METODE DE CĂUTARE DIRECTĂ
Algoritmi şi Studii Numerice

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania.
Academy of Romanian Scientists, 54,
Splaiul Independentei, Bucharest 5, Romania.
E-mail: nandrei@ici.ro

În această lucrare vom considera problema de minimizare a funcţiilor $f : R^n \to R$ utilizând metode de căutare directă, prin care înţelegem acele metode care se bazează numai pe evaluarea funcţiei f de-a lungul unui şir de puncte $\{x_k\}$ şi compararea acestor valori cu scopul de a obţine punctul de minim x^*. Cu alte cuvinte, aceste metode nu apelează la construcţia şi evaluarea gradientului funcţiei f, cu atât mai puţin a Hessianului acesteia. Metodele din această clasă sunt utilizate în următoarele circumstanţe: 1) Funcţia f nu este derivabilă. 2) Elaborarea gradientului $\nabla f(x)$ a funcţiei f este foarte complexă. 3) Evaluarea gradientului $\nabla f(x)$ a funcţiei f este mult mai dificilă decât evaluarea funcţiei f. 4) Nu este necesară o soluţie x^* de mare acurateţe.

Problema fundamentală în ceea ce priveşte proiectarea unei metode de căutare directă este următoarea: „*Date punctele* $x_1, x_2, \ldots, x_k$, *precum şi valorile funcţiei de minimizat* $f(x_1), f(x_2), \ldots, f(x_k)$, *atunci să se determine punctul* x_{k+1} *pentru care* $f(x_{k+1}) < f(x_i)$ *pentru toţi* $i = 1, \ldots, k$."

Scopul acestui capitol este de a prezenta câteva soluţii ale acestei probleme sub forma unor algoritmi care s-au dovedit mai interesanţi şi performanţi. Menţionăm faptul că aceste metode sunt dedicate în special rezolvării problemelor cu un număr mic de variabile, pentru care nu cerem soluţii de mare acurateţe. În general, construcţia unei metode directe de căutare este o activitate de imaginaţie, putându-se elabora nenumărate asemenea metode [Torczon, 1991,1997]. O analiză a principalelor caracteristici ale acestor metode, fără a se insista asupra performanţelor lor numerice este dată de Powell [1994, 1998]. Pentru aceste metode demonstrarea unor proprietăţi de convergenţă este foarte delicată, chiar dacă din punct de vedere practic algoritmii asociaţi acestor metode lucrează bine. De aceea cercetările asupra acestor metode nu sunt finalizate cu o concluzie tranşantă. Pentru probleme de mici dimensiuni, cu neliniarităţi comode, algoritmii metodelor de căutare directă deseori au performanţe comparabile cu metodele care utilizează gradientul funcţiei, dar ocazional eşuează lamentabil. Singurul avantaj al acestor metode este uşurinţa în utilizare deoarece tot ceea ce este necesar a se specifica este expresia algebrică a funcţiei de minimizat.

1. CĂUTARE PARALELĂ CU AXELE

Pentru minimizarea funcţiei $f : R^n \to R$, strategia acestei metode este de a fixa $n-1$ variabile la valoarea lor curentă şi de a se căuta minimul în raport cu variabila rămasă liberă. Deoarece minimizarea se face numai după o singură variabilă, problema devine una de căutare liniară şi deci orice metodă prezentată în capitolul 3 din [Andrei, 2009] se poate utiliza în acest scop. După ce s-a executat câte o minimizare corespunzătoare fiecărei dintre variabilele problemei se consideră că s-a încheiat un ciclu şi deci se poate trece la execuţia următorului ciclu, până când un criteriu de oprire a iteraţiilor este îndeplinit. Algoritmul este foarte simplu şi se poate prezenta astfel:

Algoritmul 1.1 (Algoritmul de căutare paralelă cu axele)

Pasul 1.	Iniţializare. Se alege un punct iniţial x_0 şi se pune $i = 1$.
Pasul 2.	Se determină direcţia de căutare sub forma: $$d_i = \begin{cases} [1,0,\cdots,0]^T, i = 1, n+1, 2n+1, \ldots \\ [0,1,\cdots,0]^T, i = 2, n+2, 2n+2, \ldots \\ \qquad \ldots\ldots \\ [0,0,\cdots,1]^T, i = n, 2n, 3n, \ldots. \end{cases}$$
Pasul 3.	Se determină sensul de deplasare pe direcţia d_i. Pentru aceasta se ia un mic pas de probă de lungime μ şi se evaluează $f^+ = f(x_i + \mu d_i)$ şi $f^- = f(x_i - \mu d_i)$. Dacă $f^+ < f(x_i)$, atunci d_i este direcţia corectă de reducere a valorilor funcţiei f. Dacă $f^- < f(x_i)$, atunci se consideră direcţia $-d_i$ ca direcţie de căutare. Dacă $f^+ > f(x_i)$ şi $f^- > f(x_i)$, atunci se consideră x_i ca fiind minimul funcţiei f de-a lungul direcţiei d_i şi se continuă cu pasul 6.
Pasul 4.	Se determină lungimea pasului α_i^* astfel încât $$f(x_i + \alpha_i^* d_i) = \min_{\alpha_i \geq 0} f(x_i + \alpha_i d_i)$$ utilizându-se căutarea liniară exactă sau inexactă.
Pasul 5.	Se calculează noul punct: $x_{i+1} = x_i + \alpha_i^* d_i$.
Pasul 6.	Se execută un test de oprire a iteraţiilor. Dacă, de exemplu, $\lVert x_{i+1} - x_i \rVert \leq \varepsilon_x$ şi $\lvert f(x_{i+1}) - f(x_i) \rvert \leq \varepsilon_f$, unde ε_x şi ε_f sunt două toleranţe de determinare a soluţiei, atunci stop; altfel, se pune $i = 1 + 1$ şi se continuă cu pasul 2. ♦

Metoda este foarte simplă şi uşor de implementat. Simplitatea acesteia provine din concepţia ei şi din faptul că nu utilizează derivatele funcţiei. Teoretic, aceasta se poate aplica pentru determinarea minimului oricărei funcţii continue. Totuşi, dacă funcţia este de tipul unei „văi alungite”, atunci este foarte posibil ca aceasta să se termine într-un punct care nu este minimul problemei.

2. METODA HOOKE-JEEVES, DE CĂUTARE A FORMEI

După cum am văzut, în metoda de căutare paralelă cu axele direcţiile de deplasare sunt fixate. Aceasta face ca metoda să fie lent convergentă. Metoda Hooke-Jeeves [1961] selectează direcţiile de căutare într-o manieră care se adaptează formei funcţiei de minimizat. Pentru aceasta, metoda execută n căutări paralele cu axele şi apoi consideră o căutare de-a lungul direcţiei $s_i = x_i - x_{i-n}$, numită direcţie de căutare a formei, unde x_i este punctul obţinut la sfârşitul celor n căutări paralele cu axele şi x_{i-n} este punctul obţinut înaintea acestor căutări. Deci, metoda Hooke-Jeeves este o procedură de căutare care constă din două tipuri de deplasări. Prima, de explorare, pentru identificarea comportării locale a funcţiei şi a doua de exploatare a formei funcţiei pentru accelerarea convergenţei.

Algoritmul 2.1 (Algoritmul Hooke-Jeeves)

Pasul 1.	Iniţializare. Se consideră un punct iniţial x_1 numit punct de bază şi se aleg lungimile paşilor de deplasare Δx_i, $i = 1, 2, \ldots, n$ de-a lungul celor n axe de coordonate. Se pune $k = 1$.
Pasul 2.	Se calculează $f_k = f(x_k)$. Se pune $y_k^0 = x_k$ şi se execută pasul 3 de explorare a comportării funcţiei în jurul punctului de bază x_k.
Pasul 3.	Pentru $i = 1, 2, \ldots, n$ perturbăm variabila x_i în jurul punctului de bază temporar y_k^{i-1} pentru a obţine un nou punct temporar. Se calculează: $f^+ = f(y_k^{i-1} + \Delta x_i e_i)$, $f^- = f(y_k^{i-1} - \Delta x_i e_i)$ şi $f = f(y_k^{i-1})$, cu care se calculează noul punct temporar: $$y_k^i = \begin{cases} y_k^{i-1} + \Delta x_i e_i, f^+ < f, \\ y_k^{i-1} - \Delta x_i e_i, f^- < f < f^+, \\ y_k^{i-1}, f < min\left\{f^+, f^-\right\}. \end{cases} \quad (2.1)$$
Pasul 4.	Dacă $y_k^n = x_k$, atunci dacă $max\left\{\Delta x_i\right\} < \varepsilon$, stop; altfel, reducem lungimile paşilor de explorare Δx_i (de exemplu pentru fiecare $i = 1, 2, \ldots, n$ punem $\Delta x_i = \Delta x_i / 2$) şi se execută pasul 3. Dacă,

	$y_k^n \neq x_k$, atunci s-a obținut un nou punct de bază $x_{k+1} = y_k^n$ și se continuă cu pasul 5.
Pasul 5.	Utilizând cele două puncte de bază x_k și x_{k+1} se stabilește direcția $s = x_{k+1} - x_k$ de-a lungul căreia se determină punctul $y_{k+1}^0 = x_{k+1} + \alpha^* s$, unde α^* este soluția problemei de minimizare a funcției f de-a lungul direcției s.
Pasul 6.	Se pune $k = k+1$, $f_k = f(y_k^0)$ și utilizând relațiile (2.1) se execută o explorare în jurul punctului y_k^0 pentru a se obține un nou punct y_k^n. Dacă $f(y_k^n) < f(x_k)$, atunci s-a obținut un nou punct de bază $x_{k+1} = y_k^n$ cu care se continuă cu pasul 5. Dacă, pe de altă parte, $f(y_k^n) \geq f(x_k)$, atunci dacă $max\{\Delta x_i\} \leq \varepsilon$, stop; altfel la fel ca în pasul 4 lungimile pașilor Δx_i se reduc, se pune $x_{k+1} = x_k$, $k = k+1$ și se continuă cu pasul 2. ♦

3. METODA POWELL, A DIRECȚIILOR CONJUGATE

Metoda Powell [1964] este o extensie a procedurii de căutare a formei dată de Hooke-Jeeves, pentru care se poate demonstra că este o metodă de direcții conjugate. Pentru clasa funcțiilor pătratice metoda converge la soluție într-un număr finit de pași. Powell a sugerat anumite modificări pentru a se depăși anumite inconveniente și a accelera convergența în cazul funcțiilor nepătratice.

La începutul primei iterații punctul x_1 este dat și direcțiile de căutare sunt $d_j^1 = e_j$, unde e_j sunt direcția axelor de coordonate. La iterația k algoritmul lucrează astfel. Fie $y_0^k = x_k$ și pentru $i = 1,\ldots,n$ din punctul y_{i-1}^k în direcția d_i^k se caută minimul funcției obținându-se astfel punctul y_i^k. Se continuă apoi căutarea din punctul y_n^k în direcția $s_k = y_n^k - x_k$ obținându-se astfel noul punct x_{k+1} cu care reluăm procedura la iterația $k+1$. Deci, direcțiile de căutare la iterația $k+1$ sunt: $d_2^k, d_3^k, \ldots, d_n^k, s_k$. Observăm că direcția d_1^k a fost eliminată, reținându-se direcția s_k care ține seama de forma funcției f. Înainte de începerea celei de-a $k+1$ iterații direcțiile de mai sus sunt rebotezate ca $d_1^{k+1}, \ldots, d_n^{k+1}$.

Vom demonstra că în n iterații metoda Powell generează n direcții conjugate. Într-adevăr, prin construcție fiecare dintre punctele x_2 și y_n^2 minimizează funcția f în direcția $s_1 = d_n^2$, astfel încât la sfârșitul cele-i de-a doua

iterații avem o pereche de direcții conjugate $s_1 = d_n^2$ şi $s_2 = y_n^2 - x_2$. Adică, în notațiile cele-i de-a treia iterații, direcțiile d_{n-1}^3 şi d_n^3 sunt conjugate. Presupunem acum că s-au executat k iterații şi că k direcții $d_{n-k+1}^{k+1}, \ldots, d_n^{k+1}$ sunt conjugate. În fapt, acestea sunt chiar ultimele k direcții de căutare din cadrul iterației k, care sunt de asemenea direcții de căutare şi în iterația $k+1$. Deci, punctul de plecare x_{k+1}, pentru iterația $k+1$, minimizează f în subspațiul care conține aceste direcții. Mai mult, deoarece direcțiile de mai sus sunt conjugate rezultă că punctul y_n^{k+1} obținut la sfârşitul iterației $k+1$ de asemenea minimizează f în acest subspațiu. Deci direcția $s_{k+1} = y_n^{k+1} - x_{k+1}$ este conjugată cu cele k direcții $d_{n-k+1}^{k+1}, \ldots, d_n^{k+1}$. Astfel, după $k+1$ iterații avem exact $k+1$ direcții conjugate. În particular, după n iterații metoda construieşte n direcții conjugate.

Defectul metodei Powell, în forma sa de bază în care a fost prezentată, constă în faptul că la o anumită iterație este posibil ca cele n direcții să devină liniar dependente. Aceasta se poate întâmpla atunci când noua direcție s_k este astfel încât $(y_n^k - y_0^k)^T (y_1^k - y_0^k) = 0$. Pentru a ieşi din această dificultate, Powell a modificat această procedură de bază în sensul de a permite eliminarea la fiecare iterație a altei direcții diferită de d_1^k, astfel încât cele n direcții de căutare să poată fi întotdeauna liniar independente. Procedura modificată a lui Powell se poate prezenta sub forma următorului algoritm.

Algoritmul 3.1 (Algoritmul Powell)

Pasul 1.	Se consideră un punct inițial x_1 şi n direcții $d_i^1 = e_i$, $i = 1, \ldots, n$. Se pune $k = 1$.
Pasul 2.	Se pune $y_0^k = x_k$ şi pentru $i = 1, \ldots, n$ din punctul y_{i-1}^k în direcția d_i^k se determină minimul funcției f, obținându-se astfel punctul y_i^k.
Pasul 3.	Se execută testul de oprire a iterațiilor. De exemplu, dacă $\left\| y_n^k - x_k \right\| \le \varepsilon$, stop; altfel se continuă cu pasul 4.
Pasul 4.	Se calculează $$\Delta = \max_{i=1,\ldots,n} \left\{ f(y_{i-1}^k) - f(y_i^k) \right\} = f(y_{q-1}^k) - f(y_q^k),$$ adică q este valoarea indicelui i care maximizează Δ.
Pasul 5.	Se defineşte $f_1 = f(y_0^k)$, $f_2 = f(y_n^k)$ şi se evaluează $f_3 = f(2y_n^k - y_0^k)$.
Pasul 6.	Dacă $f_3 \ge f_1$ sau $$(f_1 - 2f_2 + f_3)(f_1 - f_2 - \Delta)^2 \ge \frac{1}{2}\Delta(f_1 - f_3)^2,$$

	atunci la iterația următoare se utilizează vechile direcții d_i^k, $i=1,\ldots,n$, adică se consideră $y_0^{k+1}=y_n^k$, se pune $k=k+1$ şi se continuă cu pasul 2; altfel se execută pasul 7.
Pasul 7.	Se execută o căutare liniară în direcția $s_k=y_n^k-x_k$, obținându-se astfel punctul x_{k+1}, se consideră direcțiile $d_1^k,\ldots,d_{q-1}^k,d_{q+1}^k,\ldots,d_n^k,s_k$ se pune $k=k+1$ şi se continuă cu pasul 2. ♦

4. METODA ROSENBROCK, A ROTIRII COORDONATELOR

Metoda de rotire a coordonatelor, dată de Rosenbrock [1960], poate fi şi ea considerată ca o extensie a metodei Hooke-Jeeves. Aceasta constă din mai multe etape, în cadrul fiecăreia sistemul de coordonate fiind rotit astfel încât prima axă a acestuia să fie orientată către direcția local estimată a minimului, toate celelalte axe fiind bineînțeles ortogonale pe aceasta. Metoda Rosenbrock funcționează astfel.

1) Se alege un punct inițial x_1 de unde începe căutarea şi se calculează $f(x_1)$.

2) Se aleg lungimile paşilor de deplasare h_i de-a lungul direcțiilor d_i^1, $i=1,\ldots,n$, care inițial sunt paralele cu axele de coordonate.

3) Din punctul x_1 se execută o deplasare în direcția d_1^1 de lungime h_1 pentru a se obține punctul x_2. Dacă $f(x_2)>f(x_1)$, atunci acest pas este un *eşec*. În acest caz lungimea pasului pentru următoarea iterație, în aceeaşi direcție, va fi luată ca $-\beta$ înmulțit cu lungimea curentă a pasului, unde $0<\beta<1$. Pe de altă parte, dacă $f(x_2)\le f(x_1)$, atunci punctul obținut reprezintă un *succes*, caz în care lungimea pasului în aceeaşi direcție, de la iterația următoare, va li luată ca $+\alpha$ înmulțit cu lungimea curentă a pasului, unde $\alpha>1$. Valorile recomandate de Rosenbrock sunt $\alpha=3$ şi $\beta=0,5$. Continuăm acest proces de căutare cu mărirea sau micşorarea lungimii paşilor de deplasare h_i până când s-a obținut *cel puțin* câte un succes şi câte un eşec de-a lungul fiecărei din direcțiile d_i^1, unde $i=1,\ldots,n$. Aceasta reprezintă o etapă a procesului de minimizare în metoda Rosenbrock. O nouă etapă începe ori de câte ori s-a obținut cel puțin câte un succes şi un eşec de-a lungul fiecărei direcții.

4) Pentru a începe următoarea etapă, vom modifica direcțiile d_i^1 în d_i^2, $i=1,\ldots,n$, astfel încât d_1^2 să se afle aproximativ în direcția celei mai rapide descreşteri ale valorilor funcției f, iar celelalte direcții să fie ortogonale pe aceasta. Noile direcții d_i^2 sunt obținute din cele vechi printr-o procedură de

ortogonalizare Gram-Schmidt. Geometric, noua direcție d_1^2 va fi chiar linia care unește punctul de la începutul etapei de optimizare cu punctul obținut la sfârșitul acestei etape și reprezintă în fapt suma vectorială a tuturor deplasărilor cu succes. Astfel, direcția d_1^2 este cea mai promițătoare direcție pentru reducerea valorilor funcției f.

5) Cu aceste elemente executăm o nouă etapă a procesului de optimizare printr-o procedură similară cu cea descrisă mai sus în pasul 3 luând ca punct de bază cel mai bun punct obținut la sfârșitul etapei anterioare.

6) Procedura de mai sus se execută până când distanța dintre punctul de început și cel de sfârșit a unei etape este mai mică decât o toleranță $\varepsilon > 0$ suficient de mică.

Cu acestea, metoda Rosenbrock se poate prezenta sub forma următorului algoritm.

Algoritmul 4.1 (Algoritmul Rosenbrock)

Pasul 1.	Se consideră un punct inițial x_1, numit punct de bază și se aleg direcțiile d_i^1, $i = 1,\ldots,n$, de obicei paralele cu axele; lungimile pașilor de deplasare h_i, $i = 1,\ldots,n$, precum și constantele α și β ($\alpha = 3$, $\beta = 0,5$). Se pune $k = 1$.
Pasul 2.	Din punctul x_k se consideră un pas de lungime h_1 în direcția d_1^k. Dacă pasul reprezintă un succes, adică $f(x_k + h_1 d_1^k) < f(x_k)$, atunci punem $h_1 = \alpha h_1$, și se reține noul punct obținut ca punct de bază. Dacă, acest pas este un eșec, adică $f(x_k + h_1 d_1^k) \geq f(x_k)$, atunci punem $h_1 = -\beta h_1$.
Pasul 3.	Se continuă această căutare din cel mai bun punct obținut de-a lungul direcțiilor $d_2^k, d_3^k, \ldots, d_n^k, d_1^k, d_2^k, \ldots, d_n^k, \ldots$ până când pentru fiecare din cele n direcții d_i^k, $i = 1,\ldots,n$, au fost obținute cel puțin cât un succes și câte un eșec.
Pasul 4.	Se calculează noile direcții de deplasare d_i^{k+1}, $i = 1,\ldots,n$, necesare următoarei etape utilizând procedura de ortogonalizare Gram-Schmidt. Se calculează matricea $$H = \begin{bmatrix} H_1 & & & \\ H_2 & H_2 & & \\ \cdots & \cdots & & \\ H_n & H_n & \cdots & H_n \end{bmatrix}, \qquad (4.1)$$ unde H_j este suma algebrică a tuturor lungimilor pașilor cu succes în direcția d_j^k, $j = 1,\ldots,n$. Dacă $\lvert H_i \rvert < \varepsilon$ pentru toți $i = 1,\ldots,n$, atunci

stop; altfel se calculează direcțiile liniar independente $p_1,\ldots,p_n \in R^n$ sub forma

$$P=[p_1,\cdots,p_n]=[d_1^k,\cdots,d_n^k]H. \qquad (4.2)$$

Astfel, p_1 reprezintă vectorul care uneşte punctul de început cu cel de sfârşit obținut după secvența de căutări de la etapa a $k-$a. Vectorul p_2 este suma algebrică a lungimii paşilor cu succes în toate direcțiile exceptând prima, etc. Aceşti vectori liniar independenți $p_1,\ldots,p_n$ sunt utilizați pentru a genera o nouă mulțime de direcții ortogonale prin procedura Gram-Schmidt.

Presupunem că au fost alese i direcții $q_1,q_2,\ldots,q_i$ pentru etapa $k+1$ a procesului de optimizare. Cu acestea definim matricea

$$Q_i=[q_1,\ldots,q_i], \qquad (4.3)$$

şi calculăm următoarea direcție q_{i+1} ca:

$$q_{i+1}=p_{i+1}-Q_i r_{i+1}, \qquad (4.4)$$

unde r_{i+1} este un vector ales astfel încât q_{i+1} să fie ortogonal pe toate direcțiile $q_1,q_2,\ldots,q_i$. Înmulțind la stânga (4.4) cu q_j^T, pentru $j=1,\ldots,i,$ obținem

$$q_j^T q_{i+1}=q_j^T p_{i+1}-q_j^T Q_i r_{i+1}=0, \qquad (4.5)$$

pentru $j=1,\ldots,i.$ În formă matriceală relațiile (4.5) devin:

$$Q_i^T q_{i+1}=Q_i^T p_{i+1}-Q_i^T Q_i r_{i+1}=0,$$

adică

$$[Q_i^T Q_i]r_{i+1}=Q_i^T p_{i+1}. \qquad (4.6)$$

Deci

$$r_{i+1}=[Q_i^T Q_i]^{-1}Q_i^T p_{i+1}. \qquad (4.7)$$

Cu acestea, din (4.4) obținem:

$$q_{i+1}=T_i p_{i+1}, \qquad (4.8)$$

unde

$$T_i=I-Q_i[Q_i^T Q_i]^{-1}Q_i^T, \qquad (4.9)$$

care este chiar o matrice de proiecție.

Deoarece

$$Q_i^T Q_i=[q_1,\ldots,q_i]^T[q_1,\ldots,q_i]=I_i,$$

relația (4.8) devine:

$$q_{i+1}=(I-Q_iQ_i^T)p_{i+1}=p_{i+1}-\sum_{j=1}^{i}(p_{i+1}^T q_j)q_j, \qquad (4.10)$$

pentru $i=0,1,\ldots,n-1.$

	Cu acestea, pentru generarea noilor direcții d_i^{k+1}, $i=1,\ldots,n$ se procedează astfel: a) Se calculează matricea P ca în (4.2). b) Se pune $q_1 = p_1$ şi $d_1^{k+1} = q_1 / \lVert q_1 \rVert$. c) Pentru $i = 1, 2, \ldots, n-1$ se calculează: $$q_{i+1} = p_{i+1} - \sum_{j=1}^{i} (p_{i+1}^T d_j^{k+1}) d_j^{k+1},$$ $$d_{i+1}^{k+1} = q_{i+1} / \lVert q_{i+1} \rVert.$$
Pasul 5.	Se consideră cel mai bun punct obținut în etapa k drept punct de bază, se pun lungimile h_i, $i=1,\ldots,n$ la valorile lor originale, $k = k+1$ şi se continuă cu pasul 2. ♦

5. METODA NELDER-MEAD, A DEPLASĂRII SIMPLEXULUI

După cum se ştie, figura geometrică formată din $n+1$ puncte x_i, $i=1,\ldots,n+1$, care nu se află toate într-un hiperplan, împreună cu orice combinație convexă a acestora se numeşte *simplex* în R^n. Cele $n+1$ puncte x_i se numesc *vârfurile* simplexului. Când toate punctele se află la distanțe egale unele de altele, simplexul se zice că este *regulat*. Astfel, în R^2 simplexul este un triunghi, în R^3 este un tetraedru etc.

Ideea de bază a acestei metode precizată de Spendley, Hext şi Himsworth [1962] şi îmbunătățită de Nelder şi Mead [1965] este de a compara valorile funcției f în cele $n+1$ vârfuri ale simplexului şi apoi de a deplasa (rostogoli) simplexul către punctul de optim. Deplasarea este realizată utilizând trei tipuri de mişcări, cunoscute ca: *reflexie*, *contracție* şi *dilatare*. Să le prezentăm.

Reflexia. Dacă x_h este vârful corespunzător celei mai mari valori a funcției f, atunci reflectând acest punct referitor la fața opusă lui, ne aşteptăm să obținem un nou punct x_r în care valoarea funcției să se reducă. Dacă acesta este cazul, atunci putem considera un nou simplex prin eliminarea lui x_h şi includerea lui x_r în mulțimea vârfurilor simplexului. Cu acest nou simplex putem proceda ca mai sus, deplasând astfel căutarea în direcția minimizării funcției f. Dacă funcția nu prezintă o „vale adâncă”, atunci aplicarea repetată a acestui proces de reflexie ne va conduce, ce-i drept în zig-zag, către punctul de minim al problemei. Matematic punctul reflectat x_r este dat de:

$$x_r = (1+\alpha)x_c - \alpha x_h, \tag{5.1}$$

unde x_h este vârful simplexului corespunzător valorii maxime a funcției f în raport cu toate vârfurile, adică

$$f(x_h) = \max_{i=1,\ldots,n+1} \{f(x_i)\}, \tag{5.2}$$

iar x_c este centroidul vârfurilor x_i, $i = 1, 2, \ldots, n+1$, $i \neq h$, dat de:

$$x_c = \frac{1}{n} \sum_{i=1; i \neq h}^{n+1} x_i, \tag{5.3}$$

$\alpha > 0$ fiind coeficientul de reflexie definit ca:

$$\alpha = \frac{\|x_r - x_c\|}{\|x_h - x_c\|}. \tag{5.4}$$

Astfel, x_r se va afla pe linia care unește x_h cu x_c, în care $\|x_r - x_c\| = \alpha \|x_h - x_c\|$. Dacă $f(x_r)$ se află între $f(x_h)$ şi $f(x_m)$, unde x_m este vârful corespunzător valorii minime a funcției f, adică

$$f(x_m) = \min_{i=1,\ldots,n+1} \{f(x_i)\}, \tag{5.5}$$

atunci x_h este înlocuit cu x_r şi astfel am obținut un nou simplex cu care putem relua procesul de optimizare.

Procedând aşa este foarte posibil să ajungem în unele situații dificile. De exemplu, dacă un simplex ajunge într-o „vale" a funcției în care $f(x_r) = f(x_h)$, atunci prin reflexie ne vom întoarce le vechiul simplex, apărând deci fenomenul de ciclare. Această situație se poate depăşi prin eliminarea din simplexul curent a vârfului corespunzător celei de-a doua valori maxime a funcției f imediat după valoarea $f(x_h)$. În general, această strategie conduce la apropierea simplexului de punctul de minim al lui f, dar este foarte posibil ca simplexul final din nou să cicleze în jurul lui x^*, sau să se afle la o distanță egală cu ordinul de mărime față de x^*. În această situație putem considera ca punct de minim al funcției f acel vârf care apare în toate simplexurile generate prin reflexie. Dacă se doreşte o precizie mai mare, atunci simplexul trebuie contractat sau redus în dimensiune, după cum vom vedea în cele ce urmează.

Dilatarea. Dacă procesul de reflexie construieşte punctul x_r pentru care $f(x_r) < f(x_m)$, adică se obține un nou minim, atunci în general ne aşteptăm ca valorile funcției f să se reducă de-a lungul liniei care unește x_c cu x_r. Deci vom deplasa x_r în x_e prin relația:

$$x_e = \gamma x_r + (1 - \gamma) x_c, \tag{5.6}$$

unde γ este coeficientul de dilatare, definit ca:

$$\gamma = \frac{\|x_e - x_c\|}{\|x_r - x_c\|} > 1. \tag{5.7}$$

Dacă $f(x_e) < f(x_m)$, atunci înlocuim vârful x_h cu x_e şi reluăm procesul de reflexie. Dacă, pe de altă parte, $f(x_e) > f(x_m)$, atunci dilatarea este un eşec şi deci vom înlocui x_h cu x_r după care reluăm procesul de reflexie.

Contracția. Dacă procesul de reflexie furnizează un punct x_r pentru care $f(x_r) > f(x_i)$, pentru toți $i = 1, \ldots, n+1$ cu excepția $i = h$ şi $f(x_r) < f(x_h)$, atunci înlocuim vârful x_h cu x_r. În acest caz vom contracta simplexul construind:

$$x_o = \beta x_h + (1-\beta) x_c, \tag{5.8}$$

unde β este coeficientul de contracție ($0 \le \beta \le 1$) definit ca:

$$\beta = \frac{\|x_o - x_c\|}{\|x_h - x_c\|}. \tag{5.9}$$

Dacă $f(x_r) > f(x_h)$, atunci vom utiliza (5.8) pentru construcția punctului x_o. Dacă procesul de contracție a generat punctul x_o pentru care

$$f(x_o) < min\{f(x_h), f(x_r)\},$$

atunci înlocuim punctul x_h cu x_o şi reluăm procesul de reflexie. Dacă, pe de altă parte,

$$f(x_o) \ge min\{f(x_h), f(x_r)\},$$

atunci toate punctele x_i sunt înlocuite cu punctele $(x_i + x_m)/2$, după care reluăm procesul de reflexie cu acest nou simplex.

Dacă deviația standard a funcției f în cele $n+1$ vârfuri ale simplexului curent este mai mică decât o valoare prescrisă, pozitivă, suficient de mică, adică:

$$\left(\frac{1}{n+1} \sum_{i=1}^{n+1} (f(x_i) - f(x_c))^2 \right)^{1/2} \le \varepsilon, \tag{5.10}$$

atunci căutarea minimului se consideră încheiată. Nelder şi Mead sugerează valorile: $\alpha = 1$, $\beta = 0,5$ şi $\gamma = 2$.

Cu acestea algoritmul de căutare directă Nelder-Mead pentru minimizarea funcției f se poate prezenta sub forma următoare. Fie x_1 o estimație inițială a lui x^* şi considerăm vârfurile simplexului inițial $x_1, \ldots, x_{n+1}$, unde $x_{j+1} = x_1 + h_j e_j$, în care scalarii h_j sunt aleşi astfel încât cantitățile $\left| f(x_1 + h_j e_j) - f(x_1) \right|$ să fie pe cât posibil egale. Pentru simplexul curent definim:

x_h - punctul pentru care $f(x_h) = \max\limits_{i=1,\ldots,n+1} \{f(x_i)\}$,

x_s - punctul pentru care $f(x_s) = \max\limits_{i=1,\ldots,n+1; i \ne h} \{f(x_i)\}$,

x_m - punctul pentru care $f(x_m) = \min_{i=1,\ldots,n+1} \{f(x_i)\}$,

x_c - centroidul simplexului $x_c = \frac{1}{n} \sum_{i=1; i \neq h}^{n+1} x_i$.

Algoritmul 5.1 (Algoritmul Nelder-Mead)

Pasul 1.	Se aleg vârfurile $x_1, \ldots, x_{n+1}$ ale simplexului inițial şi se calculează $f(x_i),\ i = 1, \ldots, n+1$. Se determină punctele x_h, x_s, x_m şi x_c.
Pasul 2.	Se testează criteriul (5.10) de oprire a iterațiilor.
Pasul 3.	*Reflexia.* Se calculează punctul $x_r = (1+\alpha)x_c - \alpha x_h$.
Pasul 4.	Dacă $f(x_m) \le f(x_r) \le f(x_s)$, atunci înlocuim x_h cu x_r şi continuăm cu pasul 2.
Pasul 5.	*Dilatarea.* Dacă $f(x_r) < f(x_m)$, atunci dilatăm simplexul calculând punctul $x_e = \gamma x_r + (1-\gamma)x_c$. a) Dacă $f(x_e) < f(x_m)$, atunci înlocuim x_h cu x_e şi continuăm cu pasul 2. b) Dacă $f(x_e) \ge f(x_m)$, atunci înlocuim x_h cu x_r şi continuăm cu pasul 2.
Pasul 6.	*Contracția.* Dacă $f(x_r) > f(x_s)$, atunci contractăm simplexul. a) Dacă $f(x_r) < f(x_h)$, atunci calculăm $x_o = \beta x_r + (1-\beta)x_c$. b) Dacă $f(x_r) \ge f(x_h)$, atunci calculăm $x_o = \beta x_h + (1-\beta)x_c$. c) Dacă $f(x_o) < f(x_h)$ şi $f(x_o) < f(x_r)$, atunci înlocuim x_h cu x_o şi continuăm cu pasul 2. d) Dacă $f(x_o) \ge f(x_h)$ sau $f(x_o) > f(x_r)$, atunci reducem mărimea simplexului considerând $h_i = h_i / 2,\ i = 1, \ldots, n+1$ şi continuăm cu pasul 2. ♦

Extensii ale metodei simplex în ceea ce priveşte introducerea optimizării unidimensionale, ponderării centroidului sau utilizarea optimizării ciclice de tip mixt au fost date de Dumitru [1975]. O reconsiderare a metodelor de căutare directă, în particular a metodei Nelder-Mead, este dată de Wright [1996] şi Kelley [1999].

Metodele de căutare directă, după cum se vede, încearcă o serie de strategii de determinare a formei funcției de minimizat şi de utilizare a acesteia în procesul de calcul a minimului. *Toate aceste metode au o încărcătură empirică care le fac vulnerabile la pretențiile de optimizare a funcțiilor cu număr mare de variabile sau atunci când se doreşte determinarea minimului cu o acurateţe mărită. De aceea în aplicațiile curente aceste nu sunt utilizate.* Totuşi pentru cazurile în care pentru

funcția dată calculul gradientului este o problemă şi nu se doreşte decât o soluție aproximativă, atunci se pot încerca aceste metode.

Exemplul 5.1.
Să considerăm următoarea funcție cu două variabile:

$$f(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times$$
$$[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$$

care are un minim în punctul $x^* = [0, -1]^T$ cu valoarea funcției $f(x^*) = 3$. Calculul gradientului acestei funcții este o complicație, de aceea vom încerca metodele de căutare directă prezentate mai sus plecând din punctul inițial $x_0 = [1, -0.5]^T$, pentru care $f(x_0) = 436.035156$.

Tabelul 5.1 prezintă numărul de iterații (#iter), numărul de evaluări ale funcției (#eval), timpul de calcul (cpu) în secunde, punctul de minim şi valoarea funcției furnizate de algoritmii Hook-Jeeves, Powell, Rosenbrock şi Nelder-Mead în care căutarea liniară este implementată prin metodele înainte-înapoi, secțiunea de aur, Fibonacci, interpolare pătratică în trei puncte [Andrei, 2009].

Tabelul 5.1. Rezultate furnizate de algoritmii de căutare directă. $\varepsilon = 10^{-7}$.

	#iter	#eval	cpu(s)	x^*	$f(x^*)$
Hooke-Jeeves & secțiunea de aur	15	1253	28.97	(0,-1)	3.0
Hooke-Jeeves & Fibonacci	15	1211	29.45	(0,-1)	3.0
Hooke-Jeeves & interpolare pătratică	22	579	13.82	(0,-1)	3.0
Hooke-Jeeves & căutare înainte-înapoi	14	878	20.79	(0,-1)	3.0
Hooke-Jeeves & lungimea pasului egală cu 1	8	195	4.74	(0,-1)	3.0
Powell & secțiunea de aur	12	2442	56.53	(1.8,0.2)	84.0
Powell & Fibonacci	11	1919	46.75	(1.8,0.2)	84.0
Powell & interpolare pătratică	1	33	0.85	(1,-0.3346)	94.649
Powell & căutare înainte-înapoi	1	153	3.72	(0.74,-0.5)	76.292
Powell & lungimea pasului egală cu 1	12	2442	56.44	(1.8,0.2)	84.0
Rosenbrock	45	489	11.18	(0,-1)	3.0
Nelder-Mead	56	165	8.05	(0,-1)	3.0

Observăm că toate metodele au cam acelaşi comportament, excepție făcând metoda Powell care nu reuşeşte să localizeze minimul funcției cu acuratețea dată.

Studiu numeric (Performanța algoritmilor de căutare directă)[1]

Pentru a vedea performanța algoritmilor de căutare directă considerăm următorul studiu numeric în care este vorba de minimizarea a 50 de funcții de test cu numărul de variabile egal cu 5 sau 10. Deci în total s-au rezolvat 100 de probleme de minimizare. Pentru a ilustra performanța algoritmilor, se consideră numai acele probleme pentru care diferența dintre valorile minime ale funcției este mai mică decât 10^{-2}. În figura 5.1 se prezintă profilurile de performanță pentru aceşti algoritmi în funcție de numărul de evaluări ale funcției de minimizat.

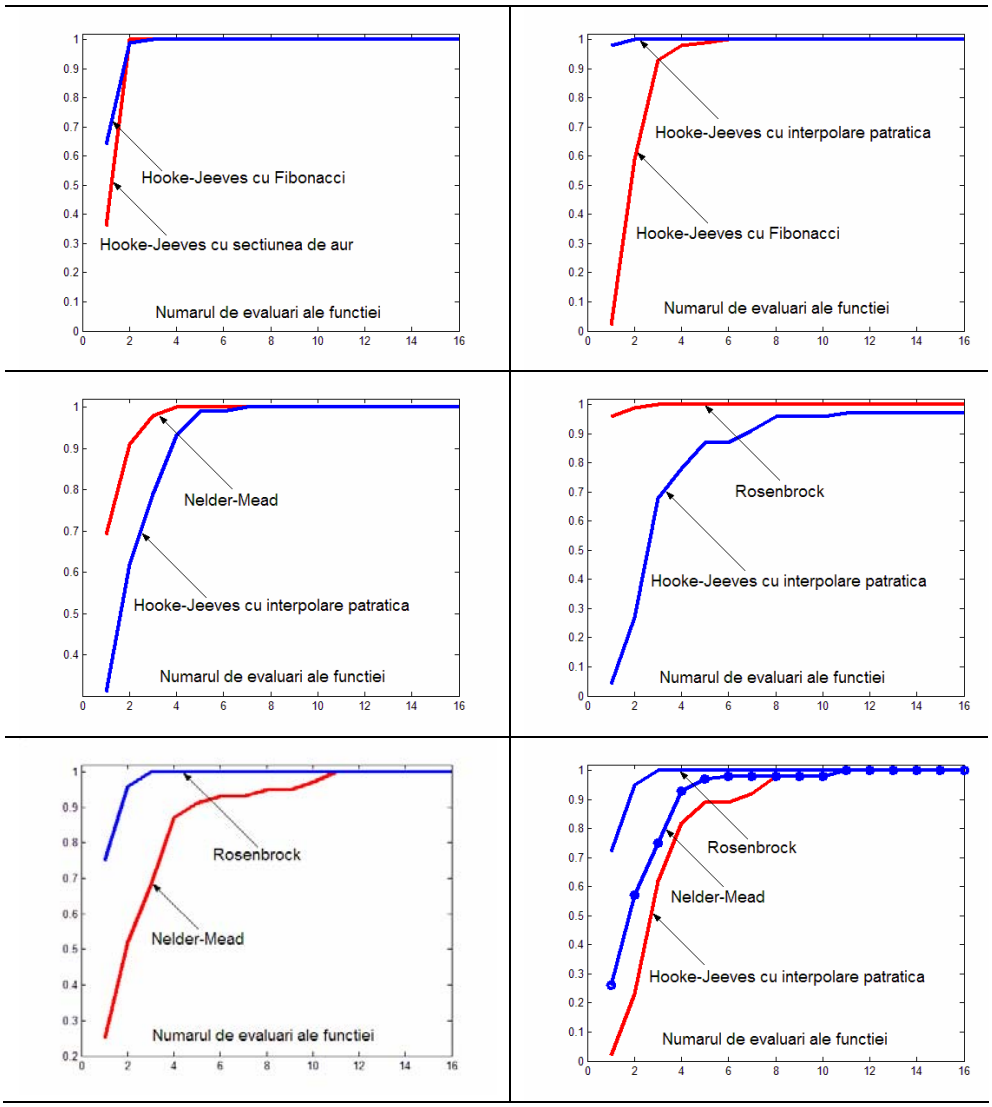

Fig. 1. Profilurile de performanță ale algoritmilor de căutare directă.

[1] Programul UNO.FOR implementează algoritmii de căutare directă consideraţi în acest studiu numeric. Programul UNOINT.FOR este o variantă interactivă a lui UNO.FOR

Observăm că din punctul de vedere al metricii dată de numărul de evaluări ale funcției de minimizat, metoda Rosenbrock pare a fi cea mai bună. Metoda Powell, în implementarea noastră, nu rezistă la comparații în acest studiu, fiind mult prea lentă pentru a fi reprezentată în profilurile de performanță Dolan-Moré.

Studiu numeric (Nelder-Mead versus Rosenbrock)

Din studiul numeric de mai sus se vede că algoritmii Nelder-Mead a deplasării simplexului şi Rosenbrock a rotirii coordonatelor sunt cei mai performanți. De aceea în acest studiu vom prezenta performanța acestora în ceea ce priveşte rezolvarea problemelor de optimizare fără restricții cu un număr de variabile ceva mai mare. Considerăm aceleaşi set de funcții de test ca mai sus din care selectăm şi minimizăm 60, pentru care numărul de variabile este egal cu 4, 8, 12, 16 şi 20. Deci, în total s-au minimizat 300 de funcții de test. Comparația dintre metoda Nelder-Mead şi Rosenbrock se face în aceeaşi manieră, prin reținerea numai a acelor probleme (din cele 300) pentru care diferența dintre valorile optime (locale) ale funcțiilor date de fiecare metodă este mai mică decât 10^{-2}. Din cele 300, numai 183 de probleme satisfac acest criteriu. În figura 5.2. se prezintă profilurile de performanță Dolan-Moré ale celor doi algoritmi în funcție de numărul de evaluări ale funcției de minimizat.

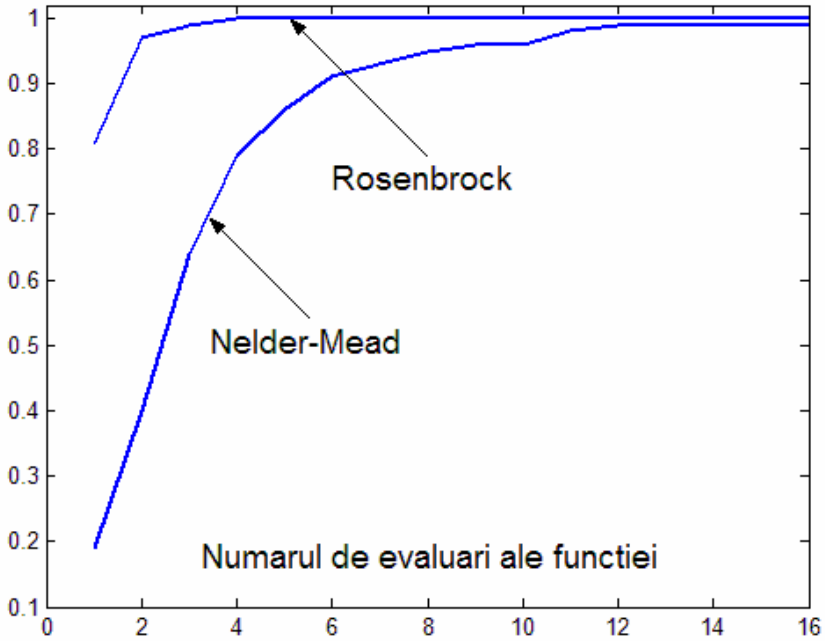

Fig. 5.2. Nelder-Mead versus Rosenbrock.

Din acest studiu numeric, în care se rezolvă probleme cu un număr de variabile ceva mai mare (20), rezultă că metoda Rosenbrock este mai performantă şi mai robustă decât metoda Nelder-Mead.

6. METODA POWELL UOBYQA, A INTERPOLĂRII PĂTRATICE

M.J.D. Powell

Înainte de trece la prezentarea metodei UOBYQA (Unconstrained Optimization BY Quadratic Approximation), elaborată de Powell [1994, 2002], să parcurgem principalele clase de metode directe de optimizare. În practica optimizării care utilizează metodele directe se identifică următoarele patru clase. În prima clasă putem include metodele de căutare a formei funcţiei de minimizat în jurul punctului curent. Metodele prezentate în secţiunile anterioare din acest capitol (Hooke-Jeeves, Rosenbrock, şi Nelder-Mead) aparţin acestei clase. Acestea sunt bazate pe o explorare a spaţiului variabilelor utilizând puncte ale unor tipare geometrice în care se evaluează funcţia de minimizat. Acestea nu utilizează netezimea funcţiei şi deci cer un număr foarte mare de evaluări ale funcţiei. O analiză a acestor metode se găseşte în [Kolda, Lewis şi Torzcon, 2003]. A doua clasă de metode directe sunt cele introduse de Powell [1964] şi care generează direcţii conjugate cuplate cu căutarea liniară (vezi secţiunea 3). Combinarea tehnicilor de diferenţe finite cu metodele quasi-Newton constituie a treia clasă de metode directe precizate de Moré şi Sorensen [1983]. În fine, ultima clasă de metode directe sunt bazate pe modelarea funcţiei obiectiv ca o funcţie pătratică, obţinută prin interpolare numai din valorile funcţiei, cuplată cu o tehnică a regiunii de încredere. Aceste metode au fost introduse de Winfield [1969, 1973] şi recent reconsiderate de Powell [1994, 2002].

În această secţiune vom prezenta metoda UOBYQA care aparţine clasei de metode directe bazate pe construcţia unui model pătratic obţinut prin interpolarea valorilor funcţiei de minimizat într-un număr dat de puncte. Fie deci problema $min\ f(x)$, unde $x \in R^n$, împreună cu un punct iniţial x_0 din care de demarează calculele şi două valori ρ_i şi ρ_f care mărginesc raza ρ a regiuni de încredere în care este valabil modelul pătratic sintetizat numai din valorile funcţiei f. Valorile razei de încredere sunt alese de algoritm astfel încât $\rho_i \le \rho \le \rho_f$. Utilizând valorile funcţiei într-un număr de puncte se construieşte modelul pătratic

$$Q(x) = c_Q + g_Q^T(x - x_0) + \frac{1}{2}(x - x_0)^T G_Q(x - x_0), \quad x \in R^n, \qquad (6.1)$$

care este obținut prin interpolarea valorilor funcției. Parametrii modelului pătratic sunt scalarul real c_Q, vectorul $g_Q \in R^n$ şi $n \times n$ matricea simetrică G_Q. Deci spațiul liniar al polinoamelor pătratice de la R^n la R are dimensiunea $m = \frac{1}{2}(n+1)(n+2)$. Ca atare, parametrii lui Q sunt definiți printr-un sistem de ecuații de interpolare de forma:

$$Q(x_i) = f(x_i), \quad i = 1, \ldots, m, \qquad (6.2)$$

în care punctele x_i, $i = 1, \ldots, m$, sunt generate de algoritm astfel încât modelul (6.1) să fie bine definit şi să asigure stabilitatea numerică a procesului de calcul. Interpolarea în (6.2) se face prin intermediul polinoamelor de interpolare Lagrange. Pentru $j = 1, \ldots, m$ a $j-$a funcție Lagrange este polinomul pătratic $l_j : R^n \to R$ cu proprietățile

$$l_j(x_i) = \delta_{ij}, \quad i = 1, \ldots, m, \qquad (6.3)$$

unde δ_{ij} este simbolul delta al lui Kronecker. Deci

$$l_j(x) = c_j + g_j^T(x - x_0) + \frac{1}{2}(x - x_0)^T G_j(x - x_0), \qquad (6.4)$$

care este asemănător cu ecuația (6.1). Din (6.2) şi (6.3) rezultă că

$$Q(x) = \sum_{j=1}^{m} f(x_j) l_j(x). \qquad (6.5)$$

Deci, din (6.4) avem următoarele valori pentru modelul pătratic (6.1):

$$c_Q = \sum_{j=1}^{m} f(x_j) c_j, \quad g_Q = \sum_{j=1}^{m} f(x_j) g_j, \quad G_Q = \sum_{j=1}^{m} f(x_j) G_j. \qquad (6.6)$$

Cu acestea, modelul pătratic este utilizat într-o schemă a regiunii de încredere, adică se determină $d \in R^n$ ca o estimație a soluției problemei

$$min \left\{ Q(x_k + d) : \|d\| \le \Delta \right\}, \qquad (6.7)$$

unde k este un întreg din $[1, m]$ pentru care $f(x_k)$ este cea mai mică valoare dintre valorile $f(x_i)$, $i = 1, \ldots, m$. Aici norma utilizată este norma Euclidiană, iar Δ este o altă rază a regiunii de încredere care satisface condiția $\Delta \ge \rho$. Introducerea lui Δ are avantajul de a permite modificarea lungimilor variabilelor care depăşesc valoarea lui ρ. Modificarea lui Δ se face într-o manieră tipică algoritmului regiunii de încredere.

La fiecare iterație algoritmul rezolvă două probleme de tipul regiunii de încredere. Una dintre acestea este dată de (6.7), cealaltă de:

$$max \left\{ \left| l_j(x_k + d) \right| : \|d\| \le \rho \right\}, \qquad (6.8)$$

unde l_j este funcția Lagrange (6.4). Utilizarea lui (6.8) menține nesingularitatea ecuațiilor de interpolare (6.2). După ce modelul pătratic s-a format, ceea ce implică m evaluări ale funcției f, se generează un nou punct utilizând una dintre cele două probleme ale regiunii de încredere (6.7) sau (6.8). Pentru rezolvarea problemei (6.7), algoritmul Powell aplică metoda regiunii de încredere a lui Moré-Sorensen [1983]. Observăm că

$$Q(x_k + d) = Q(x_k) + h_Q^T d + \frac{1}{2} d^T G_Q d, \tag{6.9}$$

unde $h_Q = g_Q + G_Q(x_k - x_0)$. Condițiile KKT (Karush-Kuhn-Tucker) pentru (6.7) arată că d trebuie să fie o soluție a sistemului

$$(G_Q + \theta I)d = -h_Q, \tag{6.10}$$

unde I este matricea identitate de ordin n, iar θ un parametru nenegativ ales astfel încât $G_Q + \theta I$ să fie o matrice pozitiv definită sau pozitiv semi-definită. Dacă θ este pozitiv, atunci evident $\|d\| = \Delta$. Ca atare, metoda Moré-Sorensen poate cere rezolvarea sistemului algebric (6.10) pentru mai multe valori ale lui θ. Pentru aceasta Powell construieşte o transformare ortogonală care face matricea G_Q tridiagonală, pentru care o estimație a celei mai mici valori proprii λ_Q este calculată. Fie acum $d = d(\theta)$ soluția sistemului (6.10) pentru orice θ pentru care matricea $G_Q + \theta I$ este pozitiv definită. Atunci, $\|d(\theta)\|$ descreşte monoton când θ creşte şi $d = d(\theta)$ este acceptat dacă θ satisface condiția

$$\frac{1}{\|d(\theta)\|} = \frac{1}{\Delta}. \tag{6.11}$$

Dar, pentru $\theta > -\lambda_Q$, vedem că $1/\|d(\theta)\|$ este o funcție de θ cu următoarele proprietăți: este monoton crescătoare, concavă şi prima derivată este mărginită inferior de $1/\|h_Q\|$. Algoritmul UOBYQA utilizează aceste proprietăți pentru rezolvarea ecuației neliniare (6.11) în privința lui θ printr-o tehnică de căutare înainte-înapoi.

Pentru rezolvarea problemei (6.8) se poate aplica de două ori procedura de mai sus, deoarece în acest caz în (6.7) se poate considera $Q = l_j$ şi apoi $Q = -l_j$ cu $\Delta = \rho$. Totuşi această abordare este ineficientă deoarece mai multe valori ale lui j trebuie încercate (din cele $n^2/2$ posibile) pentru a găsi $|l_j(x_k + d)|$ suficient de mare. De aceea, UOBYQA utilizează o procedură care maximizează $|l_j(x_k + d)|$ referitor la $\|d\| \le \rho$, numai în $O(n^2)$ operații algebrice. Întregul j din subproblema (6.8) întotdeauna este diferit de k, deci condițiile (6.3) includ cazul $l_j(x_k) = 0$. Ca atare, ținând seama de (6.4) şi deoarece vectorul

$h_j = g_j + G_j(x_k - x_0)$ este calculat, atunci din (6.9) rezultă că subproblema se poate scrie sub forma:

$$max\left\{\left|h_j^T d + \frac{1}{2} d^T G_j d\right| = \left|l_j(x_k + d)\right| : \|d\| \le \rho\right\} \qquad (6.12)$$

Acum, deoarece în metoda regiunii de încredere întotdeauna d se poate înlocui cu $-d$ rezultă că problema de mai sus este echivalentă cu

$$max\left\{\left|h_j^T d\right| + \frac{1}{2}\left|d^T G_j d\right| : \|d\| \le \rho\right\}. \qquad (6.13)$$

Dacă $\hat{d}$ şi $\tilde{d}$ sunt valorile lui d care maximizează $\left|h_j^T d\right|$ şi respectiv $\left|d^T G_j d\right|$, referitor la condiţia $\|d\| \le \rho$, atunci d poate fi o estimaţie adecvată a soluţiei problemei (6.12) dacă acesta se alege între $\pm\hat{d}$ şi $\pm\tilde{d}$ care dă cea mai mare valoare a funcţiei obiectiv a problemei. Într-adevăr, pentru orice d admisibil (care satisface condiţia $\|d\| \le \rho$), Powell stabileşte următoarea margine:

$$\left|h_j^T d\right| + \frac{1}{2}\left|d^T G_j d\right| \le \left|h_j^T \hat{d}\right| + \frac{1}{2}\left|\tilde{d}^T G_j \tilde{d}\right| + \left|h_j^T \tilde{d}\right| + \frac{1}{2}\left|\hat{d}^T G_j \hat{d}\right|$$

$$\le 2\,max\left\{\left|h_j^T \hat{d}\right| + \frac{1}{2}\left|\hat{d}^T G_j \hat{d}\right|, \left|h_j^T \tilde{d}\right| + \frac{1}{2}\left|\tilde{d}^T G_j \tilde{d}\right|\right\}. \qquad (6.14)$$

Cu acestea, $\hat{d}$ este selectat ca vectorul $\pm\rho h_j / \|h_j\|$, în timp ce $\tilde{d}$ este generat printr-o tehnică care utilizează coloanele matricei G_j într-o manieră proprie metodei puterii pentru calculul celei mai mare valori proprii a lui G_j.

Odată vectorii $\hat{d}$ şi $\tilde{d}$ selectaţi, algoritmul construieşte d ca o combinaţie liniară a acestor vectori, dar alegerea nu este restricţionată la $\pm\hat{d}$ sau $\pm\tilde{d}$ cum s-a făcut în (6.14), ci se determină vectorii $\hat{u}$ şi $\tilde{u}$ din spaţiul liniar generat de $\hat{d}$ şi $\tilde{d}$ care satisfac condiţiile $\hat{u}^T\tilde{u} = 0$ şi $\hat{u}^T G_j \tilde{u} = 0$, care în esenţă este o problemă de valori proprii. Cu acestea, d se alege ca $d = \hat{u}\cos\phi + \tilde{u}\sin\phi$, unde ϕ se alege astfel încât să se realizeze valoarea maximă a funcţiei

$$\left|l_j(x_k + d)\right| = \left|h_j^T d + \frac{1}{2} d^T G_j d\right|$$

$$= \left|h_j^T \hat{u}\cos\phi + h_j^T \tilde{u}\sin\phi + \frac{1}{2}\hat{u}^T G_j \hat{u}\cos^2\phi + \frac{1}{2}\tilde{u}^T G_j \tilde{u}\sin^2\phi\right| \qquad (6.15)$$

cu $0 \le \phi \le 2\pi$. Valoarea lui ϕ care maximizează această expresie se poate determina prin tehnici de interpolare pătratică, dar în UOBYQA Powell alege ϕ ca multipli întregi de $\pi/4$. Astfel, valoarea finală a lui $\left|l_j(x_k + d)\right|$ este cantitatea:

$$max\left[\left|h_j^T\hat{u}\right|+\frac{1}{2}\left|\hat{u}^T G_j\hat{u}\right|,\left|h_j^T\tilde{u}\right|+\frac{1}{2}\left|\tilde{u}^T G_j\tilde{u}\right|,\right.$$
$$\left.2^{-1/2}\left(\left|h_j^T\hat{u}\right|+\left|h_j^T\tilde{u}\right|\right)+2^{-2}\left|\hat{u}^T G_j\hat{u}+\tilde{u}^T G_j\tilde{u}\right|\right]. \qquad (6.16)$$

Această cantitate nu este cu mult mai mică decât valoarea maximă a funcției (6.15). Powell arată că raportul dintre (6.16) şi valoarea maximă a lui (6.15) este mărginit inferior de $(12+2\sqrt{2})/17 \approx 0.872$, care este destul de aproape de 1. Testele numerice intensive, cu diferite funcții de test, efectuate de Powell arată că această procedură de determinare a unei soluții a problemei (6.8) care implică numai $O(n^2)$ operații aritmetice, este foarte expeditivă şi aproximează foarte bine soluția lui (6.8).

Următoarea etapă a algoritmului este aceea a alegerii unui punct de interpolare care se modifică cu soluția uneia din problemele de mai sus. La începutul unei iterații se dispune de: punctele curente de interpolare x_i, $i=1,\ldots,m$; vectorul x_0 din (6.1) şi (6.4); gradientul g_j şi Hessianul G_j al fiecărei funcții de interpolare Lagrange l_j, $j=1,\ldots,m$; gradientul g_Q şi Hessianul G_Q asociate modelului pătratic curent care satisfac ecuațiile de interpolare (6.2). De asemenea se dispune de valorile razelor regiunii de încredere Δ şi ρ, unde $\Delta \geq \rho$, precum şi de întregul k care intră în (6.7) şi care corespunde celei mai mici valori $f(x_i)$, $i=1,\ldots,m$.

Dacă d este o soluție a lui (6.7) cu $\Delta=\rho$, dar $f(x_k+d)>f(x_k)$, atunci ρ trebuie să fie redus dacă Q furnizează o bună aproximație a lui f în domeniul $\{x:\|x-x_k\|\leq\rho\}$. Totuşi, dacă una sau mai multe distanțe $\|x_i-x_k\|$, $i=1,\ldots,m$, sunt mai mari decât 2ρ, atunci sigur Q nu constituie o bună aproximare a funcției f. În această situație un punct de interpolare, x_j este modificat în domeniul $\{x:\|x-x_k\|\leq\rho\}$, înainte ca raza ρ să fie redusă. Powell [2000] demonstrează că: *dacă funcția este de trei ori diferențiabilă cu derivata de ordinul trei mărginită de o constantă M, atunci diferența dintre modelul pătratic şi funcția obiectiv satisface condiția*

$$\left|Q(x)-f(x)\right|\leq\frac{1}{6}M\sum_{i=1}^{m}\left|l_i(x)\right|\left\|x-x_i\right\|^3. \qquad (6.17)$$

Acest rezultat pare numai o estimație academică deoarece presupunerea făcută asupra derivatei de ordinul trei (care aminteşte de funcțiile autoconcordante) este foarte restrictivă şi constanta M nu este disponibilă. Totuşi, Powell utilizează (6.17) într-o manieră constructivă care reduce numărul de iterații în UOBYQA. Parametrul $M/6$ este o constantă nenegativă care este actualizat la fiecare iterație ca:

$$max\left\{M/6, \left|Q(x_k+d)-f(x_k+d)\right| / \sum_{i=1}^{m}\left|l_i(x_k+d)\right|\left\|x_k+d-x_i\right\|^3\right\}.$$

Actualizarea lui Δ se face în funcție de valoarea raportului

$$r=\frac{f(x_k)-f(x_k+d)}{Q(x_k)-Q(x_k+d)}, \tag{6.18}$$

în maniera următoare:

$$\Delta=\begin{cases} max\left\{\Delta, 5\|d\|/4, \rho+\|d\|\right\}, & r\geq 0.7, \\ max\left\{\Delta/2, \|d\|\right\}, & 0.1<r<0.7, \\ \|d\|/2, & r\leq 0.1. \end{cases} \tag{6.19}$$

Pe de altă parte, valoarea lui ρ este modificată conform regulii:

$$\rho=\begin{cases} \rho_f, & \rho_f<\rho\leq 16\rho_f, \\ \sqrt{\rho\rho_f}, & 16\rho_f<\rho\leq 250\rho_f, \\ \rho/10, & \rho>250\rho_f. \end{cases} \tag{6.20}$$

UOBYQA conține o inițializare foarte interesantă. Cele m puncte de interpolare sunt selectate astfel. Se pune $x_1=x_0$ şi $x_{2j}=x_0+\rho_i e_j$, $j=1,\ldots,n$, unde e_j este al j-lea vector unitar din R^n. Powell consideră σ_j ca -1 dacă $f(x_{2j})\geq f(x_0)$ şi respectiv $+1$ dacă $f(x_{2j})<f(x_0)$ şi aplică formula:

$$x_{2j+1}=\begin{cases} x_0-\rho_i e_j, & \sigma_j=-1, \\ x_0+2\rho_i e_j, & \sigma_j=+1, \end{cases} \qquad j=1,\ldots,n. \tag{6.21}$$

În continuare se consideră $i(p,q)$ cu valorile:

$$i(p,q)=2n+1+p+\frac{1}{2}(q-1)(q-2), \quad 1\leq p<q\leq n,$$

iar celelalte puncte inițiale de interpolare sunt plasate pe pozițiile

$$x_{i(p,q)}=x_0+\rho_i(\sigma_p e_p+\sigma_q e_q), \quad 1\leq p<q\leq n. \tag{6.22}$$

Procedura de inițializare pune k drept cel mai mic întreg din $[1,m]$ astfel încât $f(x_k)$ este cea mai mică valoare dintre valorile $f(x_i)$, $i=1,\ldots,m$, şi consideră $M/6=0$, $\rho=\rho_i$ şi $\Delta=\rho_i$ cu care se demarează calculele. Această alegere a punctelor inițiale de interpolare face ca determinarea primului model pătratic (6.1) din condițiile de interpolare (6.2) să fie foarte uşoară. Într-adevăr, faptul că $x_1=x_0$ determină $c_Q=f(x_1)$. Apoi, deoarece pentru $j=1,\ldots,n$, $x_{2j}-x_0$ şi $x_{2j+1}-x_0$ sunt multipli diferiți de zero ai vectorului e_j permite ca $(g_Q)_j$ şi $(G_Q)_{jj}$ să fie determinați din diferențele $f(x_{2j})-f(x_0)$ şi $f(x_{2j+1})-f(x_0)$, unde $(g_Q)_j$ este

a $j-$ a componentă a lui g_Q, iar $(G_Q)_{pq}$ este elementul (p,q) din matricea G_Q. Elementele nediagonale ale matricei simetrice G_Q sunt calculate din identitatea:

$$c_Q + \rho_i[\sigma_p(g_Q)_p + \sigma_q(g_Q)_q] + \frac{1}{2}\rho_i^2[(G_Q)_{pp} + 2\sigma_p\sigma_q(G_Q)_{pq} + (G_Q)_{qq}] = f(x_{i(p,q)}), \quad (6.23)$$

pentru $1 \le p < q \le n$, obținută din (6.1), (6.2) şi (6.22). În acelaşi timp, toți coeficienții nenuli din polinoamele de interpolare Lagrange se pot calcula în $O(n^2)$ operații aritmetice, deoarece pentru $1 \le p < q \le n$, funcțiile $l_{i(p,q)}$ au o formă foarte simplă:

$$l_{i(p,q)}(x) = (\sigma_p\sigma_q / \rho_i^2)(x - x_0)_p(x - x_0)_q, \quad 1 \le p < q \le n, \quad (6.24)$$

ceea ce arată că $(G_{i(p,q)})_{pq} = \sigma_p\sigma_q / \rho_i^2$ este singurul coeficient nenul al lui $l_{i(p,q)}$. Coeficienții nenului din polinoamele de interpolare Lagrange l_k, $k = 2,\ldots,2n+1$, se determină din polinoamele pătratice:

$$\hat{l}_{2j}(x) = \frac{(x - x_0)_j(x - x_{2j+1})_j}{(x_{2j} - x_0)_j(x_{2j} - x_{2j+1})_j}, \quad (6.25a)$$

$$\hat{l}_{2j+1}(x) = \frac{(x - x_0)_j(x - x_{2j})_j}{(x_{2j+1} - x_0)_j(x_{2j+1} - x_{2j})_j}, \quad (6.25b)$$

pentru $j = 1,\ldots n$.

Cu acestea o prezentare informală a algoritmului UOBYQA se poate face după cum urmează. Detalii se găsesc în Powell [2000, 2001, 2002], precum şi în programul Fortran care implementează acest algoritm.

Algoritmul 6.1 (Algoritmul UOBYQA - Powell)

Pasul 1.	Inițializare. Se consideră vectorul $x_0 \in R^n$, parametrii ρ_i şi ρ_f, precum şi $M/6 = 0$, $\rho = \rho_i$, $\Delta = \rho_i$ şi se pune $j = 0$. Se determină cele m puncte de interpolare x_i, $i = 1,\ldots,m$, precum şi indicele k corespunzător celui mai mic întreg din mulțimea $1,\ldots,m$ pentru care $f(x_k) = min\{f(x_i) : i = 1,\ldots,m\}$.
Pasul 2.	Dacă $j = 0$, atunci utilizând procedura descrisă mai sus (vezi (6.10)) se rezolvă problema (6.7) care furnizează pasul d.
Pasul 3.	Se calculează $f(x_k + d)$, $Q(x_k + d)$ şi valoarea polinoamelor de interpolare Lagrange $l_i(x_k + d)$, $i = 1,\ldots,m$. Se actualizează $M/6$.
Pasul 4.	Dacă $j = 0$, atunci utilizând (6.19) se actualizează Δ. Se alege t ca acel i pentru care expresia $\lvert l_i(x_k + d)\rvert max\left\{1, \lVert x_i - x_k\rVert^3 / \rho^3\right\}$ este

	maximă şi $l_t(x_k + d) \neq 0$. Dacă, pe de altă parte, $j > 0$, atunci se pune $t = j$.
Pasul 5.	Dacă $t > 0$, atunci se deplasează x_t în poziția $\tilde{x}_t = x_k + d$. Se recalculează coeficienții polinoamelor de interpolare Lagrange şi ai modelului pătratic.
Pasul 6.	Dacă $t > 0$ şi cel puțin una din condițiile $j > 0$, $f(\tilde{x}_t) < f(x_k)$, $\|d\| > 2\rho$ şi $\|x_t - x_k\| > 2\rho$ este îndeplinită, atunci se pune $j = 0$ şi se continuă cu pasul 2.
Pasul 7.	Se alege j din mulțimea $J = \{i : \|x_i - x_k\| > 2\rho\}$ pentru care $$\|x_j - x_k\| = max\{\|x_i - x_k\| : i \in J\}. \qquad (6.26)$$ Totuşi, dacă $(M/6)\|x_j - x_k\|^3 \|l_j(x_k + d)\| \leq \varepsilon$, unde $\varepsilon = \rho^2 \lambda_Q / 2$ în care λ_Q este cea mai mică valoare proprie a matricei G_Q, atunci se elimină j din mulțimea J şi se determină un alt j care verifică (6.26). Dacă J este mulțimea vidă, atunci se pune $j = 0$.
Pasul 8.	Dacă $j > 0$, sau dacă $j = 0$ şi $\|d\| > \rho$, atunci se execută pasul 2.
Pasul 9.	Dacă $\rho > \rho_f$, atunci se actualizează ρ ca în (6.20), Δ ca în (6.19) şi se pune $x_0 = x_0 + x_k$. Dacă $\rho = \rho_f$, atunci stop; altfel se continuă cu pasul 2. ♦

Deoarece fiecare coeficient al fiecărui polinom de interpolare Lagrange este relevant, rezultă că complexitatea algoritmului în paşii 3 şi 5 este de ordinul $O(n^4)$. Aceasta reduce foarte mult posibilitățile algoritmului de a rezolva probleme de mari dimensiuni. Powell prezintă anumite scheme de simplificare a acestor paşi, dar cu toate acestea performanțele algoritmului sunt limitate la minimizarea funcțiilor cu 20 de variabile.

Algoritmul este complicat, conține foarte multă euristică care ține seama atât de profunzimile metodei regiunii de încredere, cât şi de anumite observații şi caracteristici ale minimizării funcțiilor, determinate prin numeroase experimentări numerice controlate.

Exemplul 6.1.
Să considerăm funcția de două variabile:

$$f(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times$$
$$[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$$

care are un minim în punctul $x^* = [0, -1]^T$ cu valoarea funcției $f(x^*) = 3$ şi pe care am minimizat-o cu algoritmii de căutare directă prezentați în acest capitol cu

rezultatele în tabelul 5.1. Din punctul inițial $x_0 = [1, -0.5]^T$ algoritmul UOBYQA al lui Powell furnizează soluția $x^* = [0, -1]^T$, cu valoarea funcției $f(x^*) = -3$, în 6 iterații şi 52 de evaluări ale funcției. Vedem că pentru acest exemplu, metoda de căutare prin interpolare pătratică sugerată de Powell este de departe superioară.

Studiu numeric (UOBYQA versus Rosenbrock)
La fel ca în studiul numeric anterior, în care am comparat metodele Rosenbrock şi Nelder-Mead, aici considerăm aceleaşi 60 de funcții de test cu numărul de variabile egal cu 4, 8,12, 16 şi 20 pe care le minimizăm cu UOBYQA a lui Powell[2]. Deci ca mai sus s-au efectuat 300 de experimente numerice în aceleaşi condiții computaționale. În toate acestea funcțiile au fost inițializate în acelaşi punct. Comparația dintre metoda UOBYQA şi metoda Rosenbrock a rotirii coordonatelor se face în aceeaşi manieră, prin reținerea numai a acelor probleme pentru care diferența dintre valorile optime locale ale funcțiilor date de fiecare metodă în parte este mai mică decât 10^{-2}. Din cele 300 de probleme numai 238 satisfac acest criteriu. În figura 6.1 se prezintă profilurile de performanță ale celor doi algoritmi în funcție de numărul de evaluări ale funcției de minimizat.

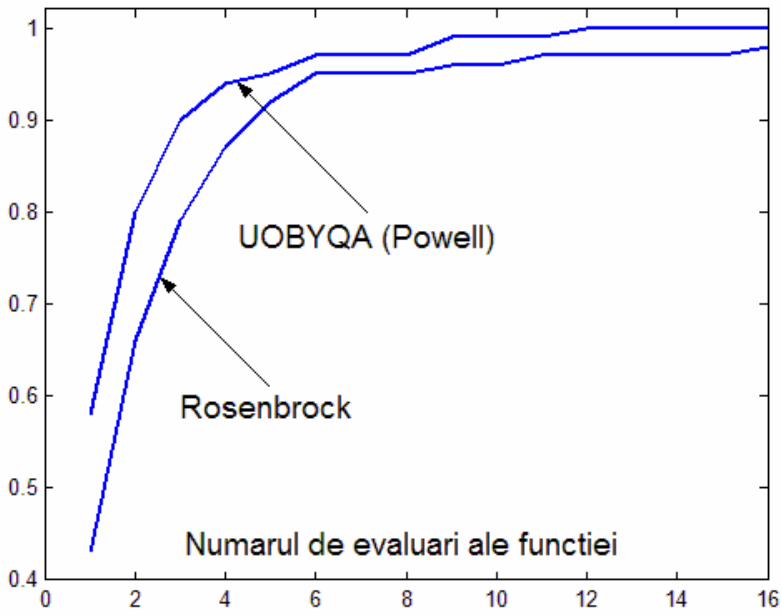

Fig. 6.1. UOBYQA versus Rosenbrock.

[2] Programul UOBYQA a fost elaborat de Powell, căruia autorul îi mulțumeşte pentru amabilitatea de ia-l fi trimis în vederea efectuării testelor numerice din această lucrare.

În raport cu metrica considerată, dată de numărul de evaluări ale funcţiei, observăm imediat că metoda UOBYQA a lui Powell este superioară metodei Rosenbrock a rotirii coordonatelor. În acelaşi timp aceasta este mai robustă. Aceste două metode rămân, cel puţin până acum, metodele directe de bază în ceea ce priveşte minimizarea funcţiilor prin căutare directă.

Studiu numeric (UOBYQA versus Nelder-Mead)

În acest studiu numeric comparăm metoda UOBYQA a lui Powell cu metoda Nelder-Mead a deplasării simplexului. Acest studiu a fost făcut în aceleaşi condiţii ca mai sus, adică s-au considerat aceleaşi funcţii de test iniţializate în acelaşi punct, pentru care numărul de variabile a fost crescut constant fiind egal cu 4, 8, 12, 16 şi 20. Ca mai sus, în ambii algoritmi, acurateţea de calcul a soluţiei a fost fixată la 10^{-5}. Algoritmul Nelder-Mead funcţionează mult mai greu decât UOBYQA. De aceea numărul de iteraţii a fost limitat la 1000. Aceasta face ca din cele 300 de probleme rezolvate numai 199 să satisfacă criteriul conform căruia numai acele probleme pentru care diferenţa dintre valorile optime ale funcţiilor este mai mică decât 10^{-2} să fie considerate în determinarea profilului de performanţă. Figura 6.2 arată profilurile de performanţă ale acestor algoritmi în funcţie de numărul de evaluări ale funcţiei de minimizat.

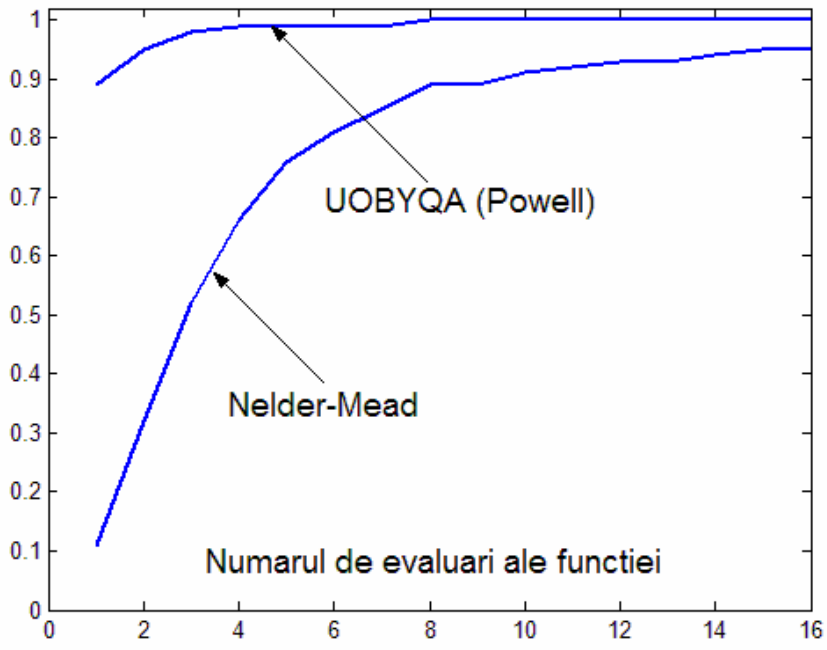

Fig. 6.2. UOBYQA versus Nelder-Mead.

Figura 6.2 ilustrează superioritatea metodei de interpolare pătratică care utilizează numai valorile funcției, față de metoda deplasării simplexului. Conceptual aceste metode sunt foarte diferite. Deplasarea simplexului caută forma locală a funcției şi face un număr de reflexii, dilatări şi contracții ale simplexului pentru a găsi minimul. Interpolarea pătratică este mai rafinată. Aceasta introduce un model local pătratic al funcției, care surprinde mai bine comportarea locală a funcției, şi beneficiază de tehnologia dată de metoda regiunii de încredere.

Exemple şi aplicații (UOBYQA)
În cele ce urmează vom considera câteva exemple şi aplicații care ilustrează capabilitățile metodei UOBYQA a lui Powell. În general aplicațiile sunt de mici dimensiuni şi soluția se calculează cu acuratețea de 10^{-6}.

Exemplul 6.2 (***Funcția HAS 64***) [Brown şi Bartholomew-Biggs, 1987a, pg.27].
Să considerăm funcția:

$$f(x) = 5x_1 + 50000/x_1 + 20x_2 + 72000/x_2 + 10x_3 + 144000/x_3 + \tau\Big((1 - 4/x_1 - 32/x_2 - 120/x_3 - x_4^2)^2 + (x_1 - 10^{-5} - x_5^2)^2 + (x_2 - 10^{-5} - x_6^2)^2 + (x_3 - 10^{-5} - x_7^2)^2\Big),$$

$\tau = 10000$, cu punctul inițial: $x_0 = [1,1,1,-10,-10,-10,-10]^T$. Algoritmul UOBYQA în implementarea dată de Powell furnizează soluția:

$$x^* = [106.283332,\ 79.409508,\ 186.328060,\ 0.000174,\ -10.309380,\ 8.911200,\ -13.650203]^T$$

în 6 iterații şi 12451 evaluări ale funcției, pentru care $f(x^*) = 6204.481878$.

Aplicația D1 (***Analiza reacției enzimelor***)[3] [Andrei, 2001, pg.266], [Andrei, 2003, pg.62].
Se propune următoarea funcție care exprimă reacția unor enzime:

$$f(x) = \sum_{i=1}^{11}\left(y_i - \frac{x_1(u_i^2 + u_i x_2)}{u_i^2 + u_i x_3 + x_4}\right)^2,$$

unde măsurătorile experimentale furnizează următoarele valori:

i	y_i	u_i	i	y_i	u_i
1	0.1957	4.000	7	0.0456	0.125
2	0.1947	2.000	8	0.0342	0.100
3	0.1735	1.000	9	0.0323	0.0833
4	0.1600	0.500	10	0.0235	0.0714
5	0.0844	0.250	11	0.0246	0.0625
6	0.0627	0.167			

[3] Programul ENZIME.FOR implementează această aplicație.

Considerând punctul inițial $x_0 = [0.25, 0.39, 0.415, 0.39]^T$, atunci algoritmul UOBYQA furnizează următoarea soluție:

$$x^* = [0.192800,\ 0.191282,\ 0.123056,\ 0.136062]^{\mathrm{T}},$$

pentru care $f(x^*) = 0.0003075$, în 6 iterații şi 200 de evaluări ale funcției.

Aplicația D2 (***Analiza rezistenței unui termistor în funcție de temperatură***)[4] [Andrei, 2001, pg.266], [Andrei, 2003, pg.62].
Pentru această aplicație se propune următoarea funcție:

$$f(x) = \sum_{i=1}^{16} \left(y_i - x_1 \exp\left(\frac{x_2}{45 + 5i + x_3} \right) \right)^2,$$

unde măsurătorile au următoarele valori:

i	y_i	i	y_i
1	34780	9	8261
2	28610	10	7030
3	23650	11	6005
4	19630	12	5147
5	16370	13	4427
6	13720	14	3820
7	11540	15	3307
8	9744	16	2872

Considerând punctul inițial $x_0 = [0.01, 6100, 340]^T$, pentru care valoarea funcției de minimizat este $f(x_0) = 0.233591E + 10$, atunci algoritmul UOBYQA în varianta dată de Powell furnizează soluția

$$x^* = [0.006197,\ 6098.460236,\ 342.422208]^{\mathrm{T}}$$

în care $f(x^*) = 175.310366$, în 4 iterații şi 135 evaluări ale funcției, cu o acuratețe mai mică decât 10^{-6}.

Aplicația D3 (***Analiza spectroscopiei solare***)[5] [Andrei, 2001, pg.272], [Andrei, 2003, pg.68]. Modelul matematic al acestei aplicații este:

$$f(x) = \sum_{i=1}^{13} \left(x_1 + x_2 \exp\left(-\frac{(i + x_3)^2}{x_4} \right) - y_i \right)^2,$$

unde măsurătorile au următoarele valori:

[4] Programul TERMISTOR implementează această aplicație
[5] Programul SOLAR.FOR implementează această aplicație.

i	y_i	i	y_i
1	0.5	8	2.5
2	0.8	9	1.6
3	1	10	1.3
4	1.4	11	0.7
5	2	12	0.4
6	2.4	13	0.3
7	2.7		

Pentru punctul inițial $x_0 = [1,1,1,1]^T$, algoritmul UOBYQA furnizează următoarea soluție: $x^* = [1.353846,\ 1.331025,\ 1.402794,\ 0.234658]^T$, cu $f(x^*) = 8.3123$ în 6 iterații şi 238 evaluări ale funcției.

Aplicația D4 (***Plasarea optimă în plan a unei facilități – Problema Weber***)[6] [Andrei, 2003, pg.58], [Maranas, Floudas, 1994].
Aplicația se referă la plasarea optimă în plan a unei facilități în aşa fel încât să se minimizeze suma ponderată a distanțelor Euclidiene ale acesteia la un număr dat de puncte din plan. Pentru cazul

$$f(x) = 2\sqrt{(x_1-2)^2+(x_2-42)^2} + 4\sqrt{(x_1-90)^2+(x_2-11)^2} - 5\sqrt{(x_1-43)^2+(x_2-88)^2},$$

algoritmul UOBYQA inițializat în punctul $x_0 = [1,1]^T$, furnizează soluția $x^* = [90,11]^T$ în 6 iterații şi 145 evaluări ale funcției, pentru care $f(x^*) = -264.453141$. Problema este dificil de rezolvat. În punctul de minim gradientul funcției nu este definit.

Aplicația D5 (***Proiectarea unui circuit electric***)[7] [Ratschek şi Rokne, 1993].
În aplicația S1 din [Andrei, 2009], capitolul 6, am prezentat modelul matematic al proiectării unui circuit electric utilizând ecuațiile Ebers-Moll. Ecuațiile Ebers-Moll [Ebers şi Moll, 1954] descriu relațiile dintre curentul din colector I_C şi căderea de tensiune dintre bază şi emitor V_{BE} sub forma

$$I_C = I_S\left(\exp\left(\frac{eV_{BE}}{kT}\right)-1\right),$$

unde I_S este curentul de saturație, $e = 1.6\times10^{-19}$ C este unitatea de sarcină electrică elementară, $k = 1.38\times10^{-23}$ J/K este constanta Boltzmann, iar T este temperatura absolută (Kelvin).

[6] Programul WEBER.FOR implementează această aplicație.
[7] Programul CIRCUIT.FOR implementează această aplicație.

Modelul matematic reprezintă problema proiectării unui circuit electric utilizând teoria Ebers – Moll cu următoarea formulare:

$$
\begin{aligned}
&(1-x_1x_2)x_3\left\{\exp\left[x_5(g_{1k}-g_{3k}x_7 10^{-3}-g_{5k}x_8 10^{-3})\right]-1\right\} \\
&\qquad\qquad +g_{4k}x_2-g_{5k}=0, \qquad\qquad k=1,\ldots,4, \\
&(1-x_1x_2)x_4\left\{\exp\left[x_6(g_{1k}-g_{2k}-g_{3k}x_7 10^{-3}-g_{4k}x_9 10^{-3})\right]-1\right\} \\
&\qquad\qquad +g_{4k}-g_{5k}x_1=0, \qquad\qquad k=1,\ldots,4, \\
&x_1x_3-x_2x_4=0,
\end{aligned}
$$

unde

$$
g=\begin{bmatrix}
0.4850 & 0.7520 & 0.8690 & 0.9820 \\
0.3690 & 1.2540 & 0.7030 & 1.4550 \\
5.2095 & 10.0677 & 22.9274 & 20.2153 \\
23.3037 & 101.7790 & 111.4610 & 191.2670 \\
28.5132 & 111.8467 & 134.3884 & 211.4823
\end{bmatrix}
$$

Problema constă în rezolvarea unui sistem de ecuații algebrice neliniare, care se transformă foarte simplu în minimizarea sumei pătratelor ecuațiilor sistemului. Utilizând aceiaşi parametri şi acelaşi punct inițial, algoritmul UOBYQA furnizează soluția:

$$x^*=[0.9,\ 0.45,\ 1,\ 1.9999,\ 7.9999,\ 8,\ 4.9999,\ 0.9999,\ 1.9999]^{\mathrm{T}},$$

în 6 iterații şi 6861 evaluări ale funcției, pentru care $f(x^*)=0$. Algoritmul Newton dă aceeaşi soluție în doar 4 iterații. Prețul plătit este construcția Jacobianului sistemului, care nu este o sarcină simplă !

Aplicația D6 (***Combustia propanului în aer-varianta redusă***)[8] [Meintjes şi Morgan, 1990], [Andrei, 2001, pg.271], [Andrei, 2003, pg.67], [Averick, Carter şi Moré, 1991].

Aplicația S2 din [Andrei, 2009], capitolul 6, descrie modelul matematic al procesului de combustie a propanului în aer ca un sistem de ecuații algebrice neliniare. Modelul matematic descrie echilibrul chimic al combustiei propanului în aer. Variabilele reprezintă numărul de moli al unui produs format din arderea fiecărui mol de propan. Se cunosc două variante ale acestei probleme. Prima conține o reacție în care participă 10 substanțe. Sistemul respectiv este greu de rezolvat datorită prezenței radicalilor în expresia algebrică a acestuia şi deci a posibilității generării iterațiilor cu componente negative. A doua variantă este o reformulare algebrică redusă a primei variante, care evită radicalii şi care conține numai 5 variabile. În cele ce urmează vom rezolva varianta redusă exprimată ca:

[8] Programul PROPAN.FOR implementează această aplicație.

$$x_1x_2 + x_1 - 3x_5 = 0,$$
$$2x_1x_2 + x_1 + 2R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 + R_9x_2x_4 + R_8x_2 - Rx_5 = 0,$$
$$2x_2x_3^2 + R_7x_2x_3 + 2R_5x_3^2 + R_6x_3 - 8x_5 = 0,$$
$$R_9x_2x_4 + 2x_4^2 - 4Rx_5 = 0,$$
$$x_1x_2 + x_1 + R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 + R_9x_2x_4 + R_8x_2 + R_5x_3^2 + R_6x_3 + x_4^2 - 1 = 0,$$

unde:

$R_5 = 0{,}193$ $R_6 = 0.4106217541E-3$ $R_7 = 0.5451766686E-3$
$R_8 = 0.44975E-6$ $R_9 = 0.3407354178E-4$ $R_{10} = 0.9615E-6$
$R = 10$.

Problema se transformă simplu la minimizarea sumei pătratelor ecuațiilor modelului. În această formă, algoritmul UOBYQA furnizează soluția:

$$x^* = [0.009871,\ 10.6377,\ 0.1167,\ 0.8581,\ 0.036828]^T,$$

în 7 iterații şi 5051 evaluări ale funcției, pentru care $f(x^*) = 0.258707E\text{-}04$.

Aplicația D7 (***Soluția staționară a unui reactor chimic***)[9] [Shacham, 1986], [Andrei, 2001, pg. 267], [Andrei, 2003, pg.63].
În aplicația S3 din [Andrei, 2009], capitolul 6, se prezintă modelul matematic al unui reactor chimic. Modelul reprezintă ecuațiile chimice ale unor componente într-un reactor. Parametrii k_i, r_i definesc coeficienții de reacție chimică, iar variabilele reprezintă cantitățile în moli ale diferitelor substanțe prezente în reacție. În regim staționar toate ratele de reacție trebuie să fie nule.

$$\begin{aligned}
&1 - x_1 - k_1x_1x_6 + r_1x_4 = 0,\\
&1 - x_2 - k_2x_2x_6 + r_2x_5 = 0,\\
&-x_3 + 2k_3x_4x_5 = 0,\\
&k_1x_1x_6 - r_1x_4 - k_3x_4x_5 = 0,\\
&1{,}5\left(k_2x_2x_6 - r_2x_5\right) - k_3x_4x_5 = 0,\\
&1 - x_4 - x_5 - x_6 = 0,
\end{aligned}$$

unde: $k_1 = 31{,}24$ $k_2 = 0{,}272$ $k_3 = 303{,}03$ $r_1 = 2{,}062$ $r_2 = 0{,}02$.
Transformând problema în aceea de minimizare a sumei pătratelor ecuațiilor, în aceleaşi condiții inițiale, algoritmul UOBYQA furnizează soluția:

$$x^* = [1.0304,\ 1.0202,\ -0.0608,\ -0.0001,\ 1.0010,\ -0.00095]^T,$$

în 6 iterații şi 6131 evaluări ale funcției, cu $f(x^*) = 0.2353\text{E-}09$.

[9] Programul REACTOR.FOR implementează această aplicație.

Aplicația D8 (***Cinematica unui robot***)[10] [Kearfott şi Novoa, 1990], [Andrei, 2003, pg.412], [Floudas, *et al.*, 1999, pg.329].
În aplicația S4 din [Andrei, 2009], capitolul 6, am rezolvat problema cinematicii unui robot formulată ca un sistem de ecuații algebrice neliniare.

$$
\begin{aligned}
&4{,}731(10^{-3})x_1x_3 - 0{,}3578x_2x_3 - \\
&0{,}1238x_1 + x_7 - 1{,}637(10^{-3})x_2 - 0{,}9338x_4 - 0{,}3571 = 0\,, \\
&0{,}2238x_1x_3 + 0{,}7623x_2x_3 + \\
&0{,}2638x_1 - x_7 - 0{,}07745x_2 - 0{,}6734x_4 - 0{,}6022 = 0\,, \\
&x_6x_8 + 0{,}3578x_1 + 4{,}731(10^{-3})x_2 = 0\,, \\
&-0{,}7623x_1 + 0{,}2238x_2 + 0{,}3461 = 0\,, \\
&x_1^2 + x_2^2 - 1 = 0\,, \\
&x_3^2 + x_4^2 - 1 = 0\,, \\
&x_5^2 + x_6^2 - 1 = 0\,, \\
&x_7^2 + x_8^2 - 1 = 0\,.
\end{aligned}
$$

Utilizând algoritmul UOBYQA al lui Powell, în care problema a fost reformulată ca una de minimizare a sumei pătratelor ecuațiilor se obține soluția:

$x^* = [0.6715, 0.7409, 0.9518, -0.3064, -0.9638, -0.2665, 0.4046, 0.9144]^T$,

în 6 iterații şi 1719 evaluări ale funcției, pentru care $f(x^*) = 0.1501\text{E-}12$.

Aplicația D9. (***Estimația parametrilor din date experimentale***)[11] [Himmelblau, 1972, pg.430].
Modelul matematic al acestei aplicații este:

$$f(x) = \sum_{i=1}^{7}\left(\frac{x_i^2 + a_i x_2^2 + a_i^2 x_3^2}{(1 + a_i x_4^2)b_i} - 1\right)^2,$$

unde măsurătorile au următoarele valori:

i	a_i	b_i
1	0.0	7.391
2	0.000428	11.18
3	0.0010	16.44
4	0.00161	16.20
5	0.00209	22.20
6	0.00348	24.02
7	0.00525	31.32

[10] Programul ROBOT.FOR implementează această aplicație.
[11] Programul ESTIMATIE.FOR implementează această aplicație.

Pentru punctul inițial $x_0 = [2.7, 90, 1500, 10]^T$, algoritmul UOBYQA furnizează următoarea soluție: $x^* = [2.71436,\ 140.43571,\ 1707.51293,\ 31.51283]^T$, cu $f(x^*) =$ 0.0318571 în 7 iterații şi 1572 evaluări ale funcției.

7. CONCLUZII

Metodele de căutare directă pentru optimizare fără restricții sunt criticabile din multe puncte de vedere. *În primul rând, posibilitățile de determinare a minimului sunt limitate la lungimea pasului de probă care caracterizează fiecare metodă. Ca atare acestea au un caracter local, minimul obținut se află în vecinătatea punctului inițial. În al doilea rând, rata de convergență este extrem de lentă, mai ales pentru funcții cu văi adânci. De obicei, aceste metode nu iau în considerație topologia locală a funcției de minimizat, ceea ce face ca pentru multe iterații valoarea funcției de minimizat să nu se îmbunătățească. În al treilea rând, aceste metode sunt incapabile să furnizeze o soluție cu o acurateț mai mare decât lungimea pasului de probă. O excepție notabilă este dată de metoda de interpolare pătratică a lui Powell. În final, menționăm faptul că aceste metode sunt limitate numai la minimizarea funcțiilor cu un număr foarte redus de variabile.*

Principalul avantaj al acestor metode este uşurința remarcabilă cu care acestea se pot utiliza, mai ales pentru acele funcții care sunt costisitor de evaluat, sau care sunt afectate de zgomot. Într-adevăr, tot ceea ce trebuie să implementăm este expresia algebrică a acestor funcții. Uşurința în utilizare face ca aceste metode de căutare să fie mult studiate şi să constituie o activitate de cercetare foarte activă. Un exemplu relevant este dat de Frimannslund şi Steihaig [2007] care propun o combinație a metodei de rotire a coordonatelor a lui Rosenbrock cu utilizarea unui model pătratic al funcției de minimizat. Un alt exemplu este cel oferit de Price şi Toint [2006] care utilizează structura problemei în căutarea formei funcției de minimizat.

Referințe

Andrei, N., (2001) *Pachete de Programe, Modele şi Probleme de Test pentru Programarea Matematică*. Editura MatrixRom, Bucureşti, 2001.

Andrei, N., (2002) *Sisteme şi Pachete de Programe pentru Programarea Matematică.* Editura Tehnică, Bucureşti, 2002.

Andrei, N., (2003) *Modele, Probleme de Test şi Aplicații de Programare Matematică.* Editura Tehnică, Bucureşti, 2003.

Andrei, N., (2009) *Critica rațiunii algoritmilor de optimizare fără restricții.* Editura Academiei Române, Bucureşti, 2009.

Averick, B.M., Carter, R.G., Moré, J.J., (1991) *The MINPACK-2 test problem collection (Preliminary version),* Mathematics and Computer Science Division, Argonne National Laboratory, Thechnical Memorandum No. 150, May 1991.

Brown, A.A., Bartholomew-Biggs, M.C., (1987a) *Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations*. Technical Report No. 178, The Hatfield Polytechnic, Numerical Optimisation Centre, April 1987.

Brown, A.A., Bartholomew-Biggs, M.C., (1987b) *ODE vs SQP methods for constrained optimization.* Technical Report No. 179, The Hatfield Polytechnic, Numerical Optimisation Centre, June 1987.

Dumitru, V., (1975) *Programare Neliniară. Algoritmi, programe, rezultate numerice.* [Apendix P în colaborare cu Fl. Luban, S. Moga, R. Şerban]. Editura Academiei Române, Bucureşti, 1975.

Ebers, J.J., Moll, J.L., (1954) *Large scale behaviour of junction transistors*. IEE Proc. Vol.42, pp.1761-1772.

Floudas, A.C., *et al.* (1999) *Handbook of test problems in local and global optimization.*Kluwer Academic Publishers, Dordrecht, Boston, London, 1999.

Frimannslund, L., Steihaug, T., (2007) *A generating set search method using curvature information.* Computational Optimization and Applications, 38 (2007), pp.105-121.

Himmelblau, D.M., (1972) *Applied Nonlinear Programming.* McGraw-Hill, New York, 1972.

Hooke, R., Jeeves, T.A., (1961) *Direct search solution of numerical and statistical problems.* Journal of the Association for Computing Machinery, 8 (1961), pp.212-229.

Kearfott, R., Novoa, M., (1990) *INTBIS, a portable interval newton bisection package.* ACM Trans. Math. Software 16, 1990, pp.152-157.

Kelley, C.T., (1999) *Iterative Methods for Optimization.* SIAM Publications, Frontier in Mathematics, Philadelphia, 1999.

Kolda, T.G., Lewis, R.M., Torzcon, V., (2003) *Optimization by direct search: new perspectives of some classical and modern methods.* SIAM Review, 45 (2003), pp.385-482.

Maranas, C.D., Floudas, C.A., (1994) *A global optimization method for Weber's problem with attraction and repulsion.* În W.W. Hager, D.W. Hearn şi P.M. Pardalos (Eds.) *Large-Scale Optimization. State of the Art*, Kluwer Academic Publishers, Dordrecht, 1994, pp.259-293.

Meintjes, K., Morgan, A.P., (1990) *Chemical-equilibrium systems as numerical test problems*. ACM Trans. Math. Soft., vol.16, (1990), pp.143-151.

Moré, J.J., Sorensen, D.C., (1983) *Computing a trust region step,* SIAM, J. Sci. Stat. Comput., 4 (1983), pp.553-572.

Nelder, J.A., Mead, R., (1965) *A simplex method for function minimization.* The Computer Journal, 7 (1965), pp.308-313.

Powell, M.J.D., (1964) *An efficient method for finding the minimum of a function of several variables without calculating derivatives,* Computer Journal, 17 (1964), pp.155-162.

Powell, M.J.D., (1994) *A direct search optimization method that models the objective and constraint functions by linear interpolation,* în Advances in Optimization and Numerical Analysis, S. Gomez, J.P. Hennart (Eds.), Kluwer Academic Publishers, 1994, pp. 51-67.

Powell, M.J.D., (1998) *Direct search algorithms for optimization calculations.* Report DAMTP 1998/NA04, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, March 1998. [Acta Numerica, vol.7, 1998]

Powell, M.J.D., (2000) *On the Lagrange functions of quadratic models that are defined by interpolation.* Report DAMTP 2000/NA10, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2000.

Powell, M.J.D., (2001) *UOBYQA: unconstrained optimization by quadratic approximation.* Report No. DAMTP 2000/NA14, University of Cambridge, December 2000, revised June 2001.

Powell, M.J.D., (2002) *UOBYQA: unconstrained optimization by quadratic approximation,* Mathematical Programming, 92 (2002), pp. 555-582.

Price, C.P., Toint, P., (2006) *Exploiting problem structure in pattern search methods for unconstrained optimization.* Optimization Methods and Software, 31 (2006), pp. 479-491.

Ratschek, H., Rokne, J., (1993) *A circuit design problem.* J. Global Optimization, 3, 501, 1993.

Rosenbrock, H.H., (1960) *An automatic method for finding the greatest or least value of a function.* The Computer Journal, 3 (1960), pp.175-184.

Shacham, M., (1986) *Numerical solution of constrained nonlinear algebraic equations.* Int. Journal for Numerical Methods in Engineering, vol.23, 1455-1481, 1986.

Spendley, W., Hext, G.R., Himsworth, F.R., (1962) *Sequential applications of simplex designs in optimization and evolutionary operation.* Technometrics, 4 (1962), pp.441-461.

Torczon, V., (1991) *On the convergence of the multi-directional search algorithm.* SIAM Journal on Optimization, 1 (1991), pp.123-145.

Torczon, V., (1997) *On the convergence of patern search algorithm.* SIAM Journal on Optimization, 7 (1997), pp. 1-25.

Winfield, D., (1969) *Function and functional optimization by interpolation in data tables.* Ph.D. Thesis, Harward University, Cambridge, 1969.

Winfield, D., (1973) *Function minimization by interpolation in a data table.* Journal of the Institute of Mathematics and its Applications, 12 (1973), pp.339-347.

Wright, M.H., (1996) *Direct search methods: once scorned, now respectable.* În D.F. Griffiths, G.A. Watson (Eds.) Numerical Analysis 1995. Proceedings of the 1995 Dundee Conference in Numerical Analysis, Addison Wesley, Longmann, Harlow, 1996, pp.191-208.

May 25, 2009