Solving PDEs using multiple CPUs
Overview

- Finite Elements
- Domain Decomposition
- Bank-Holst Paradigm
General 2\textsuperscript{nd} Order Linear PDE:

\[ Lu \equiv -\nabla \cdot (a(x)\nabla u) + b(x) \cdot \nabla u + c(x)u = f, \text{ in } \Omega \]

\[ n \cdot (a(x)\nabla u) = g_N, \text{ on } \partial_N \Omega \]

\[ u = g_D, \text{ on } \partial_D \Omega \]

Solve for \( u \).

Given are \( \Omega, \)

\( a(x), b(x), c(x), \)

\( f(x), g_N(x), g_D(x). \)

Linear Elasticity Example:

\[ -2\mu(\nabla \cdot \varepsilon(u)) - \lambda \nabla^2 u = f(x) \text{ in } \Omega \]

\[ \sigma(u) \cdot n = g(x) \text{ on } \partial_N \Omega \]

\[ u = 0 \text{ on } \partial_D \Omega \]

For more info, see Toselli and Widlund [1]
$u$ is too difficult to find, so we find $u_h$

$u$ is infinite dimensional, for example $u : \mathbb{R}^2 \rightarrow \mathbb{R}$

So we find a finite dimensional approximation $u_h = \sum_{i=1}^{n} \alpha_i \phi_i$, $\phi_i : \mathbb{R}^2 \rightarrow \mathbb{R}$
This allows us to solve for a finite number of unknowns.

Convert Strong Form into Weak Form: \( Lu = f \rightarrow \int_{\Omega} Luv = \int_{\Omega} f v \quad \forall v \)

\[
A(u,v) = F(v) \quad \forall v \in H^1(\Omega)
\]

\[
A(u,v) \equiv \int_{\Omega} \left( a(x) \nabla u \cdot \nabla v + (b(x) \cdot \nabla u) v + c(x) uv \right) dx
\]

\[
F(v) \equiv \int_{\Omega} fvdx + \int_{\partial N \Omega} g_N v - A(g_D, v)
\]

Then replace \( u \) with \( u_h \) to get a system on linear equations:

\[
A\alpha = F \quad \text{where} \quad \left[ A \right]_{i,j} = A(\phi_i, \phi_j) \quad \text{and} \quad \left[ F \right]_i = F(\phi_i)
\]
The Finite Element solution is a good approximation

With appropriate assumptions

\[ \| u - u_h \|_{\alpha, \Omega} \leq h^{2-\alpha} \| u \|_{2, \Omega} \]

See Babuska and Aziz [2]

Here is an example

using 200 unknowns.

It seems to capture continuous life well.
Many problems take too long to solve.

If we solve $A\alpha = F$ for $10^9$ unknowns in $O(n^{1.5})$ it could take days!

It is better to solve 1000 problems of size $10^6$ simultaneously.
Domain Decomposition

1. Simultaneously Solve
2. Communicate
3. Repeat

Non-overlapping:

Overlapping:

Completely overlapping:
Communication is the same for all.

**FETI Method**

\[ i = 1, 2 : \begin{align*} 
    Lu_i^{(n+\frac{1}{2})} &= f \text{ in } \Omega_i, \text{ BC on } \partial \Omega_i \setminus \Gamma, \\
    n \cdot \left( a(x) \nabla u_i^{(n+\frac{1}{2})} \right) &= \lambda^{(n)}_i \text{ on } \Gamma_i 
\end{align*} \]

Communicate \( u_i^{(n+\frac{1}{2})} \) on \( \Gamma_i \), \( i = 1, 2 \)

\[ i = 1, 2 : \begin{align*} 
    \tilde{u}^{(n+1)}_i &= 0 \text{ in } \Omega_i, \text{ BC on } \partial \Omega_i \setminus \Gamma, \\
    \tilde{u}_i^{(n+1)} &= u_i^{(n+\frac{1}{2})} - u_2^{(n+\frac{1}{2})} \text{ on } \Gamma_i 
\end{align*} \]

\[ \lambda^{n+1} = \lambda^n - \theta \left( n \cdot \left( a(x) \tilde{u}_1^{(n+1)} \right) + n \cdot \left( a(x) \tilde{u}_2^{(n+1)} \right) \right) \text{ on } \Gamma_i \]

Communicate \( \lambda^{(n+1)} \) on \( \Gamma_i \), \( i = 1, 2 \)

**Schwarz Method**

\[ i = 1, 2 : \begin{align*} 
    Lu_i^{(n+1)} &= f \text{ in } \Omega_i, \text{ BC on } \partial \Omega_i \setminus \Gamma, \\
    u_i^{(n+1)} &= g_i(x) \text{ on } \Gamma_i 
\end{align*} \]

Communicate \( u_i^{(n+1)} \) on \( \Gamma_i \), \( i = 1, 2 \)

All methods pass data proportional to their interface area.

For convergence proofs, see Toselli and Widlund [4]
Different Decompositions

Additive Schwarz Method on:

\[-\Delta u = 1 \text{ in } \Omega\]

\[u = 0 \text{ on } \partial \Omega\]

\[\Omega \equiv \text{unit cube } \in \mathbb{R}^3\]

\[P_x \cdot P_y \cdot P_z = 128\]

Deotte and Baden [5]

Actual running times on super computer ABE with 1,000,000 unknowns decomposed unto 128 cores.

Times are in seconds for 20 iterations with and without communication for comparison.
Minimize Communication and Balance Load

This is a challenging problem and can be approximately solved with Recursive Spectral Bisection.

Create Adjacency Matrix, $M$, and solve Eigenvalue Problem: $Mx_i = \lambda_i x_i$

Order Eigenvalues: $0 = \lambda_1 < |\lambda_2| \leq |\lambda_3|$

Then take vector $x_2$ and let $\Omega_1 = \{ [x_2]_i | [x_2]_i > 0 \}$ and $\Omega_2 = \{ [x_2]_i | [x_2]_i \leq 0 \}$

This method is very effective. See Mathew [6]
Analogous to resonance on a drum head.

This is similar to solving the Wave Equation for the second resonance. Informally, resonance balances area of up drum head with down and minimizes bending length to achieve minimal energy oscillation.

\[-\Delta u + \lambda u = 0\]

Deotte and Bank [7]
Place degrees of freedom where they are needed.

Better placement of degrees of freedom.
Best to use more than geometry.

Algorithms refine the mesh and add more unknowns where needed. Consequently, processors can become unbalanced.

A Posteriori Error Estimators

Under appropriate assumptions, we have

\[ \| \nabla (u - u_h) \|_{0, \Omega} \approx \| (I - S^m Q_h) \nabla u_h \|_{0, \Omega} \]

where \( S^m \) is a multigrid smoother and \( Q_h \) is an \( L^2 \) projection onto Finite Element space.

\[ \| \nabla (u - u_h) \|_{0, \Omega} \approx \| \nabla (u_q - u_l) \|_{0, \Omega} \]

where \( u_q \) is the quadratic interpolant of \( u_h \) and \( u_l \) the linear interpolant both using the recovered gradient \( S^m Q_h \nabla u_h \).

See Bank and Xu [8]
Bank-Holst Paradigm

- **Step 1: Load Balancing.** We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements or gridpoints.

- **Step 2: Adaptive Meshing.** Each processor is provided the complete coarse mesh and instructed to sequentially solve the entire problem, with the stipulation that its adaptive refinement should be limited largely to its own partition. The target number of elements and grid points for each problem is the same. Near the end of this step, the mesh is regularized such that the global mesh described in Step 3 will be conforming.

- **Step 3: DD Solve.** A final mesh is computed using the union of the refined partitions provided by each processor. A final solution computed using a domain decomposition or parallel multigrid technique.

See Bank and Holst [9]
Step 1: Load Balancing

\[ \Omega = \]

Solve PDE

High Error

Low Error

\[ \tilde{P}_1 + \tilde{P}_2 \]

Step 2: Adaptive Meshing

\[ P_1 \rightarrow P_2 \]

Step 3: DD Solve

Solve (on each \( P_i \) using their own \( A \)'s):

\[
\begin{bmatrix}
A_1 & 0 & M_1^T \\
0 & A_2 & M_2^T \\
M_1 & M_2 & 0
\end{bmatrix}
\begin{bmatrix}
\delta U_1 \\
\delta U_2 \\
\Lambda
\end{bmatrix}
= 
\begin{bmatrix}
R_1 \\
R_2 \\
U_{\nu} - U_{\gamma}
\end{bmatrix}
\]

Communicate: \( U_{\nu}, R_{\nu}, U_{\gamma}, R_{\gamma} \)
References


Thank You.