Sparse approximate Inverses

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

$$\min_{M \in \mathcal{P}} \|AM - I\|$$

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: Frobenius norm

$$\|A\|_F^2 := \sum_{i,j=1}^{n} a_{i,j}^2 = \sum_{j=1}^{n} \| A_{.,j} \|^2 = \operatorname{trace}(A^T A) = \langle A, A \rangle$$
First, we choose the pattern $\mathbf{P}$ in a static way a priori, e.g. as the pattern of $A$

$$
\min_{M \in \mathcal{P}} \|AM - I\|^2_F = \min_{M \in \mathcal{P}} \sum_{j=1}^{n} \| (AM - I) e_j \|^2 = \\
= \sum_{j=1}^{n} \min_{M_j \in \mathcal{P}_{j}} \|AM_j - e_j\|^2
$$

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 LS problems?
SPAI and LS

\[ \min_{M_j \in \varphi_j} \|AM_j - e_j\| = \min A \left( \begin{array}{c} 0 \\ \ast \\ 0 \\ 0 \\ 0 \end{array} \right) - e_j = \min \|AM_j(J_j) - e_j\| \]

Denote by \( J_j \) the set of allowed indices in the \( j \)-th column of \( M \).

\[ A \begin{array}{c} \ast \\ \ast \\ \ast \end{array} = A(:,J_j) \cdot M(J_j) - e_j \]
SPAI and Sparse LS

$A(:,\text{J}_j)$ is a sparse rectangular matrix. \( I_j \) index set, shadow of \( J_j \)

We can reduce the sparse LS to \( A(I_j,J_j) \):

$$\min_{M_j} \left\| A(I_j,J_j)M(J_j) - e(I_j) \right\| = \min_{\tilde{M}_j} \left\| \tilde{A}\tilde{M}_j - \tilde{e}_j \right\|$$

Solve small LS problem by Householder QR for \( A(I_j,J_j) \) \( \Rightarrow \tilde{M}_j \to M_j \)
Computing $M_k$

Delete superfluous zeros in Least Squares Problem:

For index set $J_k$ in $M_k$ keep only $A(:\,\,J_k)$

In $A(:\,\,J_k)$ keep only nonzero rows $A(I_k,J_k)$

Solve small Least Squares problem in $A(I_k,J_k)$, e.g. by QR-decomposition, Householder method.
Sparsity Pattern?

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 \(k=1,2\)
- \(A_\varepsilon\) with sparsified \(A\)
- a combination of above

\(A_\varepsilon\) by sparsification of \(A\): delete all entries with

\(|A_{i,j}| < \varepsilon\)
Dynamic Pattern Finding (SPAI)

- Start with thin approximate pattern $J_k$ for $M_k$
- Compute optimal column $M_{k, \text{opt}}(J_k)$ by LS
- Find new entry $j$ for $M_k$ such that $M_{k, \text{opt}}(J_k) + \lambda e_j$
  has smaller residual in the Frobenius norm.

$$
\min \left\| A(M_k + \lambda e_j) - e_k \right\|^2 = \min \left\| (AM_k - e_k) + \lambda Ae_j \right\|^2 = \\
\min \left( \left\| r_k \right\|^2 + 2\lambda (r_k^T A_j) + \lambda^2 \left\| A_j \right\|^2 \right)
$$

Choose index $j$ with $r_k^T A_j \neq 0$ and $\lambda_{opt} = -\frac{r_k^T A_j}{\left\| A_j \right\|^2}$

Improvement: One can also find the overall optimal new entry.
Iterative SPAI

Start with pattern of $A \rightarrow M_1$
Construct $M_2$ relative to new matrix $AM_1$
Construct $M_3$ relative to new matrix $AM_1M_2$

......

Advantage: Cheaper, but inferior approximation

Block SPAI

Partition the given matrix in small blocks (2 x 2 or 3 x 3) and apply the Frobenius norm minimization with blockwise pattern.

Advantage: Underlying block structure will also appear in the pattern of $A^{-1} \rightarrow$ improved pattern

Block operations are more efficient
Factorized SPAI, FSPAI

Approximate the inverse Cholesky factor of $A = L_A^T \cdot L_A$

$$A^{-1} = (L_A^T L_A)^{-1} = L_A^{-1} L_A^{-T} \approx L \cdot L^T$$

$L_A^{-1} \approx L \quad \rightarrow \quad \min \|L_A L - I\|_F$

Normal equations give $A(J_k, J_k) \cdot L(J_k) = \alpha \cdot e_k$

The same $L$ results from $\min \frac{\text{trace}(L^T AL)}{\det(L^T AL)^{(1/n)}}$
SPA I and Target Matrices

\[
\min \|AM - P\|_F
\]

Assume, P is a good sparse preconditioner for A. Improve P by computing M and solving the above Frobenius norm minimization.

Application: Assume that A is given by two parts, e.g. an advection part and an diffusion part.

We can choose P as Laplacian relative to the diffusion part - easy to solve - and then we add M for improving P relative to the advection part.
Choose e.g. $e=(1,1,…,1)^T$, and preconditioner as diagonal matrix $D=\text{diag}(d_1,…,d_n)$.
Then we have to satisfy $e^TD = e^TS = f^T$.
After the computation of $f$ the solution is given by $d_j = f_j$.

Disadvantage: Can use only very special pattern for $M$ and special probing vectors $e$.
Example: tridiagonal probing
MSPAII Probing

Generalize Frobenius norm minimization to

\[
\min_M \|CM - B\|_F = \min_M \left\| \begin{pmatrix} C_0 \\ \rho \cdot u^T \end{pmatrix} M - \begin{pmatrix} B_0 \\ \rho \cdot v^T \end{pmatrix} \right\|_F
\]

For example

\[
\min_M \|CM - B\|_F = \min_M \left\| \begin{pmatrix} A \\ \rho \cdot e^T \end{pmatrix} M - \begin{pmatrix} I \\ \rho \cdot e^T \end{pmatrix} \right\|_F
\]

Original SPAI extended by an additional norm minimization to deliver especially good results on vector e.

Similarly

\[
\min_M \left\| \begin{pmatrix} I \\ \rho \cdot e^T \end{pmatrix} (C_0 M - B_0) \right\|_F = \min_M \|W(C_0 M - B_0)\|_F
\]

\[
\min_M \left\| \begin{pmatrix} I \\ \rho \cdot e^T \end{pmatrix} (AM - I) \right\|_F = \min_M \|W(AM - I)\|_F
\]
MSPAII Probing

MSPAII allows general probing with any pattern and any collection of vectors \( e \):

\[
\min_{\phi} \left\| M - \tilde{S} \right\|_F^2 + \rho^2 \left\| e^T M - e^T S \right\|_2^2
\]

with a sparse approximation \( \tilde{S} \) of \( S \)
Further research:

(1) Apply MSPAI with the probing feature to Multigrid or regularization problems (smoother, preconditioner)

(2) Iterative SPAI:
Compute first preconditioner $M_1$ and consider $R_k = e_k - AM_{1,k}$.
Sort columns in $M_1$ according to the size of the error with permutation $P$:
$M_1^*P$ has as first columns indices with small error.
Consider

$$P^T (AM_1 - I) P = \tilde{A} \tilde{M}_1 - I, \quad \tilde{M}_1 = \begin{pmatrix} M_{\text{good}} & M_{\text{bad}} \end{pmatrix}$$

Look for next $M_2$ in the form

$$M_2 = \begin{pmatrix} I & M_{2,p} \\ 0 & \end{pmatrix}$$

Apply SPAI for

$$\min_{M_{2,p}} \left\| (\tilde{A} \tilde{M}_1) M_{2,p} - \begin{pmatrix} 0 \\ I \end{pmatrix} \right\|_F$$

and repeat.
Available code:

SPAI, Block SPAI, and MSPAI are available in parallel version, e.g. PETSc.

Other current project: Factorized (Block) SPAI for symmetric matrices.
Iterative methods in parallel

(1) Asynchronous iterations:
   Iteration function: $x = \Phi(x)$, $\mathbb{R}^n \rightarrow \mathbb{R}^n$ using $p$ processors, $x$ divided into $p$ subblocks.
   For $k=1,2,…$
   For $i=1:p$
   $$x_{\text{new},i} = \begin{cases} 
   \Phi_i(x_1,\ldots,x_p) & \text{if} \quad i \in s(k) \\
   x_{\text{old},i} & \text{if} \quad i \notin s(k)
   \end{cases}$$
   where $\{s(k)\}_k$ is a sequence of nonempty subsets of $\{1,p\}$ and all indices appear infinitely often for $k \rightarrow \infty$.
   This code describes an asynchronous fixed point iteration.
   Convergence for spectral radius of GS less than 1.

(2) Block methods: Consider $Ax_j=b_j$, $j=1,\ldots,k$ simultaneously.
   Collect $B=(b_1 \ldots b_k)$.
   Generate Krylov subspace $K(A,B)=(B, AB, A^2B,\ldots)$
(3) Parallel cg:

```c
while ( sqrt(gamma) > epsilon * error_0 ) {
    if (iteration > 1)
        q = r + gamma / gamma_old * q;
    v = A * q;
    delta = dot(v,q);
    alpha = delta / gamma;
    x = x + alpha * q;
    r = r - alpha * v;
    gamma_old = gamma;
    gamma = dot(r,r);
    iteration = iteration + 1;
}
```

dot product: requires communication (MPI_ALLREduce)
Using non-blocking collective operations in order to hide communication!

Allows overlap of numerical computations and communications.

In MPI-1 only possible for point-to-point communication: MPI_ISEND and MPI_Irecv. Will be included in new MPI-3 standard.

Additional libraries necessary for collective operations!

Example: LibNBC (non-blocking collectives)
Pseudo-code for a non-blocking reduction:

```c
MPI_Request req;
int sbuf1[SIZE], rbuf1[SIZE], buf2[SIZE];

/* compute sbuf1 */
compute(sbuf1, SIZE);

/* start non-blocking allreduce of sbuf1 */
MPI_Iallreduce(sbuf1, rbuf1, SIZE, MPI_INT, MPI_SUM,
               MPI_COMM_WORLD, &req);

/* compute buf2 (independent of buf1) */
compute(buf2, SIZE);
MPI_WAIT(&req, &stat);

/* use data in rbuf1 */
evaluate(rbuf1, buf2, SIZE);
```

computation and communication

synchronisation

final computation
Application to cg:

NBC only useful in cg for the matrix-vector product.

Assume that the vector is distributed over 2D processor array

Assume a „matrix-free“ problem:
Matrix has not to be stored, but is given only implicitly!
Non-blocking matrix-vector product:

```c
void matrix_vector-mult(struct array_2d *v_in,
                         struct array_2d *v_out,
                         struct comm_data_t *comm_data)
{
    fill_buffers(v_in, &comm_data->send_buffers);
    start_send_boundaries(comm_data);
    volume_mult(v_in, v_out, comm_data);
    finish_send_boundaries(comm_data);
    mult_boundaries(v_out, &comm_data->recv_buffers);
}
```

Overlap between sending the boundary values to neighbors and computations with the inner data.
6. Domain Decomposition

Consider elliptic PDE on region $\Omega$ with boundary $\Gamma$, e.g. with Dirichlet boundary conditions.
Example:

$$\Delta u = u_{xx} + u_{yy} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \quad \text{in} \quad \Omega$$

$$u(x, y) \big|_{\Gamma} = g(x, y) \quad \text{in} \quad \Gamma$$

Parallel solution?
Overlapping DD

Partition region $\Omega$ in two regions $\Omega_1$ and $\Omega_2$, with new boundaries $\Gamma_1$ and $\Gamma_2$ which are partially given by old boundary $\Gamma$ and some unknown parts $\Gamma'_1$ and $\Gamma'_2$:
Overlapping DD II

We can discretize and solve the given PDE on $\Omega_1$ with boundary $\Gamma_1$, but we need the values of $u(x,y)$ on the new artificial boundary $\Gamma'_1$.

In a first step we can assume any values for $\Gamma'_1$, e.g. $u(x,y)=0$.

Then we can solve the linear system relative to region $\Omega_1$.

The same can be done in parallel for $\Omega_2$.

The values of the resulting solution on $\Gamma'_2$ can be used, to compute in the next step the solution for region $\Omega_2$ with boundary $\Gamma_2$.

In the same way we can use the resulting solution on $\Gamma'_1$, to compute the solution for region $\Omega_1$ with boundary $\Gamma_1$.

So we can generate solutions on the partial regions which provide us with approximate values for the unknown boundary of the other partial solution.

The sequence of solutions converges in each region against the solution on $\Omega$. 


Overlapping DD III

First step:
Solve in parallel the PDE on all small subdomains with certain boundary cond.

Second step:
Exchange boundary values

Repeat until convergence.

For getting better interior boundary values:
Solve whole problem on coarse mesh and interpolate
Overlapping DD IV

Matrix representation of overlapping DD:

First step:

Second step:

Repeat until convergence.

Green parts are related to the other domain and we assume to know the related components in the vector \( x \) of unknowns. They are moved to the right-hand-side \( b \).
Nonoverlapping DD

Partition region $\Omega$ $\Gamma$ in the form $\Omega_1$ $\Gamma_1$ $\Omega_2$ $\Gamma_2$ $\Omega_{\text{new}}$

Discretization of the original problem with numbering of the unknowns relative to the partitioning given by $\Omega_1$ and $\Omega_2$ leads to a linear system with a matrix in dissection form:
Nonoverlapping DD II

\[ f = Au = \begin{pmatrix} A_1 & 0 & F_1 \\ 0 & A_2 & F_2 \\ G_1 & G_2 & A_3 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \]

A_3 is the so called interface matrix.

We can solve \( Au=f \) iteratively with pcg and preconditioner:

\[ \begin{pmatrix} A_1^{-1} & 0 & 0 \\ 0 & A_2^{-1} & 0 \\ 0 & 0 & M \end{pmatrix} \]

Here, for M we can use the identity or an approximate inverse for the Schur complement.

Hence, we reduce the original problem to two partial subproblems and one interface Schur complement system.
Nonoverlapping DD III

Leads to 16 block matrices on the diagonal $A_1, \ldots, A_{16}$ and Schur complement $S$.

\[
A = \begin{pmatrix}
A_1 & F_1 \\
\vdots & \vdots \\
A_{16} & F_{16} \\
G_1 & \cdots & G_{16} & A_{17}
\end{pmatrix}, \quad S = A_{17} - G_1 A_1^{-1} F_1 - \cdots - G_{16} A_{16}^{-1} F_{16}
\]

Solve small problems e.g. with Multigrid in parallel.

Precondition Schur complement e.g. with MSPAI

Overlapping easy parallel
But slow convergence

Nonoverlapping harder parallel
But more influence on Convergence in $S$
Multigrid

Starting point: Solve Partial Differential Equation, e.g.

\[-u_{xx} - u_{yy} = f(x, y)\]

with boundary conditions, e.g. Dirichlet BC.

Discretization leads to system of linear equations, resp. matrix $A$.

$A$ is sparse, (structured,) and ill-conditioned. 
Looking for $O(n)$ solver.

Direct solver $O(n^2)$ or $O(n \log(n))$
PCG also $> O(n)$ because matrix is ill-conditioned
Idea

Vector of unknowns:

(1) Project the fine discretization on a coarser (smaller) problem

(2) Solve the coarse matrix (smaller matrix)

(3) Project the coarse solution back on fine grid

Problems: - only possible for smooth vector without high oscillatory components - backprojection introduces (high-oscillatory) errors that have to be removed, too.
Observation:

For typical PDE matrices high-oscillatory vectors are related to the subspace to large eigenvalues and are removed e.g. by the stationary Gauss-Seidel iteration:

\[ || I - M^{-1}A || \text{ small for eigenvectors to large eigenvalues!} \]

To solve \( Ax = f \):

1. Apply a few steps Gauss-Seidel smoothing steps \( \rightarrow x_a \)
   
   Residual equation \( A(x_a+x) = f \rightarrow Ax = f - x_a = r \)

2. Project (restrict) \( r \), resp. \( A \) on coarse grid by mean value
   
   Solve \( A_c x_c = r_c \)

3. Project (prolongate) \( x_c \) back on fine grid \( x \) by interpolation
   
   \( x_c \rightarrow x \rightarrow \text{new approximate solution} \ x_a + x \)

4. Improve approximate solution by Gauss-Seidel steps
   
   Repeat until convergence

   Apply also recursively for solving coarse equations
V-Cycle

Fine grid, smoothing

restriction

smoothing

restriction

smoothing

restriction

smoothing

restriction

Direct solution of coarse residual equation

Repeat until convergence