CPS615—Base Course for the Simulation Track of Computational Science

Fall Semester 1995
Finite Element Methods
and Conjugate Gradient Methods

Geoffrey Fox
NPAC
Room 3-131 CST
111 College Place
Syracuse, New York 13244-4100

CPS615—Abstract

Finite Element/Conjugate Gradient Presentation

- This derives the finite element method for a simple two-dimensional Laplacian with triangular elements
- We use this to motivate conjugate gradient as a variant of steepest descent for variational principle underlying FEM
- We discuss preconditioning, parallelism and convergence of general conjugate gradient method

Integral Formulation of Finite Element Method

- Consider
$$F = \frac{1}{2} \int_\Omega (\nabla\varphi)^2 \, d^3\underline{x}$$
where φ is defined over a region Ω in d dimensional space — $d = 2$ in text as one can draw.
- φ is fixed $= f$ on boundary $\partial\Omega$ of Ω (Figure FEM 1)

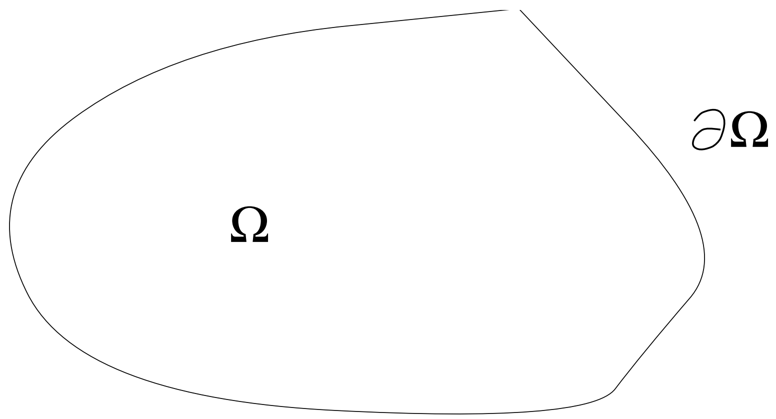


Figure: FEM 1

Variation in Integral

- Consider variation

  Let $\varphi \to \varphi + \eta$ with $\eta = 0$ on $\partial\Omega$ and η a function of $\underline{x}$ over d dimensional Ω

- Then this change in φ induces a change $F \to F + \delta F$ where, to first order in η,

  $$\delta F \approx \int_\Omega (\nabla\varphi \cdot \nabla\eta)\, d^3\underline{x}$$

- We have Gauss's theorem

  $$\int_\Omega \nabla \underline{u} d^3 x = \int_{\partial\Omega} \underline{u} \cdot d\underline{s}$$

  where $d\underline{s}$ is differential on surface $\partial\Omega$

- apply to $\underline{u} = \eta\nabla\varphi$

  $$\nabla\underline{u} = \nabla\eta \cdot \nabla\varphi + \eta\nabla^2\varphi$$

  where $\eta = 0$ on $\partial\Omega$

- Then

  $$\int_\Omega \nabla\varphi\, \nabla\eta d^3\underline{x} = -\int \eta\nabla^2\varphi d^3\underline{x}$$

  relates δF to $\nabla^2\varphi$ where

  $$\nabla\varphi = \left(\frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}, \frac{\partial\varphi}{\partial z}\right)$$

  and

  $$(\nabla\varphi)^2 = \left(\frac{\partial\varphi}{\partial x}\right)^2 + \left(\frac{\partial\varphi}{\partial y}\right)^2 + \left(\frac{\partial\varphi}{\partial z}\right)^2$$

Equivalence of Integral and Differential Formulation of Laplace's Equation

- Therefore, if F is stationary, i.e.

$$\frac{\partial F}{\partial\left(\eta(\underline{x})\right)} = 0$$

for all $\eta(\underline{x})$ consistent with boundary conditions, then $\nabla^2\varphi = 0$

- Thus, Laplace's equation is entirely equivalent to VARIATIONAL PRINCIPLE

$$\begin{aligned} \delta F &= 0 \\ F &= \int_\Omega \frac{1}{2}\left(\nabla\varphi\right)^2 d^3\underline{x} \\ \varphi(\underline{x}) &= \text{known } f(x) \text{ on } \partial\Omega \end{aligned}$$

We have converted differential equation to an integral . . .

Discretization of Integral

- Convert this to a minimization principle over a finite dimensional space defining φ
- Consider those functions φ which are piecewise linear on a set of triangles (the mesh) which covers Ω (Figure FEM 2)

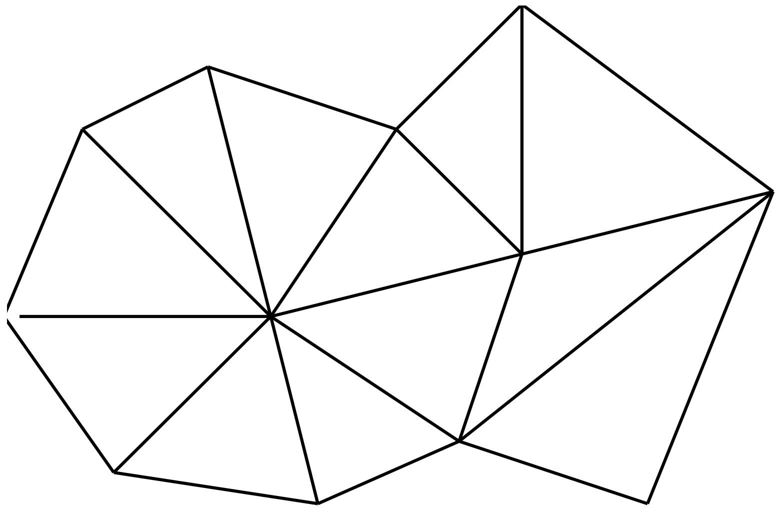


Figure: FEM 2

- i.e., we will use the generalized trapezoidal rule to integrate F

Triangular Elements in Two Dimensions

- First consider evaluating F on a triangle with a linear interpolation function (Figure FEM 3)

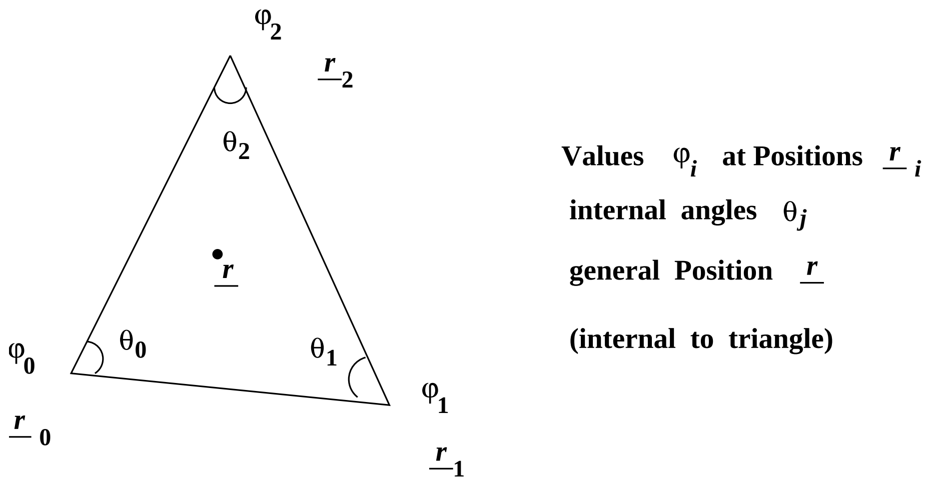


Figure: FEM 3

- so that

$$\varphi(\underline{r}) \;=\; \varphi_0 + (\varphi_1 - \varphi_0)\,\frac{(\underline{r} - \underline{r}_0) \wedge (\underline{r}_2 - \underline{r}_0)\,|_z}{(\underline{r}_1 - \underline{r}_0) \wedge (\underline{r}_2 - \underline{r}_0)\,|_z}$$

$$+ (\varphi_2 - \varphi_0)\,\frac{(\underline{r} - \underline{r}_0) \wedge (\underline{r}_1 - \underline{r}_0)\,|_z}{(\underline{r}_2 - \underline{r}_0) \wedge (\underline{r}_1 - \underline{r}_0)\,|_z}$$

- It can be seen that φ is linear in $\underline{r}$ and has value φ_0 at $\underline{r}_0$, φ_1 at $\underline{r}_1$, and φ_2 at $\underline{r}_2$

Example for Two-Dimensional Triangular Elements

- Show that integrating over one triangle gives

$$
\begin{aligned}
F &= \int_\Delta \frac{1}{2} (\nabla \varphi)^2 \, d^2x \\
&= \frac{1}{4} [\varphi_0 \, \varphi_1 \, \varphi_2] \begin{bmatrix} c_1 + c_2 & -c_2 & -c_1 \\ -c_2 & c_2 + c_0 & -c_0 \\ -c_1 & -c_0 & c_0 + c_1 \end{bmatrix} \begin{bmatrix} \varphi_0 \\ \varphi_1 \\ \varphi_2 \end{bmatrix}
\end{aligned}
$$

where $c_j = \cot \theta_j$

Bilinear Form of Integral with Triangular Elements

- Now consider a triangulation of Ω (Figure FEM 4)

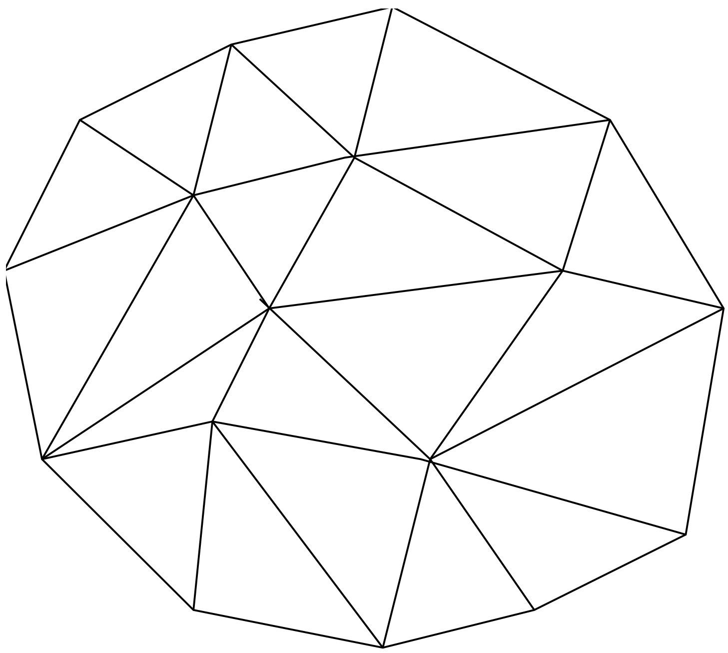


Figure: FEM 4

- We can write
$$F = \frac{1}{4}\underline{\varphi}^T S \underline{\varphi}$$
where $\underline{\varphi}$ is the vector of vertex values of φ and $\boldsymbol{S}$ is called a stiffness matrix

Formula for Stiffness Matrix Element—I

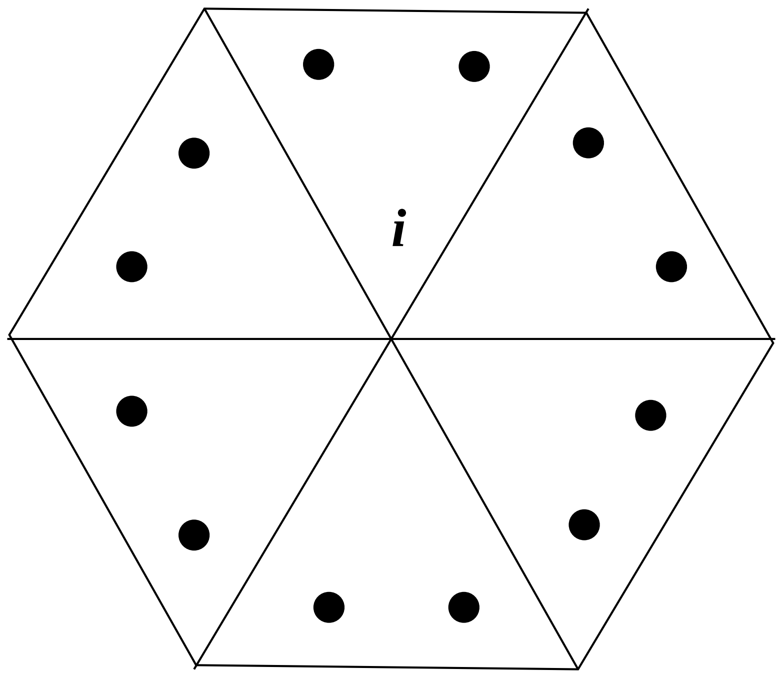


Figure: FEM 5

- $S_i i = \Sigma_{\text{angles}\,\theta} \cot\,\theta$ where angles θ are opposite nodal point i in a triangle with a vertex at i.

Formula for Stiffness Matrix

Element—II

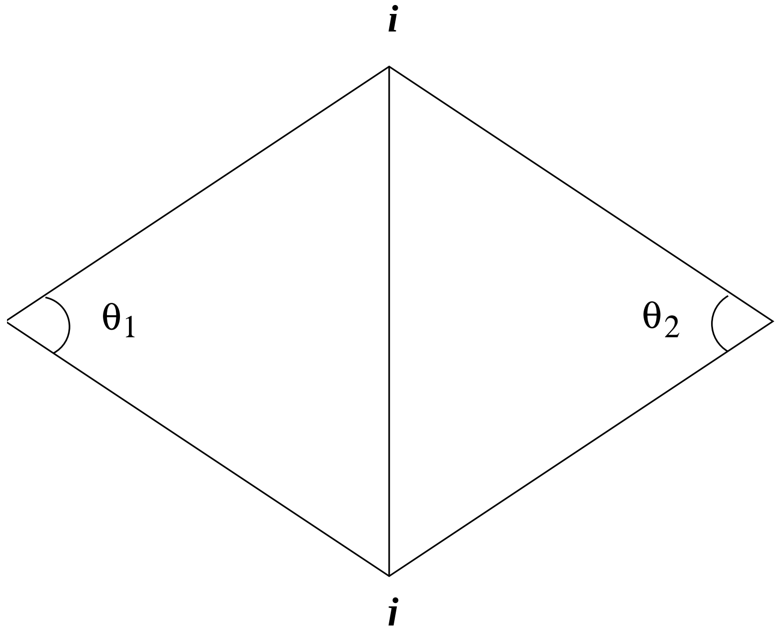


Figure: FEM 6

- If i and j are not connected in the triangulation, then $S_{ij} = 0$. If they are connected, written $i \leftrightarrow j$ and $i \neq j$ then $S_{ij} = -\left(\cot\theta_1 + \cot\theta_2\right)$ (Figure FEM 5)
- Example: Show that

$$\sum_i S_{ii} + \sum_{i \neq j} S_{ij} = 0$$

Hint: $\varphi(\underline{x}) = 1$ is a solution to Laplace's equation for which approximations are exact

Finite Element Equations

- We now have

$$\begin{aligned} F &= \frac{1}{4}\varphi^T S \varphi \\ &= \frac{1}{4}\sum_{ij} \varphi_j S_{ji} \varphi_i \end{aligned}$$

- Let

$$\begin{aligned} \varphi_i &\rightarrow \varphi_i + \eta_i \quad \text{be variation} \\ F &= \frac{1}{4}\sum_{ij} \varphi_j S_{ji} \varphi_i \\ &+ \frac{1}{4}\left(\sum_{ij} (\eta_j S_{ji} \varphi_i) + (\varphi_j S_{ji} \eta_i)\right) \end{aligned}$$

- Thus $\delta F = \frac{1}{2} \Sigma_{ij}\, \eta_j\, S_{ji}\, \varphi_i$
- This δF form is zero if and only if

$$\sum_i S_{ji}\, \varphi_i = 0$$

- This condition can be split into internal and boundary terms

$$\underbrace{\Sigma_{(j\ \text{not}\in\ \partial\Omega)} S_{ij} \varphi_j}_{\text{Unknown}} + \underbrace{\Sigma_{(j\in\partial\Omega)} S_{ij}\,(\varphi = f_j)}_{\text{Known at Boundary}} = 0$$

Structure of Stiffness Matrix and Its Assembly

- N has contributions from five triangles in picture (Figure FEM 7)
- This is called "assembly of the stiffness matrix S"
- Note by using variational formalism, we have avoided need for regular grid

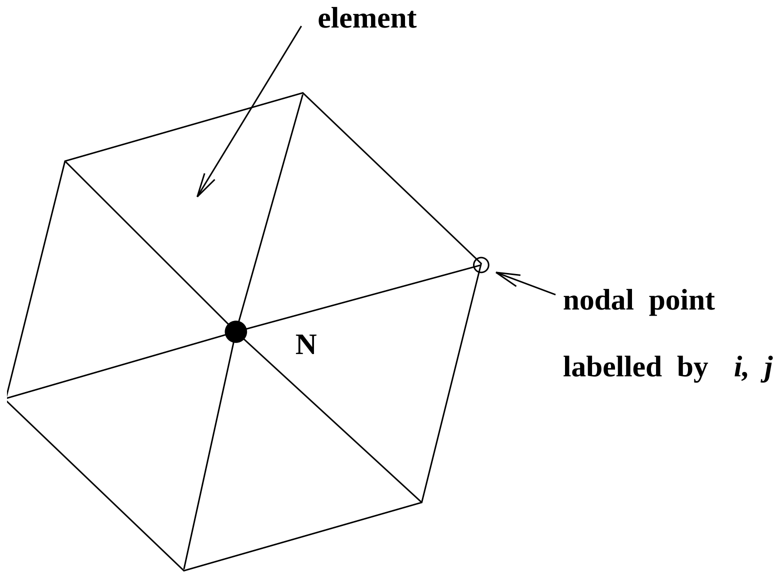


Figure: FEM 7

Conditions on Triangulation

- You need nicely shaped triangles to ensure that

$$\sum_{j \neq i} |S_{ij}| \leq |S_{ii}|$$

  for each i (diagonal dominance)

- This is satisfied if triangulation is such that $\theta_1 + \theta_2 \leq 180^\circ$ for every internal edge e (Figure FEM 8a).

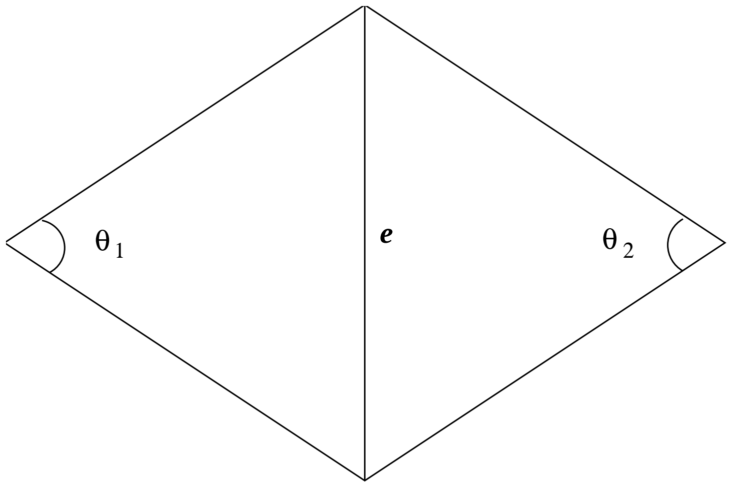


Figure: FEM 8a

- Skinny triangles violate this!

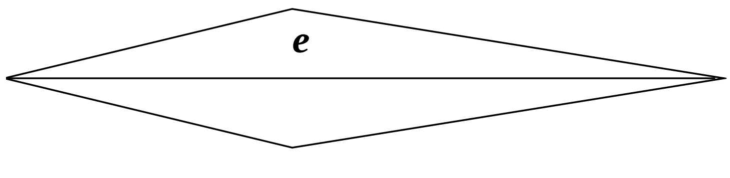


Figure: FEM 8b

Introduction to Poor Person's Conjugate Gradient

- Finite Element Method leads to minimization of a Quadratic Form

$$F(\underline{x}) = \frac{1}{2}\underline{x}^T A\underline{x} - \underline{x}^T b \tag{1}$$

with customary solution

$$A\underline{x} = \underline{b} \tag{2}$$

derived by differentiating (1) with respect to $\underline{x}$ or by completing the square

$$F(\underline{x}) = \frac{1}{2}\left(\nabla - A^{-1}\underline{b}\right)^T \underline{A}\left(\underline{x} - A^{-1}\underline{b}\right) - \frac{1}{2}\underline{b}^T A^{-1}\underline{b}$$

- We will derive an iterative approach (PPCG) to (2) by an intuitive minimization method applied to (1). This is not the true conjugate method, but has the basic idea. As usual, iterations are labelled by (k) and we start with $\underline{x}^{(k)}$

Conjugate Gradient Iteration for Quadratic Form

- $\underline{x}^{(k)}$ is a point in an n dimension space (n is number of equations in $A\underline{x} = \underline{b}$)

$$-\frac{\partial F}{\partial x} = -\nabla F = \underline{b} - A\underline{x}^{(k)} = \underline{r}^{(k)}$$

- $\underline{r}^{(k)}$ is residual
- Let $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha \underline{r}^{(k)}$ (α a scalar, i.e., a number).
- F becomes a function of α for this special choice of $\underline{x}^{(k+1)}$. This is useful as it is much easier to minimize functions of one variable than those that depend on n.

$$\begin{aligned} F(\alpha) &= \alpha^2 \left[\frac{1}{2}\underline{r}^{(k)T} A \underline{r}^{(k)}\right] \\ &- \alpha \underline{r}^{(k)T} \cdot \underline{r}^{(k)} \\ &+ \text{Terms Independent of } \alpha \end{aligned}$$

Conjugate Gradient and Method of Steepest Descent

- We can complete the square

$$\begin{aligned} F(\alpha) &= \frac{1}{2}\underline{r}^{(k)T} A \underline{r}^{(k)} \left(\alpha - \alpha_k\right)^2 \\ &+ \text{ Terms Independent of } \alpha \end{aligned}$$

- where $\alpha_k = \underline{r}^{(k)T} \cdot \underline{r}^{(k)} / \underline{r}^{(k)T} A \underline{r}^{(k)}$
- Note that if

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha_k \underline{r}^{(k)}$$

$$F\left(\underline{x}^{(k+1)}\right) \leq F\left(\underline{x}^{(k)}\right) \text{ rigorously}$$

- This is Poor Person's Conjugate Gradient or when applied to minimization, the method of steepest descent. It is called this because we move from $\underline{x}^{(k)}$ along direction $-\nabla F$—the direction along which F varies most rapidly.

Conjugate Gradient for Finite Element Problems

- Conjugate Gradient can be applied to any matrix problem.
- It is associated naturally with finite element problems because both finite element and conjugate gradient use similar minimization procedures.

Poor Person's Conjugate Gradient and Eigenvalues of Matrix

- What is wrong with this version (PPCG) of conjugate gradient?
- More generally, what is wrong with steepest descent?
- The matrix **A** is symmetric and can be diagonalized as usual

$$\begin{aligned} A &= P^{-1} \Lambda P \\ \Lambda &= \text{diag. } (\lambda_1\,, \lambda_2\,, \ldots \lambda_n\,) \\ &= \begin{bmatrix} \lambda_1 & 0 & 0 & \ldots \\ 0 & \lambda_2 & 0 & \ldots \\ 0 & 0 & \lambda_3 & \ldots \end{bmatrix} \\ & \lambda_1 \geq \lambda_2 \geq \ldots \end{aligned}$$

- P is orthogonal matrix $P^T P = I$
- Let

$$\begin{aligned} \underline{\tilde{x}}^{(k)} &= P \underline{x}^{(k)} \\ \underline{\tilde{b}} &= P \underline{b} \\ \underline{\tilde{r}}^{(k)} &= P \underline{r}^{(k)} \end{aligned}$$

Diagonalization of Quadratic Form

- Note that for any vector $\underline{x}$

$$
\begin{aligned}
& \frac{1}{2}\underline{x}^T A \underline{x} - \underline{b}^T \cdot \underline{x} \\
= \; & \frac{1}{2} \underbrace{\underline{\mathrm{x}}^{\mathrm{T}} \mathrm{P}^{-1}}_{x^T P^T} \Lambda P \underline{x} - \underbrace{\underline{b}^T P^{-1}}_{(\mathrm{P}\underline{\mathrm{b}})^{\mathrm{T}}} P \underline{x} \\
= \; & \frac{1}{2}(P\underline{x})^T \Lambda P \underline{x} - (P\underline{b})^T P \underline{x} \\
= \; & \frac{1}{2}\underline{\tilde{x}}^T \Lambda \underline{\tilde{x}} - \tilde{b}^T \underline{\tilde{x}}
\end{aligned}
$$

- Note this transformation is correct but not useful as a solution technique as it takes longer to find Λ and P than to solve equations $A\underline{x} = \underline{b}$.
- In transformed space denoted by $\tilde{}$ superscript

$$\boxed{\tilde{x}|_{i\ \mathrm{compt}} = \tilde{b}_i / \lambda_i}$$

Diagonalization of Conjugate Gradient Equations

- We can take all the PPCG equations and multiply them by the transformation matrix P and convert them into "$\tilde{}$" equations.
- Scale factor α

$$\alpha^{(k)} = \frac{\underline{\tilde{r}}^{(k-1)T} \cdot \underline{\tilde{r}}^{(k-1)}}{\underline{\tilde{r}}^{(k-1)T} \Lambda \underline{\tilde{r}}^{(k-1)}}$$

- Iteration

$$\underline{\tilde{x}}^{(k)} = \underline{\tilde{x}}^{(k-1)} + \alpha_k \underline{\tilde{r}}^{(k-1)}$$

- Residual

$$\begin{aligned} \underline{\tilde{r}}^{(k)} &= \underline{\tilde{b}} - \Lambda\, \underline{\tilde{x}}^{(k)} \\ &= \underline{\tilde{b}} - \Lambda\, \underline{\tilde{x}}^{(k-1)} - \alpha_k \Lambda \underline{\tilde{r}}^{(k-1)} \\ &= (I - \alpha_k \Lambda)\, \tilde{r}^{(k-1)} \end{aligned}$$

- Residual components in diagonalized coordinate system

$$\tilde{r}^{(k)}\big|_i = (1 - \alpha_k \lambda_i)\, \tilde{r}^{(k-1)}\big|_i$$

where $i = 1 \ldots n$ labels components

Convergence of Conjugate Gradient in Diagonalized Form

- We want $\tilde{r}_k|_i = 0$ for all i
- But

$$\alpha_k = \frac{\Sigma_i \left[\left(\tilde{r}^{(k-1)}\right)|_i\right]^2}{\Sigma_i \lambda_i \left[\tilde{r}^{(k-1)}|_i\right]^2}$$

- We order eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$
- If $\tilde{r}^{(k-1)}|_1$ non zero and reasonably big
- Then

$$\begin{aligned} \alpha_k &\sim \frac{1}{\lambda_1} \text{ and} \\ \tilde{r}^{(k)}|_1 &= \left(1 - \frac{\lambda_1}{\lambda_1}\right) \tilde{r}^{(k-1)}|_1 \\ &\sim 0 \end{aligned}$$

- But other $\tilde{r}_k|_i \;\; i > 1$ are largely unchanged if eigenvalues $\lambda_i \ll \lambda_1$ because as $\tilde{r}^{(k-1)}|_i$ is multiplied by $(1 - \lambda_i/\lambda_1) \sim 1$ (when $\lambda_i \ll \lambda_1$) on going from iteration $(k-1)$ to (k)

Clarification of Eigenvalue Analysis for Conjugate Gradient and Jacobi Iteration

- Note in eigenvalues: In Jacobi/SOR, "large" eigenvalues converged slowly, "small" eigenvalues converged fast.
- But the matrices aren't the same.
- Conjugate gradient (CG) uses complete matrix $\mathbf{A}$
- SOR uses decomposed matrix $\mathbf{A} = \mathbf{M} + \mathbf{N}$
- $M^{-1}N$ eigenvalues determine SOR
  $\mathbf{A}$ eigenvalues determine CG
- Large eigenvalues of $\mathbf{A} \rightarrow$ small eigenvalues of SOR and vice versa.

Intuitive Description of Poor Person's Conjugate Gradient Algorithm

- So Poor Person's Conjugate Gradient is very good at large eigenvalues, but converges slowly for small eigenvalues of A.
- Note A is positive definite for simple pde's $\Rightarrow \lambda_i \geq 0$.
- What we would like to do is to arrange first that $\tilde{r}^{(k)}|_1 = 0$, then we see that equations give $\tilde{r}^{(k+1)}|_2 = 0$, and then the next iteration will give $\tilde{r}^{(k+2)}|_3 = 0$, and so on.
- So qualitative goal of real conjugate gradient is to successively minimize along the "principal directions of matrix **A**."
- As described in my optimization notes, this is easily understood in case $n = 2$ in terms moving in hills and valleys.

Improvement of Poor Person's Conjugate Gradient with Orthonormal Iteration

- In practice $|\lambda_1 = \lambda_{\max}| \gg |\lambda_n = \lambda_{\min}|$ and in simple cases $\frac{\lambda_n}{\lambda_1} \sim h^2$ where h is grid spacing
- We must modify PPCG to keep search directions properly orthogonal so that eigenvectors with large eigenvalues do not dominate
- We set

$$\underline{x}^{(k)} = \underline{x}^{(k-1)} + \alpha_k \underline{p}^{(k)}$$

where we choose

$$\underline{p}^{(k)T} A \underline{p}^{(k-1)} = 0$$

Full Conjugate Gradient Algorithm

- Initialize in true conjugate gradient

$$
\begin{array}{lrcl}
 & \underline{r}^{(0)} & = & \underline{b} - A\underline{x}^{(0)} \\
k = 1 & p^{(1)} & = & \underline{r}^{(0)} \\
k > 1 & p^{(k)} & = & \underline{r}^{(k-1)} + \beta_k \underline{p}^{(k-1)}
\end{array}
$$

- This subtracts off $\underline{p}^{(k-1)}$ component in $\underline{r}^{(k-1)}$
  - as long as:

$$\beta_k = -\frac{\underline{p}^{(k-1)T} A \underline{r}^{(k-1)}}{\underline{p}^{(k-1)T} A \underline{p}^{(k-1)}}$$

- Then as before, solve for α_k so that $\underline{x}^{(k-1)} + \alpha_k \underline{p}^{(k)}$ minimizes the quadratic form $-1/2\underline{x}^T A \underline{x} - \underline{b}^T \cdot \underline{x}$
- This gives

$$
\begin{aligned}
\alpha_k &= \underline{p}^{(k)T} \underline{p}^{(k)} / \underline{p}^{(k)T} A \underline{p}^{(k)} \\
\underline{x}^{(k)} &= \underline{x}^{(k-1)} + \alpha_k \underline{p}^{k} \\
\underline{r}^{(k)} &= b - A x^{(\underline{k})}
\end{aligned}
$$

Overview of Parallelism in Conjugate Gradient

- One must
  1. Calculate matrix elements A_{ij}
  2. Calculate various:
     - Matrix-vector $\Sigma_j \, A_{ij}x_j$ products
     - and vector-vector $\Sigma_j \, x_j y_j$ scalar products
     - for several different vectors
       (**A** is always stiffness matrix).

Parallel Issues in Calculation of Matrix Elements

- Divide world up into geometric domains as usual (Figure FEM 9)

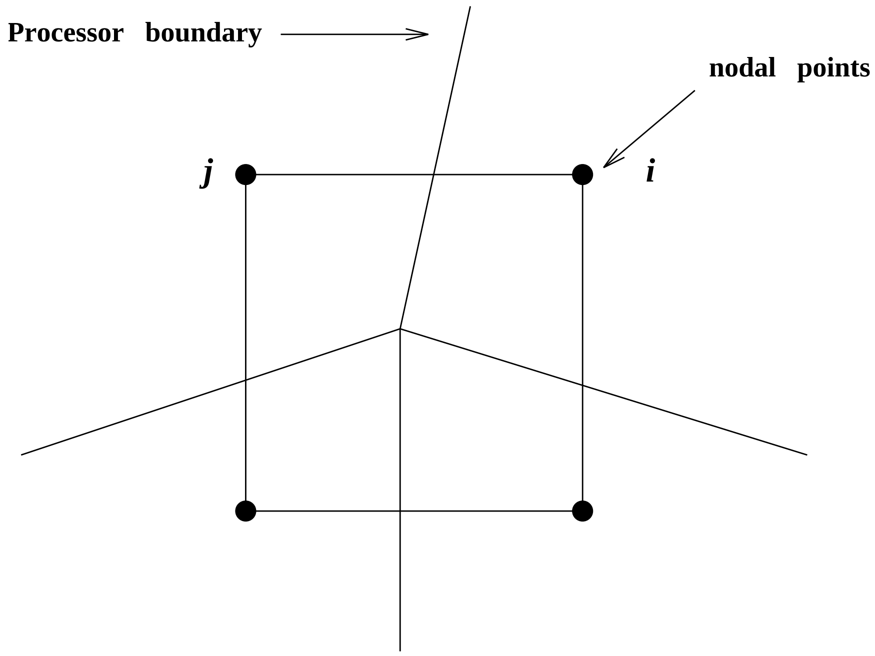


Figure: FEM 9

- Information (triangles with i, j as an edge) to calculate A_{ij} can involve ≥ 2 processors (four is most in two-dimensional example), but communication is local.
- This is called assembly.
- Each processor should know A_{ii} and A_{ij} for all j connected to each i "owned" by processor.

Scalar Products in Parallel

Conjugate Gradient

- When one calculates $\Sigma_j A_{ij} x_j$, send x_j to all processors "owning" x_i where $A_{ij} \neq 0$
- This gives local communication
- When one calculates

$$\sum_{j=1}^{n} y_j x_j$$

First calculate sums in a given processor, summing in Processor P over all nodal points j in P. Then one must form

- a Global "combine".

$$\sum_{\text{Processors } P} \left[\sum_{j \epsilon P} y_j x_j \right]$$

Typically, this same sum must be sent to all processors.

Preconditioning in Conjugate Gradient

- This is a general technique, where one finds M such that $M\underline{x} = \underline{y}$ can be easily solved and $M^{-1}A$ is numerically faster (easier or more stable) to solve than A

  i.e., M is an approximation to A whose inverse can be found easily—more precisely, for which linear equations $M\underline{x} = \underline{y}$ can be solved as one never inverts matrices in this type of problem.

- The simplest preconditioner is $M =$ Diagonal part of A.

- The best solution methods use sophisticated preconditioners such as those given by multigrid.

Convergence of Conjugate Gradient

- The number of conjugate gradient iterations needed for given accuracy for simple problems (positive definite matrix $\mathbf{A}$) is given by

$$k = \sqrt{\frac{\lambda_{\rm max}}{\lambda_{\rm min}}} \propto \frac{1}{h}$$

  where h is grid size

- Remember SOR was $1/h$ <u>if</u> overrelaxation parameter w chosen correctly, which is not easy.
- Jacobi and Gauss Seidel have a number of iterations $k \propto 1/h^2$ which is much more
- Conjugate gradient iterations are more complicated and time consuming than those for Jacobi or Gauss Seidel

Mathematical and Pseudo Code Form of Gauss Seidel Iteration Method

```
Choose an initial guess x^(0) to the solution x.
for k = 1,2,...
    for i = 1,2,...,n
        σ = 0
        for j = 1,2,...,i − 1
            σ = σ + a_{i,j} x_j^(k)
        end
        for j = i + 1,...,n
            σ = σ + a_{i,j} x_j^(k−1)
        end
        x_i^(k) = (b_i − σ) / a_{i,i}
    end
    check convergence; continue if necessary
end
```

Pseudo code for Gauss-Seidel Method

Mathematical (Matrix) Form of Gauss Seidel

- Mathematically, Gauss Seidel is

$$x_i^{(k)} = \frac{\left[b_i - \sum_{j<i} a_{i,j}\, x_j^{(k)} - \sum_{j>i} a_{i,j}\, x_j^{(k-1)} \right]}{a_{i,i}}$$

- In matrix language

$$\underline{x}^{(k)} = (D - L)^{-1} \left(U \underline{x}^{(k-1)} + \underline{b} \right)$$

- where D=Diagonal, L=lower, and U=Upper parts of matrix **A**
- **A**$= D - L - U$

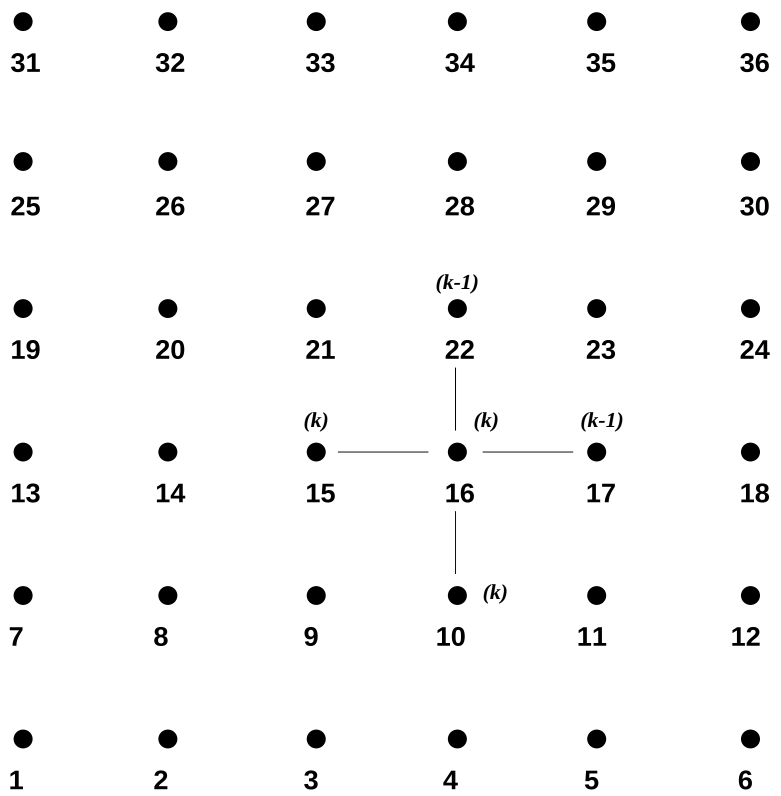
Parallelism in Gauss-Seidel Iteration

- Conventional Gauss-Seidel canNOT be easily parallelized because it has sequential update order

$$
\begin{array}{rcl}
x_n^{(k)} & \text{requires} & x_{n-1}^{(k)} \quad \ldots \\
\\
x_{n-1}^{(k)} & \text{requires} & x_{n-2}^{(k)} \quad \ldots \\
\vdots & & \\
x_2^{(k)} & \text{requires} & x_1^{(k)}, \text{etc.}
\end{array}
$$

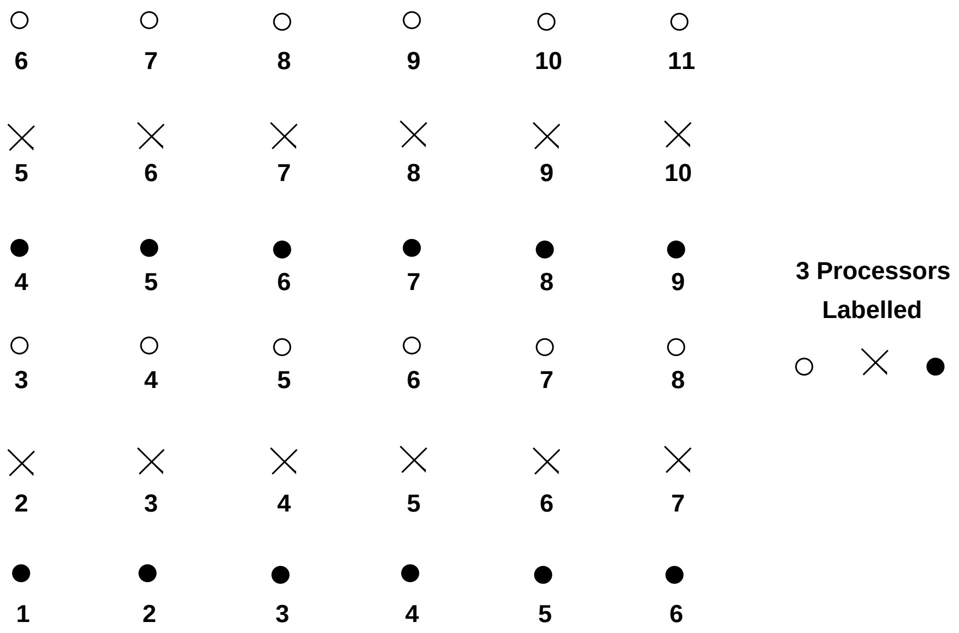
- There are two parallelization strategies
  1. Keep same algorithm—pipelined wavefront method
  2. Change algorithm—red/black two-phase

6×6 **Matrix Example Stencil**



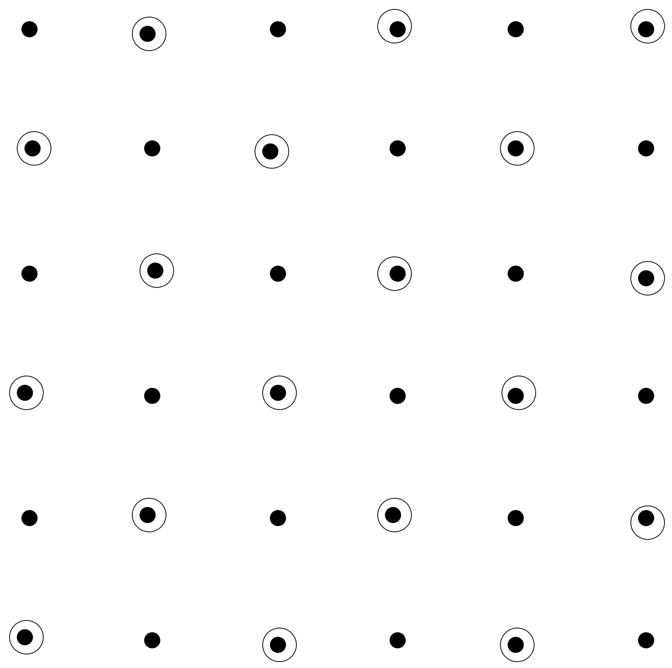
- We can exploit special structure of matrix of **A** (e.g., $x_{16}^{(k)}$ requires $x_{15}^{(k)}$, $x_{17}^{(k-1)}$, $x_{10}^{(k)}$, $x_{22}^{(k-1)}$, etc.).

6×6 Matrix—Wavefront Parallelism for Gauss Seidel



- Here number l labels $l = 1 \ldots 11$ phases
- Each phase l ONLY depends on $\underline{x}^{(k)}$ components from $(l-1)$ th phase
- Can execute each update in a given phase concurrently
- Some Load Imbalance
- Speedup of 2.5 in example of three processors

The Red-Black Two Phase Parallel Gauss Seidel Iteration



Red $\odot$ Black $\bullet$

- Note calculation of $\odot$ only depends on $\bullet$ points and vice versa
- So two phase algorithm.

  Calculate all $\odot$—Phase I.

  Then all $\bullet$—Phase II.

Analysis of Parallel Red Black Gauss Seidel

- This can be parallelized as easily as Jacobi!
  - Phase I follows by Phase II.
- Red black is a different ordering from original Gauss Seidel.
- For (some) elliptic problems, can show red black as good as and maybe better than original ordering.

  — i.e., red black does not need more iterations than pure Gauss Seidel.
- For related hyperbolic problems—see NAS benchmark SSOR discussion in CPS713—not so clear.
- For higher order differencing, need more phases (more colors) but principle the same.
- Also not clear how to use if irregular matrix, as in sparse matrices from power systems.

Eigenvalues of Gauss Seidel Iteration Matrix

- It was relatively easy to find the eigenvalues and eigenfunctions of Jacobi's method for a simple geometry.
- Gauss-Seidel it is not so easy as matrix $M^{-1}N$ is not symmetric. There is a nice discussion in

  "Numerical Computation of Internal and External Flows," C. Hirsch, Wiley, 1988.

- Eigenvalues λ_{GS} of Gauss-Seidel Iteration Matrix are related to those of Jacobi Iteration λ_J by $\lambda_{GS} = \lambda_J^2$
- We showed that

$$1 - \epsilon \lesssim \lambda_J \lesssim -1 + \epsilon \qquad \epsilon \sim \frac{\pi^2}{2N^2}$$

- Therefore,

$$|\lambda_{GS}|_{\max} = (|\lambda_J|_{\max})^2 \sim 1 - 2\epsilon$$

Comparison of Convergence of Gauss-Seidel and Jacobi Iteration

- Convergence can be characterized by a given fractional error (say .001).
- This correponds to

$$|\lambda|_{\max}^{k} \sim .001$$

  where k number of required iterations
- $\exp\left[-k\left(1-|\lambda|_{\max}\right)\right] = .001$
  i.e., value of $k \propto \frac{1}{(1-|\lambda|_{\max})}$
- We showed Gauss Seidel doubles value of

$$1/(1-|\lambda|_{\max}),$$

  i.e., it converges in half as many iterations as Jacobi

Successive Overrelaxation Iteration Method (SOR)

- This is relaxation (acceleration) method applied to Gauss-Seidel.
- In Jacobi, the optimal case was $\omega = 1$ (i.e., no gain in using relaxation).
- However, for Gauss-Seidel, we will find $\omega \sim 2$ is best and does improve the method.
- The spectral radius ($|\lambda|_{\max}$) of Jacobi is given by

$$\rho_J = 1 - \frac{\pi^2}{2N^2}$$

and for conventional (unaccelerated) Gauss Seidel by

$$\rho_{GS} = 1 - \frac{\pi^2}{N^2}$$

- For choice

$$\omega_{opt} \sim 2\left(1 - \frac{\pi}{N}\right)$$

one can show that

$$\rho_{SOR} \sim 1 - \frac{\pi}{N}$$

Convergence of SOR Compared to Jacobi and Gauss Seidel

- Remember number of iterations is proportional to

$$
\begin{aligned}
\frac{1}{1-\rho} &= \frac{2N^2}{\pi^2} \quad \text{Jacobi} \\
&= \frac{N^2}{\pi^2} \quad \text{Gauss Seidel} \\
&= \frac{N}{\pi} \quad \text{SOR}
\end{aligned}
$$

— i.e., number of iterations decreases in SOR by a factor N/π, which is large!

- Unfortunately, the above analysis is only valid in simple cases
    - simple geometry
    - some orderings in Gauss-Seidel

Estimate of Over Relaxation Parameter ω

- Thus, in general for practical problems, one cannot precalculate optimal overrelaxation parameter.
- Find ω by numerical experimentation.
- However, note that one can expect a choice $1 < \omega < 2$ to be better than natural $\omega = 1$.
- One can use the estimate

$$\begin{aligned} \omega_{opt} &= 2 - \frac{\text{constant}}{N} \\ &= 2 - \text{constant} \cdot h \end{aligned}$$

(h is grid spacing)

Pseudo Code for SOR—Successive Over Relaxation

Choose an initial guess $x^{(0)}$ to the solution x.

for $k = 1,2,\ldots$
    for $i = 1,2,\ldots,n$
        $\sigma = 0$
        for $j = 1,2,\ldots,i-1$
            $\sigma = \sigma + a_{i,j}\, x_j^{(k)}$
        end
        for $j = i+1,\ldots,n$
            $\sigma = \sigma + a_{i,j}\, x_j^{(k-1)}$
        end
        $\sigma = (b_i - \sigma)\,/a_{i,i}$
        $x_i^{(k)} = x_i^{(k-1)} + \omega\left(\sigma - x_i^{(k-1)}\right)$
    end
    check convergence; continue if necessary
end

The SOR Method