Parallel Numerical Algorithms
Chapter 9 – Band and Tridiagonal Systems

Prof. Michael T. Heath

Department of Computer Science
University of Illinois at Urbana-Champaign

CS 554 / CSE 512
Outline

1. Band Systems
2. Tridiagonal Systems
3. Cyclic Reduction
Band Systems
Tridiagonal Systems
Cyclic Reduction

Banded Linear Systems

- **Bandwidth** (or **semibandwidth**) of $n \times n$ matrix $A$ is smallest value $\beta$ such that

  $$a_{ij} = 0 \quad \text{for all} \quad |i - j| > \beta$$

- Matrix is **banded** if $\beta \ll n$

- If $\beta \gg p$, then minor modifications of parallel algorithms for dense LU or Cholesky factorization are reasonably efficient for solving banded linear system $Ax = b$

- If $\beta \lesssim p$, then standard parallel algorithms for LU or Cholesky factorization utilize few processors and are very inefficient
More efficient parallel algorithms for narrow banded linear systems are based on divide-and-conquer approach in which band is partitioned into multiple pieces that are processed simultaneously.

Reordering matrix by nested dissection is one example of this approach.

Because of fill, such methods generally require more total work than best serial algorithm for system with dense band.

We will illustrate for tridiagonal linear systems, for which $\beta = 1$, and will assume pivoting is not needed for stability (e.g., matrix is diagonally dominant or symmetric positive definite).
Tridiagonal Linear System

- *Tridiagonal* linear system has form

\[
\begin{bmatrix}
    b_1 & c_1 \\
    a_2 & b_2 & c_2 \\
    & \ddots & \ddots & \ddots \\
    & & a_{n-1} & b_{n-1} & c_{n-1} \\
    & & & a_n & b_n
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_{n-1} \\
    x_n
\end{bmatrix}
= \begin{bmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_{n-1} \\
    y_n
\end{bmatrix}
\]

- For tridiagonal system of order $n$, LU or Cholesky factorization incurs no fill, but yields serial thread of length $\Theta(n)$ through task graph, and hence no parallelism.

- Neither *cdivs* nor *cmods* can be done simultaneously.
Tridiagonal System, Natural Order

\[
G(A) \quad A \quad T(A)
\]

\[
L
\]

\[
G(A) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 5
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
10 & 9 & 0 & 0 & 0 \\
8 & 7 & 6 & 0 & 0 \\
5 & 4 & 3 & 2 & 0 \\
2 & 1 & 0 & 0 & 0
\end{bmatrix}
\]

\[
T(A) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 \\
0 & 0 & 0 & 7 & 0 \\
0 & 0 & 0 & 0 & 9
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 5
\end{bmatrix}
\]
Two-Way Elimination

- Other orderings may enable some degree of parallelism, however.

- For example, elimination from both ends (sometimes called *twisted* factorization) yields two concurrent threads (odd-numbered nodes and even-numbered nodes) through task graph and still incurs no fill.
Tridiagonal System, Two-Way Elimination

\[ G(A) \]

\[ T(A) \]

\[ L \]

\[ A \]
Repeating this idea recursively gives *odd-even* ordering (variant of nested dissection), which yields even more parallelism, but incurs some fill.
Tridiagonal System, Odd-Even Ordering

\[ A = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times \end{bmatrix} \]

\[ L = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times \end{bmatrix} \]

\[ G(A) \]

\[ T(A) \]
Cyclic Reduction

- Recursive nested dissection for tridiagonal system can be effectively implemented using cyclic reduction (or odd-even reduction).

- Linear combinations of adjacent equations in tridiagonal system are used to eliminate alternate unknowns.

- Adding appropriate multiples of \((i - 1)\)st and \((i + 1)\)st equations to \(i\)th equation eliminates \(x_{i-1}\) and \(x_{i+1}\), respectively, from \(i\)th equation.

- Resulting new \(i\)th equation involves \(x_{i-2}\), \(x_i\), and \(x_{i+2}\), but not \(x_{i-1}\) or \(x_{i+1}\).
Cyclic Reduction

- For tridiagonal system, $i$th equation

\[ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = y_i \]

is transformed into

\[ \bar{a}_i x_{i-2} + \bar{b}_i x_i + \bar{c}_i x_{i+2} = \bar{y}_i \]

where

\[
\begin{align*}
\bar{a}_i &= \alpha_i a_{i-1}, \\
\bar{b}_i &= b_i + \alpha_i c_{i-1} + \beta_i a_{i+1} \\
\bar{c}_i &= \beta_i c_{i+1}, \\
\bar{y}_i &= y_i + \alpha_i y_{i-1} + \beta_i y_{i+1}
\end{align*}
\]

with $\alpha_i = -a_i/b_{i-1}$ and $\beta_i = -c_i/b_{i+1}$
Cyclic Reduction

After transforming each equation in system (handling first two and last two equations as special cases), matrix of resulting new system has form

\[
\begin{bmatrix}
\bar{b}_1 & 0 & \bar{c}_1 \\
0 & \bar{b}_2 & 0 & \bar{c}_2 \\
\bar{a}_3 & 0 & \bar{b}_3 & 0 & \bar{c}_3 \\
& \ddots & \ddots & \ddots & \ddots \\
\bar{a}_{n-2} & 0 & \bar{b}_{n-2} & 0 & \bar{c}_{n-2} \\
\bar{a}_{n-1} & 0 & \bar{b}_{n-1} & 0 & \bar{c}_{n-2} \\
\bar{a}_n & 0 & \bar{b}_n & & 
\end{bmatrix}
\]
Cyclic Reduction

- Reordering equations and unknowns to place odd indices before even indices, matrix then has form

\[
\begin{bmatrix}
\bar{b}_1 & \bar{c}_1 \\
\bar{a}_3 & \bar{b}_3 & \ddots \\
& \ddots & \ddots & \ddots \\
& & \bar{a}_{n-1} & \bar{b}_{n-1} & 0 \\
& & & \bar{a}_n & \bar{b}_n
\end{bmatrix}
\]
Cyclic Reduction

- System breaks into two independent tridiagonal systems that can be solved simultaneously (i.e., divide-and-conquer)

- Each resulting tridiagonal system can in turn be solved using same technique (i.e., recursively)

- Thus, there are two distinct sources of potential parallelism
  - simultaneous transformation of equations in system
  - simultaneous solution of multiple tridiagonal subsystems
Cyclic reduction requires $\log n$ steps, each of which requires $\Theta(n)$ operations, so total work is $\Theta(n \log n)$.

Serially, cyclic reduction is therefore inferior to LU or Cholesky factorization, which require only $\Theta(n)$ work for tridiagonal system.

But in parallel, cyclic reduction can exploit up to $n$-fold parallelism and requires only $\Theta(\log n)$ time in best case.

Often matrix becomes approximately diagonal in fewer than $\log n$ steps, in which case reduction can be truncated and still attain acceptable accuracy.
Cost for solving tridiagonal system by best serial algorithm is about

\[ T_1 \approx 8 t_c n \]

where \( t_c \) is time for one addition or multiplication

Cost for solving tridiagonal system serially by cyclic reduction is about

\[ T_1 \approx 12 t_c n \log n \]

which means that efficiency is less than 67%, even with \( p = 1 \)
Parallel Cyclic Reduction

- **Partition**: task $i$ stores and performs reductions on $i$th equation of tridiagonal system, yielding $n$ fine-grain tasks.

- **Communicate**: data from “adjacent” equations is required to perform eliminations at each of $\log n$ stages.

- **Agglomerate**: $n/p$ equations assigned to each of $p$ coarse-grain tasks, thereby limiting communication to only $\log p$ stages.

- **Map**: Assigning contiguous rows to processes is better than cyclic mapping in this context.

- “Local” tridiagonal system within each process can be solved by serial cyclic reduction or by LU or Cholesky factorization.
Parallel execution time for cyclic reduction is about

\[ T_p \approx 12 t_c (n \log n)/p + (t_s + 4 t_w) \log p \]

To determine isoefficiency function relative to serial CR, set

\[ 12 t_c n \log n \approx E \left( 12 t_c (n \log n) + (t_s + 4 t_w) p \log p \right) \]

which holds for large \( p \) if \( n = \Theta(p) \), so isoefficiency function is at least \( \Theta(p \log p) \), since \( T_1 = \Theta(n \log n) \)

Problem size must grow even faster to maintain constant efficiency \( (E < 67\%) \) relative to best serial algorithm.
Relatively fine granularity may make cyclic reduction impractical for solving single tridiagonal system on some parallel architectures.

Efficiency may be much better, however, if there are many right-hand sides for single tridiagonal system or many independent tridiagonal systems to solve.

Cyclic reduction is also applicable to *block tridiagonal* systems, which have larger granularity and hence more favorable ratio of communication to computation and potentially better efficiency.
Iterative Methods

- Tridiagonal and other banded systems are often amenable to efficient parallel solution by iterative methods.

- For example, successive diagonal blocks of tridiagonal system can be assigned to separate tasks, which can solve “local” tridiagonal system as preconditioner for iterative method for overall system.
References – Banded Systems

References – Tridiagonal Systems


References – Tridiagonal Systems


