Direct Algorithms for Sparse Schur Complements and Inverses

Dr. Ryan Chilton

MyraMath

myracore.com

Computing Entries of inv(A)

Generally speaking, S is dense even when the A, B, C operands are sparse. The only structurally dense part is just one triangle of S = B^T * inv(A) * B as a LowerMatrix (coating half the memory and half the time). It returns a fully populated Matrix representing S = B^T * inv(A) * B.
Outline

Examine some less common sparse direct algorithms:

- Partial linear solution.
- Schur complements.
- Sampling the inverse operator.

Apply them as “frontends” for low-rank skeletonization:

- Cross approximation.
- Range estimation.
- Ritz projection.

Motivations: fast direct solvers for FE-BI’s and FE-DDM’s.
Refresher: Factor $A = LL^T$

Reorder: Left(0), Right(1), Separator(2). $A_{01} = A_{10} =$ all zero!

Right looking. Factor $A_{00}/A_{11}$, schur downdate $A_{22}$, factor.

FEM mesh:

Reordered matrix:

$$
\begin{bmatrix}
A_{00} & 0 & A_{02} \\
0 & A_{11} & A_{12} \\
A_{20} & A_{21} & A_{22}
\end{bmatrix}
$$

Separator induces these zeroes. They can’t fill-in!

Algorithm steps:

1. Factor $A_{22}$
2. Schur Downdate $A_{22}$
3. Solve $A_{20}$
4. Solve $A_{21}$
5. Factor $A_{00}$
6. Factor $A_{11}$

Note $A_{00}$ and $A_{11}$ also sparse, apply idea recursively.

Leads to a tree of operations, eliminating from bottom up.
Selected profiling data.

Example problem under study: \( I \times J \times K \) brick \((N = IJK)\)

Discrete graph laplacian (7-point): well understood spectrum.

Structured grid: easy to reorder using nested dissection.
Partial solution $x = R_i^T A^{-1} R_j b$

In plain english: only $b(j)$ nonzero, only $x(i)$ is needed.

Many engineering QoI’s use only **boundary-valued** $b$ and $x$.

$O(n^{4/3})$ time, like $x = A^{-1} b$. Only $O(n^{2/3})$ space per RHS, not $O(n)$. 
Schur complement $S = B^T A^{-1} B$

Concept: form “saddle system” of $A$ and $B$, then “quit” early.

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} = \begin{bmatrix} L & 0 \\ (L^{-1}B)^T & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & B^T (LL^T)^{-1} B \end{bmatrix} \begin{bmatrix} L^T & L^{-1}B \\ 0 & I \end{bmatrix}$$

Arise from FE-BI hybrids, eg scattering from apertures.

$$A_{ij} = \int_V (\nabla \times \vec{w}_i \cdot \mu_r^{-1} \nabla \times \vec{w}_j - k^2 \vec{w}_i \cdot \epsilon_r \vec{w}_j) \, dv$$

$$B_{ij} = \int_S \nabla \cdot (\hat{n} \times \vec{w}_i) \cdot \int_{S'} \nabla' \cdot (\hat{n}' \times \vec{w}_j) \cdot g(\vec{r}, \vec{r}') \, dS' \, dS$$

$$- k^2 \int_S (\hat{n} \times \vec{w}_i) \cdot \int_{S'} (\hat{n}' \times \vec{w}_j) \cdot g(\vec{r}, \vec{r}') \, dS' \, dS$$

$$\begin{bmatrix} A_{ee} & A_{em} \\ A_{me} & A_{mm} + 2B_{mm} \end{bmatrix} \begin{bmatrix} e \\ m \end{bmatrix} = \begin{bmatrix} 0 \\ h_{inc} \end{bmatrix}$$

$$\begin{bmatrix} A_{mm} - A_{em}^{T} A_{ee}^{-1} A_{em} + 2B_{mm} \end{bmatrix} \begin{bmatrix} m \end{bmatrix} = \begin{bmatrix} h_{inc} \end{bmatrix}$$
Sampling the inverse $Z(i,j)$, $Z = A^{-1}$

Closely related to Schur complement, $Z(i,j) = R_i^T \cdot A^{-1} \cdot R_j$

Arise in FETI/DDM, iterate/exchange fields at boundaries.

$$\begin{bmatrix}
I & I - 2\alpha R_2^T A_2^{-1} R_2 \\
I - 2\alpha R_1^T A_1^{-1} R_1 & I
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\lambda_2
\end{bmatrix} = \begin{bmatrix}
2\alpha R_1^T A_1^{-1} f_1 \\
2\alpha R_2^T A_2^{-1} f_2
\end{bmatrix}$$

Scatter, solve, gather. Scatter, solve, gather.

Tabulating $Z(i,j)$ opens up reuse/preconditioning options.
Cross Approximating $Z(i,j)$ [1/2]

Alternately sample row/column with largest error modulus.

Key idea: partialsolve() can efficiently extract rows/columns:

\[
c = Z([i],j) = \text{solver.partialsolve}([i],j,x=1.0, \text{'}Left\text{'})
\]
\[
r = Z(i,[j]) = \text{solver.partialsolve}(i,[j],x=1.0, \text{'}Right\text{'})
\]
Cross Approximating $Z(i,j)$ [2/2]

Beats `solver.inverse()` at large $N$, especially at low rank/tol.

But in parallel the gap narrows, BLAS3 vs BLAS1 effects.
Range estimation of $Z(i,j)$ [1/2]

Apply action of $Z$ to random vectors $X$, form image $Y = ZX$.
If $Z$ has rapidly decaying $\sigma$’s, $Y$ probably spans range($Z$).

```
// Find $Q = \text{span}(Z)$
X = \text{rand}(Z.cols,k)
Y = Z.apply(X)
[Q,R,\pi] = \text{QR}(Y,0)

// Build $k$-SVD from $Q$
W = Z'.apply(Q)
[U,\Sigma,V] = \text{svd}(W,0)
Z \approx (Q \cdot U) \cdot \Sigma \cdot (V)
```

Key idea: `partialsolve()` can efficiently apply $Y = Z(i,j) \cdot X$:

```
Y = Z([i],[j]) \cdot X = \text{solver.partialsolve}([i],[j],X,'Left')
```
Range estimation of $Z(i,j)$ [2/2]

All the same problem instances as before (sizes, shapes).

Availability of all forcing data up front leads to speedup.
Can be faster than parallel solver.inverse(), even at modest $N$. 
Ritz Projection of $Z(i,j)$ [1/3]

What about approximating *more* than just one block?

Optimization (BLR)/amortization (H) opportunities do exist.
Ritz Projection of $Z(i,j)$ [2/3]

First pass: find row/column spans using “fat” partialsolve()

$$Y = \text{partialsolve}([\text{all}],[\text{all}],X)$$

Second pass: Ritz projection using solver.schur(), k-SVD
Ritz Projection of $Z(i,j)$ [3/3]

Fill an H-matrix representation of $Z$ restricted to boundary.

Algorithm quickly furnishes all (admissible) blocks.

Can form H-matrix of $S=B^TA^{-1}B$ with a few minor changes.
Examined several uncommon sparse direct algorithms:

- Partial linear solution: \( x = R_i^T A^{-1} R_j b \) (sparse \( b \), sifted \( x \))
- Schur complements: \( B^T A^{-1} B, \ B^T A^{-1} C \), all sparse
- Sampling the inverse operator: \( Z(i,j) = R_i A^{-1} R_j \)

Used them as “frontends” for low-rank/skeletonization:

- Cross approximation: \text{partialsolve()}\ can extract row/column
- Range estimation: \text{partialsolve()}\ can apply \( Z(i,j) \) quickly
- Ritz projection: \text{schur()}+\text{partialsolve()}, amortization over blocks

Essential tools for FEBI/DDM methods (sparsity+lowrank).
Contact: myracore.com

MyraMath: sparse factor/solve/schur/inverse/partialsolve.

MyraKL: BLAS/LAPACK API for MyraMath, or use MKL.

Free software (GPL), or dual license (info@myracore.com)