Technische Universität München
Institut für Informatik
Dr. rer. nat. Tobias Neckel
Matous Sedlacek

WT 2012/2013

Parallel Numerics

Exercise 9: Gradient methods, Preconditioning, and Eigenvalues

1) Gradient Methods

Consider the linear system $Ax = b$ where

$$A = \begin{pmatrix} 11 & -9 \\ -9 & 11 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad x = \begin{pmatrix} \frac{11}{40} \\ \frac{9}{40} \end{pmatrix}. \tag{1}$$

i) Is the matrix A symmetric positive definite (SPD)?
We have to solve the characteristic polynomial $(A - \lambda I)x = 0$. *Therefore, we solve* $\det(A - \lambda I) = 0$. *We obtain the quadratic equation* $\lambda^2 - 22\lambda + 40 = 0$ *which has the solutions*

$$\lambda_1 = 20 \quad \textit{and} \quad \lambda_2 = 2.$$

All eigenvalues are positive. Hence, A is SPD.

ii) Apply the first two iterations of the gradient method (steepest descent). Use the initial vector $x^{(0)} = (0, 0)^T$.
It holds

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad r^{(0)} = b - Ax^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

After the first iteration $k = 0$ *we obtain*

$$\alpha^{(0)} = \frac{1}{11}, \quad x^{(1)} = \begin{pmatrix} \frac{1}{11} \\ 0 \end{pmatrix}, \quad r^{(1)} = \begin{pmatrix} 0 \\ \frac{9}{11} \end{pmatrix}.$$

Putting these results in for the second iteration $k = 1$ *we obtain*

$$\alpha^{(1)} = \frac{1}{11}, \quad x^{(2)} = \begin{pmatrix} \frac{1}{11} \\ \frac{9}{121} \end{pmatrix}, \quad r^{(2)} = \begin{pmatrix} \frac{81}{11} \\ 0 \end{pmatrix}.$$

iii) Show that the residuals $r^{(k)}$ and $r^{(k-2)}, k \geq 2$, are parallel in $\mathbb{R}^2$ for the gradient method.
For the residuals it holds: $r^{(k)} \perp r^{(k-1)}$ *and* $r^{(k-1)} \perp r^{(k-2)}$. *Hence, in the field* $\mathbb{R}^2$ *it must hold* $r^{(k)} \parallel r^{(k-2)}$ *for the gradient method.*

iv) Solve (1) with the CG method for the initial solution $x^{(0)} = (0,0)^T$. Compare your results to part ii). (see algorithm snippet below for the CG algorithm).
It holds

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad r^{(0)} = p^{(0)} = b - Ax^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The first iteration is similar to the gradient method. We obtain

$$\alpha^{(0)} = \frac{1}{11}, \quad x^{(1)} = \begin{pmatrix} \frac{1}{11} \\ 0 \end{pmatrix}, \quad r^{(1)} = \begin{pmatrix} 0 \\ \frac{9}{11} \end{pmatrix}, \quad p^{(1)} = \begin{pmatrix} \frac{81}{121} \\ \frac{9}{11} \end{pmatrix}.$$

Putting these results in for the second iteration $k = 1$ we obtain

$$\alpha^{(1)} = \frac{11}{40}, \quad x^{(2)} = \begin{pmatrix} \frac{11}{40} \\ \frac{9}{40} \end{pmatrix}$$

Hence, the CG method converges after $n = 2$ steps towards the solution in contrast to the gradient method.

v) Consider the Conjugate Gradient method that computes the solution x iteratively as a series $\{x^{(k)}\}$:

$$\begin{aligned} p^{(0)} &= r^{(0)} = b - Ax^{(0)} \\ \alpha^{(k)} &= -\frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle p^{(k)}, Ap^{(k)} \rangle} \\ x^{(k+1)} &= x^{(k)} - \alpha^{(k)} p^{(k)} \\ r^{(k+1)} &= r^{(k)} + \alpha^{(k)} Ap^{(k)} \\ \textbf{if } \|r^{(k+1)}\|_2^2 &\leq \epsilon \textbf{ then break} \\ \beta^{(k)} &= \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle} \\ p^{(k+1)} &= r^{(k+1)} + \beta^{(k)} p^{(k)} \end{aligned}$$

Implement this algorithm first for a sequential environment and afterwards for a parallel environment. To keep the parallelisation as simple as possible, use functions to compute the inner products, the matrix-vector multiplications and the addition of a vector, multiplied with a scalar, to another vector. Parallelise this algorithm by methods previously aquired in this course.
See sourcecode to corresponding tutorial on webpage and lecture slides.

2) (Block) Sparse Approximate Inverses

i) What is preconditioning and why is it necessary/useful?
Is the transformation of an original linear system $Ax = b$ using a preconditioner matrix $M \approx A^{-1}$:

$$\begin{aligned} MAx &= Mb \text{ (left preconditioning)} \\ AMy &= b,\ x = My \text{ (right preconditioning).} \end{aligned}$$

If 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 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) What are the main advantages of the SPAI (Sparse Approximate Inverse) algorithm?
Approximates the inverse of A – only matrix vector products become necessary in the solvers. The Frobenius norm ansatz allows for a decoupling of the Least Squares problems. Hence, the preconditioner can be computed completely in parallel. By univariate minimization a static approximation can be improved by using pattern updates. Therefore, no a priori choice of the sparsity pattern for M is crucial for the quality as profitable indices are found automatically by the algorithm.

iii) 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 (3) 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.

iv) Derive a block version of the static SPAI algorithm without pattern updates.
The block version of SPAI (BSPAI) is similar to the point-wise scalar SPAI. We consider the system

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

where $n_b = \frac{n}{b}$ is the block size of the block columns M_k. Note that $\mathcal{J}$ is a block sparsity pattern and $\mathcal{J}_k$ a block sparsity column pattern. We can solve (2) via QR decomposition. The sparsity pattern induces a reduction of the system. Hence, we deal with

$$\min_{\mathcal{J}(\hat{M}_k)=\mathcal{J}_k} \|\hat{A}_k\hat{M}_k - \hat{I}_k\|_F^2. \tag{3}$$

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) Derive a block version of SPAI using pattern updates.

We consider the univariate minimization

$$\min_{M_{jk}} \|A(M_k + I_j M_{jk}) - I_k\|_F = \min_{M_{jk}} \|R + A_j M_{jk}\|_F$$

with the residual $R = AM_k - I_k$. *We expand the residual to*

$$\begin{aligned} \|R + A_j M_{jk}\|_F^2 &= \operatorname{trace}((R + A_j M_{jk})^T (R + A_j M_{jk})) \\ &= \operatorname{trace}(R^T R) + 2\operatorname{trace}(M_{jk}^T A_j^T R) + \operatorname{trace}(M_{jk}^T A_j^T A_j M_{jk}). \end{aligned}$$

Derivating with respect to M_{jk} *and equating with zero we obtain the minimum*

$$\begin{aligned} &\frac{\partial}{\partial M_{jk}} \left(\operatorname{trace}(R^T R) + 2\operatorname{trace}(M_{jk}^T A_j^T R) + \operatorname{trace}(M_{jk}^T A_j^T A_j M_{jk})\right) \\ = \;& 2A_j^T R + A_j^T A_j M_{jk} + (A_j^T A_j)^T M_{jk} = 2A_j^T R + 2A_j^T A_j M_{jk} \stackrel{!}{=} 0 \\ \Leftrightarrow \;& M_{jk} = -(A_j^T A_j)^{-1} A_j^T R. \end{aligned}$$

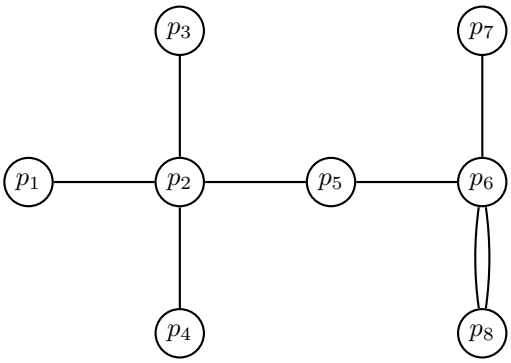
By adding this block to the pattern we require a quality metric to see how good the improvement will be. The new residual norm (and the reduction factor τ_{jk}*) can be computed via*

$$\begin{aligned} &\operatorname{trace}(R^T R) + 2\operatorname{trace}(M_{jk}^T A_j^T R) + \operatorname{trace}(M_{jk}^T A_j^T A_j M_{jk}) \\ = \;& \|R\|_F^2 + 2\operatorname{trace}\left[(-(A_j^T A_j)^{-1} A_j^T R)^T A_j^T R\right] + \\ & \quad \operatorname{trace}\left[(-(A_j^T A_j)^{-1} A_j^T R)^T A_j^T A_j (-(A_j^T A_j)^{-1} A_j^T R)\right] \\ = \;& \|R\|_F^2 - 2\operatorname{trace}\left(R^T A_j (A_j^T A_j)^{-1} A_j^T R\right) + \operatorname{trace}\left(R^T A_j (A_j^T A_j)^{-1} A_j^T R\right) \\ = \;& \|R\|_F^2 - \underbrace{\operatorname{trace}\left(R^T A_j (A_j^T A_j)^{-1} A_j^T R\right)}_{=:\,\tau_{jk}}. \end{aligned}$$

3) Sparse matrices and graphs

Let us consider the following cut-out of the munich subway network:

i) Interprete the network as a graph G and give the corresponding adjacency matrix A_G as well as the incidency matrix I_G.
We obtain a graph G *with* $n = 8$ *nodes and* $m = 8$ *edges:*



The adjacency matrix A_G is a $n \times n$-matrix, while the entry (i, j) exists if there is an edge between i and j. As G is undirected, A_G is symmetric. The incidency matrix I_G is a $n \times n$-matrix where each column corresponds to one edge in G.

$$A_G = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \quad I_G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

ii) Give the transitive closure of G.
As G is connected and undirected, the transitive closure of G is a clique of 8 nodes $p_1, ..., p_8$.

iii) We want to quantify the importance of the different subway stations. Therefore, we proceed similarly to google when computing "pageranks": A station is considered "important" if other "important" stations are connected to it. The importance r_i of a station p_i distributes simultaneously to all outgoing connections and can be computed via

$$r_i = \sum_{\substack{j \\ p_j \to p_i}} \frac{r_j}{d_j}, \tag{4}$$

where d_j is the outgoing degree of a node p_j.

a) Formulate (4) in matrix vector notation.
Denote the outgoing degree of a node p_j with d_j. Hence, each node p_i which is connected with p_j gets the weight $\frac{r_j}{d_j}$; the weight r_i of the node p_i is the sum of the weights of the neighbouring nodes:

$$r_i = \sum_{\substack{j \\ p_j \to p_i}} \frac{r_j}{d_j} \ . \tag{5}$$

Let us merge the weights r_j into a vector $r = (r_i)_{i=1,...,n}$. Hence, (5) can be written in matrix vector notation as $r = Hr$. The matrix $H = h_{i,j}$ has an entry $1/d_j$ if an edge between p_j and p_i exists, otherwise the entry is 0.

b) Which property has the occurring matrix?
The matrix H is a markov matrix: All entries are ≥ 0 and the column sum is always 1:

$$\sum_{i=1}^{n} h_{i,j} = \sum_{\substack{i \\ p_j \to p_i}} \frac{1}{d_j} = d_j \cdot \frac{1}{d_j} = 1 \ .$$

In particular, H has the eigenvalue 1.

c) Compute the "importance" of the various subway stations. Which problem occurs during the computation and how can you solve it?
H has the form:

$$H = \begin{pmatrix} 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix} .$$

The equation $r = Hr$ shows that we have to deal with an eigenvalue problem: the unknown vector r is eigenvector of H corresponding to the eigenvalue 1. The computation of the eigenvalues with numerical software (e.g. MATLAB*) gives all eigenvalues:*

$$-1.0000, -0.8660, 0.0000, 1.0000, 0.8660, 0, 0, 0.$$

The (normalized) eigenvector which corresponds to eigenvalue 1 is

$$v = (0.0625, 0.2500, 0.0625, 0.0625, 0.1250, 0.2500, 0.0625, 0.1250)^T.$$

Hence, finally, in our model the most important stations are Scheidplatz and Münchner Freiheit. Note that usually the matrix H is very large. To compute the largest eigenvalue iterative eigenvalue algorithms become indispensable.