A Parallel Linear Solver for Block Circulant Linear Systems with Applications to Acoustics

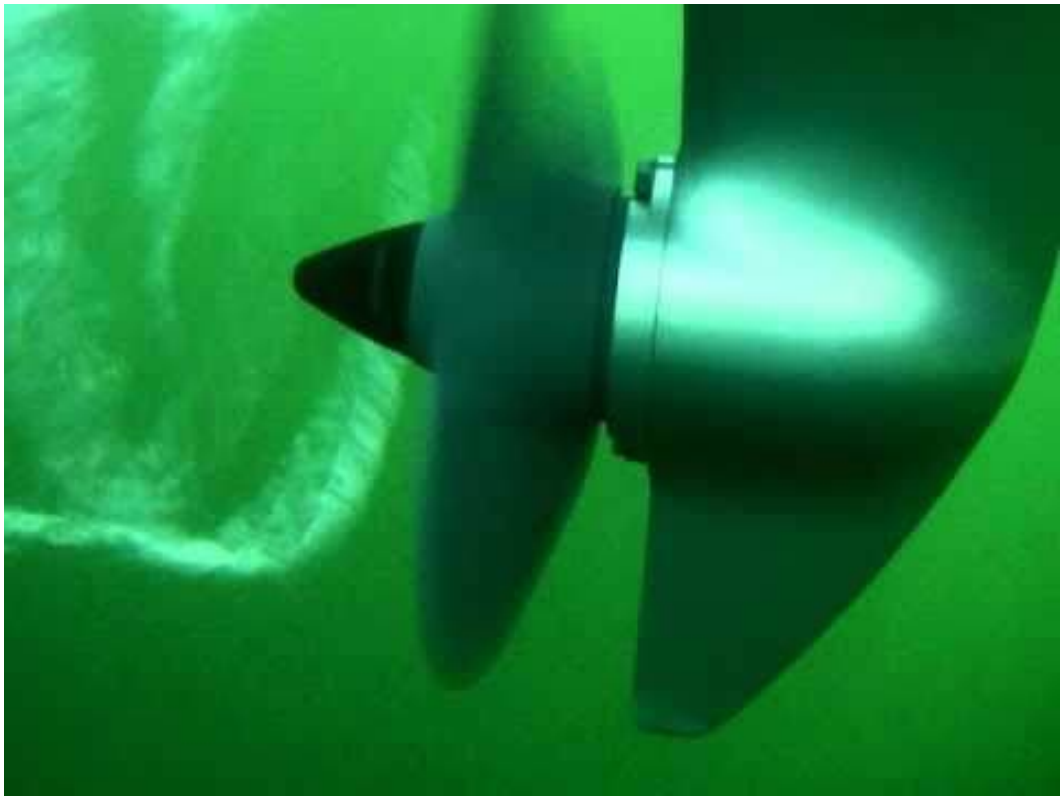
Suzanne Shontz, University of Kansas
Ken Czuprynski, University of Iowa
John Fahnline, Penn State

EECS 739: Parallel Scientific Computing
University of Kansas

January 23, 2019

MOTIVATION

Application: Vibrating Structures Immersed in Fluids



Example: Ships

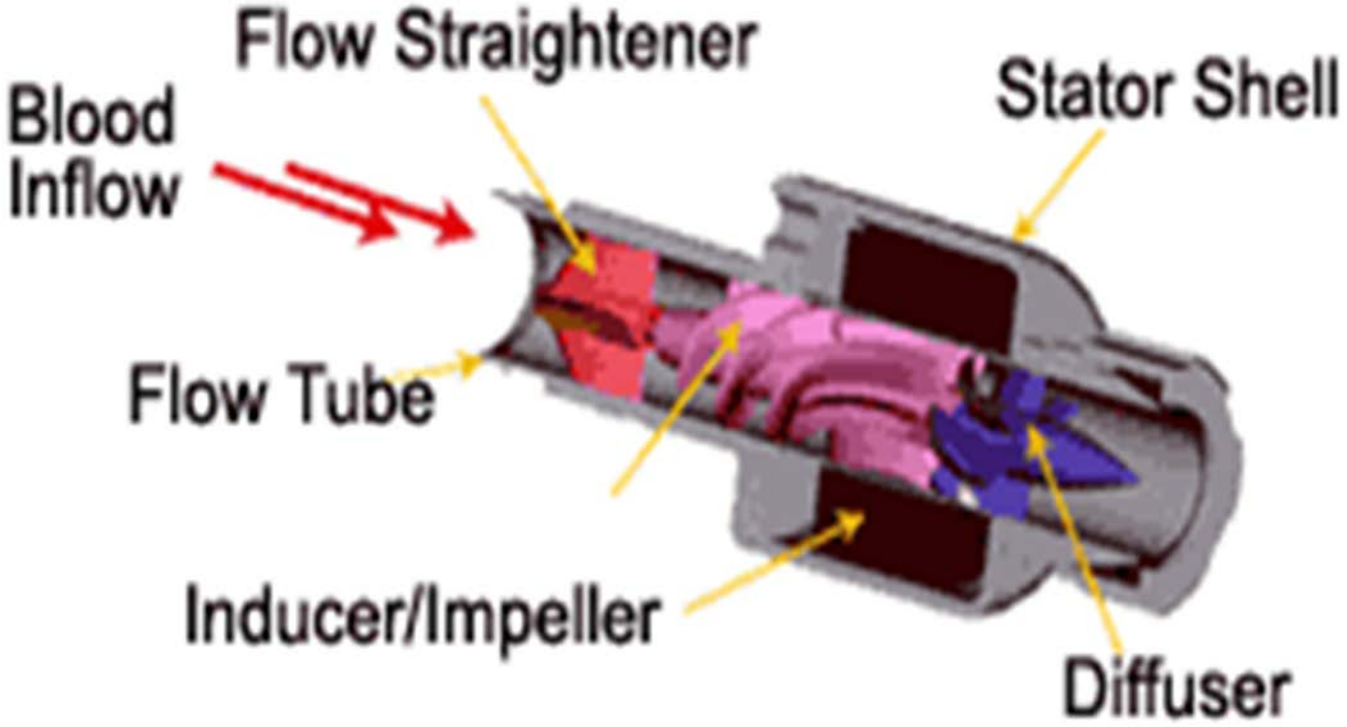
Application: Vibrating Structures Immersed in Fluids

Example: UAVs

Application: Vibrating Structures Immersed in Fluids

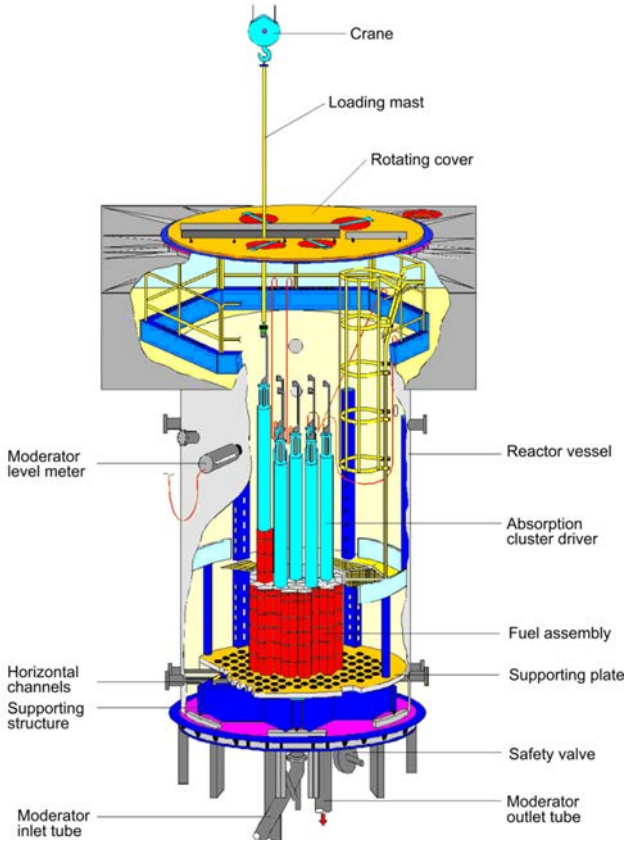
Example: Planes

Application: Vibrating Structures Immersed in Fluids



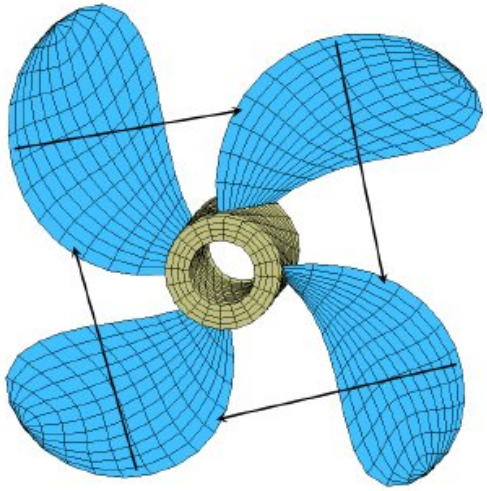
Example: Blood Pumps

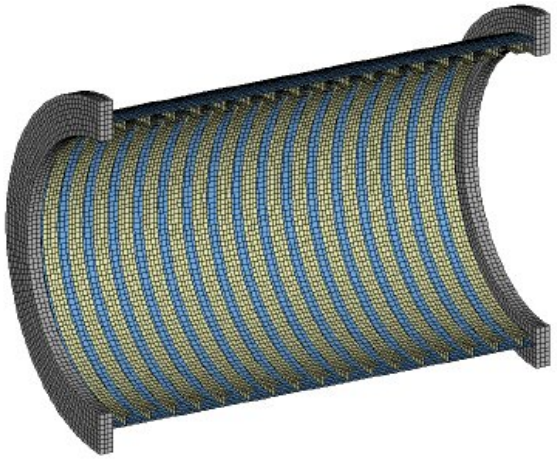
Application: Vibrating Structures Immersed in Fluids



Example: Reactors

Examples of Rotationally Symmetric Boundary Surfaces





Real-world applications: propellers, wind turbines, etcetera

THE PROBLEM

The Problem

Goal: To compute the acoustic radiation for a vibrating structure immersed in a fluid.

Our focus: Structures with rotationally symmetric boundary surfaces

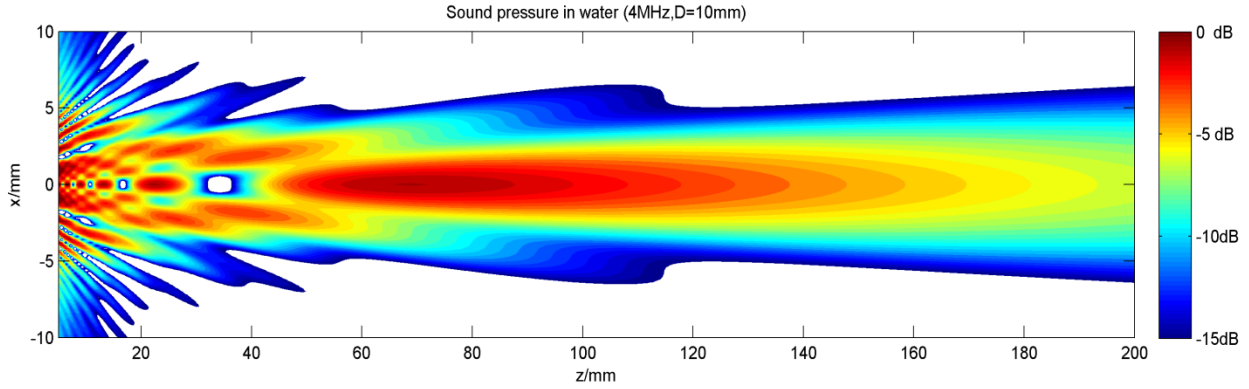


Image credit: Michael Lenz

Parallel Linear Solver for Acoustic Problems with Rotationally Symmetric Boundary Surfaces

Context: Vibrating structure immersed in fluid. Acoustic analysis using boundary element method. Coupled to a finite element method for the structural analysis. We focus on the boundary element part of the calculation.

Goal: Solve block circulant linear systems to compute acoustic radiation of vibrating structure with **rotationally symmetric boundary surface.**

Approach: Parallel linear solver for **distributed memory machines** based on known inversion formula for block circulant matrices.

THE BOUNDARY ELEMENT METHOD (BEM)

Boundary Element Method

The **boundary element method (BEM)** is a numerical method for solving linear partial differential equations (PDEs).

In particular, the BEM is a solution method for **solving boundary value problems (BVPs) formulated using a boundary integral formulation.**

Discretization: Only of the surface (not of the volume). Reduces dimension of problem by one.

BEM: Used on **exterior domain problems** and when **greater accuracy** is required.

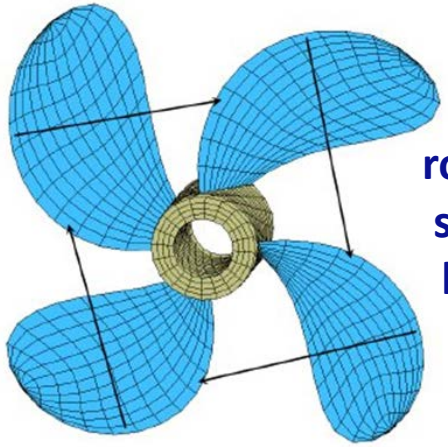
We employ the boundary element method to obtain the linear system of equations.

Comparison of the BEM with the Finite Element Method

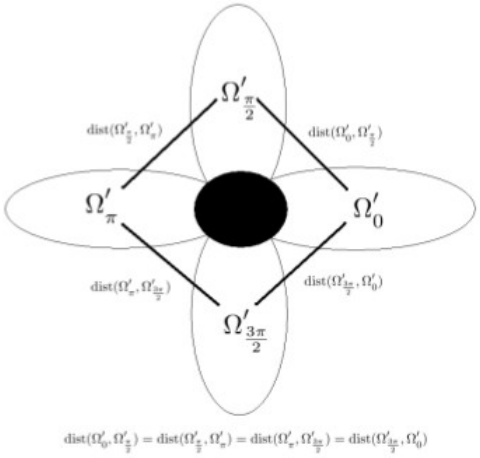
Advantages of the BEM	Disadvantages of the BEM
Less data preparation time (due to surface only modeling)	**Unfamiliar mathematics**
High resolution of PDE solution (e.g., stress)	**The interior must be modeled for nonlinear problems** (but can often be restricted to a region of the domain)
Less computer time and storage (fewer nodes and elements)	**Fully populated and unsymmetric solution matrix** (as opposed to being sparse and symmetric)
Less unwanted information (most "interesting behavior" happens on the surface)	**Poor for thin structures (shell) 3D analyses** (large surface/volume ratio causes inaccuracies in calculations)

BLOCK CIRCULANT MATRICES VIA THE BOUNDARY ELEMENT METHOD

Discretization Using the BEM



rotationally symmetric boundary surface



symmetry: m = 4

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_m \\ A_m & A_1 & \cdots & A_{m-1} \\ A_{m-1} & A_m & \cdots & A_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ A_2 & A_3 & \cdots & A_1 \end{bmatrix}$$

block circulant matrix

Block Circulant Matrices

- **Properties of circulant matrices:** Diagonalizable by Fourier matrix. Can use DFT and IDFT. **Nice properties!**

- **Related work (serial):** algorithm derived from inversion formula (Vescovo, 1997); derivation (Smyrlis and Karageorghis, 2006)

- **Related work (parallel):** parallel block Toeplitz matrix solver (Alonso et al., 2005) (neglects potential concurrent calculations); parallel linear solver for axisymmetric case (Padiy and Neytcheva, 1997)

MATHEMATICAL FORMULATION OF LINEAR SYSTEM OF EQUATIONS

Notation: Fourier Matrix

The **Fourier matrix** is given by

$$F = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_m^1 & \omega_m^2 & \cdots & \omega_m^{m-1} \\ 1 & \omega_m^2 & \omega_m^4 & \cdots & \omega_m^{2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_m^{(m-1)} & \omega_m^{2(m-1)} & \cdots & \omega_m^{(m-1)^2} \end{bmatrix}$$

where $\omega_m = e^{i2\pi/m}$.

Note: The Fourier matrix is used in Fourier transforms.

Discrete Fourier Transform (DFT)

To compute the discrete Fourier transform (DFT) of a vector x, simply multiply F times x.

Example: m = 4

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

F*x = u

Inverse Discrete Fourier Transform (IDFT)

To compute the inverse of the discrete Fourier transform (IDFT) of a vector u, simply multiply $\frac{1}{m}$ F* times u, where F* = Hermitian of F.

Continuing the example:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & +i \\ 1 & -1 & 1 & -1 \\ 1 & +i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Divide by m

F* times u = m*x

Fast Fourier Transform (FFT)

The **Fourier transform** of a vector (i.e., the DFT of a vector) of length 2m **can be computed quickly** by taking advantage of the following **relationship between F_m and F_{2m}:**

$$F_{2m} = \begin{pmatrix} I & D \\ I & -D \end{pmatrix} \begin{pmatrix} F_m & 0 \\ 0 & F_m \end{pmatrix} P,$$

where **D is a diagonal matrix** and **P is a 2m by 2m permutation matrix.**

Fast Fourier Transform (FFT): Requires two size m Fourier transforms plus two very simple matrix multiplications!

Key Equations

Let F = Fourier matrix, and let F_b denote the Kronecker product of F with I_n.

Then, the **block DFT is given by**:

$$\begin{bmatrix} \tilde{A}_1 \\ \tilde{A}_2 \\ \tilde{A}_3 \\ \vdots \\ \tilde{A}_m \end{bmatrix} = \begin{bmatrix} I_n & I_n & I_n & \cdots & I_n \\ I_n & I_n\omega_m^1 & I_n\omega_m^2 & \cdots & I_n\omega_m^{m-1} \\ I_n & I_n\omega_m^2 & I_n\omega_m^4 & \cdots & I_n\omega_m^{2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_n & I_n\omega_m^{(m-1)} & I_n\omega_m^{2(m-1)} & \cdots & I_n\omega_m^{(m-1)^2} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_m \end{bmatrix},$$

$\tilde{X}$ F_b X

To solve: $\operatorname{diag}\{(\tilde{A}_1), (\tilde{A}_2), \ldots, (\tilde{A}_m)\}\tilde{x} = \tilde{b}$

where $\tilde{x} = F_b^* x, \tilde{b} = F_b^* b.$

SERIAL ALGORITHM

Block Circulant Matrix: Storage

TABLE 1. THE PERCENTAGE OF THE INITIAL COEFFICIENT MATRIX WHICH NEEDS TO BE STORED.

m	% of A stored
2	50%
4	25%
8	12.5%
12	8.33%
16	6.25%

Block Circulant Matrix: Size of Linear Systems

TABLE 2. SIZE OF THE LINEAR SYSTEMS FOR VARYING m AND N.

N	$n,m = 4$	$n,m = 8$
13,000	3,250	1,625
15,000	3,750	1,875
19,000	4,750	2,375
24,000	6,000	3,000

Note: Solving a dense linear system is cubic in the size of the matrix.

Sequential Algorithm

Algorithm 1 Pseudocode for the sequential solution of a block circulant linear system.

1. Compute $\tilde{b} = F_b^* b$. **IDFT**
2. Compute $\tilde{X} = F_b X$. **DFT**
3. Solve $\tilde{A}_j \tilde{x}_j = \tilde{b}_j$, $j = 1, \ldots, m$. **Solution of m independent linear systems**
4. Compute $x = F_b \tilde{x}/\mathrm{m}$ **DFT**

PARALLEL ALGORITHM

How to Parallelize the Algorithm?

Ideas?

Recall, we are interested in solving

$$\mathrm{diag}\{(\tilde{A}_1),(\tilde{A}_2),\ldots,(\tilde{A}_m)\}\tilde{x}=\tilde{b}.$$

m independent linear systems to solve!

where

$$\tilde{x}=F_b^* x, \tilde{b}=F_b^* b.$$

Block DFT Algorithm

- A **block DFT calculation** is the basis for our **parallel algorithm**.
- This demonstrates **improved robustness** (over use of the FFT) and allows for **any boundary surface to be input**.

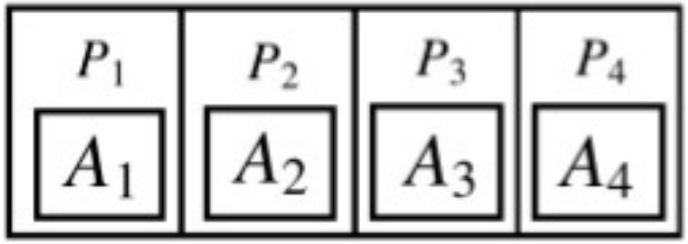


FIGURE 4. INITIAL DATA DISTRIBUTION ASSUMED IN THE DFT COMPUTATION FOR THE CASE $P = m = 4$.

DFT Computation

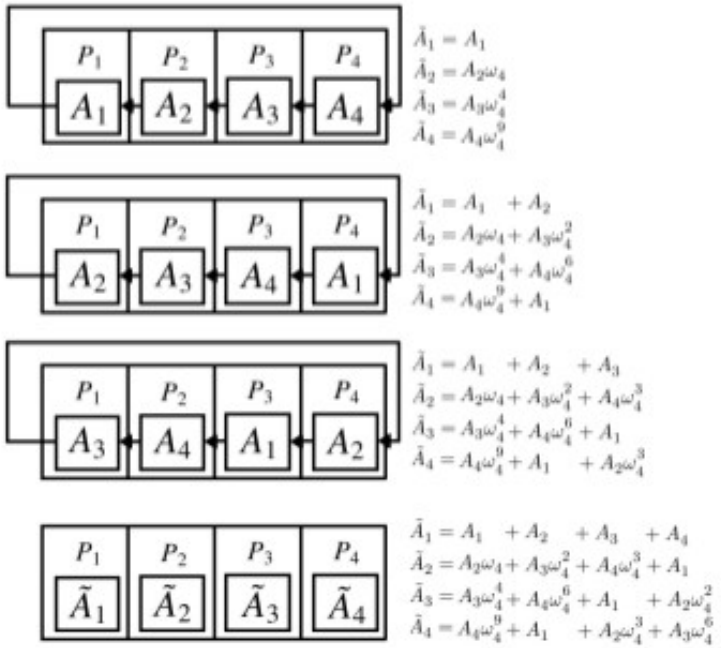


FIGURE 5. THE DFT COMPUTATION FOR THE CASE $P = m = 4$. EACH ARROW INDICATES THE COMMUNICATION OF A PROCESSOR'S OWNED SUBMATRIX TO A NEIGHBORING PROCESSOR IN THE DIRECTION OF THE ARROW.

This generalizes to the case when P > m.

Parallel Algorithm

Algorithm 2 Pseudocode for the parallel solution of a block circulant linear system, assuming $P = cm$.

1. Define m $\sqrt{c} \times \sqrt{c}$ process grids.
2. Block cyclically distribute each A_j and b_j onto grid G_j using identical blocking factors. **Asynchronous sends and receives**
3. Perform c simultaneous IDFTs transforming b_j to $\tilde{b}_j$.
4. Perform c simultaneous DFTs transforming A_j to $\tilde{A}_j$.
5. Simultaneously solve each $\tilde{A}_j \tilde{x}_j = \tilde{b}_j$ in parallel using PZGESV. **Solve using SCALAPACK. Complexity: Cubic in n**
6. Perform c simultaneous DFTs transforming $\tilde{x}_j$ to x_j.

The tradeoff of using overlapped communication and computation is additional memory.

NUMERICAL EXPERIMENTS

Computer Architecture for Experiments

Cyberstar compute cluster at Penn State:

- Run on two Intel Xeon X5550 quad-core processors
- HyperThreading = disabled
- **Total:** 8 physical cores running at 2.66 GHz
- 24 GB of RAM per node

Code:

- Fortran 90 with MPI
- ScaLAPACK library

Blocking and Communication:

- Blocking factor of 50 for block cyclic distribution of A_j and b_j onto their respective processor grids.
- DFT algorithm communications: blocks of size 4000
- Asynchronous sends/receives

Metrics for Experiments

- **Runtime** = wall clock time of parallel algorithm
  = $\mathbf{T_p}$

- **Speedup** = how much faster is the parallel algorithm than the serial algorithm = $\mathbf{S = T_s/T_p}$

- **Efficiency = E = S/P**

Experimental Results – 4 Processors

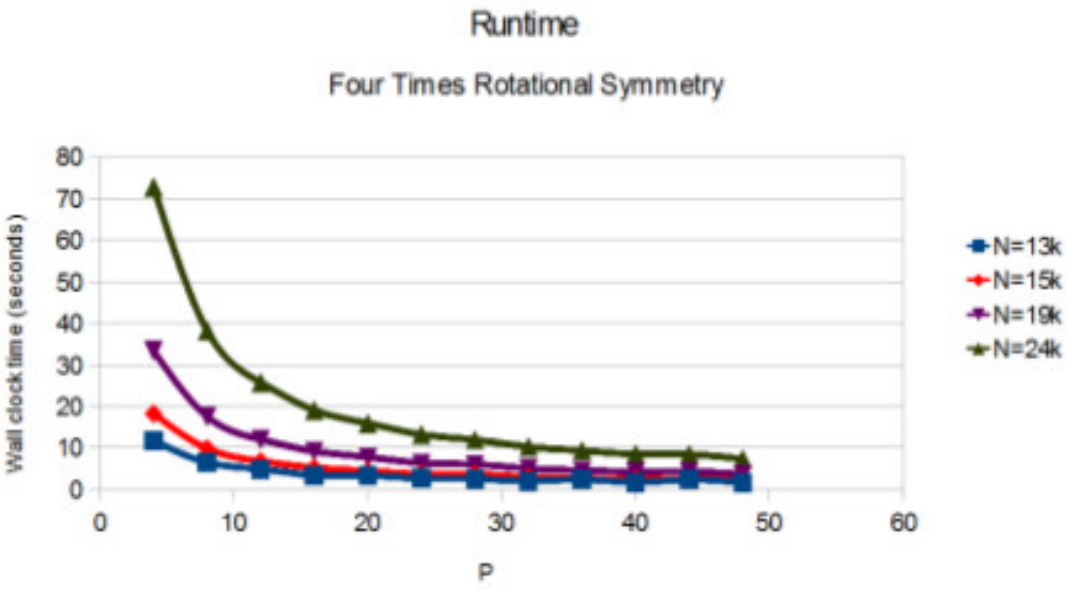


FIGURE 9. RUNTIME COMPARISON FOR VARYING P AND N WITH $m = 4$.

The runtime decreases as the number of processors increase and as the problem size decreases.

Experimental Results – 4 Processors

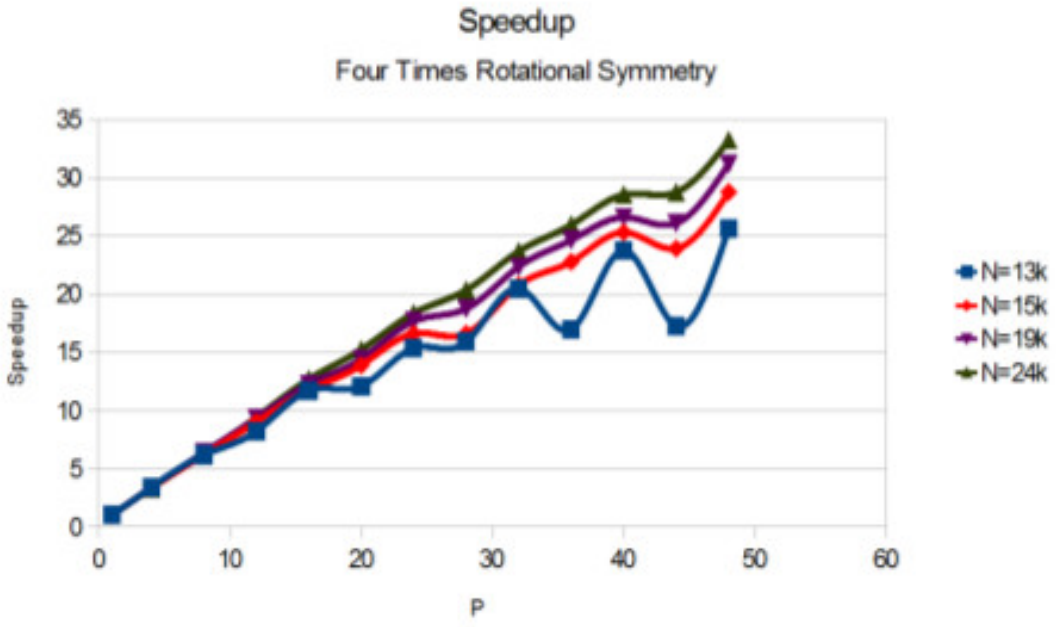


FIGURE 10. SPEEDUP COMPARISON FOR VARYING P AND N WITH $m = 4$.

Oscillations are due to small variance in small runtime numbers. They are smoothed out with increasing N.

Experimental Results – 4 Processors

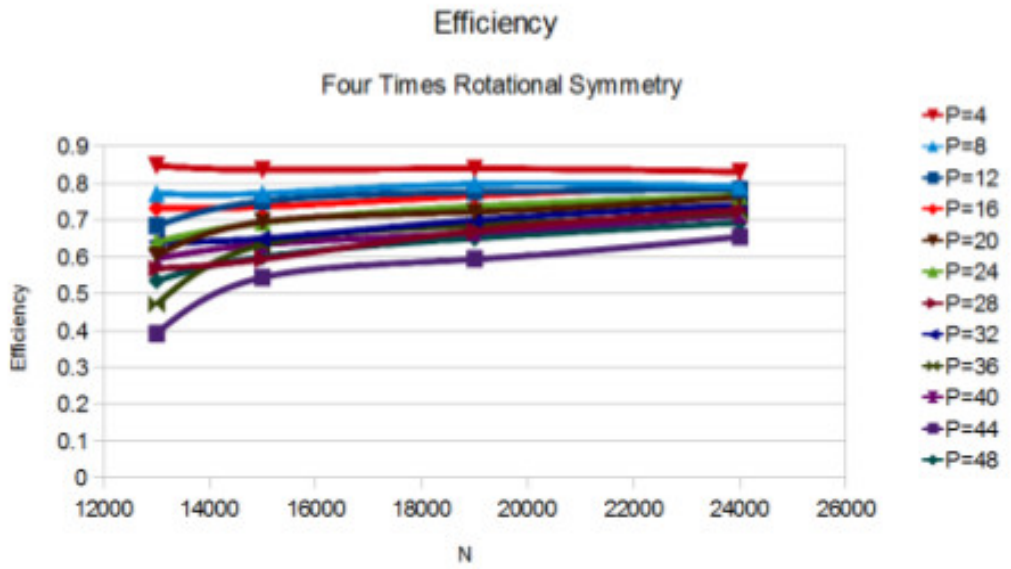


FIGURE 11. EFFICIENCY COMPARISON FOR VARYING N AND P WITH $m = 4$.

The efficiency increases for a decreased number of processors. It also increases with an increase in problem size.

Experimental Results – 8 Processors

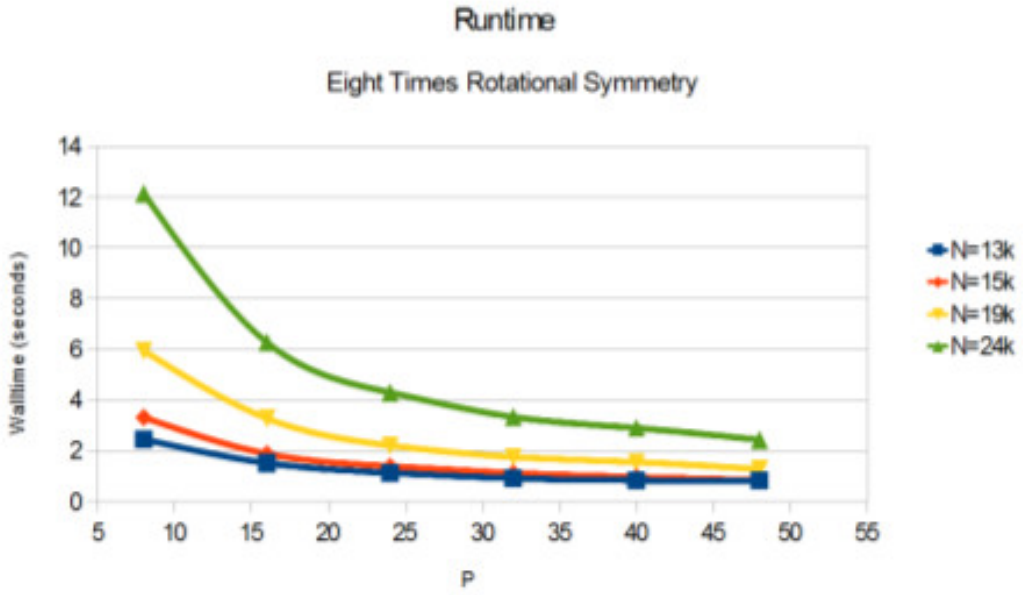


FIGURE 12. RUNTIME COMPARISON FOR VARYING P AND N WITH $m = 8$.

The runtime trend is the same as it is for m = 4.

Experimental Results – 8 Processors

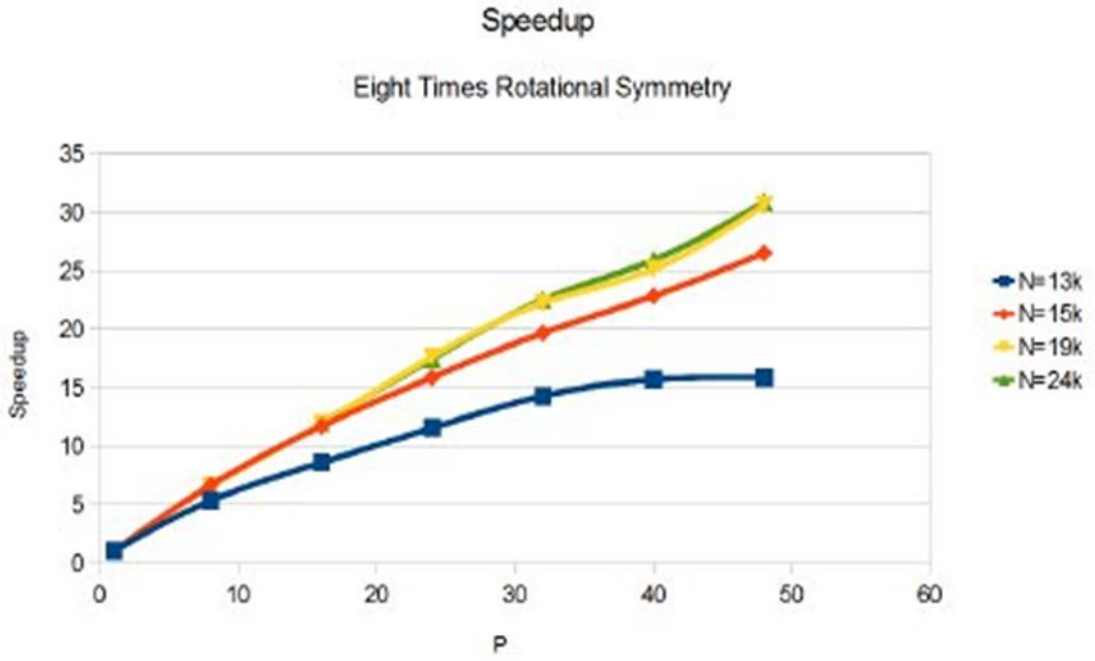


FIGURE 13. SPEEDUP COMPARISON FOR VARYING P AND N WITH $m = 8$.

For small problems, the speedup levels off due to the ratio of computation versus communication in the linear system solve. For larger problems, the speedup is nearly linear.

Experimental Results – 8 Processors

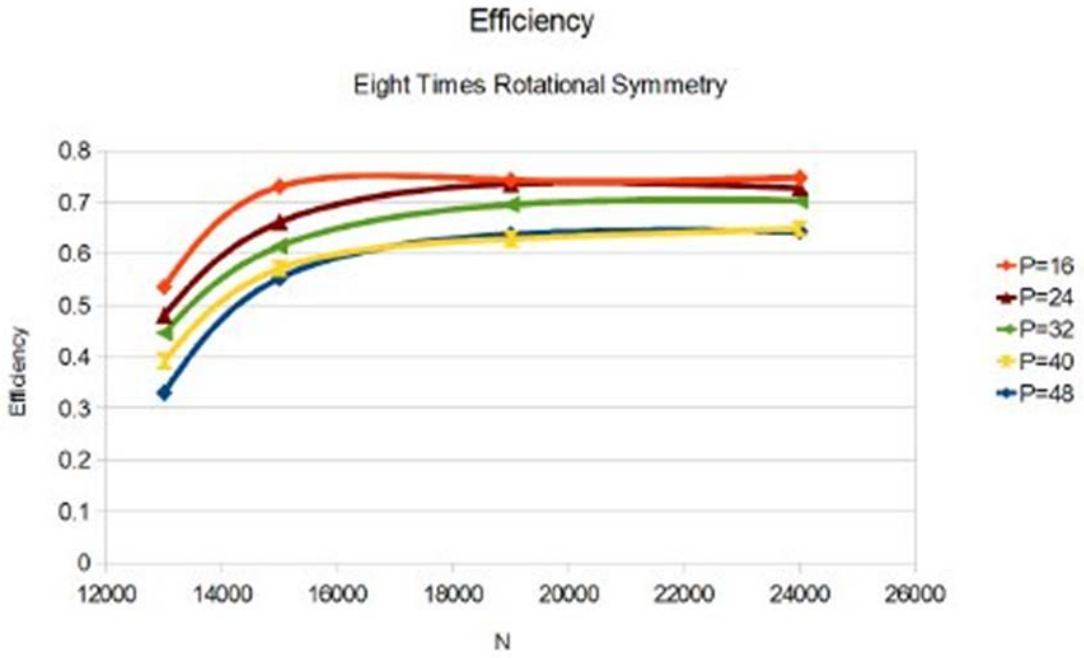


FIGURE 14. EFFICIENCY COMPARISON FOR VARYING P AND N WITH $m = 8$.

The efficiency is fairly good but not quite as high as it is for m = 4. Based on increase in communications due to DFT algorithm and size of linear system solve. Expect efficiency to remain high for increased problem size.

CONCLUSIONS

Conclusions

- We have proposed a **parallel algorithm** for **solution of block circulant linear systems**. Arise from **acoustic radiation problems** with **rotationally symmetric boundary surfaces**.
- Based on **block DFTs (more robust)** and have **embarrassingly parallel nature** based on ScaLAPACK's required data distributions.
- **Reduced memory requirement** by **exploiting block circulant structure.**
- Achieved near linear speedup for varying problem size, **linear speedup for large N. Efficiency increases with problem size.**
- **Can solve larger/higher frequency** acoustic radiation **problems.**

Reference

K.D. Czuprynski, J. Fahnline, and S.M. Shontz, *Parallel boundary element solutions of block circulant linear systems for acoustic radiation problems with rotationally symmetric boundary surfaces*, Proc. of the Internoise 2012/ASME NCAD Meeting, August 2012.

Acknowledgements

This work is based on the **M.S. Thesis of Ken Czuprynski** in addition to our Internoise 2012 paper.

Computing Infrastructure:

- NSF grant OCI-0821527

Research Funding:

- Penn State Applied research Laboratory's Walker Assistantship Program (Czuprynski)
- NSF Grant CNS-0720749 (Shontz)
- NSF CAREER Award OCI-1054459 (Shontz)