METODA MINOS

În acest raport tehnic prezentăm una dintre cele mai elaborate metode de rezolvare a problemei generale de programare matematică de mari dimensiuni datorată lui Murtagh şi Saunders [1978, 1980, 1982, 1995]. Metoda MINOS este o metodă de optimizare secvenţială liniară, fiind propusă pentru prima dată de Robinson [1972] şi Rosen şi Kreuser [1972]. Ideea este de a minimiza funcţia Lagrange modificată referitor la o aproximaţie liniară a restricţiilor problemei.

Metoda reduce rezolvarea problemei originale la rezolvarea unui şir de subprobleme (iteraţii majore) obţinute prin liniarizarea fiecărei restricţii neliniare încălcate în jurul punctului curent şi introducerea restului acestor liniarizări în funcţia obiectiv prin intermediul unui factor de penalizare. Ca atare, funcţia care trebuie minimizată la fiecare iteraţie (minoră) este Lagrangianul corespunzător restricţiilor încălcate. Algoritmul corespunzător acestei metode utilizează ideea metodei gradientului redus al lui Wolfe [1967], precum şi tehnica de reducere a variabilelor a lui McCormick [1970a,b], combinate cu diverse ingrediente foarte fine privind esenţa algoritmului simplex referitor la reprezentarea inversei bazei şi calculului gradientului redus al funcţiei de minimizat.


Michael Saunders

Metoda se poate considera ca o extensie a metodei simplex combinată cu tehnici de gradient redus pentru rezolvarea problemelor neliniare.

Scopul acestei lucrări este de a prezenta metoda MINOS cu suficiente detalii pentru a ilustra tehnica de rezolvare care utilizează ideea programării liniare succesive într-o manieră foarte eficientă. Totodată vom prezenta experienţa numerică în ceea ce priveşte rezolvarea problemelor neliniare complexe, precum şi a unor aplicaţii de programare matematică neliniare.

Metoda MINOS a fost una dintre *primele metode profesionale* de rezolvare a problemelor generale de programare matematică neliniare. Succesul acestei metode se datorează în principal utilizării ideilor din metoda simplex combinate cu tehnici de gradient redus utilizate în optimizarea neliniară.

Pentru început vom considera cazul problemelor cu restricţii liniare, urmând ca apoi să tratăm cazul problemei generale cu restricţii neliniare.

1. PROBLEME CU RESTRICȚII LINIARE

Fie problema:

$$\min F(x) = f(x_N) + c^{\mathrm{T}} x, \tag{1}$$

referitor la:

$$Ax = b,$$
$$l \le x \le u,$$

unde A este o matrice $m \times n$-dimensională $(m \le n)$ şi vectorul $x \in R^n$ este partiționat sub forma $x = [x_N \; x_L]^{\mathrm{T}}$, în care x_N reprezintă subvectorul variabilelor neliniare (adică acele variabile care apar în porțiunea neliniară a expresiei algebrice a funcției de optimizat), iar x_L subvectorul variabilelor liniare. Presupunem că $l \le u$.

Observăm că matricea A şi vectorul c operează asupra tuturor variabilelor. În anumite situații termenul $c^{\mathrm{T}} x$ care implică variabilele neliniare x_N poate fi incorporat în $f(x_N)$. În alte cazuri c poate fi nul. Presupunem că funcția $f(x_N)$ este cel puțin de ordinul unu continuu diferențiabilă pe domeniul de admisibilitate al problemei. Chiar dacă partiționarea lui x şi a lui $F(x)$ în termeni liniari şi neliniari este o foarte mare utilitate practică, totuşi pentru simplificarea prezentării vom nota $F(x)$ şi gradientul acesteia $\nabla F(x)$ cu $f(x)$, respectiv $g(x)$. Este clar că dacă $f(x_N) = 0$, atunci (1) este o problemă de programare liniară.

Variabile superbazice

După cum am văzut, în metoda simplex (vezi [Andrei, 2010], capitolul 3) conceptul de soluție admisibilă de bază este fundamental. Virtutea acestui concept constă în modul în care tratează restricțiile de mărginire a variabilelor $l \le x \le u$. Într-adevăr, atât din punct de vedere teoretic cât şi practic nu este recomandabil ca restricțiile de mărginire a variabilelor să fie incluse în matricea A. Mult mai bine este ca acestea să fie utilizate pentru eliminarea variabilelor. Metoda simplex cu variabile mărginite (vezi [Andrei, 2010], capitolul 4) realizează aceasta într-un mod foarte eficient, operând numai asupra bazei şi nu asupra întregii matrice a problemei.

În cazul problemelor neliniare nu ne putem aştepta ca o soluție optimă să fie soluție admisibilă de bază. Totuşi, dacă numărul variabilelor neliniare este mic, atunci pare rezonabil să presupunem că o soluție optimă este „aproape” o soluție admisibilă de bază. Astfel, ca o generalizare, se introduce conceptul de *variabile superbazice* [Murtagh şi Saundrers, 1978, 1980, 1982]. Partiționând vectorul x sub forma $x = [x^B \; x^S \; x^N]^{\mathrm{T}}$, atunci restricțiile liniare din (1) devin:

$$Ax = \begin{bmatrix} B & S & N \end{bmatrix} \begin{bmatrix} x^B \\ x^S \\ x^N \end{bmatrix} = b. \tag{2}$$

Matricea B este pătrată şi nesingulară, ca în metoda simplex. Matricea S este $m \times s$-dimensională, cu $0 \le s \le n-m$, iar N conţine restul coloanelor din A. Variabilele x^B, x^S şi x^N sunt numite: variabile de bază (bazice), superbazice şi respectiv nebazice. Atât variabilele bazice cât şi cel superbazice sunt libere de a lua valori între marginile lor inferioare, respectiv superioare. Variabilele superbazice sunt „variabilele de forţare" în sensul că ele pot fi deplasate în orice direcţie cu scopul de a îmbunătăţi valoarea funcţiei obiectiv. Ca în metoda simplex, variabilele nebazice sunt fixate la una dintre marginile lor, inferioară sau superioară. Variabilele bazice sunt obligate să ia valori într-un mod bine definit pentru a menţine admisibilitatea în raport cu restricţiile $Ax = b$.

Faptul că soluţiile oprime ale problemei (1) deseori vor fi „aproape" soluţii admisibile de bază este confirmat de

Teorema 1. *Presupunem că o problemă de programare matematică neliniară are t variabile neliniare (care pot apărea fie în funcţia obiectiv, fie în restricţii). Atunci există o soluţie optimă pentru care numărul variabilelor superbazice s verifică $s \le t$.*

Demonstraţie. Să considerăm variabilele neliniare fixate la valorile lor optime. Atunci, problema rămasă este una liniară pentru care există o soluţie optimă cu $s = 0$. Concluzia teoremei rezultă imediat dacă considerăm acum variabilele neliniare ca superbazice. Evident, pentru început $s = t$, dar dacă o variabilă neliniară este fixată la una dintre marginile ei, atunci o putem eticheta ca nebazică, şi deci $s < t$. ■

Prezentarea metodei

Presupunem că $f(x)$ se poate dezvolta în serie Taylor sub forma:

$$f(x+\Delta x) = f(x) + g(x)^{\mathrm{T}}\Delta x + \frac{1}{2}\Delta x^{\mathrm{T}} H(x+\gamma\Delta x)\Delta x, \tag{3}$$

unde $0 \le \gamma \le 1$ şi $H(x+\gamma\Delta x)$ este hesseianul lui f evaluat într-un punct de pe segmentul de dreaptă care uneşte x şi $x+\Delta x$. Observăm imediat că dacă f este o funcţie pătratică, atunci H este o matrice constantă.

Dacă dispunem de un punct curent admisibil x, atunci problema este de a construi o deplasare Δx astfel încât următoarele două proprietăţi să fie îndeplinite.

Proprietatea 1 (Admisibilitate)

$$\begin{bmatrix} B & S & N \\ & & I \end{bmatrix} \begin{bmatrix} \Delta x^B \\ \Delta x^S \\ \Delta x^N \end{bmatrix} = 0. \tag{4}$$

Această proprietate ne asigură că noul punct $x+\Delta x$ este admisibil, unde Δx^B, Δx^S şi Δx^N sunt componentele lui Δx corespunzătoare variabilelor bazice, superbazice şi respectiv nebazice.

Proprietatea 2 (Optimalitate)

$$\begin{bmatrix} g_B \\ g_S \\ g_N \end{bmatrix} + H \begin{bmatrix} \Delta x^B \\ \Delta x^S \\ \Delta x^N \end{bmatrix} = \begin{bmatrix} B^{\mathrm{T}} & \\ S^{\mathrm{T}} & \\ N^{\mathrm{T}} & I \end{bmatrix} \begin{bmatrix} \mu \\ \lambda \end{bmatrix}, \tag{5}$$

unde g_B, g_S şi g_N sunt componentele gradientului g corespunzător variabilelor bazice, superbazice şi respectiv nebazice, iar μ şi λ sunt multiplicatorii Lagrange.

Dacă $x + \Delta x$ este suficient de aproape de x, atunci se poate utiliza un „model pătratic" al funcției f în jurul lui x, astfel încât membrul stâng din (5) reprezintă gradientul funcției obiectiv în $x + \Delta x$. Pentru ca $x + \Delta x$ să fie un punct de optim local în raport cu mulțimea curentă a restricțiilor active în punctul curent vom impune ca gradientul funcției obiectiv să fie ortogonal la suprafața formată din mulțimea restricțiilor active. Pentru aceasta, gradientul trebuie să fie o combinație liniară a normelor restricțiilor active, dată de membrul drept din (5). Mai mult, pentru optimalitatea lui $x + \Delta x$ trebuie ca gradientul cu semn schimbat să fie direcționat *ortogonal în afara* domeniului de admisibilitate definit de restricțiile problemei. Pentru (1) vom impune deci ca $\lambda_j \le 0$ dacă $x_j^N = u_j$, sau $\lambda_j \ge 0$ dacă $x_j^N = l_j$, $j = 1, \ldots, n - m - s$. Acum, din (4) obținem:

$$\begin{aligned} & B\Delta x^B + S\Delta x^S = 0, \\ & \Delta x^N = 0. \end{aligned} \tag{6}$$

Deci

$$\Delta x^B = -W\Delta x^S, \tag{7}$$

unde

$$W = B^{-1}S. \tag{8}$$

Ca atare

$$\Delta x = \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta x^S, \tag{9}$$

ceea ce înseamnă că putem lucra numai cu vectorul Δx^S care este de dimensiune s, mult mai mică decât n.

Matricea W nu se calculează explicit, deoarece B^{-1} este reprezentată sub forma produs a inversei (FPI) sau forma de eliminare a inversei (FEI) (vezi [Andrei, 2010], capitolele 7 şi 8). Referindu-ne la (5), aceasta se poate simplifica dacă o înmulțim la stânga cu matricea

$$\begin{bmatrix} I & & \\ -W^{\mathrm{T}} & \mathrm{I} & \\ & & I \end{bmatrix}. \tag{10}$$

Considerând prima linie partiționată obținem o expresie a estimațiilor multiplicatorilor Lagrange asociați restricțiilor problemei:

$$g_B + \begin{bmatrix} I & 0 & 0 \end{bmatrix} H \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta x^S = B^{\mathrm{T}} \mu. \tag{11}$$

Observăm că dacă $\Delta x^S = 0,$ ceea ce înseamnă că nu se poate face nici un progres către optim în mulțimea curentă a restricțiilor active, atunci din (11) obținem

$$B^{\mathrm{T}} \mu = g_B. \tag{12}$$

Fie u soluția sistemului (12), care de fapt este analogul multiplicatorilor simplex din metoda simplex.

Considerând acum a treia linie partiționată din (5) înmulțită cu (10), atunci obținem

$$g_N + \begin{bmatrix} 0 & 0 & I \end{bmatrix} H \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta x^S = N^{\mathrm{T}} \mu + \lambda. \tag{13}$$

Din nou, dacă $\Delta x^S = 0,$ atunci (13) devine

$$\lambda = g_N - N^{\mathrm{T}} \mu, \tag{14}$$

care este analogul factorilor de cost redus din metoda simplex.

În final, considerând a doua linie din (5) partiționată şi înmulțită cu (10) obținem

$$\begin{bmatrix} -W^{\mathrm{T}} & I & 0 \end{bmatrix} H \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta x^S = -h, \tag{15}$$

unde

$$h = \begin{bmatrix} -W^{\mathrm{T}} & I & 0 \end{bmatrix} g = g_S - S^{\mathrm{T}} u. \tag{16}$$

Astfel, variabilele independente Δx^S se pot determina dintr-o ecuație de tip Newton, unde h se poate privi ca un tip de „gradient redus” şi matricea

$$\begin{bmatrix} -W^{\mathrm{T}} & I & 0 \end{bmatrix} H \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \tag{17}$$

ca un „hesseian redus”.

Observăm că $\|h\| = 0$ este o condiție necesară de staționaritate a punctului pe mulțimea curentă a restricțiilor active, care dacă hesseianul redus (17) este nesingular, implică imediat $\Delta x^S = 0.$ În practică se va calcula a factorizare a hesseianului redus (17) de forma $R^{\mathrm{T}} R,$ unde R este o matrice superior triunghiulară, astfel încât pasul Δx^S dat de sistemul

$$(R^{\mathrm{T}} R) \Delta x^S = -h \tag{18}$$

poate fi foarte simplu calculat prin substituţie înainte şi înapoi, la fel ca în metoda simplex cu factorizarea LU a bazei.

Gill şi Murray [1974a] au considerat o clasă de algoritmi în care direcţia de căutare de-a lungul suprafeţei restricţiilor active este caracterizată ca fiind în domeniul unei matrice Z care este ortogonală la matricea formată cu normalele restricţiilor active. Astfel, dacă $\hat{A}x = \hat{b}$ este mulţimea curentă a $n-s$ restricţii active, atunci Z este o $n \times s$-matrice astfel încât

$$\hat{A}Z = 0. \tag{19}$$

Această caracterizare este foarte importantă şi se poate utiliza pentru a descrie o serie de algoritmi în ideea programării pătratice succesive.

Utilizând această idee, principalii paşi necesari de a fi executaţi la fiecare iteraţie pentru a determina o aproximaţie a optimului sunt:

A) Calculul gradientului redus $\bar{g} = Z^{\mathrm{T}} g$.

B) Calculul unei aproximaţii a hesseianului redus $\bar{H} = Z^{\mathrm{T}} HZ$.

C) Calculul unei soluţii a sistemului de ecuaţii $\bar{H}\bar{p} = -\bar{g}$.

D) Calculul direcţiei de căutare $p = Z\bar{p}$.

E) Determinarea lungimii pasului de deplasare α^* de-a lungul direcţiei p ca soluţie a problemei

$$f(x + \alpha^* p) = \min_{\alpha} \left\{ f(x + \alpha p) : x + \alpha p \text{ admisibil} \right\}.$$

Acest algoritm este foarte general, asupra matricei Z se impun doar două condiţii: să fie de rang plin pe coloane şi să verifice (19). Ca atare Z poate avea orice formă care să verifice condiţiile de mai sus. În particular, pentru metoda MINOS, Z are structura

$$Z = \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix}. \tag{20}$$

Aceasta este o reprezentare foarte convenabilă care va fi utilizată pentru expunerea atât a algoritmului de rezolvare a problemei cu restricţii liniare cât şi pentru problema cu restricţii neliniare din cadrul metodei MINOS. Din punct de vedere computaţional (practic) matricea Z nu se calculează niciodată. Algoritmul lucrează numai cu S şi cu o factorizare LU a matrice de bază B.

Prezentarea algoritmului metodei MINOS pentru restricţii liniare

Să presupunem că dispunem de următoarele elemente:

1) Un punct admisibil x care verifică relaţiile

$$\begin{bmatrix} B & S & N \end{bmatrix} x = b, \text{ şi } l \le x \le u.$$

2) Valoarea funcţiei obiectiv în punctul x, $f(x)$, şi a gradientului acesteia

$$g(x) = \begin{bmatrix} g_B & g_S & g_N \end{bmatrix}^{\mathrm{T}}.$$

3) Numărul de variabile superbazice s, unde $0 \le s \le n - m$.

4) O factorizare LU a $m \times m$-matricei de bază B.

5) O factorizare $R^{\mathrm{T}}R$ a unei aproximații quasi-Newton a $s \times s$-matricei $Z^{\mathrm{T}}HZ$.

6) Un vector u, soluție a sistemului $B^T u = g_B$.

7) Vectorul gradientului redus $h = g_S - S^{\mathrm{T}}u$.

8) Toleranțele de convergență, pozitive, suficient de mici ε_x, ε_f, ε_λ și ε_g.

Cu acestea, algoritmul metodei MINOS pentru rezolvarea problemei cu restricții liniare (1) este următorul.

Algoritmul 1. (Algoritmul MINOS – restricții liniare)

Pasul 1.	Se inițializează algoritmul cu datele de mai sus şi se pune $k = 1$, $x_k = x$, $f(x_k) = f_k$ şi $g(x_k) = g_k$.
Pasul 2.	Se execută testul de convergență în subspațiul variabilelor superbazice curente. Dacă $\|h\| > \varepsilon_h$, atunci se execută pasul 4; altfel se continuă cu pasul 3.
Pasul 3.	Calculul unei estimații a multiplicatorilor Lagrange, modificarea numărului variabilelor superbazice: a) Se calculează $\lambda = g_N - N^{\mathrm{T}}u$. b) Deoarece pentru verificarea condițiilor Karush-Kuhn-Tucker trebuie ca pentru toate variabilele aflate la marginea lor superioară (inferioară) multiplicatorii Lagrange asociați acestor restricții să fie negativi (pozitivi), se determină: $\lambda_1 = \max\{\lambda_j : \lambda_j > 0\}$ pentru x_j^N fixat la marginea superioară u_j, $\lambda_2 = \min\{\lambda_j : \lambda_j < 0\}$ pentru x_j^N fixat la marginea inferioară l_j. Dacă $\lambda_1 \le \varepsilon_\lambda$ şi $\lambda_2 \le \varepsilon_\lambda$, stop; soluția curentă este optimă. c) Altfel se procedează după cum urmează: 1) Se alege variabila din afara bazei x_i^N corespunzătoare la $\|\lambda_i\| = \max\{\lambda_1, \|\lambda_2\|\}$ ca o nouă variabilă superbazică. 2) Se augmentează matricea S cu coloana corespunzătoare variabilei x_i^N. 3) Se augmentează gradientul redus h cu λ_i. 4) Se modifică corespunzător matricea R. d) Se pune $s = s + 1$.
Pasul 4.	Calculul direcției de căutare p. a) Se rezolvă sistemul $(R^{\mathrm{T}}R)p_S = -h$. b) Se rezolvă sistemul $(LU)p_B = -Sp_S$. c) Se pune $p = [p_B \quad p_S \quad 0]^{\mathrm{T}}$.
Pasul 5.	Determinarea valorii maxime a lungimii pasului de deplasare: a) Se determină $\alpha_{\max} \ge 0$ ca fiind valoarea maximă pozitivă a lui α

	astfel încât una dintre componentele vectorului $\left[x^B + \alpha p_B \quad x^S + \alpha p_S\right]^{\mathrm{T}}$ îşi atinge una dintre marginile sale. b) Dacă $\alpha_{\max} = 0$, atunci se execută pasul 10, altfel se continuă cu pasul 6.
Pasul 6.	Se execută o căutare liniară de-a lungul direcției p pentru a determina α^* astfel încât $f(x^k + \alpha^* p) = \min\{f(x_k + \alpha p) : 0 < \alpha \le \alpha_{\max}\}$.
Pasul 7.	Se calculează noul punct $x_{k+1} = x_k + \alpha^* p$ şi se pune $k = k+1$.
Pasul 8.	Se evaluează $f(x_k) = f_k$ şi $g(x_k) = \nabla f(x_k) = g_k$.
Pasul 9.	Calculul noului gradient redus: a) Se rezolvă sistemul $(U^T L^T)u = g_B$. b) Se calculează $\overline{h} = g_S - S^{\mathrm{T}} u$. c) Utilizând α, p şi schimbarea în gradientul redus $\overline{h} - h$ se modifică matricea R corespunzător unei actualizări quasi-Newton a lui $R^{\mathrm{T}} R$. d) Se pune $h = \overline{h}$. e) Dacă $\alpha = \alpha_{\max}$, adică dacă una dintre componentele lui x^B sau x^S îşi atinge una dintre margini, atunci se continuă cu pasul 10; altfel $(\alpha < \alpha_{\max})$ se execută pasul 2.
Pasul 10.	La acest pas al algoritmului o variabilă de bază sau superbazică, cu indicele i să zicem, îşi atinge una dintre marginile asociate ei. a) Dacă variabila i este bazică, atunci: 1) Se schimbă a i-a variabilă de bază x_i^B (adică coloana corespunzătoare lui x_i^B din B) cu a j-a variabilă superbazică x_j^S (adică cu coloana corespunzătoare lui x_j^S din S). Indicele j se alege astfel încât noua matrice de bază să fie nesingulară. 2) Se actualizează corespunzător factorii L şi U, matricea R şi vectorul u. 3) Se calculează noul gradient redus $h = g_S - S^{\mathrm{T}} u$, şi se contiună cu pasul 10.c). b) Dacă variabila a i-a este superbazică, atunci definim $j = i - m$. c) Se transformă a j-a variabilă din S în variabilă nebazică fixată la una dintre marginile ei: 1) Se elimină coloana j din S şi se introduce în matricea N. 2) Se elimină coloana j din R.

	3) Se elimină a j-a componentă din vectorii x^S şi h. 4) Se triunghiularizează superior R. d) Se pune $s = s - 1$ şi se continuă cu pasul 2. ♦

Câteva comentarii sunt necesare.

1) O iterație a algoritmului 8.1 (MINOS pentru restricții liniare) este echivalentă cu o iterație a algoritmului simplex din programarea liniară aplicat unei probleme $m \times n$-dimensionale, plus o iterație a unui algoritm de tip quasi-Newton aplicat unei probleme de minimizare fără restricții de dimensiune s.

2) Referitor la testul de convergență din pasul 2 al algoritmului, de obicei se utilizează „un grup de test". O variantă de asemenea teste ar fi, de exemplu, următoarea. Fie Δx^S schimbarea variabilelor superbazice, Δf schimbarea funcției obiectiv şi ε acuratețea de calcul a soluției. Atunci în locul testului simplu din pasul 2 al algoritmului se poate utiliza grupul de teste:
„Dacă

$$\begin{aligned} &\left\|\Delta x^S\right\| \le (\varepsilon_x + \sqrt{\varepsilon})(1 + \left\|x^S\right\|), \\ &\left|\Delta f\right| \le (\varepsilon_f + \varepsilon)(1 + \left|f\right|), \\ &\left\|h\right\| \le \varepsilon_h \quad \text{sau} \quad \left\|h\right\| \le \varepsilon_g \left\|u\right\|, \end{aligned} \tag{21}$$

atunci se execută pasul 3."

3) La fel ca în metoda simplex, utilizarea unei factorizări eficiente a bazei B joacă un rol crucial pentru rezolvarea sistemelor algebrice liniare $By = b$ sau $B^{\mathrm{T}} z = c$. Algoritmul implementează o factorizare LU a lui B deosebit de sofisticată care ține seama atât de structura factorilor matriceali (pentru minimizarea umplerii) cât şi de stabilitatea numerică a calculelor.

4) Un rol important în cadrul algoritmului 1 îl are metoda utilizată pentru modificarea matricei R din aproximarea $R^{\mathrm{T}} R = Z^{\mathrm{T}} H Z$, ori de câte ori x şi/sau Z sunt modificați. Fie $\overline{R}$ matricea care reprezintă o modificare a lui R. Pentru a asigura stabilitatea numerică toate modificările lui R sunt implementate utilizând transformări ortogonale.

5) În principiu, în pasul 9.c) se poate utiliza orice formulă quasi-Newton de actualizare a lui $R^{\mathrm{T}} R$. Murtagh şi Saunders propun şi utilizează actualizarea BFGS:

$$\overline{R}^{\mathrm{T}} \overline{R} = R^{\mathrm{T}} R + \frac{y y^{\mathrm{T}}}{y^{\mathrm{T}} p_S} - \frac{h h^{\mathrm{T}}}{h^{\mathrm{T}} p_S}, \tag{22}$$

unde $y = \overline{h} - h$.

6) Presupunem că în pasul 10.a) a i-a variabilă bazică este schimbată cu a j-a variabilă superbazică. Odată ce R a fost actualizat în pasul 9.c) este necesară o actualizare a lui Z. Relația dintre matricele de bază a noului spațiu nul şi a celui vechi este

$$\bar{Z} = Z(I + e_j v^{\mathrm{T}}), \tag{23}$$

unde e_j este a j-a coloană a matricei unitate, iar v este definit prin relațiile:

$$\begin{aligned} B^{\mathrm{T}} u &= e_i, \\ y &= S^{\mathrm{T}} u, \\ v &= -\frac{y + e_j}{y^{\mathrm{T}} e_j}. \end{aligned} \tag{24}$$

Modificarea matricei R corespunzătoare lui (23) este următoarea

$$\bar{R}^{\mathrm{T}} R = (I + v e_j^{\mathrm{T}}) R^{\mathrm{T}} R (I + e_j v^{\mathrm{T}}). \tag{25}$$

Dacă r_j este a j-a coloană a lui R, atunci (25) se poate scrie sub forma

$$\bar{R}^{\mathrm{T}} R = (R^{\mathrm{T}} + v r_j^{\mathrm{T}})(R + r_j v^{\mathrm{T}}). \tag{26}$$

Această formulă de actualizare a lui R este foarte convenabilă pentru transformarea lui $\bar{R}$ la forma triunghiulară.

7) În pasul 3.c) dacă coloana a_i, corespunzătoare variabilei x_i^N selectată ca o nouă variabilă superbazică, este adăugată la S, atunci noua matrice de bază a spațiului nul corespunzător este chiar $\bar{Z} = [Z \;\; z]$, unde

$$z = \begin{bmatrix} -B^{-1} a_i \\ e_s \\ 0 \end{bmatrix}. \tag{27}$$

Gill şi Murray [1974b, pag. 76-77] recomandă aproximarea vectorului Hz prin diferențe finite

$$v = \frac{g(x + \delta z) - g(x)}{\delta} = Hz + O(\delta \|z\|^2),$$

unde δ este un pas mic luat în direcția z, de exemplu $\delta = \varepsilon^{1/2} \|z\|$. Atunci, pentru a genera o nouă coloană a lui R se poate utiliza următoarea procedură:

1) Se rezolvă sistemul $R^{\mathrm{T}} r = Z^{\mathrm{T}} v$,
2) Se calculează $\sigma = z^{\mathrm{T}} v - \|r\|^2$ şi $\rho = \sqrt{|\sigma|}$,
3) Se consideră $\bar{R} = \begin{bmatrix} R & r \\ & \rho \end{bmatrix}$.

Comparând acum găsim

$$\bar{R}^{\mathrm{T}} R = \begin{bmatrix} R^{\mathrm{T}} & \\ r^{\mathrm{T}} & \rho \end{bmatrix} \begin{bmatrix} R & r \\ & \rho \end{bmatrix} = \begin{bmatrix} R^{\mathrm{T}} R & Z^{\mathrm{T}} v \\ v^{\mathrm{T}} Z & z^{\mathrm{T}} v \end{bmatrix}$$

şi

$$\bar{Z}^{\mathrm{T}} H Z = \begin{bmatrix} Z^{\mathrm{T}} \\ z^{\mathrm{T}} \end{bmatrix} H \begin{bmatrix} Z & z \end{bmatrix} = \begin{bmatrix} Z^{\mathrm{T}} H Z & Z^{\mathrm{T}} H z \\ z^{\mathrm{T}} H Z & z^{\mathrm{T}} H z \end{bmatrix},$$

observăm că dacă $R^{\mathrm{T}}R$ furnizează o bună aproximare pentru $Z^{\mathrm{T}}HZ$, atunci $\overline{R}^{\mathrm{T}}\overline{R}$ are toate şansele să fie o bună aproximare pentru $\overline{Z}^{\mathrm{T}}H\overline{Z}$.

Observăm că dacă termenul neliniar $f(x_N)$ din (1) nu există, adică problema nu are variabile neliniare, atunci algoritmul MINOS se reduce la algoritmul simplex pentru programarea liniară. În [Andrei, 2010, capitolul 8] se prezintă tehnica de inversare a bazei utilizând forma de eliminare a inversei (FEI) utilizată în algoritmul MINOS, precum şi un studiu numeric intensiv privind performanțele acestui algoritm pentru rezolvarea problemelor de programare liniară de mari dimensiuni.

Alte detalii asupra algoritmului MINOS şi a implementării sale sunt prezentate în [Murtagh şi Saunders, 1978, 1982, 1995].

Exemplul 1. Să considerăm aplicația care se referă la problema echilibrului chimic (vezi [Schittkowski, 1987, problema 377, pagina 196]), şi care se prezintă ca o problemă de optimizare cu restricții liniare.

$$min \sum_{j=1}^{10} x_j \left(c_j + \ln \frac{x_j}{x_1 + \cdots + x_{10}} \right)$$

referitor la:

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0,$$
$$x_4 + 2x_5 + x_6 + x_7 - 1 = 0,$$
$$x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0.$$

Introducând şi margini superioare de tipul $0.0001 \le x_i \le 10$, $i = 1,\ldots,10$, atunci algoritmul MINOS furnizează soluția din tabelul 8.1, care conține şi valorile parametrilor a_i, $i = 1,\ldots,10$, precum şi punctul inițial.

Tabelul 1.
Parametrii a_i, $i = 1,\ldots,10$, punctul inițial, marginile variabilelor şi soluția problemei.

a_i	x_0	l	x^*	u
-6.089	0.1	0.0001	10.0000	10
-17.164	0.1	0.0001	9.99955	10
-34.054	0.1	0.0001	0.99950	10
-5.914	0.1	0.0001	10.0000	10
-24.721	0.1	0.0001	9.50000	10
-14.986	0.1	0.0001	10.0000	10
-24.100	0.1	0.0001	0.0001	10
-10.708	0.1	0.0001	0.0001	10
-26.662	0.1	0.0001	0.0001	10
-22.179	0.1	0.0001	0.0001	10

Valoarea funcției obiectiv este $f(x^*) = -794.99038045$. Numărul de iterații necesare pentru a obține această soluție este 15. Funcția obiectiv a fost evaluată de 15 ori. ♦

2. PROBLEME CU RESTRICȚII NELINIARE

Să considerăm acum problema generală de programare matematică neliniară în forma

$$\min f^0(x) + c^{\mathrm{T}} x + d^{\mathrm{T}} y \tag{28}$$

referitor la:

$$\begin{aligned} & f(x) + A_1 y = b_1, \\ & A_2 x + A_3 y = b_2, \\ & l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u, \end{aligned}$$

unde $f(x) = [f^1(x) \cdots f^{m_1}(x)]$, $b_1 \in R^{m_1}$, $b_2 \in R^{m_2}$ care evidențiază atât „*partea liniară*” cât şi „*partea neliniară*” a problemei. Primele n_1 variabile ale problemei, notate cu x, sunt „variabilele neliniare”. Ele apar fie în funcția obiectiv fie în primele m_1 restricții prin intermediul unor funcții neliniare. Variabilele $y \in R^{n_2}$ sunt „*variabilele liniare*”, care eventual includ şi variabilele ecart. Presupunem că funcțiile $f^i(x)$, $i = 0, \ldots, m_1$, sunt cel puțin de două ori continuu diferențiabile şi că există un punct (x^*, y^*) în care condițiile de optimalitate Karush-Kuhn-Tucker sunt satisfăcute.

Procesul de rezolvare a problemei (28) constă într-un şir de *iterații majore*, fiecare implicând liniarizarea restricțiilor neliniare în jurul unui punct x_k, corespunzător aproximației de prim ordin a dezvoltării în serie Taylor

$$f^i(x) = f^i(x_k) + g^i(x_k)^{\mathrm{T}}(x - x_k) + O\left(\|x - x_2\|^2\right).$$

Definim:

$$\tilde{f}(x, x_k) \square f(x_k) + J(x_k)(x - x_k), \tag{29}$$

sau, mai simplu, $\tilde{f} = f_k + J_k(x - x_k)$, unde $J(x)$ este $m_1 \times n_1$-matricea Jacobian a funcției $f(x)$. Observăm că

$$f - \tilde{f} = (f - f_k) - J_k(x - x_k) \tag{30}$$

conține termenii (neliniari) de ordin superior din dezvoltarea în serie Taylor a lui $f(x)$ în jurul lui x_k.

Cu acestea, la iterația majoră k, algoritmul formează următoarea *subproblemă cu restricții liniare*:

$$\min_{x,y} L(x, y, x_k, \lambda_k, \rho) =$$

$$f^0(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y - \lambda_k^{\mathrm{T}}(f - \tilde{f}) + \frac{1}{2}\rho(f - \tilde{f})^{\mathrm{T}}(f - \tilde{f}) \tag{31}$$

referitor la:

$$\tilde{f} + A_1 y = b_1,$$
$$A_2 x + A_3 y = b_2,$$
$$l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u.$$

Funcția obiectiv $L(.)$ a problemei (31) este *Lagrangeanul modificat augmentat* în care în locul diferenței $f - A_1 y - b_1$ s-a utilizat diferența $f - \tilde{f}$.

Este clar că

$$\begin{aligned} \frac{\partial L(x,y)}{\partial x} &= g^0(x) + c - (J - J_k)^{\mathrm{T}}\left[\lambda_k - \rho(f - \tilde{f})\right], \\ \frac{\partial L(x,y)}{\partial y} &= d. \end{aligned} \tag{32}$$

Observăm imediat că neliniaritățile în L implică variabilele x, dar nu şi y, ceea ce înseamnă că subproblema (31) are aceleaşi variabile neliniare ca problema originală.

Lagrangeanul modificat a fost utilizat de Robinson [1972] cu $\rho = 0$. Utilizarea unui termen de penalizare pătratic, care asigură ca hesseianul Lagrangeanului augmentat să fie pozitiv definit în spațiul variabilelor x a fost sugerată de Arrow şi Sollow [1958]. Mai târziu, această idee a fost adoptată şi extinsă, printre alții, de Hestenes [1969], Powell [1969], Sargent şi Murtagh [1973].

Pentru a utiliza subproblema (31) într-un proces iterativ este necesar să precizăm modul de alegere a unei estimații a multiplicatorilor Lagrange λ_k, precum şi a parametrului de penalizare ρ.

Alegerea lui λ_k

Ideal λ_k trebuie ales cât mai aproape posibil de λ^* pe care însă nu-l cunoaştem. De aceea, cea mai simplă alegere este $\lambda_k = \hat{\lambda}$, unde $\hat{\lambda}$ sunt valorile multiplicatorilor corespunzători restricțiilor liniarizate ale subproblemei de la iterația anterioară. Să presupunem că problema este „pur neliniară", adică nu are restricții liniare. Atunci $\hat{\lambda}$ este soluția sistemului $B^{\mathrm{T}}\hat{\lambda} = g_B$ de la sfârşitul iterației majore anterioare. Mai mult $\hat{\lambda}$ verifică şi sistemul $S^{\mathrm{T}}\hat{\lambda} = g_S$.

Dar, după cum ştim, g este zero pentru toate variabilele ecart şi rezultă imediat că $\hat{\lambda}_i = 0$ dacă a i-a restricție liniarizată este inactivă. Alegerea $\lambda_k = \hat{\lambda}$ asigură deci că o restricție neliniară inactivă va fi exclusă din termenul Lagrangeanului $\lambda_k^T(f - \tilde{f})$ din subproblema de la iterația următoare. Această proprietate este foarte importantă şi recomandă deci această alegere a lui λ_k.

Alegerea lui ρ.

Este bine cunoscut că x^* nu trebuie să fie un minim local al funcției Lagrange cu $\rho = 0$. Dacă presupunem că $J(x^*)$ este de rang plin, atunci există λ^* astfel încât (x^*, y^*) este un punct staționar al funcției

$$L(x,\lambda) = f^0(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y - \lambda^{\mathrm{T}}(f + A_1 y - b_1),$$

unde este posibil ca $L(x^*, \lambda^*)$ să aibă o curbură negativă în raport cu x, în punctul x^*. Cel mai mult ceea ce putem spune este că dacă restricțiile sunt satisfăcute ca egalități în x^* şi le ignorăm pe cele inactive, atunci o condiție necesară (suficientă) pentru ca x^* să fie un minim local este ca:

$$Z(x^*)^{\mathrm{T}} \frac{\partial L(x^*, \lambda^*)}{\partial x} = 0$$

şi

$$Z(x^*)^{\mathrm{T}} \frac{\partial^2 L(x^*, \lambda^*)}{\partial x^2} Z(x^*)$$

este o matrice simetrică pozitiv semidefinită (pozitiv definită), unde $Z(x^*)$ este definit de (20) în care $J(x^*)$ este utilizat în matricea A.

Astfel, dacă restricționăm căutarea minimului la subspațiul restricțiilor liniare definit de $Z(x^*)$, atunci vom căuta într-adevăr un minim al Lagrangeanului şi ne putem aştepta că dacă x_k este suficient de aproape de x^* pentru ca $J(x_k)$ să fie aproape de $J(x^*)$, atunci putem minimiza (31) cu $\rho = 0$ [Robinson, 1972].

Dificultățile apar atunci când x_k este departe de x^*, deoarece în această situație restricțiile liniarizate pot defini un subspațiu în care probabil că un punct şa al Lagrangeanului este mai aproape de x^* decât minimul acestuia. Ca atare, rezolvarea succesivă a lui (31) cu $\rho = 0$ poate deci să nu fie convergentă la x^*. Introducerea termenului de penalizare $\frac{1}{2}\rho(f - \tilde{f})^{\mathrm{T}}(f - \tilde{f})$, pentru $\rho > 0$ suficient de mari, fixează proprietăți de curbură corecte în (31).

Pentru a ilustra importanța termenului de penalizare din (31) să considerăm problema cu restricții egalități

$$\min f^0(x) \tag{33}$$

referitor la:

$$f(x) = 0,$$

unde s-a presupus că toate funcțiile sunt cel puțin de două ori continuu diferențiabile. Presupunem de asemenea că există un punct x^* în care Jacobianul $J(x^*)$ este de rang plin, că există un λ^* astfel încât $\partial f^0 / \partial x = J(x^*)^{\mathrm{T}}\lambda^*$ şi că hesseianul redus $Z(x^*)^{\mathrm{T}}[\partial^2 L(x^*, \lambda^*)/\partial x^2]Z(x^*)$ este pozitiv definit.

Teorema 2. *Fie* (x_k, λ_k) *o soluție aproximativă a lui (33) şi fie* $(\hat{x}, \hat{\lambda})$ *o soluție a subproblemei liniarizate*

$$\min f^0(x) - \lambda_k^{\mathrm{T}}(f - \tilde{f}) + \frac{1}{2}\rho(f - \tilde{f})^{\mathrm{T}}(f - \tilde{f}) \tag{34}$$

referitor la:

$$\tilde{f}(x, x_k) = 0.$$

Dacă $\hat{\lambda} - \lambda_k = \varepsilon_1$ *şi* $f(\hat{x}) = \varepsilon_2$, *atunci* $(\hat{x}, \hat{\lambda})$ *este de asemenea o soluție a problemei perturbate:*

$$\min f^0(x) + (\varepsilon_1 + \rho\varepsilon_2)^{\mathrm{T}}(f - \tilde{f}) \tag{35}$$

referitor la:

$$f(x) = \varepsilon_2,$$

pentru ε_1 *şi* ε_2 *suficient de mici.*

Demonstrație. Dacă $(\hat{x}, \hat{\lambda})$ este o soluție a lui (34), atunci trebuie ca $\tilde{f} = 0$ şi

$$g^0(\hat{x}) - (\hat{J} - J_k)^{\mathrm{T}}\lambda_k + \rho(\hat{J} - J_k)^{\mathrm{T}}(f - \tilde{f}) = J_k^{\mathrm{T}}\hat{\lambda},$$

unde J_k este Jacobianul evaluat în x_k, iar $\hat{J}$, f şi $\tilde{f}$ sunt evaluate în $\hat{x}$. Adunând acum $(\hat{J} - J_k)^{\mathrm{T}}\hat{\lambda}$ la ambii membri ai relației de mai sus şi introducând expresiile pentru ε_1 şi ε_2 obținem:

$$g^0(\hat{x}) - (\hat{J} - J_k)^{\mathrm{T}}\varepsilon_1 + \rho(\hat{J} - J_k)^{\mathrm{T}}\varepsilon_2 = \hat{J}^{\mathrm{T}}\hat{\lambda},$$

care arată că $(\hat{x}, \hat{\lambda})$ verifică condițiile de staționaritate pentru (35). Se poate arăta că matricele Hesseian ale funcțiilor Lagrange asociate problemelor (34) şi (35) diferă numai prin cantitatea $\rho(\hat{J} - J_k)^{\mathrm{T}}(\hat{J} - J_k)$, care este de ordinul $\rho\|\Delta x_k\|$, unde $\Delta x_k = \hat{x} - x_k$. Deci pentru ε_1, ε_2 şi Δx_k suficient de mici, dacă hesseianul redus al lui (34) este pozitiv definit în $\hat{x}$, atunci prin continuitate hesseianul redus al lui (35) va fi de asemenea pozitiv definit, satisfăcând condițiile suficiente de optimalitate ale lui $\hat{x}$ pentru (35). ■

Este foarte instructiv să prezentăm aici un rezultate asemănător pentru problema penalizată convențional.

Teorema 3. *Fie* (x_k, λ_k) *o soluție aproximativă a lui (33) şi fie* $(\hat{x}, \hat{\lambda})$ *o soluție a subproblemei liniarizate*

$$\min f^0(x) - \lambda_k^{\mathrm{T}}(f - \tilde{f}) + \frac{1}{2}\rho f^{\mathrm{T}} f \tag{36}$$

referitor la:

$$\tilde{f}(x, x_k) = 0.$$

Dacă $\hat{\lambda} - \lambda_k = \varepsilon_1$ *şi* $f(\hat{x}) = \varepsilon_2$, *atunci* $(\hat{x}, \hat{\lambda})$ *este de asemenea o soluţie a problemei perturbate:*

$$\min f^0(x) + \varepsilon_1^{\mathrm{T}}(f - \tilde{f}) + \rho\varepsilon_2^{\mathrm{T}} f \tag{37}$$

referitor la:

$$f(x) = \varepsilon_2.$$

Demonstraţia este analogă celei din teorema 2. ■

Deci dacă ε_1 şi ε_2 sunt suficient de mici, atunci cu siguranţă putem reduce ρ la zero, în cadrul procesului de rezolvare succesivă a subproblemelor (31).

Este interesant să notăm că problema (35) este mai puţin sensibilă la variaţiile lui $\hat{x}$ decât (37). Într-adevăr, fie Δx o modificare arbitrar de mică a soluţiei $\hat{x}$ a lui (35). Atunci, funcţia obiectiv a lui (35) diferă de $f^0(x)$ prin cantitatea $\delta_1 = (\varepsilon_1 + \rho\varepsilon_2)^{\mathrm{T}}(f - \tilde{f})$. Clar, $|\delta_1| \le (\|\varepsilon_1\| + \rho\|\varepsilon_2\|)O(\|\Delta x\|^2)$.
Pe de altă parte, pentru problema (37) obţinem

$$\delta_2 = \varepsilon_1^{\mathrm{T}}(f - \tilde{f}) + \rho\varepsilon_2^{\mathrm{T}} f = \varepsilon_1^{\mathrm{T}}(f - \tilde{f}) + \rho\varepsilon_2^{\mathrm{T}}(\hat{f} + \hat{J}\Delta x + O(\|\Delta x\|^2)).$$

Dar $|\delta_2| \le (\|\varepsilon_1\| + \rho\|\varepsilon_2\|)O(\|\Delta x\|^2) + \rho\|\varepsilon_2\|^2 + \rho\left\|\hat{J}^{\mathrm{T}}\varepsilon_2\right\|\|\Delta x\|$. Deoarece δ_1 este de ordinul lui $\|\Delta x\|^2$, în timp ce δ_2 este de ordinul lui $\|\Delta x\|$ rezultă că termenul de penalizare modificat din (34) conduce la proprietăţi superioare de convergenţă faţă de termenul de penalizare din (36).

Cu acestea algoritmul metodei MINOS pentru rezolvarea problemei generale neliniare (28) este următorul.

Algoritmul 2. (Algoritmul MINOS – restricţii neliniare)

Pasul 1.	Se pune $k = 0$. Se aleg estimaţiile iniţiale x_k, y_k şi λ_k. Se precizează o valoare a parametrului de penalizare $\rho > 0$ şi o toleranţă de convergenţă $\varepsilon_c > 0$.
Pasul 2.	Rezolvarea subproblemei liniare. a) Date x_k, y_k, λ_k şi ρ, utilizând algoritmul 8.1 se rezolvă subproblema cu restricţii liniare (31) obţinându-se astfel x_{k+1}, y_{k+1} şi u, unde u este vectorul multiplicatorilor Lagrange asociaţi restricţiilor din (31). b) B) Se formează vectorul λ_{k+1} care conţine primele m_1 componente ale lui u.
Pasul 3.	Se execută testul de convergenţă. Dacă (x_k, y_k) verifică condiţiile de optimalitate Karush-Kuhn-Tucker, atunci stop.
Pasul 4.	Dacă

	$\dfrac{\left\lVert f(x_{k+1}) + A_1 y_{k+1} - b_1 \right\rVert}{1 + \left\lVert [x_{k+1} \;\; y_{k+1}] \right\rVert} \le \varepsilon_c$, şi $\dfrac{\left\lVert \lambda_{k+1} - \lambda_k \right\rVert}{1 + \left\lVert \lambda_{k+1} \right\rVert} \le \varepsilon_c$, Atunci se pune $\rho = 0$; altfel se pune $\rho = 10\rho$.
Pasul 5.	Se reliniarizează restricţiile neliniare în jurul noului punct x_{k+1}, se pune $k = k+1$ şi se continuă cu pasul 2. ♦

În continuare prezentăm performanţele algoritmului MINOS în implementarea dată de Saunders şi Murtagh [1982].

Exemplul 2. Fie problema de programare neliniară [Wright (No.4), 1976], [Murtagh şi Saunders, 1982]:

$$\min (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4$$

referitor la:

$$x_1 + x_2^2 + x_3^3 = 2 + 3\sqrt{2},$$
$$x_2 - x_3^2 + x_4 = -2 + 2\sqrt{2},$$
$$x_1 x_5 = 2.$$

Considerând punctele iniţiale:

$$A = [1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]^{\mathrm{T}}, \quad B = [2 \;\; 2 \;\; 2 \;\; 2 \;\; 2]^{\mathrm{T}}, \quad C = [-1 \;\; 3 \;\; -0.5 \;\; -2 \;\; -3]^{\mathrm{T}},$$
$$D = [-1 \;\; 2 \;\; 1 \;\; -2 \;\; -2]^{\mathrm{T}}, \quad E = [-2 \;\; -2 \;\; -2 \;\; -2 \;\; -2]^{\mathrm{T}},$$

algoritmul 2 (MINOS pentru restricţii neliniare) furnizează următoarele soluţii:

$$A, B \to x_1^* = [1.11663 \;\; 1.22043 \;\; 1.53779 \;\; 1.97277 \;\; 1.79110]^{\mathrm{T}},$$
$$f(x_1^*) = 0.0029307,$$
$$C \to x_2^* = [-0.703393 \;\; 2.63570 \;\; -0.0963618 \;\; -1.79799 \;\; -2.84336]^{\mathrm{T}},$$
$$f(x_2^*) = 44.022089,$$
$$D \to x_3^* = [-1.27305 \;\; 2.41035 \;\; 1.19486 \;\; -0.154239 \;\; -1.57103]^{\mathrm{T}},$$
$$f(x_3^*) = 27{,}8718714,$$
$$E \to x_4^* = [-2.79087 \;\; -3.00414 \;\; -.20538 \;\; 3.87474 \;\; -0.71662]^{\mathrm{T}},$$
$$f(x_4^*) 607{,}03036.$$

Tabelul 2 conţine caracteristicile procesului de rezolvare a acestei probleme, unde $\#itm$ este numărul de iteraţii majore, $\#it$ este numărul de iteraţii (minore) necesare rezolvării subproblemelor cu restricţii liniare şi $\#evf$ este numărul de evaluări ale funcţiei obiectiv şi ale restricţiilor, inclusiv gradienţii acestor funcţii.

Tabelul 2.
Comparaţie între procesele de optimizare ale algoritmului MINOS, pentru diferite iniţializări ale problemei 2. $\varepsilon_h = 10^{-16}$.

	A	B	C	D	E
$\#itm$	8	9	7	6	8
$\#it$	30	21	14	17	21
$\#evf$	64	49	35	35	55

Observăm caracterul local al algoritmului MINOS. Pentru diferite puncte iniţiale se obţin diferite soluţii, fiecare dintre acestea necesitând eforturi de calcul diferite.

Exemplul 3. Să ilustrăm funcţionarea algoritmului MINOS pe problema neliniară [Wright (No.9), 1976], [Murtagh şi Saunders, 1982].

$$\min 10x_1x_4 - 6x_2^2x_3 + x_1^3x_2 + 9\sin(x_5 - x_3) + x_2^3x_4^2x_5^4$$

referitor la:

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \le 20,$$
$$x_1^2x_3 + x_4x_5 \ge -2,$$
$$x_2^2x_4 + 10x_1x_5 \ge 5.$$

Pentru punctele iniţiale:

$$A = [1\ 1\ 1\ 1\ 1]^{\mathrm{T}},\ B = [1\ -3\ 1\ -1\ 2]^{\mathrm{T}},\ C = [5\ -5\ 1\ -3\ 1]^{\mathrm{T}},$$
$$D = [10\ -1\ 1\ 10\ 2]^{\mathrm{T}},$$

algoritmul 2 furnizează soluţiile:

$A \to x_1^* = [-0.08145\ \ 3.69238\ \ 2.48741\ \ 0.37713\ \ 0.17398]^{\mathrm{T}}$,

$f(x_1^*) = -210.407817$,

$B, C \to x_2^* = [1.47963\ \ -2.63661\ \ 1.05467\ \ -1.61151\ \ 2.67388]^{\mathrm{T}}$,

$f(x_2^*) = -2500.584488$,

$D \to x_3^* = [-0.0774\ \ -2.58139\ \ 0.01143\ \ 2.10766\ \ 2.98229]^{\mathrm{T}}$,

$f(x_3^*) = -6043.539113$.

Tabelul 3 conţine caracteristicile procesului de rezolvare a problemei considerate.

Tabelul 3.
Comparaţie între procesele de optimizare ale algoritmului MINOS, pentru diferite iniţializări ale problemei 3. $\varepsilon_h = 10^{-16}$.

	A	B	C	D	E
$\#itm$	10	8	20	20	9
$\#it$	51	25	105	105	46
$\#evf$	136	64	195	195	97

3. STUDIU NUMERIC. REZOLVAREA UNOR APLICAȚII

În cele ce urmează prezentăm performanțele algoritmului MINOS în ceea ce priveşte rezolvarea unor aplicații concrete de programare matematică neliniară. În acest sens vom considera 15 aplicații extrase din lucrarea [Andrei, 2003]. Tabelul 4 conține caracteristicile procesului de rezolvare, unde *n* = numărul de variabile, *m* = numărul de restricții, *#vs* = numărul variabilelor superbazice, *#fac* = numărul de factorizări ale bazei şi *vfo* = valoarea funcției obiectiv[1].

Tabelul 4.
Caracteristicile procesului de rezolvare a aplicațiilor de programare matematică considerate în acest studiu numeric. Algoritmul MINOS.

	n	*m*	*#itm*	*#it*	*#evf*	*#vs*	*#fac*	*vfo*
A1	6	2	8	31	61	4	9	135.07595549
A3	8	11	38	97	307	0	39	0.34173955312
A4	9	6	13	54	64	1	14	5055.0118035
A6	12	13	5	32	63	0	6	146.25000000
A7	20	1	16	520	1311	19	17	5.0690569643
A9	10	15	11	71	153	0	12	-4430.0879298
A14	8	6	8	39	83	4	9	3.9511635079
A15	10	3	3	19	46	7	4	-47.761090859
A17	13	15	7	60	114	1	8	97.591034664
A18	16	21	10	111	200	2	12	174.92584076
A19	10	11	11	24	60	1	12	-1768.8069633
A31	8	6	48	206	577	2	49	7049.2480257
A37	4	7	15	28	70	0	16	1.0898639714
A41	9	9	17	37	71	0	18	2.5486006623
A42	7	12	9	43	95	0	10	6.2746343365

<u>Aplicația A1</u> (*Proiectarea unui transformator*)

$$min\ 0,0204x_1x_4\left(x_1+x_2+x_3\right)+0,0187x_2x_3\left(x_1+1,57x_2+x_4\right)+$$
$$0,0607x_1x_4x_5^2\left(x_1+x_2+x_3\right)+0,0437x_2x_3x_6^2\left(x_1+1,57x_2+x_4\right)$$

referitor la:

$$0,001x_1x_2x_3x_4x_5x_6-2,07\geq 0\,,$$

$$1-0,00062x_1x_4x_5^2(x_1+x_2+x_3)-0,00058x_2x_3x_6^2\left(x_1+1,57x_2+x_4\right)\geq 0.$$

[1] Detalii asupra procesului de optimizare cât şi descrierea fişierelor asociate aplicațiilor considerate se găsesc în lucrarea: N. Andrei, *MINOS - Nonlinear Programming Problems solved by MINOS package*. Technical Report No.1 / 2011, Bucharest, January 3, 2011; care se găseşte pe CD-ul ataşat acestui raport tehnic.

Aplicația A3 (*Stabilitatea robustă a sistemelor liniare*)

$$min\ x_6$$

referitor la:

$$\begin{aligned}
&x_5x_7-(x_4+10x_2+10x_3)x_7+2x_1=0,\\
&(x_2+x_3+10)x_7-10x_4-x_1=0,\\
&x_4-x_3x_8=0,\\
&x_5-x_7=0,\\
&x_2-x_8=0,\\
&-x_1+800x_6+800\geq 0,\\
&x_1+800x_6-800\geq 0,\\
&-x_2+2x_6+4\geq 0,\\
&x_2+2x_6-4\geq 0,\\
&-x_3+3x_6+6\geq 0,\\
&x_3+3x_6-6\geq 0.
\end{aligned}$$

Aplicația A4 (*Planificarea puterii statice electrice*)

$$min\ 3000x_1+1000x_1^3+2000x_2+666{,}667x_2^3$$

referitor la

$$\begin{aligned}
&0{,}4-x_1+2cx_5^2-x_5x_6(dy_1+cy_2)-x_5x_7(dy_3+cy_4)=0,\\
&0{,}4-x_2+2cx_6^2+x_5x_6(dy_1-cy_2)+x_6x_7(dy_5-cy_6)=0,\\
&0{,}8+2cx_7^2+x_5x_7(dy_3-cy_4)-x_6x_7(dy_5+cy_6)=0,\\
&0{,}2-x_3+2dx_5^2+x_5x_6(cy_1-dy_2)+x_5x_7(cy_3-dy_4)=0,\\
&0{,}2-x_4+2dx_6^2-x_5x_6(cy_1+dy_2)-x_6x_7(cy_5+dy_6)=0,\\
&-0{,}337+2dx_7^2-x_5x_7(cy_3+dy_4)+x_6x_7(cy_5-dy_6)=0,
\end{aligned}$$

$$\begin{aligned}
&y_1=\sin(x_8), && y_2=\cos(x_8),\\
&y_3=\sin(x_9), && y_4=\cos(x_9),\\
&y_5=\sin(x_8-x_9), && y_6=\cos(x_8-x_9),\\
&c=\frac{48{,}4}{50{,}176}\sin(0{,}25), && d=\frac{48{,}4}{50{,}176}\cos(0{,}25).
\end{aligned}$$

<u>Aplicația A6</u> (*Compactizarea circuitelor electrice*)

$$min\ x_1x_7 + x_2x_8 + x_3x_9 + x_4x_{10} + x_5x_{11} + x_6x_{12}$$

referitor la:

$$-x_4 + x_5 = 0,$$
$$-x_1 + x_2 - x_4 + x_5 = 0,$$
$$-x_1 + x_3 - x_4 + x_6 = 0,$$
$$-x_9 + x_{12} = 0,$$
$$-x_7 - x_8 - x_9 + x_{10} + x_{11} + x_{12} = 0,$$
$$x_2 - x_3 \geq 1, \qquad x_7 - x_{10} \geq 1,$$
$$x_1x_7 \geq 30, \qquad x_2x_8 \geq 20,$$
$$x_3x_9 \geq 20, \qquad x_4x_{10} \geq 25,$$
$$x_5x_{11} \geq 15, \qquad x_6x_{12} \geq 20.$$

<u>Aplicația A7</u> (*Curba câinelui - lănțişorul*)

$$min\ h\sum_{i=1}^{n+1}\left(\frac{x_i + x_{i-1}}{2}\right)\sqrt{1+\left(\frac{x_i - x_{i-1}}{h}\right)^2}$$

referitor la:

$$\sum_{i=1}^{n+1}\sqrt{1+\left(\frac{x_i - x_{i-1}}{h}\right)^2} = \frac{L}{h},$$

unde:

$$h = \frac{1}{n+1}, \quad x_0 = a, \quad x_{n+1} = b.$$

<u>Aplicația A9</u> (*Optimizarea procesului de strunjire a unui arbore*)

$$min\left(-20000\frac{(0{,}15x_1 + 14x_2 - 0{,}06)}{(0{,}002 + x_1 + 60x_2)}\right)$$

referitor la:

$$x_1 - \frac{0{,}75}{(x_3x_4)} \geq 0,$$

$$x_1 - \frac{x_9}{(x_5 x_4)} \geq 0,$$

$$x_1 - \frac{x_{10}}{(x_6 x_4)} - \frac{10}{x_4} \geq 0,$$

$$x_1 - \frac{0{,}19}{(x_4 x_7)} - \frac{10}{x_4} \geq 0,$$

$$x_1 - \frac{0{,}125}{(x_4 x_8)} \geq 0,$$

$$10000x_2 - 0{,}00131 x_9 x_5^{0,666} x_4^{1,5} \geq 0,$$

$$10000x_2 - 0{,}001038 x_{10} x_6^{1,6} x_4^{3} \geq 0,$$

$$10000x_2 - 0{,}000223 x_7^{0,666} x_4^{1,5} \geq 0,$$

$$10000x_2 - 0{,}000076 x_8^{3,55} x_4^{5,66} \geq 0,$$

$$10000x_2 - 0{,}000698 x_3^{1,2} x_4^{2} \geq 0,$$

$$10000x_2 - 0{,}00005 x_3^{1,6} x_4^{3} \geq 0,$$

$$10000x_2 - 0{,}00000654 x_3^{2,42} x_4^{4,17} \geq 0,$$

$$10000x_2 - 0{,}000257 x_3^{0,666} x_4^{1,5} \geq 0,$$

$$30 - 2{,}003 x_4 x_5 - 1{,}885 x_6 x_4 - 0{,}184 x_8 x_4 - 2x_3^{0,803} x_4 \geq 0,$$

$$x_9 + x_{10} - 0{,}255 = 0.$$

<u>Aplicația A14</u> (*Proiectarea unui reactor chimic*)

$$min\ 0{,}4x_1^{0,67} x_7^{-0,67} + 0{,}4x_2^{0,67} x_8^{-0,67} + 10 - x_1 - x_2$$

referitor la:

$$1 - 0{,}0588 x_5 x_7 - 0{,}1 x_1 \geq 0,$$

$$1 - 0{,}0588 x_6 x_8 - 0{,}1 x_1 - 0{,}1 x_2 \geq 0,$$

$$1 - 4x_3 x_5^{-1} - 2x_3^{-0,71} x_5^{-1} - 0{,}0588 x_3^{-1,3} x_7 \geq 0,$$

$$1 - 4x_4 x_6^{-1} - 2x_4^{-0,71} x_6^{-1} - 0{,}0588 x_4^{-1,3} x_8 \geq 0$$

$$0{,}4x_1^{0,67} x_7^{-0,67} + 0{,}4x_2^{0,67} x_8^{-0,67} + 10 - x_1 - x_2 \geq 1,$$

$$0{,}4x_1^{0{,}67}x_7^{-0{,}67} + 0{,}4x_2^{0{,}67}x_8^{-0{,}67} + 10 - x_1 - x_2 \le 4{,}2 .$$

<u>Aplicația A15</u> (*Echilibrul chimic*)

$$min \sum_{j=1}^{10} x_j \left(c_j + \ln \frac{x_j}{x_1 + \cdots + x_{10}} \right)$$

referitor la:

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 ,$$
$$x_4 + 2x_5 + x_6 + x_7 - 1 = 0 ,$$
$$x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 .$$

<u>Aplicația A17</u> (*Optimizarea procesului de separare într-o membrană cu trei straturi*)

$$min\ x_{11} + x_{12} + x_{13}$$

referitor la:

$$x_3 - x_2 \ge 0 ,$$
$$x_2 - x_1 \ge 0 ,$$
$$1 - 0{,}002x_7 + 0{,}002x_8 \ge 0 ,$$
$$x_{13} - 1{,}262626x_{10} + 1{,}231059x_3x_{10} \ge 0 ,$$
$$x_5 - 0{,}03475x_2 - 0{,}975x_2x_5 + 0{,}00975x_2^2 \ge 0 ,$$
$$x_6 - 0{,}03475x_3 - 0{,}975x_3x_6 + 0{,}00975x_3^2 \ge 0 ,$$
$$x_5x_7 - x_1x_8 - x_4x_7 + x_4x_8 \ge 0 ,$$
$$1 - 0{,}002\left(x_2x_9 + x_5x_8 - x_1x_8 - x_6x_9\right) - x_5 - x_6 \ge 0 ,$$
$$x_2x_9 - x_3x_{10} - x_6x_9 - 500x_2 + 500x_6 + x_2x_{10} \ge 0 ,$$
$$x_2 - 0{,}9 - 0{,}002\left(x_2x_{10} - x_3x_{10}\right) \ge 0 ,$$
$$x_4 - 0{,}03475x_1 - 0{,}975x_1x_4 + 0{,}00975x_1^2 \ge 0 ,$$
$$x_{11} - 1{,}262626x_8 + 1{,}231059x_1x_8 \ge 0 ,$$
$$x_{12} - 1{,}262626x_9 + 1{,}231059x_2x_9 \ge 0 ,$$
$$x_{11} + x_{12} + x_{13} \ge 50 ,$$
$$x_{11} + x_{12} + x_{13} \le 250 .$$

<u>Aplicația A18</u> (*Optimizarea procesului de separare într-o membrană cu cinci straturi*)

$$min\ 1{,}262626\left(x_{12}+x_{13}+x_{14}+x_{15}+x_{16}\right)-$$
$$-1{,}231060\left(x_1x_{12}+x_2x_{13}+x_3x_{14}+x_4x_{15}+x_5x_{16}\right)$$

referitor la:

$$x_6-0{,}03475x_1-0{,}975x_1x_6+0{,}00975x_1^2\geq 0\,,$$
$$x_7-0{,}03475x_2-0{,}975x_2x_7+0{,}00975x_2^2\geq 0\,,$$
$$x_8-0{,}03475x_3-0{,}975x_3x_8+0{,}00975x_3^2\geq 0\,,$$
$$x_9-0{,}03475x_4-0{,}975x_4x_9+0{,}00975x_4^2\geq 0\,,$$
$$x_{10}-0{,}03475x_5-0{,}975x_5x_{10}+0{,}00975x_5^2\geq 0\,,$$
$$x_7x_{11}-x_6x_{11}-x_1x_{12}+x_6x_{12}\geq 0\,,$$
$$x_8-x_7-0{,}002x_7x_{12}-0{,}002x_2x_{13}+0{,}002x_8x_{13}+$$
$$+0{,}002x_1x_{12}\geq 0\,,$$
$$1-x_8-0{,}002x_8x_{13}-0{,}002x_3x_{14}-x_9+0{,}002x_2x_{13}+$$
$$+0{,}002x_9x_{14}\geq 0\,,$$
$$x_3x_{14}-x_9x_{14}-x_4x_{15}-500x_{10}+500x_9+x_8x_{15}\geq 0\,,$$
$$x_4x_{15}-x_5x_{16}-x_{10}x_{15}-500x_4+x_4x_{16}+500x_{10}\geq 0\,,$$
$$x_4-0{,}002x_4x_{16}+0{,}002x_5x_{16}-0{,}9\geq 0\,,$$
$$1-0{,}002x_{11}+0{,}002x_{12}\geq 0\,,$$
$$x_{11}-x_{12}\geq 0\,,$$
$$x_5-x_4\geq 0\,,$$
$$x_4-x_3\geq 0\,,$$
$$x_3-x_2\geq 0\,,$$
$$x_2-x_1\geq 0\,,$$
$$x_{10}-x_9\geq 0\,,$$
$$x_9-x_8\geq 0\,,$$
$$1{,}262626\left(x_{12}+x_{13}+x_{14}+x_{15}+x_{16}\right)$$
$$-1{,}231060\left(x_1x_{12}+x_2x_{13}+x_3x_{14}+x_4x_{15}+x_5x_{16}\right)-50\geq 0\,,$$
$$-1{,}262626\left(x_{12}+x_{13}+x_{14}+x_{15}+x_{16}\right)$$
$$+1{,}231060\left(x_1x_{12}+x_2x_{13}+x_3x_{14}+x_4x_{15}+x_5x_{16}\right)+250\geq 0\,.$$

Aplicația A19 (*Optimizarea unui proces de alkilare. Varianta 1*)

$$min\ 5{,}04x_1 + 0{,}035x_2 + 10x_3 + 3{,}36x_5 - 0{,}063x_4x_7$$

referitor la:

$$35{,}82 - 0{,}222x_{10} - bx_9 \geq 0\,,$$
$$-133 + 3x_7 - ax_{10} \geq 0\,,$$
$$-35{,}82 + 0{,}222x_{10} + bx_9 + (1/b - b)x_9 \geq 0\,,$$
$$133 - 3x_7 + ax_{10} + (1/a - a)x_{10} \geq 0\,,$$
$$1{,}12x_1 + 0{,}13167x_1x_8 - 0{,}00667x_1x_8^2 - ax_4 \geq 0\,,$$
$$57{,}425 + 1{,}098x_8 - 0{,}038x_8^2 + 0{,}325x_6 - ax_7 \geq 0\,,$$
$$-1{,}12x_1 - 0{,}13167x_1x_8 + 0{,}00667x_1x_8^2 + ax_4 + (1/a - a)x_4 \geq 0\,,$$
$$-57{,}425 - 1{,}098x_8 + 0{,}038x_8^2 - 0{,}325x_6 + ax_7 + (1/a - a)x_7 \geq 0\,,$$
$$1{,}22x_4 - x_1 - x_5 = 0\,,$$
$$98000x_3 / \left(x_4x_9 + 1000x_3\right) - x_6 = 0\,,$$
$$\left(x_2 + x_5\right) / x_1 - x_8 = 0\,,$$

unde a=0,99 şi b=0,9 .

Aplicația A31 (*Proiectarea unui schimbător de căldură*)

$$min\ x_1 + x_2 + x_3$$

referitor la:

$$1 - 0{,}0025\left(x_4 + x_6\right) \geq 0\,,$$
$$1 - 0{,}0025\left(x_5 + x_7 - x_4\right) \geq 0\,,$$
$$1 - 0{,}01\left(x_8 - x_5\right) \geq 0\,,$$
$$x_1x_6 - 833{,}33252x_4 - 100x_1 + 83333{,}333 \geq 0\,,$$
$$x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0\,,$$
$$x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geq 0\,.$$

Aplicația A37 (*Analiza stabilității robuste a unui sistem liniar dinamic*)

$$min\, x_4$$

referitor la:

$$-x_1^4x_2^4 + x_1^4 + x_2^4x_3 \geq 0\,,$$

$$0{,}25x_4 + x_1 - 1{,}4 \geq 0,$$
$$0{,}25x_4 - x_1 + 1{,}4 \geq 0,$$
$$0{,}2x_4 + x_2 - 1{,}5 \geq 0,$$
$$0{,}2x_4 - x_2 + 1{,}5 \geq 0,$$
$$0{,}2x_4 + x_3 - 0{,}8 \geq 0,$$
$$0{,}2x_4 - x_3 + 0{,}8 \geq 0.$$

Aplicaţia A41 (*Analiza stabilităţii sistemului de aprindere a motorului Fiat Dedra*)

$$min\ x_9$$

referitor la:

$$-a_6x_8^6 + a_4x_8^4 - a_2x_8^2 + a_0 = 0,$$
$$a_7x_8^6 - a_5x_8^4 + a_3x_8^2 - a_1 = 0,$$
$$-1{,}2721x_9 - x_1 + 3{,}4329 \leq 0,$$
$$-0{,}06x_9 - x_2 + 0{,}1627 \leq 0,$$
$$-0{,}0782x_9 - x_3 + 0{,}1139 \leq 0,$$
$$-0{,}3068x_9 + x_4 - 0{,}2539 \leq 0,$$
$$-0{,}0108x_9 - x_5 + 0{,}0208 \leq 0,$$
$$-2{,}4715x_9 + x_6 - 2{,}0247 \leq 0,$$
$$-9x_9 + x_7 - 1 \leq 0,$$

unde:

$$a_0 = 6{,}82079(10^{-5})x_1x_3x_4^2 + 6{,}82079(10^{-5})x_1x_2x_4x_5,$$

$$a_1 = 0{,}00076176x_2^2x_5^2 + 0{,}00076176x_3^2x_4^2 + 0{,}000402141x_1x_2x_5^2 + 0{,}00337606x_1x_3x_4^2 + 6{,}82079(10^{-5})x_1x_4x_5 + 0{,}00051612x_2^2x_5x_6 + 0{,}00337606x_1x_2x_4x_5 + 6{,}82079(10^{-5})x_1x_2x_4x_7 + 6{,}28987(10^{-5})x_1x_2x_5x_6 + 0{,}000402141x_1x_3x_4x_5 + 6{,}28987(10^{-5})x_1x_3x_4x_6 + 0{,}00152352x_2x_3x_4x_5 + 0{,}00051612x_2x_3x_4x_6,$$

$$a_2 = 0{,}000402141x_1x_5^2 + 0{,}00152352x_2x_5^2 + 0{,}0552x_2^2x_5^2 + 0{,}0552x_3^2x_4^2 + 0{,}0189477x_1x_2x_5^2 + 0{,}034862x_1x_3x_4^2 + 0{,}00336706x_1x_4x_5 + 6{,}82079(10^{-5})x_1x_4x_7 + 6{,}28987(10^{-5})x_1x_5x_6 + 0{,}00152352x_3x_4x_5 + 0{,}00051612x_3x_4x_6 - 0{,}00234048x_3^2x_4x_6 + 0{,}034862x_1x_2x_4x_5 + 0{,}0237398x_2^2x_5x_6 + 0{,}00152352x_2^2x_5x_7 +$$

$0{,}00051612x_2^2x_6x_7 + 0{,}00336706x_1x_2x_4x_7 + 0{,}00287416x_1x_2x_5x_6 +$
$0{,}000804282x_1x_2x_5x_7 + 6{,}28987(10^{-5})x_1x_2x_6x_7 + 0{,}0189477x_1x_3x_4x_5 +$
$0{,}00287416x_1x_3x_4x_6 + 0{,}000402141x_1x_3x_4x_7 + 0{,}1104x_2x_3x_4x_5 +$
$0{,}0237398x_2x_3x_4x_6 + 0{,}00152352x_2x_3x_4x_7 - 0{,}00234048x_2x_3x_5x_6 +$
$0{,}00103224x_2x_5x_6$,

$a_3 = 0{,}189477x_1x_5^2 + 0{,}1104x_2x_5^2 + 0{,}00051612x_5x_6 + x_2^2x_5^2 +$
$0{,}00076176x_2^2x_7^2 + x_3^2x_4^2 + 0{,}1586x_1x_2x_5^2 + 0{,}000402141x_1x_2x_7^2 +$
$0{,}0872x_1x_3x_4^2 + 0{,}034862x_1x_4x_5 + 0{,}00336706x_1x_4x_7 +$
$0{,}00287416x_1x_5x_6 + 6{,}28987(10^{-5})x_1x_6x_7 + 0{,}00103224x_2x_6x_7 +$
$0{,}1104x_3x_4x_5 + 0{,}0237398x_3x_4x_6 + 0{,}00152352x_3x_4x_7 -$
$0{,}00234048x_3x_5x_6 + 0{,}1826x_2^2x_5x_6 + 0{,}1104x_2^2x_5x_7 +$
$0{,}0237398x_2^2x_6x_7 - 0{,}0848x_3^2x_4x_6 + 0{,}0872x_1x_2x_4x_5 +$
$0{,}034862x_1x_2x_4x_7 + 0{,}0215658x_1x_2x_5x_6 + 0{,}0378954x_1x_2x_5x_7 +$
$0{,}00287416x_1x_2x_6x_7 + 0{,}1586x_1x_3x_4x_5 + 0{,}0215658x_1x_3x_4x_6 +$
$0{,}0189477x_1x_3x_4x_7 + 2x_2x_3x_4x_5 + 0{,}1826x_2x_3x_4x_6 +$
$0{,}1104x_2x_3x_4x_7 - 0{,}0848x_2x_3x_5x_6 - 0{,}00234048x_2x_3x_6x_7 +$
$0{,}00076176x_5^2 + 0{,}0474795x_2x_5x_6 + 0{,}000804282x_1x_5x_7 +$
$0{,}00304704x_2x_5x_7$,

$a_4 = 0{,}1586x_1x_5^2 + 0{,}000402141x_1x_7^2 + 2x_2x_5^2 + 0{,}00152352x_2x_7^2 +$
$0{,}0237398x_5x_6 + 0{,}00152352x_5x_7 + 0{,}00051612x_6x_7 +$
$0{,}0552x_2^2x_7^2 + 0{,}0189477x_1x_2x_7^2 + 0{,}0872x_1x_4x_5 +$
$0{,}034862x_1x_4x_7 + 0{,}0215658x_1x_5x_6 + 0{,}00287416x_1x_6x_7 +$
$0{,}0474795x_2x_6x_7 + 2x_3x_4x_5 + 0{,}1826x_3x_4x_6 + 0{,}1104x_3x_4x_7 -$
$0{,}0848x_3x_5x_6 - 0{,}00234048x_3x_6x_7 + 2x_2^2x_5x_7 + 0{,}1826x_2^2x_6x_7 +$
$0{,}0872x_1x_2x_4x_7 + 0{,}3172x_1x_2x_5x_7 + 0{,}0215658x_1x_2x_6x_7 +$
$0{,}1586x_1x_3x_4x_7 + 2x_2x_3x_4x_7 - 0{,}0848x_2x_3x_6x_7 + 0{,}0552x_5^2 +$
$0{,}3652x_2x_5x_6 + 0{,}0378954x_1x_5x_7 + 0{,}2208x_2x_5x_7$,

$a_5 = 0{,}0189477x_1x_7^2 + 0{,}1104x_2x_7^2 + 0{,}1826x_5x_6 + 0{,}1104x_5x_7 +$
$0{,}0237398x_6x_7 + x_2^2x_7^2 + 0{,}1586x_1x_2x_7^2 + 0{,}0872x_1x_4x_7 +$
$0{,}0215658x_1x_6x_7 + 0{,}3652x_2x_6x_7 + 2x_3x_4x_7 - 0{,}0848x_3x_6x_7 +$
$x_5^2 + 0{,}00076176x_7^2 + 0{,}3172x_1x_5x_7 + 4x_2x_5x_7$,

$a_6 = 0{,}1586x_1x_7^2 + 2x_2x_7^2 + 2x_5x_7 + 0{,}1826x_6x_7 + 0{,}0552x_7^2$,

$a_7 = x_7^2$.

<u>Aplicația A42</u> (*Analiza stabilității unui sistem mecanic*)

$$\min x_7$$

referitor la:

$$a_4x_6^4 - a_2x_6^2 + a_0 = 0,$$
$$a_3x_6^2 - a_1 = 0,$$
$$-x_1 - x_7 + 10 \le 0,$$
$$x_1 - x_7 - 10 \le 0,$$
$$-x_2 - 0{,}1x_7 + 1 \le 0,$$
$$x_2 - 0{,}1x_7 - 1 \le 0,$$
$$-x_3 - 0{,}1x_7 + 1 \le 0,$$
$$x_3 - 0{,}1x_7 - 1 \le 0,$$
$$-x_4 - 0{,}01x_7 + 0{,}2 \le 0,$$
$$x_4 - 0{,}01x_7 - 0{,}2 \le 0,$$
$$-x_5 - 0{,}005x_7 + 0{,}05 \le 0,$$
$$x_5 - 0{,}005x_7 - 0{,}05 \le 0,$$

unde:

$$a_0 = 54{,}387x_2x_3,$$
$$a_1 = \frac{1}{5}\left(-147{,}15x_2x_3x_4 + 1364{,}67x_2x_3 - 27{,}72x_5\right),$$
$$a_2 = 3\left(-9{,}81x_2^2x_3 - 9{,}81x_1x_2x_3 - 4{,}312x_2x_3^2 + 264{,}896x_2x_3\right) + 3\left(x_4x_5 - 9{,}274x_5\right),$$
$$a_3 = 7x_2x_3^2x_4 - 64{,}918x_2x_3^2 + 380{,}067x_2x_3 + 3x_2x_5 + 3x_1x_5,$$
$$a_4 = 7x_1x_2x_3^2 + 4x_2^2x_3^2.$$

4. STUDIU NUMERIC. REZOLVAREA UNOR APLICAȚII ÎN TEHNOLOGIA GAMS

În această secțiune vom prezenta performanțele algoritmului MINOS în implementarea dată de Murtagh şi Saunders [1995] în ceea ce priveşte rezolvarea unui număr de 10 aplicații de programare matematică neliniară în tehnologia GAMS. Toate aceste aplicații sunt extrase din lucrarea [Andrei, 2011]. Pentru fiecare dintre acestea se prezintă: expresia algebrică, expresia în limbajul GAMS, precum şi performanțele algoritmului MINOS versus CONOPT şi KNITRO.

Aplicația G1 (*Polygon*)
Dintre toate poligoanele cu n_v *laturi şi diametrul* $d \le 1$, *să se determine poligonul de arie maximă.*
Problema este descrisă în [Dolan, Moré şi Munson, 2004, pagina 3] şi [Graham, 1975]. Dacă (r_i, θ_i) sunt coordonatele vârfurilor unui poligon, atunci expresia matematică a problemei de optimizare este:

$$\max\left(\frac{1}{2}\sum_{i=1}^{n_v-1} r_{i+1} r_i \sin(\theta_{i+1}-\theta_i)\right)$$

referitor la

$$\begin{aligned} & r_i^2 + r_j^2 - 2r_i r_j \cos(\theta_i - \theta_j) \le 1, && 1 \le i < n_v, \quad i < j \le n_v, \\ & \theta_i \le \theta_{i+1}, && 1 \le i < n_v, \\ & \theta_i \in [0,\pi], \quad r_i \ge 0, && 1 \le i \le n_v. \end{aligned}$$

Problema este interesantă deoarece are n_v^2 restricții inegalități neliniare neconvexe. Dacă $n_v \to \infty$, atunci aria maximă tinde la aria unui cerc cu diametrul egal cu 1, de valoare $\pi/4 \cong 0.7854$. Problema este descrisă şi în [Andrei, 2001, pagina 328].
Expresia GAMS a acestei probleme este prezentată în figura 1.1.

```
$ontext
Polygon
Determinarea celui mai mare poligon cu n laturi de diametru
d <=1.
$offtext

$if     set n  $set nv %n%
$if not set nv $set nv 300
set i sides /i1 * i%nv%/;
alias(i,j)

scalar pi

positive variables
  r(i)
```

```
  theta(i)
variable
  polygon_area;

equations
  obj
  distance(i,j)
  ordered(i);

* equatiile
obj.. polygon_area =E= 0.5 * sum(j(i+1),
      r(i)*r(i+1)*sin(theta(i+1)-theta(i)));
distance(i,j)$(ord(j)>ord(i)).. sqr(r(i))+sqr(r(j))-
                  2*r(i)*r(j)*cos(theta(j)-theta(i)) =L=1;
ordered(i+1).. theta(i) =L= theta(i+1);

pi = 2*arctan(inf);
r.up(i) = 1;
theta.up(i) = pi;

r.l(i) = 4*ord(i)*(card(i)+1-ord(i))/sqr(card(i)+1);
theta.l(i) = pi*ord(i)/card(i);

model polygon /all/;

$onecho >bench.opt
solvers conopt knitro minos
$offecho

polygon.optfile=1;
polygon.iterlim=50000;
option nlp=bench
option reslim = 3600;
solve polygon using nlp maximizing polygon_area;
display polygon_area.l;
* End polygon
```

Fig. 1.1. Expresia GAMS a aplicației G1 (Polygon).

Pentru diverse valori ale lui n_v performanțele algoritmilor CONOPT, KNITRO şi MINOS sunt descrise în tabelele de mai jos, unde **#iter** este numărul de iterații, **time** este timpul CPU în secunde şi **vfo** este valoarea optimă a funcției obiectiv.

Tabelul 1.1

$n_v = 100,\ n = 201$ (variabile), $m = 5050$ (restricții)

	#iter	**time**	**vfo**
CONOPT	442	20.738	0.78481114
KNITRO	49	21.490	0.71973268
MINOS	48	13.238	0.6749814

Tabelul 1.2.

$n_v = 200,\ \ n = 401$ (variabile), $m = 20100$ (restricții)

	#iter	**time**	**vfo**
CONOPT	1066	180.730	0.78515482
KNITRO	39	264.530	0.72685233
MINOS	217	236.164	0.7322066

Tabelul 1.3.

$n_v = 300,\ \ n = 601$ (variabile), $m = 45150$ (restricții)

	#iter	**time**	**vfo**
CONOPT	2319	762.438	0.78529080
KNITRO	49	3476.609	0.72686364
MINOS	282	763.039	0.68840451

Figura 1.2 prezintă poligonul de arie maximă cu $n_v = 10$, respectiv $n_v = 20$, furnizat SNOPT.

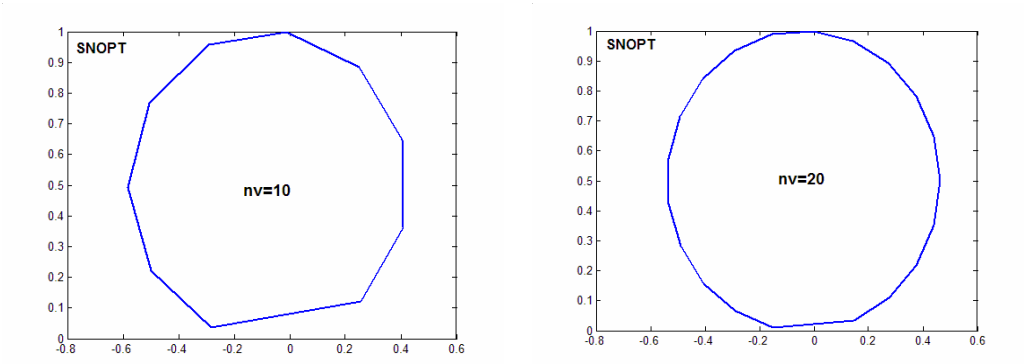

Fig.1.2. Poligonul de arie maximă cu $n_v = 10$ şi respectiv $n_v = 20$.

Aplicația G2 (*Electron*)

Distribuția electronilor pe o sferă conductoare. Dați n_p electroni, să se determine starea de echilibru a acestora pe o sferă conductoare.

Problema, cunoscută ca problema Thomson, constă în a determina configurația de energie minimă a n_p puncte încărcate electric plasate pe o sferă conductoare. Aceasta este o problemă importantă în fizică şi chimie referitoare la determinarea structurii de potențial Coulomb minim corespunzătoare pozițiilor unor atomi. Dacă (x_i, y_i, z_i) sunt pozițiile celor n_p puncte (electroni), atunci energia potențială a acestora este:

$$\sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} \left((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \right)^{-\frac{1}{2}}$$

care trebuie minimizată în virtutea restricțiilor:

$$x_i^2 + y_i^2 + z_i^2 = 1, \quad i = 1, \ldots, n_p.$$

Problema este descrisă şi în [Andrei, 2001, pagina 292]. Aceasta are o multitudine de minime locale în care valoarea funcției obiectiv este apropiată de minimul global. Mai mult, numărul de minime locale creşte exponențial cu n_p. Ca atare, determinarea minimului global computațional este o problemă dificilă. Optimizatoarele profesionale determină numai punctele de minim local.
Expresia GAMS a acestei probleme este prezentată în figura 2.1.

```
$ontext
Electron
Given n electrons, find the equilibrium state distribution (of
minimal Coulomb potential) of the electrons positioned on a
conducting sphere.
$offtext

$if     set n  $set np %n%
$if not set np $set np 50

Set i       electrons /i1 * i%np%/
    ut(i,i) upper triangular part;

Alias (i,j);
    ut(i,j)$(ord(j) > ord(i)) = yes;

Variables   x(i) x-coordinate of the electron
            y(i) y-coordinate of the electron
            z(i) z-coordinate of the electron
            potential Coulomb potential;

Equations   obj     objective
            ball(i) points on unit ball;

obj.. potential =e=
  sum{ut(i,j), 1.0/sqrt(sqr(x[i]-x[j]) + sqr(y[i]-y[j]) +
                        sqr(z[i]-z[j]))};

ball(i).. sqr(x(i)) + sqr(y(i)) + sqr(z(i)) =e= 1;

* Set the starting point to a quasi-uniform distribution
* of electrons on a unit sphere

scalar pi a famous constant;
pi = 2*arctan(inf);

parameter theta(i), phi(i);
theta(i) = 2*pi*uniform(0,1);
phi(i)   = pi*uniform(0,1);

x.l(i) = cos(theta(i))*sin(phi(i));
y.l(i) = sin(theta(i))*sin(phi(i));
z.l(i) =               cos(phi(i));
```

```
model electron /all/;

electron.iterlim = 500000;
option reslim=5000;

$onecho >bench.opt
solvers conopt knitro minos
$offecho
electron.optfile=1;
electron.workfactor=20;
option nlp=bench;

solve electron using nlp minimizing potential;
* End electron
```

Fig. 2.1. Expresia GAMS a aplicației G2 (Electron).

Pentru diverse valori ale lui n_p performanțele algoritmilor CONOPT, KNITRO şi MINOS sunt descrise în tabelele de mai jos.

Tabelul 2.1

$n_p = 50,\ n = 151$ (variabile), $m = 51$ (restricții)

	#iter	**time**	**vfo**
CONOPT	37	0.602	1055.1823147
KNITRO	234	19.758	1055.1823147
MINOS	54	6.891	1055.1823147

Tabelul 2.2

$n_p = 100,\ n = 301$ (variabile), $m = 101$ (restricții)

	#iter	**time**	**vfo**
CONOPT	59	3.404	4448.4104206
KNITRO	1275	1433.160	4448.3506343
MINOS	Prea multe iterații		

Aplicația G3 (*Chain*)

Să se determine forma unui lanț de densitate uniformă, de lungime L, suspendat între două puncte fixe, cu energie potențială minimă.

Această problemă clasică, cunoscută încă ca şi problema câinelui, a fost sugerată de Hans Mittelman. Problema este descrisă în [Dolan, Moré şi Munson, 2004, pagina 9] (vezi de asemenea [Cesari, 1983]).

Problema constă în a determina funcția $x(t)$, forma lanțului (înălțimea la care se află lanțul), care minimizează energia potențială:

$$\int_0^1 x(t)\left(1 + x'(t)^2\right)^{1/2} \mathrm{d}t,$$

referitor la restricția impusă de lungimea lanțului:

$$\int_0^1 \left(1 + x'(t)^2\right)^{1/2} \mathrm{d}t = L,$$

precum şi condiţiile la capetele lanţului $x(0) = a$ şi $x(1) = b$, unde a şi b sunt înălţimile la care este suspendat lanţul la cele două capete.

O altă formulare a problemei se poate face introducând variabila de control $u(t) = x_1'(t)$ şi funcţia care reprezintă energia potenţială

$$x_2(t) = \int_0^t x_1(s)\left(1 + u(s)^2\right)^{1/2} \mathrm{d}s.$$

Această formulare ne conduce la minimizarea energiei potenţiale (totale) $x_2(1)$ referitor la ecuaţiile

$$x_1'(t) = u,$$
$$x_2'(t) = x_1(t)\left(1 + u(t)^2\right)^{1/2},$$
$$x_3'(t) = \left(1 + u(t)^2\right)^{1/2}.$$

Introducând un număr n de puncte de discretizare care definesc lungimea pasului de discretizare $h = 1/(n+1)$, atunci, varianta discretă a problemei este

$$\min h \sum_{i=1}^{n+1} \left(\frac{x_i + x_{i-1}}{2}\right) \sqrt{1 + \left(\frac{x_i - x_{i-1}}{h}\right)^2}$$

referitor la

$$\sum_{i=1}^{n+1} \sqrt{1 + \left(\frac{x_i - x_{i-1}}{h}\right)^2} = \frac{L}{h},$$

unde $x_0 = a$ şi $x_{n+1} = b$ [Bondarenko, Bortz, Moré, 1999].

În reprezentarea GAMS a problemei se utilizează varianta integrării sistemului de ecuaţii diferenţiale de mai sus. Varianta discretă de mai sus a problemei, descrisă în [Bondarenko, Bortz, Moré, 1999], a fost utilizată în pachetul SPENBAR [Andrei, 2001]. În experimentele numerice vom considera $a = 1$, $b = 3$ şi $L = 4$.

Expresia GAMS a problemei este prezentată în figura 3.1

```
$ontext
Find the chain (of uniform density) of length L suspended
between two points with minimal potential energy.
$offtext

$if     set n  $set nh %n%
$if not set nh $set nh 1000

set nh /i0 * i%nh%/;

alias(nh,i);
scalars L  length of the suspended chain      / 4 /
```

```
        a  height of the chain at t=0 (left)  / 1 /
        b  height of the chain at t=1 (left)  / 3 /
        tf ODEs defined in [0 tf]             / 1 /
        h  uniform interval length
        n  number of subintervals
        tmin;

if (b>a, tmin = 0.25 else tmin = 0.75);
n = card(nh) - 1;
h = tf/n;

variables
  x(i)   height of the chain
  u(i)   derivative of x
  energy potential energy ;

x.fx('i0')    = a;
x.fx('i%nh%') = b;
x.l(i) = 4*abs(b-a)*((ord(i)-1)/n)*(0.5*((ord(i)-1)/n)
                      - tmin) + a;
u.l(i) = 4*abs(b-a)*(((ord(i)-1)/n) - tmin);

* Equations
equations obj, x_eqn(i), length_eqn ;

obj.. energy =e=
       0.5*h*sum(nh(i+1), x(i)*sqrt(1+sqr(u(i))) +
                          x(i+1)*sqrt(1+sqr(u(i+1))));
x_eqn(i+1).. x(i+1) =e= x(i) + 0.5*h*(u(i)+u(i+1));
length_eqn.. 0.5*h*sum(nh(i+1), sqrt(1+sqr(u(i))) +
                                sqrt(1+sqr(u(i+1)))) =e= L;

model chain /all/;

chain.optfile=1;
chain.workspace=120;
option nlp=conopt
solve chain using nlp minimizing energy;

*file res /chain.dat/;
*put res
*loop(i, put x.l(i):10:5, put/)
* End Hanging Chain
```

Fig. 3.1. Expresia GAMS a aplicației G3 (Hanging Chain).

Pentru diverse valori ale lui n_h performanțele algoritmilor CONOPT, KNITRO, MINOS şi SPENBAR sunt descrise în tabelele de mai jos.

Tabelul 3.1

$n_h = 200$, $n = 403$ (variabile), $m = 202$ (restricții)

	#iter	time	vfo
CONOPT	17	0.258	5.06891734
KNITRO	8	0.230	5.06891707
MINOS	36	1.266	5.068917

Tabelul 3.2

$n_h = 500,\ \ n = 1003$ (variabile), $m = 502$ (restricții)

	#iter	time	vfo
CONOPT	21	0.711	5.06857779
KNITRO	8	0.190	5.06857764
MINOS	96	10.586	5.068578
SPENBAR	3 / 20072†	117.03	5.068480

† 3 iterații majore,
20072 iterații minore în metoda Newton trunchiată.

Tabelul 3.3

$n_h = 1000,\ \ n = 2003$ (variabile), $m = 1002$ (restricții)

	#iter	time	Vfo
CONOPT	20	2.654	5.0685101
KNITRO	8	0.380	5.0685099
MINOS	202	73.145	5.068510

Tabelul 3.4

$n_h = 1200,\ \ n = 2403$ (variabile), $m = 1202$ (restricții)

	#iter	time	Vfo
CONOPT	26	4.008	5.0685019
KNITRO	8	0.490	5.0685018
MINOS	253	130.316	5.068502

Pentru $n_h = 400$ forma lanțului este arătată în figura 3.2.

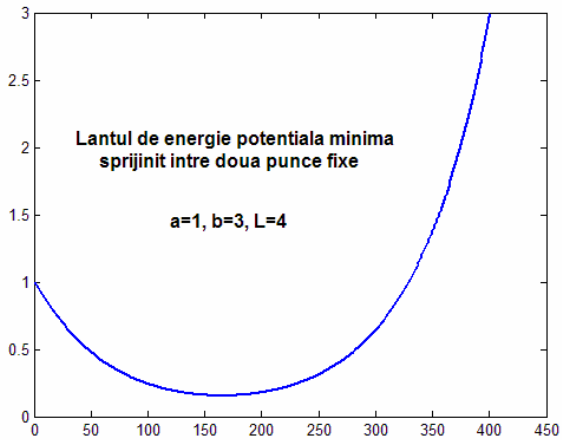

Fig. 3.2. Forma lanțului de energie potențială minimă sprijinit între două puncte fixe.

Aplicaţia G4 (*Rocket*) *Date masa iniţială, masa combustibilului şi caracteristicile rezistenţei aerului să se determine altitudinea maximă a unei rachete lansată vertical utilizând forţa de tracţiune ca variabilă de control.*
Astfel formulată, problema este una de control optimal. Detalii se găsesc în [Dolan, Moré şi Munson, 2004, pagina 23] (vezi de asemenea [Bryson, 1999, pp. 392-394]).
Ecuaţiile de mişcare ale rachetei sunt următoarele;

$$h' = v, \qquad v' = \frac{r - D(h,v)}{m} - g(h), \qquad m' = -\frac{r}{c},$$

unde h este altitudinea de la centrul Terrei, v viteza pe verticala locului, r forţa de tracţiune, D coeficientul de rezistenţă aerodinamică, g forţa gravitaţională şi c o constantă care măsoară impulsul combustibilului asupra rachetei.
Forţa de tracţiune este mărginită ca:

$$0 \le r(t) \le r_{\max}.$$

Coeficientul de rezistenţa aerodinamică şi forţa gravitaţională sunt definite sub forma:

$$D(h,v) = D_c v^2 \exp\left(-h_c \frac{h - h(0)}{h(0)}\right), \qquad g(h) = g_0 \left(\frac{h(0)}{h}\right)^2,$$

unde D_c şi h_c sunt constante, iar g_0 este forţa gravitaţională la suprafaţa Terrei.
Iniţial racheta se află în repaus, adică $v(0) = 0$, iar masa la sfârşitul zborului este o fracţie din masa iniţială, adică

$$m(t_f) = m_c m(0),$$

unde t_f este timpul final de zbor şi m_c o constantă. Pe lângă acestea, masa, altitudinea şi viteza sunt mărginite sub forma:

$$m(t_f) \le m(t) \le m(0), \qquad h(t) \ge h(0), \qquad v(t) \ge 0.$$

Prin scalare ecuaţiile de mişcare ale rachetei se pot face adimensionale. În model parametrii $r_{\max}$, D_c şi c se aleg în funcţie de mărimile $h(0)$, $m(0)$ şi g_0. Ca în [Bryson, 1999, pp. 392-394] definim:

$$r_{\max} = 3.5 g_0 m(0), \qquad D_c = \frac{1}{2} v_c \frac{m(0)}{g_0}, \qquad c = \frac{1}{2}\left(g_0 h(0)\right)^{1/2}.$$

Fără pierderea generalităţii putem presupune că $h(0) = m(0) = g_0 = 1$ şi considerăm următoarele valori pentru parametrii h_c, m_c şi v_c:

$$h_c = 500, \qquad m_c = 0.6, \qquad v_c = 620.$$

Valorile iniţiale ale variabilelor modelului sunt următoarele: $t_f = 1$,

$$v(t) = \frac{t}{t_f}\left(1 - \frac{t}{t_f}\right), \qquad m(t) = (m_f - m_0)\left(\frac{t}{t_f}\right) + m_0,$$

evaluate în punctele de discretizare. Valoarea iniţială a tracţiunii este $r = r_{\max}/2$.

Cu acestea, utilizând o rețea de discretizare uniformă cu n_h intervale şi schema trapezoidală de aproximare, în limbajul GAMS ecuațiile de mişcare se exprimă ca în figura 4.1.

```
$ontext
Goddard Rocket.
Maximize the final altitude of a vertically launched rocket,
using the  thrust as a control and given the initial mass, the
fuel mass, and the drag characteristics of the rocket.
$offtext

$if     set n  $set nh %n%
$if not set nh $set nh 1000

set h intervals / h0 * h%nh%/

scalars
   h_0  Initial height          / 1 /
   v_0  Initial velocity        / 0 /
   m_0  Initial mass            / 1 /
   g_0  Gravity at the surface  / 1 /
   nh   Number of intervals in mesh / %nh% /
   r_c  Thrust constant         /3.5/
   v_c  / 620 /
   h_c  / 500 /
   m_c  / 0.6 /
   D_c
   m_f  final mass
   c ;

* Constants:
c   = 0.5*sqrt(g_0*h_0);
m_f = m_c*m_0;
D_c = 0.5*v_c*(m_0/g_0);

variable   final_velocity

positive
variables  step   step size
           v(h)   velocity
           ht(h)  height
           g(h)   gravity
           m(h)   mass
           r(h)   thrust   (tractiunea)
           d(h)   drag     (rezistenta aerului) ;

* Bounds:
ht.lo(h) = h_0;

r.lo(h)  = 0.0;
r.up(h)  = r_c*(m_0*g_0);

m.lo(h)  = m_f;
m.up(h)  = m_0;
```

```
* Initial values:
ht.l(h) = 1;
v.l(h)  = ((ord(h)-1)/nh)*(1 - ((ord(h)-1)/nh));
m.l(h)  = (m_f - m_0)*((ord(h)-1)/nh) + m_0;
r.l(h)  = r.up(h)/2;
step.l  = 1/nh;

d.l(h) = D_c*sqr(v.l(h))*exp(-h_c*(ht.l(h)-h_0)/h_0);
g.l(h) = g_0*sqr(h_0/ht.l(h));

* Fixed values for variables:
ht.fx('h0')   = h_0;
v.fx('h0')    = v_0;
m.fx('h0')    = m_0;
m.fx('h%nh%') = m_f;

equations  df(h)  Drag function
           gf(h)  Gravity function
           obj
           h_eqn(h), v_eqn(h), m_eqn(h);

obj.. final_velocity =e= ht('h%nh%');

df(h).. d(h) =e= D_c*sqr(v(h))*exp(-h_c*(ht(h)-h_0)/h_0);

gf(h).. g(h) =e= g_0*sqr(h_0/ht(h));

h_eqn(h-1).. ht(h) =e= ht(h-1) + .5*step*(v(h) + v(h-1));

m_eqn(h-1).. m(h) =e= m(h-1) - .5*step*(r(h) + r(h-1))/c;

v_eqn(h-1).. v(h) =e= v(h-1)
             + .5*step*((r(h)   - D(h)   - m(h)  *g(h))/m(h)
             +(r(h-1) - D(h-1) - m(h-1)*g(h-1))/m(h-1));

model rocket /all/;

$onecho >bench.opt
  solvers conopt knitro minos
$offecho

rocket.optfile=1;
rocket.iterlim=50000;
option nlp=bench;

solve rocket using nlp maximizing final_velocity;
*file res1 /g4.dat/;
*put res1
*loop(h, put v.l(h):10:7, put/)
* End Rocket
```

Fig. 4.1. Expresia GAMS a aplicației G4 (Goddard Rocket).

Pentru diverse valori ale lui n_h performanțele algoritmilor CONOPT, KNITRO şi MINOS se prezintă în tabelele de mai jos.

Tabelul 4.1

$n_h = 500$, $n = 3008$ (variabile), $m = 2503$ (restricţii)

	#iter	time	vfo
CONOPT	1997	22.533	1.012836662
KNITRO	70	5.938	1.012629770
MINOS	43	11.197	1.012603

Tabelul 4.2

$n_h = 1000$, $n = 6008$ (variabile), $m = 5003$ (restricţii)

	#iter	time	vfo
CONOPT	2700	49.801	1.012835932
KNITRO	71	19.327	1.012397382
MINOS	35	38.846	1.012243

Tabelul 4.3

$n_h = 1200$, $n = 7208$ (variabile), $m = 6003$ (restricţii)

	#iter	time	vfo
CONOPT	3175	63.117	1.01283609
KNITRO	78	29.282	1.01230050
MINOS	29	40.195	1.011832

În figurile de mai jos se arată evoluţia variabilelor rachetei. Figura 4.2 arată altitudinea şi masa rachetei ca funcţii de timp. Vedem că altitudinea creşte până la valoarea maximă $h = 1.01$. În acelaşi timp masa rachetei descreşte aproximativ liniar până la valoarea finală $m(t_f) = 0.6$ care este atinsă pentru $h = 186$ adică $t = 0.372$.

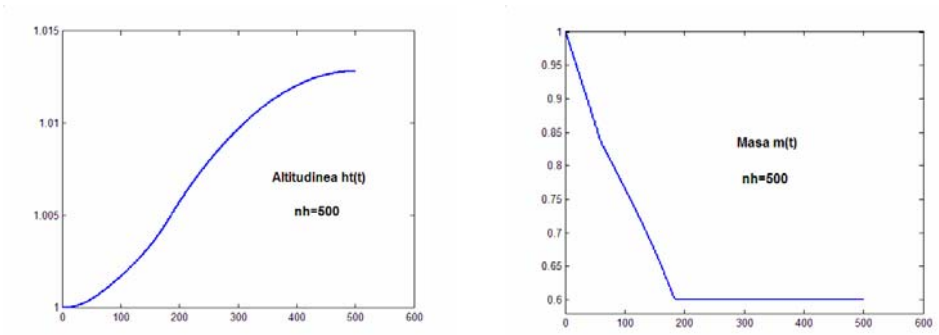

Fig. 4.2. Evoluţia altitudinii $ht(t)$ şi a masei $m(t)$.

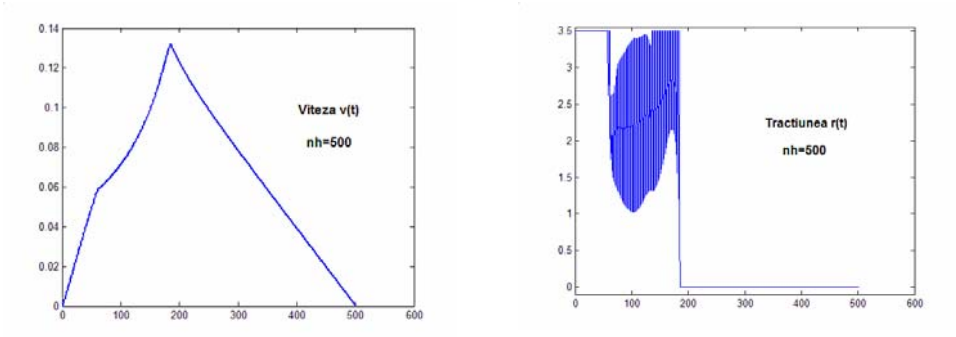

Fig. 4.3. Evoluția $v(t)$ vitezei şi a tracțiunii $r(t)$.

În figura 4.3 vedem evoluția vitezei $v(t)$ şi a tracțiunii $r(t)$ pe intervalul $[0,t_f]$, unde $t_f = 1$. Pentru $h = 185$ viteza atinge valoarea maximă $v_{\max} = 0.13218$. Pe de altă parte pe subintervalul $[59,185]$ tracțiunea are un caracter de bang-bang singular, o comportare tipică pentru sistemele de control optimal cu variabile mărginite. Este intetresant de observat cum pe subintervalul [59,185], adică pe perioada bang-bang-ului, tracțiunea are un caracter haotic, valoarea maximă a acesteia fiind limitată la 3.5 aşa cum s-a stabilit în model. De fapt, acesta este intervalul critic în lansarea oricărei rachete. Pe de altă parte, vedem cum tracțiunea are o valoare maximă de 3.5 pe durata $[0,58]$ ceea ce permite ridicarea rachetei. La momentul $h = 185$ viteza are valoarea maximă, de aici încolo tracțiunea se anulează (vezi fig. 4.3).

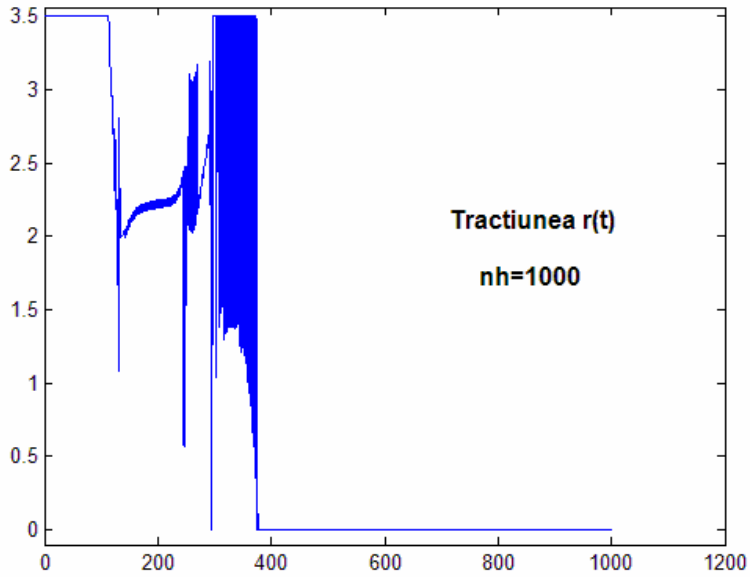

Fig. 4.4. Evoluția tracțiunii $r(t)$ pentru $nh = 1000$.

Este foarte instructiv să vedem comportarea tracțiunii pentru o discretizare mai fină. În figura 4.4 se arată evoluția tracțiunii pentru $nh = 1000$. Vedem imediat cum pe intervalul critic, în care apare bang-bang-ul, se detaliază evoluția tracțiunii. Caracterul haotic se menține, dar pentru un interval de timp destul de mic, către finalul intervalului de bang-bang, tracțiunea atinge valoarea maximă. Aceasta asigură obținerea vitezei maxime a rachetei. După aceasta tracțiunea se anulează până la sfârşitul intervalului de evoluție a rachetei.
Din figura 4.4. vedem că tracțiunea are aceeaşi comportare. Pentru $h \in [0,113]$ tracțiunea are valoarea maximă de 3.5 care permite ridicarea rachetei. Pentru $h \in [114,377]$ tracțiunea are o comportare de bang-bang singular în care se vede că aceasta de multe ori atinge valoarea maximă admisă. De la momentul 378 încolo tracțiunea este nulă. Şi în acest caz vedem corelația care există între viteză şi tracțiune, pentru $h = 375$ viteza atinge valoarea maximă $v_{\max} = 0.1303938$ după care începe să scadă. De aici încolo tracțiunea este nulă.

Aplicația G5 (*Camshape*)
În această aplicație este vorba de optimizarea formei unei came. Problema a fost modelată de Anițescu şi Şerban [1998] şi constă în a *maximiza aria unei valve prin rotația unei came convexe de curbură şi rază date.*
Problema este descrisă în [Dolan, Moré şi Munson, 2004, pagina 7]. Presupunem că forma camei este circulară pentru un unghi de valoare $6\pi/5$ din circumferință, de rază $r_{\min}$. Variabilele de proiectare sunt r_i, $i = 1,\ldots,n$, care reprezintă raza camei la unghiuri egal distribuite de-a lungul unghiului $2\pi/5$. Problema constă în a maximiza aria valvei, adică a maximiza funcția

$$f(r) = \pi r_v^2 \left(\frac{1}{n} \sum_{i=1}^{n} r_i \right)$$

referitor la următoarele restricții asupra razelor r_i, $i = 1,\ldots,n$. În funcția de mai sus r_v este un parametru al problemei care depinde de geometria valvei, considerat de valoare $r_v = 1$. Asupra razelor se impun restricțiile

$$r_{\min} \le r_i \le r_{\max}.$$

Cerința de convexitate a camei este exprimată sub forma

$$\text{aria}(r_{i-1}, r_{i+1}) \le \text{aria}(r_{i-1}, r_i) + \text{aria}(r_i, r_{i+1}),$$

unde $\text{aria}(r_i, r_j)$ este aria triunghiului determinat de origine şi punctele r_i şi r_j de pe suprafața camei. Această relație de convexitate se poate încă exprima sub forma

$$2r_{i-1}r_{i+1}\cos(\theta) \le r_i(r_{i-1} + r_{i+1}), \quad i = 0,\ldots,n+1,$$

unde $r_{-1} = r_0 = r_{\min}$, $r_{n+1} = r_{\max}$, $r_{n+2} = r_n$ şi $\theta = 2\pi/5(n+1)$. Cerința de curbură se exprimă sub forma

$$-\alpha \le \left(\frac{r_{i+1} - r_i}{\theta} \right) \le \alpha, \quad i = 0,\ldots,n.$$

În această aplicație considerăm $r_{\min} = 1$, $r_{\max} = 2$ şi $\alpha = 1.5$ în restricția de curbură. Deoarece forma camei este simetrică, vom considera numai o jumătate din unghiul de proiectare.
Expresia GAMS a acestei probleme este arătată în figura 5.1.

```
$ontext
Maximize the area of the valve opening for one rotation of a
convex cam with constraints on the curvature and on the radius
of the cam.
$offtext

$if not set n $set n 800

Set i discretization points /i1 * i%n%/;

Alias (i,j);

Scalar
    R_v     design parameter related to the valve shape /1/
    R_max   maximum allowed radius of the cam           /2/
    R_min   minimum allowed radius of the cam           /1/
    pi      a famous constant
    alpha   curvature limit parameter                   /1.5/
    d_theta angle between discretization points;

pi = 2*arctan(inf);
d_theta = 2*pi/(5*(%n%+1));

set first(i), last(i), middle(i);
first('i1')  = yes;
last('i%n%') = yes;
middle(i) = yes; middle(first) = no; middle(last) = no;

Variables  r(i)      radius of the cam at discretization points
           rdiff(i)  intermediate
           area      valve area;

* Bounds
r.lo(i) = R_min;
r.up(i) = R_max;

rdiff.lo(i(j+1)) = -alpha*d_theta;
rdiff.up(i(j+1)) =  alpha*d_theta;

r.lo('i1') = max(-alpha*d_theta + R_min, r.lo('i1'));
r.up('i1') = min( alpha*d_theta + R_min, r.up('i1'));

r.lo('i%n%') =  max(R_max - alpha*d_theta, r.lo('i%n%'));
r.up('i%n%') =  min(R_max + alpha*d_theta, r.up('i%n%'));

r.up('i1') = min( R_min/(2*cos(d_theta)-1), r.up('i1'));

* Initial values
r.l(i) = (R_min+R_max)/2;
```

```
Equations  obj              objective
           convexity(i)
           convex_edge1(i)
           convex_edge3(i)
           convex_edge4(i)
           eqrdiff(i);

obj.. area =e= ((pi*R_v)/%n%) * sum(i, r(i));

convexity(middle(i)).. -r(i-1)*r(i) - r(i)*r(i+1) +
                       2*r(i-1)*r(i+1)*cos(d_theta) =l= 0;

convex_edge1(first(i)).. -R_min*r(i) - r(i)*r(i+1) +
                       2*R_min*r(i+1)*cos(d_theta) =l= 0;

convex_edge3(last(i)).. -r(i-1)*r(i) - r(i)*R_max +
                       2*r(i-1)*R_max*cos(d_theta) =l= 0;

convex_edge4(last(i)).. -2*R_max*r(i) +
                       2*sqr(r(i))*cos(d_theta) =l= 0;

eqrdiff(j(i+1)).. rdiff(i) =e= r(i+1) - r(i);

model camshape /all/;

*$onecho >bench.opt
*  solvers conopt
*$offecho

camshape.optfile=1;
camshape.iterlim=50000;
camshape.workspace=200;

option nlp=minos;

solve camshape using nlp maximizing area;

*file rez /camshape.dat/;
*put rez
*loop(i, put r.l(i):10:5, put/)
* End camshape
```

Fig. 5.1. Expresia GAMS a aplicației G5.

În tabelele de mai jos se prezintă performanțele algoritmilor CONOPT şi KNITRO. MINOS nu poate rezolva problema.

Tabelul 5.1

$n = 800$, $n = 1600$ (variabile), $m = 1601$ (restricții)

	#iter	time	vfo
CONOPT	44	0.750	4.27427414
KNITRO	47	0.911	4.27401472

Tabelul 5.2

$n = 1000$, $n = 2000$ (variabile), $m = 2001$ (restricții)

	#iter	**time**	**vfo**
CONOPT	66	1.251	4.27399125
KNITRO	49	1.211	4.27366794

Tabelul 5.3

$n = 10000$, $n = 20000$ (variabile), $m = 20001$ (restricții)

	#iter	**time**	**vfo**
CONOPT	46	22.673	4.27265943
KNITRO	54	32.136	4.26883424

În figura 5.2 se arată forma camei în domeniul de unghi $2\pi/5$, pentru $\alpha = 1.5$.

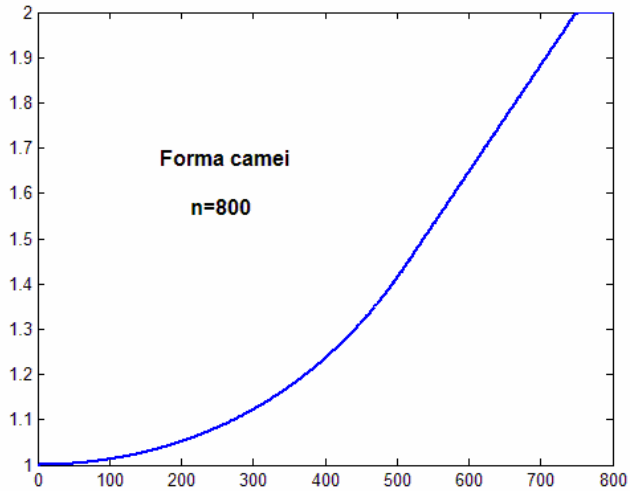

Fig. 5.2. Forma camei pentru $\alpha = 1.5$.

Aplicația G6 (*Robot*) *Minimizarea timpului de deplasare al unui robot între două puncte fixe.*

Problema a fost formulată de Mössner-Beigel [1995] în teza sa de doctorat. Este vorba despre un braț robotizat format dintr-o bară rigidă de lungime L care poate parcurge o distanță ρ de la origine. Dacă punctul de articulație al brațului este originea unui sistem sferic de coordonate, atunci problema se poate exprima în termenii distanței ρ, a unghiului din planul orizontal θ, a unghiului din planul vertical φ, a mărimilor de control $(u_\rho, u_\theta, u_\varphi)$ asociate acestor elemente şi a timpului final t_f.

Asupra acestor variabile se impun condițiile:

$$\rho(t) \in [0, L], \quad |\theta(t)| \le \pi, \quad 0 \le \varphi(t) \le \pi,$$
$$|u_\rho| \le 1, \quad |u_\theta| \le 1, \quad |u_\varphi| \le 1.$$

Ecuațiile de mişcare ale robotului sunt:

$$L\rho'' = u_\rho, \quad I_\theta \theta'' = u_\theta, \quad I_\varphi \varphi'' = u_\varphi,$$

unde I este momentul de inerție, definit ca:

$$I_\theta = \frac{((L-\rho)^3 + \rho^3)}{3} \sin(\varphi)^2, \qquad I_\varphi = \frac{(L-\rho)^3 + \rho^3}{3}.$$

Pentru acest sistem de ecuații diferențiale se impun condițiile la limită:

$$\rho(0) = \rho(t_f) = 4.5, \qquad \theta(0) = 0, \ \theta(t_f) = \frac{2\pi}{3}, \qquad \varphi(0) = \varphi(t_f) = \frac{\pi}{4},$$
$$\rho'(0) = \theta'(0) = \varphi'(0) = \rho'(t_f) = \theta'(t_f) = \varphi'(t_f) = 0.$$

Valorile inițiale ale variabilelor ρ şi φ sunt $\rho = 4.5$, respectiv $\varphi = \pi/4$ pentru toate punctele de discretizare. Similar, valorile inițiale ale variabilei θ sunt date de valorile funcției

$$\theta(t) = \frac{2\pi}{3}\left(\frac{t}{t_f}\right)^2$$

calculate în punctele de discretizare. Pentru t_f ca valoare inițială se consideră $t_f = 1$. Valorile inițiale ale tuturor mărimilor de comandă sunt nule.

Deoarece sistemul de coordonate sferice nu este un sistem inerțial, în această formulare, modelul matematic al robotului nu conține forțele Coriolis şi centrifugale.

După cum vedem ecuațiile de mişcare ale robotului sunt un sisteme de ecuații diferențiale de ordinul doi. Introducând variabilele suplimentare ρ', θ' şi φ', atunci acestea se exprimă ca un sistem de ecuații diferențiale de ordinul unu. Discretizând intervalul $[0, t_f]$ uniform în n_h intervale şi utilizând schema de aproximare trapezoidală, expresia GAMS a modelului de minimizare a timpului de deplasare este ca în figura 6.1.

```
$ontext
Minimize the time taken for a robot arm to travel between two
points.
$offtext

$if     set n  $set nh %n%
$if not set nh $set nh 200

sets  h intervals / h0 * h%nh%/

scalars
    pi a famous constant
    nh number of intervals / %nh% /
    L  total length of arm / 5 /
```

```
    max_u_rho  / 1 /
    max_u_the  / 1 /
    max_u_phi  / 1 /;

pi = 2*arctan(inf);

variables
    rho(h)        distance from the origin
    the(h)        horizontal angle
    phi(h)        vertical angle
    rho_dot(h)
    the_dot(h)
    phi_dot(h)
    u_rho(h)      control
    u_the(h)
    u_phi(h)
    step
    tf            final time
    i_the(h)      moment of inertia
    i_phi(h) ;

* Bounds
rho.lo(h) = 0;    rho.up(h) = L;
the.lo(h) = -pi;  the.up(h) = pi;
phi.lo(h) = 0;    phi.up(h) = pi;

u_rho.lo(h) = -max_u_rho;  u_rho.up(h) = max_u_rho;
u_the.lo(h) = -max_u_the;  u_the.up(h) = max_u_the;
u_phi.lo(h) = -max_u_phi;  u_phi.up(h) = max_u_phi;

i_the.lo(h) = 0.0001;
i_phi.lo(h) = 0.0001;

set firstlast(h) / h0, h%nh% /;

* Fixed variables
the.fx('h0')    = 0;
the.fx('h%nh%') = 2*pi/3;

rho.fx(firstlast)      = 4.5;
phi.fx(firstlast)      = pi/4;
rho_dot.fx(firstlast)  = 0;
the_dot.fx(firstlast)  = 0;
phi_dot.fx(firstlast)  = 0;
i_phi.fx(firstlast(h)) = (power(L-rho.l(h),3)+
                          power(rho.l(h),3))/3.0;
i_the.fx(firstlast(h)) = i_phi.l(h)*sqr(sin(phi.l(h)));

*Initialization
rho.l(h) = 4.5;
the.l(h) = (2*pi/3)*sqr(ord(h)/nh);
phi.l(h) = pi/4;

rho_dot.l(h) = 0.0;
the_dot.l(h) = (4*pi/3)*(ord(h)/nh);
phi_dot.l(h) = 0.0;
```

```
step.l = 1/nh;

i_phi.l(h) = (power(L-rho.l(h),3)+power(rho.l(h),3))/3.0;
i_the.l(h) = i_phi.l(h)*sqr(sin(phi.l(h)));

equations
     tf_eqn
     rho_eqn(h)
     the_eqn(h)
     phi_eqn(h)
     u_rho_eqn(h)
     u_the_eqn(h)
     u_phi_eqn(h)
     i_the_eqn(h)
     i_phi_eqn(h);

tf_eqn.. tf =e= step*nh;

i_phi_eqn(h).. i_phi(h) =e= (power(L-rho(h),3)+
                              power(rho(h),3))/3.0;

i_the_eqn(h).. i_the(h) =e= i_phi(h)*sqr(sin(phi(h)));

rho_eqn(h-1)..  rho(h) =e= rho(h-1) + 0.5*step*(rho_dot(h) +
                           rho_dot(h-1));

the_eqn(h-1)..  the(h) =e= the(h-1) + 0.5*step*(the_dot(h) +
                           the_dot(h-1));
phi_eqn(h-1)..  phi(h) =e= phi(h-1) + 0.5*step*(phi_dot(h) +
                           phi_dot(h-1));

u_rho_eqn(h-1)..  rho_dot(h) =e=
                    rho_dot(h-1) + 0.5*step*(u_rho(h) +
                    u_rho(h-1))/L;

u_the_eqn(h-1)..  the_dot(h) =e=
          the_dot(h-1) + 0.5*step*(u_the(h)/i_the(h) +
          u_the(h-1)/i_the(h-1));

u_phi_eqn(h-1)..  phi_dot(h) =e=
          phi_dot(h-1) + 0.5*step*(u_phi(h)/i_phi(h) +
          u_phi(h-1)/i_phi(h-1));

model robot /all/;

$onecho >bench.opt
  solvers conopt
$offecho

robot.optfile=1;
robot.iterlim=50000;
robot.workspace=120;

option nlp=bench
```

```
solve robot miniziming tf using nlp;

*file res /robot.dat/;
*put res
*loop(h, put rho.l(h):10:5, put/)
*End Robot
```

Fig. 6.1. Expresia GAMS a aplicației G6.

În tabelul 6.1 se prezintă performanțele algoritmului CONOPT pentru diferite valori ale lui n_h, unde n este numărul de variabile al problemei, m numărul de restricții, iar **time** este exprimat în secunde. Din acest tabel se vede că, aparent pentru orice discretizare, valoarea optimă a funcției obiectiv este aceeaşi.

Tabelul 6.1.
Performanțele algoritmului CONOPT pentru rezolvarea aplicației G6.

n_h	n	m	**#iter**	**time**	**vfo**
100	1113	803	200	0.871	9.1426853843
200	2213	1603	279	1.863	9.1413954984
400	4413	3203	429	7.379	9.1410260259
800	8813	6403	786	18.598	9.1409409706
1000	11013	8003	953	27.520	9.1409312779
1200	13213	9603	996	36.152	9.1409253934
1400	15413	11203	1302	53.035	9.1409214713
2000	22013	16003	1820	110.168	9.1409165109
2100	23113	16803	1640	115.055	9.1409158906
2200	24213	17603	1713	123.238	9.1409158061
2400	26413	19203	2160	154.484	9.1409151373
2600	28613	20803	2336	188.238	9.1409145137
2800	30813	22403	2506	213.398	9.1409139367
3000	33013	24003	2274	225.953	9.1409137707

Problemă greu de rezolvate pentru pachetele KNITRO şi MINOS. KNITRO cere un număr de iterații prea mare. MINOS are dificultăți în evaluarea funcțiilor problemei. În figurile de mai jos se prezintă evoluția variabilelor asociate robotului.

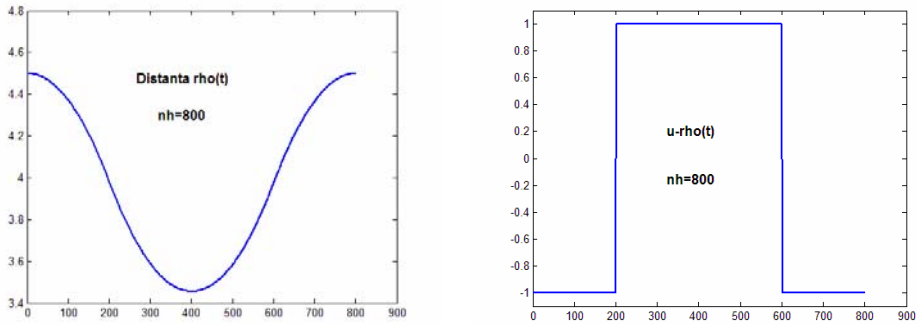

Fig. 6.2. Evoluția variabilelor $\rho(t)$ şi $u_\rho(t)$.

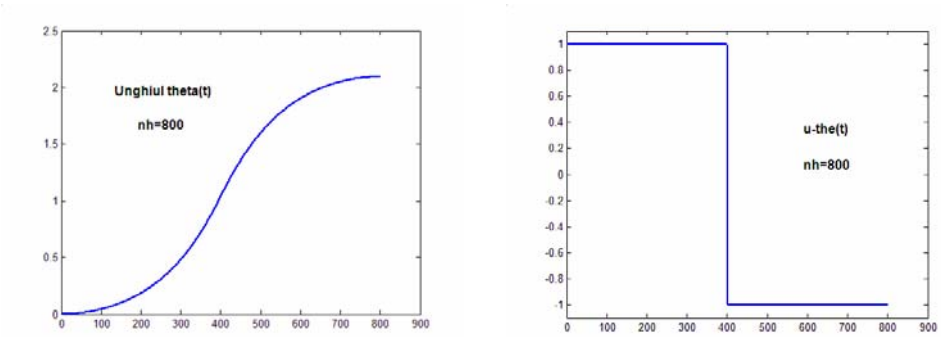

Fig. 6.3. Evoluția variabilelor $\theta(t)$ şi $u_\theta(t)$.

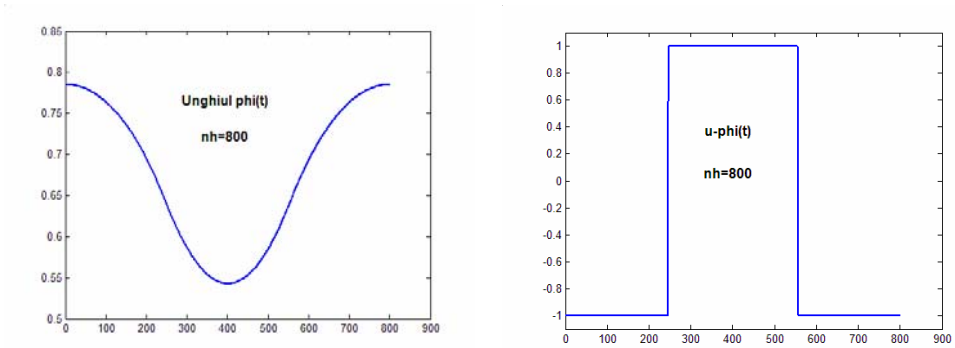

Fig. 6.4. Evoluția variabilelor $\varphi(t)$ şi $u_\varphi(t)$.

Problema este de a minimiza timpul de deplasare ale brațului robotului între două puncte fixe. Evoluția variabilelor $\rho(t)$ (distanța), $\theta(t)$ (unghiul din planul orizontal) şi $\varphi(t)$ (unghiul din planul vertical), în raport cu condițiile inițiale, este foarte naturală. Şi în acest caz, variabilele fiind mărginite, mărimile de control asociate acestor variabile au un caracter de bang-bang.

Aplicația G7 (*Steering*) *Minimizarea timpului de deplasare a unei particule, supusă unei tracțiuni date, pentru a realiza o altitudine şi viteză finală impuse.*
Utilizând [Bryson şi Ho, 1975, pp. 59-62] şi [Dolan, Moré şi Munson, 2004, pp. 21-22], ecuațiile de mişcare sunt

$$\ddot{y}_1 = a\cos u, \qquad \ddot{y}_2 = a\sin u,$$

unde (y_1, y_2) este poziția particulei, u este unghiul de control cu

$$|u(t)| \le \frac{\pi}{4},$$

iar a este forța de tracțiune. Inițial particula este în repaus, astfel încât

$$y_1(0) = y_2(0) = \dot{y}_1(0) = \dot{y}_2(0) = 0.$$

Problema este de a minimiza timpul final de deplasare a particulei t_f astfel încât aceasta atinge o altitudine dată $y_2(t_f)$ şi o viteză finală cunoscută $(\dot{y}_1(t_f), \dot{y}_2(t_f))$. În această aplicaţie considerăm: $a = 100$ şi $y_2(t_f) = 5$, $\dot{y}_1(t_f) = 45$ şi $\dot{y}_2(t_f) = 0$. Utilizând o discretizare uniformă a timpului de deplasare şi schema trapezoidală de aproximare pe n_h intervale expresia GAMS a problemei este ca în figura 7.1.

```
$ontext
Minimize the time take for a particle, acted upon by a thrust
of constant magnitude, to achieve a given altitude and terminal
velocity.
$offtext

$if set n $set nh %n%
$if not set nh $set nh 2000

sets h intervals / h0 * h%nh% /
c coordinates /
      y1 first position coordinate
      y2 second position coordinate
      y3 first velocity coordinate
      y4 second velocity coordinate /

scalars pi a famous constant

nh number of intervals / %nh% /

a magnitude of force / 100.0 / ;

variables u(h)            control
          y(c,h)          coordinates
          tf              final time ;

positive variables step   step size ;

y.l('y2',h) = 5*(ord(h)-1)/nh;
y.l('y3',h) = 45*(ord(h)-1)/nh;
step.l = 1.0/nh;

equations tf_eqn, pos_eqn(c,h), velo1_eqn(h), velo2_eqn(h);

tf_eqn.. tf =e= step*nh;

pos_eqn(c+2,h+1).. y(c,h+1) =e= y(c,h) +
                      0.5*step*(y(c+2,h) + y(c+2,h+1));

velo1_eqn(h+1).. y('y3',h+1) =e= y('y3',h) +
                      0.5*step*(a*cos(u(h)) + a*cos(u(h+1)));

velo2_eqn(h+1).. y('y4',h+1) =e= y('y4',h) +
                      0.5*step*(a*sin(u(h)) + a*sin(u(h+1)));

pi = 2*arctan(inf);
```

```
u.lo(h) = -pi/2;
u.up(h) = pi/2;

y.fx(c,'h0') = 0;
y.fx('y2','h%nh%') = 5;
y.fx('y3','h%nh%') = 45;
y.fx('y4','h%nh%') = 0;

model steering /all/;

$onecho >bench.opt
  solvers conopt knitro
$offecho
steering.optfile=1;

option nlp=bench;

solve steering using nlp minimizing tf;

*End steering
```

Fig. 7.1. Expresia GAMS a aplicației G7.

În tabelele de mai jos se arată performanțele algoritmilor CONOPT, KNITRO şi MINOS pentru diferite valori ale lui n_h considerând că valorile inițiale ale lui $y_2(t)$ şi $y_3(t) = \dot{y}_1(t)$ sunt alese ca funcții de timpul t de forma:

$$y_1(t) = 5\left(\frac{t}{t_f}\right), \quad y_3(t) = 45\left(\frac{t}{t_f}\right).$$

Tabelul 7.1

$n_h = 400$, $n = 2007$ (variabile), $m = 1601$ (restricții)

	#iter	**time**	**vfo**
CONOPT	875	3.695	0.5545724137
KNITRO	18	0.510	0.5545724136
MINOS	57	7.551	0.554572

Tabelul 7.2

$n_h = 800$, $n = 4007$ (variabile), $m = 3201$ (restricții)

	#iter	**time**	**vfo**
CONOPT	130	13.809	0.5545712623
KNITRO	22	1.271	0.5545712622

Tabelul 7.3

$n_h = 1000$, $n = 5007$ (variabile), $m = 4001$ (restricții)

	#iter	**time**	**vfo**
CONOPT	27	3.508	0.5545712366
KNITRO	22	1.882	0.5545711240

Tabelul 7.4

$n_h = 2000$, $n = 10007$ (variabile), $m = 8001$ (restricții)

	#iter	time	vfo
CONOPT	83	25.406	0.5545709399
KNITRO	30	7.020	0.5545709433

Aplicația G8 (*Planor*) *Să se maximizeze poziția finală orizontală a unui planor suspendat supus unei forțe ascensionale termice date.*
Ecuațiile de mişcare ale planorului suspendat sunt [Bulirsch, *et al*, 1993]:

$$x'' = \frac{1}{m}\left(-L\sin\eta - D\cos\eta\right), \qquad y'' = \frac{1}{m}\left(L\cos\eta - D\sin\eta\right) - g,$$

unde (x, y) este poziția planorului, m masa acestuia, g accelerația gravitațională şi funcția η este definită de:

$$\sin\eta = \frac{w(x,y')}{v(x,x',y')}, \qquad \cos\eta = \frac{x'}{v(x,x',y')},$$

unde

$$v(x,x',y') = \sqrt{x'^2 + w(x,y')^2}, \qquad w(x,y') = y' - u(x),$$

$$u(x) = u_c\left(1 - r(x)\right)\exp\left(-r(x)\right), \qquad r(x) = \left(\frac{x}{r_c} - 2.5\right)^2,$$

în care constantele u_c şi r_c au valorile $u_c = 2.5$ şi respectiv $r_c = 100$. Vedem că forța ascensională u este pozitivă în vecinătatea lui $x = 2.5r_c$, dar se reduce la zero exponențial departe de $x = 2.5r_c$. Funcțiile D şi L sunt definite ca:

$$D(x,x',y',c_L) = \frac{1}{2}\left(c_0 + c_1 c_L^2\right)\rho S v(x,x',y')^2, \quad L(x,x',y',c_L) = \frac{1}{2}c_L \rho S v(x,x',y')^2,$$

unde S este aria aripei, ρ densitatea aerului, c_L coeficientul de ridicare aerodinamică şi $c_0 + c_1 c_L^2$ este coeficientul de rezistența a aerului. Pentru acest planor considerăm:

$$c_0 = 0.034, \quad c_1 = 0.069662, \quad S = 14, \quad \rho = 1.13.$$

Coeficientul de ridicare aerodinamică c_L satisface restricțiile de mărginire:

$$0 \le c_L \le c_{\max}.$$

În acelaşi timp se impun condițiile $x \ge 0$ şi $x' \ge 0$. În această problemă considerăm $c_{\max} = 1.4$, $m = 100$, $g = 9.81$ şi condițiile la limită:

$$x(0) = 0, \quad y(0) = 1000, \quad y(t_f) = 900,$$

$$x'(0) = x'(t_f) = 13.23, \quad y'(0) = y'(t_f) = -1.288.$$

Pentru rezolvarea acestei probleme intervalul $[0,t_f]$ se discretizează uniform în n_h intervale, iar ecuațiile de mişcare sunt aproximate prin schema trapezoidală. Condițiile inițiale sunt

$$x(t) = x(0) + x'(0)\left(\frac{t}{t_f}\right), \qquad y(t) = y(0) + (y(t_f) - y(0))\left(\frac{t}{t_f}\right), \qquad c_L(t) = c_{\max}/2.$$

evaluate în punctele de discretizare. Se consideră $t_f = 1$.

Expresia GAMS a modelului discretizat cu aceste condiții (inițiale şi la limită) este prezentat în figura 8.1.

```
$Ontext
Maximize the final horizontal position of a hang glider while
in the presence of a thermal updraft.
$Offtext

$if set n $set nh %n%
$if not set nh $set nh 1000
sets c coordinates / x distance
                     y altitude /
h intervals        / h0 * h%nh%/ ;

alias(h,i);

scalars nh   Number of intervals in mesh  / %nh% /
cL_min       bound on control variable    / 0.0 /
cL_max       bound on control variable    / 1.4 /
u_c   / 2.5 /
r_0   / 100 /
m     / 100 /
g     / 9.81 /
c0    / 0.034 /
c1    / 0.069662 /
S     / 14 /
rho   / 1.13 / ;

parameters c_0(c) initial position  / x 0, y 1000/
           v_0(c) initial velocity  / x 13.23, y -1.288/
           c_f(c) final position    / y 900/
           v_f(c) final velocity    / x 13.23, y -1.288/ ;

variables t_f
          pos(c,h)   position x distance y altitude
          vel(c,h)   velocity x distance y altitude
          cl(h)      control variables
          r(h)       the r function
          u(h)       the u function
          w(h)       the w function
          v(h)       the v function
          D(h)       the D function
          L(h)       the L function
          v_dot(c,h)
```

```
          final_x
          step       step size

positive variables step;

equations tf_eqn
          rdef(h)
          udef(h)
          wdef(h)
          vdef(h)
          Ddef(h)
          Ldef(h)
          vx_dot_def(h)
          vy_dot_def(h)
          obj
          pos_eqn(c,h)
          vel_eqn(c,h);

tf_eqn.. t_f =e= step*nh;

rdef(i).. r[i] =e= sqr(pos['x',i]/r_0 - 2.5);

udef(i).. u[i] =e= u_c*(1-r[i])*exp(-r[i]);

wdef(i).. w[i] =e= vel['y',i] - u[i];

vdef(i).. v[i] =e= sqrt(sqr(vel['x',i]) + sqr(w[i]));

Ddef(i).. D[i] =e= .5*(c0+c1*sqr(cL[i]))*rho*S*sqr(v[i]);

Ldef(i).. L[i] =e= .5* cL[i] *rho*S*sqr(v[i]);

vx_dot_def(i).. v_dot['x',i] =e= (-L[i]*w[i]/v[i] –
                                  D[i]*vel['x',i]/v[i])/m;

vy_dot_def(i).. v_dot['y',i] =e= ( L[i]*vel['x',i]/v[i] –
                                  D[i]*w[i]/v[i])/m - g;

obj.. final_x =e= pos('x','h%nh%');

pos_eqn(c,i-1).. pos[c,i] =e= pos[c,i-1] +
                        .5*step*(vel[c,i] + vel[c,i-1]);

vel_eqn(c,i-1).. vel[c,i] =e= vel[c,i-1] +
                        .5*step*(v_dot[c,i] + v_dot[c,i-1]);

* Boundary Conditions
cl.lo(h) = cL_min;
cl.up(h) = cL_max;
pos.lo('x',h) = 0;
vel.lo('x',h) = 0;
v.lo(h) = 0.01;

* Fixed values
pos.fx(c,'h0') = c_0(c);
pos.fx('y','h%nh%') = c_f('y');
```

```
vel.fx(c,'h0') = v_0(c);
vel.fx(c,'h%nh%') = v_f(c);

* Initial point
pos.l('x',h) = c_0('x') + v_0('x')*((ord(h)-1)/nh);
pos.l('y',h) = c_0('y') +
               ((ord(h)-1)/nh)*(c_f('y') - c_0('y'));
vel.l(c,h) = v_0(c);
cL.l(h) = cL_max/2;
step.l = 1.0/nh;

* Initial values for intermediate variables
t_f.l = step.l*nh;
r.l[i] = sqr(pos.l['x',i]/r_0 - 2.5);
u.l[i] = u_c*(1-r.l[i])*exp(-r.l[i]);
w.l[i] = vel.l['y',i] - u.l[i];
v.l[i] = sqrt(sqr(vel.l['x',i]) + sqr(w.l[i]));
D.l[i] = .5*(c0+c1*sqr(cL.l[i]))*rho*S*sqr(v.l[i]);
L.l[i] = .5* cL.l[i] *rho*S*sqr(v.l[i]);
v_dot.l['x',i] = (-L.l[i]*w.l[i]/v.l[i] –
                 D.l[i]*vel.l['x',i]/v.l[i])/m;
v_dot.l['y',i] = ( L.l[i]*vel.l['x',i]/v.l[i] –
                 D.l[i]*w.l[i]/v.l[i])/m - g;

model planor /all/;

$onecho >bench.opt
  solvers conopt knitro
$offecho
planor.optfile=1;

option nlp=bench;

planor.workspace = 40;

solve planor maximizing final_x using nlp;

*file res /g8.dat/;
*put res
*loop(h, put cl.l(h):10:7, put/)
* End G8
```

Fig. 8.1. Expresia GAMS a aplicației G8.

În tabelele de mai jos se prezintă performanțele algoritmilor CONOPT şi KNITRO. MINOS nu poate rezolva problema.

Tabelul 8.1

$n_h = 200,\ \ n = 2616$ (variabile), $m = 2410$ (restricții)

	#iter	Time	vfo
CONOPT	704	23.781	1248.8094263
KNITRO	366	21.671	1248.8963550

Tabelul 8.2

$n_h = 400,\ n = 5216$ (variabile), $m = 4810$ (restricții)

	#iter	Time	vfo
CONOPT	1078	45.328	1247.9743175
KNITRO	508	86.244	1247.9743047

Tabelul 8.3

$n_h = 800,\ n = 10416$ (variabile), $m = 9610$ (restricții)

	#iter	Time	vfo
CONOPT	1357	176.484	1247.9838908
KNITRO	599	309.264	1247.9838658

Tabelul 8.4

$n_h = 1000,\ n = 13016$ (variabile), $m = 12010$ (restricții)

	#iter	Time	vfo
CONOPT	1320	177.633	1247.9852155
KNITRO	662	498.276	1247.9851843

Tabelul 8.5

$n_h = 1200,\ n = 15616$ (variabile), $m = 14410$ (restricții)

	#iter	Time	vfo
CONOPT	1547	217.242	1247.9859370
KNITRO	604	631.468	1247.9858984

În figurile de mai jos se prezintă evoluția anumitor variabile ale acestei probleme. Figura 8.2 arată evoluția în timp a altitudinii $y(t)$ şi a coeficientului de ridicare ascensională $c_L(t)$. Figura 8.3 prezintă evoluția în timp a vitezelor $x'(t)$ şi $y'(t)$. Din figura 8.2 vedem că planorul începe zborul de la altitudinea $y(0) = 1000$. Apoi coboară până când acesta întâlneşte curentul ascendent centrat în jurul valorii $x = 2.5r_c = 250$. Ca atare, acesta urcă şi apoi coboară până la altitudinea finală $y(t_f) = 900$. În ceea ce priveşte evoluția vitezelor, din figura 8.3 vedem comportarea de tip bang-bang a acestora.

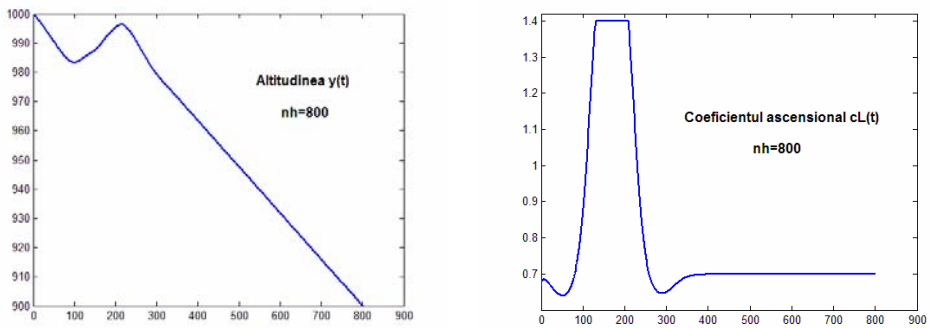

Fig. 8.2. Evoluția altitudinii $y(t)$ şi a coeficientului de ridicare aerodinamică $c_L(t)$.

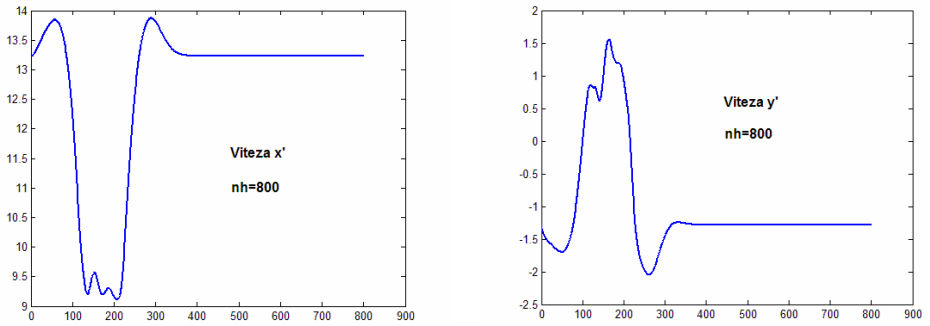

Fig. 8.3. Evoluția vitezelor $x'(t)$ şi $y'(t)$.

Aplicația G9 (Torsion) *Torsiunea elasto-plastică a unei bare.* Problema constă în determinarea câmpului de eforturi într-o bară cilindrică infinit lungă.
Versiunea infinit dimensională a acestei probleme este următoarea.

$$\min\{q(v) : v \in K\},$$

unde $q : K \to R$ este funcția pătratică:

$$q(v) = \frac{1}{2}\int_D \|\nabla v(x)\|^2 \, \mathrm{d}x - c\int_D v(x)\mathrm{d}x$$

pentru o constantă oarecare c (unghiul de torsiune pe unitatea de lungime) şi D este un domeniu mărginit cu frontieră netedă (secțiunea transversală în bară). Mulțimea convexă K este definită ca:

$$K = \left\{ v \in H_0^1(D) : |v(x)| \le dist(x, \partial D), x \in D \right\},$$

unde $dist(., \partial D)$ este distanța la frontiera lui D, şi $H_0^1(D)$ este spațiul Hilbert al tuturor funcțiilor cu suport compact în D astfel încât v şi $\|\nabla v\|^2$ aparțin lui

$L^2(D)$. Această formulare, precum şi interpretarea fizică a acestei probleme este prezentată de Glowinski [1984, pp.41-55].

Aproximarea prin elemente finite a problemei se obține prin discretizarea lui D şi înlocuirea problemei de minimizare a lui q pe $H_0^1(D)$ prin minimizarea lui q pe mulțimea funcțiilor liniare pe porțiuni care satisfac restricțiile specificate de K, aşa cum este descris de Averick, Carter, Moré şi Xue [1994]. Se consideră $D=[0,1]\times[0,1]$ şi se utilizează o triangulație cu n_x, respectiv n_y puncte de discretizare în direcția coordonatelor. Pentru unghiul de torsiune pe unitatea de lungime se consideră $c=5$. Ca valori inițiale se consideră funcția de distanță $dist(x,\partial D)$ evaluată în punctele de discretizare ale domeniului. Expresia GAMS a problemei este arătată în figura 9.1.

```
$ontext
Determine the stress potential in an infinitely long cylinder
when torsion is applied.
$offtext

$if not set nx $set nx 40
$if not set ny $set ny 40

sets nx grid points in 1st direction / x0*x%nx% /
     ny grid points in 2st direction / y0*y%ny% /

alias(nx,i),(ny,j);

parameters D(nx,ny) Distance to the boundary
           hx grid spacing for x
           hy grid spacing for y
           area area of triangle
           c some constant / 5.0 /;

hx := 1/(card(nx)-1);
hy := 1/(card(ny)-1);
area := 0.5*hx*hy;

D(i,j) := min(min(ord(i)-1,card(nx)-ord(i))*hx,
              min(ord(j)-1,card(ny)-ord(j))*hy);

variables v(nx,ny) the finite element approximation stress,
          linLower,
          linUpper,
          quadLower,
          quadUpper,
          stress;

Equations defLL,
          defLU,
          defQL,
          defQU,
          defstress;
```

```
defLL.. linLower =e= sum((nx(i+1),ny(j+1)), v[i+1,j] + v[i,j] +
                                            v[i,j+1]);
defLU.. linUpper =e= sum((nx(i-1),ny(j-1)), v[i,j] + v[i-1,j] +
                                            v[i,j-1]);
defQL.. quadLower =e= sum((nx(i+1),ny(j+1)),
                              sqr((v[i+1,j]-v[i,j])/hx) +
                              sqr((v[i,j+1]-v[i,j])/hy));
defQU.. quadUpper =e= sum((nx(i-1),ny(j-1)),
                              sqr((v[i,j]-v[i-1,j])/hx) +
                              sqr((v[i,j]-v[i,j-1])/hy));
defstress.. stress =e= area*( (quadLower + quadUpper)/2 -
                          c*(linLower + linUpper )/3);

model torsion / all /;

v.lo(i,j) = -d(i,j);
v.up(i,j) = d(i,j);
v.l (i,j) = d(i,j);

display d,hx,hy,area;

*$onecho >minos.opt
*  superbasics limit = 5000
*$offecho

*$onecho >bench.opt
* solvers conopt knitro minos.1
*$offecho

torsion.optfile=1;
torsion.workspace=120;
option nlp=conopt;
solve torsion minimizing stress using nlp;

file res1 /torsion.dat/
res1.pw=4000;
put res1;
loop(nx, loop(ny, put v.l(nx,ny):6:2; ); put /;); put /;

* End Torsion
```

Fig. 9.1. Expresia GAMS a aplicației G9.

Performanțele algoritmilor CONOPT, KNITRO şi MINOS pentru diferite valori ale lui n_x şi n_y sunt arătate în tabelele de mai jos.

Tabelul 9.1

$n_x = 40,\ n_y = 40,\ n = 1686$ (variabile), $m = 5$ (restricții)

	#iter	time	vfo
CONOPT	30	3.109	-0.4178327477
KNITRO	59	6.379	-0.4176095651
MINOS	1000	11.008	-0.4178327

Tabelul 9.2

$n_x = 50,\ n_y = 50,\ n = 2709$ (variabile), $m = 5$ (restricții)

	#iter	time	vfo
CONOPT	34	1.230	-0.4180876320
KNITRO	86	22.492	-0.4178122579

Tabelul 9.3

$n_x = 100,\ n_y = 100,\ n = 10409$ (variabile), $m = 5$ (restricții)

	#iter	time	vfo
CONOPT	60	8.621	-0.418391026
KNITRO	68	289.926	-0.4168094513

Tabelul 9.4

$n_x = 200,\ n_y = 200,\ n = 40406$ (variabile), $m = 5$ (restricții)

	#iter	time	vfo
CONOPT	109	85.191	-0.4184683991

În figura 9.1 se prezintă potențialul efortului în bara considerată pentru discretizarea $n_x = 100,\ n_y = 100$ şi $c = 5$.

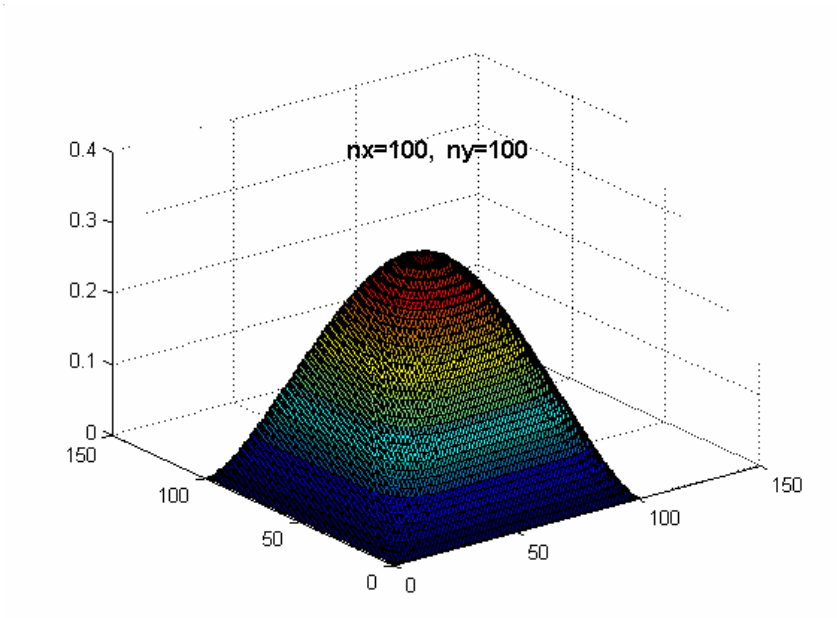

Fig.9.2. Soluția aplicației G9. $nx = 100,\ ny = 100$.

Astfel formulată problema ridică dificultăți majore algoritmilor de optimizare neliniară cu restricții. Chiar pentru dimensiuni modeste ale discretizării, algoritmul MINOS nu o poate rezolva. Şi algoritmul KNITRO pentru discretizări mai fine implică un timp de calcul total nerezonabil.

Problema se poate reformula ca una de optimizare fără restricții (vezi [Andrei, 2009]). În acest caz, performanțele algoritmilor de optimizare fără restricții sunt mult superioare. De exemplu algoritmul SCALCD [Andrei, 2007], [Andrei, 2009, pg.432], pentru discretizarea $n_x = 200$, $n_y = 200$, furnizează o soluție optimă a acestei problemei în 33.64 secunde. Algoritmul L-BFGS [Andrei, 2009, pg. 537] pentru aceeaşi discretizare furnizează soluția în 20.90 secunde, iar algoritmul TN (Newton trunchiat) [Andrei, 2009, pg. 571] în 19.75 secunde. Vedem imporatnța formulării problemei şi a utilizării algoritmilor specializați.

Aplicația G10 (*Lagar*) *Distribuția presiunii într-un lagăr de alunecare (cuzinet). Problema constă în determinarea distribuției presiunii într-un film subțire de lubrifiant între doi cilindri circulari.*
Versiunea infinit-dimensională a problemei este:

$$min\left\{q(v) : v \in K\right\},$$

$$q(v) = \frac{1}{2}\int_D w_q(x)\left\|\nabla v(x)\right\|^2 \mathrm{d}x - \int_D w_l(x)v(x)\mathrm{d}x$$

cu

$$w_q(z_1, z_2) = (1 + \varepsilon\cos z_1)^3, \quad w_l(z_1, z_2) = \varepsilon\sin z_1,$$

pentru o anumită constantă $\varepsilon \in (0,1)$ şi $D = (0, 2\pi) \times (0, 2b)$ unde $b > 0$ este o constantă. Mulțimea convexă K este $K = \left\{v \in H_0^1(D) : v \in D, v \geq 0\right\}$. Aproximarea prin elemente finite a acestei probleme se obține exact ca în aplicația de mai sus, unde de data aceasta $w_q(\xi_1, \xi_2) = (1 + \varepsilon\cos\xi_1)^3$ şi $w_l(\xi_1, \xi_2) = \varepsilon\sin\xi_1$.
Domeniul de admisibilitate este mulțimea

$$\Omega = \left\{v \in R^{n_x n_y} : v_{i,j} \geq 0\right\}.$$

Considerând $b = 10$ şi $\varepsilon = 0.1$, şi o discretizare $n_x \times n_y$ a domeniului $D = (0, 2\pi) \times (0, 2b)$, atunci pentru diferite valori a lui n_x şi n_y, performanțele algoritmilor CNOPT, KNITRO şi MINOS sunt prezentate în tabelele de mai jos. În figura 10.1 se prezintă expresia GAMS a acestei aplicații.

```
$ontext
Given the eccentricity e of the journal bearing, find the
pressure distribution in the lubricant separating the shaft
from the bearing.
$offtext

$if not set nx $set nx 40
$if not set ny $set ny 40

Set nx / 0*%nx% /
```

```
    ny / 0*%ny% /

alias (nx,i),(ny,j);
Scalar pi a famous constant
       b "grid is (0,2*pi)x(0,2*b)"  /10/
       e eccentricity                /0.1/
       hx, hy grid spacing
       area area of triangle;

pi = 2*arctan(inf);
hx = 2*pi/%nx%;
hy = 2*b/%ny%;
area = 0.5*hx*hy;

Parameter wq(nx);
wq(nx) = power(1+e*cos((ord(nx)-1)*hx),3);

Positive variable v(nx,ny);
Variable obj;

Equation defobj Objective function;

defobj.. obj =e= (hx*hy/12)*sum{(nx(i+1),ny(j+1)),
     (wq[i]+2*wq[i+1])*
     (sqr((v[i+1,j]-v[i,j])/hx) + sqr((v[i,j+1]-v[i,j])/hy))}+
     (hx*hy/12)*sum{(nx(i+1),ny(j+1)), (2*wq[i+1]+2*wq[i])*
     (sqr((v[i,j+1]-v[i+1,j+1])/hx) +
     sqr((v[i+1,j]-v[i+1,j+1])/hy))} -
     hx*hy*sum {(nx,ny), e*sin((ord(nx)-1)*hx)*v[nx,ny]};

* Starting point
v.l[nx,ny] = max(sin((ord(nx)-1)*hx),0);

v.fx[nx, '0'] = 0;
v.fx[nx,'%ny%'] = 0;
v.fx[ '0',ny] = 0;
v.fx['%nx%',ny] = 0;

model lagar /all/;
$onecho >minos.opt
  superbasic limit 5000
$offecho
$onecho >bench.opt
  solvers conopt knitro
$offecho
lagar.optfile=1;
lagar.workspace=125;
lagar.reslim=6000;
option nlp=bench;
solve lagar minimizing obj using nlp;
*file rez /lagar.dat/
*put rez;
*loop(i, loop(j, put v.l(i,j):6:2); put/;);put/;
```

Fig. 10.1. Expresia GAMS a aplicației G10.

Tabelul 10.1

$n_x = 40,\ n_y = 40,\ n = 1682$ (variabile), $m = 1$ (restricții)

	#iter	time	vfo
CONOPT	29	6.121	-0.154791668
KNITRO	86	8.201	-0.154754911
MINOS	1300	13.590	-0.1547917

Tabelul 10.2

$n_x = 50,\ n_y = 50,\ n = 2602$ (variabile), $m = 1$ (restricții)

	#iter	time	vfo
CONOPT	56	22.133	-0.154820504
KNITRO	90	18.847	-0.154765045
MINOS	2100	50.379	-0.1548205

Tabelul 10.3

$n_x = 100,\ n_y = 100,\ n = 10202$ (variabile), $m = 1$ (restricții)

	#iter	time	vfo
CONOPT	38	6.012	-0.154839385
KNITRO	56	160.661	-0.153782131

Tabelul 10.4

$n_x = 200,\ n_y = 200,\ n = 40402$ (variabile), $m = 1$ (restricții)

	#iter	time	vfo
CONOPT	63	56.691	-0.154828784
KNITRO	90	3952.403	-0.154701894

Figura 10.2 prezintă distribuția presiunii în lagăr pentru discretizarea $n_x = 40$, $n_y = 40$ şi excentricitatea $\varepsilon = 0.1$.

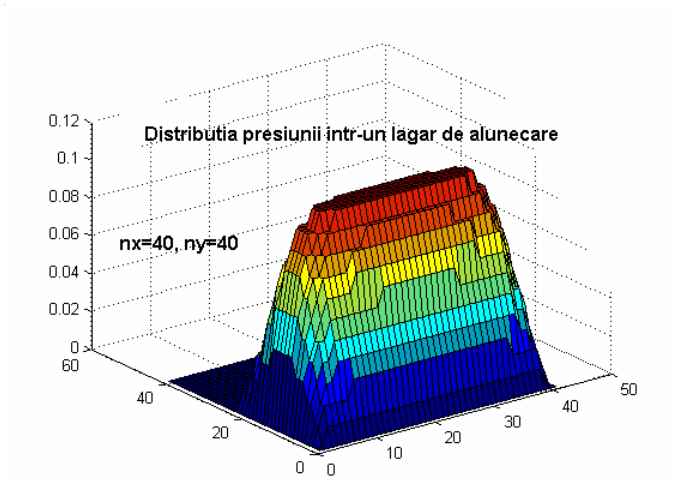

Fig. 10.2. Soluția aplicației G10 pentru $\varepsilon = 0.1$ şi $b = 10$.

Referințe

Andrei, N., (2001) *Numerical examples solved with SPENBAR – modified penalty barrier method for large-scale nonlinear programming problems.* ICI Technical Report No. 1/2001, February 2001.

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

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

Andrei, N., (2010) *Critica raţiunii algoritmilor deprogramare liniară.* Editura Academiei Române, Bucureşti. Manuscris predat Editurii Academiei Române, 2010.

Andrei, N., (2007) *SCALCG – Scaled conjugate gradient algorithms for unconstrained optimization.* Technical Report No. 17/2007, Research Institute for Informatics, Bucharest, March 30, 2007.

Andrei, N., (2011) CAON: *O colecţie de aplicaţii de optimizare neliniară în limbajul GAMS.* Raport Tehnic, No.1/2011, Institutul Naţional de Cercetare - Dezvoltare în Informatică, Bucureşti, 31 Ianuarie 2011. (105 pagini şi CD) (Raport introdus în Biblioteca Academiei Române)

Anițescu, M., Şerban, R., (1998) *A sparse superlinearly convergent SQP with applications to two-dimensional shape optimization.* Preprint ANL/MCS-P706-0198, Argonne National Laboratory, Argonne, Illinois, 1998.

Arrow, K.J., Solow, R.M., (1958) *Gradient methods for constrained maxima, with weakened assumptions.* in: Arrow, K.J., Hurwicz, L., Uzawa, H., (Eds.) *Studies in Linear and Nonlinear Programming.* Stanford University Press, Stanford, California, 1958, pp.166-176.

Averick, B.M., Carter, R.G., Moré, J.J., Xue, G.-L., (1994) *The MINPACK-2 test problem collection.* Preprint MCS-P153-0694, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1994.

Bondarenko, A.S., Bortz, D.M., Moré, J.J., (1999) *COPS: Large-scale nonlinearly constrained optimization problems.* Technical Report ANL/MCS-TM-237, Argonne National Laboratory, Argonne, Illinois, September 1998, October 1999 (revision).

Bryson, A.E., (1999) *Dynamic optimization.* Addison-Wesley, New-York, 1999.

Bulirsch, R., Nerz, E., Pesch, H.J., von Stryk, O., (1993) *Combining direct and indirect methods in nonlinear optimal control: Range maximization of a hang glider.* În Optimal Control, R. Bulirsch, A. Miele, J. Stoer and K.H. Well (Eds.) Birkhäuser Verlag, 1993, pp.273-288.

Cesari, L., (1983) *Optimization – theory and applications*. Springer Verlag, Bonn, 1983.

Dolan, E.D., Moré, J.J., Munson, T.S., (2004) *Benchmarking optimization software with COPS 3.0.* Preprint ANL/MCS-TM-273, Argonne National Laboratory, Argonne, Illinois, February 2004.

Gill, Ph.E., Murray, W., (1974a) *Newton-type methods for unconstrained and linear constrained optimization.* Mathematical Programming, vol.28, 1974, pp.311-350.

Gill, Ph.E., Murray, W., (1974b) *Quasi-Newton methods for linearly constrained optimization.* in: Gill, Ph.E., Murray, W., (Eds.) *Numerical Methods for Constrained Optimization.* Academic Press, London, New York, San Francisco, 1974, pp.67-92.

Gill, Ph.E., Murray, W., (1974c) *Methods for large-scale linearly constrained problems.* in: Gill, Ph.E., Murray, W., (Eds.) *Numerical Methods for Constrained Optimization.* Academic Press, London, New York, San Francisco, 1974, pp.93-147.

Glowinski, R., (1984) *Numerical Methods for Nonlinear Variational Problems.* Springer-Verlag, Berlin, 1984.

Graham, R.L., (1970) *The largest small hexagon.* Journal of Combinatorial Theory, (A) 18, 1975, pp.165-170.

Hestens, M.R., (1969) *Multiplier and gradient methods.* Journal of Optimization Theory and Applications, vol.4, 1969, pp.303-320.

Hock, W., Schittkowski, K., (1981) *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems, vol.187, Springer-Verlag, Berlin 1981.

McCormick, G.P., (1970a) *The variable reduction method fot nonlinear programming.* Management Science, 17 (3), 1970, pp.146-160.

McCormick, G.P., (1970b) *A second order method for the linearly constrained nonlinear programming.* În J.B. Rosen, O.L. Mangasarian şi K. Ritter (Eds.) Nonlinear Programming, Academic Press, New York, 1970, pp.207-243.

Murtagh, B.A., Saunders, M.A., (1978) *Large-scale linearly constrained optimization.* Mathematical Programming, vol.14, 1978, pp.41-72.

Murtagh, B.A., Saunders, M.A., (1980) *MINOS/AUGMENTED user's manual.* Technical Report SOL 80-14, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, CA 94305, 1980.

Murtagh, B.A., Saunders, M.A., (1982) *A projected lagrangian algorithm and its implementation for sparse nonlinear constraints.* Mathematical Programming Study, vol.16, 1982, pp.84-117.

Murtagh, B.A., Saunders, M.A., (1995) *MINOS 5.4 user's guide.* Technical Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, CA 94305, February 1995.

Powell, M.J.D., (1969) *A method for nonlinear constraints in optimization problems.* in: Fletcher, R., (Ed.) *Optimization.* Academic Press, New York, 1969, pp.283-297.

Robinson, S.M., (1972) *A quadratically convergent algorithm for general nonlinear programming problems.* Mathematical Programming, vol.3, 1972, pp.145-156.

Rosen, J.B., Kreuser, J., (1972) *A gradient projection algorithm for nonlinear constraints.* in: Lootsma, F.A., (Ed.) *Numerical Methods for Non-Linear Optimization.* Academic Press, London, 1972, pp.297-300.

Sargent, E., Murtagh, B.A., (1973) *Projection methods for nonlinear programming.* Mathematical Programming, vol.4, 1973, pp.245-268.

Schittkowski, K., (1987) *More Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems, vol.282, Springer-Verlag, Berlin, 1987.

Wolfe, P., (1967) *Methods of nonlinear programming.* În J. Abadie (Ed.) Nonlinear Programming, North-Holland, Amsterdam, 1967, pp.97-131.

Wright, M.H., (1976) *Numerical methods for nonlinearly constrained optimization.* SLAC Report No.193, 1976, Stanford University, California, 1976.

Dr. Neculai Andrei
Institutul Naţional de Cercetare-Dezvoltare în Infornatică
8-10, Bdl. Averescu, Bucureşti

August 8, 2011