

# Parallel algorithm for tridiagonal matrix

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2007-08/msg00154.html>

---

- *From:* [arpiruk@xxxxxxxxxx](mailto:arpiruk@xxxxxxxxxx)
  - *Date:* Wed, 22 Aug 2007 11:59:24 -0700
- 

Hello everyone,

I have been searching for parallel algorithm for compact scheme, i.e. Lele1993 (J.Comp.Phy).

This means I have to solve  $Ax=Bf$ . The matrix A I intended to use is a tridiagonal matrix.

I believe it will always be positive definite on ordinary smooth grid. (still don't know how to prove it)

My data will be distributed in domain decomposition fashion. Topology is 2D grid.

Due to the coupling nature of compact scheme, each grid cannot solve the solution on its own and need some kind of communication with other grid.

I did some research and believe that fine-grained parallelism such as Recursive doubling (Stone1975 TOMS) and Cyclic reduction (Hockney1965 J.ACM ) are not suited for current supercomputers which are now very coarse grained ( A cluster with multi-core node ).

Family of pipelining algorithm also requires too much communications.

It seems that the reduced parallel diagonal dominant of SUN (<http://mack.itc.ku.edu/sun95application.html>) are the most efficient for this problem.

The complexity is  $5n/p+4J$  ( $p$ =#of proc,  $J$  = truncated bandwidth) which almost matches the best serial code ( which is  $5n-3$  ).

Another approach is transposed algorithm

(as I was suggested from <http://www.cfd-online.com/Forum/main.cgi?read=54361>)

where we collect the strip of data on each grid line and send to certain processor in that line. Then the data in each strip line can be solve on one processor. After solving it, the data can be distributed back to original processor.

Are you aware of any algorithm that is more efficient than this two algorithms ?

Are there any particular problem that I should be aware of concerning these two method?

## Parallel algorithm for tridiagonal matrix

My target problem will be a direct numerical simulation. The problem may take a size of  $N_x N_y N_z = 1000^3$ . Time step can take up to  $10E6$ . This problem should be well scalable up to 512 processors.

Any suggestions are appreciated.

Regards,  
Arpiruk

(also posted in math group)

.