Homework I
Iterative methods for sparse matrices

Moritz Wolter

October 22, 2015

1 Theoretical background

1.1 GMRES

The GMRES abbreviates generalized minimal residuals. It's idea is to solve $Ax = b$ by using a vector $x_n \in \mathcal{K}_n$ that minimizes the residual $r_n = b - Ax_n$ [1]. In general the process is implemented using Arnoldi-iterations to compute the Krylov subspace and the Hessenberg representation of the original problem. The least square problem is then solved by coupling a QR decomposition into the Arnoldi iterations.

1.2 BICGSTAB

The Bi-CGSTAB, method is a modified version of the biconjugate gradient method. The biconjugate gradient method uses recurrences which only require data from the last iteration, which keeps memory requirements under control. The original BCG algorithm does not converge nicely. The main added benefit of the Bi-CGSTAB algorithm is better convergence behavior. A table containing the most important methods is given below[2]

1.3 Spectrum

1.4 Preconditioning

The convergence of iterative methods depends on the properties of the matrix a . . .

[1]Numerical Linear Algebra, Trefethen, Bau page 266
[2] as found in numerical linear algebra, Trefethen, Bau page 235.

	$Ax = b$	$Ax = \lambda x$
$A = A*$ (hermitian)	CG	Lanczos
$A \neq A*$ (non-hermitian)	GMRES, CGN, BCG et al.	Arnoldi

Table 1: Krylov subspace methods

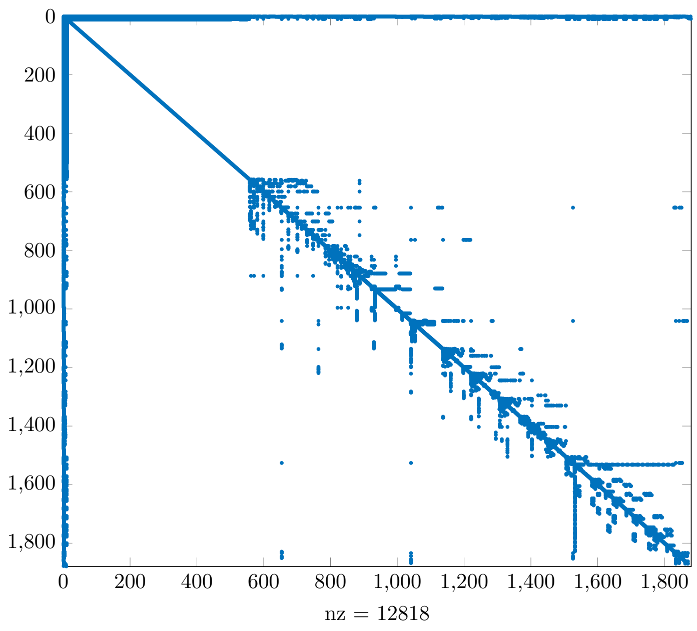

Figure 1: Sparsity representation of the `rajat12` matrix.

2 Results

2.1 rajat12

Figure 1 shows the `rajat12` circuit matrix. This matrix is real and the matlab routine `issymmetric` finds it to be not symmetric. From a complex point of view it can thus be considered non-hermitian. However a closer look at Figure 1 reveals that the matrix is very close to being symmetric. In a first series of experiments the iterative GMRES method is used. The level of the incomplete LU-factorization, which is used for preconditioning is varied from between zero and one. Furthermore the amount of available vectors is increased successively from zero to one hundred.
Results for the time measurements are shown in figure 2. The data shows a significant increase in computation time, for higher numbers of available vectors, with a zero level of fill. The same observation does not hold with a level of fill of one, here computations oscillate around an expected value of approximately two seconds.
Figure 3 shows convergence of the residual using 10 Vectors with zero level of fill in blue and level of fill 1 in red. Here it is important to note, that sufficient accuracy is reached using 10 Vectors within ≈ 230 iterations for ILU level zero. It takes ≈ 20 when the level of fill is set to one. Curiously the decrease in iterations not lead to on overall reduction of computation time. Probably in this case single iterations are much faster with zero level of fill, which would account for the timing difference. An overall increase of computations time can be observed for more vectors with in the zero level of fill setting. This is probably due to the fact that the overhead caused by the additional vectors becomes significant only when the amount of total iterations is large. Interestingly, when more vectors are used fewer iterations are required until convergence is reached. Large amounts of vectors are probably beneficial in settings, where larger matrices are used. Memory requirements are shown in table 2. From the date it can be concluded that the higher ilu level does is neither advantageous in terms of memory consumption or computing time when the

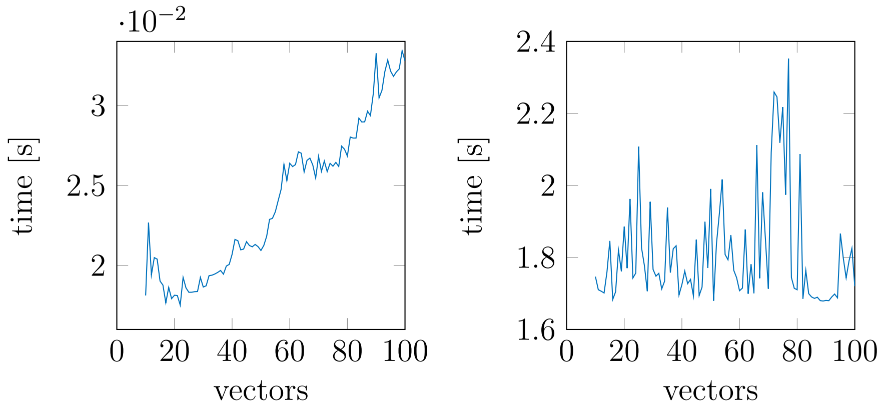

Figure 2: CPU-Time of running `./ilu --method gmres --file rajat12.mtx --nvectors $vectors --tolerance 1.e-10` with `--ilu-level 0`(left) and `--ilu-level 1`(right) the value of the `$vectors` variable is shown on the x axis.

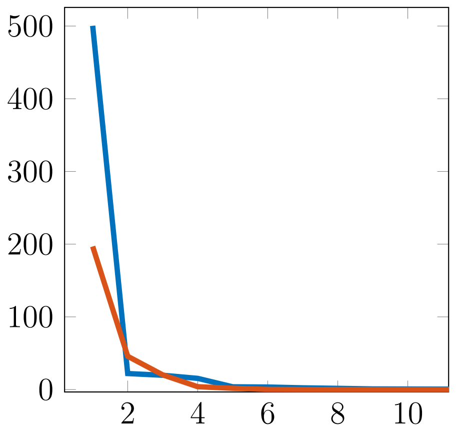

Figure 3: GMRES convergence with the rajat12 matrix with ILu level 0 and 1.

relatively small circuit matrix is considered.
Figure 4, shows the initial and preconditioned spectra. The normalizing effect of both preconditioners is clearly visible. Following the rule of thumb[3]:
"A preconditioner M is good if M^{-1}A is not too far from normal and its eigenvalues are clustered."
Both preconditioners work but the higher level of fill gives more clustering, therefore it is better in this case. The direct method implemented in the `mumps` requires `5.08 MB` of ram and finishes within $0.004035s = 4.035 * 10^{-3}s$. It is thus faster then any iterative scheme tried above, with more then acceptable memory consumption. Storing some data on the hard drive using the `--ooc` option increases the computation time to $0.007442s$. In this cases this is clearly not necessary, as the memory needed is available on almost any modern computer.

#vectors	ilu level of fill	memory
10	1	`155.9 MB`
100	1	`157.3 MB`
10	0	`3.1 MB`
100	0	`4.35 MB`

Table 2: Memory requirements of GMRES when run on the rajat12 matrix.

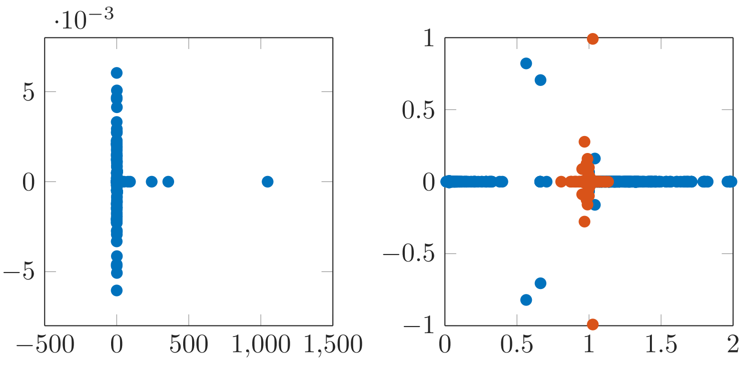

Figure 4: Original and preconditioned spectra of the rajat 12 matrix. Preconditioned matrices are shown for 0 level of fill in blue and level 1 in red.

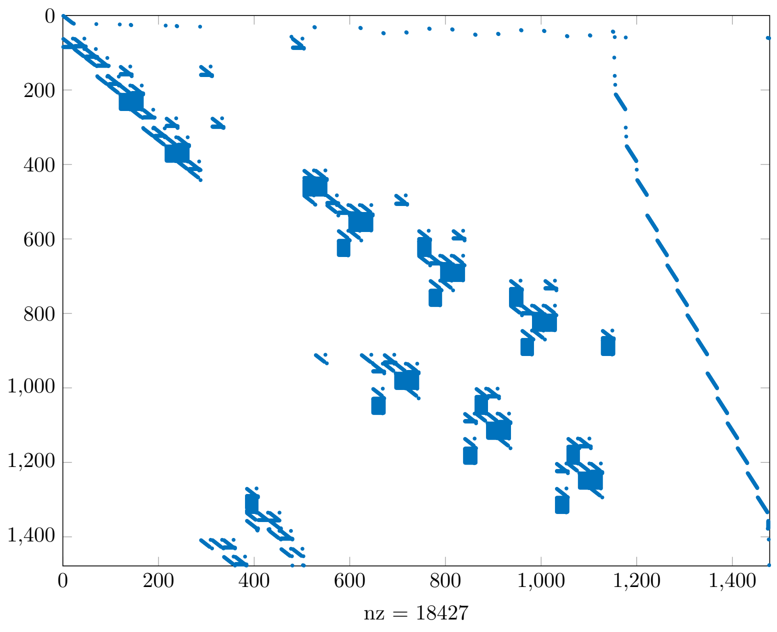

Figure 5: Sparsity representation of the `lhr01` matrix.

2.2 lhr01

Figure 5 shows the sparsity pattern of the `lhr01` light hydrocarbon recovery matrix. This matrix is real and not symmetric. From a complex point of view it again can be considered non-hermitian. However this matrix is a lot less symmetric then the circuit equations that have been explored earlier. The iterative methods implemented in the provided executable fail in this case givig the error message:

```
ILU factorization failed on equation 21.
```

In this case the Gaussian elimination process is unstable. The direct solver however is able to solve the problem in $0.009011s$ using `7.365 MB`.

2.3 ship003

2.4 Fault639

3 Conclusion

A lot of assumptions have been made throughout the creation of the model. Most of these are oversimplifications of the real world and we barley scratched the surface of the real optimization problem behind the airplane.

[3]Numerical linear algebra, Trefethen, Bau, page 314