Universität Stuttgart
Institut für parallele und verteilte Systeme
Prof. Dr. rer. nat. Miriam Mehl
Dipl.-Ing. Florian Lindner

SS 2016

Parallel Numerics

Exercise 5: Sparse Approximate Inverse

1 Sparse Approximate Inverses

i) What is preconditioning and why is it necessary/useful?
It is the transformation of an original linear system $Ax = b$ using a preconditioner matrix $M \approx A$:

$$M^{-1} Ax = M^{-1} b$$

Preconditioning changes the original system in a way that retains the solution while improving certain properties of the problem.

If the system is ill-conditioned, preconditioning will transform the spectrum such that it has better properties, e.g., clustering of eigenvalues, isolating extremal eigenvalues, or produce gaps in the spectrum. Thus, usually, the (spectral) condition number is reduced and the system becomes easier to solve. M should be cheap to apply (should be sparse) and to construct. Note that in many cases preconditioning is indispensable as the unpreconditioned system diverges due to the severe ill-conditioning. Here preconditioning causes convergence.

ii) Remember the Richardson iteration and apply preconditioning to it. What happens when you use $P = A$ or $P = I$ as preconditioning matrix?
Richardson splits by $A = A - I + I$

$$\begin{aligned} Ax = b \quad &\Rightarrow \quad M^{-1}Ax = M^{-1}b \\ (M^{-1}A - I + I)x &= M^{-1}b \\ x &= M^{-1}b - (M^{-1}A - I)x \\ x^{(n+1)} &= x^{(n)} - M^{-1}Ax^{(n)} + M^{-1}b \end{aligned}$$

iii) What are the main advantages of the SPAI (Sparse Approximate Inverse) algorithm?
Approximates the inverse of A. Only matrix-vector products become necessary for the preconditioning. The Frobenius norm ansatz allows for a decoupling of the least squares problems. Hence, the preconditioner can be computed completely in parallel.

iv) How is it possible to parallelize the computation of a static SPAI algorithm performing no pattern updates? What about the load balancing? (Give only thoughts, no pseudocode)
A possibility is to scatter A and the pattern of M, i.e., $\mathcal{J}(M)$ column-wise across the PEs. E.g., p_0 will be responsible for columns M_1, M_2, M_3, p_1 will be responsible for columns M_4, M_5, M_6, etc.. During the reduction phase (2) it is possible that communication becomes necessary as some required columns of A may lie on a remote PE. A possibility is to store a column-rank mapping on all PEs such that it is easy to get the rank of the PE which has the required column. Due to arbitrary sparsity structure (also due to fill-in during the pattern updates) it is mostly possible that some PEs will have to do more work. A proper load-balancing mechanism is useful, e.g., could be implemented in a work-stealing manner: if a PE has finished his local work chunk, it will request to compute columns of other PEs.

We consider the system

$$\min_{M^{-1}\in\mathcal{P}} \|AM - I\|_F^2 = \sum_{k=1}^{n} \min_{M_k^{-1}\in\mathcal{P}_k} \|AM_k - I_k\|_F^2, \tag{1}$$

Note that $\mathcal{P}$ is a predefined set of sparsity patterns and $\mathcal{P}_k$ of sparsity column pattern. We can solve (1) via QR decomposition. The sparsity pattern induces a reduction of the system. Hence, we deal with

$$\min_{\mathcal{J}(\hat{M}_k)=\mathcal{J}_k} \|\hat{A}_k\hat{M}_k - \hat{I}_k\|_F^2. \tag{2}$$

We obtain the solution via $\hat{A}_k = QR$ and solve for

$$\hat{M}_k = R^{-1}Q^T\hat{I}_k.$$

A backmapping has to be performed via $M_k(\mathcal{J}_k) = \hat{M}_k$.

v) Implement a SPAI for a tridiagonal matrix resulting from the heat equation, looking like:

$$A = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Use the same pattern for M^{-1}, i.e. $M^{-1} = \mathcal{J}(A)$.

vi) Implement a preconditioned CG with SPAI for the beforementioned matrix. Compare runtimes \ iterations with and without preconditioning.