Programy pro řešení rovnic chemické rovnováhy

Lukšan, Ladislav
2005

Institute of Computer Science
Academy of Sciences of the Czech Republic

Programy pro řešení rovnic chemické rovnováhy

L.Lukšan, M.Rozložník, M.Tůma

Technical report No. V954

Listopad 2005

Pod Vodárenskou věží 2, 182 07 Prague 8 phone: +420 2 688 42 44, fax: +420 2 858 57 89,
e-mail:e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Programy pro řešení rovnic chemické rovnováhy

L.Lukšan, M.Rozložník, M.Tůma [1]

Technical report No. V954

Listopad 2005

Abstrakt:

V práci jsou analyzovány rovnice popisující chemickou rovnováhu a vyvinuty efektivní metody pro jejich řešení. Kromě metod pro výpočet vhodné počáteční aproximace řešení jsou studovány a implementovány metody pro řešení systémů nelineárních rovnic. Jsou též popsány příslušné programy napsané v jazyce Fortran a uvedeny jejich výpisy. Přiloženy jsou ukázky výpočtu a výsledky vzorového problému.

Keywords:
Chemická rovnováha, nelineární rovnice, nepodmíněná minimalizace, modifikovaná Newtonova metoda, softwarová realizace.

[1]Tato práce byla podpořena Ministerstvem průmyslu a obchodu ČR, grant TANDEM: FT-TA/066, a výzkumným záměrem AV0Z10300504 Akademie věd ČR.

1 Rovnice popisující chemickou rovnováhu

Chemická rovnováha je popsána rovnicí

$$R - C_{sum} = 0,$$

kde

$$R = B^T * C, \quad C = 10^{\log C}, \quad \log C = H * \log C_{main} + \log K.$$

Zde $B \in R^{m \times n}$, $B \in R^{m \times n}$ jsou konstantní matice s m řádky a n sloupci, K je konstantní vektor dimenze m C_{sum} je konstantní vektor dimenze n (data úlohy) a log značí dekadický logaritmus. Logaritmem vektoru budeme chápat vektor, jehož všechny složky jsou logaritmy složek původního vektoru. Podobně budeme chápat exponenciální funkci vektoru. Pro zjednodušení zápisu budeme používat označení uvedené v následující tabulce.

Chemie	Matematika
$\log C_{main}$	x
C_{main}	X
$\log C$	c
C	C
$\log K$	k
B	B
H	H
C_{sum}	S

Tedy malá písmena budou označovat logaritmické veličiny a velká originální hodnoty a matice. Pak lze chemickou rovnováhu vyjádřit ve tvaru

$$f(x) = S - B^T 10^{Hx+k} = 0. \tag{1}$$

Tento systém nelineárních rovnic je možné převést na optimalizační úlohu zavedením účelové funkce

$$F(x) = \frac{1}{2} f^T(x) f(x) = \frac{1}{2} \sum_{i=1}^{n} f_i^2(x) \tag{2}$$

Vzhledem k tomu, že se v definici této funkce vyskytují exponenciály, nastávají problémy s velkými čísly (přetékání). Již poměrně malé odchylky od optimálního řešení způsobují, že funkce $F(x)$ nabývá obrovských hodnot (větších než 10^{100}). Další problémy jsou způsobeny tím, že logaritmy jsou definovány pouze pro kladné hodnoty. Hodnoty $X = C_{main}$ musí být tudíž větší než nula, což vyžaduje použití omezení ve tvaru nerovností. Je však problém stanovit vhodné dolní meze, neboť hodnoty jednotlivých proměnných se liší o mnoho řádů. Proto je výhodné volit za proměnné nikoliv veličiny $X = C_{main}$, ale $x = \log C_{main}$. Tyto veličiny mohou nabývat libovolných hodnot, takže není třeba uvažovat žádná omezení.

Vzhledem k těmto skutečnostem je velmi důležité zvolit dobrý počáteční odhad řešení. V podstatě jde o to nalézt vektor x_1 tak, aby složky vektoru 10^{Hx_1+k} nabývaly rozumných hodnot. Jednou z možností je zvolit x_1 tak, aby norma $\|Hx_1 + k\|$ byla minimální. Tato formulace je ekvivalentní lineární úloze nejmenších čtverců, minimalizovat

$$\frac{1}{2}\|Hx + k\|^2 = \frac{1}{2}(x^T H^T Hx + 2k^T Hx + k^T k),$$

což vede na soustavu lineárních rovnic $H^T Hx + H^T k = 0$, která má jednoznačné řešení $x_1 = -(H^T H)^{-1} H^T k$. Jednodušší a také výhodnější je převést rovnici $Hx_1 + k = 0$ (která nemá řešení, neboť matice H je obdélníková) na rovnici $B^T Hx_1 + B^T k = 0$. Tím dostaneme počáteční odhad $x_1 = -(B^T H)^{-1} B^T k$. Další možností je aproximovat exponenciální funkci rozvojem $\exp(x) \approx 1 + x$, což vede na rovnici

$$B^T(e + (Hx + k)\ln 10) = S,$$

kde e je vektor, jehož všechny prvky jsou jednotky. Tato rovnice má řešení

$$x = \frac{1}{\ln 10}(B^T H)^{-1}(S - B^T(e + k\ln 10)) = S.$$

Velmi výhodnou možností je zvolit počáteční odhad x_1 tak, aby byl řešením soustavy lineárních nerovností $Hx + k \leq 0$ (nebo jeho aproximací pokud toto řešení neexistuje), neboť v tomto případě platí $10^{Hx_1+k} \leq 1$. Soustavu nerovností $Hx + k \leq 0$ lze řešit metodami lineárního programování, například simplexovou metodou, jak je ukázáno v oddílu 2.

Jak už bylo naznačeno, nastávají při řešení úlohy (1) pomocí minimalizace funkce (2) problémy s rozsahem hodnot jednotlivých veličin. Například hodnota $F(x) = (1/2)f^T(x)f(x)$ a norma gradientu $g(x) = \nabla F(x) = J^T(x)f(x)$ jsou úměrné čtverci normy vektoru $f(x)$ a skalární součin $g^T(x)g(x)$, který používá většina optimalizačních metod je úměrný čtvrté mocnině $\|f(x)\|$. Z tohoto důvodu prakticky nelze použít gradientní optimalizační metody. Jistou možností jsou komparativní metody, které používají pouze hodnoty účelové funkce. Tyto metody však konvergují pomalu a lze je použít pouze pro úlohy menších rozměrů. V našich prvních pokusech jsme použili kombinaci komparativní metody "patern search" a metody s proměnnou metrikou BFGS (Broyden, Fletcher, Goldfarb, Shanno), která je podrobně popsána například v [20]). Výsledky jsou uvedeny oddílu 8.

Abychom se vyhnuli problémům s extrémně velkými hodnotami, zaměřili jsme se na metody pro řešení systémů nelineárních rovnic. Tyto metody používají pouze vektor $f(x)$ a Jacobiovu matici

$$J(x) = \ln 10\, B^T 10^{Hx+k} H, \tag{3}$$

takže nedochází k problémům se čtverci nebo čtvrtými mocninami. Metody pro řešení systémů nelineárních rovnic jsou však citlivější na přesnost výpočtu Jacobiovy matice nebo na volbu její aproximace. Jelikož požadujeme aby metoda byla globálně konvergentní, je třeba pečlivě upravit iterační proces tak aby nedocházelo k divergenci nebo

k selhání metody. Těmto úpravám a teoretickému vyšetření metod pro řešení systémů nelineárních rovnic je věnován oddíl 3 této práce. Další oddíly obsahují popis podprogramů `PNEQU` a `PBEQU`, vyvinutých v rámci tohoto grantového projektu, a výpisy všech modulů podprogramu `PNEQU`, který byl nakonec použit k řešení úloh chemické rovnováhy.

2 Řešení soustav lineárních nerovností

Jak již bylo zmíněno, je výhodné volit počáteční odhad x_1 tak, aby byly splněny nerovnosti $Hx_1 + k \leq 0$. Tuto úlohu lze řešit pomocí posloupnosti úloh lineárního programování: Minimalizovat

$$\sum_{i \in I_3(x)} h_i^T x + k_i$$

na množině zadané omezeními

$$h_i x + k_i \leq 0, \quad i \in I_1(x) \cup I_2(x),$$

kde

$$\begin{aligned} I_1(x) &= \{1 \leq i \leq m : h_i x + k_i < 0\} \\ I_2(x) &= \{1 \leq i \leq m : h_i x + k_i = 0\} \\ I_3(x) &= \{1 \leq i \leq m : h_i x + k_i > 0\} \end{aligned}$$

jsou množiny neaktivních, aktivních a porušených omezení (h_i je i-tý řádek matice H a k_i je i-tý prvek vektoru k). Úloha lineárního programování se změní, když se změní množina $I_3(x)$.

Uvedená posloupnost úloh lineárního programování se řeší iteračně metodou promítaných gradientů, což je jistá úprava simplexové metody. V každém iteračním kroku máme k disposici bod x, množiny $I_1(x)$, $I_2(x)$, $I_3(x)$, matice H_1, H_2, H_3 (části matice H, odpovídající množinám $I_1(x)$, $I_2(x)$, $I_3(x)$) a matici $W_2 = (H_2 H_2^T)^{-1}$, která reprezentuje lineární varietu určenou aktivními omezeními. Nejprve určíme směrový vektor $s = -P_2 g_3$ (záporně vzatý promítnutý gradient), kde

$$P_2 = I - H_2^T W_2 H_2, \quad g_3 = \sum_{i \in I_3(x)} h_i.$$

Je-li směrový vektor s nulový, určíme vektor Lagrangeových multiplikátorů $u_2 = W_2 H_2 g_3$. Pokud $u_2 \geq 0$ a $I_3(x) = \emptyset$, je bod x řešením soustavy $Hx + k \leq 0$. Pokud $u_2 \geq 0$ a $I_3(x) \neq \emptyset$, nemá soustava $Hx + k \leq 0$ řešení. Pokud $u_2 < 0$, najdeme nejmenší Lagrangeův multiplikátor, ubereme odpovídající index z množiny $I_2(x)$, přidáme ho do množiny $I_1(x)$, upravíme matici W_2 a vrátíme se zpět na začátek iteračního kroku.

Není-li směrový vektor nulový, najdeme maximální délky kroku

$$\alpha_1 = \min_{i \in I_1, h_i s > 0} \left(-\frac{h_i x + k_i}{h_i s} \right)$$

$$\alpha_3 \quad = \quad \min_{i \in I_3, h_i s < 0} \left(-\frac{h_i x + k_i}{h_i s} \right)$$

a položíme $\alpha = \min(\alpha_1, \alpha_3)$. Jestliže $\alpha = \infty$, nemá soustava $Hx + k \leq 0$ řešení. Jestliže $\alpha = \alpha_1$, ubereme odpovídající index z množiny $I_1(x)$, přidáme ho do množiny $I_2(x)$ a upravíme matici W_2. Jestliže $\alpha = \alpha_3$, ubereme odpovídající index z množiny $I_3(x)$, přidáme ho do množiny $I_2(x)$ a upravíme matici W_2. Nakonec položíme $x^+ = x + \alpha s$ (nový odhad řešení) a zahájíme další iterační krok.

Zbývá ukázat, jak se upravuje matice W_2 při ubrání nebo přidání aktivního omezení. Při ubrání aktivního omezení s indexem i, budeme pro jednoduchost předpokládat, že toto omezení odpovídá poslednímu řádku a poslednímu sloupci matice W_2. Pak

$$W_2 = \begin{bmatrix} W, & w \\ w^T, & \omega \end{bmatrix} \quad \Rightarrow \quad W_2^- = W - \frac{ww^T}{\omega}.$$

Při přidání aktivního omezení s indexem i platí

$$W_2^+ = \begin{bmatrix} W_+, & w_+ \\ w_+^T, & \omega_+ \end{bmatrix},$$

kde

$$\begin{aligned} W_+ &= W_2 + \frac{W_2 H_2 h_i^T h_i H_2^T W_2}{h_i h_i^T - h_i H_2^T W_2 H_2 h_i^T}, \\ w_+ &= -\frac{W_2 H_2 h_i^T}{h_i h_i^T - h_i H_2^T W_2 H_2 h_i^T}, \\ \omega_+ &= \frac{1}{h_i h_i^T - h_i H_2^T W_2 H_2 h_i^T}. \end{aligned}$$

Algoritmus, který jsme stručně popsali je realizován podprogramem `MXDRFP`, jehož výpis je uveden v oddílu 6.3.

3 Teoretické vlastnosti metod pro řešení systémů nelineárních rovnic

Nechť $f : \mathcal{R}^n \to \mathcal{R}^n$ je spojitě diferencovatelné zobrazení. Uvažujme systém nelineárních rovnic

$$f(x) = 0, \tag{4}$$

kde $x \in \mathcal{R}^n$ je vektor neznámých. Označme $J(x)$ Jacobiovu matici zobrazení f, takže

$$(J(x))_{ij} = \frac{\partial f_i(x)}{\partial x_j}, 1 \leq i \leq n, 1 \leq j \leq n.$$

Nechť $x_1 \in \mathcal{R}^n$, $\overline{F} \geq \|f(x_1)\|$ a $\overline{\Delta} \geq 0$. Označme

$$\mathcal{L}(\overline{F}) = \{x \in \mathcal{R}^n : \|f(x)\| \leq \overline{F}\}$$

a

$$\mathcal{D}(\overline{F}, \overline{\Delta}) = \{x \in \mathcal{R}^n : \|x - y\| \leq \overline{\Delta} \text{ pro nějaké } y \in \mathcal{L}(\overline{F})\}$$

Abychom mohli vyšetřovat konvergenci iteračních metod pro řešení soustavy rovnic (4), budeme používat tyto předpoklady.

A1: *Jacobiova matice $J(x)$ je definovaná a omezená na $\mathcal{D}(\overline{F}, \overline{\Delta})$, čili*

$$\|J(x)\| \leq \overline{J}, \quad \forall x \in \mathcal{D}(\overline{F}, \overline{\Delta})$$

A2: *Jacobiova matice $J(x)$ je lipschitzovsky spojitá na $\mathcal{D}(\overline{F}, \overline{\Delta})$, čili*

$$\|J(y) - J(x)\| \leq \overline{L}\|y - x\| \quad \forall x, y \in \mathcal{D}(\overline{F}, \overline{\Delta})$$

Zaměříme se na metody spádových směrů s Armijovým výběrem délky kroku, které generují posloupnost bodů $x_i \in \mathcal{R}^n$, $i \in \mathcal{N}$, takovou, že

$$x_{i+1} = x_i + \alpha_i s_i, \quad i \in \mathcal{N}, \tag{5}$$

kde $s_i \in \mathcal{R}^n$ je směrový vektor určený nepřesným řešením soustavy lineárních rovnic $A_i s + f_i = 0$ a kde délka kroku α_i se vybírá tak, aby byl zaručen dostatečný pokles funkce $\|f(x)\|$. Zde A_i je aproximace matice $J_i = J(x_i)$ a $f_i = f(x_i)$.

Pro vyšetřování metod spádových směrů s Armijovým výběrem délky kroku budeme používat účelovou funkci

$$F(x) = \frac{1}{2}\|f(x)\|^2, \tag{6}$$

která má stejná lokální i globální minima jako norma $\|f(x)\|$, a označíme $F_i = F(x_i)$, $g_i = g(x_i)$, $i \in \mathcal{N}$, kde $g(x) = J^T(x)f(x)$ je gradient funkce $F(x)$.

Vliv nepřesnosti řešení soustavy $A_i s + f_i = 0$ na globální konvergenci metody (5) je studován a úspěšně popsán v [3]-[4], [13]-[14], [17]. Popsat vliv nepřesnosti aproximace A_i Jacobiovy matice J_i je mnohem složitější. V [2] je studován případ aproximace Jacobiovy matice pomocí diferencí. Je však vyšetřována pouze lokální konvergence. První výsledky týkající se obecného případu jsou uvedeny v [22]. Zde se budeme podrobně zabývat oběma typy nepřesnosti.

Začneme definicí třídy metod spádových směrů s Armijovým výběrem délky kroku. Detailnější informace o těchto metodách lze vyčíst z Algoritmu 1.

Definice 1 *Řekneme, že základní metoda $x_{i+1} = x_i + \alpha_i s_i$, $i \in \mathcal{N}$, pro řešení systému nelineárních rovnic $f(x) = 0$ je metodou spádových směrů s Armijovým výběrem délky kroku, jestliže:*

D1: *Směrové vektory $s_i \in \mathcal{R}^n$, $i \in \mathcal{N}$, se určují tak, že*

$$\|A_i s_i + f_i\| \leq \overline{\omega}\|f_i\|, \tag{7}$$

kde $0 \leq \overline{\omega} < 1$.

D2: *Délka kroku $\alpha_i > 0$, $i \in \mathcal{N}$, se vybírá tak, že α_i je první člen posloupnosti α_i^j, $j \in \mathcal{N}$, kde $\alpha_i^1 = 1$ a $\underline{\beta}\alpha_i^j \leq \alpha_i^{j+1} \leq \overline{\beta}\alpha_i^j$ s $0 < \underline{\beta} \leq \overline{\beta} < 1$, pro který platí*

$$F_{i+1} - F_i \leq -2\underline{\rho}(1 - \overline{\omega})\alpha_i F_i, \tag{8}$$

kde $0 < \underline{\rho} < 1$.

Podmínka (8) je velmi podobná podmínce

$$\|f_{i+1}\| - \|f_i\| \leq -\underline{\rho}(1 - \overline{\omega})\alpha_i\|f_i\|$$

použité v [14]. Pro další úvahy budeme často používat tyto předpoklady:

A3: *Matice $J_i^{-1} = J^{-1}(x_i)$, $i \in \mathcal{N}$, jsou definované a stejnoměrně omezené v bodech posloupnosti $x_i \in \mathcal{L}(\overline{F})$, $i \in \mathcal{N}$, generované metodou spádových směrů s Armijovým výběrem délky kroku (D), čili*

$$\|J_i^{-1}\| \leq 1/\underline{J}, \quad \forall i \in \mathcal{N}.$$

A4: *Platí A3 a existuje konstanta $0 \leq \overline{\vartheta} < (1/2)(1 - \overline{\omega})\underline{J}$ taková, že*

$$\|A_i - J_i\| \leq \overline{\vartheta}, \quad \forall i \in \mathcal{N}.$$

A5: *Matice A_i^{-1}, $i \in \mathcal{N}$, jsou definované a stejnoměrně omezené, čili*

$$\|A_i^{-1}\| \leq 1/\underline{A}, \quad \forall i \in \mathcal{N}.$$

Lemma 1 *Nechť je splněn předpoklad A4 a nechť platí (7). Pak existuje konstanta $0 \leq \overline{\eta} < 1$ taková, že*

$$\|J_i s_i + f_i\| \leq \overline{\eta}\|f_i\|, \quad \forall i \in \mathcal{N}. \tag{9}$$

Důkaz Jelikož platí A4, můžeme psát

$$\|A_i s_i\| \geq \|J_i s_i\| - \|(A_i - J_i)s_i\| \geq (\underline{J} - \overline{\vartheta})\|s_i\|. \tag{10}$$

Podobně, (7) implikuje

$$\|A_i s_i\| \leq (1 + \overline{\omega})\|f_i\|. \tag{11}$$

Spojením obou nerovností dostaneme

$$\|s_i\| \leq \frac{1+\overline{\omega}}{\underline{J}-\overline{\vartheta}}\|f_i\|, \tag{12}$$

takže můžeme psát

$$\begin{aligned}\|J_i s_i + f_i\| &\leq \|A_i s_i + f_i\| + \|(J_i - A_i)s_i\| \leq \overline{\omega}\|f_i\| + \overline{\vartheta}\|s_i\| \\ &\leq \left(\overline{\omega} + \overline{\vartheta}\frac{1+\overline{\omega}}{\underline{J}-\overline{\vartheta}}\right)\|f_i\| \triangleq \overline{\eta}\|f_i\|.\end{aligned}$$

Použijeme-li nerovnost $0 \leq \overline{\vartheta} < (1/2)(1-\overline{\omega})\underline{J}$, dostaneme $0 \leq \overline{\eta} < 1$. □

Lemma 1 ukazuje, že nepřesnost Jacobiovy matice může být zahrnuta do nepřesnosti řešení soustavy lineárních rovnic, takže prakticky všechny výsledky týkající se nepřesné Newtonovy metody (to jest výsledky uvedené v [4] nebo [14]) mohou být použity pokud je splněn předpoklad A4. Bohužel předpoklad A4 nemůžeme ověřit, neznáme-li Jacobiovu matici. Z tohoto důvodu musíme použít jiný postup pro konstrukci globálně konvergentního algoritmu. Jednou možností je použití vhodné strategie přerušování. Ta je založena na jednoduchém rozhodovacím pravidle:

D3: Jestliže $A_i \neq J_i$ a podmínka (8) nebyla splněna v $\overline{j}_1$ po sobě jdoucích Armijových krocích, položíme $A_i = J_i$ a opakujeme iteraci.

Potom buď $A_i = J_i$, takže může být použita teorie vyvinutá pro nepřesnou Newtonovu metodu, nebo $A_i \neq J_i$ a $\alpha_i \geq \underline{\beta}^{\overline{j}_1}$, což eliminuje předpoklad A4 z důkazu globální konvergence (viz větu 1). Dosavadní úvahy lze realizovat pomocí jednoduchého algoritmu:

Algoritmus 1

Data: $0 < \underline{\beta} \leq \overline{\beta} < 1$, $0 < \underline{\rho} < 1$, $0 \leq \overline{\omega} < 1$, $\overline{\varepsilon} > 0$, $\overline{i} > 0$, $0 < \underline{j} \leq \overline{j}_1 \leq \overline{j}_2$, $0 < \overline{k} \leq \infty$ (∞ je povoleno).

Krok 1: Inicializace. Zvolíme počáteční bod $x_1 \in \mathcal{R}^n$ a vypočteme vektor $f_1 := f(x_1)$. Položíme $k := 1$ a $i := 1$.

Krok 2: Test na ukončení. Jestliže $\|f_i\| \leq \overline{\varepsilon}$, ukončíme výpočet je nalezeno řešení). Jestliže $i > \overline{i}$, ukončíme výpočet (příliš mnoho iterací).

Krok 3: Určení směrového vektoru. Jestliže $k = 1$, vypočteme matici $J_i := J(x_i)$ a položíme $A_i = J_i$. Určíme $0 \leq \overline{\omega}_i \leq \overline{\omega}$ a vypočteme vektor $s_i \in \mathcal{R}^n$ splňující podmínku $\|A_i s_i + f_i\| \leq \overline{\omega}_i\|f_i\|$ (například pomocí algoritmu 2 popsaným v dále).

Krok 4: Výběr délky kroku. (a) Položíme $\alpha_i^1 := 1$ a $j := 1$.
(b) Položíme $x_{i+1} := x_i + \alpha_i^j s_i$ a vypočteme $f_{i+1} := f(x_{i+1})$. Jestliže platí (8), přejdeme na Krok 5.
(c) Jestliže $k = 1$ a $j > \overline{j}_2$, ukončíme výpočet (selhání algoritmu). Jestliže $k > 1$ a $j > \overline{j}_1$, položíme $k := 1$ a přejdeme na Krok 3. V opačném případě určíme hodnotu $\underline{\beta}\alpha_i^j \leq \alpha_i^{j+1} \leq \overline{\beta}\alpha_i^j$, položíme $j := j + 1$ a přejdeme ne Krok 4b.

Krok 5: Aktualizace. Jestliže $j \leq \underline{j}$ a $k \leq \overline{k}$, vypočteme matici A_{i+1} pomocí zvolené metody, položíme $k := k + 1$, $i := i + 1$ a přejdeme na Krok 2. Jestliže $j > \underline{j}$ nebo $k > \overline{k}$, položíme $k := 1$, $i := i + 1$ a přejdeme na Krok 2.

Nyní ukážeme, že posloupnost bodů $x_i \in \mathcal{R}^n$, $i \in \mathcal{N}$, generovaná Algoritmem 1, je globálně konvergentní za předpokladu, že nedojde k selhání v Kroku 4. Navíc zformulujeme podmínky, které vyloučí toto selhání. Protože požadované výsledky nelze snadno nalézt v literatuře, uvedeme poměrně stručný úplný důkaz.

Věta 1 *Nechť Jacobiova matice funkce $f : \mathcal{R}^n \to \mathcal{R}^n$ je definovaná na $\mathcal{D}(\overline{F}, \overline{\Delta})$ a nechť $x_i \in \mathcal{R}^n$, $i \in \mathcal{N}$, je posloupnost generovaná Algoritmem 1, který neselže v Kroku 4. Pak $f_i \to 0$. Je-li navíc splněn předpoklad A5, platí $x_i \to x^\star$ a $f(x^\star) = 0$.*

Důkaz Použitím (8) dostaneme

$$\begin{aligned} \|f_i\|(\|f_{i+1}\| - \|f_i\|) &\leq \frac{1}{2}(\|f_{i+1}\| + \|f_i\|)(\|f_{i+1}\| - \|f_i\|) = F_{i+1} - F_i \\ &\leq -2\underline{\rho}(1-\overline{\omega})\alpha_i F_i \leq -\underline{\rho}(1-\overline{\omega})\underline{\beta}^{\overline{j}_2}\|f_i\|^2. \end{aligned}$$

Tudíž

$$\|f_{i+1}\| \leq (1 - \underline{\rho}(1-\overline{\omega})\underline{\beta}^{\overline{j}_2})\|f_i\| \triangleq \overline{\lambda}\|f_i\|,$$

kde $0 < \overline{\lambda} < 1$, takže

$$\sum_{i=1}^{\infty} \|f_i\| = \frac{1}{1-\overline{\lambda}}\|f_1\| < \infty,$$

což implikuje $f_i \to 0$. Jestliže platí A5, pak $\|A_i s_i\| \geq \underline{A}\|s_i\|$, což dohromady s (11) dává

$$\sum_{i=1}^{\infty} \|x_{i+1} - x_i\| = \sum_{i=1}^{\infty} \alpha_i \|s_i\| \leq \frac{1+\overline{\omega}}{\underline{A}} \sum_{i=1}^{\infty} \|f_i\| < \infty$$

takže posloupnost x_i, $i \in \mathcal{N}$, je cauchyovská. Proto $x_i \to x^\star$, což dohromady s $f_i \to 0$ dává $f(x^\star) = 0$. □

Věta 2 *Nechť jsou splněny předpoklady A1 - A3, nechť $x_i \in \mathcal{L}(\overline{F})$, $A_i = J(x_i)$ a nechť je splněna podmínka D1. Pak existuje přirozené číslo $\overline{j}_2$, nezávislé na $i \in \mathcal{N}$ takové, že Armijův výběr D2 nalezne, po nejvýše $\overline{j}_2$ krocích, délku kroku $\alpha_i \geq \underline{\beta}^{\overline{j}_2}$ splňující (8) pro dané $0 < \rho < 1$.*

Důkaz (a) Předpokládejme, že neplatí (8) s $x_{i+1} = x_i + \alpha_i s_i$, čili

$$F(x_i + \alpha_i s_i) - F(x_i) > -2\underline{\rho}(1-\overline{\omega})\alpha_i F_i = -\underline{\rho}(1-\overline{\omega})\alpha_i\|f_i\|^2.$$

Použitím předpokladů A1, A2 a nerovnosti

$$g_i^T s_i = f_i^T (J_i s_i + f_i) - f_i^T f_i \leq -(1-\overline{\omega})\|f_i\|^2, \tag{13}$$

která platí když $A_i = J_i$, můžeme psát

$$\begin{aligned} F(x_i + \alpha_i s_i) - F(x_i) &= \alpha_i s_i^T g(x_i + \mu\alpha_i s_i) \\ &\leq \alpha_i \left(g_i^T s_i + \|s_i\| \|g(x_i + \mu\alpha_i s_i) - g(x_i)\| \right) \\ &\leq \alpha_i \left(-(1-\overline{\omega})\|f_i\|^2 + \alpha_i(\overline{J}^2 + \overline{LF})\|s_i\|^2 \right), \end{aligned}$$

kde $0 \leq \mu \leq 1$, neboť

$$\begin{aligned} \|g(x_i + \mu\alpha_i s_i) - g(x_i)\| &= \|J^T(x_i + \mu\alpha_i s_i) f(x_i + \mu\alpha_i s_i) - J^T(x_i) f(x_i)\| \\ &\leq \|J^T(x_i + \mu\alpha_i s_i)(f(x_i + \mu\alpha_i s_i) - f(x_i))\| \\ &\quad + \|(J^T(x_i + \mu\alpha_i s_i) - J^T(x_i)) f(x_i)\| \\ &\leq \overline{J}\|(f(x_i + \mu\alpha_i s_i) - f(x_i)\| + \overline{L}\mu\alpha_i \|s_i\| \|f_i\| \\ &= \overline{J}\| \int_0^1 J(x_i + \tau\mu\alpha_i s_i)\mu\alpha_i s_i d\tau\| + \overline{L}\mu\alpha_i \|s_i\| \|f_i\| \\ &\leq (\overline{J}^2 + \overline{LF})\alpha_i \|s_i\|. \end{aligned}$$

Spojením obou těchto nerovností a použitím vztahu (12), kde pokládáme $\overline{\vartheta} = 0$, dostaneme

$$(1 - \overline{\omega} - \underline{\rho}(1-\overline{\omega}))\|f_i\|^2 < \alpha_i(\overline{J}^2 + \overline{LF})\|s_i\|^2 \leq \alpha_i(\overline{J}^2 + \overline{LF}) \left(\frac{1+\overline{\omega}}{\underline{J}} \right)^2 \|f_i\|^2,$$

takže $\alpha_i > \underline{\alpha}$, kde

$$0 < \underline{\alpha} = \frac{(1-\underline{\rho})(1-\overline{\omega})\underline{J}^2}{(\overline{J}^2 + \overline{LF})(1+\overline{\omega})^2} < 1. \tag{14}$$

(b) Nechť $\overline{j}_2$ je nejmenší přirozené číslo takové, že $\overline{\beta}^{\overline{j}_2} \leq \underline{\alpha}$. Jelikož platí $\underline{\beta}^{\overline{j}_2} \leq \alpha_i \leq \overline{\beta}^{\overline{j}_2}$ po $\overline{j}_2$ Armijových krocích, musí podle části (a) důkazu po nejvýše $\overline{j}_2$ těchto krocích platit (8). □

Větu 2 můžeme bezprostředně použít na Algoritmus 1. Jelikož selhání v Kroku 4 může nastat pouze tehdy, když $A_i = J_i$, můžeme tento případ vyloučit volbou dostatečně velkého čísla $\overline{j}_2$, konkrétně $\overline{\beta}^{\overline{j}_2} \leq \underline{\alpha}$, kde hodnota $\underline{\alpha}$ je určena vztahem (14). Odhad (14) je obvykle zbytečně silný a Algoritmus 1 pracuje dobře s relativně malou hodnotou $\overline{j}_2 = 10$.

Nyní se zaměříme na některé detaily, které jsou nezbytné pro implementaci metod spádových směrů. Nejprve uvedeme několik poznámek týkajících se Algoritmu 1:

1) Matice J_i, $i \in \mathcal{N}$, vystupující v Kroku 3, mohou být vypočteny analyticky nebo pomocí numerického derivování. Obvykle se používá druhá možnost, neboť numerické derivování je pro velké řídké systémy velmi efektivní. Poznamenejme, že matice J_i,

$i \in \mathcal{N}$, nemusí být určovány explicitně, pokud k určení směrového vektoru používáme iterační metody nevyžadující transponovanou matici. V tomto případě dostaneme metody používající numerické derivování místo násobení vektoru Jacobiovou maticí.

2) Nerovnost $0 \leq \overline{\omega}_i \leq \overline{\omega} < 1$, použitou v Kroku 3, můžeme snadno splnit dosazením $\overline{\omega}_i = \overline{\omega}$. Nicméně pečlivý výběr hodnoty $\overline{\omega}_i$ může zvýšit účinnost metody. Doporučujeme použít hodnotu $\overline{\omega}_i = \min(\max(\|f_i\|^\nu, \gamma(\|f_i\|/\|f_{i-1}\|)^\alpha), 1/i, \overline{\omega})$, s $\nu = 1/2$, $\gamma = 1$, $\alpha = (1+\sqrt{5})/2$, která se osvědčila v numerických experimentech. Tato hodnota, která je kombinací hodnot uvedených v [12] a [15], zaručuje superlineární konvergenci metody, neboť $\overline{\omega}_i \to 0$ pro $i \to \infty$ (viz [11]).

3) V Kroku 4c doporučujeme použít experimentálně zjištěné hodnoty $\underline{j} = 1$, $\overline{j}_1 = 5$, $\overline{j}_2 = 10$. Hodnota $\underline{j}$ není důležitá pro důkaz konvergence, řídí však frekvenci restartů v případech, kdy metoda pomalu konverguje. Ze zkušenosti lze konstatovat, že větší hodnota $\underline{j}$ zvyšuje celkový čas výpočtu.

4) Hodnota $\underline{\beta}\alpha_i^j \leq \alpha_i^{j+1} \leq \overline{\beta}\alpha_i^j$, počítaná v Kroku 4c, může být určena pomocí konstantní redukce nebo pomocí složitějších metod jakými jsou kvadratická nebo kubická interpolace. Numerické experimenty ukazují, že konstantní redukce s $\underline{\beta} = \overline{\beta} = 1/2$ je vhodnou robustní strategií.

5) Pokud $\overline{k} = \infty$, pak přerušování iteračního procesu je podmíněno pouze nemožností vybrat vhodnou délku kroku. Konečná hodnota $\overline{k}$ je podstatná pro kvazinewtonovské metody s omezenou pamětí, které nemohou ukládat více než $O(\overline{k})$ vektorů. Položíme-li $\overline{k} = 0$, dostaneme diskrétní Newtonovu metodu.

Nyní se zaměříme na určení směrového vektoru. Vektor $s_i \in \mathcal{R}^n$, $i \in \mathcal{N}$, splňující nerovnost $\|A_i s_i + f_i\| \leq \overline{\omega}\|f_i\|$ se nejčastěji určuje jako přibližné řešení soustavy lineárních rovnic $A_i s + f_i = 0$ pomocí vhodné iterační metody. Abychom zjednodušili symboliku při popisu iteračních metod, budeme vynechávat index i, takže budeme psát A, f, x místo A_i, f_i, x_i. Místo toho budeme používat vnitřní index j při popisu iteračních metod pro řešení soustav lineárních rovnic. Abychom splnili podmínku $\|As + f\| \leq \overline{\omega}\|f\|$, pro libovolnou hodnotu $0 \leq \overline{\omega} < 1$, potřebujeme mít iterační metodu, která nalezne řešení po konečném počtu kroků. Numerické experimenty navíc ukazují, že je výhodné, když tyto metody generují posloupnost bodů s_j, $j \in \mathcal{N}$, a odpovídajících reziduálních vektorů $r_j = As_j + f$, $j \in \mathcal{N}$, tak, že normy $\|r_j\|$, $j \in \mathcal{N}$, nevzrůstají. Tento požadavek lze splnit pomocí metod minimalizujících rezidua nebo pomocí zhlazených Krylovovských metod. Navíc, protože matice A nebývá vždy známa, ale násobení touto maticí může být nahraženo numerickým derivováním, budeme uvažovat pouze iterační metody nevyžadující násobení transponovanou maticí.

Jednou z nejznámějších metod tohoto typu je metoda GMRES popsaná v[26]. Tato metoda však používá dlouhé rekurence ($O(n^3)$ operací a $O(n^2)$ paměťových míst bez použití restartů nebo $O(m^2 n)$ operací a $O(mn)$ paměťových míst při m-krokovém restartování), takže může být neefektivní pro řešení velkých řídkých úloh. Dobré výsledky lze získat pomocí předpodmíněné zhlazené popsané v [29] a realizované následujícím algoritmem.

Algoritmus 2. Předpodmíněná zhlazená metoda CGS.

Vypočteme $s = -C^{-1}f$ *a* $r = As + f$. *Jestliže* $\|r\| \leq \overline{\omega}\|f\|$, *ukončíme výpočet. V opačném případě položíme* $s_1 = 0,\ \overline{s}_1 = 0,\ r_1 = f,\ \overline{r}_1 = f,\ p_1 = f,\ u_1 = f.$

For $j = 1, 2, 3, \ldots$ **do**

Pokud $\|r_j\| \leq \overline{\omega}\|f\|$, *položíme* $s = s_j,\ r = r_j$ *a ukončíme výpočet. V opačném případě položíme*

$v_j = AC^{-1}p_j,\ \alpha_j = f^T\overline{r}_j / f^T v_j,$

$q_j = u_j - \alpha_j v_j,$

$\overline{s}_{j+1} = \overline{s}_j + \alpha_j C^{-1}(u_j + q_j),$

$\overline{r}_{j+1} = \overline{r}_j + \alpha_j AC^{-1}(u_j + q_j),\ \beta_j = f^T\overline{r}_{j+1} / f^T\overline{r}_j,$

$u_{j+1} = \overline{r}_{j+1} + \beta_j q_j,$

$p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j),$

$[\lambda_j, \mu_j]^T = \arg \min_{[\lambda,\mu]^T \in \mathcal{R}^2} \|\overline{r}_{j+1} + \lambda(r_j - \overline{r}_{j+1}) + \mu v_j\|,$

$s_{j+1} = \overline{s}_{j+1} + \lambda_j(s_j - \overline{s}_{j+1}) + \mu_j C^{-1}p_j,$

$r_{j+1} = \overline{r}_{j+1} + \lambda_j(r_j - \overline{r}_{j+1}) + \mu_j v_j.$

end do

Matice C slouží pro předpodmínění. Je výhodné použít neúplný LU rozklad matice $A + \varepsilon\mathrm{diag}(A)$ jako předpodmiňovač. Zde $\mathrm{diag}(A)$ je diagonální matice, která má stejné diagonální prvky jako matice A a $\varepsilon > 0$ je malé číslo. Jelikož používáme dvojnásobné zhlazení originální CGS metody [28], posloupnost norm reziduí je nerostoucí. Zhlazená metoda CGS používá krátké rekurence ($O(n)$ operací a paměťových míst v každé iteraci), ale může selhat z důvodu dělení nulou (break down), jestliže buď $f^T\overline{r}_j = 0$ nebo $f^T v_j = 0$. Řešení soustavy lineárních rovnic je nalezeno po nejvýše n iteracích (pokud nedojde k selhání a zaokrouhlovací chyby nezpomalí konvergenci). Poznamenejme, že k selhání dochází pouze výjimečně.

4 Přehled metod pro řešení nelineárních rovnic

V této části popíšeme třídu metod pro řešení systémů nelineárních rovnic, které lze realizovat Algoritmem 1. Tyto metody se navzájem liší pouze způsobem aproximace Jacobiovy matice $J(x)$. Protože potřebujeme znát Jacobiovu matici po každém přerušení, začneme popisem jejího výpočtu. Pro menší úlohy s hustou Jacobiovou maticí je nejvýhodnější počítat prvky této matice analyticky, neboť numerické derivování vyžaduje v tomto případě velký počet operací. Pro velké úlohy s řídkou Jacobiovou maticí se numerické derivování vyplácí, neboť vhodnou organizací výpočtů můžeme ušetřit velký počet operací. Při popisu numerického derivování použijeme označení $\mathcal{S}$ pro strukturu řídkosti matice $J(x)$, takže $(i, j) \in \mathcal{S}$ právě tehdy, pokud $J_{ij}(x) \neq 0$ (strukturálně).

Řídká Jacobiova matice může být získána dvěma různými způsoby. První způsob,

derivování po prvcích, je založen na aproximaci

$$J_{ij}(x) = \frac{f_i(x + \delta_j e_j) - f_i(x)}{\delta_j}, \tag{15}$$

použité pro všechny prvky s indexy $(i, j) \in \mathcal{S}$. Pak potřebujeme m skalárních funkčních výpočtů (čili m/n ekvivalentních vektorových funkčních výpočtů), kde m je počet nenulových prvků matice $J(x)$.

Druhý způsob, derivování po skupinách, je založen na rozdělení sloupců matice $J(x)$ do skupin $\mathcal{C}_k$, $1 \leq k \leq p$, tak aby každý prvek patřil pouze do jedné skupiny a aby navíc z $(i, j_1) \in \mathcal{S}$, $(i, j_2) \in \mathcal{S}$, $j_1 \neq j_2$ plynulo, že $j_1 \in \mathcal{C}_{k_1}$, $j_2 \in \mathcal{C}_{k_2}$, $k_1 \neq k_2$. Pak pro každou skupinu $\mathcal{C}_k$, $1 \leq k \leq p$, spočítáme diferenci $f(x + \sum_{j \in \mathcal{C}_k} \delta_j e_j) - f(x)$ a položíme

$$J_{ij}(x) = \frac{e_i^T (f(x + \sum_{j \in \mathcal{C}_k} \delta_j e_j) - f(x))}{\delta_j}, \tag{16}$$

pro všechny dvojice $(i, j) \in \mathcal{S} \cap \mathcal{C}_k$ (používáme označení $\mathcal{S} \cap \mathcal{C}_k = \{(i, j) \in \mathcal{S} : j \in \mathcal{C}_k\}$ a $\mathcal{S} \backslash \mathcal{C}_k = \{(i, j) \in \mathcal{S} : j \notin \mathcal{C}_k\}$). Potřebujeme tudíž p vektorových funkčních výpočtů. Jelikož počet skupin nemůže být menší než počet nenulových prvků v libovolném řádku, je počet vektorových funkčních výpočtů obvykle poněkud vyšší než v případě derivování po prvcích. Nicméně výpočty mohou být nyní lépe organizovány (výrazy, které jsou společné všem skalárním funkcím mohou být vyčísleny pouze p krát), takže derivování po skupinách je obvykle rychlejší. Derivování po skupinách je detailně popsáno v práci [10]. Optimální rozdělení sloupců do skupin a odpovídající problém barvení grafu jsou studovány v [8]. Účinná implementace výsledného algoritmu je předložena v [9].

Nyní popíšeme jednotlivé metody pro řešení systémů nelineárních rovnic. Označování se vztahuje k Algoritmu 1.

M1: Modifikovaná Newtonova metoda s přesnou (analyticky vypočtenou) Hessovou maticí (MNA). Tato metoda používá hodnotu $\overline{k} = 0$.

M2: Diskrétní Newtonova metoda s numericky vypočtenou Hessovou maticí (MND). Tato metoda používá hodnotu $\overline{k} = 0$.

M3: Broydenova dobrá metoda (BG). Tato metoda je popsána v [5]. Pokud $k < \overline{k}$, použijeme aktualizaci

$$A^+ = A + \frac{(y - Ad)d^T}{d^T d}$$

Zde $d = x^+ - x$ a $y = f(x^+) - f(x)$. Používáme hodnotu $\overline{k} = \infty$.

M4: Diskrétní Newtonova metoda s derivováním po prvcích (DNE). Tato metoda používá hodnotu $\overline{k} = 0$. Prvky $J(x)$ jsou počítány podle vzorce (15).

M5: Diskrétní Newtonova metoda s derivováním po skupinách (DNG). Tato metoda používá hodnotu $\overline{k} = 0$. Prvky $J(x)$ jsou počítány podle vzorce (16).

M6: Broydenova-Schubertova metoda (BS). Tato metoda je popsána v [6] and [27]. Pokud $k < \overline{k}$, použijeme aktualizaci

$$A_{ij}^{+} = A_{ij} + \frac{e_i^T(y - Ad)d_j}{\sum_{(i,k)\in\mathcal{S}} d_k^2}$$

$\forall(i,j) \in \mathcal{S}$. Zde $d = x^{+} - x$ a $y = f(x^{+}) - f(x)$. Zřejmě $A_{ij}^{+} = 0$, pokud $(i,j) \notin \mathcal{S}$. Používáme hodnotu $\overline{k} = \infty$.

M7: Bogleova-Perkinsova metoda (BP). Tato metoda je popsána v [1], Pokud $k < \overline{k}$, použijeme aktualizaci

$$A_{ij}^{+} = A_{ij} + \frac{e_i^T(y - Ad)A_{ij}^2 d_j}{\sum_{(i,k)\in\mathcal{S}} A_{ik}^2 d_k^2}$$

$\forall(i,j) \in \mathcal{S}$. Zde $d = x^{+} - x$ a $y = f(x^{+}) - f(x)$. Zřejmě $A_{ij}^{+} = 0$, pokud $(i,j) \notin \mathcal{S}$. Používáme hodnotu $\overline{k} = \infty$.

M8: Liova metoda (LI). Tato metoda, popsaná v [18], je založena na derivování po skupinách. Jestliže $k < \overline{k}$, pouze jedna skupina sloupců je aktualizována pomocí numerického derivování. Ostatní sloupce se nemění. Jinými slovy, položíme $l := l+1$, pokud $l < p$ a $l := 1$, pokud $l = p$ ($l = 0$ je počáteční hodnota) a pak dosadíme

$$A_{ij}^{+} = \frac{e_i^T(f(x + \sum_{j\in\mathcal{C}_l} \delta_j e_j) - f(x))}{\delta_j},$$

$\forall(i,j) \in \mathcal{S} \cap \mathcal{C}_l$ a

$$A_{ij}^{+} = A_{ij},$$

$\forall(i,j) \in \mathcal{S}\backslash\mathcal{C}_l$. Zřejmě $A_{ij}^{+} = 0$ if $(i,j) \notin \mathcal{S}$. Používáme hodnotu $\overline{k} = \infty$.

M9: Kombinace Liovy metody s Broydenovou-Schubertovou metodou (LIBS). Tato metoda je opět založena na derivování po skupinách. Jestliže $k < \overline{k}$, pouze jedna skupina sloupců je aktualizována pomocí numerického derivování. Ostatní sloupce jsou aktualizovány pomocí Broydenova-Schubertova algoritmu. Jinými slovy, položíme $l := l + 1$, pokud $l < p$ a $l := 1$, pokud $l = p$ ($l = 0$ je počáteční hodnota) a pak dosadíme

$$A_{ij}^{+} = \frac{e_i^T(f(x + \sum_{j\in\mathcal{C}_l} \delta_j e_j) - f(x))}{\delta_j},$$

$\forall(i,j) \in \mathcal{S} \cap \mathcal{C}_l$ a

$$A_{ij}^{+} = A_{ij} + \frac{e_i^T(y - Ad)d_j}{\sum_{(i,k)\in\mathcal{S}\backslash\mathcal{C}_l} d_k^2},$$

$\forall(i,j) \in \mathcal{S}\backslash\mathcal{C}_l$. Zde $d = x^+ - x$ a $y = f(x^+) - f(x)$. Zřejmě $A^+_{ij} = 0$, pokud $(i,j) \notin \mathcal{S}$. Používáme hodnotu $\overline{k} = \infty$.

M10: Modifikovaná Newtonova metoda (MN) Položíme $A^+ := A$, pokud $k < \overline{k}$. Tato metoda vyžaduje konečnou hodnotu $\overline{k}$. Hodnota $\overline{k} = 5$ byla získána experimentálně.

M11: Metoda škálování řádků (RS). Položíme $A^+ := DA$, pokud $k < \overline{k}$, kde diagonální matice D je určena z kvazinewtonovské podmínky $DAd = y$, čili

$$e_i^T D e_i = \frac{e_i^T y}{e_i^T A d}$$

pro $1 \leq i \leq n$ (zde $d = x^+ - x$ a $y = f(x^+) - f(x)$). Metoda škálování řádků uvedená v [16] používá kompletní LU rozklad. Tato metoda vyžaduje konečnou hodnotu $\overline{k}$. Hodnota $\overline{k} = 5$ byla získána experimentálně.

M12: Broydenova metoda s omezenou pamětí (LMB). Tato metoda je modifikací Broydenovy metody popsané v [5] a je založena na kompaktní reprezentaci kvazinewtonovské matice uvedené v [7]. Označme $D = [d, d_{-1}, \ldots, d_{-k}]$ a $Y = [y, y_{-1}, \ldots, y_{-k}]$ matice zkonstruované z posledních k differencí $d = x^+ - x$, $d_{-1} = x - x_{-1}$, …, $d_{-k} = x_{1-k} - x_{-k}$ a $y = f(x^+) - f(x)$, $y_{-1} = f(x) - f(x_{-1})$, …, $y_{-k} = f(x_{1-k}) - f(x_{-k})$, a definujme horní trojúhelníkovou matici

$$R = \begin{bmatrix} d^T d, & d^T d_{-1}, & \ldots, & d^T d_{-k} \\ 0, & d_{-1}^T d_{-1} & \ldots, & d_{-1}^T d_{-k} \\ \ldots, & \ldots, & \ldots, & \ldots \\ 0, & 0, & \ldots, & d_{-k}^T d_{-k} \end{bmatrix}.$$

Pak jestliže $k < \overline{k}$, položíme

$$A^+ = A_{-k} + (Y - A_{-k}D)R^{-1}D^T.$$

Broydenova metoda s omezenou pamětí vyžaduje konečnou hodnotu $\overline{k}$. Hodnota $\overline{k} = 5$ byla získána experimentálně.

M13: Metoda aktualizace sloupců s omezenou pamětí (LMC). Tato metoda je modifikací metody aktualizace sloupců popsané v [24] a je založena na kompaktní reprezentaci kvazinewtonovské matice uvedené v [7]. Nechť D a Y jsou matice použité v předchozím případě. Označme $e = \arg\max_{e_i} |e_i^T d|$, $e_{-1} = \arg\max_{e_i} |e_i^T d_{-1}|$, …, $e_{-k} = \arg\max_{e_i} |e_i^T d_{-k}|$ (arg max počítáme ze všech e_i, $1 \leq i \leq n$) položíme $E = [e, e_{-1}, \ldots, e_{-k}]$ a definujeme horní trojúhelníkovou matici

$$R = \begin{bmatrix} e^T d, & e^T d_{-1}, & \ldots, & e^T d_{-k} \\ 0, & e_{-1}^T d_{-1} & \ldots, & e_{-1}^T d_{-k} \\ \ldots, & \ldots, & \ldots, & \ldots \\ 0, & 0, & \ldots, & e_{-k}^T d_{-k} \end{bmatrix}.$$

Pak jestliže $k < \overline{k}$, položíme

$$A^+ = A_{-k} + (Y - A_{-k}D)R^{-1}E^T$$

Poznamenejme, že vektory e, e_{-1}, ..., e_{-k} není třeba ukládat. Používáme pouze jejich jediné nenulové prvky. Metoda aktualizace sloupců s omezenou pamětí vyžaduje konečnou hodnotu $\overline{k}$. Hodnota $\overline{k} = 5$ byla získána experimentálně.

M14: Inverzní metoda aktualizace sloupců s omezenou pamětí (LMI). Tato metoda, popsaná v [25], používá aproximaci $S = A^{-1}$ inverzní Jacobiovy matice $J^{-1}(x)$. Pokud $k < \overline{k}$, položíme jednoduše $s := -Sf$ místo toho, abychom použili Algoritmus 2. Označme $e_{-1} = \arg\max_{e_i} |e_i^T y_{-1}|$, ..., $e_{-k} = \arg\max_{e_i} |e_i^T y_{-k}|$ (arg max počítáme ze všech e_i, $1 \le i \le n$). Pak vektor Sf může být spočítán podle vzorce

$$Sf = S_{-k}f + \frac{e_{-1}^T f}{e_{-1}^T y_{-1}} v_{-1} + \ldots + \frac{e_{-k}^T f}{e_{-k}^T y_{-k}} v_{-k},$$

kde $v_{-1} = d_{-1} - S_{-1}y_{-1}$, ..., $v_{-k} = d_{-k} - S_{-k}y_{-k}$. Tyto vektory lze počítat rekurentně podle vzorce

$$Sy = S_{-k}y + \frac{e_{-1}^T y}{e_{-1}^T y_{-1}} v_{-1} + \ldots + \frac{e_{-k}^T y}{e_{-k}^T y_{-k}} v_{-k}.$$

Oba tyto vzorce používají matici $S_{-k} = (L_{-k}U_{-k})^{-1}$, kde $L_{-k}U_{-k}$ je neúplný LU rozklad Jacobiovy matice $J(x_{-k})$. Poznamenejme, že vektory e_{-1}, ..., e_{-k} není třeba ukládat. Používáme pouze jejich jediné nenulové prvky. Inverzní metoda aktualizace sloupců s omezenou pamětí vyžaduje konečnou hodnotu $\overline{k}$. Hodnota $\overline{k} = 6$ byla získána experimentálně.

M15: Diskrétní Newtonova methoda s postupným výpočtem směrových derivací (DNS). Tato methoda, uvedená v in [19], nepoužívá Jacobiovu matici. Součiny $Av = Jv$, užité v Algoritmu 2, jsou nahraženy numerickým derivováním

$$Av = \frac{f(x + v\delta/\|v\|) - f(x)}{\delta/\|v\|},$$

kde δ je malá diference (obvykle $\delta = 10^{-8}$ ve dvojnásobné strojové aritmetice). Jelikož Jacobiova matice se explicitně nepočítá, nemůžeme použít neúplný LU rozklad jako předpodmiňovač. Místo toho numericky počítáme tridiagonální část Jacobiovy matice pomocí diferencí a potom aplikujeme tuto tridiagonální matici jako předpodmiňovač.

Protože úlohy chemické rovnováhy neobsahují příliš velký počet rovnic a neznámých, použili jsme metody M1 a M2 implementované podprogramem `PNEQU` s `IDER=0` a `IDER=1` a metodu M3 implementovanou podprogramem `PBEQU`. Tyto podprogramy jsou popsány v dalším oddílu.

5 Implementace vybraných metod a popis podprogramů

V tomto oddílu popíšeme podprogramy PNEQU a PBEQU, které lze snadno volat z hlavního programu sestaveného uživatelem. Při popisu formálních parametrů bude uveden typ parametru složený ze dvou písmen. První písmeno je buď I pro celočíselný argument nebo R pro reálný parametr (ve dvojnásobné strojové přesnosti). Druhé písmeno specifikuje, jde-li o vstupní hodnotu I, výstupní hodnotu O, aktualizovanou (vstupní a výstupní) hodnotu U, nebo pomocnou hodnotu A. Kromě formálních parametrů používáme blok COMMON /STAT/, obsahující statistické údaje. Tento blok, použitý v každém podprogramu, má jednotnou formu:

```
COMMON /STAT/ NRES,NDEC,NIIT,NIT,NFV,NFG,NFH
```

Jeho prvky mají následující význam:

Prvek	Typ	Význam
NRES	IO	Počet restartů.
NDEC	IO	Počet maticových rozkladů.
NIIT	IO	Počet vnitřních iterací (pro řešení soustav lineárních rovnic).
NIT	IO	Počet vnějších iterací.
NFV	IO	Počet vyčíslení hodnot účelové funkce.
NFG	IO	Počet vyčíslení gradientů.
NFH	IO	Počet vyčíslení Hessových matic.

Podprogramy PNEQU a PBEQU se vyvolávají pomocí příkazů:

```
CALL PNEQU(N,X,AF,RA,NRA,IPAR,RPAR,F,GMAX,ITERM)
CALL PBEQU(N,X,AF,RA,NRA,IPAR,RPAR,F,GMAX,ITERM)
```

Význam formálních parametrů:

Parametr	Typ	Význam
N	II	Počet neznámých a současně počet nelineárních rovnic.
X(N)	RU	Na vstupu počáteční odhad řešení. Na výstupu získaná aproximace řešení.
AF(N)	RO	Vektor obsahující rezidua (přesnosti) jednotlivých nelineárních rovnic.
RA(NRA)	RA	Reálné pracovní pole dimenze NRA.
IPAR(4)	IA	Pole celočíselných parametrů v pořadí IPRNT, MIT, MFV, IDER.
RPAR(6)	RA	Pole reálných parametrů v pořadí XMAX, TOLX, TOLF, TOLB, TOLG, XDEL
F	RO	Hodnota účelové funkce (přesnost řešení). X.

GMAX	RO	Maximální absolutní hodnota parciální derivace účelové funkce.
ITERM	IU	Proměnná udávající způsob ukončení výpočtu:

ITERM = 1: Hodnota $|\mathtt{X} - \mathtt{X}_{\mathtt{old}}|$ nebyla větší než TOLX ve dvou po sobě následujících iteracích,
ITERM = 2: Hodnota $|\mathtt{F} - \mathtt{F}_{\mathtt{old}}|$ nebyla větší než TOLF ve dvou po sobě následujících iteracích,
ITERM = 3: Hodnota F nebyla větší než TOLB,
ITERM = 4: Hodnota GMAX nebyla větší než TOLG,
ITERM = 11: Hodnota NIT převýšila MIT,
ITERM = 12: Hodnota NFV převýšila MFV,
ITERM = 13: Hodnota NFG převýšila MFG,
ITERM < 0: Metoda selhala.

Dimeze NRA pracovního pole RA musí splňovat nerovnost NRA $\geq$ (N+6)*N+N*(N+1)/2. Pokud hodnota NRA není dostatečně velká, vytiskne se zpráva udávající požadovanou hodnotu a ukončí se výpočet. Tato standardní akce může být potlačena vstupní volbou ITERM < 0.

Podprogramy PNEQU a PBEQU vyvolávají externí podprogramy FUN a DFUN (sestavené uživatelem), které definují funkci $f(x)$ a Jacobiovu matici $J(x)$ a které mají tvar

```
SUBROUTINE  FUN(N,KA,X,FA)
SUBROUTINE DFUN(N,KA,X,GA)
```

Externí podprogram DFUN se vyvolává pouze v podprogramu PNEQU a to pouze tehdy, pokud IDER = 1.

Význam parametrů externích podprogramů:

Argument	Type	Significance
N	II	Počet neznámých a současně počet nelineárních rovnic.
KA	II	Pořadové číslo nelineární rovnice.
X(NF)	RI	Vektor neznámých (odhad řešení).
FA	RO	Reziduum (chyba) KA-té rovnice v bodě X.
GA(NF)	RO	Gradient of the KA-té particular function at the point X.

Podprogramy PNEQU a PBEQU jsou velmi jednoduché, vyvolávají však obecné podprogramy PNEQ a PBEQ pro řešení nelineárních systémů, které jsou odpovídajícími implementacemi Algoritmu 1. Podprogramy PNEQ a PBEQ se vyvolávají pomocí příkazů:

```
 CALL PNEQ(N,X,GA,AG,G,S,XO,GO,H,AF,AFO,XMAX,TOLX,TOLF,TOLB,TOLG,
& XDEL,GMAX,F,IPRNT,MIT,MFV,IDER,ITERM)
 CALL PBEQ(N,X,GA,AG,G,S,XO,GO,H,AF,AFO,XMAX,TOLX,TOLF,TOLB,TOLG,
& XDEL,GMAX,F,IPRNT,MIT,MFV,ITERM)
```

Parametry N, X, AF, GMAX, F, ITERM mají stejný význam jako v podprogramech `PNEQU` a `PBEQU`. Význam ostatních parametrů:

Parametr	Typ	Význam
`GA(N)`	`A`	Gradient funkce definující nelineární rovnici (řadek Jacobiovy matice).
`AG(N*N)`	`A`	Jacobiova matice uložená po řádcích.
`G(N)`	`A`	Gradient účelové funkce.
`S(N)`	`A`	Směrový vektor.
`XO(N)`	`A`	Diference dvou po sobě jdoucích aproximací řešení.
`GO(N)`	`A`	Diference dvou po sobě jdoucích gradientů.
`H(M)`	`A`	Pracovní symetrická matice uložená úsporným způsobem.
`AFO(N)`	`A`	Vektor obsahující stará rezidua jednotlivých nelineárních rovnic.
`IPRNT`	`II`	Specifikace tisku: `IPRNT` $= 0$: tisk je potlačen, `IPRNT` $= 1$: základní úroveň tisku konečných výsledků, `IPRNT` $= -1$: rozšířená úroveň tisku konečných výsledků, `IPRNT` $= 2$: základní úroveň tisku mezivýsledků a konečných výsledků, `IPRNT` $= -2$: rozšířená úroveň tisku mezivýsledků a konečných výsledků.
`MIT`	`II`	Maximální počet iterací; volba `MIT` $= 0$ způsobí, že se použije doporučená hodnota `MIT` $= 9000$.
`MFV`	`II`	Maximální počet vyčíslení hodnoty účelové funkce; volba `MFV` $= 0$ způsobí, že se použije doporučená hodnota `MIT` $= 9000$.
`IDER`	`II`	Způsob výpočtu Jacobiovy matice: `IDER` $= 0$: Jacobiova matice se počítá numericky, `IDER` $= 1$: Jacobiova matice se počítá analyticky,
`XMAX`	`RI`	Maximální délka kroku; volba `XMAX` $= 0$ způsobí, že se použije doporučená hodnota `XMAX` $= 10^3$.
`TOLX`	`RI`	Tolerance pro změnu prvků vektoru `X`; volba `TOLX` $= 0$ způsobí, že se použije doporučená hodnota `TOLX` $= 10^{-16}$.
`TOLF`	`RI`	Tolerance pro změnu hodnoty účelové funkce; volba `TOLF` $= 0$ způsobí, že se použije doporučená hodnota `TOLF` $= 10^{-16}$.
`TOLB`	`RI`	Minimální hodnota účelové funkce; volba `TOLB` $= 0$ způsobí, že se použije doporučená hodnota `TOLX` $= 10^{-16}$.
`TOLG`	`RI`	Tolerance pro normu gradientu; volba `TOLG` $= 0$ způsobí, že se použije doporučená hodnota `TOLG` $= 10^{-5}$.
`XDEL`	`RI`	Počáteční poloměr oblasti přijatelnosti; volba `XDEL` $= 0$ způsobí, že `XDEL` $= 0$ vypočte z počáteční funkční hodnoty a gradientu účelové funkce.

6 Výpisy podprogramů

6.1 Podrogramy pro řešení systémů nelineárních rovnic

```
* SUBROUTINE PNEQU              ALL SYSTEMS                  05/01/22
* PURPOSE :
* EASY TO USE SUBROUTINE FOR SOLUTION OF DENSE SYSTEMS OF NONLINEAR
* EQUATIONS USING THE NEWTON METHOD.
*
* PARAMETERS :
*  II  N  NUMBER OF VARIABLES.
*  RI  X(N)  VECTOR OF VARIABLES.
*  RO  AF(N)  VECTOR CONTAINING VALUES OF THE APPROXIMATED
*        FUNCTIONS.
*  RA  RA((N+6)*N+N*(N+1)/2)  AUXILIARY ARRAY.
*  II  NRA  DIMENSION OF THE ARRAY RA.
*  II  IPAR(4)  INTEGER PAREMETERS:
*      IPAR(1)  PRINT SPECIFICATION. IPAR(1)=0-NO PRINT.
*        ABS(IPAR(1))=1-PRINT OF FINAL RESULTS.
*        ABS(IPAR(1))=2-PRINT OF FINAL RESULTS AND ITERATIONS.
*        IPAR(1)>0-BASIC FINAL RESULTS. IPAR(1)<0-EXTENDED FINAL
*        RESULTS.
*      IPAR(2)  MAXIMUM NUMBER OF ITERATIONS.
*      IPAR(3)  MAXIMUM NUMBER OF FUNCTION EVALUATIONS.
*      IPAR(4)  DEGREE OF ANALYTICALLY COMPUTED DERIVATIVES (0 OR 1).
*  RI  RPAR(6)  REAL PARAMETERS:
*      RPAR(1)  MAXIMUM STEPSIZE.
*      RPAR(2)  TOLERANCE FOR CHANGE OF VARIABLES.
*      RPAR(3)  TOLERANCE FOR CHANGE OF FUNCTION VALUES.
*      RPAR(4)  TOLERANCE FOR THE FUNCTION FALUE.
*      RPAR(5)  TOLERANCE FOR THE GRADIENT NORM.
*      RPAR(6)  TRUST REGION STEPSIZE.
*  RO  F  VALUE OF THE OBJECTIVE FUNCTION.
*  RO  GMAX  MAXIMUM PARTIAL DERIVATIVE.
*  IO  ITERM  CAUSE OF TERMINATION.
*
* VARIABLES IN COMMON /STAT/ (STATISTICS) :
*  IO  NRES  NUMBER OF RESTARTS.
*  IO  NDEC  NUMBER OF MATRIX DECOMPOSITION.
*  IO  NIIT  NUMBER OF INNER ITERATIONS.
*  IO  NIT  NUMBER OF ITERATIONS.
*  IO  NFV  NUMBER OF FUNCTION EVALUATIONS.
*  IO  NFG  NUMBER OF GRADIENT EVALUATIONS.
*  IO  NFH  NUMBER OF HESSIAN EVALUATIONS.
*
* SUBPROGRAMS USED :
*  S   PNEQ  SOLUTION OF DENSE NONLINEAR SYSTEMS OF EQUATIONS BY THE
*        NEWTON METHOD.
*
* EXTERNAL SUBROUTINES :
*  SE  FUN  COMPUTATION OF THE VALUE OF THE APPROXIMATED FUNCTION.
*        CALLING SEQUENCE: CALL FUN(N,KA,X,FA) WHERE N IS THE NUMBER
*        OF VARIABLES, KA IS THE INDEX OF THE APPROXIMATED FUNCTION,
*        X(N) IS A VECTOR OF VARIABLES AND FA IS THE VALUE OF THE
*        APPROXIMATED FUNCTION.
*  SE  DFUN  COMPUTATION OF THE GRADIENT OF THE APPROXIMATED FUNCTION.
*        CALLING SEQUENCE: CALL DFUN(N,KA,X,GA) WHERE N IS THE NUMBER
*        OF VARIABLES, KA IS THE INDEX OF THE APPROXIMATED FUNCTION,
*        X(N) IS A VECTOR OF VARIABLES AND GA IS THE GRADIENT OF THE
*        APPROXIMATED FUNCTION.
*
      SUBROUTINE PNEQU(N,X,AF,RA,NRA,IPAR,RPAR,F,GMAX,ITERM)
      DOUBLE PRECISION F,GMAX
      INTEGER          ITERM,N
```

```
      DOUBLE PRECISION AF(*),RA(*),RPAR(6),X(*)
      INTEGER          NRA,IPAR(4)
      INTEGER          NRES,NDEC,NIIT,NIT,NFV,NFG,NFH
      INTEGER          LAFO,LAG,LG,LGA,LGO,LH,LXO,LS,M,MMAXR
      EXTERNAL         PNEQ
      COMMON           /STAT/NRES,NDEC,NIIT,NIT,NFV,NFG,NFH
*
*     POINTERS FOR AUXILIARY ARRAYS
*
      LGA = 1
      LAG = LGA + N
      LG = LAG + N * N
      LS = LG + N
      LXO = LS + N
      LGO = LXO + N
      LAFO = LGO + N
      LH = LAFO + N
      MMAXR=NRA-(N+6)*N-N*(N+1)/2
      IF (MMAXR.LT.0) THEN
      WRITE(6,*)
     & 'LACK OF SPACE IN THE ARRAY RA - MINIMUM REQUIRED SPACE:',
     & (N+6)*N+N*(N+1)/2
      ENDIF
      IF (MMAXR.LT.0) STOP
      CALL PNEQ(N,X,RA(LGA),RA(LAG),RA(LG),RA(LS),RA(LXO),RA(LGO),
     &          RA(LH),AF,RA(LAFO),RPAR(1),RPAR(2),RPAR(3),RPAR(4),
     &          RPAR(5),RPAR(6),GMAX,F,IPAR(1),IPAR(2),IPAR(3),
     &          IPAR(4),ITERM)
      RETURN
      END

* SUBROUTINE PNEQ                  ALL SYSTEMS               95/12/01
* PURPOSE :
* SOLUTION OF SPARSE NONLINEAR SYSTEMS OF EQUATIONS BY THE NEWTON
* METHOD USING THE PRECONDITIONED SMOOTHED CGS SUBALGORITHM FOR
* ITERATIVE SOLUTION OF LINEARIZED SYSTEMS.
*
* PARAMETERS :
*  II  N  NUMBER OF VARIABLES.
*  RI  X(N)  VECTOR OF VARIABLES.
*  RA  GA(N)  GRADIENT OF THE APPROXIMATED FUNCTION.
*  RA  AG(N*N)  RECTANGULAR MATRIX WHICH IS USED FOR THE DIRECTION
*        VECTOR DETERMINATION.
*  RA  G(N)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  RA  S(N)  DIRECTION VECTOR.
*  RA  XO(N)  AUXILIARY VECTOR.
*  RA  GO(N)  AUXILIARY VECTOR.
*  RA  H(N*(N+1)/2)  AUXILIARY MATRIX.
*  RO  AF(N)  VECTOR WHOSE ELEMENTS ARE VALUES OF THE APPROXIMATED
*        FUNCTIONS.
*  RA  AFO(N)  AUXILIARY VECTOR.
*  RI  XMAX  MAXIMUM STEPSIZE.
*  RI  TOLX  TOLERANCE FOR CHANGE OF VARIABLES.
*  RI  TOLF  TOLERANCE FOR CHANGE OF FUNCTION VALUES.
*  RI  TOLB  TOLERANCE FOR THE FUNCTION FALUE.
*  RI  TOLG  TOLERANCE FOR THE GRADIENT OF THE LAGRANGIAN FUNCTION.
*  RI  XDEL  TRUST REGION STEPSIZE.
*  RO  GMAX  MAXIMUM PARTIAL DERIVATIVE.
*  RO  F  VALUE OF THE OBJECTIVE FUNCTION.
*  II  IPRNT  PRINT SPECIFICATION. IPRNT=0-NO PRINT.
*        ABS(IPRNT)=1-PRINT OF FINAL RESULTS.
*        ABS(IPRNT)=2-PRINT OF FINAL RESULTS AND ITERATIONS.
*        IPRNT>0-BASIC FINAL RESULTS. IPRNT<0-EXTENDED FINAL
*        RESULTS.
*  II  MIT  MAXIMUN NUMBER OF ITERATIONS.
```

```
*  II  MFV  MAXIMUN NUMBER OF FUNCTION EVALUATIONS.
*  II  IDER  DEGREE OF ANALYTICALLY COMPUTED DERIVATIVES (0 OR 1).
*  IO  ITERM  CAUSE OF TERMINATION.
*
* VARIABLES IN COMMON /STAT/ (STATISTICS) :
*  IO  NRES  NUMBER OF RESTARTS.
*  IO  NDEC  NUMBER OF MATRIX DECOMPOSITION.
*  IO  NIIT  NUMBER OF INNER ITERATIONS.
*  IO  NIT  NUMBER OF ITERATIONS.
*  IO  NFV  NUMBER OF FUNCTION EVALUATIONS.
*  IO  NFG  NUMBER OF GRADIENT EVALUATIONS.
*  IO  NFH  NUMBER OF HESSIAN EVALUATIONS.
*
* SUBPROGRAMS USED :
*  S   PA1SQ1  COMPUTATION OF THE VALUE AND THE GRADIENT OF THE
*         OBJECTIVE FUNCTION WHICH IS DEFINED AS A SUM OF SQUARES
*         OF THE APPROXIMATED FUNCTIONS.
*  S   PNSTEP  VECTOR ON THE BOUNDARY OF THE TRUST REGION.
*  S   PSOGO1  TRUST REGION STEPSIZE USING ONLY FUNCTION VALUES.
*  S   PYFUT1  TEST ON TERMINATION.
*  S   PYTRUD  COMPUTATION OF THE VARIABLE VECTOR AND THE GRADIENT
*         DIFFERENCES.
*  S   MXDCMM  MATRIX-VECTOR PRODUCT. RECTANGULAR MATRIX IS STORED
*         COLUMNWISE.
*  S   MXDPRB  BACK SUBSTITUTION USING THE CHOLESKI DECOMPOSITION
*         OBTAINED BY MXDPRF.
*  S   MXDPRC  CORRECTION OF THE CHOLESKI DECOMPOSITION OBTAINED BY
*         MXDPRF.
*  S   MXDPRM  MATRIX-VECTOR PRODUCT USING THE CHOLESKI DECOMPOSITION
*         OBTAINED BY MXDPRF.
*  S   MXDRMM  MATRIX-VECTOR PRODUCT. RECTANGULAR MATRIX IS STORED
*         ROWWISE.
*  S   MXDRQF  QR DECOMPOSITION OF ROWWISE STORED RECTANGULAR MATRIX
*         USING HOUSEHOLDER TRANSFORMATIONS WITHOUT PIVOTING.
*  S   MXVCOP  COPYING OF A VECTOR.
*  S   MXVDIF  DIFFERENCE OF TWO VECTORS.
*  S   MXVDIR  VECTOR AUGMENTED BY THE SCALED VECTOR.
*  RF  MXVDOT  DOT PRODUCT OF TWO VECTORS.
*  S   MXVNEG  COPYING OF A VECTOR WITH CHANGE OF THE SIGN.
*
* EXTERNAL SUBROUTINES :
*  SE  FUN  COMPUTATION OF THE VALUE OF THE APPROXIMATED FUNCTION.
*         CALLING SEQUENCE: CALL FUN(N,KA,X,FA) WHERE N IS A NUMBER
*         OF VARIABLES, KA IS THE INDEX OF THE APPROXIMATED FUNCTION,
*         X(N) IS A VECTOR OF VARIABLES AND FA IS THE VALUE OF THE
*         APPROXIMATED FUNCTION.
*  SE  DFUN  COMPUTATION OF THE GRADIENT OF THE APPROXIMATED FUNCTION.
*         CALLING SEQUENCE: CALL DFUN(N,KA,X,GA) WHERE N IS THE NUMBER
*         OF VARIABLES, KA IS THE INDEX OF THE APPROXIMATED FUNCTION,
*         X(N) IS A VECTOR OF VARIABLES AND GA IS THE GRADIENT OF THE
*         APPROXIMATED FUNCTION.
*
* METHOD :
* MODIFIED ORIGINAL OR DISCRETE NEWTON METHOD.
*
      SUBROUTINE PNEQ(N,X,GA,AG,G,S,XO,GO,H,AF,AFO,XMAX,TOLX,TOLF,
     &                TOLB,TOLG,XDEL,GMAX,F,IPRNT,MIT,MFV,IDER,ITERM)
      DOUBLE PRECISION F,GMAX,TOLB,TOLF,TOLG,TOLX,XMAX,XDEL
      INTEGER          IDER,IPRNT,ITERM,MFV,MIT,N
      DOUBLE PRECISION AF(*),AFO(*),AG(*),G(*),GA(*),GO(*),H(*),
     &                 S(*),X(*),XO(*)
      INTEGER          NRES,NDEC,NIIT,NIT,NFV,NFG,NFH
      DOUBLE PRECISION BET1,BET2,GAM1,GAM2,DMAX,EPS4,EPS5,
     &                 ETA0,ETA2,ETA9,FMIN,FO,FP,GNORM,P,PO,PP,R,
     &                 RMAX,RO,SNORM,TEMP,B1,B2,B3,D3,S1,S2,
```

```
     &                  XDELO,UMAX,T,
     &                  MXVDOT
      INTEGER          I,INF,IREST,ITERD,ITERS,NRED,
     &                  KD,KIT,LD,MA,MD,MRED,MTESF,MTESX,NTESF,NTESX,LDS,
     &                  IDEC,INITS,KTERS,IEST,ITES,MAXST,IRES1,IRES2,MM,
     &                  ISYS,MFG,INITD,MES1,MES2,MES3,MOT3,MOS1,KDA,IDIR
      COMMON           /STAT/NRES,NDEC,NIIT,NIT,NFV,NFG,NFH
      IF (ABS(IPRNT).GT.1) WRITE (6,FMT='(1X,''ENTRY TO PNEQ :'')')
*
*     INITIATION
*
      NRES=0
      NDEC=0
      NIIT=0
      NIT=0
      NFV=0
      NFG=0
      NFH=0
      ISYS = 0
      IEST = 1
      ITES = 1
      MTESX = 2
      MTESF = 2
      INITS = 1
      ITERM = 0
      ITERD = 0
      ITERS = 2
      KTERS = 0
      IREST = 1
      IRES1=999
      IRES2 = 0
      MRED = 5
      IDEC = 0
      EPS4=0.10D 0
      EPS5=0.90D 0
      ETA0 = 1.0D-15
      ETA2 = 1.0D-18
      ETA9 = 1.0D 60
      BET1=0.05D 0
      BET2=0.75D 0
      GAM1=2.0D 0
      GAM2=1.0D 6
      FMIN = 0.0D 0
      IF (TOLX.LE.0.0D0) TOLX = 1.0D-16
      IF (TOLF.LE.0.0D0) TOLF = 1.0D-16
      IF (TOLB.LE.0.0D0) TOLB = 1.0D-16
      IF (TOLG.LE.0.0D0) TOLG = 1.0D-16
      IF (XMAX.LE.0.0D0) XMAX = 1.0D 3
      IF (MIT.LE.0) MIT = 1000
      IF (MFV.LE.0) THEN
      IF (IDER.EQ.0) MFV = 5000
      IF (IDER.EQ.1) MFV = 2000
      ENDIF
      MFG = 2*MIT
      MES1 = 3
      MES2 = 2
      MES3 = 1
      MOT3 = 0
      KDA = IDER
      KD =  0
      LD = -1
      KIT = 0
      FO = FMIN
      GMAX = ETA9
      DMAX = ETA9
```

```
      T=0.0D 0
      FO=FMIN
*
*     -----------------
*     MODEL DESCRIPTION
*     -----------------
*
      CALL PA1SQ1(N,X,F,AF,GA,AG,G,ETA0,KDA,KD,LD,NFV,NFG)
*
*     ------------------------
*     END OF MODEL DESCRIPTION
*     ------------------------
*
11020 CONTINUE
      IF (ABS(IPRNT).GT.1)
     & WRITE (6,'(1X,''NIT='',I5,2X,''NFV='',I5,2X,''NFG='',I5,2X,
     & ''F ='', D15.8)') NIT,NFV,NFG,F
      CALL PYFUT1(N,F,FO,UMAX,GMAX,DMAX,TOLX,TOLF,TOLB,TOLG,KD,
     & NIT,KIT,MIT,NFV,MFV,NFG,MFG,NTESX,MTESX,NTESF,MTESF,ITES,IRES1,
     & IRES2,IREST,ITERS,ITERM)
      IF(ITERM.NE.0) GOTO 11090
11030 CONTINUE
      IF (IREST.LE.0) GO TO 12320
      KD=1
      CALL PA1SQ1(N,X,F,AF,GA,AG,G,ETA0,KDA,KD,LD,NFV,NFG)
      IDEC=0
      IF (IDIR.EQ.0) THEN
      IF (KIT.LT.NIT) THEN
      NRES=NRES+1
      KIT = NIT
      ELSE
      ITERM=-10
      IF (ITERS.LT.0) ITERM=ITERS-5
      ENDIF
      ENDIF
12320 CONTINUE
      IF(ITERM.NE.0) GOTO 11090
*
*     ------------------------
*     DIRECTION DETERMINATION
*     TEMPLATE : UDGGA1
*     ------------------------
*
      IF (IDEC.LT.0) THEN
      IDEC=1
      INF=0
      ENDIF
      IF (IDEC.EQ.0) THEN
      ELSEIF(IDEC.EQ.1) THEN
      ELSE
      ITERD=-1
      GO TO 12430
      ENDIF
      IF (LD.LE.0) THEN
      CALL MXDCMM(N,N,AG,AF,G)
      IF (IDEC.EQ.1) THEN
      CALL MXVCOP(N,G,GO)
      CALL MXDPRM(N,H,G,-1)
      ENDIF
      ELSE IF (IDEC.EQ.1) THEN
      CALL MXDCMM(N,N,AG,AF,GO)
      ENDIF
      B2=MXVDOT(N,G,G)
      GNORM=SQRT(B2)
      IF (KD.LE.0) THEN
```

```
      IF (GNORM.LE.TOLG) THEN
      ITERM=4
      GO TO 12430
      ENDIF
      ENDIF
      INITD=MAX(ABS(INITD),1)
      IF (IDEC.EQ.0) THEN
      CALL MXDRMM(N,N,AG,G,AFO)
      B1=MXVDOT(N,AFO,AFO)
      ELSE
      CALL MXVCOP(N,G,S)
      CALL MXDPRM(N,H,S,1)
      B1=MXVDOT(N,S,S)
      ENDIF
      IF (XDEL.LE.0.0D 0) THEN
*
*     INITIAL TRUST REGION BOUND
*
      IF (B1.LE.0.0D 0) THEN
      XDEL=GNORM
      ELSE
      XDEL=(B2/B1)*GNORM
      ENDIF
      IF(IEST.EQ.1) XDEL=MIN(XDEL,4.0D0*(F-FMIN)/GNORM)
      XDEL=MIN(XDEL,XMAX)
      ENDIF
      IF (B1.LE.0.0D0.OR.B2*GNORM.GE.B1*XDEL) THEN
*
*     SCALED STEEPEST DESCENT DIRECTION IS ACCEPTED
*
      CALL MXVSCL(N,-XDEL/GNORM,G,S)
      SNORM=XDEL
      ITERD=3
      GO TO 12420
      ENDIF
      IF (IDEC.EQ.0) THEN
*
*     QR DECOMPOSITION
*
      S1=ETA2
      CALL MXDRQF(N,N,AG,H)
      CALL MXDPRC(N,H,INF,S1)
      CALL MXDCMM(N,N,AG,AF,GO)
      NDEC=NDEC+1
      IDEC=1
      ENDIF
*
*     COMPUTATION OF THE NEWTON DIRECTION
*
      CALL MXDPRB(N,H,GO,-1)
      D3=SQRT(MXVDOT(N,GO,GO))
*
*     COMPUTATION OF THE STEEPEST DESCENT DIRECTION
*
      B2=B2/B1
      SNORM=B2*GNORM
      CALL MXVSCL(N,-B2,G,S)
      CALL MXVNEG(N,GO,GO)
      CALL MXVDIF(N,GO,S,XO)
      B1=MXVDOT(N,S,XO)
      B2=MXVDOT(N,XO,XO)
      IF (B2.LE.1.0D-8*XDEL*XDEL) THEN
*
*     NEWTON AND THE STEEPEST DESCENT DIRECTION ARE
*     APPROXIMATELY EQUAL
```

```
*
      CALL MXVCOP(N,GO,S)
      SNORM=D3
      ITERD=1
      ELSE IF (B1.LE.0.0D 0) THEN
*
*     BOUNDARY STEP WITH NEGATIVE INCREMENT
*
      CALL PNSTEP(XDEL,SNORM,-B1,B2,B3)
      CALL MXVDIR(N,-B3,XO,S,S)
      SNORM=XDEL
      ITERD=3
      ELSE IF (D3.LE.XDEL) THEN
*
*     NEWTON DIRECTION IS ACCEPTED
*
      CALL MXVCOP(N,GO,S)
      SNORM=D3
      ITERD=1
      ELSE
*
*     DOUBLE DOGLEG STRATEGY
*
      D3=XDEL/D3
      B3=MXVDOT(N,S,GO)
      D3=MAX(D3,SNORM*SNORM/B3)
      CALL MXVDIR(N,-D3,GO,S,XO)
      B1=SNORM*SNORM-D3*B3
      B2=MXVDOT(N,XO,XO)
      CALL PNSTEP(XDEL,SNORM,-B1,B2,B3)
      CALL MXVDIR(N,-B3,XO,S,S)
      SNORM=XDEL
      ITERD=3
      ENDIF
12420 CONTINUE
      IF (IDEC.EQ.0) THEN
      CALL MXDRMM(N,N,AG,S,AFO)
      PP=MXVDOT(N,AFO,AFO)*0.5D0
      ELSE
      CALL MXVCOP(N,S,GO)
      CALL MXDPRM(N,H,GO,1)
      PP=MXVDOT(N,GO,GO)*0.5D0
      ENDIF
12430 CONTINUE
*
*     ------------------------------
*     END OF DIRECTION DETERMINATION
*     ------------------------------
*
      P=MXVDOT(N,S,G)
      IF (KD.GT.0) P=MXVDOT(N,G,S)
      IF(ITERD.LT.0) THEN
        ITERM=ITERD
      ENDIF
      IF (INITD.LT.0.AND.P.GT.0.0D 0) THEN
*
*     CHANGE OF THE SIGN
*
        CALL MXVNEG(N,S,S)
        P=-P
      ENDIF
*
*     TEST ON LOCALLY CONSTRAINED STEP
*
      IF(SNORM.LE.0.0D 0) THEN
```

```
        IREST=MAX(IREST,1)
      ELSE
        IREST=0
      ENDIF
      IF (IREST.EQ.0) THEN
*
*     PREPARATION OF SIMPLE SEARCH
*
        RMAX=XMAX/SNORM
      ENDIF
      IF(ITERM.NE.0) GOTO 11090
      IF(IREST.NE.0) GOTO 11030
      LDS=LD
11050 CONTINUE
      FP = FO
      RO = 0.0D 0
      FO = F
      PO = P
      CALL MXVCOP(N,X,XO)
      CALL MXVCOP(N,G,GO)
      CALL MXVCOP(N,AF,AFO)
11060 CONTINUE
      CALL PSOGO1(R,F,FO,PO,PP,XDEL,XDELO,XMAX,RMAX,SNORM,BET1,BET2,
     & GAM1,GAM2,EPS4,EPS5,KD,LD,IDIR,ITERS,ITERD,MAXST,NRED,MRED,
     & KTERS,MES1,MES2,MES3,ISYS)
      GOTO (11064,11062) ISYS+1
11062 CONTINUE
      CALL MXVDIR(N,R,S,XO,X)
      CALL PA1SQ1(N,X,F,AF,GA,AG,G,ETAO,KDA,KD,LD,NFV,NFG)
      GOTO 11060
11064 CONTINUE
      IF (ITERS.LE.0) THEN
      IF (IDIR.EQ.0.OR.MOT3.LE.0) THEN
      R=0.0D 0
      F=FO
      P=PO
      CALL MXVCOP(N,XO,X)
      ENDIF
      IF (MOT3.LE.0) CALL MXVCOP(N,AFO,AF)
      IF(IDIR.EQ.0) THEN
      IREST=MAX(IREST,1)
      LD=LDS
      GO TO 11030
      ELSE IF (MOT3.EQ.0) THEN
      LD=LDS
      GO TO 11030
      ELSE
      ITERD=0
      ENDIF
      ENDIF
      IREST=1
      CALL PYTRUD(N,X,XO,G,GO,R,F,FO,P,PO,DMAX,KD,LD,ITERS)
      IF(IDIR.EQ.0) THEN
      GOTO 11020
      ELSE
      GOTO 11030
      ENDIF
11090 CONTINUE
11100 CONTINUE
      IF (IPRNT.GT.1.OR.IPRNT.LT.0)
     & WRITE(6,'(1X,''EXIT FROM PNEQ :'')')
      IF (IPRNT.NE.0)
     & WRITE (6,'(1X,''NIT='',I5,2X,''NFV='',I5,2X,''NFG='',I5,2X,
     & ''F ='', D15.8,2X,''ITERM='',I3)') NIT,NFV,NFG,
     & F,ITERM
```

```
      IF (IPRNT.LT.0)
     & WRITE (6,'(1X,''X ='',5D15.7:/(4X,5D15.7))')(X(I),I=1,N)
13299 CONTINUE
*
*     -----------------
*     END OF METHOD (1)
*     -----------------
*
10399 CONTINUE
      END
```

6.2 Podprogramy realizující iterační algoritmus

```
* SUBROUTINE PA0GS1             ALL SYSTEMS                 97/12/01
* PURPOSE:
* NUMERICAL COMPUTATION OF THE GRADIENT OF THE MODEL FUNCTION.
*
* PARAMETERS :
*  II  N  NUMBER OF VARIABLES.
*  II  KA  INDEF OF THE APPROXIMATED FUNCTION.
*  RI  X(N)  VECTOR OF VARIABLES.
*  RO  GA(N)  GRADIENT OF THE APPROXIMATED FUNCTION.
*  RI  FA  VALUE OF THE APPROXIMATED FUNCTION.
*  RI  ETA1  PRECISION OF THE COMPUTED VALUES.
*  IU  NAV  NUMBER OF APPROXIMATED FUNCTION EVALUATIONS.
*
      SUBROUTINE PA0GS1(N,KA,X,GA,FA,ETA1,NAV)
      INTEGER N,KA,NAV
      REAL*8 X(*),GA(*),FA,ETA1
      REAL*8 XSTEP,XTEMP,FTEMP,ETA
      INTEGER IVAR
      ETA=SQRT(ETA1)
      FTEMP=FA
      DO 4 IVAR=1,N
*
*      STEP SELECTION
*
      XSTEP=1.0D 0
      XSTEP=ETA*MAX(ABS(X(IVAR)),XSTEP)*SIGN(1.0D 0,X(IVAR))
      XTEMP=X(IVAR)
      X(IVAR)=X(IVAR)+XSTEP
      XSTEP=X(IVAR)-XTEMP
      NAV=NAV+1
      CALL FUN(N,KA,X,FA)
*
*      NUMERICAL DIFFERENTIATION
*
      GA(IVAR)=(FA-FTEMP)/XSTEP
      X(IVAR)=XTEMP
    4 CONTINUE
      FA=FTEMP
      RETURN
      END

* SUBROUTINE PA1SQ1             ALL SYSTEMS                 97/12/01
* PURPOSE :
* COMPUTATION OF THE VALUE AND THE GRADIENT OF THE OBJECTIVE FUNCTION
* WHICH IS DEFINED AS A SUM OF SQUARES.
*
* PARAMETERS:
*  II  N  NUMBER OF VARIABLES.
*  RI  X(N)  VECTOR OF VARIABLES.
*  RO  F  VALUE OF THE OBJECTIVE FUNCTION.
*  RO  AF(N)  VALUES OF THE APPROXIMATED FUNCTIONS.
```

```
*  RI  GA(NF)  GRADIENT OF THE APPROXIMATED FUNCTION.
*  RI  AG(N*N)  RECTANGULAR MATRIX WHICH IS USED FOR THE DIRECTION
*         VECTOR DETERMINATION.
*  RO  G(NF)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  RI  ETA1  PRECISION OF THE COMPUTES FUNCTION VALUES.
*  II  KDA  DEGREE OF COMPUTED DERIVATIVES.
*  II  KD  DEGREE OF REQUIRED DERVATIVES.
*  IO  LD  DEGREE OF PREVIOUSLY COMPUTED DERIVATIVES.
*  IU  NFV  NUMBER OF OBJECTIVE FUNCTION VALUES COMPUTED.
*  IU  NFG  NUMBER OF OBJECTIVE FUNCTION GRADIENTS COMPUTED.
*
* SUBPROGRAMS USED :
*  S   MXVCOP  COPYING OF A VECTOR.
*  S   MXVDIR  VECTOR AUGMENTED BY THE SCALED VECTOR.
*  S   MXVSET  INITIATION OF A VECTOR.
*
      SUBROUTINE PA1SQ1(N,X,F,AF,GA,AG,G,ETA1,KDA,KD,LD,NFV,NFG)
      INTEGER N,KDA,KD,LD,NFV,NFG
      REAL*8 X(*),F,AF(*),GA(*),AG(*),G(*),ETA1
      INTEGER KA,NAV
      REAL*8 FA
      IF (KD.LE.LD) RETURN
      IF (KD.GE.0.AND.LD.LT.0) THEN
      F=0.0D0
      NFV=NFV+1
      ENDIF
      IF (KD.GE.1.AND.LD.LT.1) THEN
      CALL MXVSET(N,0.0D0,G)
      IF (KDA.GT.0) NFG=NFG+1
      ENDIF
      NAV=0
      DO 30 KA=1,N
      IF (KD.LT.0) GO TO 30
      IF (LD.GE.0) THEN
      FA = AF(KA)
      GO TO 10
      ELSE
      CALL FUN(N,KA,X,FA)
      AF(KA) = FA
      ENDIF
      IF (LD.GE.0) GO TO 10
      F=F+FA*FA
   10 IF (KD.LT.1) GO TO 30
      IF (KDA.GT.0) THEN
      CALL DFUN(N,KA,X,GA)
      ELSE
      CALL PAOGS1(N,KA,X,GA,FA,ETA1,NAV)
      ENDIF
      CALL MXVDIR(N,FA,GA,G,G)
      CALL MXVCOP(N,GA,AG((KA-1)*N+1))
   30 CONTINUE
      NFV=NFV+NAV/N
      IF (KD.GE.0.AND.LD.LT.0) F=0.5D0*F
      LD=KD
      RETURN
      END

* SUBROUTINE PNSTEP                ALL SYSTEMS                89/12/01
* PURPOSE :
* DETERMINATION OF A SCALING FACTOR FOR THE BOUNDARY STEP.
*
* PARAMETERS :
*  RI  DEL  MAXIMUM STEPSIZE.
*  RI  A  INPUT PARAMETER.
*  RI  B  INPUT PARAMETER.
```

```
*  RI  C  INPUT PARAMETER.
*  RO  ALF  SCALING FACTOR FOR THE BOUNDARY STEP SUCH THAT
*         A**2+2*B*ALF+C*ALF**2=DEL**2.
*
      SUBROUTINE PNSTEP(DEL,A,B,C,ALF)
      REAL*8  DEL, A, B, C, ALF
      REAL*8  DEN, DIS
      REAL*8  ZERO
      PARAMETER (ZERO = 0.0D 0)
      ALF = ZERO
      DEN = (DEL+A) * (DEL-A)
      IF (DEN .LE. ZERO) RETURN
      DIS = B*B + C*DEN
      IF (B .GE. ZERO) THEN
      ALF = DEN / (SQRT(DIS) + B)
      ELSE
      ALF = (SQRT(DIS) - B) / C
      ENDIF
      RETURN
      END

* SUBROUTINE PSOG01                ALL SYSTEMS                97/12/01
* PURPOSE :
* SIMPLE SEARCH WITH TRUST REGION UPDATE.
*
* PARAMETERS :
*  RO  R  VALUE OF THE STEPSIZE PARAMETER.
*  RO  F  VALUE OF THE OBJECTIVE FUNCTION.
*  RI  FO  INITIAL VALUE OF THE OBJECTIVE FUNCTION.
*  RI  PO  INITIAL VALUE OF THE DIRECTIONAL DERIVATIVE.
*  RI  PP  QUADRATIC PART OF THE PREDICTED FUNCTION VALUE.
*  RU  XDEL  TRUST REGION BOUND.
*  RO  XDELO  PREVIOUS TRUST REGION BOUND.
*  RI  XMAX MAXIMUM STEPSIZE.
*  RI  RMAX  MAXIMUM VALUE OF THE STEPSIZE PARAMETER.
*  RI  SNORM  EUCLIDEAN NORM OF THE DIRECTION VECTOR.
*  RI  BET1  LOWER BOUND FOR STEPSIZE REDUCTION.
*  RI  BET2  UPPER BOUND FOR STEPSIZE REDUCTION.
*  RI  GAM1  LOWER BOUND FOR STEPSIZE EXPANSION.
*  RI  GAM2  UPPER BOUND FOR STEPSIZE EXPANSION.
*  RI  EPS4  FIRST TOLERANCE FOR RATIO DF/DFPRED. STEP BOUND IS
*         DECREASED IF DF/DFPRED<EPS4.
*  RI  EPS5  SECOND TOLERANCE FOR RATIO DF/DFPRED. STEP BOUND IS
*         INCREASED IF IT IS ACTIVE AND DF/DFPRED>EPS5.
*  II  KD  DEGREE OF REQUIRED DERVATIVES.
*  IO  LD  DEGREE OF PREVIOUSLY COMPUTED DERIVATIVES.
*  IU  IDIR INDICATOR FOR DIRECTION DETERMINATION.
*         IDIR=0-BASIC DETERMINATION. IDIR=1-DETERMINATION
*         AFTER STEPSIZE REDUCTION. IDIR=2-DETERMINATION AFTER
*         STEPSIZE EXPANSION.
*  IO  ITERS  TERMINATION INDICATOR. ITERS=0-ZERO STEP. ITERS=1-STEP
*         BOUND WAS DECREASED. ITERS=2-STEP BOUND WAS UNCHANGED.
*         ITERS=3-STEP BOUND WAS INCREASED. ITERS=6-FIRST STEPSIZE.
*  II  ITERD TERMINATION INDICATOR. ITERD<0-BAD DECOMPOSITION.
*         ITERD=0-DESCENT DIRECTION. ITERD=1-NEWTON LIKE STEP.
*         ITERD=2-INEXACT NEWTON LIKE STEP. ITERD=3-BOUNDARY STEP.
*         ITERD=4-DIRECTION WITH THE NEGATIVE CURVATURE.
*         ITERD=5-MARQUARDT STEP.
*  IO  MAXST  MAXIMUM STEPSIZE INDICATOR. MAXST=0 OR MAXST=1 IF MAXIMUM
*         STEPSIZE WAS NOT OR WAS REACHED.
*  IO  NRED  ACTUAL NUMBER OF EXTRAPOLATIONS OR INTERPOLATIONS.
*  II  MRED  MAXIMUM NUMBER OF EXTRAPOLATIONS OR INTERPOLATIONS.
*  II  KTERS  TERMINATION SELECTION. KTERS=1-NORMAL TERMINATION.
*         KTERS=6-FIRST STEPSIZE.
*  II  MES1  SWITCH FOR EXTRAPOLATION. MES1=1-CONSTANT INCREASING OF
```

```
*         THE INTERVAL. MES1=2-EXTRAPOLATION SPECIFIED BY THE PARAMETER
*         MES. MES1=3 SUPPRESSED EXTRAPOLATION.
*  II  MES2  SWITCH FOR TERMINATION. MES2=1-NORMAL TERMINATION.
*         MES2=2-TERMINATION AFTER AT LEAST TWO STEPS (ASYMPTOTICALLY
*         PERFECT LINE SEARCH).
*  II  MES3  SAFEGUARD AGAINST ROUNDING ERRORS. MES3=0-SAFEGUARD
*         SUPPRESSED. MES3=1-FIRST LEVEL OF SAFEGUARD. MES3=2-SECOND
*         LEVEL OF SAFEGUARD.
*  IU  ISYS  CONTROL PARAMETER.
*
* COMMON DATA :
*
* METHOD :
* G.A.SCHULTZ, R.B.SCHNABEL, R.H.BYRD: A FAMILY OF TRUST-REGION-BASED
* ALGORITHMS FOR UNCONSTRAINED MINIMIZATION WITH STRONG GLOBAL
* CONVERGENCE PROPERTIES, SIAM J. NUMER.ANAL. 22 (1985) PP. 47-67.
*
      SUBROUTINE PSOGO1(R,F,FO,PO,PP,XDEL,XDELO,XMAX,RMAX,SNORM,
     & BET1,BET2,GAM1,GAM2,EPS4,EPS5,KD,LD,IDIR,ITERS,ITERD,
     & MAXST,NRED,MRED,KTERS,MES1,MES2,MES3,ISYS)
      INTEGER KD,LD,LDA,LDC,IDIR,ITERS,ITERD,MAXST,NRED,MRED,KTERS,
     & MES1,MES2,MES3,ISYS
      REAL*8 R,F,FO,PO,PP,XDEL,XDELO,XMAX,RMAX,SNORM,BET1,BET2,GAM1,
     & GAM2,EPS4,EPS5
      REAL*8 DF,DFPRED
      INTEGER NRED1,NRED2
      SAVE NRED1,NRED2
      GO TO (1,2) ISYS+1
    1 CONTINUE
      IF (IDIR.EQ.0) THEN
      NRED1=0
      NRED2=0
      ENDIF
      IDIR=0
      XDELO=XDEL
*
*     COMPUTATION OF THE NEW FUNCTION VALUE
*
      R=MIN(1.0D 0,RMAX)
      KD= 0
      LD=-1
      ISYS=1
      RETURN
    2 CONTINUE
      IF(KTERS.LT.0.OR.KTERS.GT.5) THEN
      ITERS=6
      ELSE
      DF=FO-F
      DFPRED=-R*(PO+R*PP)
      IF (DF.LT.EPS4*DFPRED) THEN
*
*     STEP IS TOO LARGE, IT HAS TO BE REDUCED
*
      IF (MES1.EQ.1) THEN
      XDEL=BET2*SNORM
      ELSE IF (MES1.EQ.2) THEN
      XDEL=BET2*MIN(0.5D 0*XDEL,SNORM)
      ELSE
      XDEL=0.5D 0*PO*SNORM/(PO+DF)
      XDEL=MAX(XDEL,BET1*SNORM)
      XDEL=MIN(XDEL,BET2*SNORM)
      ENDIF
      ITERS=1
      IF (MES3.LE.1) THEN
      NRED2=NRED2+1
```

```
      ELSE
      IF (ITERD.GT.2) NRED2=NRED2+1
      ENDIF
      ELSE IF (DF.LE.EPS5*DFPRED) THEN
*
*     STEP IS SUITABLE
*
      ITERS=2
      ELSE
*
*     STEP IS TOO SMALL, IT HAS TO BE ENLARGED
*
      IF (MES2.EQ.2) THEN
      XDEL=MAX(XDEL,GAM1*SNORM)
      ELSE IF (ITERD.GT.2) THEN
      XDEL=GAM1*XDEL
      ENDIF
      ITERS=3
      ENDIF
      XDEL=MIN(XDEL,XMAX,GAM2*SNORM)
      IF (FO.LE.F) THEN
      IF (NRED1.GE.MRED) THEN
      ITERS=-1
      ELSE
      IDIR=1
      ITERS=0
      NRED1=NRED1+1
      ENDIF
      ENDIF
      ENDIF
      MAXST=0
      IF (XDEL.GE.XMAX) MAXST=1
      IF (MES3.EQ.0) THEN
      NRED=NRED1
      ELSE
      NRED=NRED2
      ENDIF
      ISYS=0
      RETURN
      END

* SUBROUTINE PYFUT1                ALL SYSTEMS                98/12/01
* PURPOSE :
* TERMINATION CRITERIA AND TEST ON RESTART.
*
* PARAMETERS :
*  II  N  ACTUAL NUMBER OF VARIABLES.
*  RI  F  NEW VALUE OF THE OBJECTIVE FUNCTION.
*  RI  FO  OLD VALUE OF THE OBJECTIVE FUNCTION.
*  RI  UMAX  MAXIMUN ABSOLUTE VALUE OF THE NEGATIVE LAGRANGE MULTIPLIER.
*  RO  GMAX  NORM OF THE TRANSFORMED GRADIENT.
*  RI  DMAX  MAXIMUM RELATIVE DIFFERENCE OF VARIABLES.
*  RI  TOLX  LOWER BOUND FOR STEPLENGTH.
*  RI  TOLF  LOWER BOUND FOR FUNCTION DECREASE.
*  RI  TOLB  LOWER BOUND FOR FUNCTION VALUE.
*  RI  TOLG  LOWER BOUND FOR GRADIENT.
*  II  KD  DEGREE OF REQUIRED DERIVATIVES.
*  IU  NIT  ACTUAL NUMBER OF ITERATIONS.
*  II  KIT  NUMBER OF THE ITERATION AFTER RESTART.
*  II  MIT  MAXIMUM NUMBER OF ITERATIONS.
*  IU  NFV  ACTUAL NUMBER OF COMPUTED FUNCTION VALUES.
*  II  MFV  MAXIMUM NUMBER OF COMPUTED FUNCTION VALUES.
*  IU  NFG  ACTUAL NUMBER OF COMPUTED GRADIENT VALUES.
*  II  MFG  MAXIMUM NUMBER OF COMPUTED GRADIENT VALUES.
*  IU  NTESX  ACTUAL NUMBER OF TESTS ON STEPLENGTH.
```

```
*  II  MTESX  MAXIMUM NUMBER OF TESTS ON STEPLENGTH.
*  IU  NTESF  ACTUAL NUMBER OF TESTS ON FUNCTION DECREASE.
*  II  MTESF  MAXIMUM NUMBER OF TESTS ON FUNCTION DECREASE.
*  II  ITES  SYSTEM VARIBLE WHICH SPECIFIES TERMINATION. IF ITES=0
*         THEN TERMINATION IS SUPPRESSED.
*  II  IRES1  RESTART SPECIFICATION. RESTART IS PERFORMED AFTER
*         IRES1*N+IRES2 ITERATIONS.
*  II  IRES2  RESTART SPECIFICATION. RESTART IS PERFORMED AFTER
*         IRES1*N+IRES2 ITERATIONS.
*  IU  IREST  RESTART INDICATOR. RESTART IS PERFORMED IF IREST>0.
*  II  ITERS  TERMINATION INDICATOR FOR STEPLENGTH DETERMINATION.
*         ITERS=0 FOR ZERO STEP.
*  IO  ITERM  TERMINATION INDICATOR. ITERM=1-TERMINATION AFTER MTESX
*         UNSUFFICIENT STEPLENGTHS. ITERM=2-TERMINATION AFTER MTESF
*         UNSUFFICIENT FUNCTION DECREASES. ITERM=3-TERMINATION ON LOWER
*         BOUND FOR FUNCTION VALUE. ITERM=4-TERMINATION ON LOWER BOUND
*         FOR GRADIENT. ITERM=11-TERMINATION AFTER MAXIMUM NUMBER OF
*         ITERATIONS. ITERM=12-TERMINATION AFTER MAXIMUM NUMBER OF
*         COMPUTED FUNCTION VALUES.
*
      SUBROUTINE PYFUT1(N,F,FO,UMAX,GMAX,DMAX,TOLX,TOLF,TOLB,TOLG,KD,
     & NIT,KIT,MIT,NFV,MFV,NFG,MFG,NTESX,MTESX,NTESF,MTESF,ITES,IRES1,
     & IRES2,IREST,ITERS,ITERM)
      INTEGER N,KD,NIT,KIT,MIT,NFV,MFV,NFG,MFG,NTESX,MTESX,NTESF,MTESF,
     & ITES,IRES1,IRES2,IREST,ITERS,ITERM
      REAL*8  F,FO,UMAX,GMAX,DMAX,TOLX,TOLF,TOLG,TOLB
      REAL*8  TEMP
      IF (ITERM.LT.0) RETURN
      IF (ITES .LE.0) GOTO 2
      IF (ITERS.EQ.0) GOTO 1
      IF (NIT.LE.0) FO=F+MIN(SQRT(ABS(F)),ABS(F)/1.0D 1)
      IF (F.LE.TOLB) THEN
      ITERM = 3
      RETURN
      ENDIF
      IF (KD.GT.0) THEN
      IF (GMAX.LE.TOLG.AND.UMAX.LE.TOLG) THEN
      ITERM = 4
      RETURN
      ENDIF
      ENDIF
      IF (NIT.LE.0) THEN
      NTESX = 0
      NTESF = 0
      ENDIF
      IF (DMAX.LE.TOLX) THEN
      ITERM = 1
      NTESX = NTESX + 1
      IF (NTESX .GE. MTESX) RETURN
      ELSE
      NTESX = 0
      ENDIF
      TEMP=ABS(FO-F)/MAX(ABS(F),1.0D 0)
      IF (TEMP.LE.TOLF) THEN
      ITERM = 2
      NTESF = NTESF+1
      IF (NTESF.GE.MTESF) RETURN
      ELSE
      NTESF = 0
      ENDIF
    1 IF (NIT.GE.MIT) THEN
      ITERM = 11
      RETURN
      ENDIF
      IF (NFV.GE.MFV) THEN
```

```
      ITERM = 12
      RETURN
      ENDIF
      IF (NFG.GE.MFG) THEN
      ITERM = 13
      RETURN
      ENDIF
    2 ITERM = 0
      IF (N.GT.0.AND.NIT-KIT.GE.IRES1*N+IRES2) THEN
      IREST=MAX(IREST,1)
      ENDIF
      NIT = NIT + 1
      RETURN
      END

* SUBROUTINE PYTRUD             ALL SYSTEMS                    98/12/01
* PURPOSE :
* VECTORS OF VARIABLES DIFFERENCE AND GRADIENTS DIFFERENCE ARE COMPUTED
* AND SCALED. TEST VALUE DMAX IS DETERMINED.
*
* PARAMETERS :
*  II  NF DECLARED NUMBER OF VARIABLES.
*  RI  X(NF)  VECTOR OF VARIABLES.
*  RU  XO(NF)  VECTORS OF VARIABLES DIFFERENCE.
*  RI  G(NF)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  RU  GO(NF)  GRADIENTS DIFFERENCE.
*  RO  R  VALUE OF THE STEPSIZE PARAMETER.
*  RO  F  NEW VALUE OF THE OBJECTIVE FUNCTION.
*  RI  FO  OLD VALUE OF THE OBJECTIVE FUNCTION.
*  RO  P  NEW VALUE OF THE DIRECTIONAL DERIVATIVE.
*  RI  PO  OLD VALUE OF THE DIRECTIONAL DERIVATIVE.
*  RO  DMAX  MAXIMUM RELATIVE DIFFERENCE OF VARIABLES.
*  II  KD  DEGREE OF REQUIRED DERVATIVES.
*  IO  LD  DEGREE OF PREVIOUSLY COMPUTED DERIVATIVES.
*  II  ITERS  TERMINATION INDICATOR FOR STEPLENGTH DETERMINATION.
*         ITERS=0 FOR ZERO STEP.
*
* SUBPROGRAMS USED :
*  S   PYSET1  DEGREE DEFINITION OF THE COMPUTED DERIVATIVES.
*  S   MXVDIF  DIFFERENCE OF TWO VECTORS.
*  S   MXVSAV  DIFFERENCE OF TWO VECTORS WITH COPYING AND SAVING THE
*         SUBSTRACTED ONE.
*
      SUBROUTINE PYTRUD(NF,X,XO,G,GO,R,F,FO,P,PO,DMAX,KD,LD,ITERS)
      INTEGER NF,KD,LD,ITERS
      REAL*8 X(*),XO(*),G(*),GO(*),R,F,FO,P,PO,DMAX
      INTEGER I
      IF (ITERS.GT.0) THEN
      CALL MXVDIF(NF,X,XO,XO)
      CALL MXVDIF(NF,G,GO,GO)
      PO=R*PO
      P=R*P
      ELSE
      F = FO
      P = PO
      CALL MXVSAV(NF,X,XO)
      CALL MXVSAV(NF,G,GO)
      LD=KD
      ENDIF
      DMAX = 0.0D 0
      DO 1 I=1,NF
      DMAX=MAX(DMAX,ABS(XO(I))/MAX(ABS(X(I)),1.0D 0))
    1 CONTINUE
      RETURN
      END
```

6.3 Maticové podprogramy

```
* SUBROUTINE MXDCMM               ALL SYSTEMS                91/12/01
* PURPOSE :
* MULTIPLICATION OF A COLUMNWISE STORED DENSE RECTANGULAR MATRIX A
* BY A VECTOR X.
*
* PARAMETERS :
*  II  N  NUMBER OF ROWS OF THE MATRIX A.
*  II  M  NUMBER OF COLUMNS OF THE MATRIX A.
*  RI  A(N*M)  RECTANGULAR MATRIX STORED COLUMNWISE IN THE
*         ONE-DIMENSIONAL ARRAY.
*  RI  X(M)  INPUT VECTOR.
*  RO  Y(N)  OUTPUT VECTOR EQUAL TO A*X.
*
* SUBPROGRAMS USED :
*  S   MXVDIR  VECTOR AUGMENTED BY THE SCALED VECTOR.
*  S   MXVSET  INITIATION OF A VECTOR.
*
      SUBROUTINE MXDCMM(N,M,A,X,Y)
      INTEGER N,M
      DOUBLE PRECISION A(*),X(*),Y(*)
      INTEGER J,K
      CALL MXVSET(N,0.0D 0,Y)
      K=0
      DO 1 J=1,M
      CALL MXVDIR(N,X(J),A(K+1),Y,Y)
      K=K+N
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXDCMU               ALL SYSTEMS                91/12/01
* PURPOSE :
* UPDATE OF A COLUMNWISE STORED DENSE RECTANGULAR MATRIX A. THIS MATRIX
* IS UPDATED BY THE RULE A:=A+ALF*X*TRANS(Y).
*
* PARAMETERS :
*  II  N  NUMBER OF ROWS OF THE MATRIX A.
*  II  M  NUMBER OF COLUMNS OF THE MATRIX A.
*  RU  A(N*M)  RECTANGULAR MATRIX STORED COLUMNWISE IN THE
*         ONE-DIMENSIONAL ARRAY.
*  RI  ALF  SCALAR PARAMETER.
*  RI  X(N)  INPUT VECTOR.
*  RI  Y(M)  INPUT VECTOR.
*
      SUBROUTINE MXDCMU(N,M,A,ALF,X,Y)
      INTEGER N,M
      REAL*8 A(N*M),ALF,X(N),Y(M)
      REAL*8 TEMP
      INTEGER I,J,K
      K=0
      DO 2 J=1,M
      TEMP=ALF*Y(J)
      DO 1 I=1,N
      A(K+I)=A(K+I)+TEMP*X(I)
    1 CONTINUE
      K=K+N
    2 CONTINUE
      RETURN
      END

* SUBROUTINE MXDGGB                ALL SYSTEMS                94/12/01
* PURPOSE :
* SOLUTION OF A SYSTEM OF LINEAR EQUATIONS WITH A DENSE UNSYMMETRIC
```

```
* MATRIX A USING THE FACTORIZATION A=P*L*U OBTAINED BY THE SUBROUTINE
* UXDGGB.
*
* PARAMETERS :
*  II  N ORDER OF THE MATRIX A.
*  RU  A(N*N)  ON INPUT A GIVEN DENSE GENERAL SQUARE MATRIX A STORED IN
*         THE PACKED FORM. ON OUTPUT THE COMPUTED FACTORIZATION A=P*L*U.
*  IO  IA(N)  A PERMUTATION FIELD WHICH CONTAINS AN INFORMATION ABOUT
*         PERMUTATION MATRIX P.
*  RU  X(N)  ON INPUT THE RIGHT HAND SIDE OF A SYSTEM OF LINEAR
*         EQUATIONS. ON OUTPUT THE SOLUTION OF A SYSTEM OF LINEAR
*         EQUATIONS.
*  RA  Y(N)  AUXILIARY VECTOR.
*  JOB  OPTION. JOB=0 - SOLUTION WITH THE ORIGINAL MATRIX.
*         JOB=1 - SOLUTION WITH THE MATRIX TRSANSPOSE.
*
* METHOD :
* BACK SUBSTITUTION.
*
      SUBROUTINE MXDGGB(N,A,IA,X,Y,JOB)
      INTEGER N,IA(N),JOB
      REAL*8 A(N*N),X(N),Y(N)
      INTEGER J,JP,K,L
      IF (JOB.LE.0) THEN
      L=0
      DO 2 K=1,N
      DO 1 J=1,K-1
      JP=IA(J)
      X(K)=X(K)-A(L+JP)*X(J)
    1 CONTINUE
      L=L+N
    2 CONTINUE
      L=N*N
      DO 4 K=N,1,-1
      L=L-N
      DO 3 J=K+1,N
      JP=IA(J)
      X(K)=X(K)-A(L+JP)*X(J)
    3 CONTINUE
      X(K)=X(K)/A(L+IA(K))
    4 CONTINUE
      CALL MXVSBP(N,IA,X,Y)
      ELSE
      CALL MXVSFP(N,IA,X,Y)
      L=0
      DO 6 K=1,N
      X(K)=X(K)/A(L+IA(K))
      DO 5 J=K+1,N
      JP=IA(J)
      X(J)=X(J)-A(L+JP)*X(K)
    5 CONTINUE
      L=L+N
    6 CONTINUE
      L=N*N
      DO 8 K=N,1,-1
      L=L-N
      DO 7 J=1,K-1
      JP=IA(J)
      X(J)=X(J)-A(L+JP)*X(K)
    7 CONTINUE
    8 CONTINUE
      ENDIF
      RETURN
      END
```

```
* SUBROUTINE MXDGGF                ALL SYSTEMS                94/12/01
*
* PURPOSE :
* FACTORIZATION A=P*L*U OF A DENSE GENERAL SQUARE MATRIX A WHERE P IS
* A PERMUTATION MATRIX, L IS A LOWER TRIANGULAR MATRIX AND U IS AN UPPER
* TRIANGULAR MATRIX,
*
* PARAMETERS :
*  II  N ORDER OF THE MATRIX A.
*  RU  A(N*N)  ON INPUT A GIVEN DENSE GENERAL SQUARE MATRIX A STORED IN
*         THE PACKED FORM. ON OUTPUT THE COMPUTED FACTORIZATION A=P*L*U.
*  IO  IA(N)  A PERMUTATION FIELD WHICH CONTAINS AN INFORMATION ABOUT
*         PERMUTATION MATRIX P.
*  IO  INF  AN INFORMATION OBTAINED IN THE FACTORIZATION PROCESS. IF
*         INF=0 THEN A IS SUFFICIENTLY NONSINGULAR. IF INF>0 THEN INF
*         IS A NUMBER OF THE ELIMINATION STEP IN WHICH A SINGULARITY
*         WAS DETECTED.
*  RI  TOL  ON INPUT A DESIRED TOLERANCE FOR POSITIVE DEFINITENESS. ON
*         OUTPUT THE MOST NEGATIVE DIAGONAL ELEMENT USED IN THE
*         FACTORIZATION PROCESS (IF INF>0).
*
* METHOD :
* GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING.
*
      SUBROUTINE MXDGGF(N,A,IA,INF,TOL)
      INTEGER N,IA(N),INF
      REAL*8 A(N*N),TOL
      REAL*8 APIV,APOM
      REAL*8 ZERO,ONE
      PARAMETER (ZERO=0.0D 0,ONE=1.0D 0)
      INTEGER I,IP,J,K,KP,L,M
      INF=0
      L=0
      CALL MXVINP(N,IA)
      DO 10 K=1,N
      KP=IA(K)
      J=0
      APIV=ZERO
      DO 3 I=K,N
      IP=IA(I)
      APOM=ABS(A(L+IP))
      IF (APIV.LT.APOM) THEN
      APIV=APOM
      J=I
      ENDIF
    3 CONTINUE
      IF (J.EQ.0) THEN
      INF=K
      A(L+KP)=TOL
      ELSE
      IA(K)=IA(J)
      IA(J)=KP
      KP=IA(K)
      IF (APIV.LE.TOL) THEN
      INF=K
      A(L+KP)=SIGN(TOL,A(L+KP))
      ENDIF
      ENDIF
      APIV=ONE/A(L+KP)
      M=L
      DO 9 J=K+1,N
      M=M+N
      APOM=APIV*A(M+KP)
      DO 8 I=K+1,N
      IP=IA(I)
```

```
      A(M+IP)=A(M+IP)-A(L+IP)*APOM
    8 CONTINUE
      A(M+KP)=APOM
    9 CONTINUE
      L=L+N
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXDPRB                ALL SYSTEMS                89/12/01
* PURPOSE :
* SOLUTION OF A SYSTEM OF LINEAR EQUATIONS WITH A DENSE SYMMETRIC
* POSITIVE DEFINITE MATRIX A USING THE FACTORIZATION A=TRANS(R)*R.
*
* PARAMETERS :
*  II  N ORDER OF THE MATRIX A.
*  RI  A(N*(N+1)/2) FACTORIZATION A=TRANS(R)*R.
*  RU  X(N)  ON INPUT THE RIGHT HAND SIDE OF A SYSTEM OF LINEAR
*         EQUATIONS. ON OUTPUT THE SOLUTION OF A SYSTEM OF LINEAR
*         EQUATIONS.
*  II  JOB  OPTION. IF JOB=0 THEN X:=A**(-1)*X. IF JOB>0 THEN
*         X:=TRANS(R)**(-1)*X. IF JOB<0 THEN X:=R**(-1)*X.
*
* METHOD :
* BACK SUBSTITUTION
*
      SUBROUTINE MXDPRB(N,A,X,JOB)
      INTEGER JOB,N
      DOUBLE PRECISION A(*),X(*)
      INTEGER I,II,IJ,J
      IF (JOB.GE.0) THEN
*
*     PHASE 1 : X:=TRANS(R)**(-1)*X
*
      IJ = 0
      DO 20 I = 1,N
      DO 10 J = 1,I - 1
      IJ = IJ + 1
      X(I) = X(I) - A(IJ)*X(J)
   10 CONTINUE
      IJ = IJ + 1
      X(I) = X(I)/A(IJ)
   20 CONTINUE
      END IF
      IF (JOB.LE.0) THEN
*
*     PHASE 2 : X:=R**(-1)*X
*
      II = N* (N+1)/2
      DO 40 I = N,1,-1
      IJ = II
      DO 30 J = I + 1,N
      IJ = IJ + J - 1
      X(I) = X(I) - A(IJ)*X(J)
   30 CONTINUE
      X(I) = X(I)/A(II)
      II = II - I
   40 CONTINUE
      END IF
      RETURN
      END

* SUBROUTINE MXDPRC                ALL SYSTEMS                92/12/01
* PURPOSE :
* CORRECTION OF A SINGULAR DENSE SYMMETRIC POSITIVE SEMIDEFINITE MATRIX
```

```
* A DECOMPOSED AS A=TRANS(R)*R.
*
* PARAMETERS :
*  II  N  ORDER OF THE MATRIX A.
*  RU  A(N*(N+1)/2)  DENSE SYMMETRIC MATRIX STORED IN THE PACKED FORM.
*  IO  INF  AN INFORMATION OBTAINED IN THE CORRECTION PROCESS. IF
*         INF=0 THEN A IS SUFFICIENTLY POSITIVE DEFINITE. IF
*         INF<0 THEN A IS NOT SUFFICIENTLY POSITIVE DEFINITE.
*         PROCESS.
*  RI  TOL  DESIRED TOLERANCE FOR POSITIVE DEFINITENESS.
*
      SUBROUTINE MXDPRC(N,A,INF,TOL)
      INTEGER N,INF
      REAL*8  A(N*(N+1)/2),TOL
      REAL*8 TOL1,TEMP
      INTEGER L,I
      REAL*8 ZERO
      PARAMETER (ZERO=0.0D 0)
      INF=0
      TOL1=SQRT(TOL)
      TEMP=TOL1
      DO 3 I=1,N*(N+1)/2
      TEMP=MAX(TEMP,ABS(A(I)))
    3 CONTINUE
      TEMP=TEMP*TOL1
      L=0
      DO 1 I=1,N
      L=L+I
      IF (ABS(A(L)).LE.TEMP) THEN
      A(L)=SIGN(TEMP,A(L))
      INF=-1
      ENDIF
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXDPRM                ALL SYSTEMS                91/12/01
* PURPOSE :
* MULTIPLICATION OF A GIVEN VECTOR X BY A DENSE SYMMETRIC POSITIVE
* DEFINITE MATRIX A USING THE FACTORIZATION A=TRANS(R)*R.
*
* PARAMETERS :
*  II  N ORDER OF THE MATRIX A.
*  RI  A(N*(N+1)/2) FACTORIZATION A=TRANS(R)*R.
*  RU  X(N)  ON INPUT THE GIVEN VECTOR. ON OUTPUT THE RESULT OF
*         MULTIPLICATION.
*  II  JOB  OPTION. IF JOB=0 THEN X:=A*X. IF JOB>0 THEN X:=R*X.
*         IF JOB<0 THEN X:=TRANS(R)*X.
*
      SUBROUTINE MXDPRM( N, A, X, JOB)
      INTEGER N, JOB
      REAL*8  A(N*(N+1)/2), X(N)
      INTEGER  I, J, II, IJ
      IF (JOB .GE. 0)  THEN
C
C     PHASE 1 : X:=R*X
C
      II = 0
      DO 3 I = 1, N
      II = II + I
      X(I) = A(II) * X(I)
      IJ = II
      DO 2 J = I+1, N
      IJ = IJ + J - 1
      X(I) = X(I) + A(IJ) * X(J)
```

```
    2 CONTINUE
    3 CONTINUE
      ENDIF
      IF (JOB .LE. 0)  THEN
C
C     PHASE 2 : X:=TRANS(R)*X
C
      IJ = N*(N+1)/2
      DO 6  I = N, 1, -1
      X(I) = A(IJ) * X(I)
      DO 5  J = I-1, 1, -1
      IJ = IJ - 1
      X(I) = X(I) + A(IJ) * X(J)
    5 CONTINUE
      IJ = IJ - 1
    6 CONTINUE
      ENDIF
      RETURN
      END

* SUBROUTINE UXDRFA               ALL SYSTEMS                05/12/01
* PURPOSE :
* TRIANGULAR DECOMPOSITION OF KERNEL OF THE ORTHOGONAL PROJECTION
* IS UPDATED AFTER CONSTRAINT ADDITION.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  IU  N  ACTUAL NUMBER OF VARIABLES.
*  II  NC  NUMBER OF LINEARIZED CONSTRAINTS.
*  IU  ICA(NF)  VECTOR CONTAINING INDICES OF ACTIVE CONSTRAINTS.
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RI  CR(NF*(NF+1)/2)  TRIANGULAR DECOMPOSITION OF KERNEL OF THE
*         ORTHOGONAL PROJECTION.
*  RA  S(NF)  AUXILIARY VECTOR.
*  RI  EPS7  TOLERANCE FOR LINEAR INDEPENDENCE OF CONSTRAINTS.
*  RO  GMAX  MAXIMUM ABSOLUTE VALUE OF A PARTIAL DERIVATIVE.
*  RO  UMAX  MAXIMUM ABSOLUTE VALUE OF A NEGATIVE LAGRANGE MULTIPLIER.
*  II  INEW  INDEX OF THE NEW ACTIVE CONSTRAINT.
*  IO  IER  ERROR INDICATOR.
*
* SUBPROGRAMS USED :
*  S   MXSPRB  SPARSE BACK SUBSTITUTION.
*  S   MXVCOP  COPYING OF A VECTOR.
*  RF  MXVDOT  DOT PRODUCT OF TWO VECTORS.
*
      SUBROUTINE MXDRFA(NF,N,NC,ICA,CG,CR,S,EPS7,GMAX,UMAX,INEW,IER)
      INTEGER NF,N,NC,ICA(NF),INEW,NADD,IER
      REAL*8 CG(NF*NC),CR(NF*(NF+1)/2),S(NF),EPS7,GMAX,UMAX
      REAL*8 MXVDOT
      INTEGER NCA,NCR,I,J,K,L
      IER=0
      IF (N.LE.0) IER=2
      IF (INEW.EQ.0) IER=3
      IF (IER.NE.0) RETURN
      NCA=NF-N
      NCR=NCA*(NCA+1)/2
      CALL MXVCOP(NF,CG((INEW-1)*NF+1),S)
      GMAX=MXVDOT(NF,CG((INEW-1)*NF+1),S)
      DO 1 J=1,NCA
      L=ICA(J)
      CR(NCR+J)=MXVDOT(NF,CG((L-1)*NF+1),S)
    1 CONTINUE
      IF (NCA.EQ.0) THEN
      UMAX=GMAX
```

```
      ELSE
      CALL MXDPRB(NCA,CR,CR(NCR+1),1)
      UMAX=GMAX-MXVDOT(NCA,CR(NCR+1),CR(NCR+1))
      ENDIF
      IF (UMAX.LE.EPS7*GMAX) THEN
      IER=1
      RETURN
      ELSE
      N=N-1
      NCA=NCA+1
      NCR=NCR+NCA
      ICA(NCA)=INEW
      CR(NCR)=SQRT(UMAX)
      ENDIF
      RETURN
      END

* SUBROUTINE MXDRFD               ALL SYSTEMS                   05/12/01
* PURPOSE :
* DETERMINATION OF THE NEW VALUES OF THE CONSTRAINT FUNCTIONS.
*
* PARAMETERS :
*  II  NC  NUMBER OF CONSTRAINTS.
*  RI  CF(NF)  VECTOR CONTAINING VALUES OF THE CONSTRAINT FUNCTIONS.
*  RO  CFD(NF)  VECTOR CONTAINING INCREMENTS OF THE CONSTRAINT
*         FUNCTIONS.
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  RO  STEP  CURRENT STEPSIZE.
*
      SUBROUTINE MXDRFD(NC,CF,CFD,IC,STEP)
      INTEGER NC,IC(NC)
      REAL*8 CF(NC),CFD(NC),STEP
      INTEGER KC
      DO 1 KC=1,NC
      IF (IC(KC).GE.0.AND.IC(KC).LE.10) THEN
      CF(KC)=CF(KC)+STEP*CFD(KC)
      ELSE IF (IC(KC).LT.-10) THEN
      CF(KC)=CF(KC)+STEP*CFD(KC)
      ENDIF
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXDRFG             ALL SYSTEMS                   05/12/01
* PURPOSE :
* GRADIENT DETERMINATION IN THE FIRST PHASE OF LP SUBROUTINE.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  II  NC  NUMBER OF CONSTRAINTS.
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RI  G(NF)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  IU  INEW  INDEX OF THE NEW ACTIVE CONSTRAINT.
*
* SUBPROGRAMS USED :
*  S   MXVDIR  VECTOR AUGMENTED BY THE SCALED VECTOR.
*  S   MXVSET  INITIATION OF A VECTOR.
*
      SUBROUTINE MXDRFG(NF,NC,IC,CG,G,INEW)
      INTEGER NF,NC,IC(NC),INEW
      REAL*8 CG(NF*NC),G(NF)
      INTEGER KC
      CALL MXVSET(NF,0.0D 0,G)
```

```
      INEW=0
      DO 4 KC=1,NC
      IF (IC(KC).EQ.-11) THEN
      CALL MXVDIR(NF,-1.0D 0,CG((KC-1)*NF+1),G,G)
      INEW=1
      ELSE IF (IC(KC).EQ.-12) THEN
      CALL MXVDIR(NF, 1.0D 0,CG((KC-1)*NF+1),G,G)
      INEW=1
      ENDIF
    4 CONTINUE
      RETURN
      END

* SUBROUTINE MXDRFL             ALL SYSTEMS                   97/12/01
* PURPOSE :
* TEST ON ACTIVITY OF A GIVEN LINEAR CONSTRAINT.
*
* PARAMETERS :
*  II  NC  NUMBER OF CONSTRAINTS.
*  II  KC  INDEX OF A GIVEN CONSTRAINT.
*  RI  CF(NC)  VECTOR CONTAINING VALUES OF THE CONSTRAINT FUNCTIONS.
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  RI  CB(NC)  VECTOR CONTAINING LOWER BOUNDS FOR CONSTRAINT FUNCTIONS.
*  RI  EPS9  TOLERANCE FOR ACTIVE CONSTRAINTS.
*  IO  INEW  INDEX OF THE NEW ACTIVE CONSTRAINT.
*
      SUBROUTINE MXDRFL(NC,KC,CF,IC,CB,EPS9,INEW)
      INTEGER NC,KC,IC(NC),INEW
      REAL*8 CF(NC),CB(NC),EPS9
      REAL*8 TEMP
      IF (IC(KC).LT.-10) IC(KC)=-IC(KC)-10
      IF (IC(KC).EQ.1) THEN
      TEMP=EPS9*MAX(ABS(CB(KC)),1.0D 0)
      IF (CF(KC).GT.CB(KC)+TEMP) THEN
      ELSE IF (CF(KC).GE.CB(KC)-TEMP) THEN
      IC(KC)=11
      INEW=KC
      ELSE
      IC(KC)=-11
      ENDIF
      ELSE IF (IC(KC).EQ.2) THEN
      TEMP=EPS9*MAX(ABS(CB(KC)),1.0D 0)
      IF (CF(KC).LT.CB(KC)-TEMP) THEN
      ELSE IF (CF(KC).LE.CB(KC)+TEMP) THEN
      IC(KC)=12
      INEW=KC
      ELSE
      IC(KC)=-12
      ENDIF
      ENDIF
      RETURN
      END

* SUBROUTINE MXDRFM              ALL SYSTEMS                   05/12/01
* PURPOSE :
* DETERMINATION OF THE MAXIMUM STEPSIZE USING LINEAR CONSTRAINTS.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  II  NC  NUMBER OF CURRENT LINEAR CONSTRAINTS.
*  RI  CF(NF)  VECTOR CONTAINING VALUES OF THE CONSTRAINT FUNCYIONS.
*  RO  CFD(NF)  VECTOR CONTAINING INCREMENTS OF THE CONSTRAINT
*         FUNCTIONS.
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  RI  CB(NC)  VECTOR CONTAINING BOUNDS FOR CONSTRAINT FUNCTIONS.
```

```
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RI  S(NF)  DIRECTION VECTOR.
*  RO  STEP  MAXIMUM STEPSIZE.
*  II  KBC  SPECIFICATION OF LINEAR CONSTRAINTS. KBC=0-NO LINEAR
*         CONSTRAINTS. KBC=1-ONE SIDED LINEAR CONSTRAINTS. KBC=2=TWO
*         SIDED LINEAR CONSTRAINTS.
*  II  KREM  INDICATION OF LINEARLY DEPENDENT GRADIENTS.
*  IO  INEW  INDEX OF THE NEW ACTIVE FUNCTION.
*
* SUBPROGRAMS USED :
*  RF  MXVDOT  DOT PRODUCT OF TWO VECTORS.
*
      SUBROUTINE MXDRFM(NF,NC,CF,CFD,IC,CB,CG,S,STEP,KREM,INEW)
      INTEGER NF,NC,IC(NC),KBC,KREM,INEW
      REAL*8 CF(NC),CFD(NC),CB(NC),CG(NF*NC),S(NF),STEP
      REAL*8 TEMP,MXVDOT
      INTEGER JCG,KC
      JCG=1
      DO 1 KC=1,NC
      IF (KREM.GT.0.AND.IC(KC).GT.10) IC(KC)=IC(KC)-10
      IF (IC(KC).GT.0.AND.IC(KC).LE.10) THEN
      TEMP=MXVDOT(NF,CG(JCG),S)
      CFD(KC)=TEMP
      IF (TEMP.LT.0.0D 0) THEN
      IF (IC(KC).EQ.1) THEN
      TEMP=(CB(KC)-CF(KC))/TEMP
      IF (TEMP.LE.STEP) THEN
      INEW=KC
      STEP=TEMP
      ENDIF
      ENDIF
      ELSE IF (TEMP.GT.0.0D 0) THEN
      IF (IC(KC).EQ.2) THEN
      TEMP=(CB(KC)-CF(KC))/TEMP
      IF (TEMP.LE.STEP) THEN
      INEW=KC
      STEP=TEMP
      ENDIF
      ENDIF
      ENDIF
      ELSE IF (IC(KC).LT.-10) THEN
      TEMP=MXVDOT(NF,CG(JCG),S)
      CFD(KC)=TEMP
      IF (TEMP.GT.0.0D 0) THEN
      IF (IC(KC).EQ.-11) THEN
      TEMP=(CB(KC)-CF(KC))/TEMP
      IF (TEMP.LE.STEP) THEN
      INEW=KC
      STEP=TEMP
      ENDIF
      ENDIF
      ELSE IF (TEMP.LT.0.0D 0) THEN
      IF (IC(KC).EQ.-12) THEN
      TEMP=(CB(KC)-CF(KC))/TEMP
      IF (TEMP.LE.STEP) THEN
      INEW=KC
      STEP=TEMP
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      JCG=JCG+NF
    1 CONTINUE
      RETURN
```

```
      END

* SUBROUTINE MXDRFP             ALL SYSTEMS                   05/12/01
* PURPOSE :
* DETERMINATION OF THE FEASIBLE POINT.
*
* PARAMETERS :
*  II  NF  NUMBER OF VARIABLES.
*  II  NC  NUMBER OF LINEAR CONSTRAINTS.
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RI  CB(NC)  VECTOR CONTAINING LOWER BOUNDS FOR CONSTRAINT FUNCTIONS.
*  RO  X(NF)  VECTOR OF VARIABLES.
*  RA  CF(NF)  VECTOR CONTAINING VALUES OF THE CONSTRAINT FUNCTIONS.
*  RA  CFD(NF)  VECTOR CONTAINING INCREMENTS OF THE CONSTRAINT
*         FUNCTIONS.
*  IA  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  IA  ICA(NF)  VECTOR CONTAINING INDICES OF ACTIVE CONSTRAINTS.
*  RA  CR(NF*(NF+1)/2)  TRIANGULAR DECOMPOSITION OF KERNEL OF THE
*         ORTHOGONAL PROJECTION.
*  RA  CZ(NF)  VECTOR OF LAGRANGE MULTIPLIERS.
*  RA  G(NF)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  RA  S(NF)  DIRECTION VECTOR.
*
* SUBPROGRAMS USED :
*  S   MXDRFM  MAXIMUM STEPSIZE USING LINEAR CONSTRAINTS.
*  S   MXDRFN  IDENTIFICATION OF ACTIVE LINEAR CONSTRAINTS.
*  S   MXDRFD  NEW VALUES OF CONSTRAINT FUNCTIONS.
*  S   MXDRFG  DETERMINATION OF THE FIRST PHASE GRADIENT VECTOR.
*  S   MXDRFL  GRADIENT OF THE LAGRANGIAN FUNCTION IS DETERMINED.
*  S   MXDRFS  NEGATIVE PROJECTED GRADIENT IS DETERMINED.
*  S   MXDRFT  THE OPTIMUM LAGRANGE MULTIPLIER IS DETERMINED.
*  S   MXDRFV  AN AUXILIARY VECTOR IS DETERMINED.
*  S   MXDRFA  CONSTRAINT ADDITION.
*  S   MXDRFR  CONSTRAINT DELETION.
*  S   MXDPRB  BACK SUBSTITUTION AFTER CHOLESKI DECOMPOSITION.
*  S   MXDSMI  DETERMINATION OF THE INITIAL UNIT DENSE SYMMETRIC
*         MATRIX.
*  S   MXVCOP  COPYING OF A VECTOR.
*  S   MXVDIF  DIFFERENCE OF TWO VECTORS.
*  S   MXVINA  ABSOLUTE VALUES OF ELEMENTS OF AN INTEGER VECTOR.
*  S   MXVINC  UPDATE OF AN INTEGER VECTOR.
*  S   MXVIND  CHANGE OF THE INTEGER VECTOR FOR CONSTRAINT ADDITION.
*  S   MXVINT  CHANGE OF THE INTEGER VECTOR FOR TRUST REGION BOUND
*         ADDITION.
*  S   MXVMUL  DIAGONAL PREMULTIPLICATION OF A VECTOR.
*  S   MXVNEG  COPYING OF A VECTOR WITH CHANGE OF THE SIGN.
*  S   MXVSET  INITIATION OF A VECTOR.
*
      SUBROUTINE MXDRFP(NF,NC,CG,CB,X,CF,CFD,IC,ICA,CR,CZ,G,S,JOB,INF)
      INTEGER NF,NC,IC(NC),ICA(NF),JOB,INF
      REAL*8 X(NF),CF(NC),CFD(NC),CB(NC),CG(NF*NC),CR(NF*(NF+1)/2),
     & CZ(NF),G(NF),S(NF)
      REAL*8 POM,UMAX,GMAX,DMAX
      INTEGER N,IPOM,I,IOLD,JOLD,INEW,JNEW,MODE,IER,KREM,KC
      REAL*8 CON,CON7,CON9
      PARAMETER(CON=1.0D 60,CON7=1.0D-10,CON9=1.0D-10)
      INF=0
C
C     INITIATION
C
      IPOM=0
      JOLD=0
      JNEW=0
      KREM=0
```

```
      DMAX=0.0D 0
      CALL MXVSET(NF,0.0D 0,X)
      CALL MXVSET(NC,0.0D 0,CF)
      N=NF
      CALL MXVINS(NC,JOB,IC)
      DO 2 KC=1,NC
      INEW=0
      CALL MXDRFL(NC,KC,CF,IC,CB,CON9,INEW)
      CALL MXDRFA(NF,N,NC,ICA,CG,CR,S,CON7,GMAX,UMAX,INEW,IER)
      IF (IER.LE.1) CALL MXVIND(NC,IC,KC,0)
      IF (IC(KC).LT.-10) IPOM=1
    2 CONTINUE
    3 IF (IPOM.EQ.1) THEN
C
C     CHECK OF FEASIBILITY AND UPDATE OF THE FIRST PHASE OBJECTIVE
C     FUNCTION
C
      CALL MXDRFG(NF,NC,IC,CG,G,INEW)
      IF (INEW.EQ.0) IPOM=0
      ENDIF
      IF (IPOM.EQ.0) THEN
C
C     FEASIBILITY ACHIEVED
C
      INF=1
      GO TO 6
      ENDIF
C
C     LAGRANGE MULTIPLIERS DETERMINATION
C
    5 IF (NF.GT.N) THEN
      CALL MXDRFV(NF,N,NC,ICA,CG,G,CZ)
      CALL MXDPRB(NF-N,CR,CZ,0)
      CALL MXDRFT(NF,N,NC,IC,ICA,CZ,CON7,UMAX,IOLD)
      ELSE
      IOLD=0
      UMAX=0.0D 0
      ENDIF
C
C     PROJECTED GRADIENT DETERMINATION
C
      IF (N.GT.0) THEN
      CALL MXVNEG(NF,G,S)
      CALL MXDRFS(NF,N,NC,ICA,CG,CZ,S,CON7,GMAX)
      ELSE
      GMAX=0.0D 0
      ENDIF
      INEW=0
      IF (GMAX.EQ.0.0D 0) THEN
C
C     OPTIMUM ON A LINEAR MANIFOLD OBTAINED
C
      IF (IOLD.EQ.0) THEN
      IPOM=0
      DO 22 KC=1,NC
      IF (IC(KC).LT.-10) THEN
      INEW=0
      CALL MXDRFL(NC,KC,CF,IC,CB,CON9,INEW)
      IF (IC(KC).LT.-10) IPOM=1
      ENDIF
   22 CONTINUE
      IF (IPOM.EQ.0) THEN
C
C     OPTIMAL SOLUTION ACHIEVED
C
```

```
      INF= 2
      GO TO 6
      ELSE
C
C     FEASIBLE SOLUTION DOES NOT EXIST
C
      INF=-1
      GO TO 6
      ENDIF
      ELSE
C
C     CONSTRAINT DELETION
C
      CALL MXDRFR(NF,N,NC,IC,ICA,CR,S,IOLD,KREM,IER)
      DMAX=0.0D 0
      JOLD=IOLD
      JNEW=0
      GO TO 5
      ENDIF
      ELSE
C
C     STEPSIZE SELECTION
C
      POM=CON
      CALL MXDRFM(NF,NC,CF,CFD,IC,CB,CG,S,POM,KREM,INEW)
      IF (INEW.EQ.0) THEN
C
C     FEASIBLE SOLUTION DOES NOT EXIST
C
      INF=-3
      GO TO 6
      ELSE
C
C     STEP REALIZATION
C
      CALL MXVDIR(NF,POM,S,X,X)
      CALL MXDRFD(NC,CF,CFD,IC,POM)
C
C     CONSTRAINT ADDITION
C
      IF (INEW.GT.0) THEN
      KC=INEW
      INEW=0
      CALL MXDRFL(NC,KC,CF,IC,CB,CON9,INEW)
      CALL MXDRFA(NF,N,NC,ICA,CG,CR,S,CON7,GMAX,UMAX,INEW,IER)
      IF (IER.LE.1) CALL MXVIND(NC,IC,KC,0)
      ENDIF
      DMAX=POM
      JNEW=INEW
      JOLD=0
      GO TO 3
      ENDIF
      ENDIF
    6 CONTINUE
      RETURN
      END

* SUBROUTINE MXDRFR             ALL SYSTEMS                  05/12/01
* PURPOSE :
* OPERATIONS AFTER CONSTRAINT DELETION.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  II  N  ACTUAL NUMBER OF VARIABLES.
*  II  NC  NUMBER OF CONSTRAINTS.
```

```
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  II  ICA(NF+1)  VECTOR CONTAINING INDICES OF ACTIVE FUNCTIONS.
*  RI  CR((NF+1)*(NF+2)/2)  TRIANGULAR DECOMPOSITION OF KERNEL OF THE
*         ORTHOGONAL PROJECTION.
*  RA  G(NF+1)  AUXILIARY VECTOR.
*  II  IOLD  INDEX OF THE OLD ACTIVE CONSTRAINT.
*  IO  KREM  AUXILIARY VARIABLE.
*  IO  IER  ERROR INDICATOR.
*
      SUBROUTINE MXDRFR(NF,N,NC,IC,ICA,CR,G,IOLD,KREM,IER)
      INTEGER NF,N,NC,IC(NC),ICA(NF+1),IOLD,KREM,IER
      REAL*8 CR((NF+1)*(NF+2)/2),G(NF+1)
      INTEGER NCA,I,J,K,L,KC
      REAL*8 CK,CL
      NCA=NF-N
      IF (IOLD.LT.NCA) THEN
      K=IOLD*(IOLD-1)/2
      KC=ICA(IOLD)
      CALL MXVCOP(IOLD,CR(K+1),G)
      CALL MXVSET(NCA-IOLD,0.0D 0,G(IOLD+1))
      K=K+IOLD
      DO 2 I=IOLD+1,NCA
      K=K+I
      CALL MXVORT(CR(K-1),CR(K),CK,CL,IER)
      CALL MXVROT(G(I-1),G(I),CK,CL,IER)
      L=K
      DO 1 J=I,NCA-1
      L=L+J
      CALL MXVROT(CR(L-1),CR(L),CK,CL,IER)
    1 CONTINUE
    2 CONTINUE
      K=IOLD*(IOLD-1)/2
      DO 3 I=IOLD,NCA-1
      L=K+I
      ICA(I)=ICA(I+1)
      CALL MXVCOP(I,CR(L+1),CR(K+1))
      K=L
    3 CONTINUE
      ICA(NCA)=KC
      CALL MXVCOP(NCA,G,CR(K+1))
      ENDIF
      KREM=1
      N=N+1
      L=ICA(NF-N+1)
      IC(L)=-IC(L)
      RETURN
      END

* SUBROUTINE MXDRFS               ALL SYSTEMS                97/12/01
* PURPOSE :
* NEGATIVE PROJECTED GRADIENT IS DETERMINED USING LAGRANGE MULTIPLIERS.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  II  N  ACTUAL NUMBER OF VARIABLES.
*  II  NC  NUMBER OF LINEARIZED CONSTRAINTS.
*  II  ICA(NF+1)  VECTOR CONTAINING INDICES OF ACTIVE FUNCTIONS.
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RO  CZ(NF+1)  VECTOR OF LAGRANGE MULTIPLIERS.
*  RO  S(NF)  NEGATIVE PROJECTED GRADIENT OF THE QUADRATIC FUNCTION.
*  RI  EPS7  TOLERANCE FOR LINEAR AND QUADRATIC PROGRAMMING.
*  RO  GMAX  NORM OF THE TRANSFORMED GRADIENT.
*
* SUBPROGRAMS USED :
```

```
*  S   MXVDIR  VECTOR AUGMENTED BY THE SCALED VECTOR.
*  RF  MXVMAX  L-INFINITY NORM OF A VECTOR.
*
      SUBROUTINE MXDRFS(NF,N,NC,ICA,CG,CZ,S,EPS7,GMAX)
      INTEGER NF,N,NC,ICA(NF+1)
      REAL*8 CG(NF*NC),CZ(NF+1),S(NF),EPS7,GMAX
      REAL*8 MXVMAX
      INTEGER NCA,J,L
      NCA=NF-N
      DO 3 J=1,NCA
      L=ICA(J)
      CALL MXVDIR(NF,CZ(J),CG((L-1)*NF+1),S,S)
    3 CONTINUE
      GMAX=MXVMAX(NF,S)
      IF (GMAX.LE.EPS7) GMAX=0.0D 0
      RETURN
      END

* SUBROUTINE MXDRFT               ALL SYSTEMS                05/12/01
* PURPOSE :
* MAXIMUM ABSOLUTE VALUE OF THE NEGATIVE LAGRANGE MULTIPLIER IS
* COMPUTED.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
*  II  N  ACTUAL NUMBER OF VARIABLES.
*  II  NC  NUMBER OF LINEARIZED CONSTRAINTS.
*  II  IC(NC)  VECTOR CONTAINING TYPES OF CONSTRAINTS.
*  II  ICA(NF+1)  VECTOR CONTAINING INDICES OF ACTIVE FUNCTIONS.
*  RO  CZ(NF+1)  VECTOR OF LAGRANGE MULTIPLIERS.
*  RI  EPS7  TOLERANCE FOR LINEAR AND QUADRATIC PROGRAMMING.
*  RO  UMAX  MAXIMUM ABSOLUTE VALUE OF THE NEGATIVE LAGRANGE MULTIPLIER.
*  IO  IOLD  INDEX OF THE REMOVED CONSTRAINT.
*
      SUBROUTINE MXDRFT(NF,N,NC,IC,ICA,CZ,EPS7,UMAX,IOLD)
      INTEGER NF,N,NC,IC(NC),ICA(NF+1),IOLD
      REAL*8 CZ(NF+1),EPS7,UMAX
      REAL*8 TEMP
      INTEGER NCA,J,K,L
      IOLD=0
      UMAX=0.0D 0
      NCA=NF-N
      DO 2 J=1,NCA
      TEMP=CZ(J)
      L=ICA(J)
      K=IC(L)
      IF (K.EQ.-1.AND.UMAX+TEMP.GE.0.0D 0) THEN
      ELSE IF (K.EQ.-2.AND.UMAX-TEMP.GE.0.0D 0) THEN
      ELSE
      IOLD=J
      UMAX=ABS(TEMP)
      ENDIF
    2 CONTINUE
      IF (UMAX.LE.EPS7) IOLD=0
      RETURN
      END

* SUBROUTINE MXDRFV               ALL SYSTEMS                05/12/01
* PURPOSE :
* GRADIENT OF THE OBJECTIVE FUNCTION IS PREMULTIPLIED BY TRANSPOSE
* OF THE MATRIX WHOSE COLUMNS ARE NORMALS OF CURRENT ACTIVE CONSTRAINTS
* AND GRADIENTS OF CURRENT ACTIVE FUNCTIONS.
*
* PARAMETERS :
*  II  NF  DECLARED NUMBER OF VARIABLES.
```

```
*  II  N  ACTUAL NUMBER OF VARIABLES.
*  II  NC  NUMBER OF LINEARIZED CONSTRAINTS.
*  II  ICA(NF+1)  VECTOR CONTAINING INDICES OF ACTIVE FUNCTIONS.
*  RI  CG(NF*NC)  MATRIX WHOSE COLUMNS ARE NORMALS OF THE LINEAR
*         CONSTRAINTS.
*  RI  G(NF)  GRADIENT OF THE OBJECTIVE FUNCTION.
*  RO  GN(NF+1)  OUTPUT VECTOR.
*
* SUBPROGRAMS USED :
*  RF  MXVDOT  DOT PRODUCT OF TWO VECTORS.
*
      SUBROUTINE MXDRFV(NF,N,NC,ICA,CG,G,GN)
      INTEGER NF,N,NC,ICA(NF+1)
      REAL*8 CG(NF*NC),G(NF),GN(NF+1)
      REAL*8 MXVDOT
      INTEGER NCA,J,L
      NCA=NF-N
      DO 1 J=1,NCA
      L=ICA(J)
      GN(J)=MXVDOT(NF,CG((L-1)*NF+1),G)
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXDRMD               ALL SYSTEMS                91/12/01
* PURPOSE :
* MULTIPLICATION OF A ROWWISE STORED DENSE RECTANGULAR MATRIX A BY
* A VECTOR X AND ADDITION OF A SCALED VECTOR ALF*Y.
*
* PARAMETERS :
*  II  N  NUMBER OF COLUMNS OF THE MATRIX A.
*  II  M  NUMBER OF ROWS OF THE MATRIX A.
*  RI  A(M*N)  RECTANGULAR MATRIX STORED ROWWISE IN THE
*         ONE-DIMENSIONAL ARRAY.
*  RI  X(N)  INPUT VECTOR.
*  RI  ALF  SCALING FACTOR.
*  RI  Y(M)  INPUT VECTOR.
*  RO  Z(M)  OUTPUT VECTOR EQUAL TO A*X+ALF*Y.
*
      SUBROUTINE MXDRMD(N,M,A,X,ALF,Y,Z)
      INTEGER N,M
      REAL*8 A(M*N),X(N),ALF,Y(M),Z(M)
      REAL*8 TEMP
      INTEGER I,J,K
      K=0
      DO 2 J=1,M
      TEMP=ALF*Y(J)
      DO 1 I=1,N
      TEMP=TEMP+A(K+I)*X(I)
    1 CONTINUE
      Z(J)=TEMP
      K=K+N
    2 CONTINUE
      RETURN
      END

* SUBROUTINE MXDRMM               ALL SYSTEMS                91/12/01
* PURPOSE :
* MULTIPLICATION OF A ROWWISE STORED DENSE RECTANGULAR MATRIX A BY
* A VECTOR X.
*
* PARAMETERS :
*  II  N  NUMBER OF COLUMNS OF THE MATRIX A.
*  II  M  NUMBER OF ROWS OF THE MATRIX A.
*  RI  A(M*N)  RECTANGULAR MATRIX STORED ROWWISE IN THE
```

```
*         ONE-DIMENSIONAL ARRAY.
*  RI  X(N)  INPUT VECTOR.
*  RO  Y(M)  OUTPUT VECTOR EQUAL TO A*X.
*
      SUBROUTINE MXDRMM(N,M,A,X,Y)
      INTEGER N,M
      DOUBLE PRECISION A(*),X(*),Y(*)
      DOUBLE PRECISION TEMP
      INTEGER I,J,K
      K=0
      DO 2 J=1,M
      TEMP=0.0D 0
      DO 1 I=1,N
      TEMP=TEMP+A(K+I)*X(I)
    1 CONTINUE
      Y(J)=TEMP
      K=K+N
    2 CONTINUE
      RETURN
      END

* FUNCTION  MXDRMN               ALL SYSTEMS                91/12/01
* PURPOSE :
* EUCLIDEAN NORM OF A PART OF THE I-TH COLUMN OF A ROWWISE STORED DENSE
* RECTANGULAR MATRIX A IS COMPUTED.
*
* PARAMETERS :
*  II  N  NUMBER OF COLUMNS OF THE MATRIX A.
*  II  M  NUMBER OF ROWS OF THE MATRIX A.
*  RI  A(M*N)  RECTANGULAR MATRIX STORED ROWWISE IN THE
*         ONE-DIMENSIONAL ARRAY.
*  II  I  INDEX OF THE COLUMN WHOSE NORM IS COMPUTED.
*  II  J  INDEX OF THE FIRST ELEMENT FROM WHICH THE NORM IS COMPUTED.
*
      FUNCTION MXDRMN(N,M,A,I,J)
      INTEGER N,M,I,J
      REAL*8 A(M*N),MXDRMN
      REAL*8 POM,DEN
      INTEGER K,L
      REAL*8 ZERO
      PARAMETER (ZERO=0.0D 0)
      DEN=ZERO
      L=(J-1)*N
      DO 1 K=J,M
      DEN=MAX(DEN,ABS(A(L+I)))
      L=L+N
    1 CONTINUE
      POM=ZERO
      IF (DEN.GT.ZERO) THEN
      L=(J-1)*N
      DO 2 K=J,M
      POM=POM+(A(L+I)/DEN)**2
      L=L+N
    2 CONTINUE
      ENDIF
      MXDRMN=DEN*SQRT(POM)
      RETURN
      END

* SUBROUTINE MXDRQF              ALL SYSTEMS                92/12/01
* PURPOSE :
* QR DECOMPOSITION OF ROWWISE STORED DENSE RECTANGULAR MATRIX Q USING
* HOUSEHOLDER TRANSFORMATIONS WITHOUT PIVOTING.
*
* PARAMETERS :
```

```
*  II  N  NUMBER OF COLUMNS OF THE MATRIX Q.
*  II  M  NUMBER OF ROWS OF THE MATRIX Q.
*  RU  Q(M*N)  RECTANGULAR MATRIX STORED ROWWISE IN THE
*         ONE-DIMENSIONAL ARRAY.
*  RO  R(N*(N+1)/2)  UPPER TRIANGULAR MATRIX STORED IN THE PACKED FORM.
*
* SUBPROGRAMS USED :
*  S   MXDRMN  EUCLIDEAN NORM OF A PART OF THE ROWWISE STORED
*         RECTANGULAR MATRIX COLUMN.
*
* METHOD :
* P.A.BUSSINGER, G.H.GOLUB : LINEAR LEAST SQUARES SOLUTION BY
* HOUSEHOLDER TRANSFORMATION. NUMER. MATH. 7 (1965) 269-276.
*
      SUBROUTINE MXDRQF(N,M,Q,R)
      INTEGER N,M
      REAL*8 Q(M*N),R(N*(N+1)/2)
      REAL*8 ALF,POM,MXDRMN
      INTEGER I,J,K,L,JP,KP,NM
      REAL*8 ZERO,ONE
      PARAMETER (ZERO=0.0D 0,ONE=1.0D 0)
      NM=MIN(N,M)
C
C     QR DECOMPOSITION
C
      L=0
      KP=0
      DO 6 K=1,NM
      POM=MXDRMN(N,M,Q,K,K)
      IF (Q(KP+K).NE.ZERO) POM=SIGN(POM,Q(KP+K))
      R(L+K)=-POM
      JP=0
      DO 1 J=1,K-1
      R(L+J)=Q(JP+K)
      Q(JP+K)=ZERO
      JP=JP+N
    1 CONTINUE
      IF (POM.NE.ZERO) THEN
C
C     HOUSEHOLDER TRANSFORMATION
C
      DO 2 J=K,M
      Q(JP+K)=Q(JP+K)/POM
      JP=JP+N
    2 CONTINUE
      Q(KP+K)=Q(KP+K)+ONE
      DO 5 I=K+1,N
      ALF=ZERO
      JP=KP
      DO 3 J=K,M
      ALF=ALF+Q(JP+K)*Q(JP+I)
      JP=JP+N
    3 CONTINUE
      ALF=ALF/Q(KP+K)
      JP=KP
      DO 4 J=K,M
      Q(JP+I)=Q(JP+I)-ALF*Q(JP+K)
      JP=JP+N
    4 CONTINUE
    5 CONTINUE
      ENDIF
      L=L+K
      KP=KP+N
    6 CONTINUE
C
```

```
C     EXPLICIT FORMULATION OF THE ORTHOGONAL MATRIX
C
      KP=N*N
      DO 11 K=N,1,-1
      KP=KP-N
      IF (Q(KP+K).NE.ZERO) THEN
      DO 9 I=K+1,N
      ALF=ZERO
      JP=KP
      DO 7 J=K,M
      ALF=ALF+Q(JP+K)*Q(JP+I)
      JP=JP+N
    7 CONTINUE
      ALF=ALF/Q(KP+K)
      JP=KP
      DO 8 J=K,M
      Q(JP+I)=Q(JP+I)-ALF*Q(JP+K)
      JP=JP+N
    8 CONTINUE
    9 CONTINUE
      JP=KP
      DO 10 J=K,M
      Q(JP+K)=-Q(JP+K)
      JP=JP+N
   10 CONTINUE
      ENDIF
      Q(KP+K)=Q(KP+K)+ONE
   11 CONTINUE
      RETURN
      END

* SUBROUTINE MXVCOP                ALL SYSTEMS                88/12/01
* PURPOSE :
* COPYING OF A VECTOR.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  X(N)  INPUT VECTOR.
*  RO  Y(N)  OUTPUT VECTOR WHERE Y:= X.
*
      SUBROUTINE MXVCOP(N,X,Y)
      INTEGER N
      DOUBLE PRECISION X(*),Y(*)
      INTEGER I
      DO 10 I = 1,N
      Y(I) = X(I)
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXVDIF                ALL SYSTEMS                88/12/01
* PURPOSE :
* VECTOR DIFFERENCE.
*
* PARAMETERS :
*  RI  X(N)  INPUT VECTOR.
*  RI  Y(N)  INPUT VECTOR.
*  RO  Z(N)  OUTPUT VECTOR WHERE Z:= X - Y.
*
      SUBROUTINE MXVDIF(N,X,Y,Z)
      INTEGER N
      DOUBLE PRECISION X(*),Y(*),Z(*)
      INTEGER I
      DO 10 I = 1,N
      Z(I) = X(I) - Y(I)
```

```
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXVDIR               ALL SYSTEMS                91/12/01
* PURPOSE :
* VECTOR AUGMENTED BY THE SCALED VECTOR.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  A  SCALING FACTOR.
*  RI  X(N)  INPUT VECTOR.
*  RI  Y(N)  INPUT VECTOR.
*  RO  Z(N)  OUTPUT VECTOR WHERE Z:= Y + A*X.
*
      SUBROUTINE MXVDIR(N,A,X,Y,Z)
      DOUBLE PRECISION A
      INTEGER N
      DOUBLE PRECISION X(*),Y(*),Z(*)
      INTEGER I
      DO 10 I = 1,N
      Z(I) = Y(I) + A*X(I)
   10 CONTINUE
      RETURN
      END

* FUNCTION MXVDOT                 ALL SYSTEMS                91/12/01
* PURPOSE :
* DOT PRODUCT OF TWO VECTORS.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  X(N)  INPUT VECTOR.
*  RI  Y(N)  INPUT VECTOR.
*  RR  MXVDOT  VALUE OF DOT PRODUCT MXVDOT=TRANS(X)*Y.
*
      DOUBLE PRECISION FUNCTION MXVDOT(N,X,Y)
      INTEGER N
      DOUBLE PRECISION X(*),Y(*)
      DOUBLE PRECISION TEMP
      INTEGER I
      TEMP = 0.0D0
      DO 10 I = 1,N
      TEMP = TEMP + X(I)*Y(I)
   10 CONTINUE
      MXVDOT = TEMP
      RETURN
      END

* SUBROUTINE MXVIND               ALL SYSTEMS                91/12/01
* PURPOSE :
* CHANGE OF THE INTEGER VECTOR ELEMENT FOR THE CONSTRAINT ADDITION.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  IU  IX(N)  INTEGER VECTOR.
*  II  I  INDEX OF THE CHANGED ELEMENT.
*  II JOB  CHANGE SPECIFICATION. IF JOB.EQ.0 THEN IX(I)=1O-IX(I).
*
      SUBROUTINE MXVIND(N,IX,I,JOB)
      INTEGER N,IX(N),I,JOB
      IF (JOB.EQ.0) IX(I)=1O-IX(I)
      RETURN
      END
```

```
* SUBROUTINE MXVINP             ALL SYSTEMS                   91/12/01
* PURPOSE :
* INITIATION OF A INTEGER PERMUTATION VECTOR.
*
* PARAMETERS :
*  II  N DIMENSION OF THE INTEGER VECTOR.
*  IO  IP(N)  INTEGER VECTOR SUCH THAT IP(I)=I FOR ALL I.
*
      SUBROUTINE MXVINP(N,IP)
      INTEGER N,IP(N)
      INTEGER I
      DO 1 I=1,N
      IP(I)=I
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXVINS             ALL SYSTEMS                   90/12/01
* PURPOSE :
* INITIATION OF THE INTEGER VECTOR.
*
* PARAMETERS :
*  II  N DIMENSION OF THE INTEGER VECTOR.
*  II  IP  INTEGER PARAMETER.
*  IO  IX(N)  INTEGER VECTOR SUCH THAT IX(I)=IP FOR ALL I.
*
      SUBROUTINE MXVINS(N,IP,IX)
      INTEGER N,IP,IX(N)
      INTEGER I
      DO 1 I=1,N
      IX(I)=IP
    1 CONTINUE
      RETURN
      END

* SUBROUTINE MXVNEG               ALL SYSTEMS               88/12/01
* PURPOSE :
* CHANGE THE SIGNS OF VECTOR ELEMENTS.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  X(N)  INPUT VECTOR.
*  RO  Y(N)  OUTPUT VECTOR WHERE Y:= - X.
*
      SUBROUTINE MXVNEG(N,X,Y)
      INTEGER N
      DOUBLE PRECISION X(*),Y(*)
      INTEGER I
      DO 10 I = 1,N
      Y(I) = -X(I)
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXVSAV               ALL SYSTEMS               91/12/01
* PURPOSE :
* DIFFERENCE OF TWO VECTORS RETURNED IN THE SUBSTRACTED ONE.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  X(N)  INPUT VECTOR.
*  RU  Y(N)  UPDATE VECTOR WHERE Y:= X - Y.
*
      SUBROUTINE MXVSAV(N,X,Y)
      INTEGER N
```

```
      DOUBLE PRECISION X(*),Y(*)
      DOUBLE PRECISION TEMP
      INTEGER I
      DO 10 I = 1,N
      TEMP = Y(I)
      Y(I) = X(I) - Y(I)
      X(I) = TEMP
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXVSCL               ALL SYSTEMS               88/12/01
* PURPOSE :
* SCALING OF A VECTOR.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  X(N)  INPUT VECTOR.
*  RI  A  SCALING FACTOR.
*  RO  Y(N)  OUTPUT VECTOR WHERE Y:= A*X.
*
      SUBROUTINE MXVSCL(N,A,X,Y)
      DOUBLE PRECISION A
      INTEGER N
      DOUBLE PRECISION X(*),Y(*)
      INTEGER I
      DO 10 I = 1,N
      Y(I) = A*X(I)
   10 CONTINUE
      RETURN
      END

* SUBROUTINE MXVSET               ALL SYSTEMS               88/12/01
* PURPOSE :
* A SCALAR IS SET TO ALL THE ELEMENTS OF A VECTOR.
*
* PARAMETERS :
*  II  N  VECTOR DIMENSION.
*  RI  A  INITIAL VALUE.
*  RO  X(N)  OUTPUT VECTOR SUCH THAT X(I)=A FOR ALL I.
*
      SUBROUTINE MXVSET(N,A,X)
      DOUBLE PRECISION A
      INTEGER N
      DOUBLE PRECISION X(*)
      INTEGER I
      DO 10 I = 1,N
      X(I) = A
   10 CONTINUE
      RETURN
      END
```

7 Použití programu PNEQ na řešení úloh chemické rovnováhy

7.1 Hlavní program

```
*
*     PROGRAM STROFEC
*
      INTEGER N,NK
      PARAMETER(N=16,NK=108)
      INTEGER IW1(NK),IW2(N)
```

```
      REAL*8 WX(N),WK(NK),WB(N*NK),WH(N*NK),WC(N),WS(N)
      REAL*8 WY(N),WZ(NK),WD(NK),WTMP
      REAL*8 WX1(N),WX2(N),WX3(N),WK1(NK),WK2(NK),WW(N*(N+1)/2)
      COMMON /STROF/ WX,WK,WB,WH,WC,WS,WY,WZ,WD,WTMP
      INTEGER NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
      COMMON /STAT/ NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
      INTEGER NRA,IPAR(4),ITERM,I,INF,IWA(N)
      REAL*8 X(N),AF(N),RA((N+6)*N+N*(N+1)/2),RPAR(6),F,GMAX
      REAL*8 WR(N),WA(N),WALF,WBET
      NRA=(N+6)*N+N*(N+1)/2
*
*     CHOICE OF INTEGER AND REAL PARAMETERS
*
      DO 10 I = 1,4
      IPAR(I) = 0
   10 CONTINUE
      IPAR(1) = 2
      IPAR(4) = 1
      DO 20 I = 1,6
      RPAR(I) = 0.0D0
   20 CONTINUE
      RPAR(2) = 1.0D-16
      RPAR(3) = 1.0D-16
      RPAR(4) = 1.0D-16
      RPAR(5) = 1.0D-16
*
*     INPUT DATA
*
      OPEN (11,FILE='Z.DAT',STATUS='UNKNOWN')
      OPEN (12,FILE='K.DAT',STATUS='UNKNOWN')
      OPEN (13,FILE='B.DAT',STATUS='UNKNOWN')
      OPEN (14,FILE='H.DAT',STATUS='UNKNOWN')
      OPEN (15,FILE='S.DAT',STATUS='UNKNOWN')
      READ (11,*) (WX(I),I=1,N)
      READ (12,*) (WK(I),I=1,NK)
      READ (13,*) (WB(I),I=1,N*NK)
      READ (14,*) (WH(I),I=1,N*NK)
      READ (15,*) (WS(I),I=1,N)
*
*     INITIAL ESTIMATION OF VARIABLES
*
      CALL MXVNEG(NK,WK,WK)
      CALL MXDRFP(N,NK,WH,WK,X,WK1,WK2,IW1,IW2,WW,WX1,WX2,WX3,2,INF)
      CALL MXVNEG(NK,WK,WK)
      WTMP=LOG(1.0D1)
*
*     SOLUTION
*
      CALL PNEQU(N,X,AF,RA,NRA,IPAR,RPAR,F,GMAX,ITERM)
      DO 100 I=1,N
      X(I)=1.0D1**X(I)
  100 CONTINUE
  200 FORMAT ( ' ', 'X =',4D18.10:/(4X,4D18.10))
      WRITE(6,200) (X(I),I=1,N)
      STOP
      END
*
*     USER SUPPLIED SUBROUTINES (CALCULATION OF FA AND GA)
*
      SUBROUTINE FUN(NF,KA,X,FA)
      INTEGER N,NK
      PARAMETER(N=16,NK=108)
      REAL*8 WX(N),WK(NK),WB(N*NK),WH(N*NK),WC(N),WS(N)
      REAL*8 WY(N),WZ(NK),WD(NK),WTMP
      COMMON /STROF/ WX,WK,WB,WH,WC,WS,WY,WZ,WD,WTMP
```

```
      INTEGER NF,KA,I
      REAL*8 X(N),FA,AF(N)
      IF (KA.EQ.1) THEN
      CALL MXDRMD(N,NK,WH,X,1.0D0,WK,WZ)
      DO 11 I=1,NK
      WD(I)=1.0D1**WZ(I)
   11 CONTINUE
      CALL MXDCMM(N,NK,WB,WD,AF)
      ENDIF
      FA=AF(KA)-WS(KA)
      RETURN
      END
      SUBROUTINE DFUN(NF,KA,X,GA)
      INTEGER N,NK
      PARAMETER(N=16,NK=108)
      REAL*8 WX(N),WK(NK),WB(N*NK),WH(N*NK),WC(N),WS(N)
      REAL*8 WY(N),WZ(NK),WD(NK),WTMP
      COMMON /STROF/ WX,WK,WB,WH,WC,WS,WY,WZ,WD,WTMP
      INTEGER NF,KA,I
      REAL*8 X(N),GA(N),AG(N*N)
      IF (KA.EQ.1) THEN
      CALL MXVSET(N*N,0.0D0,AG)
      DO 12 I=1,NK
      CALL MXDCMU(N,N,AG,WTMP*WD(I),WB((I-1)*N+1),WH((I-1)*N+1))
   12 CONTINUE
      ENDIF
      CALL MXVCOP(N,AG((KA-1)*N+1),GA)
      RETURN
      END
```

7.2 Výsledky

```
ENTRY TO PNEQ :
NIT=    0  NFV=    1  NFG=    0  F = 0.20074686D+04
NIT=    1  NFV=    2  NFG=    1  F = 0.39733490D+03
NIT=    2  NFV=    3  NFG=    2  F = 0.15047002D+03
NIT=    3  NFV=    4  NFG=    3  F = 0.90950506D+02
NIT=    4  NFV=    5  NFG=    4  F = 0.69358750D+02
NIT=    5  NFV=    6  NFG=    5  F = 0.57889997D+02
NIT=    6  NFV=    7  NFG=    6  F = 0.49871445D+02
NIT=    7  NFV=    8  NFG=    7  F = 0.38258544D+02
NIT=    8  NFV=    9  NFG=    8  F = 0.29869472D+02
NIT=    9  NFV=   10  NFG=    9  F = 0.18722318D+02
NIT=   10  NFV=   11  NFG=   10  F = 0.12057233D+02
NIT=   11  NFV=   12  NFG=   11  F = 0.79653125D+01
NIT=   12  NFV=   13  NFG=   12  F = 0.54070501D+01
NIT=   13  NFV=   14  NFG=   13  F = 0.27265052D+01
NIT=   14  NFV=   15  NFG=   14  F = 0.15275883D+01
NIT=   15  NFV=   16  NFG=   15  F = 0.90398200D+00
NIT=   16  NFV=   17  NFG=   16  F = 0.54501919D+00
NIT=   17  NFV=   18  NFG=   17  F = 0.33151072D+00
NIT=   18  NFV=   19  NFG=   18  F = 0.20320801D+00
NIT=   19  NFV=   20  NFG=   19  F = 0.12561674D+00
NIT=   20  NFV=   21  NFG=   20  F = 0.78377149D-01
NIT=   21  NFV=   22  NFG=   21  F = 0.49382169D-01
NIT=   22  NFV=   23  NFG=   22  F = 0.20194974D-01
NIT=   23  NFV=   24  NFG=   23  F = 0.85176141D-02
NIT=   24  NFV=   25  NFG=   24  F = 0.36633444D-02
NIT=   25  NFV=   26  NFG=   25  F = 0.15961364D-02
NIT=   26  NFV=   27  NFG=   26  F = 0.70269894D-03
NIT=   27  NFV=   28  NFG=   27  F = 0.31232385D-03
NIT=   28  NFV=   29  NFG=   28  F = 0.14015606D-03
NIT=   29  NFV=   30  NFG=   29  F = 0.63562975D-04
NIT=   30  NFV=   31  NFG=   30  F = 0.29183579D-04
```

```
 NIT=   31  NFV=   32  NFG=   31  F = 0.13593198D-04
 NIT=   32  NFV=   33  NFG=   32  F = 0.64358336D-05
 NIT=   33  NFV=   34  NFG=   33  F = 0.31030011D-05
 NIT=   34  NFV=   35  NFG=   34  F = 0.15234825D-05
 NIT=   35  NFV=   36  NFG=   35  F = 0.76347931D-06
 NIT=   36  NFV=   37  NFG=   36  F = 0.39725136D-06
 NIT=   37  NFV=   38  NFG=   37  F = 0.32377811D-06
 NIT=   38  NFV=   39  NFG=   38  F = 0.24453054D-06
 NIT=   39  NFV=   40  NFG=   39  F = 0.17214480D-06
 NIT=   40  NFV=   41  NFG=   40  F = 0.13494857D-06
 NIT=   41  NFV=   42  NFG=   41  F = 0.12841750D-06
 NIT=   42  NFV=   44  NFG=   42  F = 0.12244119D-06
 NIT=   43  NFV=   45  NFG=   43  F = 0.11787536D-06
 NIT=   44  NFV=   46  NFG=   44  F = 0.11343056D-06
 NIT=   45  NFV=   47  NFG=   45  F = 0.10839086D-06
 NIT=   46  NFV=   48  NFG=   46  F = 0.10210860D-06
 NIT=   47  NFV=   50  NFG=   47  F = 0.98614875D-07
 NIT=   48  NFV=   51  NFG=   48  F = 0.95116547D-07
 NIT=   49  NFV=   52  NFG=   49  F = 0.92244320D-07
 NIT=   50  NFV=   53  NFG=   50  F = 0.87940719D-07
 NIT=   51  NFV=   55  NFG=   51  F = 0.79599761D-07
 NIT=   52  NFV=   56  NFG=   52  F = 0.63107675D-07
 NIT=   53  NFV=   57  NFG=   53  F = 0.37161009D-07
 NIT=   54  NFV=   58  NFG=   54  F = 0.20051633D-07
 NIT=   55  NFV=   59  NFG=   55  F = 0.10402284D-07
 NIT=   56  NFV=   60  NFG=   56  F = 0.52842419D-08
 NIT=   57  NFV=   61  NFG=   57  F = 0.27402863D-08
 NIT=   58  NFV=   62  NFG=   58  F = 0.16560038D-08
 NIT=   59  NFV=   63  NFG=   59  F = 0.12402948D-08
 NIT=   60  NFV=   64  NFG=   60  F = 0.91399954D-09
 NIT=   61  NFV=   65  NFG=   61  F = 0.79873436D-09
 NIT=   62  NFV=   66  NFG=   62  F = 0.65141266D-09
 NIT=   63  NFV=   67  NFG=   63  F = 0.56887978D-09
 NIT=   64  NFV=   68  NFG=   64  F = 0.46997132D-09
 NIT=   65  NFV=   69  NFG=   65  F = 0.39534102D-09
 NIT=   66  NFV=   70  NFG=   66  F = 0.31491800D-09
 NIT=   67  NFV=   71  NFG=   67  F = 0.24322420D-09
 NIT=   68  NFV=   72  NFG=   68  F = 0.17000035D-09
 NIT=   69  NFV=   73  NFG=   69  F = 0.10167577D-09
 NIT=   70  NFV=   74  NFG=   70  F = 0.28643723D-10
 NIT=   71  NFV=   75  NFG=   71  F = 0.18509182D-12
 NIT=   72  NFV=   76  NFG=   72  F = 0.87582390D-18
 EXIT FROM PNEQ :
 NIT=   72  NFV=   76  NFG=   72  F = 0.87582390D-18  ITERM=  3
 X =  0.7316375637D-09  0.1435448623D-18  0.2333503067D-17  0.1000010129D-04
      0.9746949033D-05  0.1176228029D-05  0.9991458937D-05  0.1234573213D-23
      0.9998929280D-05  0.9632625874D-05  0.9998784199D-05  0.6365895629D-81
      0.5514607218D-08  0.8318432586D-05  0.1341640066D-04  0.1059007027D+07
```

8 Použití systému UFO na řešení úloh chemické rovnováhy

8.1 Vstupní šablona

```
$NF=16
$NK=108
INTEGER IW1($NK),IW2($NF)
$FLOAT WX($NF),WK($NK),WB($NF*$NK),WH($NF*$NK),WC($NF),WS($NF)
$FLOAT WY($NF),WZ($NK),WD($NK),WR($NF),WM($NF),WTMP
$FLOAT WX1($NF),WX2($NF),WX3($NF),WK1($NK),WK2($NK),WW($NF*($NF+1)/2)
$SET(INPUT)
  NK=$NK
  OPEN (11,FILE='STROF\X.DAT',STATUS='UNKNOWN')
```

```
  OPEN (12,FILE='STROF\K.DAT',STATUS='UNKNOWN')
  OPEN (13,FILE='STROF\B.DAT',STATUS='UNKNOWN')
  OPEN (14,FILE='STROF\H.DAT',STATUS='UNKNOWN')
  OPEN (15,FILE='STROF\S.DAT',STATUS='UNKNOWN')
  READ (11,*) (WX(I),I=1,NF)
  READ (12,*) (WK(I),I=1,NK)
  READ (13,*) (WB(I),I=1,NF*NK)
  READ (14,*) (WH(I),I=1,NF*NK)
  READ (15,*) (WS(I),I=1,NF)
  CALL UXVNEG(NK,WK,WK)
  CALL UXDRFP(NF,NK,WH,WK,X,WK1,WK2,IW1,IW2,WW,WX1,WX2,WX3,2,IER)
  CALL UXVNEG(NK,WK,WK)
  WTMP=LOG(1.0D1)
$ENDSET
$SET(FMODELF)
  CALL UXDRMD(NF,NK,WH,X,ONE,WK,WZ)
  DO 11 I=1,NK
  WD(I)=TEN**WZ(I)
11 CONTINUE
  CALL UXDCMD(NF,NK,WB,WD,ONE,WS,WR)
  FF=HALF*UXVDOT(NF,WR,WR)
$ENDSET
$SET(OUTPUT)
  CALL UXDRMD(NF,NK,WH,X,ONE,WK,WZ)
  DO 100 I=1,NF
  WM(I)=TEN**X(I)
100 CONTINUE
200 FORMAT ( ' ', 'WM =',4$P 18.10:/(5X,4$P 18.10))
  WRITE(6,200) (WM(I),I=1,NF)
  WRITE(2,200) (WM(I),I=1,NF)
$ENDSET
$MOUT=-2
$NOUT=1
$BATCH
$GLOBAL

$REM ------------------ THE FIRST METHOD  -----------------
$TOLB='1.0$P 1'
$MIT=800
$MFV=8000
$CLASS='HM'
$TYPE='P'
$XDEL='1.0$P-2'
$INITIATION
$INPUT
$METHOD

$METERASE
$REM ------------------ THE SECOND METHOD -----------------
$TOLB='1.0$P-16'
$MIT=800
$MFV=8000
$CLASS='VM'
$TYPE='L'
$DECOMP='I'
$NUMBER=1
$METHOD

$END
```

8.2 Výsledky

```
 CLASS = HM - PN1   UPDATE = N   MODEL = FF   HESF = N   NF =      16
     NIT=    0 NFV=    1 F=  2007.537532
```

```
    NIT=    1 NFV=   34 F=  276.3278179
    NIT=    2 NFV=   51 F=  60.59700450
    NIT=    3 NFV=   68 F=  18.56202751
    NIT=    4 NFV=   85 F=  4.250878433
  0 NIT=    4 NFV=   85 NDC=    0 NCG=   0 F=0.425D+01
FF =  0.4250878433D+01
X  = -0.4329518674D+01 -0.5243407562D+01 -0.1096818534D+02 -0.2434075625D+00
     -0.2543407563D+01 -0.8752185340D+01 -0.4176740896D+01 -0.1594340756D+02
     -0.1093407563D+01 -0.2613407562D+01 -0.9434075625D+00 -0.7956929645D+02
     -0.1162418534D+02 -0.2434075625D+00 -0.2434075625D+00 -0.2434075625D+00
CLASS = CD - LV1   UPDATE = N   MODEL = FF   HESF = N   NF =      16
    NIT=    0 NFV=   17 NFG=    0  F=  4.250878433     G=0.161D+02
    NIT=    1 NFV=   35 NFG=    0  F=  2.002570412     G=0.355D+02
    NIT=    2 NFV=   52 NFG=    0  F= 0.4396123050E-01 G=0.101D+00
    NIT=    3 NFV=   69 NFG=    0  F= 0.4246610583E-01 G=0.973D-01
    NIT=    4 NFV=   86 NFG=    0  F= 0.1603195398E-01 G=0.364D-01
    NIT=    5 NFV=  103 NFG=    0  F= 0.8995324748E-02 G=0.202D-01
    NIT=    6 NFV=  120 NFG=    0  F= 0.4407778301E-02 G=0.968D-02
    NIT=    7 NFV=  137 NFG=    0  F= 0.2312375981E-02 G=0.491D-02
    NIT=    8 NFV=  154 NFG=    0  F= 0.1204409194E-02 G=0.242D-02
    NIT=    9 NFV=  171 NFG=    0  F= 0.6378589564E-03 G=0.120D-02
    NIT=   10 NFV=  188 NFG=    0  F= 0.3367521476E-03 G=0.584D-03
    NIT=   11 NFV=  205 NFG=    0  F= 0.1778379866E-03 G=0.320D-03
    NIT=   12 NFV=  222 NFG=    0  F= 0.9419502354E-04 G=0.175D-03
    NIT=   13 NFV=  239 NFG=    0  F= 0.4957868582E-04 G=0.767D-04
    NIT=   14 NFV=  256 NFG=    0  F= 0.2591113498E-04 G=0.444D-04
    NIT=   15 NFV=  273 NFG=    0  F= 0.1328669208E-04 G=0.167D-04
    NIT=   16 NFV=  290 NFG=    0  F= 0.6785032686E-05 G=0.106D-04
    NIT=   17 NFV=  307 NFG=    0  F= 0.3492774560E-05 G=0.403D-05
  0 NIT=   17 NFV=  307 NFG=    0 NDC=    0 NCG=   0 F=0.349D-05 G=0.403D-05
FF =  0.3492774560D-05
X  = -0.5093169492D+01 -0.8822659918D+01 -0.1122605531D+02 -0.2880304109D+01
     -0.3885879562D+01 -0.1025704932D+02 -0.5860480440D+01 -0.1646302522D+02
     -0.3004848924D+01 -0.3959570094D+01 -0.3002729794D+01 -0.7956964279D+02
     -0.1318350097D+02 -0.2880669249D+01 -0.1049627526D+02 -0.1283974254D+01
```

Reference

[1] Bogle, I.D.L., and Perkins, J.D., *A New Sparsity Preserving Quasi-Newton Update for Solving Nonlinear Equations,* SIAM Journal on Scientific and Statistical Computations, Vol 11, pp. 621-630, 1990.

[2] Brown, P.N., *A Local Convergence Theory for Combined Inexact-Newton/Finite-Difference Projection Methods,* SIAM Journal on Numerical Analysis, Vol. 24, pp. 407-434, 1987.

[3] Brown, P.N., and Saad, Y., *Hybrid Krylov Methods for Nonlinear Systems of Equations,* SIAM Journal on Scientific and Statistical Computations, Vol. 11, pp. 450-481, 1990.

[4] Brown, P.N., and Saad, Y., *Convergence Theory of Nonlinear Newton-Krylov Algorithms,* SIAM Journal on Optimization, Vol. 4, pp. 297-330, 1994.

[5] Broyden, C.G., *A Class of Methods for Solving Simultaneous Equations,* Mathematics of Computation, Vol. 19, pp. 577-593, 1965.

[6] Broyden, C.G., *The Convergence of an Algorithm for Solving Sparse Nonlinear Systems,* Mathematics of Computation, Vol. 25, pp. 285-294, 1971.

[7] Byrd, R.H, and Nocedal J., and Schnabel R.B., *Representations of Quasi-Newton Matrices and their Use in Limited Memory Methods,* Mathematical Programming, Vol. 63, pp. 129-136, 1994.

[8] Coleman, T.F., and Mor, J.S., *Estimation of Sparse Jacobian and Graph Coloring Problem,* SIAM Journal on Numerical Analysis, Vol. 20, pp. 187-209, 1983.

[9] Coleman, T.F., and Garbow, B.S., and Mor, J.S., *Software for Estimating Sparse Jacobian Matrices,* ACM Transactions of Mathematical Software, Vol. 10, pp. 329-345, 1984.

[10] Curtis, A.R., and Powell, M.J.D., and Reid, J.K., *On the estimation of sparse Jacobian matrices,* IMA Journal of Aplied Mathematics, Vol. 13, pp. 117-119, 1974.

[11] Dembo, R.S, Eisenstat, S.C., and Steihaug T., *Inexact Newton Methods,* SIAM J. on Numerical Analysis, Vol. 19, pp. 400-408, 1982.

[12] Dembo, R.S., and Steihaug T., *Truncated Newton Algorithms for Large-Scale Optimization,* Mathematical Programming, Vol. 26, pp. 190-212, 1983.

[13] Dennis, J.E., and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[14] Eisenstat, S.C., and Walker, H.F., *Globally convergent Inexact Newton Methods,* SIAM Journal on Optimization, Vol. 4, pp. 393-422, 1994.

[15] Eisenstat, S.C., and Walker, H.F., *Choosing the Forcing Terms in an Inexact Newton Method,* SIAM Journal on Scientific Computation, Vol. 17, pp. 16-32, 1996.

[16] Gomez-Ruggiero, M.A., and Martinez, J.M., and Moretti, A.C., *Comparing Algorithms for Solving Sparse Nonlinear Systems of Equations,* SIAM Journal on Scientific and Statistical Computations, Vol. 13, pp. 459-483, 1992.

[17] Kelley, C.T., *Iterative Methods for Linear and Nonlinear Equations,* SIAM, Philadelphia, Pennsylvania, 1995.

[18] Li, G., *Successive Column Correction Algorithms for Solving Sparse Nonlinear Systems of Equations,* Mathematical Programming, Vol. 43, pp. 187-207, 1989.

[19] Lukšan, L., *Inexact Trust Region Method for Large Sparse Systems of Nonlinear Equations,* Journal of Optimization Theory and Applications, Vol. 81, pp. 569-590, 1994.

[20] Lukšan, L., Spedicato, E.: Variable metric methods for unconstrained optimization and nonlinear least squares. Journal of Computational and Applied Mathematics, Vol. 124, pp. 61-93, 2000.

[21] Lukšan, L., Vlček, J. Sparse and partially separable test problems for unconstrained and equality constrained optimization. Report V-767, Prague, ICS AS CR, 1998.

[22] Lukšan, L., Vlček, J.: Computational Experience with Globally Convergent Descent Methods for Large Sparse Systems of Nonlinear Equations. Optimization Methods and Software 8 (1998) 201-223.

[23] Lukšan, L., Šiška, M., Tůma, M., Vlček, J., Ramešová, N., *Interactive System for Universal Functional Optimization (UFO), Version 2004,* Research Report No. V-923, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 1994.

[24] Martinez, J.M., *SOR-Secant Methods,* SIAM Journal on Numerical Analysis, Vol. 31, pp. 217-226, 1994.

[25] Martinez, J.M., and Zambaldi, M.C., *An Inverse Column-Updating Method for Solving Large-Scale Nonlinear Systems of Equations,* Optimization Methods and Software, Vol. 1, pp. 129-140, 1992.

[26] Saad, Y., Schultz, M., *GMRES a Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,* SIAM Journal on Scientific and Statistical Computations, Vol. 7, pp. 856-869, 1986.

[27] Schubert, L.K., *Modification of a Quasi-Newton Method for Nonlinear Equations with Sparse Jacobian,* Mathematics of Computation, Vol. 24, pp. 27-30, 1970.

[28] Sonneveld, P., *CGS, a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems,* SIAM Journal on Scientific and Statistical Computations, Vol. 10, pp. 36-52, 1989.

[29] Tong, C.H., *A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems,* Sandia National Laboratories, Sandia Report No. SAND91-8240B, Livermore, 1992.