Fachbereich Mathematik
Prof. Dr. Stefan Ulbrich
Dipl.-Math. Christian Brandenburg

TECHNISCHE UNIVERSITÄT DARMSTADT

25./26. Jan. 2007

Programmierübung

Gruppenübung

G 1 Lösen linearer Gleichungssysteme

Eine wichtige Rolle beim effizienten Implementieren des Simplex-Algorithmus spielt das Lösen linearer Gleichungssysteme. Ein erster Schritt ist es, das direkte Lösen der Gleichungssysteme durch eine $LU-$Fakorisierung zu ersetzen. Da es jedoch ebenfalls teuer ist in jeder Iteration eine $LU-$Fakorisierung durchzuführen, werden geeignete Update-Strategien eingesetzt (siehe Skript Kapitel 5.4).

Wandeln Sie Ihre MATLAB-Implementierung des Simplex Algorithmus aus Hausübung H23 wie folgt ab (alternativ können Sie auch die Datei `simplex.m` im Verzeichnis *Übungen* auf der Veranstaltungshomepage benutzen):

- *LU-Faktorisierung:*
  Führen Sie in jeder l-ten Iteration, $l \in 100 \cdot \mathbb{N}_0 + 1$, eine LU-Faktorisierung durch und lösen Sie mit Hilfe dieser die linearen Gleichungssysteme $(A_B^{(l)})^T y^{(l)} = c_B^{(l)}$ und $A_B^{(l)} w^{(l)} = A_{\cdot j}^{(l)}$ in *BTRAN* bzw *FTRAN*. Benutzen Sie dazu den MATLAB-Befehl `lu`.
- *Update-Formeln:*
  In den übrigen Iterationen sollen die Gleichungssysteme $(A_B^{(k)})^T y^{(k)} = c_B^{(k)}$ und $A_B^{(k)} w^{(k)} = A_{\cdot j}^{(k)}$ mittels Update-Formeln gelöst werden. Diese lauten wie folgt für die k-te Iteration, wobei zum Lösen mit $A_B^{(l)}$ die $LU-$Faktorisierung genutzt werden soll:
  - **BTRAN:**

$$\bar{c}^{(k)} = E_{l+1}^T E_{l+2}^T \cdots E_k^T c_B^{(k)} \tag{1}$$

$$(A_B^{(l)})^T y^{(k)} = \bar{c}^{(k)} \tag{2}$$

  - **FTRAN:**

$$A_B^{(l)} \bar{w}^{(k)} = A_{\cdot j}^{(k)} \tag{3}$$

$$w^{(k)} = E_k E_{k-1} \cdots E_{l+1} \bar{w}^{(k)} \tag{4}$$

E_k ist dabei jeweils definiert durch

$$E_k = \begin{array}{c} \begin{pmatrix} 1 & & & \eta_1^{(k-1)} & & & \\ & \ddots & & \vdots & & 0 & \\ & & 1 & \vdots & & & \\ & & & \eta_i^{(k-1)} & & & \\ & & & \vdots & 1 & & \\ & 0 & & \vdots & & \ddots & \\ & & & \eta_m^{(k-1)} & & & 1 \end{pmatrix} \\ \begin{matrix} \uparrow \\ i \end{matrix} \end{array} \tag{5}$$

mit

$$\eta_j^{(k-1)} = \begin{cases} \dfrac{1}{w_i^{(k-1)}}, & \text{falls } j = i \\ -\dfrac{w_j^{(k-1)}}{w_i^{(k-1)}}, & \text{sonst.} \end{cases},$$

wobei $w^{(k-1)}$ die Lösung aus *FTRAN* in der $(k-1)$-ten Iteration bezeichne.

Hinweis: Es gibt zwei mögliche Vorgehensweisen: Speichern Sie entweder in jeder Iteration k das Matrixprodukt $E_k E_{k-1} \cdots E_{l+1}$ (vorteilhaft, falls die Produkte dünn besetzt bleiben, da dann wenig Speicher verbraucht wird), oder speichern sie alle k Matrizen $E_{l+1} \dots E_k$ (als `sparse`-Matrizen) und bilden sie in jeder Iteration k Matrix-Vektor-Produkte.

Testen Sie nun Ihr Programm mit dem Problem aus der Mittelseminaraufgabe. Modifizieren Sie dazu Ihren Algorithmus falls nötig wie folgt (in der Datei `simplex.m` auf der Homepage sind diese Änderungen bereits vorgenommen):

- fügen Sie im Funktionskopf den Ausgabeparameter `B_opt` hinzu, der die optimale Basis zurückgibt (wird zur Berechnung einer zulässigen Startbasis durch die Phase I der Simplex benötigt).
- wählen Sie im Pricing den Index j mit den negativsten reduzierten Kosten.
- da es beim Rechnen mit Computern zu numerischen Ungenauigkeiten kommt, liefern logische Ausdrücke der Form `(x == 0)` oder `(z(N) >= 0)` in der Regel nicht das gewünschte Ergebnis. Eine Lösung für dieses Problem ist es, gewisse Fehlertoleranzen `tol` (z.B. 10^{-12}) zuzulassen.
  - ersetzen Sie im *Optimalitätstest* $z_N \geq 0$ durch $z_N \geq -tol$
  - ersetzen Sie im *Ratio-Test* alle $|w_i| \leq tol$ mit $w_i = 0$
  - setzen Sie im *Update* explizit $x_{B_i} = 0$ uns setzen Sie zusätzlich $x_{B_p} = 0$ falls $x_{B_p} \leq tol$

Zum Lösen des Problems aus der Mittelseminaraufgabe mittels Ihres Simplex-Algorithmus benötigen Sie nun nur noch die folgenden Dateien, die sich auf der Homepage im Verzeichnis *Übungen* befinden: `runtruss.m`, `opttruss.m`, `simplexLP.m` und `plottruss.m`. Die Hierarchie der Dateien ist dabei wie folgt:
`runtruss.m` → `opttruss.m` → `simplexLP.m` → `simplex.m` ,
d.h. `runtruss.m` ruft die Datei `opttruss.m` auf, welche wiederum `simplexLP.m` aufruft, die schließlich die von Ihnen implementierte Datei `simplex.m` startet. Abschließend ruft `runtruss.m` die Datei `plottruss.m` auf, welche das resultierende Stabwerk zeichnet.
Zum Lösen des Problems reicht es also aus, alle Dateien im gleichen Verzeichnis zu speichern und `runtruss.m` zu starten.

G 2 Dualer Simplex Algorithmus

a) Implementieren Sie den dualen Simplex Algorithmus in MATLAB. Verwenden Sie beim Pricing und im Ratio-Test die folgende Auswahlregel

   - Pricing: die Nichtbasisvariable x_{B_i} mit den kleinsten Wert tritt in die Basis ein,
   - Ratio-Test: stehen mehrere Nichtbasisvariablen $z_j, j \in N$ zur Auswahl, so tritt die mit dem kleinsten Index $\widetilde{j}$ die Basis ein.

   Verwenden Sie den Funktionskopf

   ```
   function [x_opt, y_opt, B_opt] = dualsimplex (A, b, c, B)
   ```

   mit folgenden Parametern:
   A, b, c: definiert gemäß dem LP in Standardform
   B: dual zulässige Basis
   x_{opt}: Optimallösung des primalen Problems
   y_{opt}: Optimallösung des dualen Problems
   B_{opt}: optimale Basis
   Hinweis: Zum Erzeugen von N ist der Befehl `setdiff` hilfreich. Zum Finden bestimmter Indizes eignet sich der Befehl `find`.

b) Testen Sie Ihren Code am folgenden Problem:

$$
\begin{array}{lrcrcrcrcr}
\min & 3x_1 & + & 3x_2 & + & 6x_3 & + & 5x_4 & & \\
\text{s.t.} & x_1 & + & 2x_2 & + & 4x_3 & - & 6x_4 & \leq & 14 \\
 & -2x_1 & - & x_2 & - & 5x_3 & + & 3x_4 & \leq & -25 \\
 & x_1 & & & + & 2x_3 & - & 2x_4 & \leq & 14 \\
 & & & & & & x_1, x_2, & x_3, x_4 & \geq & 0
\end{array}
$$

   (Benutzen Sie die Schlupfvariablen als Startbasis)

Hausübung

H 30 (5 Punkte)

Zeigen Sie: (a) Für jedes $r \in \mathbb{Q}$ gilt: $|r| \leq 2^{\langle r \rangle - 1} - 1$.

(b) Für je zwei rationale Zahlen $r, s \in \mathbb{Q}$ gilt: $\langle rs \rangle \leq \langle r \rangle + \langle s \rangle$.

H 31 (5 Punkte)

(A) Betrachten Sie das Polyeder $\mathcal{P} = \{x \in \mathbb{R}^n : Cx \leq d\}$ mit

$$C = \begin{pmatrix} 3 & -1 \\ -6 & 2 \end{pmatrix} \qquad \text{und} \qquad d = \begin{pmatrix} 4 \\ -9 \end{pmatrix}.$$

Wie viele Iterationen benötigt die Ellipsoidmethode höchstens um zu entscheiden, ob $\{x \in \mathbb{R}^n : Cx < d\}$ leer ist oder $\mathcal{P}$ einen Punkt enthält?

(B) In der j-ten Iteration der Ellipsoidmethode seien $a_j = (0,0)^T$ und $A_j = \begin{pmatrix} 9 & 0 \\ 0 & 4 \end{pmatrix}$ gegeben. Sei $x + y \leq -1$ eine der verletzten Ungleichungen. Stellen Sie diese Situation graphisch dar. Bestimmen Sie a_{j+1} und A_{j+1} und stellen Sie das zugehörige Ellipsoid graphisch dar.

H 32 (5 Punkte)

(A) Betrachten Sie folgendes Polytop in $\mathbb{R}^2$:

$$\begin{array}{rcrcr} -x_1 & - & x_2 & \leq & -2 \\ 3x_1 & & & \leq & 4 \\ -2x_1 & + & 2x_2 & \leq & 3 \end{array}$$

Stellen Sie mit Hilfe der Ellipsoidmethode fest, ob das Polytop leer ist oder nicht. Verwenden Sie als Anfangsellipsoid einen Kreis mit Mittelpunkt 0 und Radius 7. (Rechnen Sie bitte auf vier Nachkommastellen genau.)

(B) Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Zeigen Sie, dass A durch die Definition

$$\langle x, y \rangle_A := x^T A y$$

ein inneres Produkt auf $\mathbb{R}^n$ induziert. Mit $\|x\|_A := \sqrt{\langle x, x \rangle_A}$ ist dann eine Norm auf $\mathbb{R}^n$ definiert. Beweisen Sie für diese Norm die Ungleichung von Cauchy-Schwarz

$$|\langle x, y \rangle_A| \leq \|x\|_A \|y\|_A.$$

Welche Gestalt hat die Einheitskugel dieser Norm?