\|$, so that the previous inequality is transformed into

$$\|\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1)\| \quad \leq \quad K\left\|D(S)^{-T}\right\|\|D(S)\|\,\sigma_+(S). \tag{1.69}$$

The l^2 norm of the matrix $D(S)$ is the largest eigenvalue of the matrix $D(S)^TD(S)$, so that the norm is not affected by transposition, which implies that $\left\|D(S)^{-T}\right\| = \|D(S)^{-1}\|$. The condition number of the matrix of direction $\kappa(S)$ is equal to $\|D(S)^{-1}\|\,\|D(S)\|$ ([3], section 2.7.2, "Condition"), which concludes the proof. ∎

Example (*Simplex gradient with a non-degenerate simplex*) In the following Scilab session, we define the function $f(\mathbf{x}) = x_1^2 + x_2^2$, where $\mathbf{x} \in \mathbb{R}^2$. The exact gradient of this function is $\mathbf{g} = (x_1, x_2)^T$. We create an axis-by-axis simplex based on the relatively small length $\ell = 10^{-3}$. This simplex defines a rectangular triangle, similar to the one presented in figure 2.3, but with smaller edges.

```
function y = myfunction ( x )
  y = x(1)^2 + x(2)^2
endfunction
x0 = [1.0 1.0];
len = 1.e-3;
si = optimsimplex_new ( "axes" , x0 , myfunction , len );
sg = optimsimplex_gradientfv ( si );
mprintf ( "Simplex␣Gradient=(%f␣%f)^T\n",sg(1),sg(2));
eg = [2 * x0(1) 2 * x0(2)].';
mprintf ( "Exact␣Gradient=(%f␣%f)^T\n",eg(1),eg(2));
err = norm(sg-eg)/norm(eg);
mprintf ( "Relative␣Error␣=␣%e\n",err);
err = norm(sg-eg);
mprintf ( "Absolute␣Error␣=␣%e\n",err);
D = optimsimplex_dirmat ( si );
k = cond(D);
mprintf ( "k(D)=%f\n",k);
ss = optimsimplex_size ( si );
mprintf ( "sigma_+(D)=%e\n",ss);
optimsimplex_destroy(si);
```

The previous script produces the following output.

```
Simplex Gradient=(2.001000 2.001000)^T
Exact Gradient=(2.000000 2.000000)^T
Absolute Error = 1.414214e-003
k(D)=1.000000
sigma_+(D)=1.000000e-003
```

We check that the inequality 2.55 gives an accurate measure of the approximation. Indeed, since the Lipshitz constant for the gradient $\mathbf{g}$ is $L = 2$, we have the constant $K = \sqrt{2}$. □

Example (*Simplex gradient with a simplex close to degenerate*) We consider what happens when an axis-by-axis simplex is transformed into a degenerate simplex. This situation is presented in

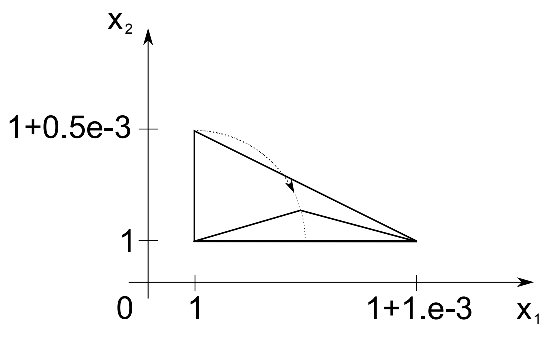

Fig. 1.5 : An axis-by-axis simplex which degenerates into a "flat" simplex in 2 dimensions.

θ (°)	$\sigma_+(S)$	$\|\overline{\mathbf{g}}(S) - \mathbf{g}(\mathbf{v}_1)\|$	$\kappa(S)$
90.000000	1.000000e-003	1.118034e-003	2.000000e+000
10.000000	1.000000e-003	2.965584e-003	1.432713e+001
1.000000	1.000000e-003	2.865807e-002	1.432397e+002
0.100000	1.000000e-003	2.864799e-001	1.432395e+003
0.010000	1.000000e-003	2.864789e+000	1.432394e+004
0.001000	1.000000e-003	2.864789e+001	1.432394e+005

figure 2.5, where the third vertex moves on a circle with radius $0.5.10^{-3}$ toward the center of an edge. Therefore the simplex degenerates and its condition number increases dramatically.

In the following Scilab script, we create a simplex as presented in figure 2.5. We use decreasing values of the angle θ between the two directions, starting from $\theta = 90$ (°) and going down to $\theta = 0.001$ (°). Then we compute the gradient and the absolute error, as well as the condition number and the size of the simplex.

```
R = 0.5e-3
coords = [
  1.0 1.0
  1.0+1.e-3 1.0
];
for theta = [90.0 10.0 1.0 0.1 0.01 0.001]
  C(1,1) = 1.0 + R * cos(theta*%pi/180);
  C(1,2) = 1.0 + R * sin(theta*%pi/180);
  coords(3,1:2) = C;
  si = optimsimplex_new ( coords , myfunction );
  sg = optimsimplex_gradientfv ( si );
  eg = [2 * x0(1) 2 * x0(2)].';
  err = norm(sg-eg);
  D = optimsimplex_dirmat ( si );
  k = cond(D);
  ss = optimsimplex_size ( si );
  mprintf ( "%f %e %e %e\n",theta , ss , err , k);
  optimsimplex_destroy(si);
end
```

The results are presented in table 2.4.3.

We see that while the oriented length $\sigma_+(S)$ is constant, the simplex gradient is more and more inaccurate as the condition number $\kappa(S)$ is increasing. □

1.5 References and notes

The section 2.4.3 and some elements of the section 2.2 are taken from Kelley's book [5], "Iterative Methods for Optimization". While this book focus on Nelder-Mead algorithm, Kelley gives a broad view on optimization and present other algorithms for noisy functions, like implicit filtering, multidirectional search and the Hooke-Jeeves algorithm.

The section 2.4.2, which present Taylor's formula with a Lisphitz continous gradient is based on [4], "Elements of Analysis, Geometry, Topology", section "Mean Value Theorem".

Bibliography

[1] M. J. Box. A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, 8(1):42–52, 1965.

[2] Ellen Fan. Global optimization of lennard-jones atomic clusters. Technical report, McMaster University, February 2002.

[3] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[4] Stephen J. Wright Jorge Nocedal. *Numerical Optimization.* Springer, 1999.

[5] C. T. Kelley. *Iterative Methods for Optimization*, volume 19. SIAM Frontiers in Applied Mathematics, 1999.

[6] 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.

[7] J. C. Nash. *Compact numerical methods for computers : linear algebra and function minimisation.* Hilger, Bristol, 1979.

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

[9] Parkinson and Hutchinson. An investigation into the efficiency of variants on the simplex method. *F. A. Lootsma, editor, Numerical Methods for Non-linear Optimization*, pages 115–135, 1972.

[10] W. H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, Second Edition.* 1992.

[11] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

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