2 Das Verfahren von Nelder-Mead

Das *Nelder-Mead-* oder auch *Simplexverfahren* genannte Verfahren [1965] wird zur globalen Minimierung von stetigen Funktionen benutzt und verwendet entsprechend keine Informationen über Ableitungen der zu minimierenden Funktion. Es besteht darin, eine Folge von Simplexen[1] ($n = 2$: Dreiecke, $n = 3$: Tetraeder) zu erzeugen, in der Hoffnung, daß die Simplexe immer kleiner werden und die gesuchte Minimalstelle einschließen. Es liefert in manchen Fällen überraschend gute Ergebnisse, liefert aber auch manchmal eine Scheinkonvergenz mit falschen Resultaten.

Das Verfahren ist erst in neuerer Zeit theoretisch untersucht und Schwächen sind durch alternative Formen ersetzt worden, insbesondere von P. Tseng[2]. Eine Übersicht über Ergebnisse bis etwa 1995 gibt M. H. Wright [1995].

Das Verfahren ist konzipiert für Funktionen

$$f \in \mathbf{C}(\mathbb{R}^n \to \mathbb{R}), \tag{2.1}$$

aber am besten zu erklären für $n = 2$. Es ist für $n \geq 3$ aufwendig.

2.1 Der zweidimensionale Fall

Wir beschreiben das Verfahren für $n = 2$. Die Beschreibung ist aber so gefaßt, daß sie fast ohne Änderungen für den allgemeinen Fall gilt.

Für $n = 2$ starten wir mit einem beliebigen Dreieck Δ. Die Menge der Ecken von Δ bezeichnen wir mit $\mathcal{E}$. Für die Ecken von Δ mit dem kleinsten, größten, und zweitgrößten Wert von f führen wir die folgenden Bezeichnungen ein:

$$\begin{aligned}
z_b &:= \arg\min\{f(z) : z \in \mathcal{E}\} \quad \text{(bester Wert)}, && (2.2)\\
z_s &:= \arg\max\{f(z) : z \in \mathcal{E},\ z \neq z_b\} \quad \text{(schlechtester Wert)}, && (2.3)\\
z_2 &:= \arg\max\{f(z) : z \in \mathcal{E},\ z \neq z_b,\ ,\ z \neq z_s\} \quad \text{(zweitschlechtester Wert)}. && (2.4)
\end{aligned}$$

Die Bezeichnung $\arg\min\{\ldots\}$ soll bedeuten, daß wir die Stelle suchen, an der das Minimum (entsprechend bei $\arg\max\{\ldots\}$ das Maximum) angenommen wird, und nicht den Wert an dieser Stelle. Gibt es mehrere beste, schlechteste, zweitgrößte Werte, so entscheiden wir uns für irgend eine mögliche Wahl. Sind z. B. alle Werte an den Ecken gleich groß, so können wir z_b, z_s, z_2 beliebig festsetzen. Wir wählen jetzt zur Beschreibung des weiteren Vorgehens

$$\alpha := 1, \quad \beta := 2, \quad \gamma := 0.5, \tag{2.5}$$

[1] Wir verwenden die Formen: das Simplex, die Simplexe

[2] ftp://ftp.math.washington.edu/pub/tseng/papers/

wollen aber offen lassen, diese Werte zu einem späteren Zeitpunkt zu verändern. Unser Ziel ist die Bestimmung eines neuen Dreiecks, mit dem wir dann die angestellten Rechnungen wiederholen.

Wir projizieren dazu die schlechteste Ecke z_s an dem Mittelpunkt z_M der Seite $\overline{z_b, z_2}$ auf die andere Seite des Dreiecks und erhalten eine neue Ecke z_p (p von projiziert) und damit ein weiteres Dreieck, das wir $\tilde{\Delta}$ nennen mit den Ecken z_2, z_b, z_p (auf die Reihenfolge der Ecken achten wir hier nicht). Man vgl. Abbildung 2.1, rechte Seite. Formelmäßig ausgedrückt ist

$$z_p = z_M + \alpha(z_M - z_s), \quad z_M = \frac{1}{n} \sum_{\substack{z \in \mathcal{E} \\ z \neq z_s}} z. \tag{2.6}$$

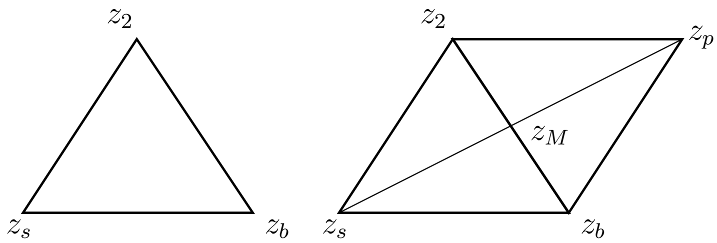

Abb. 2.1. Das Ausgangsdreieck und der erste Schritt beim Verfahren von Nelder-Mead

Die Menge der Ecken von $\tilde{\Delta}$ bezeichnen wir entsprechend mit $\tilde{\mathcal{E}}$. Wir unterscheiden 3 Fälle:

$$z_p = \tilde{z}_b \quad := \quad \arg\min\{f(z) : z \in \tilde{\mathcal{E}}\}, \tag{2.7}$$
$$z_p = \tilde{z}_s \quad := \quad \arg\max\{f(z) : z \in \tilde{\mathcal{E}},\ z \neq \tilde{z}_b\}, \tag{2.8}$$
$$z_p \neq \tilde{z}_b,\ z_p \neq \tilde{z}_s. \tag{2.9}$$

Fall (2.7): In diesem Fall prüfen wir, ob wir noch über z_p hinausgehen können. Wir definieren daher versuchsweise

$$z_E := z_M + \beta(z_p - z_M) \quad (E \text{ von Expansion}). \tag{2.10}$$

Ist z_E besser als z_p (also $f(z_E) < f(z_p)$), so ist der Expansionsschritt gelungen, und wir ersetzen endgültig z_s durch z_E. Ist jedoch z_E nicht besser als z_p, so ersetzen wir z_s durch z_p. Man vergleiche dazu die Abbildung 2.2.

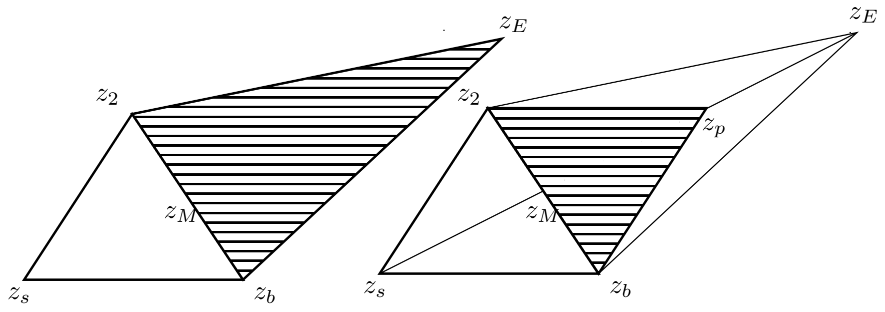

Abb. 2.2. Der Fall (2.7) (mögliche Expansion) beim Verfahren von Nelder-Mead

Fall (2.8): Wir verkleinern das Polyeder $\tilde{\Delta}$ durch Kontraktion. Wir definieren versuchsweise

$$z_K := \begin{cases} z_M + \gamma(z_p - z_M) & \text{falls } z_p \text{ besser ist als } z_s, \\ z_M + \gamma(z_s - z_M) & \text{sonst,} \end{cases} \quad (K \text{ von Kontraktion}). \qquad (2.11)$$

Ist z_K besser als z_s, so ersetzen wir z_s durch z_K. Ist das nicht der Fall, so bilden wir ein neues, kleineres Polyeder mit den neuen Ecken

$$z_{\text{neu}} := (1-\gamma)z + \gamma z_b, \quad z \in \mathcal{E}. \qquad (2.12)$$

Der erste Fall ist in Abbildung 2.3, der zweite Fall in Abbildung 2.4 illustriert.

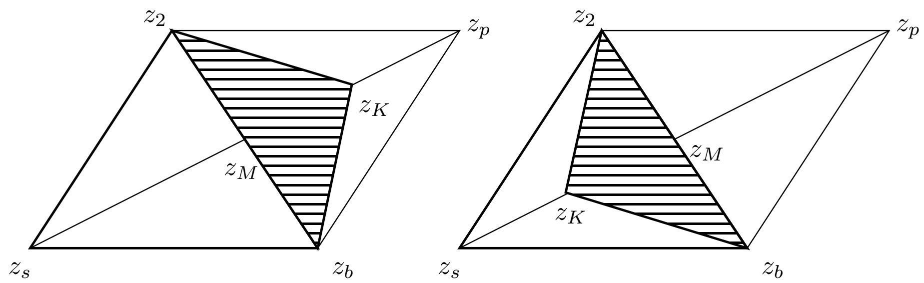

Abb. 2.3. Der Fall (2.8) (Kontraktionsfall, z_K ist besser als z_p) beim Verfahren von Nelder-Mead

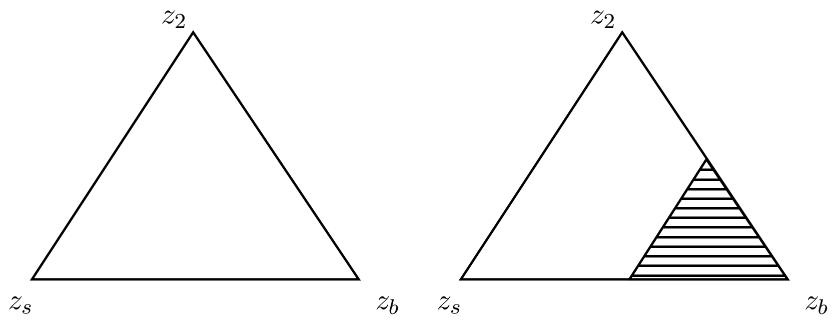

Abb. 2.4. Der Fall (2.8) (Kontraktionsfall, z_K ist nicht besser als z_p) beim Verfahren von Nelder-Mead

Fall (2.9): In diesem Fall ersetzen wir z_s durch z_p. Zur Illustration diene Abbildung 2.2, rechte Seite.

Damit haben wir beschrieben, wie wir vom Ausgangsdreieck Δ zu einem neuen Dreieck gelangen. Wir wiederholen dieses Verfahren und brechen es ab, wenn das dann ausgerechnete Dreieck hinreichend klein ist (numerische Konvergenz), oder wenn die Anzahl der Iterationen eine vorgegebene Zahl erreicht hat.

Es bleibt noch zu beschreiben, wie man das Ausgangssimplex ($n = 2$, Ausgangsdreieck) mit den Ecken $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ konstruiert. Wollen wir ein gleichseitiges Simplex ($n = 2$: Dreieck) konstruieren, so gehen wir so vor: Wir definieren für $c > 0$ und beliebiges $\boldsymbol{x}_0 \in \mathbb{R}^n$ die übrigen Ecken $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ folgendermaßen:

$$\boldsymbol{e}_j := (0, 0, \cdots, 1, 0 \cdots, 0)^{\mathrm{T}} \in \mathbb{R}^n, \quad \boldsymbol{e}_j(k) = \delta_{jk}, \tag{2.13}$$

$$\boldsymbol{q} := (q, q, \ldots, q) \in \mathbb{R}^n, \quad q := c\frac{\sqrt{n+1} - 1}{\sqrt{2}n}, \tag{2.14}$$

$$\boldsymbol{x}_j := \boldsymbol{x}_0 + \boldsymbol{q} + \frac{c}{\sqrt{2}}\boldsymbol{e}_j, \; j = 1, 2, \ldots, n. \tag{2.15}$$

Es ist dabei $||\boldsymbol{x}_j - \boldsymbol{x}_k||_2 = c$ für $j \neq k$. Hat man nach ℓ Schritten ein Simplex mit den Ecken $\boldsymbol{x}_0^{(\ell)}, \boldsymbol{x}_1^{(\ell)}, \ldots, \boldsymbol{x}_n^{(\ell)}$ berechnet, so kann man die Rechnung beenden, wenn

$$\max_{j,k} ||\boldsymbol{x}_j^{(\ell)} - \boldsymbol{x}_k^{(\ell)}||_2 \leq \varepsilon \tag{2.16}$$

bei vorgegebener Toleranz ε. Hier wird nur abgebrochen, wenn der Simplex hinreichend klein ist. Eine Abbruchbedingung, die nur die Funktionswerte berücksichtigt, erhält man mit $\bar{f} = \frac{1}{n+1}\sum_{i=1}^{n+1} f(\boldsymbol{x}_i)$ durch

$$\sqrt{\sum_{i=1}^{n+1} \frac{(f(\boldsymbol{x}_i) - \bar{f})^2}{n}} < \varepsilon. \tag{2.17}$$

Beispiel 2.5. Wir behandeln zwei einfache Beispiele. (a)$f(x,y) = \text{const}$, (b) $f(x,y) = (x-1)^2 + (y-10)^4$. Im ersten Fall ist jeder Punkt (x,y) eine Minimalstelle, im zweiten Fall $(x,y) = (1,10)$.

Das Ausgangsdreieck hat mit $c = 1$ und $\boldsymbol{x}_0 = \mathbf{0}$ nach (2.13) - (2.15) die Ecken

$$((0,0),(q+\sqrt{2}/2,q),(q,q+\sqrt{2}/2)), \tag{2.18}$$

mit $q = (\sqrt{6}-\sqrt{2})/4 \approx 0.2588, q+\sqrt{2}/2 \approx 0.9659$. Wir könnten aber auch mit den folgenden, einfacher ausrechenbaren Ecken beginnen:

$$((0,0),(0,1),(1,0)). \tag{2.19}$$

Im Fall der konstanten Funktion ist die Verteilung z_b, z_s, z_2 auf die Ecken willkürlich. Verwendet man die erste Abbruchbedingung (2.16), so kann es also eintreten, daß der Kontraktionsfall (2.8) nie eintritt. Dann werden nur die Dreiecke $\Delta, \tilde{\Delta}$ erzeugt, und Konvergenz tritt nicht ein. Kommt jedoch der Fall (2.8) immer wieder vor, so gibt es wenigstens eine Folge von kleiner werden Dreiecken. Bei Abbruchbedingung (2.17) wird das Verfahren schon im ersten Schritt gestoppt.

Im zweiten Fall kann man Konvergenz beobachten. Startet man mit dem einfachen Dreieck (2.19), so sind die Schritte 1–10 und die Schritte 26–30 in Abbildung 2.6 angegeben. Eine Vergleichsrechnung mit jeweils 40 Schritten und den Anfangsdreiecken (2.19) und (2.18) haben wir in Abbildung 2.7 angegeben. Das dabei verwendete matlab-Programm ist zum Schluß als Programm 2.8 abgedruckt.

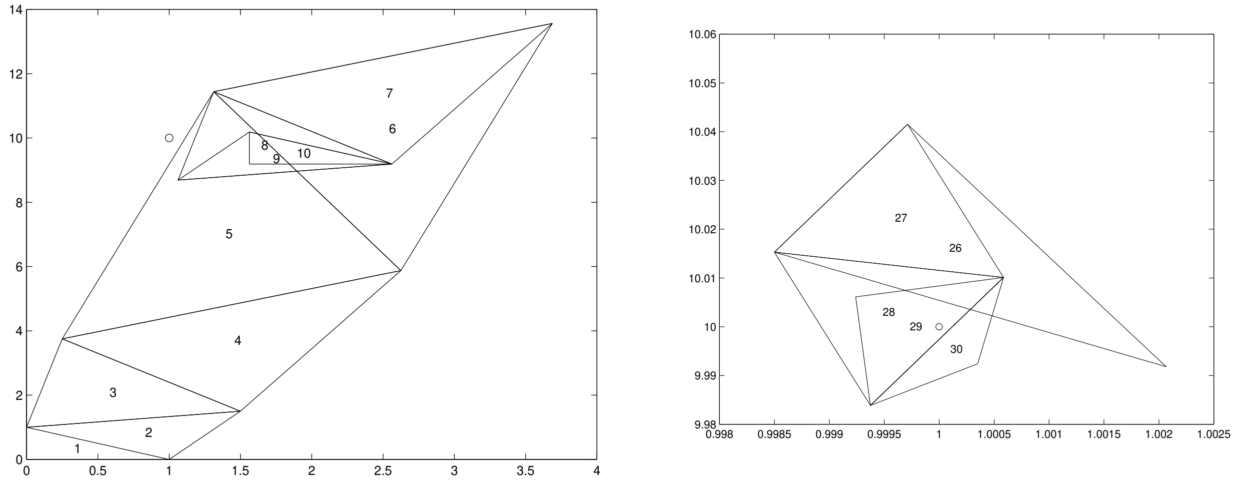

Abb. 2.6. Einige Schritte des Nelder-Mead-Verfahrens für $(x-1)^2 + (y-10)^4$

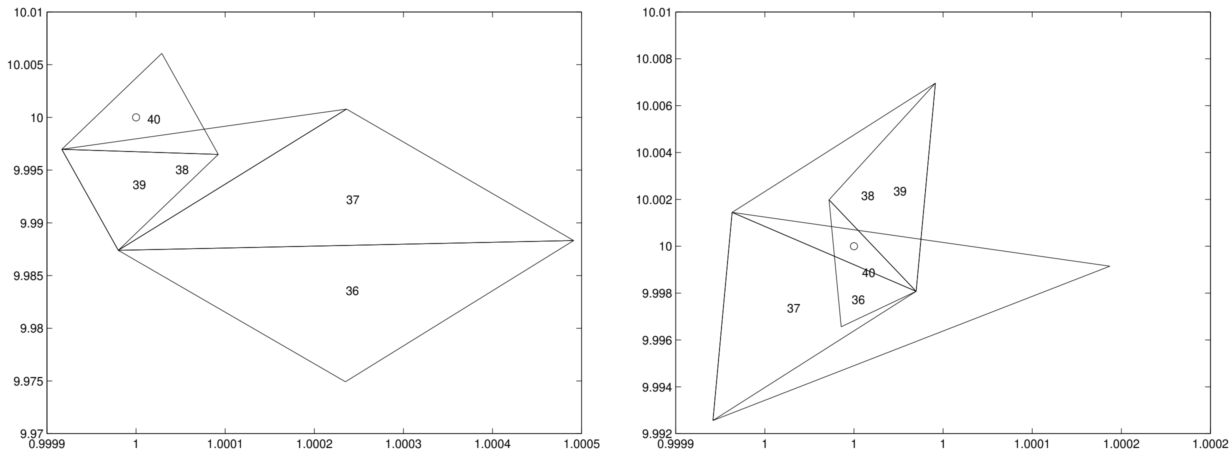

Abb. 2.7. Schritte 36–40 beim Nelder-Mead-Verfahren für $(x-1)^2+(y-10)^4$ mit Startdreieck (2.18) und mit Startdreieck (2.19)

Programm 2.8. Ein Nelder-Mead-matlab-Programm

```
function [lsg,iteration,nfc]=neldermead(fktname,x,iterationmax,c,subroutine)
% ======================================================================
% Eine Implementation der Polyeder-Methode von Nelder-Mead.
% Erstellt fuer die Veranstaltung
% Weiterbildenes Studium fuer Ingenieure WS 98/99
% Helge Baumann, 11.8.1998
% ----------------------------------------------------------------------
% Aurufparameter: (nur die ersten zwei Parameter zwingend)
% fktname : Name der Funktion F, deren Minimum bestimmt
% x : Spaltenvektor der Dimension n oder eine n x (n+1) Matrix.
% iterationmax : Anzahl der maximalen Iterationen. Standard: 100
% c : fuer regelmaessiges Startpolyeder (hier weggelassen)
% subroutine : Unterroutine zum Protokolieren (hier weggelassen)
% ---------------------------------------------------------------------
% Rueckgabeparameter:
% lsg : Loesung (Spaltenvektor!)
% iteration : Anzahl der benoetigten Iterationen
global nfc %: Anzahl der Funktionsauswertungen (in fktname.m gezaehlt)
% ---------------------------------------------------------------------
% Programmvariablen:
% n  : Dimension der Vektorraumes
% m  : Anzahl der uebergebenen Vektoren (entweder m=1 oder m=n+1)
% x  : n x (n+1) Matrix, die n+1 Spaltenvektoren bestimmen das aktuelle
%      Polyeder
```

```
% fx : Vektor der Funktionswerte der n+1 Polyedervektoren
% b  : Index des Vektors mit dem "b"esten Funktionswert
% s  : Index des Vektors mit dem "s"chlechtesten Funktionswert
% z  : Index des Vektors mit dem "z"weitschlechtesten Funktionswert
% alpha, beta, gamma : Parameter der Polyeder-Methode
% epsilon : Abbruchgenauigkeit
% iterationmax : Anzahl der maximalen Iterationen
% abbruchbed : Abschaetzung des Fehlers nach Geiger S. 15
% ======================================================================
epsilon = 1E-10; alpha = 1; beta = 2; gamma = 0.5;
abbruchbed = 1;%   ( damit beim ersten Mal kein Abbruch erfolgt)
iteration = 0; lsg = []; [n,m] = size(x);
%-------------------------------------------------------------
% (Ueberpruefung der Parameter ausgelassen)
% (Startpolyeder ausgelassen )
%-------------------------------------------------------------
[fx] = feval(fktname,x);
% Beginn der Iterationsschleife
while (iteration < iterationmax) & (abbruchbed > epsilon)
   [tt,b]=min(transpose(fx));    % Index des besten Punktes
   [tt,s]=max(transpose(fx));    % Index des schlechtesten Punktes
   [tt,z] =max(fx([1:s-1,s+1:n+1]));
   if z>= s, z=z+1; end          % Index des zweitschlechtesten Punktes
   % Bestimme das Zentrum der n besseren Punkte:
   xquer = (sum(x(:,[1:s-1,s+1:n+1])')' ) / n;
   % Reflektiere den schlechtesten Punkts an xquer:
   xr = xquer + alpha * (xquer - x(:,s));
   [fxr] = feval(fktname,xr);
   % Unterscheidung von 3 Faellen:
   % Ist f(xr) zwischen bestem und zweitschlechtestem Wert, so wird
   % der schlechteste Punkt xs durch xr ersetzt.
   if (fx(b) <= fxr) & (fxr <= fx(z))
      x(:,s) = xr; fx(s) = fxr;
   % Ist f(xr) sogar noch besser als der bislang bekannte beste Wert...
   elseif (fxr < fx(b))
      % ...so koennte in der gleichen Richtung noch mehr zu holen sein.
      xe = xquer + beta*( xr - xquer);
      [fxe] = feval(fktname,xe);
      % Ist die weitere Expansion erfolgreich, so tausche xe ansonsten xr
      % gegen den schlechtesten Punkt xs aus.
      if (fxe < fxr))
```

```
         x(:,s) = xe; fx(s) = fxe;
      else
         x(:,s) = xr; fx(s) = fxr;
      end;
      b = s;
   % auf dem Index des schlechtesten ist nun der beste Wert abgelegt.
   % Ist f(xr) schlechter als der zweitschlechteste Wert ist vermutlich
   % das Polyeder zu gross und eine Kontraktion erforderlich.
   elseif (fx(z) < fxr)
      if fxr >= fx(s)
         xk = xquer + gamma*( x(:,s)-xquer);
      else
         xk = xquer + gamma*( xr - xquer);
      end;
      [fxk] = feval(fktname,xk);
      % Ist f(xk) besser als der schlechteste Wert, so wird xs gegen xk
      % ausgetauscht. Andernfalls wird das Polyeder um den besten Punkt
      % zusammengezogen.
      if (fxk < fx(s))
         x(:,s) = xk; fx(s) = fxk;
      else
         for i=1:n
            g=gamma;
            x(i,[1:b-1,b+1:n+1]) = (1-g) * x(i,[1:b-1,b+1:n+1]) + g * x(i,b);
         end
         % Neuberechnung der Funktionswerte aller Vektoren nach Kontraktion:
         [fx] = feval(fktname,x);
      end;
   end;
   % Abbruchbedingung auswerten:
   fquer = sum(fx) / (n+1);
   abbruchbed = sqrt( sum( (fx-fquer).^2 )/n );
   iteration = iteration + 1;
%Ende der Iterationsschleife
end;
lsg = x(:,b);
```