A Combinatorial Approach to Preconditioners for Iterative System Solvers

Gary L. Miller

School of Computer Science
Carnegie Mellon University

Joint work with:
Keith Gremban

The Problem

Consider solving

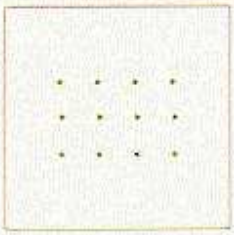
$$Ax = b$$

for A sparse, symmetric, diagonally dominant.

nonpositive off diagonal

Example:

finite difference formulation of elliptic PDE



$$\nabla^2 u = f ; u|_{\partial D} = 0$$

yields, for 4 points $A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}$

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Standard Methods

Goal: Solve Sparse Linear System $Ax=b$

Standard Methods

Direct

Iterative

Multigrid

Domain Decomposition

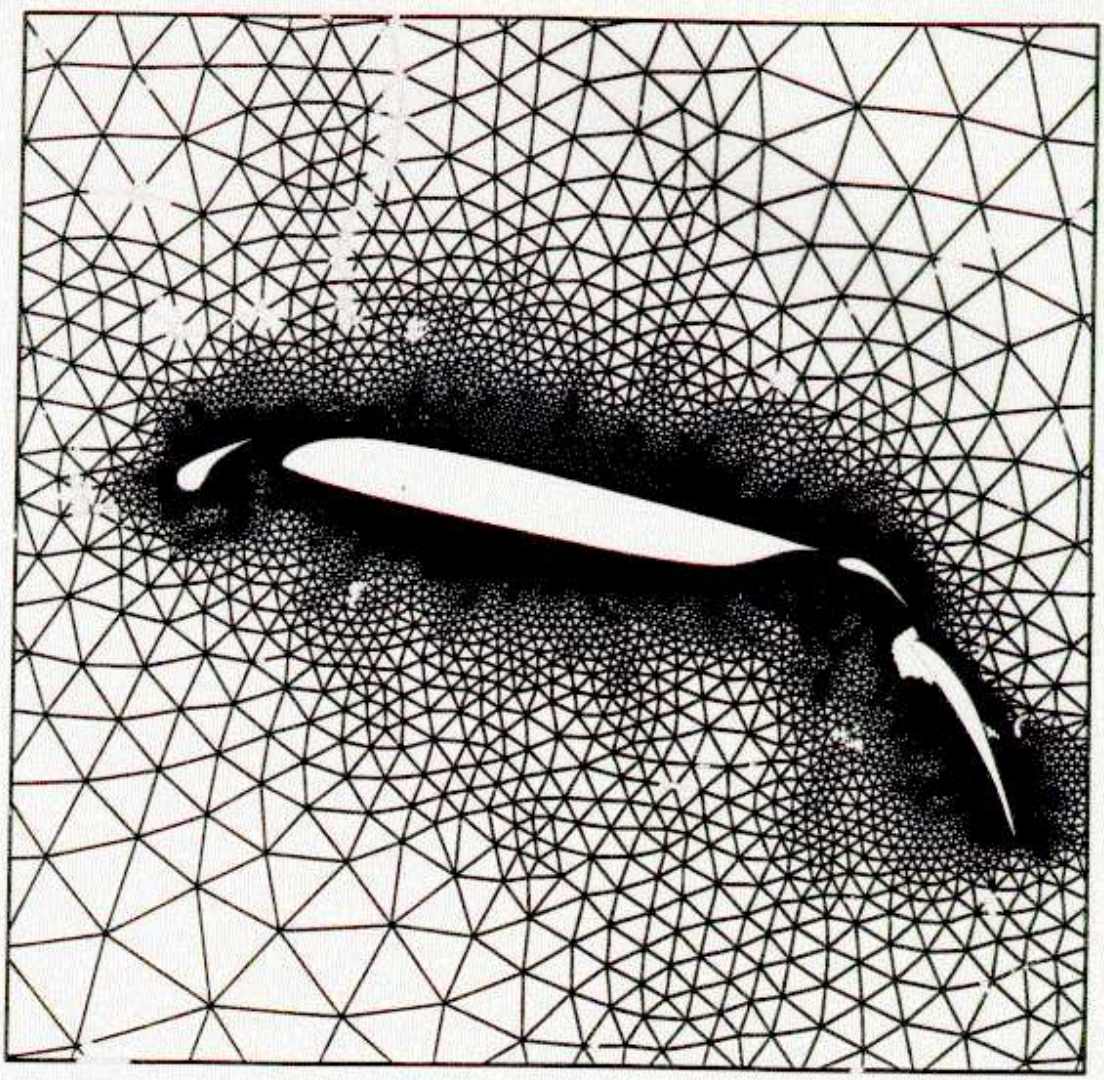
Direct Methods

Nested Dissection

$O(n^{3\alpha})$ work where n^{α} is separator size of graph.

Planar Case (2-dim) Separator size $n^{1/2}$
Work $n^{1.5}$

(3-dim) Separator Size $n^{2/3}$
Work n^2



Laplacian of a Graph

$$
\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \end{array}
\begin{array}{c}
\begin{array}{ccc} V_1 & V_2 & V_3 \end{array} \\
\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}
\end{array} = M
$$

Note: M is singular

The Basic Iterative Method

Input Linear System: $A \cdot x = b$

Basic Iterative Method or Richardson's Method is:
Start with some guess x^0 and iteratively compute

$$x^{(i+1)} = (I - A) \cdot x^{(i)} + b$$

- A fixed point is a solution to our linear system.
- Richardson's method may not converge
- If it does converge it may do so very slowly

An $n \times n$ invertible matrix B is **preconditioner** which gives us a new preconditioned system::

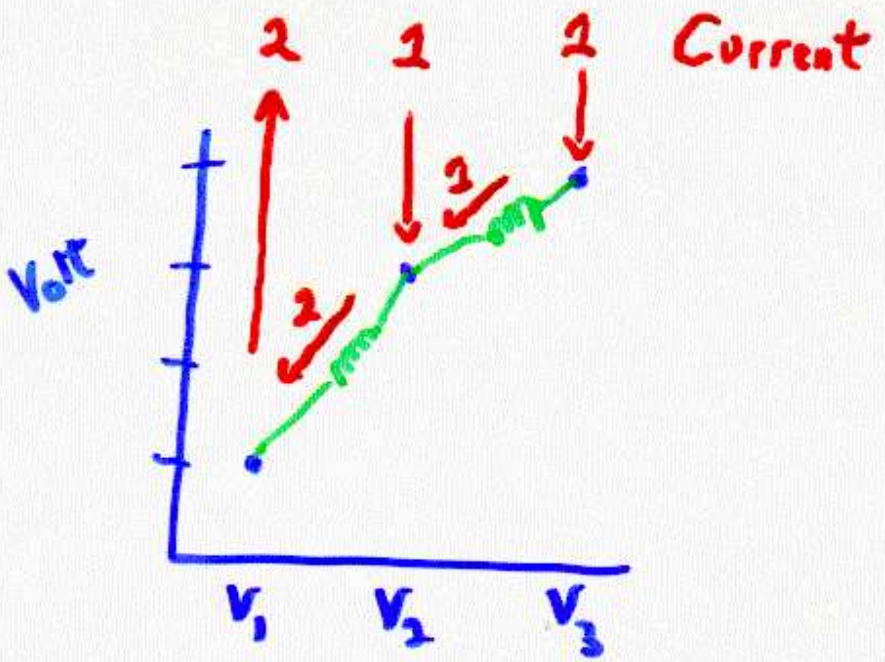
$$B^{-1}A \cdot x = B^{-1}b = b'$$

Note: A x is solution to the original system
$\iff$ it is one for the new.

8

Voltage / Current Model

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1-3 \\ (3-1)+(3-4) \\ 4-3 \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix}$$



The Two Steps in a Preconditioned Iterative Method

Combining the preconditioner with Richardson method we get:

$$x^{(i+1)} = (I - B^{-1}A) \cdot x^{(i)} + b'$$

We view each iteration as two major steps:

1. $y = Ax^{(i)}$

2. $z = B^{-1}y$ i.e. solve $Bz = y$

Finally: $x^{(i+1)} = x^{(i)} - z + b'$

The goal is to pick a B so that the cost of solving $Bz = y$ every iteration is out weighted by the decrease in the number of iterations.

Famous Iterative Methods

If we take Richardson's Method,
$x^{(i+1)} = (I - A) \cdot x^{(i)} + b$, and precondition it using some natural choices for B we get famous methods:

$$B = \begin{cases} D & \text{Jacobi's Method} \\ L + D & \text{Gauss-Seidel's Method} \\ (L + \frac{1}{\omega}D)\frac{1}{\omega}D(L^T + \frac{1}{\omega}D) & \text{SSOR}^1 \end{cases}$$

where

D = Diagonal of A
L = Lower triangular part of A, zero diagonal

[1]SSOR = symmetric successive over relaxation

Gauss - Seidel

set $\hat{L} = L + D$

1) $y = Ax^{(i)}$ (sparse-matrix vector product)

2) $\hat{L}z = y$ back - substitution

3) $x^{(i+1)} = x^{(i)} + z + b'$

* Step 2) not parallel

* **Note:**
Steps 1) & 2) $\equiv$ back-substition

Main Goal of Preconditioner

Properties B should have:

1. Solving $Bz = y$ should be easy even in **Parallel**.
2. $B^{-1}A \approx I$ i.e. $B \approx A^{-1}$.

NOTE: Gauss-Seidel and SSOR require the solution to lower triangular systems each iteration.

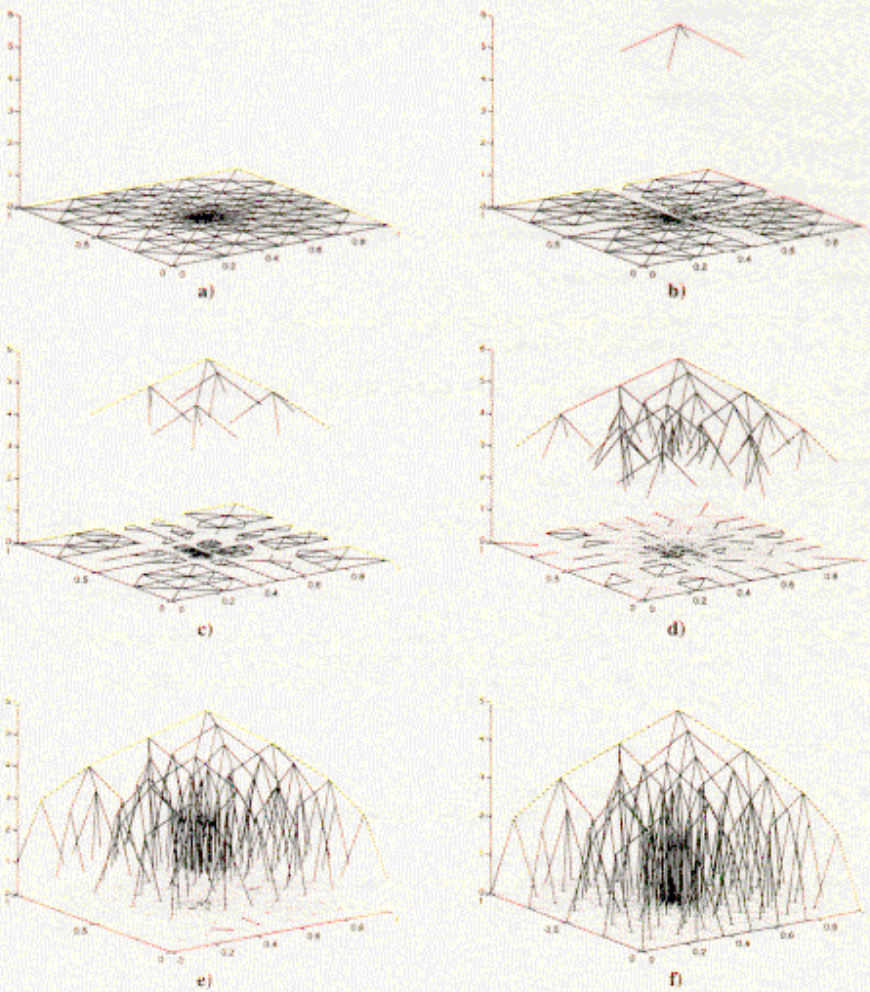
- **Sequentially** solving lower triangular systems requires no more time than matrix-vector product.
- **Parallel** algorithms for solving lower triangular system is not know to be work efficient.

Our Idea for a Preconditioner

IDEA: External Tree Preconditioner

- View the matrix as a graph.
- Make a Steiner like tree for a preconditioner.
- Use separators to guide the construction.

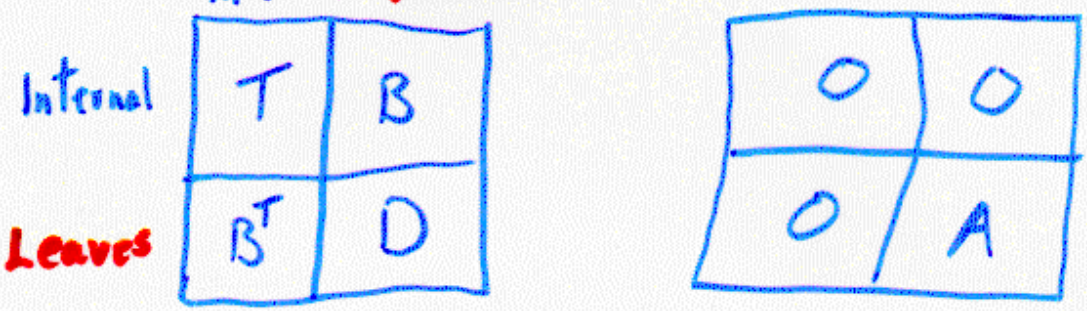
Support Tree Example



School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Support Tree + Basic Iterative

The Matrices



The Tree

A Padded

$$\begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \end{pmatrix} = \left[I - \begin{pmatrix} T & B \\ B^T & D \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 \\ 0 & A \end{pmatrix} \right] \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix} + \begin{pmatrix} 0 \\ b' \end{pmatrix}$$

Support Tree + Basic

Algorithm

1) $A x_2^{(i)} = y_2$

2) $\begin{pmatrix} T & B \\ B^T & D \end{pmatrix} \begin{pmatrix} \partial_1 \\ \partial_2 \end{pmatrix} = \begin{pmatrix} 0 \\ y_2 \end{pmatrix}$ (new)

3) $x_1^{(i+1)} = 0$

$x_2^{(i+1)} = x_2^{(i)} - \omega \partial_2 + b'$

Our Idea for a Preconditioner

IDEA: External Tree Preconditioner

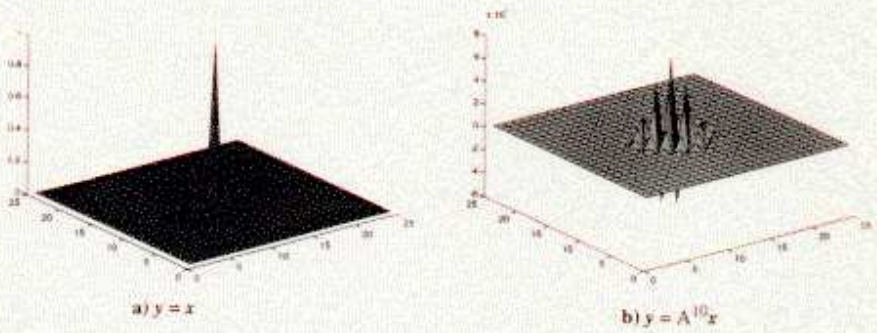
- View the matrix as a graph.
- Make a Steiner like tree for a preconditioner.
- Use separators to guide the construction.

We use the boundary size for edge wieghts.

Support Tree Intuition (1)

Iteration is like mixing:

- **data propagates across mesh**
- **convergence when mixing is complete**



a) $y = x$

b) $y = A^{10}x$

Preconditioners approximate the original system:

B = A is the perfect preconditioner!

We want: rapid mixing
good approximation
good parallel properties

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The Error

$$u^{(i+1)} = G u^{(i)} + b \qquad G = I - A$$

$$e^{(i)} = u^{(i)} - \bar{u} \qquad \text{Error at } i$$

where $\bar{u}$ $= G\bar{u} + b$

Claim: $e^{(i+1)} = G e^{(i)}$

$$e^{(i)} = G^{*i}\, e^{(0)}$$

Condition Number

Def $Ax = \lambda x$ then λ Eigenvalue
x Eigenvector

$\lambda(A) = \{0 \le \lambda_1 \le \dots \lambda_n\}$ Our A's

Def $K(A) = \lambda_n / \lambda_1$ Condition #

IfG

Graph G d-D mesh on n-nod

$$K(A) = \Theta(d \cdot n^{2/d})$$

$n^{2/d} \approx$ square of diameter

Convergence Rate

Basic Method

Convergence Rate = $O(1/\kappa(A))$

Conjugate Gradient

Convergence Rate = $O(1/\sqrt{\kappa(A)})$

$\approx 1/\text{diameter}$

Support Tree Analysis

Goal: Bound the condition number of $B^{-1}A$

$$B^{-1}Ax = \lambda x \iff Ax = \lambda Bx$$

Def λ is a generalized eigenvalue of (A,B)

$$\lambda(A,B) = \{\lambda_1, \ldots, \lambda_n\}$$

Def A is positive semi-definite if $\forall x \quad x^T Ax \geq 0$

Estimating Support

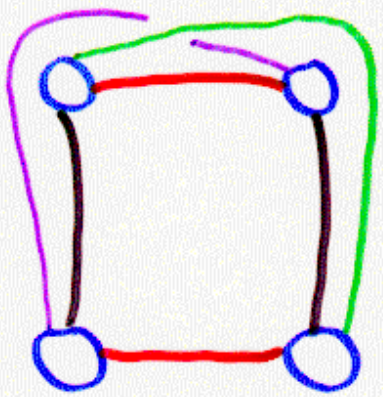
G, H are graphs s.t. $V(G) = V(H)$

ϕ: $E(G) \to$ paths of H

s.t. $f(e_{ij}) = v_i \cdots v_j$

is called a <u>Path Embedding</u>

$G = K_4$ $H =$ 4-cycle



3 = Congestion

2 = Dilation

6

Uses of Graph Embeddings

Gross — Graph Genus

Leighton — VLSI

Jerrum & Sinclair — Mixing

Dyer Frieze Kannan — Volume

Guattery Miller — Spectral Partition

Gremban Miller — Iterative Sys. Sol.

M Talmor Teng Walkington — Control-Volume Method

Support Tree Analysis

lemma (support lemma):

For A, B Laplacian matrices, if $\lambda \in \underline{\lambda}(A, B)$ and $\tau B - A$ is positive semi-definite, then $\lambda \leq \tau$.

definition (support):

The support of matrix B for matrix A is defined by

$\sigma(A,B) = min\{\tau: \tau B\text{-}A \text{ is positive semi-definite}\}$

$\sigma(A,B)$ is the *support number* of B for A.

Note:

$$\lambda_{max}(A,B) \leq \sigma(A, B)$$

$$\kappa(A, B) \leq \sigma(A, B) \cdot \sigma(B, A)$$

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Extended Analysis

Define:

congestion: $\gamma(A,B)$ = the maximum number of edges of A supported by any edge of B.

dilation: $\delta(A,B)$ = the maximum length of any path in B supporting an edge in A.

gain: $g(A,B)$ = the ratio of the conductance of an edge in A to the minimum conductance of an edge of B supporting A

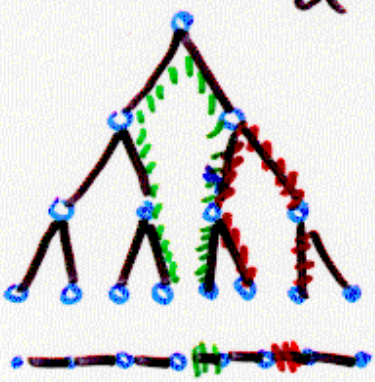
Then,

$$\sigma(A, B) \leq \gamma(A, B) \cdot \delta(A, B) \cdot g(A, B)$$

γ, δ, g can be bounded without explicit mapping!

Supporting a Mesh by a Tree.

$d = 1$



Congestion = 2

Dilation = $2 \log n$

$\tau(M, T) \leq 4 \log n$

Supporting a Tree by a Mesh

Not well defined!

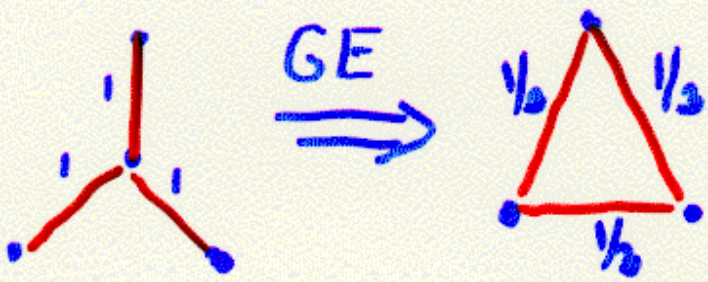
$V(M) \not\subseteq V(T)$

Trick Remove extra variables in T by Gaussian Elimination.

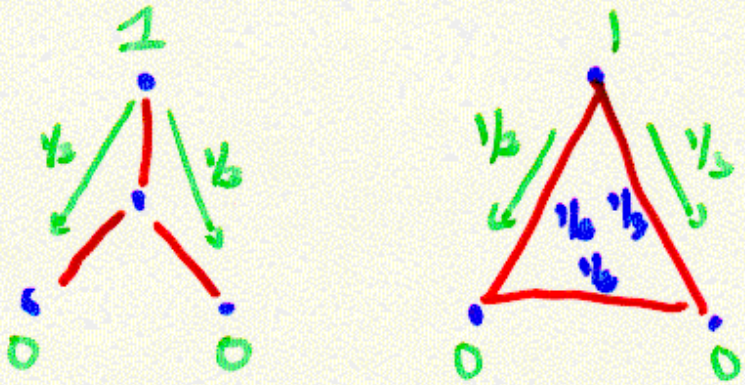
Goal: $\tau(\hat{T}, M)$

Note: $\hat{T}$ is a dense matrix

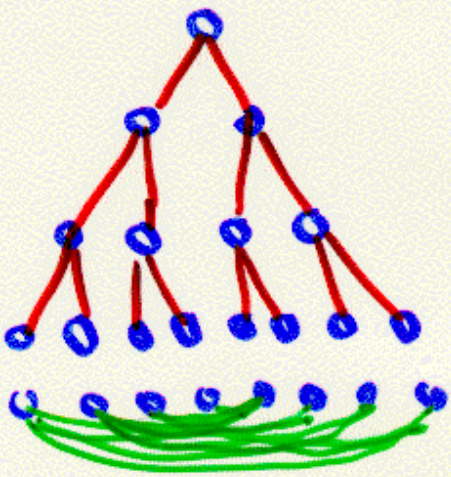
Gaussian Elimination
Electrical Conductance



Test



Electrical Conductance



Claim: All green edge weights are equal

edges $= (n/2)^2$

weight $= 1/(n/2)^2 \simeq n^{-2}$

$C \equiv n^2$ $D \equiv n$ Gain $\equiv n^{-2}$

$\nabla(T, M) = O(n)$

Support Tree for a Mesh

Let T be a support tree for a d-dim mesh M.

Thm $K(T,M) = O(n^{1/d} \log n)$

1) $\tau(M,T) = O(\log n)$

2) $\tau(T,M) = O(n^{1/d})$

STCG Performance (2)

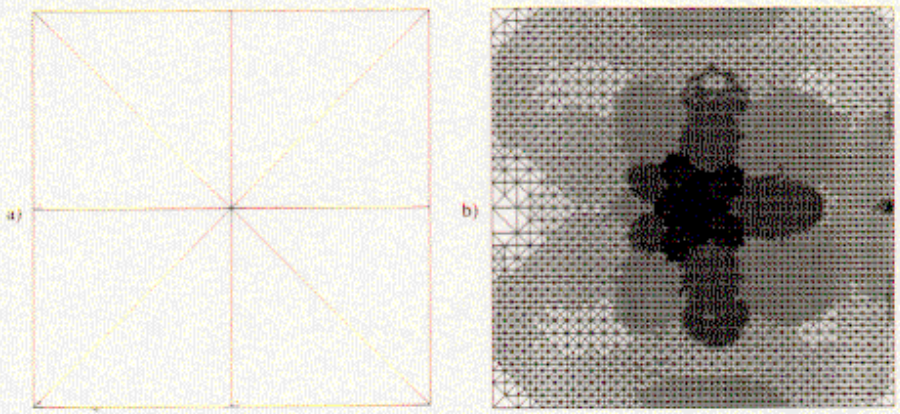


Figure 6.3: ***Crack Meshes.***
a) crack00 with 10 nodes.
b) crack10 with 10240 nodes.

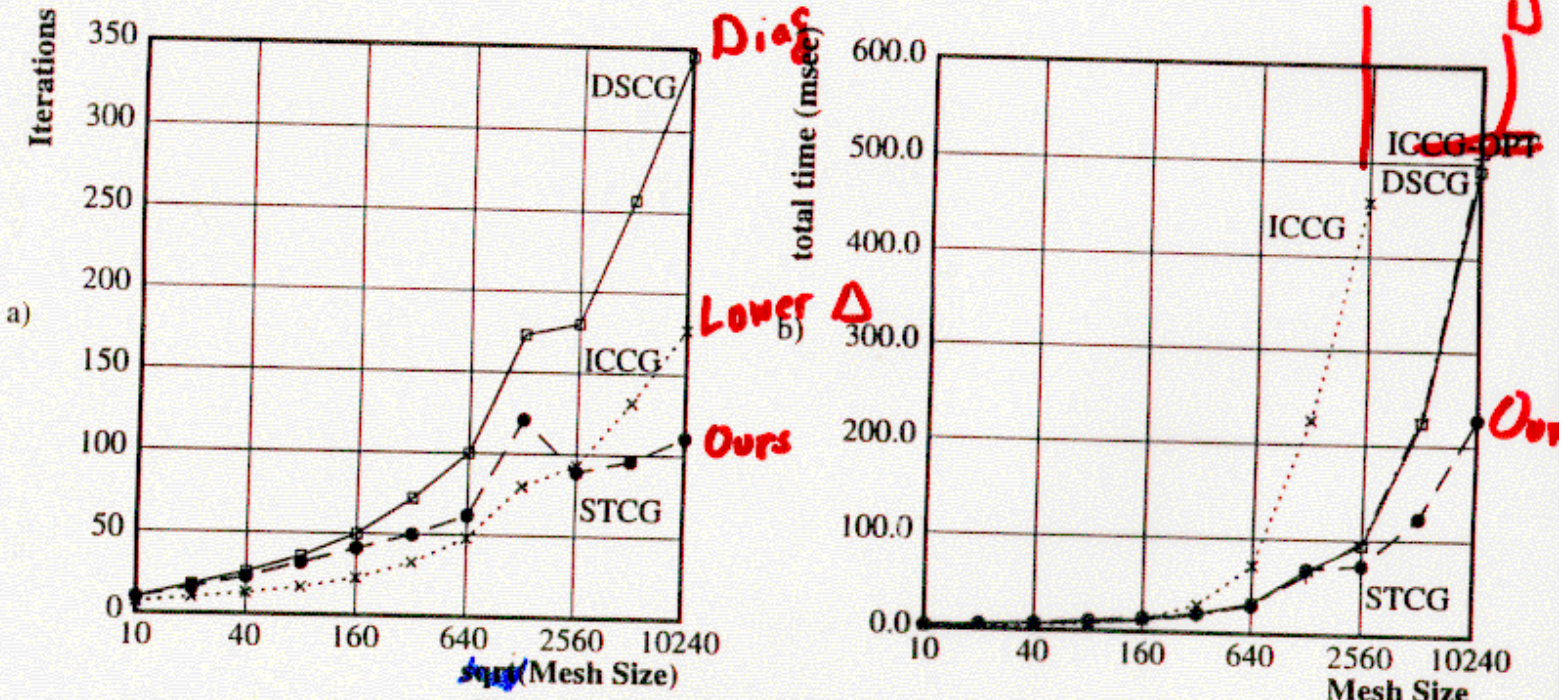


Figure 11: Results on 2D Irregular Mesh

a) Iterations to converge b) Total execution time for iterative process on Cray C-90 (msecs)

STCG Performance (1)

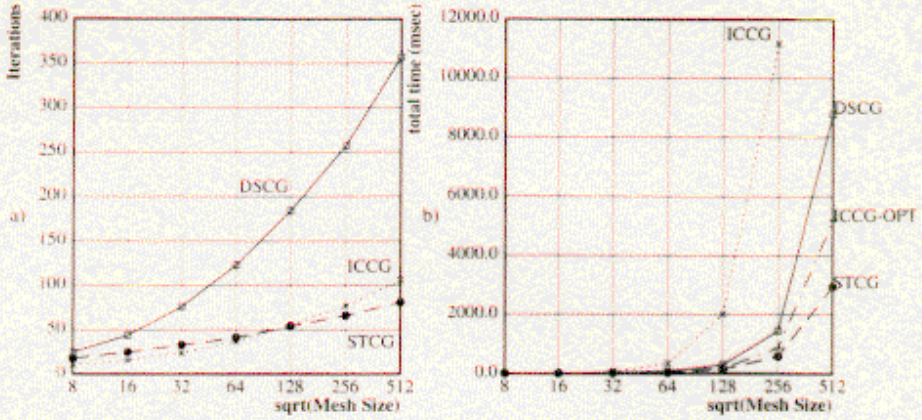


Figure 6.1: ***Results for 2D Regular Meshes, Random Input.***
a) Iterations to convergence.
b) Total time for iterative process on Cray C-90 (msecs).

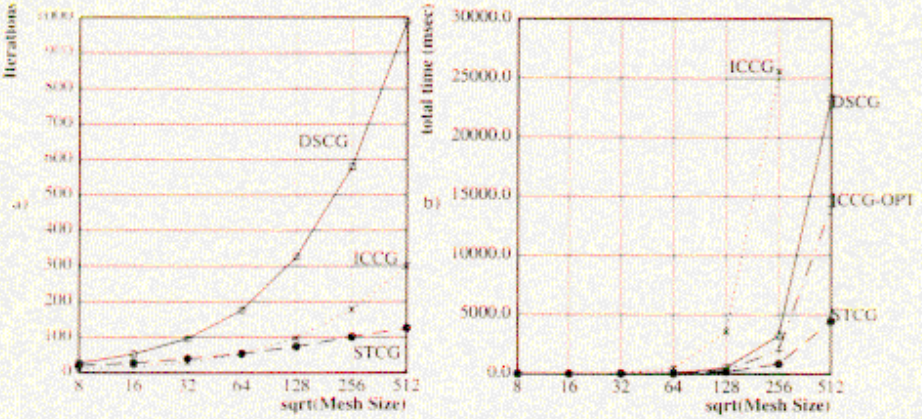


Figure 6.2: ***Results for 2D Regular Meshes, Impulse Function Input.***
a) Iterations to convergence.
b) Total time for iterative process on Cray C-90 (msecs).

STCG Performance (2)

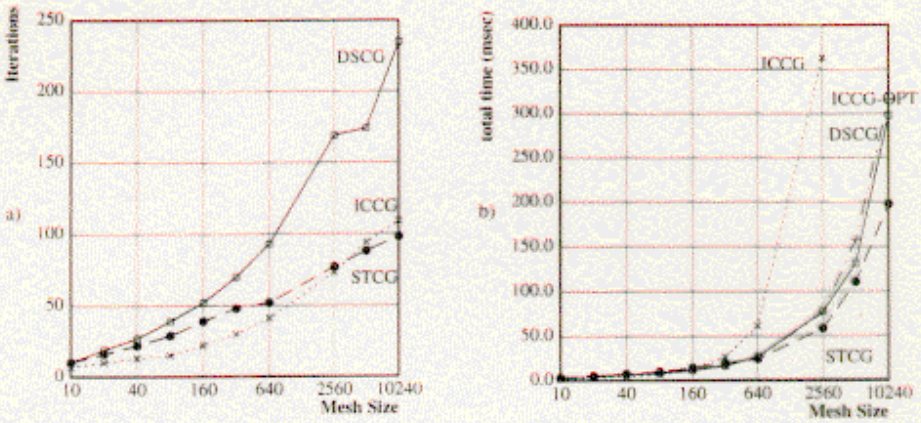


Figure 6.4: ***Results on 2D Irregular Meshes, Random Input.***
a) Iterations to convergence.
b) Total execution time for iterative process on Cray C-90 (msecs).

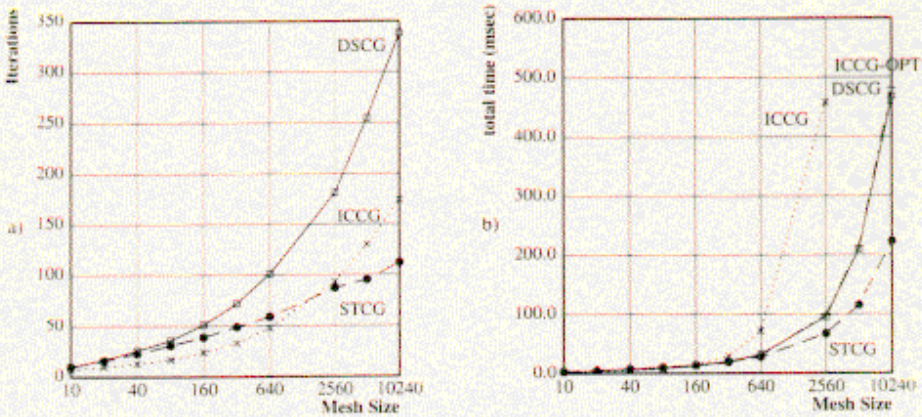


Figure 6.5: ***Results on 2D Irregular Meshes, Impulse Function Input.***
a) Iterations to convergence.
b) Total execution time for iterative process on Cray C-90 (msecs).

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

STCG Performance (3)

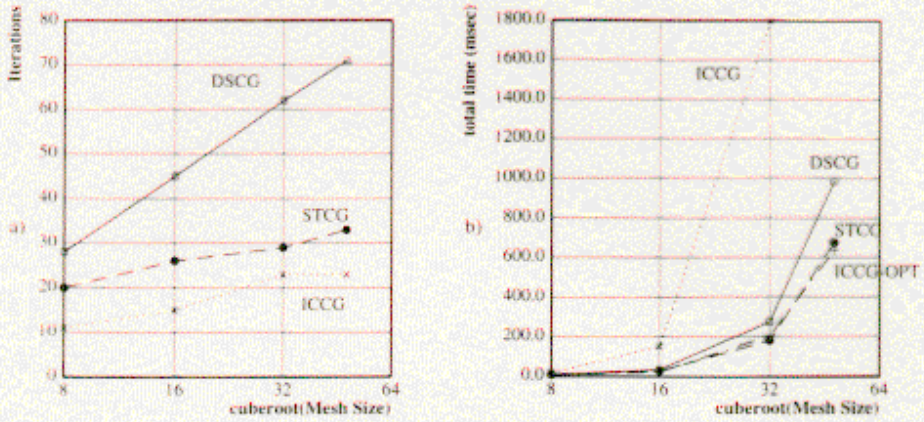


Figure 6.6: ***Results on 3D nxnxn Meshes, Random Input.***
a) Iterations to convergence.
b) Total execution time for iterative process on Cray C-90 (msecs).

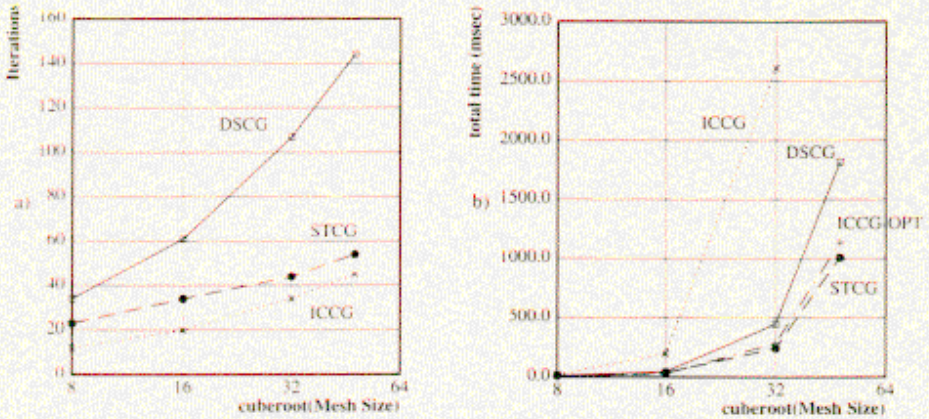


Figure 6.7: ***Results on 3D nxnxn Meshes, Impulse Function Input.***
a) Iterations to convergence.
b) Total execution time for iterative process on Cray C-90 (msecs).

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213