Direct Methods for Sparse Linear Systems:

MATLAB sparse backslash

Tim Davis
davis@cise.ufl.edu

University of Florida
So what is a sparse matrix ... ?

a matrix “... that allows special techniques to take advantage of the large number of zero elements” (Wilkinson)

sparse matrices arise in a wide range of applications ...
Sparse matrices arise in ... computational fluid dynamics, finite-element methods, statistics, time/frequency domain circuit simulation, dynamic and static modeling of chemical processes, cryptography, magneto-hydrodynamics, electrical power systems, differential equations, quantum mechanics, structural mechanics (buildings, ships, aircraft, human body parts...), heat transfer, MRI reconstructions, vibroacoustics, linear and non-linear optimization, financial portfolios, semiconductor process simulation, economic modeling, oil reservoir modeling, astrophysics, crack propagation, Google page rank, 3D computer vision, cell phone tower placement, tomography, multibody simulation, model reduction, nano-technology, acoustic radiation, density functional theory, quadratic assignment, elastic properties of crystals, natural language processing, DNA electrophoresis, ...
Outline

- Sparse matrix algorithms: numerics plus graph theory
- Goal: sparse matrix methods from the ground up
- Lower triangular solve ($x = L \backslash b$)
- Sparse LU factorization ($[L, U, P] = \text{lu}(A)$)
- Sparse Cholesky factorization ($L = \text{chol}(A)'$)
- Supernodal and multifrontal methods ($x = A \backslash b$)
  - cache-friendly dense matrix kernels (BLAS)
  - supernodal (left-looking)
  - multifrontal (right-looking)

... next: sparse matrix data structures

SIAM 2006 – p. 4/25
Sparse data structures

- compressed sparse column format
- column $j$ is $A_i[A_p[j] \ldots A_p[j+1]-1]$, ditto in $Ax$
- Thus, $A(:,j)$ is easy in MATLAB; $A(i,:)$ hard

$$A = \begin{bmatrix}
4.5 & 0 & 3.2 & 0 \\
3.1 & 2.9 & 0 & 0.9 \\
0 & 1.7 & 3.0 & 0 \\
3.5 & 0.4 & 0 & 1.0
\end{bmatrix}$$

$A_p$: [0, 3, 6, 8, 10]
$A_i$: [0, 1, 3, 1, 2, 3, 0, 2, 1, 3]
$Ax$: [4.5, 3.1, 3.5, 2.9, 1.7, 0.4, 3.2, 3.0, 0.9, 1.0]
Sparse lower triangular solve, \( x = L \backslash b \)

\[
\begin{align*}
x &= b \\
\text{for } j &= 1:n \\
&\quad \text{if } (x(j) \neq 0) \\
&\quad \quad x(j+1:n) = x(j+1:n) - L(j+1:n,j) \times x(j)
\end{align*}
\]

- \( O(n+flops) \) time too high
- the problem:
  - for \( j=1:n \)
    - \( \text{if } (x(j) \neq 0) \)
  - need pattern of \( x \) before computing it
Sparse lower triangular solve, $x = L \backslash b$

\[
x = b \\
\text{for } j = 1:n \\
\quad \text{if } (x(j) \neq 0) \\
\quad \quad x(j+1:n) = x(j+1:n) - L(j+1:n,j) \times x(j) \\
\text{end}
\text{end}
\]

- $b_i \neq 0 \Rightarrow x_i \neq 0$
- $x_j \neq 0 \land l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- let $G(L)$ have an edge $j \rightarrow i$ if $l_{ij} \neq 0$
- let $\mathcal{B} = \{i \mid b_i \neq 0\}$ and $\mathcal{X} = \{i \mid x_i \neq 0\}$
- then $\mathcal{X} = \text{Reach}_{G(L)}(\mathcal{B})$
Sparse lower triangular solve, \( x = L \backslash b \)

If \( B = \{4, 6\} \),
then \( X = \{6, 10, 11, 4, 9, 12, 13, 14\} \)
Sparse lower triangular solve, \( x = L \backslash b \)

```matlab
def function x = lsolve(L,b)
    x = b
    for j = 1:n
        if (x(j) \neq 0)
            x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

Time: \( O(n + \text{flops}) \), need \( \mathcal{X} \) to get \( O(\text{flops}) \)
Sparse lower triangular solve, $x = \mathbb{L} \backslash b$

function $x = \text{lsolve}(\mathbb{L}, b)$
\[
\mathcal{X} = \text{Reach}(\mathbb{L}, \mathcal{B})
\]
\[
x = b
\]
for each $j$ in $\mathcal{X}$
\[
x(j+1:n) = x(j+1:n) - \mathbb{L}(j+1:n,j) \times x(j)
\]

function $\mathcal{X} = \text{Reach}(\mathbb{L}, \mathcal{B})$
for each $i$ in $\mathcal{B}$ do
if (node $i$ is unmarked) $\text{dfs}(i)$

function $\text{dfs}(j)$
mark node $j$
for each $i$ in $\mathcal{L}_j$ do
if (node $i$ is unmarked) $\text{dfs}(i)$
push $j$ onto stack for $\mathcal{X}$

Total time: $O(\text{flops})$
Sparse LU (Gilbert/Peierls)

- left-looking. $k$:th step computes $k$:th column of $L$ and $U$

$L = \text{speye}(n)$
$U = \text{speye}(n)$
for $k = 1:n$
  $x = L \setminus A(:,k)$
  $U(1:k,k) = x(1:k)$
  $L(k:n,k) = \ldots$
  $x(k:n) / U(k,k)$
end

$LU = PAQ$

- $P$: partial pivoting on $x$
- $Q$: fill-reducing column pre-ordering

$k$:th column of $L$ and $U$ computed

columns 1 to $k-1$ accessed

not modified or computed

SIAM 2006 – p. 11/25
Sparse Cholesky, $LL^T = A$

$$
\begin{bmatrix}
L_{11} & l_{T12} & l_{22} \\
l_{12}^T & L_{11} & l_{22} \\
l_{22} & l_{22} & l_{22}
\end{bmatrix}
\begin{bmatrix}
L_{11} & l_{12} \\
l_{12} & a_{12} \\
a_{12} & a_{22}
\end{bmatrix}
$$

1. factorize $L_{11}L_{11}^T = A_{11}$
2. solve $L_{11}l_{12} = a_{12}$ for $l_{12}$
3. $l_{22} = \sqrt{a_{22} - l_{12}^Tl_{12}}$

for $k = 1$ to $n$

solve $L_{11}l_{12} = a_{12}$ for $l_{12}$

$l_{22} = \sqrt{a_{22} - l_{12}^Tl_{12}}$

an up-looking method

accessed

compute $k$th row

not accessed
Sparse Cholesky: etree

- elimination tree
- arises in many direct methods
  - Compute nonzero pattern of $x = L \backslash b$ for a Cholesky $L$ in time $O(|x|)$, the number of nonzeros in $x$
  - ...

Sparse Cholesky: etree

Elimination tree $\mathcal{T}$: pruning the graph of $L$. Consider computing $k$th row of $L$:
Sparse Cholesky: etree

Elimination tree $T$: pruning the graph of $L$.
Consider computing $k$th row of $L$:

$$l_{ki} \neq 0 \iff x_i \neq 0$$

$$(l_{ji} \neq 0 \text{ and } x_i \neq 0) \Rightarrow x_j \neq 0$$

$$l_{kj} \neq 0 \iff x_j \neq 0$$

Thus, $l_{ki}$ redundant for $X = \text{Reach}(b)$.

- $\text{parent}(i) = \min\{j > i \mid l_{ji} \neq 0\}$; other edges redundant
- $\mathcal{L}_{k*} = \text{Reach}(A_{1:k,k})$ in $O(|\mathcal{L}_{k*}|)$ time
Sparse Cholesky: etree

\[ A \]

\[ \text{Cholesky factor } L \text{ of } A \]

\[ \text{elimination tree} \]
Sparse Cholesky: overview

- **Symbolic analysis:**
  - fill-reducing ordering, \( \bar{A} = P A P^T = LL^T \)
  - etree of \( \bar{A} \): nearly \( \mathcal{O}(|A|) \)
  - depth-first postordering of etree: \( \mathcal{O}(n) \)
  - column counts of \( L \): nearly \( \mathcal{O}(|A|) \)
  - some methods find \( L \): \( \mathcal{O}(|L|) \) or less

- **Numeric factorization:**
  - up-looking
  - left-looking, supernodal
  - right-looking, multifrontal
Sparse Cholesky: left-looking

for $k = 1$ to $n$

\begin{align*}
x &= A(k:n, k) \\
\text{for each } j \in \text{Reach}(L, A(1 : k, k)) &
\begin{align*}
x(k:n) &= x(k:n) - L(k:n, j) \times L(k, j) \\
L(k, k) &= \sqrt{x(k)} \\
L(k+1:n, k) &= x(k) / L(k, k)
\end{align*}
\end{align*}
Sparse Cholesky: supernodal

- Adjacent columns of $L$ often have identical pattern
- A chain in the elimination tree
- Can exploit dense submatrix operations
Sparse Cholesky: supernodal

block left-looking

for $j$th supernode:

(1) sparse block matrix multiply

(2) dense Cholesky

(3) dense block $Lx = b^T$ solve
Sparse LU: multifrontal
Sparse LU: UMFPACK

LU factors

A
**x = A \ b when A is sparse**

- if A diagonal: scale each row
- if A banded: LAPACK
- if A lower or upper triangular: forward/backsolve
- if A rectangular: Givens-based QR
- if A=A' and all(diag(A) > 0): try chol, supernodal or up-looking (CHOLMOD)
- else: lu, either multifrontal (UMFPACK) or left-looking (GPLU)
- http://www.cise.ufl.edu/research/sparse
Postscript


- Sparse Cholesky update/downdate

  - Given $A = LL^T$, compute $LL^T = A \pm ww^T$
  - "among the most important algorithms in linear algebra“, Wilkinson
  - time proportional to number of entries that change
  - columns of $L$ that change: sparsity pattern of $x=L\backslash w$
Direct methods for sparse linear systems

\[ x = A \backslash b \]

- Sparse matrix algorithms: numerics plus graph theory
- Goal: sparse matrix methods from the ground up
- Lower triangular solve \((x=L \backslash b)\)
- Sparse LU factorization \(([L, U, P]=lu(A))\)
- Sparse Cholesky factorization \((L=\text{chol}(A))'\)
- Supernodal and multifrontal methods \((x=A \backslash b)\)
- Sparse Cholesky update/downdate \((\text{cholupdate})\)