Acta Univ. Sapientiae, Mathematica, **5**, 2 (2013) 169–183

A hyperbolic variant of the Nelder–Mead simplex method in low dimensions

Levente Lócsi
Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary
email: locsi@inf.elte.hu

Abstract. The Nelder–Mead simplex method is a widespread applied numerical optimization method with a vast number of practical applications, but very few mathematically proven convergence properties. The original formulation of the algorithm is stated in $\mathbb{R}^n$ using terms of Euclidean geometry. In this paper we introduce the idea of a hyperbolic variant of this algorithm using the Poincaré disk model of the Bolyai–Lobachevsky geometry. We present a few basic properties of this method and we also give a Matlab implementation in 2 and 3 dimensions.

1 Introduction

The Nelder–Mead simplex method [10] was published in 1965 and since then it has been applied in an enormous amount of practical optimization problems basically in every area of applied science. It has became one of the most widely known direct search methods for function minimization, it is also incorporated in the `fminsearch` command of numerical computational software systems such as Matlab or Scilab. Also the method's mathematical study has gained much attention, unfortunately there is very little known about its convergence properties. A breakthrough in lack of proven properties appeared in [5] in 1998, unfortunately the most useful theorems are only stated in 1 and 2 dimensions. Immediately on the following pages of the same volume [9] a counterexample is given in 2 dimensions where the method would fail to converge

2010 Mathematics Subject Classification: 65K05, 30F45, 51M10
Key words and phrases: Nelder–Mead simplex method, hyperbolic geometry, Poincaré disk model, Blaschke functions

to the unique minimizer given a tricky (but smooth and convex) function and a well-established initialization of the method. There are also quite a number of attempts to modify or restrict the method to enable convergence proofs, see e.g. [6, 11], or further related experiments in [3, 4]. A nice overview on the history of this method is given in [12]. Some of our recent works also summarizes some application experiences using this algorithm [2, 7].

Today also the non-Euclidean geometries are well known and accepted, we should mention the names of János Bolyai and Nikolai Lobachevsky who have independently clarified the notions of hyperbolic geometry in the early nineteenth century and after whom it is sometimes referred to as the Bolyai–Lobachevsky geometry. Later many models of hyperbolic geometry have been developed, one of these being the Poincaré disk model. Hyperbolic geometry is many times introduced or studied throughout this model, where the points of the plane are those inside the unit circle and the lines are the circular arcs intersecting the unit circle perpendicularly.[1] Even nowadays works appear with adapting some Euclidean notions and theorems to this model of hyperbolic geometry, see e.g. [1].

Our current work is also a member of this family. Specifically we adapt the Nelder–Mead simplex method (originally formulated in $\mathbb{R}^n$ using terms of Euclidean geometry) to the hyperbolic space, to the Poincaré disk model (in 2 dimensions) and its analogue using a unit sphere (in 3 dimensions). The motivation for this research came from our practise. We had the problem to choose some adequate points ("poles") inside the unit circle, and we have found the Nelder–Mead method to be the first to find a suitable set of poles without any a priori knowledge of their location. But this way we had to map the natural domain ($\mathbb{R}^n$) of the Nelder–Mead algorithm inside the unit circle, for details see e.g. [2, 7]. Along the way the adaptation of this method to the natural domain of our problem (which corresponds to the Poincaré disk model) seemed to be another promising path. In this paper we summarize the current results of our efforts, we introduce the hyperbolic Nelder–Mead method. We present a few basic properties of this method and we also give a Matlab implementation. Our hope is that with this approach some new directions will be opened both to the application and to the mathematical study of this algorithm.

The software tools (Matlab programs) can be downloaded from `http://numanal.inf.elte.hu/~locsi/hypnm/` in order to enable the Reader to reproduce the results presented in this paper.

[1] And of course diameters of the circle are also considered as lines.

2 The original method by Nelder and Mead

In this section we describe 'a simplex method for function minimization' following the original publication [10]. The statement of the algorithm shall now be given so that the specialities and calculations of $\mathbb{R}^n$ are skipped, only geometric terms shall be used, and therefore we may immediately imagine both the original idea and the hyperbolic realization.

The method relies on the comparison of the function values at the vertices of a non-degenerate simplex in our n-dimensional space $\mathbb{X}$. Let us call the vertices of the simplex $x_1, x_2, \dots, x_{n+1} \in \mathbb{X}$, the real valued function to be minimized f, and $y_i := f(x_i)$ $(i = 1, 2, \dots, n+1)$. We may start with an arbitrary simplex, which is usually chosen as a point $x_s \in \mathbb{X}$ and some 'nearby' points.

One *step* of the algorithm is basically a substitution of one point of the simplex, with a better one. Let us define the indices h and l such that y_h and y_l are respectively the *highest* (worst) and *lowest* (best) function values, and $\overline{x}$ the *centroid* of the points x_i with $i \neq h$. To carry out an update of the simplex, four operations are used.

1. *Reflection.* Reflect x_h across the point $\overline{x}$ to get x_r, $y_r := f(x_r)$. If $y_l \leq y_r < y_h$ then replace x_h with x_r and continue with the next step.

2. *Expansion.* If $y_r < y_l$, then reflect $\overline{x}$ across the point x_r to get x_e, $y_e := f(x_e)$. Now replace x_h with the x_e if $y_e < y_r$ or with x_r if $y_e \geq y_r$ and continue with the next step. (Thus we either stick with a new simplex gained with reflection, or create an expanded simplex.)

3. *Contraction.* If $y_r \geq y_i$ for all $i \neq h$ then define x_b as x_r if $y_r < y_h$ or as x_h if $y_r \geq y_h$ (so the 'better' of x_r and x_h), and find the midpoint x_c of $\overline{x}$ and x_b, $y_c := f(x_c)$. Now replace x_h with x_c and continue with the next step (with this contracted simplex); unless $y_c > \min\{y_h, y_r\}$, in this case perform the following operation.

4. *Shrink.* Leave only the point x_l and for all $i \neq l$ replace x_i with the midpoint of x_i and x_l; continue with the next step. (This operation is reported in [5] to be needed very rarely.)

With these steps our initial simplex 'adapts itself to the local landscape' (defined by the function f) and finally 'contracts on to the final minimum' as [10] summarizes the behaviour of the algorithm.[2]

[2] In the original description of the algorithm these four operations each depend on a given

We may stop the iteration e.g. if the function values have smaller standard deviation than a small given $\varepsilon > 0$ value or if we have made more steps than a prescribed limit etc.

Figure 1 presents the operations possible in one step of the Nelder–Mead algorithm in 2 dimensions, on the Euclidean plane. In this case the simplex is a triangle. On Figure 2 an example is shown for the progress of the algorithm optimizing the $\mathbb{R}^2 \to \mathbb{R}$ quadratic function $f(x, y) = x^2 + 6y^2 + 2xy$ with the initial simplex of coordinates $(1.2, 0.7), (1.1, 1.4)$ and $(1.7, 1.1)$. The first 10 steps are presented.

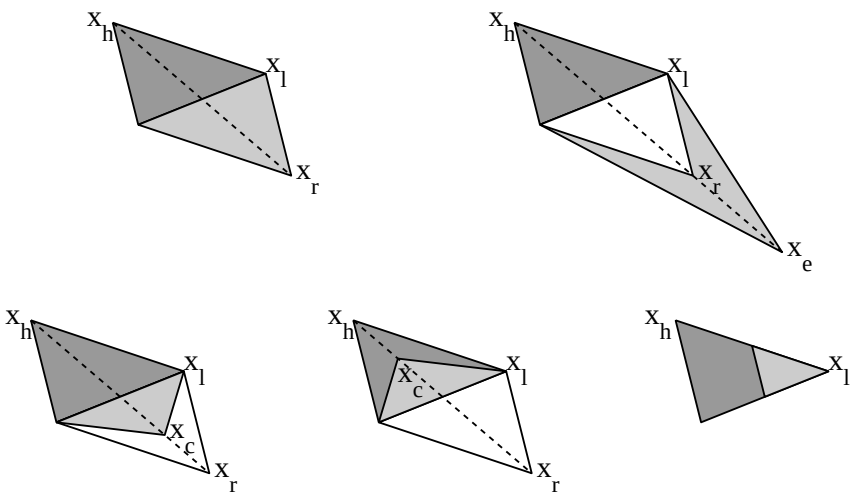

Figure 1: Operations on an Euclidean simplex in 2 dimensions. Respectively: reflection, expansion, (outside and inside) contraction and shrink. The old simplex is marked with dark gray, the new simplex with light grey, the reflected simplex is shown in white for the expansion and contraction operations.

Note that midpoints and reflected points can be calculated through simple linear combinations. This makes the (Euclidean) Nelder–Mead algorithm easy to implement, and quite effective, even in higher dimensions.

parameter. It is also shown that the 'natural choice' of these parameters, which are equivalent of calculating reflections and midpoints (as used above), are also the most efficient choice in practise.

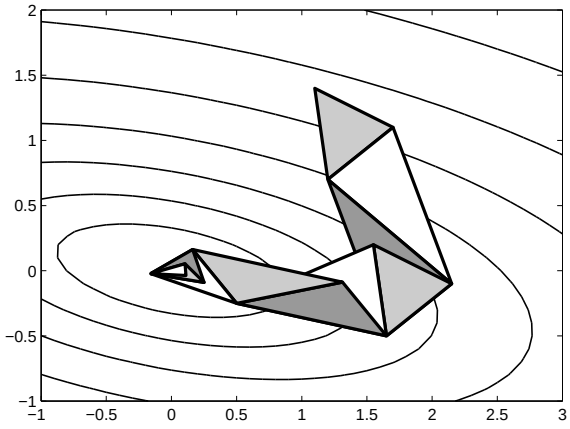

Figure 2: The Nelder–Mead algorithm optimizing a quadratic function on the Euclidean plane.

3 Constructions in hyperbolic spaces

We have given the statement of the Nelder–Mead simplex method so that in each step of the iteration some simple geometric calculations shall be done: finding centroid, midpoint, reflection across a point. These are valid constructions not only in Euclidean geometry, but also in hyperbolic geometry. In this section we give an overview of some possible approaches to numerically calculate the locations of the required points. Using these we will be able to put together the hyperbolic version of the Nelder–Mead algorithm in 2 and 3 dimensions.

3.1 In 2 dimensions

We will use the Poincaré disk model of hyperbolic geometry. It is useful to identify this model with the complex unit disk $\mathbb{D} := \{ z \in \mathbb{C} : |z| < 1 \}$. So the points of the plane are the complex numbers $z \in \mathbb{D}$. Let us also define the unit torus $\mathbb{T} = \{ z \in \mathbb{C} : |z| = 1 \}$, and $\mathbb{D}^* := \mathbb{C} \setminus (\mathbb{D} \cup \mathbb{T})$. The isometric transforms in this model (except for reflecting through a line) can be written by means of Blaschke functions, defined as

$$B_{a,d}(z) := d \cdot \frac{z - a}{1 - \overline{a}z} \qquad (a \in \mathbb{D}, d \in \mathbb{T}, z \in \mathbb{C}).$$

One approach makes use of the fact that for any $w_1, w_2 \in \mathbb{D}$, $w_1 \neq w_2$

there exists a unique set of values $(a, d, p) \in \mathbb{D} \times \mathbb{T} \times (0, 1)$ such that $B_{a,d}(0) = w_1, B_{a,d}(p) = w_2$ and $B_{a,d}$ maps the interval $[0, p]$ onto the hyperbolic line segment connecting w_1 and w_2. This way the calculation of midpoints and reflected points can be reduced to finding the appropriate points on $(0, 1)$. More on this method can be found in [2]. The advantage of this approach is the elegant and straightforward calculation with complex functions, the downside is that it is too much bound to the complex domain, to two dimensions, the generalization to higher dimensions is troublesome, if not impossible.

In contrast to the analytic techniques of the first approach, the second approach arises from geometric considerations. (Of course in two dimensions sometimes the use of complex expressions and Blaschke functions again makes the calculations easier.) We give a more detailed overview here, because our implementation relies on this second approach. Basically we have to imitate the regular constructions numerically.

- A *hyperbolic line* is basically a circular arc intersecting the unit circle perpendicularly.[3] It turns out that for the centre $c \in \mathbb{C}$ and radius $r \in \mathbb{R}$, $r > 0$ of such circles $c \in \mathbb{D}^*$ and $c\overline{c} = |c|^2 = 1 + r^2$ holds.

- Given two points $a, b \in \mathbb{D}$, $a \neq b$ we can *fit a hyperbolic line* on these two points by finding the centre of the circle which intersects the unit circle perpendicularly and passes through a and b. We know that the inverse image of a with respect to the unit circle can be expressed as $1/\overline{a}$ and also lies on the circle in question. (The same holds for b.) Now this circle can be found as the one fitted on the three points a, b and $1/\overline{b}$. This can be done e.g. by solving a linear system of equations.

- Finding *the intersection of two hyperbolic lines* translates to finding the intersection points of two circles (if they exist) and choosing the one inside $\mathbb{D}$. Note that also in hyperbolic geometry it cannot occur that two lines have exactly two intersections.

- The *perpendicular bisector* of a line segment between a and b can be found as a circle with centre both on the Euclidean line on a, b and the radical axis of the unit circle and the hyperbolic line (as Euclidean circle) on a and b.

[3] Since the diameters are also considered as lines, they should be handled as special cases in an implementation. The calculation is also easier in that case. These special cases will also occur in 3 dimensions, but we will not treat them here in more detail.

- The above three constructions allow us to find the *midpoint* of a line segment, which is also a *centroid* of two points: find the intersection of the hyperbolic line fitted on the two points and the perpendicular bisector of the line segment between the two points.

- The *reflection across a point* $\mathfrak{a} \in \mathbb{D}$ can be formalized e.g. using Blaschke functions. The formula $\mathrm{B}_{-\mathfrak{a},0}(-\mathrm{B}_{\mathfrak{a},0}(z))$ gives the reflected image of $z \in \mathbb{D}$ through $\mathfrak{a}$. (The idea is to reduce to reflect across the origin.)

Now we have all the constructions which are needed to adapt the Nelder–Mead method to the hyperbolic plane. But further geometric constructions can also be formulated. We have the hyperbolic analogue of: translation, reflection through a line, rotation around a point, perpendicular line at a point.

Figure 3 presents some basic elements in the Poincaré disk model of hyperbolic geometry, and the operations possible in one step of the Nelder–Mead algorithm (c.f. Figure 1 in the Euclidean case, and to the outline of the algorithm given in Section 2). In this case the simplex is a hyperbolic triangle.

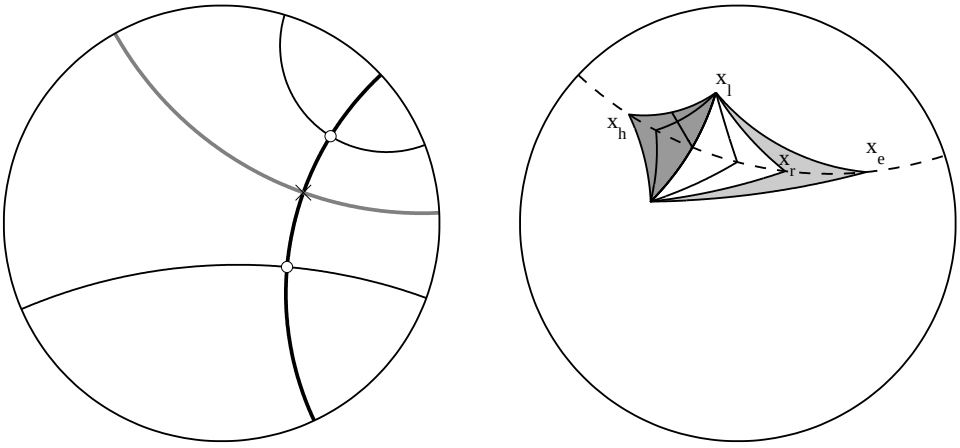

Figure 3: Left: Some basic elements of hyperbolic geometry. The line fitted on two points, perpendicular lines from the points, and a perpendicular bisector. Right: Operations on a hyperbolic simplex in 2 dimensions: reflection, expansion, (outside and inside) contraction and shrink. The old simplex is marked with dark gray, the reflected simplex with white, the further simplices are either shown in light grey or just their outlines.

3.2 In 3 dimensions

To define the analogue of the Poincaré disk model in 3 dimensions, i.e. a hyperbolic space, we will use $\mathbb{S} := \left\{ (x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1 \right\}$, the unit sphere. The points will be the ones inside $\mathbb{S}$, the lines will be the circular arcs intersecting the surface of $\mathbb{S}$ perpendicularly, the planes will be the spherical caps intersecting $\mathbb{S}$ perpendicularly.

It turns out that the usual basic constructions in Euclidean space (such as fitting a line on two points, fitting a plane on three points, finding the intersection line of two planes, finding the perpendicular bisector plane of a line segment etc.) can be also done and calculated in this hyperbolic space. Of course now we can not use the help of complex analysis, we have to deal with terms of analytic geometry in $\mathbb{R}^3$ by translating the required notions of hyperbolic lines and planes to circular arcs and spherical caps.

We refer to the program codes referenced in this paper (see end of Section 1) for construction and implementation details. Figure 4 presents some basic elements in the geometry of this three-dimensional hyperbolic space and the possible moves of a simplex in one step of the Nelder–Mead algorithm. In this case the simplex is a hyperbolic tetrahedron.

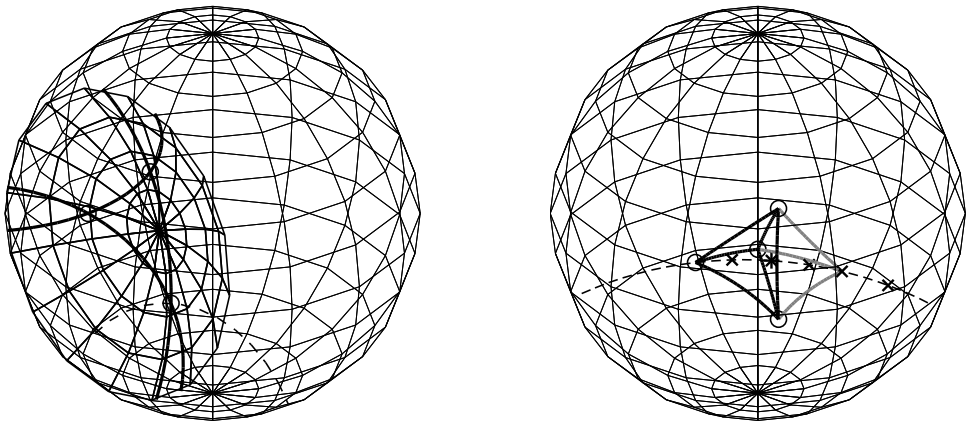

Figure 4: Left: Some basic elements of geometry in hyperbolic space. Three points on a plane, the lines of the arising triangle's edges and a perpendicular line at one of the points. Right: Operations on a hyperbolic simplex in 3 dimensions (without shrink), circles denote the vertices of the original simplex, a star the centroid of one side, exes the possible new vertices of a new simplex.

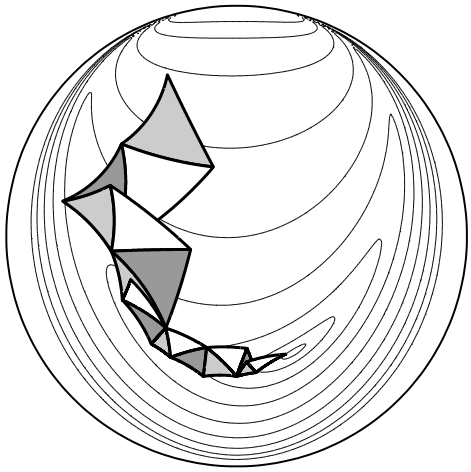

Figure 5: The Nelder–Mead algorithm adapted to the hyperbolic plane.

4 The hyperbolic simplex method

Having defined the Nelder–Mead simplex method in geometric terms (Section 2), and the needed constructions being present on the Poincaré disk model (and its three-dimensional analogue) of hyperbolic geometry (Section 3), now we have the hyperbolic realization of the method in our hands.

Figure 5 gives an example of the Nelder–Mead method optimizing a function on the hyperbolic plane. The function being minimized is similar to the famous Rosenbrock-function (or banana function, see [10]), with its natural domain $\mathbb{R}^2$ being mapped onto $\mathbb{D}$ using a map detailed in e.g. [7]. This function earned his fame, because numerical methods prior to the one proposed by Nelder and Mead were unable to determine its minimum. Now we see that the hyperbolic version is also capable of tending towards the optimum.

Furthermore an optimization process can be observed on Figure 6 in case of a quadratic function on the hyperbolic space as carried out by the Nelder–Mead algorithm.

The Reader is encouraged to download the collection of Matlab programs from the referenced homepage (see Section 1) and experiment with the algorithm, specific functions to optimize and constructions in hyperbolic geometry. Especially the three-dimensional graphics are more comprehensible when the user can interact with the figures, not just observe a printed planar projection.

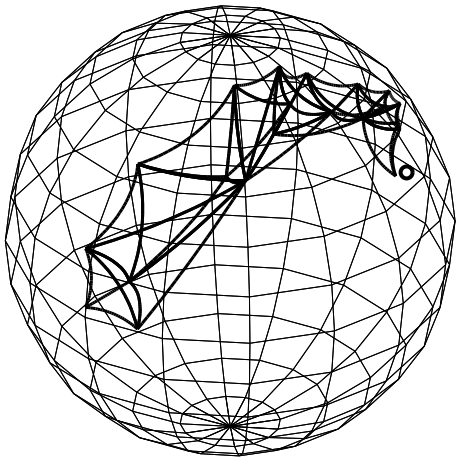

Figure 6: The Nelder–Mead algorithm adapted to the hyperbolic space.

5 Some basic properties

The mathematical study of the original Nelder–Mead simplex method (without any restrictions or modifications) has quite few proven properties or convergence theorems. Basically all known results are summarized in [5]. Some of its general results easily translate also to the hyperbolic versions introduced above. In this section we revisit these straightforward properties of the algorithm.

Proposition 1 (Nondegeneracy[4] of hyperbolic simplices) *If the initial simplex is nondegenerate, so are all subsequent simplices produced by the hyperbolic version of the Nelder–Mead algorithm.* (C.f. [5, Lemma 3.1.(1)].)

Proof. By construction, each of the trial points $\mathtt{x}_\mathtt{r}, \mathtt{x}_\mathtt{e}$ and $\mathtt{x}_\mathtt{c}$ (either inside or outside) lies strictly outside the face defined by the $\mathtt{n}$ best vertices, along the line joining $\mathtt{x}_\mathtt{h}$ and $\bar{\mathtt{x}}$. If a nonshrink operation occurs, the worst vertex is replaced by one of these trial points, thus the simplex remains nondegenerate. If a shrink operation occurs, then each vertex (except the best) is replaced by the midpoint of the line segment defined by the current and the best vertex.

[4] We understand by nondegeneracy that the vertices of a simplex are not collinear on the hyperbolic plane, not coplanar in the hyperbolic space, and in general: no lower-dimensional hyperspace can be found which contains all vertices of the simplex.

Also in this case it is clear from the geometry that the simplex' nondegeneracy is again preserved. □

The function values at the vertices of the simplex were denoted by y_i, now let us also mark the number of iterations, and require that at the beginning of each iteration step the vertices are ordered, i.e. in step k

$$y_1^{(k)} \leq y_2^{(k)} \leq \ldots \leq y_{n+1}^{(k)}$$

holds in case of a simplex in n dimensions.

Proposition 2 (Convergence of function values at vertices) *Let f be a function defined on the n-dimensional hyperbolic space $\mathbb{X}$, that is bounded from below. When the Nelder–Mead algorithm is applied to minimize f, starting with a nondegenerate simplex, then* (c.f. [5, Lemma 3.3])

1. *the sequence $\left(y_1^{(k)}\right)$ always converges;*
2. *at every nonshrink iteration k, $y_i^{(k+1)} \leq y_i^{(k)}$ $(1 \leq i \leq n+1)$ with strict inequality for at least one value of i;*
3. *if there are only a finite number of shrink steps, then*
   (a) *each sequence $\left(y_i^{(k)}\right)$ $(1 \leq i \leq n+1)$ converges as $k \to \infty$,*
   (b) $\lim_{k\to\infty} y_i^{(k)} =: y_i^* \leq y_i^{(k)}$ *for $1 \leq i \leq n+1$ and all k,*
   (c) $y_1^* \leq y_2^* \leq \ldots \leq y_{n+1}^*$.

Note that this proposition does not state that the algorithm will converge to a global (or even local) minimum point. This is unfortunately not true in general (see counterexample in [9]).

Proof.

1. Since the algorithm never replaces the best vertex with a point of higher function value, the sequence $\left(y_1^{(k)}\right)$ is monotonically non-increasing and bounded from below (like f), thus it is convergent.
2. A shrink step could result in higher function values at the simplex' vertices, but other operations always replace the worst value with a better one, thus—taking to account also the ordering at the beginning of each step—some values will be strictly lower and none of them will increase.

3. Shrink steps are reported to be taken extremely rarely, so assuming their finiteness is a very weak restriction. Otherwise these statements are immediate consequences of the previous arguments and the properties of convergent sequences.

□

Now we will show that shrink steps will not occur at all if the method is applied to a strictly convex function on the hyperbolic space $\mathbb{X}$, endowed with the metric $\rho\colon \mathbb{X}\times\mathbb{X}\to\mathbb{R}$.[5]

Definition 1 (Strict convexity) *The function* f *defined on the points of hyperbolic space* $\mathbb{X}$ *is called* strictly convex *if for every* $a, b \in \mathbb{X},\ a \neq b$*, and for every point* p *on the line segment connecting* a *and* b *(with endpoints excluded) the following formula holds:*

$$f(p) < \lambda\cdot f(a) + (1-\lambda)\cdot f(b), \quad \textit{with} \quad \lambda := \frac{\rho(p,b)}{\rho(a,b)}.$$

Basically this is the usual definition of strict convexity, but now, because of dealing with hyperbolic spaces, the usual terms with linear combinations also with the points of the metric space had to be omitted.

It is easy to see that in case of a strictly convex function f, for every point p on the open line segment connecting a and b

$$f(p) < \max\{f(a), f(b)\}$$

holds[6], specifically also a centroid of 2, 3 (or more) points has lower function value than the maximum of the function values at the given points.

Proposition 3 (No shrink for strictly convex functions) *Assume that* f *is a strictly convex function defined on the points of the hyperbolic space* $\mathbb{X}$ *and that the Nelder–Mead algorithm is applied to minimize* f*, starting with a non-degenerate simplex. Then no shrink steps will be taken.* (C.f. [5, Lemma 3.5].)

Proof. A shrink should be performed when we fail to accept the relevant contraction point x_c. We will show now that this can not happen.

[5] For instance the usual metric on the Poincaré disk model can be expressed using Blaschke functions as $\rho(a,b) = |B_a(b)|$.

[6] Furthermore it turns out that this property would have been sufficient to prove Proposition 3.

It follows from the statement of the algorithm that if we are considering a contraction point, then $y_n \leq y_r$ and of course $y_n \leq y_{n+1} = y_h$ holds. We can assume that $y_r < y_{n+1}$ (i.e. $y_n \leq y_r < y_{n+1}$ holds), the other case is settled similarly. Now $\overline{x}$ is the centroid of $x_1, \ldots, x_n$, so by the strict convexity of f, $f(\overline{x}) < y_n$ holds. The point x_c is the midpoint of $\overline{x}$ and x_r, so $y_c < \max\{f(\overline{x}), y_r\} = y_r$. So now $y_c < y_r < y_h$ holds, hence x_c will be accepted, a contraction shall be made and a shrink step will not be taken. □

6 Summary

In this paper we have introduced a hyperbolic variant of the Nelder–Mead simplex method. The algorithm was adapted to the Poincaré disk model of the Bolyai–Lobachevsky geometry (in two dimensions), as well as its three-dimensional analogue.

Matlab implementations (and resulting graphics) were presented about the necessary geometric constructions in the hyperbolic spaces at hand, which are—together with the adapted variant of the Nelder–Mead method—available to download at http://numanal.inf.elte.hu/~locsi/hypnm/.

Finally some straightforward mathematical properties of the original simplex method were translated to the hyperbolic case.

7 Directions of further research

Apart from the Poincaré disk model, it might be interesting to adapt the Nelder–Mead simplex method to other models (such as the Klein model, Poincaré half-plane model etc.) or other geometries.

Naturally also the higher-dimensional cases should be investigated and implemented.

Of course the detailed analysis of both

- the practical convergence properties of this method (compared to the original Nelder–Mead method), and
- the mathematical convergence properties of the hyperbolic Nelder–Mead method

awaits to be carried out, with the first task requiring more of an engineering approach, and the second one likely to be quite unpromising taking to account the similar efforts in the case of the original algorithm. Nevertheless we may have some hope at least in low dimensions.

Our current investigation lacks the most basic special case: optimization in 1 dimension, on a hyperbolic line—a line or line segment endowed with a hyperbolic metric. Note that [5] contains results also in 1 dimensions.

Acknowledgements

The author would like to thank Prof. Ferenc Schipp for his valuable hints and constant motivation in this research area.

Preliminary work related to this paper was presented at the Workshop on Dyadic Analysis and Related Areas with Applications[7] in 2009, Dobogókő and at the János Bolyai Memorial Conference in 2010, Budapest–Târgu Mureş [8]. Thanks to the organizers of both conferences.

Some of the implementation was carried out at the Systems and Control Lab of the Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA SZTAKI). Thanks for the friendly atmosphere at this research group.

References

[1] C. Barbu, L-I. Pişcoran, The orthopole theorem in the Poincaré disc model of hyperbolic geometry, *Acta Univ. Sapientiae Math.*, **4** (2012), 20–25.

[2] S. Fridli, P. Kovács, L. Lócsi, F. Schipp, Rational modeling of multi-lead QRS complexes in ECG signals, *Ann. Univ. Sci. Budapest., Sect. Comput.*, **36** (2012), 145–155.

[3] L. Han, M. Neumann, Effect of dimensionality on the Nelder–Mead simplex method, *Optim. Methods Softw.*, **21** (2006), 1–16.

[4] L. Han, M. Neumann, J. Xu, On the roots of certain polynomials arising from the analysis of the Nelder–Mead simplex method, *Linear Algebra Appl.*, **363** (2003), 109–124.

[5] J. C. Lagarias, J. A. Reeds, M. H. Wright, P. E. Wright, Convergence properties of the Nelder–Mead algorithm in low dimensions, *SIAM J. Optim.*, **9** (1998), 112–147.

[7] http://wda2009.inf.elte.hu/

[6] J. C. Lagarias, B. Poonen, M. H. Wright, Convergence of the restricted Nelder–Mead algorithm in two dimensions, *SIAM J. Optim.*, **22** (2012), 501–532.

[7] L. Lócsi, Approximating poles of complex rational functions, *Acta Univ. Sapientiae Math.*, **1** (2009), 169–182.

[8] L. Lócsi, Investigating hyperbolic variants of the Nelder–Mead simplex method, *Abstracts of the János Bolyai Memorial Conference*, (2010), 49.

[9] K. I. M. McKinnon, Convergence of the Nelder–Mead simplex method to a non-stationary point, *SIAM J. Optim.*, **9** (1998), 148–158.

[10] J. A. Nelder, R. Mead, A simplex method for function minimization, *Comput. J.*, **7** (1965), 308–313.

[11] C. J. Price, I. D. Coope, D. Byatt, A convergent variant of the Nelder–Mead algorithm, *J. Optim. Theory App.*, **113** (2002), 5–19.

[12] M. H. Wright, Nelder, Mead, and the other simplex method, *Doc. Math.*, Extra Volume: Optimization Stories (2012), 271–276.

Received: 4 September 2013