Prof. Dr. Karsten Urban

Dipl.-Math. Katharina Becker-Steinberger,
Dipl.-Math. oec. Sebastian Kestler

Institut für Numerische Mathematik
Wintersemester 2012/13

ulm university universität ulm

Numerik 1

Matlab-Blatt 4

(**Abgabe** wie auf dem ersten Matlab-Übungsblatt beschrieben
bis Dienstag 18.12.2012, 24:00 per Email)

Aufgabe 7 *(Sparse matrices in CCS format)* (5+2+3 Punkte)

Internally sparse matrices have to be encoded in special formats. In the Appendix D.2 of the lecture script the CRS format (Compressed Row Storage) is introduced, which uses three arrays to store a sparse matrix $A \in \mathbb{R}^{n\times n}$. These are

- `val`: value vector, which components are the non-zeros of A (left-to-right, then top-to-bottom)
- `col_ind`: index vector, column indices corresponding to the component of `val`
- `row_ptr`: index vector, pointer to rows, list of value indexes where each row starts

Matlab uses the CCS format (Compressed Column Storage), which is just CRS for the transposed matrix and relies on the arrays

- `val`: value vector, which components are the non-zeros of A (top-to-bottom, then left-to-right-bottom)
- `row_ind`: index vector, row indices corresponding to the component of `val`
- `col_ptr`: index vector, pointer to columns, list of value indexes where each column starts

a) Implement the Matlab function `[val, row_ind, col_ptr] = CCS(A)` returning the given square matrix $A \in \mathbb{R}^{n\times n}$ in the CCS format. Hint: Use the Matlab function `find`. For that reason, make yourself familiar with this function. You may assume that in the matrix A there are no columns with all elements equal to 0.

b) What is the computational complexity of your function `CCS(A)`:

(i) with respect to the matrix size n, where $A \in \mathbb{R}^{n\times n}$?

(ii) with respect to p, which denotes the number of non-zero elements of A?

Write your solution as a comment in your function `CCS(A)`. Hint: Analyse the following two cases: when A is in full and in sparse format. The number of non-zero elements of A can be obtained by `p = nnz(A)`

c) Compare the runtime of your implementation of `CCS(A)` for sparse matrices $A \in \mathbb{R}^{n\times n}$ in sparse and full format, for $n = 2^{5,6,\ldots,12}$. Use the sprase arrow matrix

```
A = [n+1, (n:-1:1); ones(n,1), eye(n, n)];
```

Further, use the routines `tic` and `toc` as on Praxis-Blatt 2.

Aufgabe 8 *(Jacobi- und Gauss-Seidel-Verfahren)* (15 Punkte)

In dieser Aufgabe sollen sie das Jacobi- und das Gauss-Seidel-Verfahren vergleichen. Schreiben Sie dazu ein Skript `main2.m`, in dem Sie für eine zufällige, diagonaldominante 1000×1000 - Matrix A und einen zufälligen 1000×1 - Vektor b das Gleichungssystem $Ax = b$ mit beiden Verfahren lösen. Gehen Sie dabei wie folgt vor.

a) Initialisieren Sie die Matrix A, den Vektor b und den Startvektor $x^{(0)}$, der nur Nullen enthält. Achtung: A soll zufällig und diagonaldominant sein.

b) Berechnen Sie die exakte Lösung x_{ex} mit dem Backslash-Operator.

c) Führen Sie jeweils 12 Schritte des Jacobi- und Gauss-Seidel-Verfahrens durch. Berechnen Sie für jeden Schritt den Fehler $||x^{(k)} - x_{ex}||_2$ (Sie dürfen für die Norm die Matlab-Funktion `norm` verwenden).

d) Plotten Sie beide Fehler über k in eine Grafik. Wählen Sie die Skalierung der Achsen so, dass näherungsweise Geraden entstehen.

Achten Sie darauf, dass ihre Iterationen effizient ausgeführt werden. Z.B. ist es effizienter die Iterationsmatrizen für die beiden Verfahren vor der Iteration einmal zu berechnen und nicht in jedem Iterationsschritt neu.

Was lässt sich über das Konvergenzverhalten der beiden Verfahren sagen?