Lecture 3

Sparse Direct Method: Combinatorics

Xiaoye Sherry Li
Lawrence Berkeley National Laboratory, USA
xsli@lbl.gov

crd-legacy.lbl.gov/~xiaoye/G2S3/
Lecture outline

- Linear solvers: direct, iterative, hybrid
- Gaussian elimination
- Sparse Gaussian elimination: elimination graph, elimination tree
- Symbolic factorization, ordering, graph traversal
  - only integers, no FP involved
Strategies of sparse linear solvers

- Solving a system of linear equations $Ax = b$
  - **Sparse:** many zeros in $A$; worth special treatment
- Iterative methods (CG, GMRES, …)
  - $A$ is not changed (read-only)
  - Key kernel: sparse matrix-vector multiply
  - Easier to optimize and parallelize
  - Low algorithmic complexity, but may not converge
- Direct methods
  - $A$ is modified (factorized)
  - Harder to optimize and parallelize
  - Numerically robust, but higher algorithmic complexity
- Often use direct method (factorization) to **precondition** iterative method
  - Solve an easy system: $M^{-1}Ax = M^{-1}b$
Gaussian Elimination (GE)

Solving a system of linear equations $Ax = b$

First step of GE

$$A = \begin{bmatrix} \alpha \begin{bmatrix} w^T \\ v \end{bmatrix} \\ v \begin{bmatrix} v^T \\ B \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/\alpha & I \end{bmatrix} \cdot \begin{bmatrix} \alpha & w^T \\ 0 & C \end{bmatrix}$$

$$C = B - \frac{v \cdot w^T}{\alpha}$$

Repeat GE on $C$

Result in LU factorization ($A = LU$)
- L lower triangular with unit diagonal, U upper triangular

Then, $x$ is obtained by solving two triangular systems with $L$ and $U$
Numerical Stability: Need for Pivoting

One step of GE:

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/\alpha & I \end{bmatrix} \cdot \begin{bmatrix} \alpha & w^T \\ 0 & C \end{bmatrix}$$

$$C = B - \frac{v \cdot w^T}{\alpha}$$

- If $\alpha$ small, some entries in $B$ may be lost from addition

Pivoting: swap the current diagonal with a larger entry from the other part of the matrix

Goal: control element growth (pivot growth) in L & U
Sparse GE

- **Goal:** Store only nonzeros and perform operations only on nonzeros
- **Scalar algorithm:** 3 nested loops
  - Can re-arrange loops to get different variants: left-looking, right-looking, . . .
- **Fill:** new nonzeros in factor

Typical fill-ratio: 10x for 2D problems, 30-50x for 3D problems

```matlab
for i = 1 to n
    A(:,j) = A(:,j) / A(j,j)  \( \% \) cdv(j) col_scale
    for k = i+1 to n s.t. A(i,k) != 0
        for j = i+1 to n s.t. A(j,i) != 0
            A(j,k) = A(j,k) - A(j,i) \* A(i,k)
```
Useful tool to discover fill: Reachable Set

Given certain elimination order \((x_1, x_2, \ldots, x_n)\), how do you determine the fill-ins using original graph of A?

- **An implicit elimination model**

Definition: Let \(S\) be a subset of the node set. The reachable set of \(y\) through \(S\) is:
\[
\text{Reach}(y, S) = \{ x \mid \text{there exists a directed path } (y, v_1, \ldots, v_k, x), v_i \in S \}
\]

“Fill-path theorem” [Rose/Tarjan ’78] (general case):
Let \(G(A) = (V,E)\) be a directed graph of \(A\), then an edge \((v,w)\) exists in the filled graph \(G^+(A)\) if and only if
\[
w \in \text{Reach}(v, \{v_1, \ldots, v_k\}), \quad \text{where, } v_i < \min(v, w), \quad 1 \leq i \leq k
\]

- \(G^+(A) = \text{graph of the } \{L,U\} \text{ factors}\)
Concept of reachable set, fill-path

Edge \((x,y)\) exists in filled graph \(G^+\) due to the path: \(x \rightarrow 7 \rightarrow 3 \rightarrow 9 \rightarrow y\)

- Finding fill-ins \(\leftrightarrow\) finding transitive closure of \(G(A)\)
Sparse Column Cholesky Factorization $LL^T$

for $j = 1 : n$

$L(j:n, j) = A(j:n, j)$;

for $k < j$ with $L(j, k)$ nonzero

% sparse cmod(j,k)

$L(j:n, j) = L(j:n, j) - L(j, k) * L(j:n, k)$;

end;

% sparse cdiv(j)

$L(j, j) = \sqrt{L(j, j)}$;

$L(j+1:n, j) = L(j+1:n, j) / L(j, j)$;

end;

**Fill-path theorem** [George ’80] (symmetric case)

After $x_1, \ldots, x_i$ are eliminated, the set of nodes adjacent to $y$ in the elimination graph is given by $\text{Reach}(y, \{x_1, \ldots, x_i\})$, $x_i < \min(x, y)$
Elimination Tree

\[
T(A) : \text{parent}(j) = \min \{ i > j : (i, j) \text{ in } G^+(A) \}
\]

parent(col \ j) = first nonzero row below diagonal in \( L \)

- \( T \) describes dependencies among columns of factor
- Can compute \( G^+(A) \) easily from \( T \)
- Can compute \( T \) from \( G(A) \) in almost linear time
Symbolic Factorization

precursor to numerical factorization

- Elimination tree
- Nonzero counts
- Supernodes
- Nonzero structure of \{L, U\}

Cholesky [Davis’06 book, George/Liu’81 book]
  - Use elimination graph of L and its transitive reduction (elimination tree)
  - Complexity linear in output: $O(\text{nnz}(L))$

LU
  - Use elimination graphs of L & U and their transitive reductions (elimination DAGs) [Tarjan/Rose ‘78, Gilbert/Liu ‘93, Gilbert ‘94]
  - Improved by symmetric structure pruning [Eisenstat/Liu ‘92]
  - Improved by supernodes
  - Complexity greater than $\text{nnz}(L+U)$, but much smaller than $\text{flops}(LU)$
Can we reduce fill?

- Reordering, permutation

```
1 2 3 4 5
2 2
3 3 (all filled after elimination)
4 4
5 5
```

```
1 1 2 3 4 5
1 2 2
1 3 3
1 4 4
1 5 5
```

```
1 1 2 3 4 5
1 2 2
1 3 3
1 4 4
1 5 5
1 5
```

(no fill after elimination)
Fill-in in Sparse GE

➢ Original zero entry $A_{ij}$ becomes nonzero in L or U
  • Red: fill-ins

Natural order: NNZ = 233

Min. Degree order: NNZ = 207
### Ordering: Minimum Degree (1/3)

#### Graph game:

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>j</th>
<th>k</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>i</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eliminate 1

Maximum fill: all the edges between neighboring vertices ("clique")
Minimum Degree Ordering (2/3)

- **Greedy approach:** do the best locally
  - Best for modest size problems
  - Hard to parallelize

- At each step
  - Eliminate the vertex with the smallest degree
  - Update degrees of the neighbors

- Straightforward implementation is slow and requires too much memory
  - Newly added edges are more than eliminated vertices
Minimum Degree Ordering (3/3)

- Use quotient graph (QG) as a compact representation [George/Liu ’78]
- Collection of cliques resulting from the eliminated vertices affects the degree of an uneliminated vertex
- Represent each connected component in the eliminated subgraph by a single “supervertex”
- Storage required to implement QG model is bounded by size of A

Large body of literature on implementation variants
  ...

- Extended the QG model to nonsymmetric using bipartite graph [Amestoy/Li/Ng ‘07]
Ordering: Nested Dissection

- Model problem: discretized system $Ax = b$ from certain PDEs, e.g., 5-point stencil on $k \times k$ grid, $n = k^2$
- Recall fill-path theorem:
  After $x_1, \ldots, x_i$ are eliminated, the set of nodes adjacent to $y$ in the elimination graph is given by $\text{Reach}(y, \{x_1, \ldots, x_i\})$, $x_i < \min(x,y)$

![Diagram of nested dissection]

$D_1$ and $D_2$ denote the dissection regions, and $S$ is the separator set.
ND ordering: recursive application of bisection

ND gives a separator tree (i.e. elimination tree)
ND analysis on a square grid \((k \times k = n)\)

**Theorem** [George ’73, Hoffman/Martin/Ross]: ND ordering gave optimal complexity in exact factorization.

**Proof:**
- Apply ND by a sequence of “+” separators
- By “reachable set” argument, all the separators are essentially dense submatrices
- **Fill-in estimation:** add up the nonzeros in the separators

\[
k^2 + 4\left(\frac{k}{2}\right)^2 + 4^2\left(\frac{k}{4}\right)^2 + \cdots = O(k^2 \log_2 k) = O(n \log_2 n)
\]

( more precisely: \(31/4 \left(k^2 \log_2 k\right) + O(k^2)\) )

Similarly: Operation count: \(O(k^3) = O(n^{3/2})\)
Complexity of direct methods

Time and space to solve any problem on any well-shaped finite element mesh

<table>
<thead>
<tr>
<th></th>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space (fill):</td>
<td>$O(n \log n)$</td>
<td>$O(n^{4/3})$</td>
</tr>
<tr>
<td>Time (flops):</td>
<td>$O(n^{3/2})$</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>
ND Ordering: generalization

Generalized nested dissection [Lipton/Rose/Tarjan ’79]
- Global graph partitioning: top-down, divide-and-conquer
- First level

\[
\begin{bmatrix}
A & 0 & x \\
0 & B & x \\
x & x & S
\end{bmatrix}
\]

- Recurse on A and B

Goal: find the smallest possible separator S at each level
- Multilevel schemes:
  - (Par)Metis [Karypis/Kumar `95], Chaco
    [Hendrickson/Leland `94], (PT-)Scotch [Pellegrini et al. `07]
  - Spectral bisection [Simon et al. `90-`95]
  - Geometric and spectral bisection [Chan/Gilbert/Teng `94]
ND Ordering
CM / RCM Ordering

- Cuthill-McKee, Reverse Cuthill-McKee
- Reduce bandwidth
  - Construct level sets via breadth-first search, start from the vertex of minimum degree
  - At any level, priority is given to a vertex with smaller number of neighbors

[Diagram of graph with nodes labeled 1 to 17]

- RCM: Simply reverse the ordering found by CM
**RCM good for envelop (profile) Solver**
(also good for SpMV)

- Define bandwidth for each row or column
  - Data structure a little more sophisticated than band solver, but simpler than general sparse solver

- Use Skyline storage (SKS)
  - Lower triangle stored row by row
    - Upper triangle stored column by column
  - In each row (column), first nonzero defines a profile
  - All entries within the profile (some may be zeros) are stored
  - All the fill is contained inside the profile

- A good ordering would be based on bandwidth reduction
  - E.g., Reverse Cuthill-McKee
Envelop (profile) solver (2/2)

- **Lemma:** \( \text{env}(L+U) = \text{env}(A) \)
  - No more fill-ins generated outside the envelop!

**Inductive proof:** After \( N-1 \) steps,

\[
A = \begin{pmatrix} A_1 & w \end{pmatrix} \begin{pmatrix} v^T \\ s \end{pmatrix} = \begin{pmatrix} L_1 & w_1 \\ 1 & t \end{pmatrix}, \quad \text{s.t. } A_1 = L_1 U_1
\]

Then,

solve \( L_1 w_1 = w \), first nonzero position of \( w_1 \) is the same as \( w \)
solve \( U_1^T v_1 = v \), first nonzero position of \( v_1 \) is the same as \( v \)
Envelop vs. general solvers

- Example: 3 orderings (natural, RCM, MD)
- Envelop size = sum of bandwidths

Env = 31775  Env = 22320  Env = 61066
NNZ(L, MD) = 12259
Ordering for unsymmetric LU – symmetrization

- Can use a symmetric ordering on a symmetrized matrix . . .

  **Case of partial pivoting (sequential SuperLU):**
  Use ordering based on $A^TA$
  - If $R^TR = A^TA$ and $PA = LU$, then for any row permutation $P$,
    $\text{struct}(L+U) \subseteq \text{struct}(R^T+R)$ [George/Ng `87]
  - Making $R$ sparse tends to make $L$ & $U$ sparse . . .

  **Case of diagonal pivoting (static pivoting in SuperLU_DIST):**
  Use ordering based on $A^T+A$
  - If $R^TR = A^T+A$ and $A = LU$, then $\text{struct}(L+U) \subseteq \text{struct}(R^T+R)$
  - Making $R$ sparse tends to make $L$ & $U$ sparse . . .
Unsymmetric variant of “Min Degree” ordering (Markowitz scheme)

- Bipartite graph
- After a vertex is eliminated, all the row & column vertices adjacent to it become fully connected – “bi-clique” (assuming diagonal pivot)
- The edges of the bi-clique are the potential fill-ins (upper bound !)
Results of Markowitz ordering [Amestoy/Li/Ng’02]

- Extend the QG model to bipartite quotient graph
- Same asymptotic complexity as symmetric MD
  - Space is bounded by $2*(m + n)$
  - Time is bounded by $O(n \times m)$

For 50+ unsym. matrices, compared with MD on $A’+A$:
- Reduction in fill: average 0.88, best 0.38
- Reduction in FP operations: average 0.77, best 0.01

How about graph partitioning for unsymmetric LU?
- Hypergraph partition [Boman, Grigori, et al. `08]
- Similar to ND on $A^TA$, but no need to compute $A^TA$
Remark: Dense vs. Sparse GE

Dense GE: $P_r A P_c = LU$
- $P_r$ and $P_c$ are permutations chosen to maintain stability.
- Partial pivoting suffices in most cases: $P_r A = LU$

Sparse GE: $P_r A P_c = LU$
- $P_r$ and $P_c$ are chosen to maintain stability, preserve sparsity, increase parallelism.
- Dynamic pivoting causes dynamic structural change.
  - Alternatives: threshold pivoting, static pivoting, . . .
Numerical Pivoting

Goal of pivoting is to control element growth in L & U for stability

- For sparse factorizations, often relax the pivoting rule to trade with better sparsity and parallelism (e.g., threshold pivoting, static pivoting, . . .)

Partial pivoting used in sequential SuperLU and SuperLU_MT (GEPP)

- Can force diagonal pivoting (controlled by diagonal threshold)
- Hard to implement scalably for sparse factorization

Static pivoting used in SuperLU_DIST (GESP)

- Before factor, scale and permute A to maximize diagonal: $P_D r A D_c = A'$
- During factor $A' = LU$, replace tiny pivots by $\sqrt{\epsilon \|A\|}$, without changing data structures for L & U
- If needed, use a few steps of iterative refinement after the first solution

quite stable in practice
Use many heuristics

Finding an optimal fill-reducing ordering is NP-complete → use heuristics:
- Local approach: Minimum degree
- Global approach: Nested dissection (optimal in special case), RCM
- Hybrid: First permute the matrix globally to confine the fill-in, then reorder small parts using local heuristics
  - Local methods effective for smaller graph, global methods effective for larger graph

Numerical pivoting: trade-off stability with sparsity and parallelism
- Partial pivoting too restrictive
- Threshold pivoting
- Static pivoting
- …
Algorithmic phases in sparse GE

1. Minimize number of fill-ins, maximize parallelism
   - Sparsity structure of L & U depends on that of A, which can be changed by row/column permutations (vertex re-labeling of the underlying graph)
   - Ordering (combinatorial algorithms; “NP-complete” to find optimum [Yannakis ’83]; use heuristics)

2. Predict the fill-in positions in L & U
   - Symbolic factorization (combinatorial algorithms)

3. Design efficient data structure for storage and quick retrieval of the nonzeros
   - Compressed storage schemes

4. Perform factorization and triangular solutions
   - Numerical algorithms (F.P. operations only on nonzeros)
   - Usually dominate the total runtime

For sparse Cholesky and QR, the steps can be separate; for sparse LU with pivoting, steps 2 and 4 may be interleaved.
References

Exercises

Homework3 in Hands-On-Exercises/ directory

Show that:
If $R^T R = A^T + A$ and $A = LU$, then $\text{struct}(L+U) \subseteq \text{struct}(R^T+R)$

Show that: [George/Ng `87]
If $R^T R = A^T A$ and $PA = LU$, then for any row permutation $P$, $\text{struct}(L+U) \subseteq \text{struct}(R^T+R)$