Lecture 16:
Sparse Direct Solvers

David Bindel

17 Mar 2010

HW 3

Given serial implementation of:

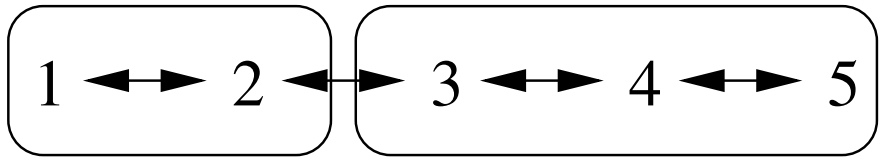
- 3D Poisson solver on a regular mesh
- PCG solver with SSOR and additive Schwarz preconditioners

Wanted:

- Basic timing experiments
- Parallelized CG solver (MPI or OpenMP)
- Study of scaling with n, p

Reminder: Sparsity and partitioning

$$A = \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ & & & * & * \end{bmatrix}$$

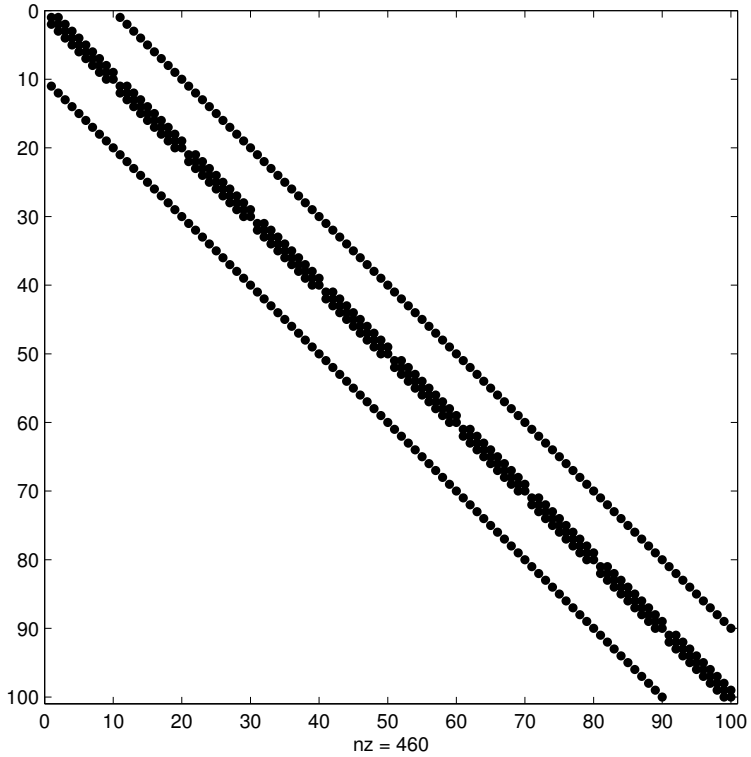


For SpMV, want to partition sparse graphs so that

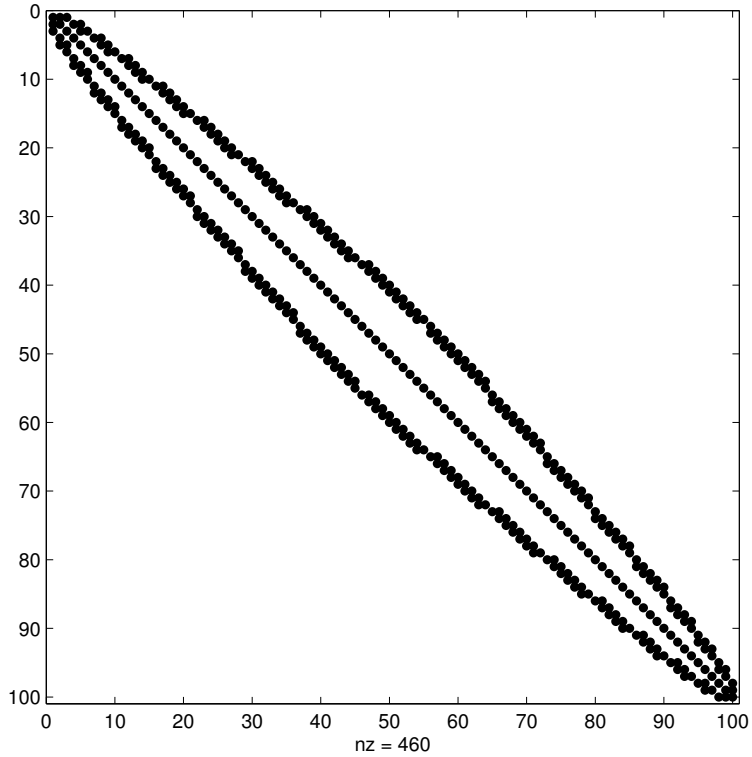
- Subgraphs are same size (load balance)
- Cut size is minimal (minimize communication)

Matrices that are “almost” diagonal are good?

Reordering for bandedness



Natural order



RCM reordering

Reverse Cuthill-McKee

- Select “peripheral” vertex v
- Order according to breadth first search from v
- Reverse ordering

From iterative to direct

- RCM ordering is great for SpMV
- But isn't narrow banding good for solvers, too?
  - LU takes $O(nb^2)$ where b is bandwidth.
  - Great if there's an ordering where b is small!

Skylines and profiles

- *Profile* solvers generalize band solvers
- Use skyline storage; if storing lower triangle, for each row i:
  - Start and end of storage for nonzeros in row.
  - *Contiguous* nonzero list up to main diagonal.
- In each column, first nonzero defines a profile.
- All fill-in confined to profile.
- RCM is again a good ordering.

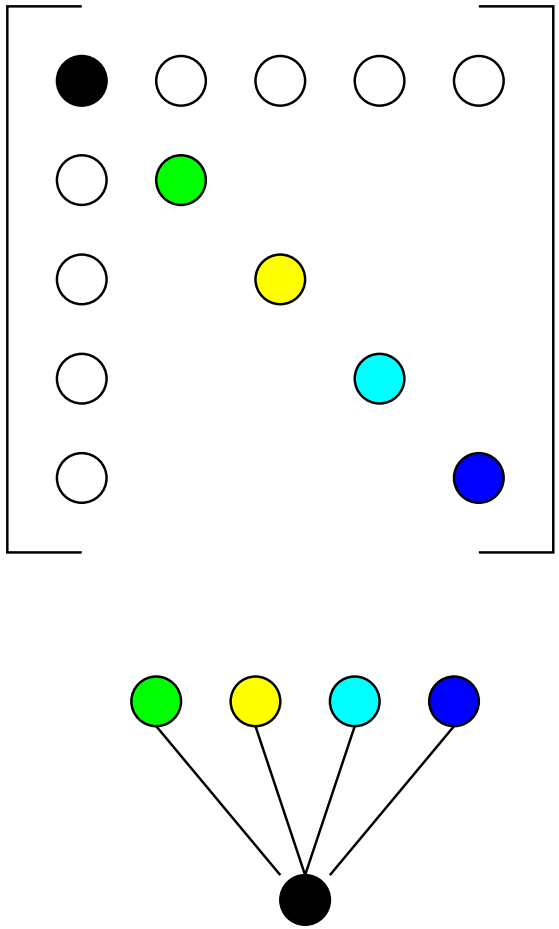
Beyond bandedness

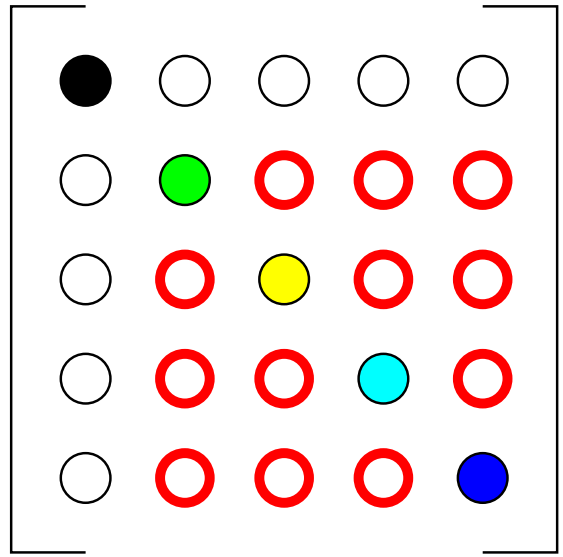
- Bandedness only takes us so far
  - Minimum bandwidth for 2D model problem? 3D?
  - Skyline only gets us so much farther
- But more general solvers have similar structure
  - Ordering (minimize fill)
  - Symbolic factorization (where will fill be?)
  - Numerical factorization (pivoting?)
  - ... and triangular solves

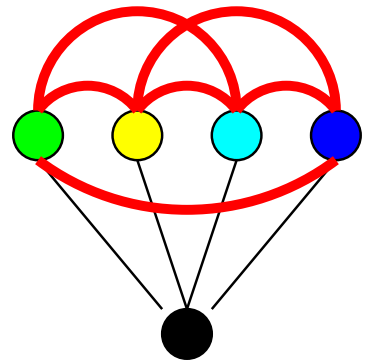
Reminder: Matrices to graphs

- $A_{ij} \neq 0$ means there is an edge between i and j
- Ignore self-loops and weights for the moment
- Symmetric matrices correspond to undirected graphs

Troublesome Trees

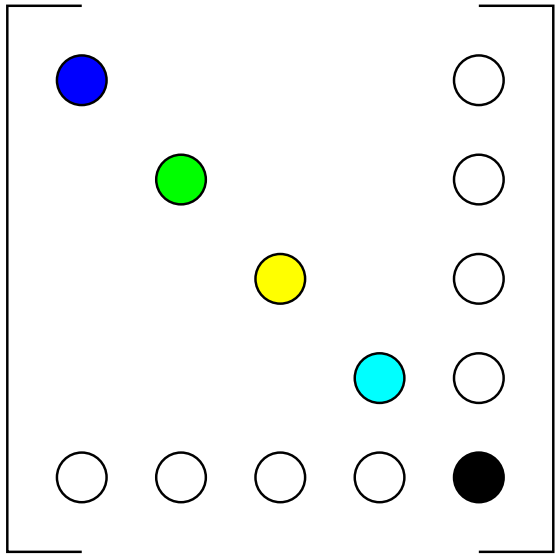


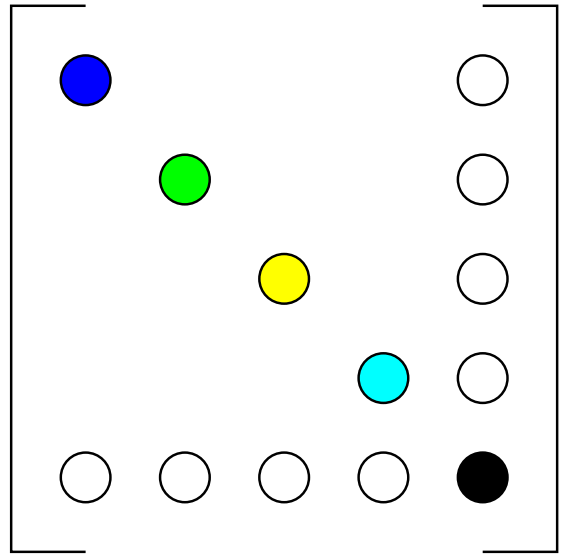


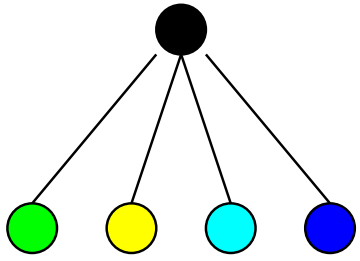


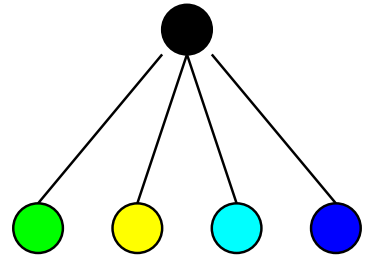
One step of Gaussian elimination *completely* fills this matrix!

Terrific Trees



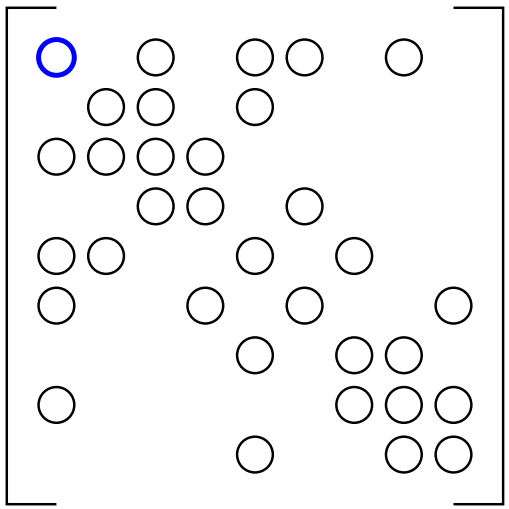


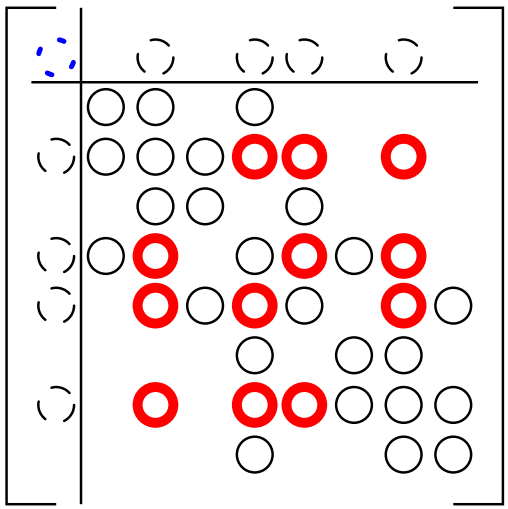


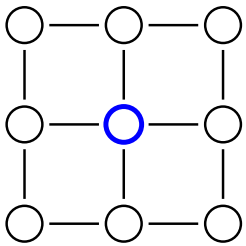


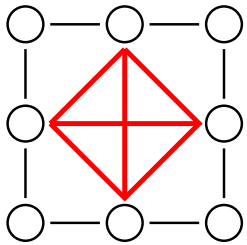
Full Gaussian elimination generates *no* fill in this matrix!

Graphic Elimination









Eliminate a variable, connect all neighbors.

Graphic Elimination

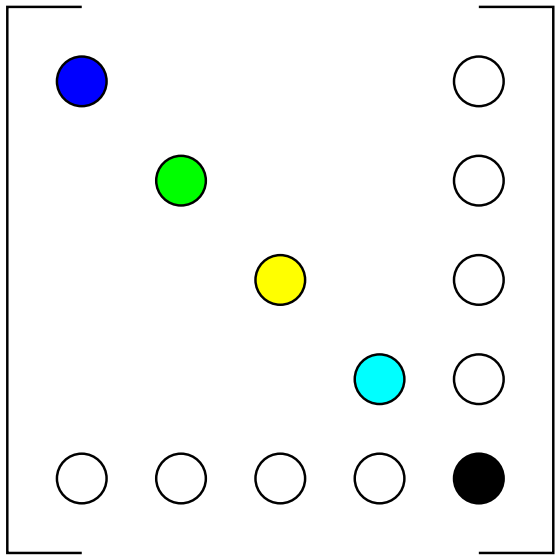
Consider first steps of GE

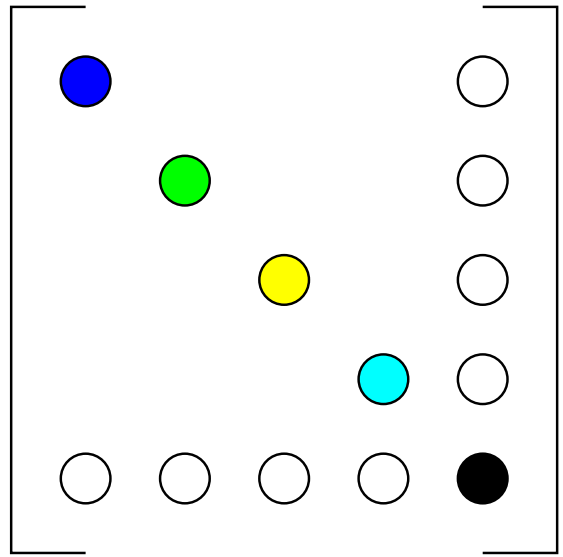
```
A(2:end,1)       = A(2:end,1)/A(1,1);
A(2:end,2:end)   = A(2:end,2:end)-...
                     A(2:end,1)*A(1,2:end);
```

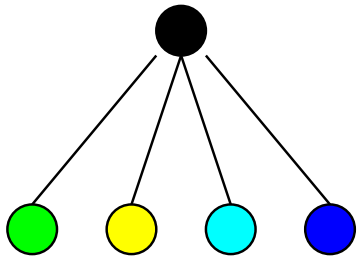
Nonzero in the outer product at (i, j) if `A(i,1)` and `A(j,1)` both nonzero — that is, if i and j are both connected to 1.

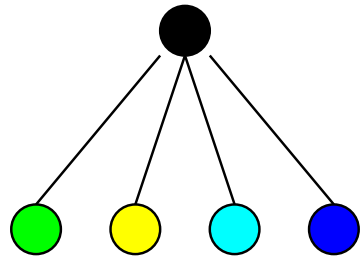
General: Eliminate variable, connect remaining neighbors.

Terrific Trees Redux



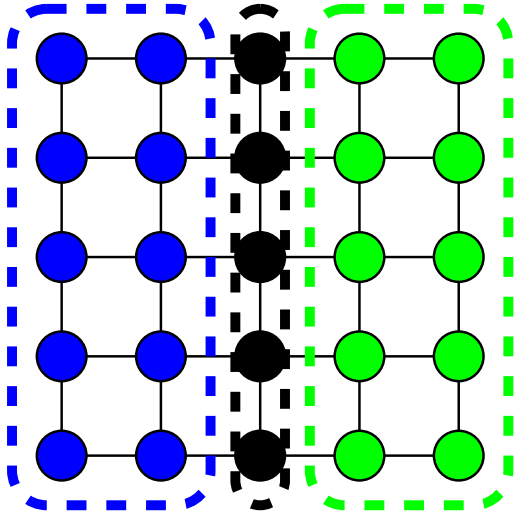


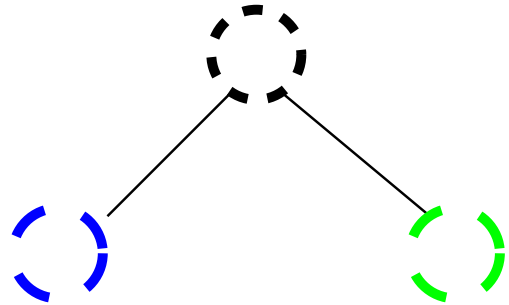




Order leaves to root $\Longrightarrow$
on eliminating i, parent of i is only remaining neighbor.

Nested Dissection





- Idea: Think of *block* tree structures.
- Eliminate block trees from bottom up.
- Can recursively partition at leaves.
- Rough cost estimate: how much just to factor dense Schur complements associated with separators?
- Notice graph partitioning appears again!
  - And again we want small separators!

Nested Dissection

Model problem: Laplacian with 5 point stencil (for 2D)

- ND gives optimal complexity in exact arithmetic (George 73, Hoffman/Martin/Rose)
- 2D: $O(N \log N)$ memory, $O(N^{3/2})$ flops
- 3D: $O(N^{4/3})$ memory, $O(N^2)$ flops

Minimum Degree

- Locally greedy strategy
  - Want to minimize upper bound on fill-in
  - Fill $\leq$ (degree in remaining graph)^2
- At each step
  - Eliminate vertex with smallest degree
  - Update degrees of neighbors
- Problem: Expensive to implement!
  - But better varients via *quotient graphs*
  - Variants often used in practice

Elimination Tree

- Variables (columns) are nodes in trees
- j a descendant of k if eliminating j updates k
- Can eliminate disjoint subtrees in parallel!

Cache locality

Basic idea: exploit “supernodal” (dense) structures in factor

- e.g. arising from elimination of separator Schur complements in ND
- Other alternatives exist (multifrontal solvers)

Pivoting

Pivoting is a tremendous pain, particularly in distributed memory!

- Cholesky — no need to pivot!
- Threshold pivoting — pivot when things look dangerous
- Static pivoting — try to decide up front

What if things go wrong with threshold/static pivoting?
Common theme: Clean up sloppy solves with good residuals

Direct to iterative

Can improve solution by *iterative refinement*:

$$
\begin{aligned}
PAQ &\approx LU \\
x_0 &\approx QU^{-1}L^{-1}Pb \\
r_0 &= b - Ax_0 \\
x_1 &\approx x_0 + QU^{-1}L^{-1}Pr_0
\end{aligned}
$$

Looks like approximate Newton on $F(x) = Ax - b = 0$.
This is just a stationary iterative method!
Nonstationary methods work, too.

Variations on a theme

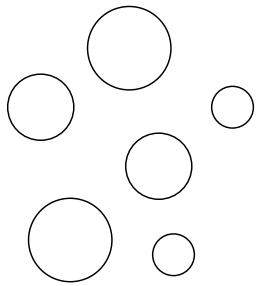
If we're willing to sacrifice some on factorization,

- Single precision + refinement on double precision residual?
- Sloppy factorizations (marginal stability) + refinement?
- Modify m small pivots as they're encountered (low rank updates), fix with m steps of a Krylov solver?

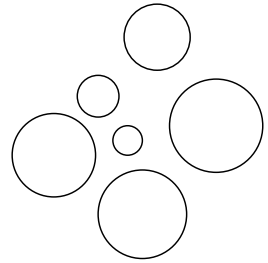
A fun twist

Let me tell you about something I've been thinking about...

Sparsification: a Motivating Example



A



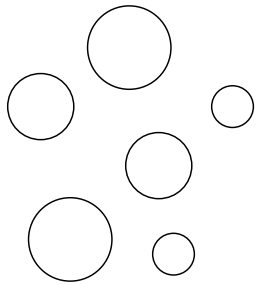
B

Gravitational potential at mass j from other masses is

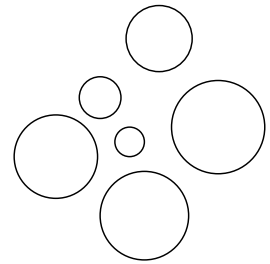
$$\phi_j(x) = \sum_{i \neq j} \frac{Gm_i}{|x_i - x_j|}.$$

In cluster A, don't *really* need everything about B. Just summarize.

A motivating example



A



B

Gravitational potential is a linear function of masses

$$\begin{bmatrix} \phi_A \\ \phi_B \end{bmatrix} = \begin{bmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{bmatrix} \begin{bmatrix} m_A \\ m_B \end{bmatrix}.$$

In cluster A, don't *really* need everything about B. Just summarize.
That is, represent P_{AB} (and P_{BA}) compactly.

Low-rank interactions

Summarize masses in B with a few variables:

$$z_B = V_B^T m_B, \quad m_B \in \mathbb{R}^{n_B}, z_B \in \mathbb{R}^p.$$

Then contribution to potential in cluster A is $U_A z_B$. Have

$$\phi_A \approx P_{AA} m_A + U_A V_B^T m_B.$$

Do the same with potential in cluster B; get system

$$\begin{bmatrix} \phi_A \\ \phi_B \end{bmatrix} = \begin{bmatrix} P_{AA} & U_A V_B^T \\ U_B V_A^T & P_{BB} \end{bmatrix} \begin{bmatrix} m_A \\ m_B \end{bmatrix}.$$

Idea is the basis of fast n-body methods (e.g. fast multipole method).

Sparsification

Want to solve $Ax = b$ where $A = S + UV^T$ is sparse plus low rank.

If we knew x, we could quickly compute b:

$$z = V^T x$$
$$b = Sx + Uz.$$

Use the same idea to write $Ax = b$ as a bordered system[1]:

$$\begin{bmatrix} S & U \\ V^T & -I \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Solve this using standard sparse solver package (e.g. UMFPACK).

[1] This is Sherman-Morrison in disguise

Sparsification in gravity example

Suppose we have ϕ, want to compute m in

$$\begin{bmatrix} \phi_A \\ \phi_B \end{bmatrix} = \begin{bmatrix} P_{AA} & U_A V_B^T \\ U_B V_A^T & P_{BB} \end{bmatrix} \begin{bmatrix} m_A \\ m_B \end{bmatrix}.$$

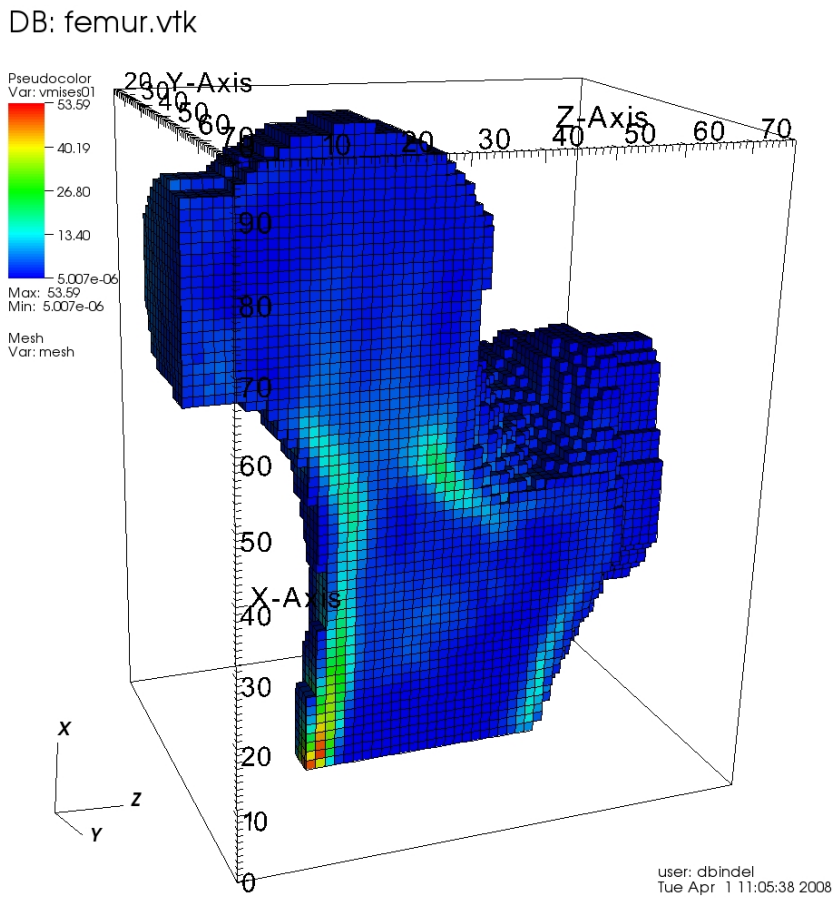
Add auxiliary variables to get

$$\begin{bmatrix} \phi_A \\ \phi_B \\ 0 \\ 0 \end{bmatrix} = \left[\begin{array}{cc|cc} P_{AA} & 0 & 0 & U_A \\ 0 & P_{BB} & U_B & 0 \\ \hline V_A^T & 0 & -I & 0 \\ 0 & V_B^T & 0 & -I \end{array}\right] \begin{bmatrix} m_A \\ m_B \\ z_A \\ z_B \end{bmatrix}.$$

Preliminary work

- Parallel sparsification routine (with Tim Mitchell)
  - User identifies low-rank blocks
  - Code factors the blocks and forms a sparse matrix as above
- Works pretty well on an example problem (charge on a capacitor)
- My goal state: Sparsification of separators for fast PDE solver

Goal state



I want a direct solver for this!