

GMRES Algorithm: First Step

Start: $b, u_1 := \frac{b}{\|b\|}$

$$h_{2,1}u_2 = Au_1 - h_{1,1}u_1, \quad H_{2,1} = \begin{pmatrix} h_{1,1} \\ h_{2,1} \end{pmatrix}, \quad G_1 H_{2,1} = \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix},$$



GMRES Algorithm: First Step

$$\text{Start: } b, u_1 := \frac{b}{\|b\|}$$

$$h_{2,1}u_2 = Au_1 - h_{1,1}u_1, \quad H_{2,1} = \begin{pmatrix} h_{1,1} \\ h_{2,1} \end{pmatrix}, \quad G_1 H_{2,1} = \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix},$$

$$\begin{aligned} \|Ax - b\| &= \|AU_1 y_1 - b\| = \|H_{2,1} y_1 - \|b\| e_1\| \\ &= \left\| G_1^T \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| e_1 \right\| = \left\| \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| G_1 e_1 \right\| \end{aligned}$$



GMRES Algorithm: First Step

$$\text{Start: } b, u_1 := \frac{b}{\|b\|}$$

$$h_{2,1}u_2 = Au_1 - h_{1,1}u_1, \quad H_{2,1} = \begin{pmatrix} h_{1,1} \\ h_{2,1} \end{pmatrix}, \quad G_1 H_{2,1} = \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix},$$

$$\begin{aligned} \|Ax - b\| &= \|AU_1 y_1 - b\| = \|H_{2,1} y_1 - \|b\| e_1\| \\ &= \left\| G_1^T \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| e_1 \right\| = \left\| \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| G_1 e_1 \right\| \end{aligned}$$

$$y_1 = \frac{(G_1 e_1 \|b\|)_1}{\nu_{1,1}}, \quad x_1 = u_1 y_1 = U_1 y_1$$



GMRES Algorithm: Second Step

$$h_{3,2}u_3 = Au_2 - h_{2,2}u_2 - h_{1,2}u_1, \quad H_{3,2} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ 0 & h_{32} \end{pmatrix},$$

$$G_1 H_{3,2} = \begin{pmatrix} \nu_1 & * \\ 0 & * \\ 0 & * \end{pmatrix}, \quad G_2 G_1 H_{3,2} = \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix}$$



GMRES Algorithm: Second Step

$$h_{3,2}u_3 = Au_2 - h_{2,2}u_2 - h_{1,2}u_1, \quad H_{3,2} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ 0 & h_{32} \end{pmatrix},$$

$$G_1 H_{3,2} = \begin{pmatrix} \nu_1 & * \\ 0 & * \\ 0 & * \end{pmatrix}, \quad G_2 G_1 H_{3,2} = \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \|Ax - b\| &= \|AU_2 y_2 - b\| = \|H_{3,2} y_2 - \|b\| e_1\| \\ &= \left\| G_1^T G_2^T \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix} y_2 - \|b\| e_1 \right\| = \left\| \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_{2,1} \\ y_{2,2} \end{pmatrix} - \|b\| G_2 G_1 e_1 \right\| \end{aligned}$$

$$y_{2,2} = \frac{(G_2 G_1 e_1 \|b\|)_2}{\nu_{2,2}}, \quad y_{2,1} = \frac{(G_2 G_1 e_1 \|b\|)_1 - \nu_{1,2} y_{2,2}}{\nu_{1,1}}$$

$$x_2 = U_2 y_2 = u_1 y_{2,1} + u_2 y_{2,2} \quad \text{and so forth...}$$



GMRES Error (A diagonalisable)

$$\begin{aligned}
\|r_m\| &= \|Ax_m - b\| = \min_{x \in U_m} \|Ax - b\| \\
&= \min_{\alpha_1, \dots, \alpha_m} \left\| A \left(\sum_{j=0}^{m-1} \alpha_j A^j b \right) - b \right\| = \min_{\mathcal{P}^{(m-1)}} \|A \mathcal{P}^{(m-1)}(A)b - b\| \\
&= \min_{Q^{(m)}(0)=1} \|Q^{(m)}(A)b\| = \min_{Q^{(m)}(0)=1} \|V Q^{(m)}(\Lambda) V^{-1} b\| \\
&= \min_{Q^{(m)}(0)=1} \|V \operatorname{diag}(Q^{(m)}(\lambda_j)) V^{-1} b\| \leq \\
&\leq \kappa_2(V) \cdot \min_{Q^{(m)}(0)=1} \left[\max_{j=1, \dots, n} |Q^{(m)}(\lambda_j)| \right] \|r_0\|
\end{aligned}$$



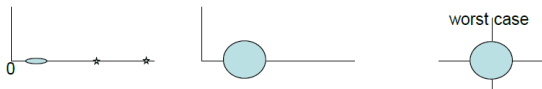
Iterative Methods: Survey

- Basic iterative methods (not considering multigrid!)
- SPD A : PCG (preconditioned conjugate gradient method)
- symmetric indefinite A : MINRES, SYMMLQ, (PCG on normal equations)
- general A :
 - GMRES (optimal like CG, but expensive)
 - BiCG, BiCGSTAB, CGS (not optimal, but cheap, instable?)
 - QMR (quasi optimal, cheap)



Iterative Methods: Survey

- Basic iterative methods (not considering multigrid!)
- SPD A : PCG (preconditioned conjugate gradient method)
- symmetric indefinite A : MINRES, SYMMLQ, (PCG on normal equations)
- general A :
 - GMRES (optimal like CG, but expensive)
 - BiCG, BiCGSTAB, CGS (not optimal, but cheap, instable?)
 - QMR (quasi optimal, cheap)
- Essential for convergence is position of eigenvalues (singular values)
- Fast convergence for well conditioned problems or matrices with clustered spectrum



5.5. Preconditioning

- Direct solvers: sequential, losing sparsity
- Iterative solvers: easy parallel and sparse, but possibly slowly convergent



5.6. Preconditioning

- Direct solvers: sequential, losing sparsity
- Iterative solvers: easy parallel and sparse, but possibly slowly convergent
- Combination of both methods (2 variants):
 1. Include preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$, such that
 - M is easy to deal with in parallel (reduced approximate direct solver)
 - spectrum of $M^{-1}A$ is much better clustered
 2. Or include preconditioner $M \approx A^{-1}$ in the form $MAx = Mb$, such that
 - M is easy to deal with in parallel (reduced approximate inverse)
 - Spectrum of MA is much better clustered



5.7. Preconditioning

- Direct solvers: sequential, losing sparsity
- Iterative solvers: easy parallel and sparse, but possibly slowly convergent
- Combination of both methods (2 variants):
 1. Include preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$, such that
 - M is easy to deal with in parallel (reduced approximate direct solver)
 - spectrum of $M^{-1}A$ is much better clustered
 2. Or include preconditioner $M \approx A^{-1}$ in the form $MAx = Mb$, such that
 - M is easy to deal with in parallel (reduced approximate inverse)
 - Spectrum of MA is much better clustered
- General both-sided preconditioning:

$$Ax = b \Leftrightarrow MA(Ky) = Mb \text{ and } Ky = x$$

Overview of Following Preconditioners

- Stationary preconditioners
- ILU preconditioner



Overview of Following Preconditioners

- Stationary preconditioners
- ILU preconditioner
- Polynomial preconditioners
- Sparse approximate inverse preconditioners:
 - SPAI for the general nonsymmetric case
 - FSPAI for the SPD case
 - MSPAI using SPAI with probing



5.7.1. Stationary Preconditioners

Consider preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$.

- Stationary iteration to splitting $A = M - N$. Convergence depends on

$$\|I - M^{-1}A\| < 1$$

- That is exactly a condition for a good preconditioner: spectrum clustered around 1, $M^{-1}A \approx I$

5.7.2. Stationary Preconditioners

Consider preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$.

- Stationary iteration to splitting $A = M - N$. Convergence depends on

$$\|I - M^{-1}A\| < 1$$

- That is exactly a condition for a good preconditioner: spectrum clustered around 1, $M^{-1}A \approx I$
- If splitting leads to fast convergence, than M is also a good preconditioner.
- In this sense, PCG with stationary preconditioner M can be seen as an acceleration of the stationary method with splitting $M - N$.



Stationary Preconditioners (cont.)

- Jacobi splitting with $D = \text{diag}(A)$ gives Jacobi preconditioner $M := D$.
- Gauss-Seidel splitting $M = D - L$ leads to GS preconditioner.



Stationary Preconditioners (cont.)

- Jacobi splitting with $D = \text{diag}(A)$ gives Jacobi preconditioner $M := D$.
- Gauss-Seidel splitting $M = D - L$ leads to GS preconditioner.
- Relaxation:

$$x^{(k,\text{new})} := (1 - \omega)x^{(k-1)} + \omega x^{(k)}$$

As convex combination of old and new iterate (Jacobi or GS)



Stationary Preconditioners (cont.)

- Jacobi splitting with $D = \text{diag}(A)$ gives Jacobi preconditioner $M := D$.
- Gauss-Seidel splitting $M = D - L$ leads to GS preconditioner.
- Relaxation:

$$x^{(k,\text{new})} := (1 - \omega)x^{(k-1)} + \omega x^{(k)}$$

As convex combination of old and new iterate (Jacobi or GS)

- Symmetrization: first iteration with preconditioner M , second iteration with M^T .

$$M_{\text{new}} := M + M^T - M^T A M$$



Stationary Preconditioners (cont.)

- Jacobi splitting with $D = \text{diag}(A)$ gives Jacobi preconditioner $M := D$.
- Gauss-Seidel splitting $M = D - L$ leads to GS preconditioner.
- Relaxation:

$$x^{(k,\text{new})} := (1 - \omega)x^{(k-1)} + \omega x^{(k)}$$

As convex combination of old and new iterate (Jacobi or GS)

- Symmetrization: first iteration with preconditioner M , second iteration with M^T .

$$M_{\text{new}} := M + M^T - M^T A M$$

- Special case: damped GS \rightarrow SSOR:

$$M_{\text{new}} = \frac{1}{2 - \omega} \left(\frac{1}{\omega} D - L \right) \left(\frac{1}{\omega} D \right)^{-1} \left(\frac{1}{\omega} D - L \right)^T$$



5.7.3. ILU Preconditioner, Sequentially

- Apply Gaussian elimination algorithm, but only on allowed pattern \Rightarrow incomplete LU factorization called ILU.
- Reduce in the `for`-loops the indices to the indices with
 - allowed pattern, e.g. ILU(0) for pattern of A
 - values that are not too small, ILUT for ILU with threshold



5.7.4. ILU Preconditioner, Sequentially

- Apply Gaussian elimination algorithm, but only on allowed pattern \Rightarrow incomplete LU factorization called ILU.
- Reduce in the `for`-loops the indices to the indices with
 - allowed pattern, e.g. ILU(0) for pattern of A
 - values that are not too small, ILUT for ILU with threshold
- Leads to approximate LU factorization

$$A = LU + R, \quad \text{preconditioner} \quad M = LU$$

with all ignored fill-in entries collected in R .

5.7.5. ILU Preconditioner, Sequentially

- Apply Gaussian elimination algorithm, but only on allowed pattern \Rightarrow incomplete LU factorization called ILU.
- Reduce in the `for`-loops the indices to the indices with
 - allowed pattern, e.g. ILU(0) for pattern of A
 - values that are not too small, ILUT for ILU with threshold
- Leads to approximate LU factorization

$$A = LU + R, \quad \text{preconditioner} \quad M = LU$$

with all ignored fill-in entries collected in R .

- MILU (modified ILU): collect all ignored fill-in entries on the related main diagonal elements \rightarrow maintains the row sum or the action on $(1, 1, \dots, 1)^T$.

5.7.6. Parallel ILU

Consider iterative solution of the system $A = LU$ on given pattern \mathcal{S} . So L and U are sparse with pattern \mathcal{S} , and the equations have to be satisfied only on the pattern \mathcal{S} :

$$A_{i,j} = \sum_{k, (i,k) \in \mathcal{S}, (k,j) \in \mathcal{S}} L_{i,k} U_{k,j} \text{ for } (i,j) \in \mathcal{S}$$



5.7.7. Parallel ILU

Consider iterative solution of the system $A = LU$ on given pattern \mathcal{S} . So L and U are sparse with pattern \mathcal{S} , and the equations have to be satisfied only on the pattern \mathcal{S} :

$$A_{i,j} = \sum_{k, (i,k) \in \mathcal{S}, (k,j) \in \mathcal{S}} L_{i,k} U_{k,j} \text{ for } (i,j) \in \mathcal{S}$$

Use $A = LU$ as fixed point iteration in the form:

- freeze L , compute new U
- freeze U , compute new L
- repeat until convergence



5.7.8. Parallel ILU

Consider iterative solution of the system $A = LU$ on given pattern \mathcal{S} . So L and U are sparse with pattern \mathcal{S} , and the equations have to be satisfied only on the pattern \mathcal{S} :

$$A_{i,j} = \sum_{k, (i,k) \in \mathcal{S}, (k,j) \in \mathcal{S}} L_{i,k} U_{k,j} \text{ for } (i,j) \in \mathcal{S}$$

Use $A = LU$ as fixed point iteration in the form:

- freeze L , compute new U
- freeze U , compute new L
- repeat until convergence

$$l_{i,j} = \frac{1}{u_{j,j}} \left(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} u_{k,j} \right) \quad (1)$$

$$u_{i,j} = \frac{1}{l_{i,i}} \left(a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} u_{k,j} \right) . \quad (2)$$



5.7.9. Iterative ILU Algorithm

Set unknowns $l_{i,j}$ and $u_{i,j}$ to initial values

FOR sweep = 1,2,... until convergence

 WHILE $(i,j) \in S$

 IF $i > j$

$$l_{i,j} = \frac{1}{u_{j,j}} (a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} u_{k,j})$$

 ELSE

$$u_{i,j} = a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} u_{k,j}$$

 END

 END

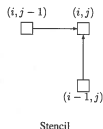
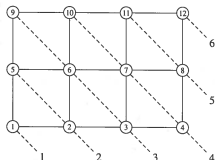
END



5.7.10. Direct Triangular Solves

Remaining problem: Solve $Lx = b$ in parallel!

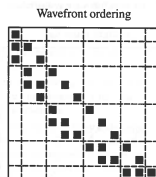
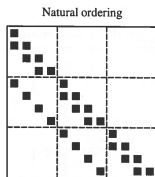
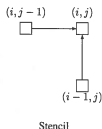
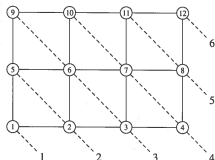
- There exist parallel direct solvers based on graph coloring, but with restricted parallelism.
- Using data dependency graph:



5.7.11. Direct Triangular Solves

Remaining problem: Solve $Lx = b$ in parallel!

- There exist parallel direct solvers based on graph coloring, but with restricted parallelism.
- Using data dependency graph:



- Collect indices that can be directly derived: LEVEL 1
- From data dependency graph collect indices depending only on the first level: LEVEL 2
- and so on.

Unknowns in the same level can be updated in parallel. Compare colored GS and multifrontal solvers!

5.7.12. Iterative Triangular Solves

Resort: Jacobi iteration for $b = Lx = (D + L_0)x$ resulting in the iteration

$$Dx^{k+1} = b - L_0x^k, k = 0, 1, 2, \dots$$



5.7.13. Iterative Triangular Solves

Resort: Jacobi iteration for $b = Lx = (D + L_0)x$ resulting in the iteration

$$Dx^{k+1} = b - L_0x^k, k = 0, 1, 2, \dots$$

Nice feature: Iteration matrix L_0 has zero eigenvalues. Therefore convergence guaranteed via special norm:

$$\|e_k\| \leq \|L_0\|^k \|e_0\|$$

and there exists a norm with $\|L_0\| < 1$. But often slow convergence!

Improvement: Block-Jacobi.

Search for iterative fast converging and fully parallel methods! Later on more!



Overview Explicit Preconditioners

- ILU and stationary methods use approximations on A itself.
- The resulting preconditioners are given by triangular matrices L , that have to be solved in each iteration step: $L^{-1}x^{(k)}$!
- Jacobi easy to parallelize, but slow convergence.



Overview Explicit Preconditioners

- ILU and stationary methods use approximations on A itself.
- The resulting preconditioners are given by triangular matrices L , that have to be solved in each iteration step: $L^{-1}x^{(k)}$!
- Jacobi easy to parallelize, but slow convergence.
- Question: How to derive preconditioners that lead to fast convergence and are easy to parallelize?
- Until now considered variant 1.; In the following variant 2.:
Find approximations on $M \approx A^{-1}$. Then the solution of the linear system given by the preconditioner, is only $Mx^{(k)}$, a matrix vector product!



Parallel Preconditioning

- Find preconditioner M , that satisfies three conditions:
 - (i) The computation of M is fast in parallel
 - (ii) The application Mx in each iteration step is easy in parallel
 - (iii) The spectrum AM or MA is clustered \rightarrow fast convergence



Parallel Preconditioning

- Find preconditioner M , that satisfies three conditions:
 - (i) The computation of M is fast in parallel
 - (ii) The application Mx in each iteration step is easy in parallel
 - (iii) The spectrum AM or MA is clustered \rightarrow fast convergence
- Examples:
 - For GS is (i) ok, but not (ii)
 - For Jacobi (i) and (ii) ok, but not (iii)
 - For ILU (iii) ok, (i) also iteratively, but not (ii)
- Note that preconditioners also have to be well defined! Zero on diagonals?



5.7.14. Polynomial Preconditioners

- Characteristic polynomial for A :

$$0 = q(A) = \gamma_n A^n + \gamma_{n-1} A^{n-1} + \dots + \gamma_1 A + \gamma_0 I$$

Gives polynomial representation for A^{-1} ($\gamma_0 \neq 0$):

$$A^{-1} = \frac{1}{\gamma_0} (-\gamma_n A^{n-1} - \gamma_{n-1} A^{n-2} - \dots - \gamma_1 I) = p(A)$$

- Therefore, it makes sense to approximate A^{-1} by a polynomial.



5.7.15. Polynomial Preconditioners

- Characteristic polynomial for A :

$$0 = q(A) = \gamma_n A^n + \gamma_{n-1} A^{n-1} + \dots + \gamma_1 A + \gamma_0 I$$

Gives polynomial representation for A^{-1} ($\gamma_0 \neq 0$):

$$A^{-1} = \frac{1}{\gamma_0} (-\gamma_n A^{n-1} - \gamma_{n-1} A^{n-2} - \dots - \gamma_1 I) = p(A)$$

- Therefore, it makes sense to approximate A^{-1} by a polynomial.
- Better approximation by finding region $S \in \mathbb{R}$ or \mathbb{C} that contains nearly all eigenvalues, and then find polynomial p that is near the inverse in S :

$$\min_{p_n} \|P(A)A - I\|, \quad \min_{p_n(x)} \left(\max_{\lambda \text{ eigenvalue}} |p(\lambda)\lambda - 1| \right)$$

- Solution: Normalized Chebyshev polynomials.



5.7.16. Polynomial Preconditioners

- Characteristic polynomial for A :

$$0 = q(A) = \gamma_n A^n + \gamma_{n-1} A^{n-1} + \dots + \gamma_1 A + \gamma_0 I$$

Gives polynomial representation for A^{-1} ($\gamma_0 \neq 0$):

$$A^{-1} = \frac{1}{\gamma_0} (-\gamma_n A^{n-1} - \gamma_{n-1} A^{n-2} - \dots - \gamma_1 I) = p(A)$$

- Therefore, it makes sense to approximate A^{-1} by a polynomial.
- Better approximation by finding region $S \in \mathbb{R}$ or \mathbb{C} that contains nearly all eigenvalues, and then find polynomial p that is near the inverse in S :

$$\min_{p_n} \|P(A)A - I\|, \quad \min_{p_n(x)} \left(\max_{\lambda \text{ eigenvalue}} |p(\lambda)\lambda - 1| \right)$$

- Solution: Normalized Chebyshev polynomials.
- + Advant. of pol. prec.: better in parallel (mtx-vec product, BLAS 2)
- Disadvantage: non-optimal approximation in Krylov subspace



5.7.17. Sparse Approximate Inverses

- Other approach for approximating A^{-1} by norm minimization

$$\min_{M \in \mathcal{P}} \|AM - I\| \quad \text{over some sparsity pattern } \mathcal{P}.$$



5.7.18. Sparse Approximate Inverses

- Other approach for approximating A^{-1} by norm minimization

$$\min_{M \in \mathcal{P}} \|AM - I\| \quad \text{over some sparsity pattern } \mathcal{P}.$$

- Choice of the norm?
 - analytic (to allow the explicit solution of this problem)
 - easy to compute (in parallel)



5.7.19. Sparse Approximate Inverses

- Other approach for approximating A^{-1} by norm minimization

$$\min_{M \in \mathcal{P}} \|AM - I\| \quad \text{over some sparsity pattern } \mathcal{P}.$$

- Choice of the norm?
 - analytic (to allow the explicit solution of this problem)
 - easy to compute (in parallel)
- Optimal norm is Frobenius norm:

$$\|A\|_F^2 := \sum_{i,j=1}^n a_{i,j}^2 = \sum_{j=1}^n \|A_{\bullet,j}\|^2 = \text{trace}(A^T A)$$



SPAI in Parallel

- First, we choose pattern \mathcal{P} in a static way a priori, e.g. as the pattern of A

$$\min_{M \in \mathcal{P}} \|AM - I\|_F^2 = \min_{M \in \mathcal{P}} \sum_{k=1}^n \|(AM - I)e_k\|_2^2 = \sum_{k=1}^n \min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2$$



SPAI in Parallel

- First, we choose pattern \mathcal{P} in a static way a priori, e.g. as the pattern of A

$$\min_{M \in \mathcal{P}} \|AM - I\|_F^2 = \min_{M \in \mathcal{P}} \sum_{k=1}^n \|(AM - I)e_k\|_2^2 = \sum_{k=1}^n \min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2$$

- Hence, to minimize the Frobenius norm, we have to solve n least squares problems in the sparse columns of M !
- This can be done fully in parallel!
- But costs for least squares problems?



SPAI and Least Squares

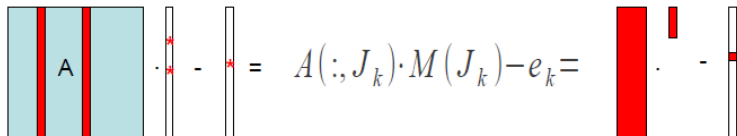
$$\min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2 = \min_{M_k \in \mathcal{P}_k} \left\| A \begin{pmatrix} 0 \\ * \\ 0 \\ * \\ 0 \\ 0 \end{pmatrix} - e_k \right\|_2^2 = \min_{M_k \in \mathcal{P}_k} \|A(:, \mathcal{J}_k)M_k(\mathcal{J}_k) - e_k\|_2^2$$



SPAI and Least Squares

$$\min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2 = \min_{M_k \in \mathcal{P}_k} \left\| A \begin{pmatrix} 0 \\ * \\ 0 \\ * \\ 0 \\ 0 \\ 0 \end{pmatrix} - e_k \right\|_2^2 = \min_{M_k \in \mathcal{P}_k} \|A(:, \mathcal{J}_k)M_k(\mathcal{J}_k) - e_k\|_2^2$$

Denote by \mathcal{J}_k the set of allowed indices in the k th column of M .

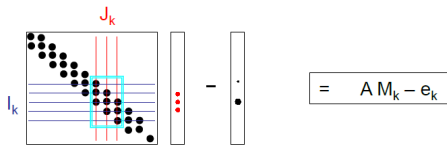


$$A \cdot \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} - \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} = A(:, \mathcal{J}_k) \cdot M(\mathcal{J}_k) - e_k = \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} \cdot \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix} - \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{matrix}$$



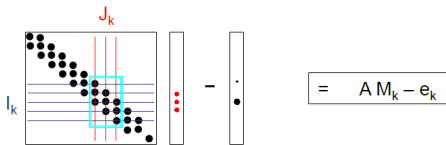
SPAI and Sparse Least Squares

- $A(:, \mathcal{J}_k)$ is a sparse rectangular matrix. \mathcal{I}_k is index set called *shadow* of \mathcal{J}_k .
- Delete superfluous zeros in least squares problem:
 - For index set \mathcal{J}_k in M_k keep only $A(:, \mathcal{J}_k)$
 - In $A(:, \mathcal{J}_k)$ keep only nonzero rows $A(\mathcal{I}_k, \mathcal{J}_k)$



SPAI and Sparse Least Squares

- $A(:, \mathcal{J}_k)$ is a sparse rectangular matrix. \mathcal{I}_k is index set called *shadow* of \mathcal{J}_k .
- Delete superfluous zeros in least squares problem:
 - For index set \mathcal{J}_k in M_k keep only $A(:, \mathcal{J}_k)$
 - In $A(:, \mathcal{J}_k)$ keep only nonzero rows $A(\mathcal{I}_k, \mathcal{J}_k)$



- Hence, reduction of sparse LS to $A(\mathcal{I}_k, \mathcal{J}_k)$:

$$\min_{M_k \in \mathcal{P}_k} \|A(\mathcal{I}_k, \mathcal{J}_k)M_k - e(\mathcal{I}_k)\|_2^2 = \min_{\tilde{M}_k \in \tilde{\mathcal{P}}_k} \|\tilde{A}\tilde{M}_k - \tilde{e}_k\|_2^2$$

- Solve small LS problem by Householder QR for

$$A(\mathcal{I}_k, \mathcal{J}_k) \rightarrow \tilde{M}_k \rightarrow M_k$$



Sparsity Pattern: Static A Priori Choice

- A^{-1} will be no more sparse!
- As a priori choice of a good approximate sparsity pattern for M we can choose the pattern of
 - A^k or $(A^T)^k$
 - $(A^T A)^k A^T$ for some $k = 1, 2$
 - A_ϵ with sparsified A
 - a combination of above
- A_ϵ by sparsification of A : delete all entries with $|A_{ij}| < \epsilon$

