Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
Dipl.-Tech. Math. P. Neumann
Dipl.-Inf. W. Eckhardt

SS 2013
Sheet 7
17.06.2013

Scientific Computing II

Incomplete Cholesky Decomposition

Programming Exercise 7: Incomplete Cholesky Decomposition

In the following, we review the incomplete Cholesky factorisation which computes an approximation $\tilde{A} \in \mathbb{R}^{N \times N}$ to a matrix $A \approx \tilde{A}$. We decompose $\tilde{A}$ into a lower left triangular matrix L and—analogous to the slides from the lecture—a diagonal matrix D such that $\tilde{A} = LD^{-1}L^{\top}$:

```
for i=1 to N do
  for j=1 to i-1 do
```
$$L_{ij} = A_{ij} - \sum_{k=1;(i,k),(j,k)\in S}^{j-1} L_{ik} D_{kk}^{-1} L_{jk} \quad \texttt{if} \quad (i,j) \in S$$
```
  end
```
$$L_{ii} = D_{ii} = A_{ii} - \sum_{k=1;(i,k)\in S}^{i-1} L_{ik}^2 D_{kk}^{-1}$$
```
end
```

where S denotes the set of all indices (i,j) with $A_{ij} \neq 0$.

(a) For which matrices can we apply the (incomplete) Cholesky decomposition?

(b) Implement the algorithm in a matlab function `incompleteLDL(A)` which returns the matrices L, D^{-1}.
Test your algorithm using a 2D Poisson problem with Dirichlet conditions as input. The respective stencil reads

$$S := \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

for all inner points. Write a function `generate2DPoisson(x,y)` which for a given number of grid points $x \times y$ generates the (full) matrix A. Include both inner and Dirichlet boundary points into the matrix description. You may use a lexicographic ordering of the grid points.
Validate your implementation by comparison with Matlab's internal routine `ichol(A)`.
Hint: `ichol(A)` works on matrices in sparse format only. You may use Matlab's routines `full(sparseMatrix)` to convert a sparse matrix into a full matrix and `sparse(matrix)` to convert a full matrix into a sparse matrix.

(c) Optimise your implementation `incompleteLDL(A)` such that you obtain a `matrix-free` (with respect to the input data) incomplete LDL-factorisation. The corresponding function `incompleteLDLPoisson2D(x,y)` only takes the number of grid points $x \times y$ as arguments.
Validate your implementations by comparing the resulting matrices L and D^{-1} with the results from `incompleteLDL(A)`.
What is the complexity of your optimised implementation?

Solution:

(a) The decomposition works for symmetric positive definite matrices.

(b) See `generate2DPoisson.m`, `incompleteLDL.m` and `exercise7.m`. We compare Matlab's matrix L_{matlab} and our matrices L, D^{-1} by computing $\tilde{L} := L \cdot D^{-1}_{\sqrt{}}$ where $D^{-1}_{\sqrt{}}$ is a diagonal matrix with entries $(D^{-1}_{\sqrt{}})_{ii} = \sqrt{D^{-1}_{ii}}$. Both matrices $\tilde{L}$ and L_{matlab} should be identical.

(c) See `incompleteLDLPoisson2D.m` and `exercise7.m`. Considering the nested for-loops (xx,yy) in the function, we observe that we have a $O(N)$-algorithm.