The Nelder-Mead Algorithm in Two Dimensions

CEE 201L. Uncertainty, Design, and Optimization
Department of Civil and Environmental Engineering
Duke University
Henri P. Gavin
Spring, 2016

The Nelder-Mead algorithm provides a means of minimizing an objective function of \( n \) design parameters, \( f(x) \), \( x = [x_1, x_2, \ldots, x_n]^T \). The algorithm may be extended to constrained minimization problems through the addition of a penalty function. The Nelder-Mead algorithm iterates on a simplex, which is a set of \( n + 1 \) designs, \( [x^{(1)}, x^{(2)}, \ldots, x^{(n+1)}] \). The Nelder-Mead algorithm specifies a sequence of steps for iteratively updating the worst design in the simplex (\( x^{(n+1)} \)) in order to converge on the smallest value of \( f(x^{(1)}) \).

The simplex may be thought of as a polygon with \( n + 1 \) vertices. If \( n = 2 \), the simplex is a triangle, and the Nelder-Mead algorithm may be easily visualized. If \( n = 3 \), the simplex is a tetrahedron. This document introduces the Nelder-Mead algorithm for triangular simplexes.

Consider a simplex of three points \( [u, v, w] \) in the \( x_1 - x_2 \) plane, the triangle connecting them, and the objective function evaluated at the three points, \( f(u), f(v), \) and \( f(w) \). The steps listed below, and illustrated in Figure 1, iteratively improve the vertices of the triangle in order to minimize \( f(x) \).

Steps for one iteration of the Nelder-Mead Algorithm

1. **Sort** the vertices such that \( f(u) < f(v) < f(w) \). Point \( u \) is the best point, point \( v \) is the next-to-worst point, and point \( w \) is the worst point.

2. **Reflect** the worst point, \( w \), through the centroid of the remaining points (\( u \) and \( v \)) to obtain the reflected point, \( r \), and evaluate \( f(r) \).
   
   If the cost at the reflected point, \( f(r) \), is between the best and next-to-worst cost (\( f(u) < f(r) < f(v) \)) then replace the worst point, \( w \), with the reflected point, \( r \), and go to step 5.

3. If the cost at the reflected point, \( f(r) \), is better than \( f(u) \) (\( f(r) < f(u) \)) then **extend** the reflected point, \( r \), further past the average of \( u \) and \( v \), to point \( e \), and evaluate \( f(e) \).
   
   (a) If the cost at the extended point, \( f(e) \), is better than the reflected point cost, \( f(r) \), then replace the worst point, \( w \), with the extended point, \( e \), and go to step 5.

   (b) Otherwise replace the worst point, \( w \) with the reflected point, \( r \), and go to step 5.

4. If the inequalities of steps 2 and 3 are not satisfied, then it is certain that the reflected point, \( r \), is worse than the next-to-worst point, \( v \), (\( f(r) > f(v) \)) and, a smaller value of \( f \) might be found between points \( w \) and \( r \). So try to **contract** the worst point, \( w \), to a point \( e \) between \( w \) and \( r \), and evaluate \( f(w) \). The best distance along the line from \( w \) to \( r \) can be hard to
determine, and, in general, it is not worth trying too hard to find this minimum. Typical
values of $c$ are one-quarter and three-quarters of the way from $w$ to $r$. These are called inside
and outside contraction points, $c_i$ and $c_o$.

(a) If the cost at the better of the two contraction points is better than the next-to-worst
cost, $\min[f(c_i), f(c_o)] < f(v)$, then replace $w$ with the better contraction point, $c_i$ or $c_o$,
and go to step 5.

(b) Otherwise shrink the simplex into the best point, $u$, and go to step 5.

5. Check convergence.

The simplex has converged if the simplex is “sufficiently small” and if function values at the
simplex vertices are “sufficiently close.” “Sufficiently” is quantified by convergence tolerances
$\epsilon_x$ and $\epsilon_f$. These convergence tolerances are problem-dependent and can be user-specified.

(a) If the largest difference between the adjacent vertices is less than $\epsilon_x$ times the average of
adjacent vertices,

$$2 \max \left| \frac{u, v}{} - \frac{v, w}{} \right| < \epsilon_x$$

for $n = 2$, as in this example

$$2 \max \left| \frac{X(:, 1 : n) - X(:, 2 : n + 1)}{X(:, 1 : n) + X(:, 2 : n + 1)} \right| < \epsilon_x$$

for $n \geq 2$, more generally

and if the difference between worst and best function values is less than $\epsilon_f$ times the
best function value,

$$\left| \frac{f(w) - f(u)}{f(u) + 10^{-9}} \right| < \epsilon_f$$

for $n = 2$, as in this example

$$\left| \frac{f(n + 1) - f(1)}{f(1) + 10^{-9}} \right| < \epsilon_f$$

for $n \geq 2$, more generally

then the iterations have converged, and the algorithm is terminated.

In the convergence criteria for $n \geq 2$, the simplex $X$ is a $n$-by-$(n + 1)$ matrix, in which
each column of $X$ is a vertex of the simplex. The objective function for each column of
the simplex is in a vector $f$ of dimension $1$-by-$(n + 1)$. The vector of objective functions
is sorted in increasing order $f(1) < f(2) < \cdots < f(n + 1)$. The columns of $X$ correspond
to the elements of $f$ . . . $f(1) = f(X(:, 1)), f(2) = f(X(:, 2))$, etc.

In this 2D example, $X = [u, v, w]_{2 \times 3}$ and $f = [f(u), f(v), f(w)]_{1 \times 3}$.

(b) If the number of function evaluations has exceeded a user-specified limit then the algo-
rithm is terminated

If the convergence criteria are not met and if the number of function evaluations has not been
exceeded, then the algorithm returns to step 1 for the next iteration.

An extension of this method, for any value of $n$ and including constraints on parameter
values and general inequality constraints, is implemented in the .m-function NMAopt.m available at:

http://www.duke.edu/~hpgavin/cee201/NMAopt.m
The Nelder-Mead Algorithm in Two Dimensions

Remarks

1. In an iteration, the Nelder-Mead method requires one (r), two (r and e), three (r, c_i, and c_o), or 3 + n (r, c_i, c_o, and n to shrink) function evaluations.

2. Within any iteration, the best point is not adjusted. The best point can be re-assigned when the simplex is re-sorted.

3. In 2-D, a simplex and its reflection make a parallelogram.

4. If a reflection point is selected, the simplex remains the same size.

5. In an extension, the line from w to e passes through point r. If an extension point is selected, the simplex grows in size.

6. After a shrink operation, if w' remains the worst point the reflection direction is not changed.

7. The simplex will decrease in size only if it does not improve by reflection or extension.

8. The six steps listed here apply to problems with n > 2 as well, with u = x^{(1)}, v = x^{(n)}, and w = x^{(n+1)}.

9. The Nelder-Mead method is efficient in getting to the general area of a minimum point, but is not efficient in converging to a precise minimum (Lagarias, 1998; McKinnon, 1998). The simplex can tend to oscillate around a minimum point, can shrink into itself, or can converge to a non-minimum point. These problems can arise when a penalty function is used to enforce a constraint at a minimum point and all contraction points are “inside contractions” (McKinnon, 1998). This is related to the fact that, as the method is defined, the factors used for reflecting, extending, contracting and shrinking the simplex (typically 1, 2, 1/2, and 1/2) are fixed and do not depend on how a problem is converging. Nelder and Mead recognized this issue in their original (1965) paper, and hinted at a way to compute the Hessian of the objective function from the simplex once the simplex converged.

10. In step 4, comparing f(c) to f(v) instead of f(w) decreases the likelihood that c will be accepted, and therefore increases the likelihood of a shrink operation. Shrink operations are preferable to contractions for two reasons. First, contractions tend to make the simplex degenerate (more than two vertices on a line). Second, shrink operations decrease the size of the simplex, leading to faster convergence with respect to \( \epsilon_x \). On the other hand, shrink operations require n function evaluations, and are therefore more computationally expensive than the other operations.

11. Convergent variants of the Nelder-Mead method have been proposed, (e.g., David Byatt, 2000 and Tseng 2001).

12. Using the Nelder-Mead method to converge in the general region of a precise solution before switching to a gradient-based method, such as sequential quadratic programming (SQP), can sometimes work well with difficult optimization problems.
References


The Nelder-Mead Algorithm in Two Dimensions

1. Sort
   - If $f(u) < f(v) < f(w)$, then $w = r$.

2. Reflect
   - If $f(u) < f(r) < f(v)$, then $w = e$.
   - Otherwise, $w = r$.

3. Extend
   - If $f(c_i) < \min[f(c_o), f(v)]$, then $w = c_i$.
   - Otherwise, shrink.

4.a. Contract
   - If $f(c_i) < \min[f(c_o), f(v)]$, then $v' = v'$.

4.b. Shrink
   - Otherwise, shrink.

Figure 1. Illustration of the sequence of steps in one iteration of the Nelder-Mead method (for $n = 2$).

Figure 2. A sequence of seven iterations of the Nelder-Mead method. Dashed lines show attempted reflection steps.