Prof. Dr. Helmut Abels
Andreas Schöttl
Josef Weber

Fakultät für Mathematik
Universität Regensburg
30. Oktober 2015

1. Programmierübungsblatt

zur Vorlesung: *Numerik I* im WiSe 15/16

Abgabe bis 16.11.2015 um 12:00 Uhr im Übungskasten 34 der Mathematik sowie in GRIPS.

Laden Sie den Quellcode für die von Ihnen bearbeitete Programmieraufgabe bis zum Abgabetermin in GRIPS hoch und geben Sie einen Ausdruck dieses Quellcodes bis 16.11.2015 um 12:00 Uhr im Übungskasten 34 der Mathematik ab. Die Datei soll dabei den Namen *Projekt*1*_Nachname_Vorname*(*_Nachname_Vorname*).*c* (oder auch Dateiendung .cpp) haben mit den Vor- und Nachnamen der Personen, die die Aufgabe bearbeitet haben. Laden Sie nicht das ausführbare Programm in GRIPS hoch. In den ersten beiden Zeilen Ihres Quellcodes müssen als Kommentar die Nummer der von Ihnen bearbeiteten Aufgabe, also Aufgabe 1 oder Aufgabe 2, sowie die Namen der Personen, die die Aufgabe bearbeitet haben, vermerkt sein.
Die Programmieraufgabe gilt als bestanden, wenn die Lösungen abgegeben wurden, einem Assistenten in den Programmierübungen vorgestellt wurden und Fragen zur Lösung hinreichend beantwortet werden konnten. Dabei können die Lösungen einzeln oder in Zweiergruppen abgegeben werden. Bei Abgabe in Zweiergruppen müssen beide Personen die Fragen zur Lösung beantworten können.
Die Terminvergabe für die Vorstellung Ihrer Lösung erfolgt über GRIPS. Bitte tragen Sie sich dazu in die dortige Terminliste ein.
Versehen Sie Ihren Quellcode mit hinreichend vielen Kommentarzeilen, so dass Sie ihn auch nach drei Wochen noch problemlos lesen und erklären können. Deklarieren Sie die zu programmierenden Funktionen so, wie es in der Aufgabenstellung vorgegeben ist. Weiterhin soll in beiden Aufgaben der Speicherplatz dynamisch erzeugt werden.

Lösen Sie eine der beiden Aufgaben.

Aufgabe 1: (LR-Zerlegung mit und ohne Pivotsuche)

i) Programmieren Sie eine Funktion mit der Form

int LR (int n, double** A, int* s, int flag).

Diese soll für $flag = 0$ die LR-Zerlegung ohne Pivotsuche von einer $n \times n$-Matrix A berechnen. Für $flag = 1$ soll die Funktion die LR-Zerlegung mit Zeilenpivotisierung durchführen, also eine Zerlegung $AP = LR$ mit P Permutationsmatrix liefern, siehe 2. Übungsblatt, Aufgabe 2. In beiden Fällen soll die übergebene Matrix mit den entsprechenden Einträgen der LR-Zerlegung überschrieben werden. Falls die Zerlegung nicht durchführbar ist, so soll die Funktion abbrechen und die Nummer des nicht mehr durchführbaren Schrittes zurückgeben. Weiterhin soll in s die Permutationsmatrix speicherschonend als Vektor gespeichert werden. In jedem Schritt der LR-Zerlegung soll auf dem Bildschirm die Nummer der Spalte ausgegeben werden, in welcher sich das Pivotelement befindet.

ii) Um das nun auftretende Gleichungssystem der Form $LRx = b$ mit L normierte untere Dreiecksmatrix und R obere Dreiecksmatrix zu lösen, benötigen wir Funktionen, die bei Übergabe der Dimension, einer Matrix und eines Vektors folgende Probleme lösen:
   (a) Die Vorwärtssubstitution, d.h. die Lösung von $Lx = b$ mit $b \in \mathbb{R}^n$.
   (b) Die Rückwärtssubstitution, d.h. die Lösung von $Rx = b$ mit $b \in \mathbb{R}^n$.

   Erweitern Sie Ihr Programm aus i) um Funktionen, welche die Form

   int VwSubs(int n, double** L, double* b) für die Vorwärtssubstitution,
   int RwSubs(int n, double** R, double* b) für die Rückwärtssubstitution

   besitzen und als Rückgabewert 1, falls ein Fehler auftritt, oder 0, falls die Vorwärts- bzw. Rückwärtssubstitution erfolgreich war, zurückgeben. Weiterhin sollen beide Funktionen den Vektor b mit der Lösung x überschreiben, um Speicherplatz zu sparen.

iii) Erweitern Sie Ihr Programm um eine Funktion

int Solve(int n, double** A, double* b, int flag).

Diese Funktion soll für eine $n \times n$-Matrix A und einen Vektor b das Gleichungssystem $Ax = b$ lösen. Hierbei soll für $flag = 0$ die LR-Zerlegung ohne Pivotsuche und für $flag = 1$ die LR-Zerlegung mit Zeilenpivotisierung ausgeführt werden. Falls die Zerlegung nicht durchführbar ist, so soll auf dem Bildschirm die Nummer des nicht mehr durchführbaren Schrittes ausgegeben werden.
Rufen Sie im Hauptprogramm die Funktion *Solve* auf und lösen Sie damit die Gleichung für

$$(A|b) = \begin{pmatrix} 2 & 1 & 1 & | & 1 \\ 4 & 2 & -1 & | & 2 \\ -1 & 0 & 7 & | & 3 \end{pmatrix}.$$

Geben Sie das Ergebnis x auf dem Bildschrim aus.

Aufgabe 2: (Cholesky-Zerlegung für Tridiagonalmatrizen)

i) Programmieren Sie eine Funktion von der Form

int Cholesky(int n, double** A),

welcher eine symmetrische, positiv definite $n \times n$-Matrix A übergeben wird. Die Funktion *Cholesky* soll für A die Cholesky-Zerlegung berechnen und diese in A speichern. Testen Sie die Funktion an der Matrix

$$A = \begin{pmatrix} 4 & 2 & 0 & 0 \\ 2 & 5 & 2 & 0 \\ 0 & 2 & 10 & 3 \\ 0 & 0 & 3 & 2 \end{pmatrix}$$

und geben Sie die Cholesky-Zerlegung von A am Bildschirm aus.

ii) Wir betrachten die inhomogene Helmholtz-Gleichung

$$-u''(x) + \kappa^2 u(x) = f(x) \qquad (*)$$

auf $\Omega = [0, 1]$ mit den Randwerten $u(0) = 1$ und $u(1) = 2$. Sei nun zu $N \in \mathbb{N}$ durch die Knoten $x_i = \frac{i}{N}$, $0 \leq i \leq N$, ein Gitter auf $[0, 1]$ festgelegt. Die Schrittweite ist $h = \frac{1}{N}$. Ziel ist es, die Lösung u von $(*)$ auf dem Gitter, also die Werte $u_i := u(x_i)$, zu approximieren. Dazu wird u'' mit dem folgenden zweiten zentralen Differenzenquotienten approximiert:

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}.$$

Stellen Sie als Diskretisierung von $(*)$ das folgende Gleichungssystem auf:

$$-\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \kappa^2 u(x_i) = f(x_i), \qquad 1 \leq i \leq N - 1.$$

Setzen Sie die bekannten Randwerte $u_0 = u(x_0)$ und $u_N = u(x_N)$ ein, so dass Sie ein Gleichungssystem in den Unbekannten u_i mit $1 \leq i \leq N - 1$ erhalten. Geben Sie das daraus resultierende Gleichungssystem zur Kontrolle mit Ihrem Quellcode im Übungskasten ab.

iii) Lösen Sie das Gleichungssystem für $f(x) = 9x$, $\kappa = 3$ und $N \in \{2, 8, 64\}$ numerisch. Programmieren Sie dazu eine Funktion

int Cholesky_Tridiagonal (int n, double** A),

wobei nun A eine $n \times 2$-Matrix sei, in der die Einträge einer Tridiagonal-Matrix B speicherschonend gespeichert seien. Diese Funktion soll nun die Cholesky-Zerlegung für eine Tridiagonal-Matrix B

berechnen, die als $n \times 2$-Matrix übergeben wird, und diese Zerlegung in A speichern. Überlegen Sie sich dazu eine passende Speicherstruktur für A und setzen Sie diese im Programm um.
Programmieren Sie weiterhin zwei Funktionen

int VwSubs_Tridiagonal(int n, double** A, double * b) für die Vorwärtssubstitution,

int RwSubs_Tridiagonal(int n, double** B, double * b) für die Rückwärtssubstitution,

denen die Diagonale und Nebendiagonale der Matrix $\overline{L}$ aus der Cholesky-Zerlegung speicherschonend als $n \times 2$-Matrix übergeben werden. Diese Funktionen sollen die Vorwärts- und Rückwärtssubstitution für Diagonalmatrizen mit einer Nebendiagonale unter Berücksichtigung der besonderen Speicherstruktur ausführen.

iv) Speichern Sie die Werte x_i und u_i aus Aufgabenteil *iii*) für die Fälle $N \in \{2, 8, 64\}$ jeweils in einer Datei 'data2.txt', 'data8.txt' und 'data64.txt' und rufen Sie *gnuplot* auf. Mit dem Befehl *plot* sowie dem zugehörigen Dateinamen erhalten Sie die gewünschten Graphiken.
Geben Sie diese Graphiken zusammen mit dem Quellcode und dem Gleichungssystem aus Aufgabenteil ii) im Übungskasten ab.