Solving PDEs using multiple CPUs

Overview

- Finite Elements
- Domain Decomposition
- Bank-Holst Paradigm

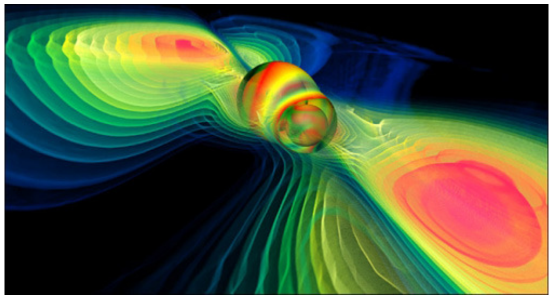
General 2nd Order Linear PDE:

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

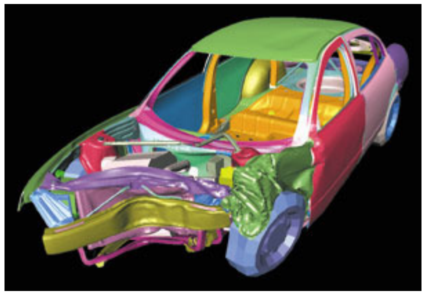
$$n \cdot \left(a(x)\nabla u\right) = g_N, \text{ on } \partial_N \Omega$$

$$u = g_D, \text{ on } \partial_D \Omega$$

Solve for u.
Given are Ω,
$a(x), b(x), c(x)$,
$f(x), g_N(x), g_D(x)$.



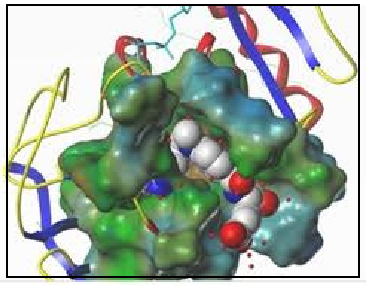
Linear Elasticity Example:



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

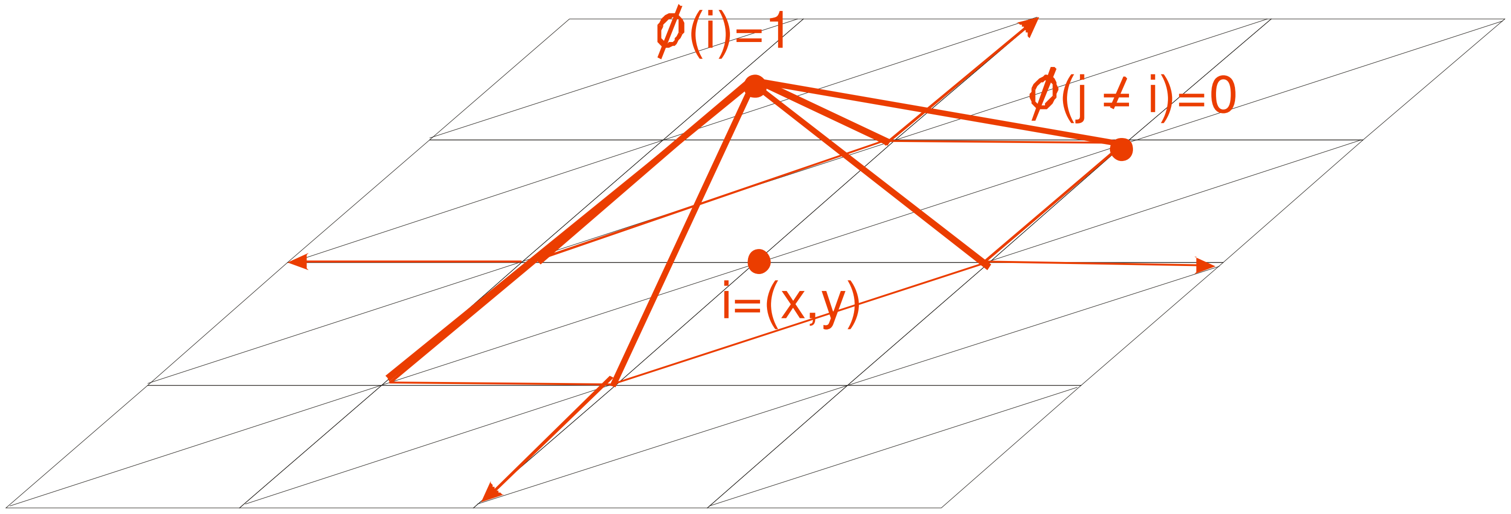
$$\sigma(u) \cdot n = g(x) \text{ on } \partial_N \Omega$$

$$u = 0 \text{ on } \partial_D \Omega$$



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: $Lu = f \rightarrow \int_{\Omega} Luv = \int_{\Omega} fv \ \forall v$

$$A(u,v) = F(v) \ \forall v \in H^1(\Omega)$$

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

$$F(v) \equiv \int_{\Omega} fv dx + \int_{\partial_N \Omega} g_N v - A(g_D, v)$$

Then replace *u* with u_h to get a system on linear equations:

$$\boxed{A\alpha = F} \text{ where } [A]_{i,j} = A(\phi_i, \phi_j) \text{ and } [F]_i = F(\phi_i)$$

The Finite Element solution is a good approximation

Deotte and Holst [3]

With appropriate assumptions

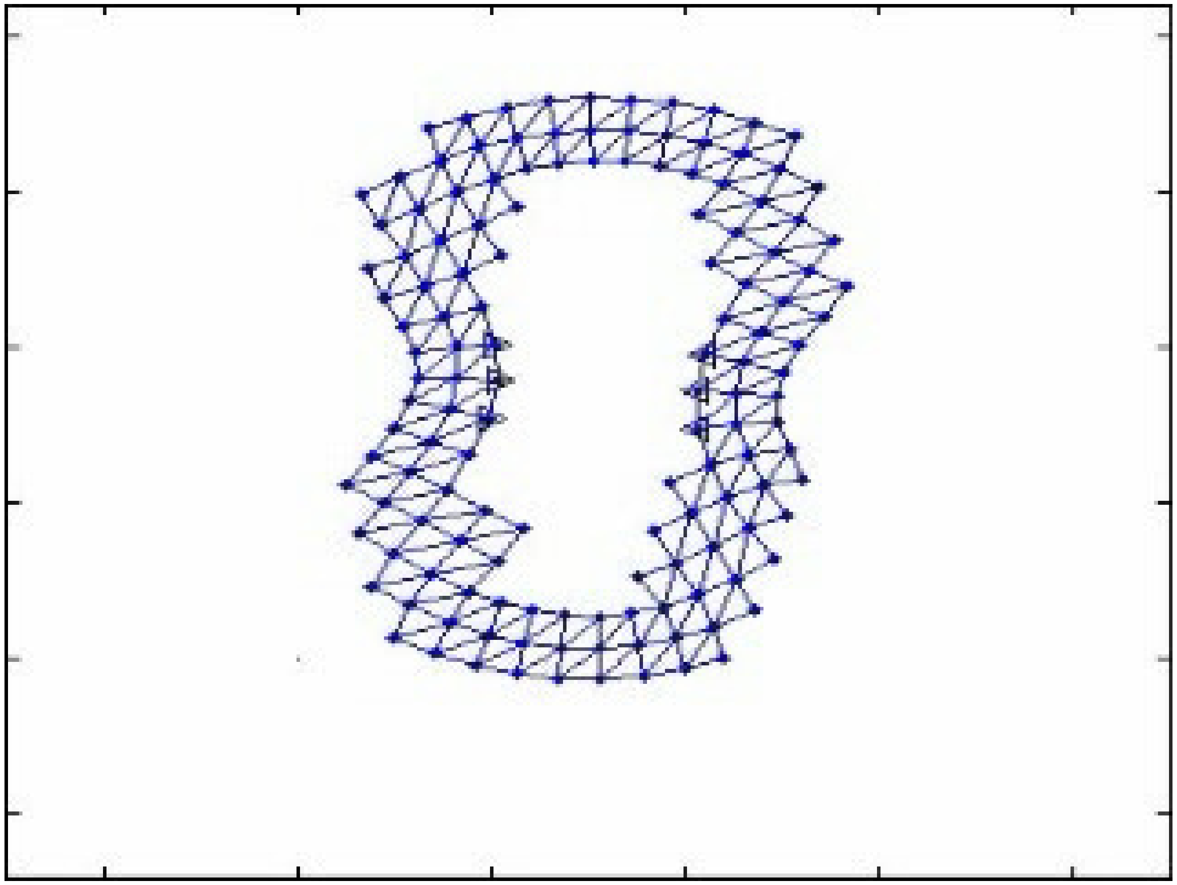
$$\left\| u - u_h \right\|_{\alpha,\Omega} \leq h^{2-\alpha} \left\| u \right\|_{2,\Omega}$$

See Babuska and Aziz [2]

Here is an example

using 200 unknowns.

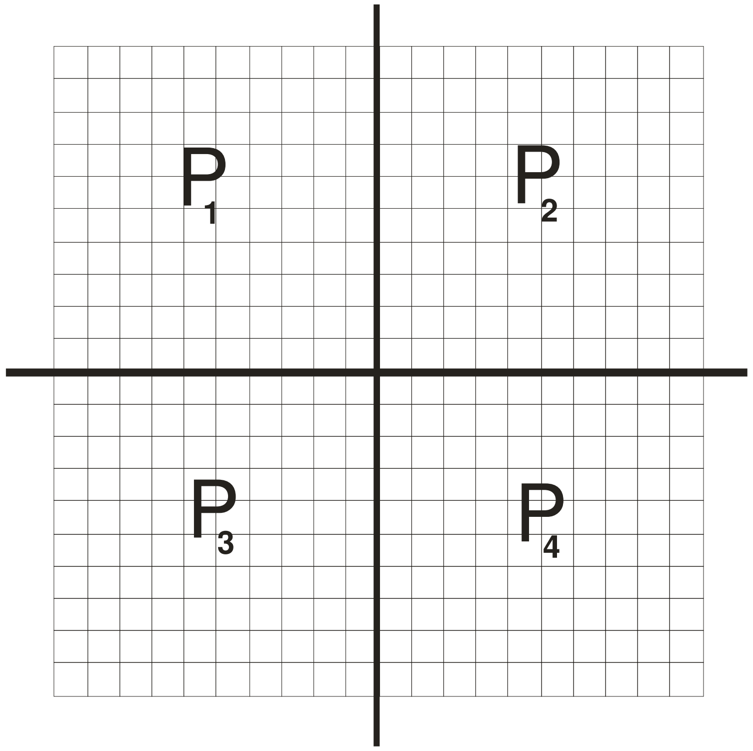
It seems to capture continuous life well.



Many problems take too long to solve.

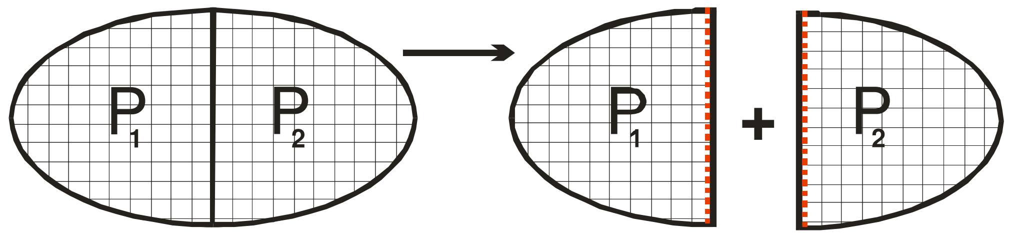
If we solve $A\alpha = F$ for 10^9 unknowns in $O(n^{1.5})$ it could take days!

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



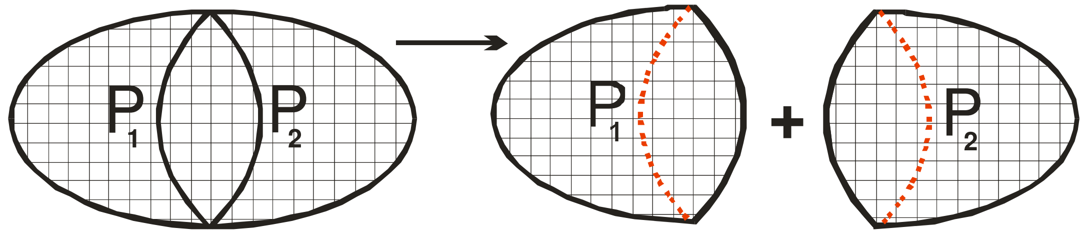
Domain Decomposition

Non-overlapping:

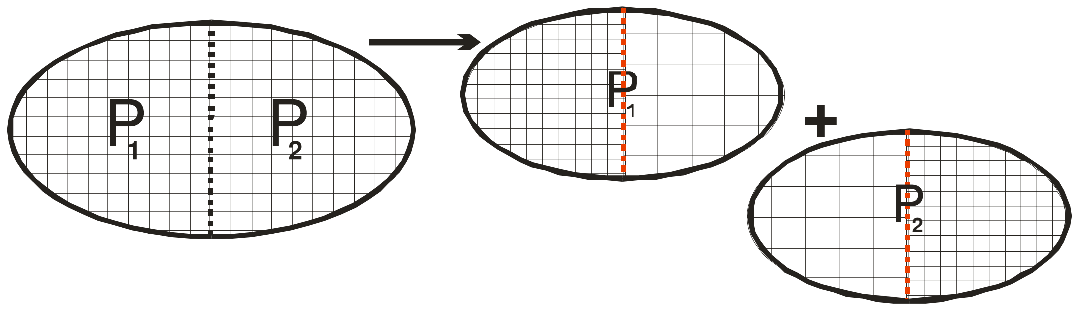


1. Simultaneously Solve
2. Communicate
3. Repeat

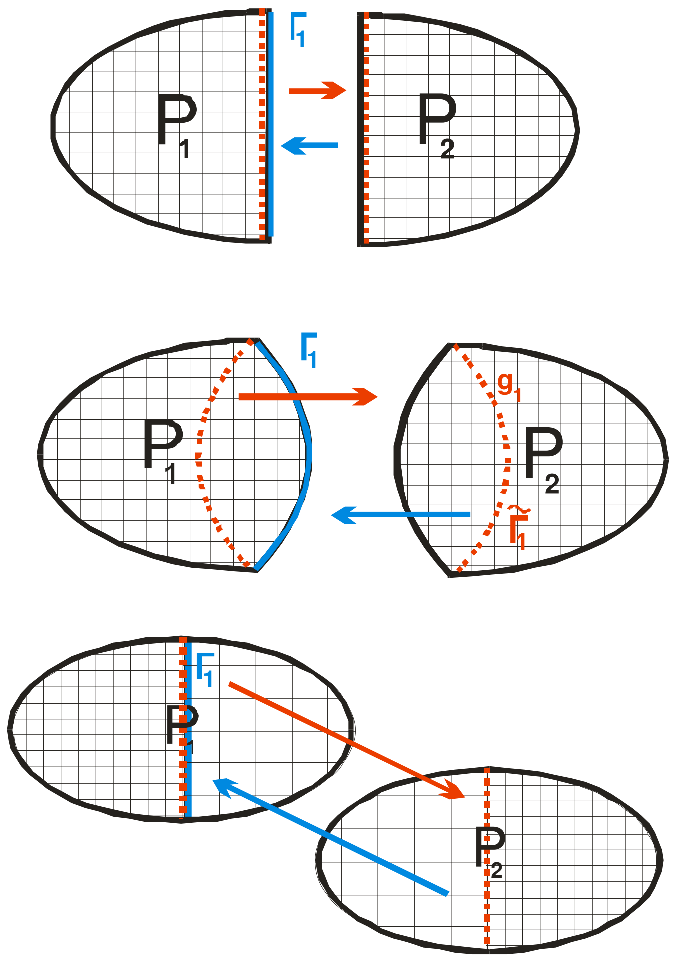
Overlapping:



Completely overlapping:



Communication is the same for all.



FETI Method

$$i=1,2:\ Lu_i^{(n+\frac{1}{2})}=f \text{ in } \Omega_i,\ \text{BC on } \partial\Omega_i\setminus\Gamma,\ n\cdot\left(a(x)\nabla u_i^{(n+\frac{1}{2})}\right)=\lambda_i^{(n)} \text{ on } \Gamma_i$$

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

$$i=1,2:\ L\tilde{u}_i^{(n+1)}=0 \text{ in } \Omega_i,\ \text{BC on } \partial\Omega_i\setminus\Gamma,\ \tilde{u}_i^{(n+1)}=u_1^{(n+\frac{1}{2})}-u_2^{(n+\frac{1}{2})} \text{ on } \Gamma_i$$

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

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

Schwarz Method

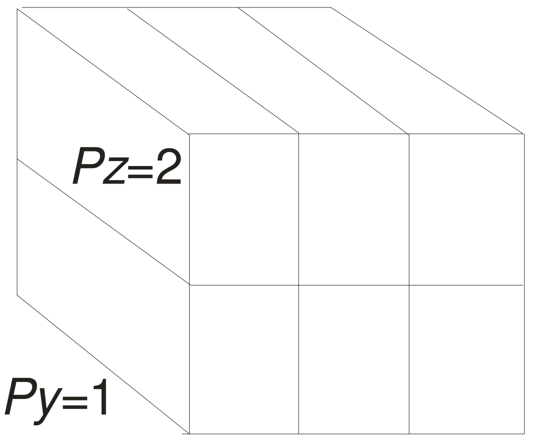
$$i=1,2:\ Lu_i^{(n+1)}=f \text{ in } \Omega_i,\ \text{BC on } \partial\Omega_i\setminus\Gamma,\ u_i^{(n+1)}=g_i(x) \text{ on } \Gamma_i$$

Communicate $u_i^{(n+1)}$ on $\tilde{\Gamma}_i,\ i=1,2$

All methods pass data proportional to their interface area.

For convergence proofs, see Toselli and Widlund [4]

Different Decompositions



Example decomposition

Additive Schwarz Method on:

$$-\Delta u = 1 \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega$$

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

$$Px \cdot Py \cdot Pz = 128$$

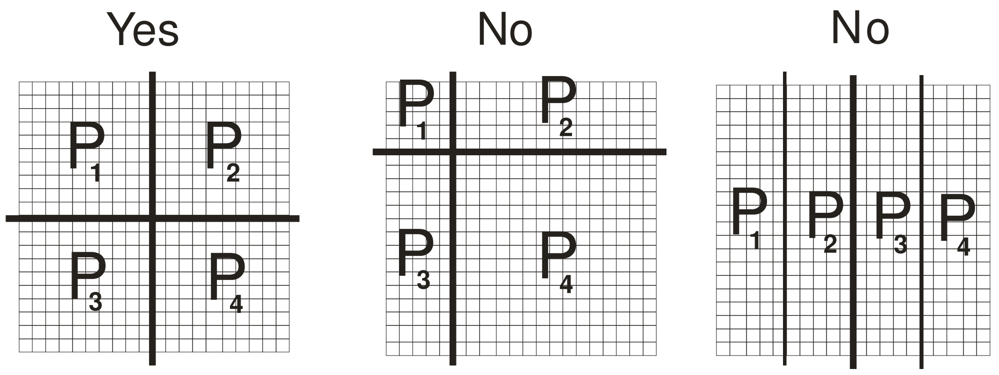
Deotte and Baden [5]

Px	Py	Pz	w/ comm	no comm	comm	percent
1	1	128	23.83	14.95	8.88	37.27
1	2	64	20.33	14.63	5.69	28.01
1	4	32	21.12	14.49	6.63	31.40
1	8	16	17.48	14.20	3.28	18.78
1	16	8	16.55	14.07	2.48	14.99
1	32	4	19.25	13.43	5.81	30.21
1	64	2	16.89	11.88	5.01	29.65
1	128	1	22.35	12.03	10.31	46.16
2	1	64	23.46	14.66	8.80	37.51
2	2	32	21.76	14.49	7.27	33.42
2	4	16	18.26	14.18	4.08	22.32
2	8	8	18.21	13.92	4.29	23.54
2	16	4	16.02	13.36	2.66	16.61
2	32	2	15.23	11.66	3.58	23.49
2	64	1	19.31	11.53	7.78	40.31
4	1	32	20.01	14.54	5.46	27.32
4	2	16	19.11	14.21	4.90	25.65
4	4	8	17.27	13.93	3.34	19.33
4	8	4	19.04	13.20	5.84	30.67
4	16	2	16.26	11.26	4.99	30.72
4	32	1	16.97	11.19	5.78	34.05
8	1	16	18.83	14.31	4.52	24.00
8	2	8	20.26	14.01	6.25	30.85
8	4	4	17.19	13.21	3.98	23.17
8	8	2	15.36	11.43	3.93	25.60
8	16	1	16.70	10.92	5.78	34.60
16	1	8	19.69	14.29	5.40	27.43
16	2	4	18.57	13.29	5.28	28.43
16	4	2	17.34	11.41	5.92	34.17
16	8	1	16.66	10.91	5.75	34.50
32	1	4	21.89	13.99	7.91	36.11
32	2	2	17.28	12.13	5.15	29.80
32	4	1	19.13	11.30	7.83	40.91
64	1	2	22.89	12.64	10.24	44.76
64	2	1	20.97	11.86	9.10	43.42
128	1	1	33.00	13.18	19.83	60.07

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.

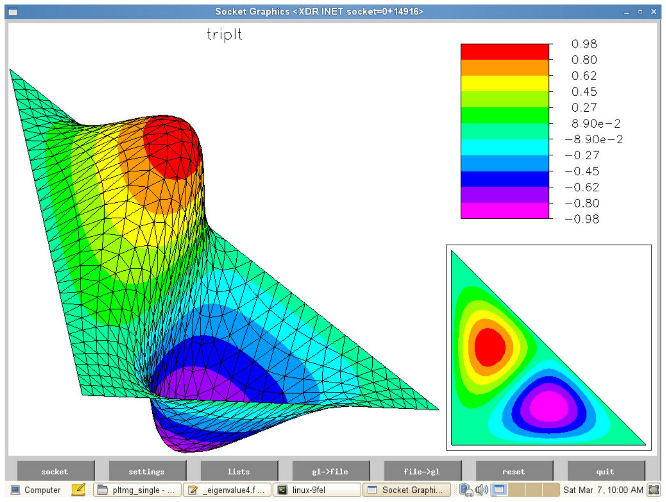
$$[M]_{i,j} = \begin{cases} -1, & \text{if } i \text{ and } j \text{ share edge} \\ \#\text{ of edges}, & \text{if } i = j \\ 0, & \text{else} \end{cases}$$

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

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

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

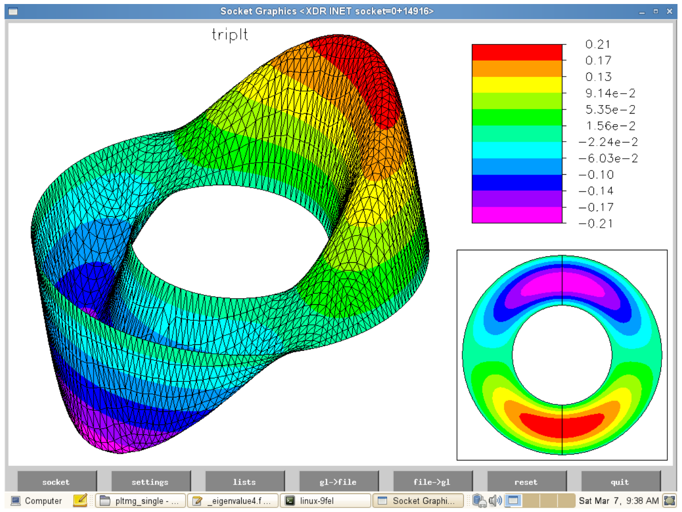
This method is very effective. See Mathew [6]

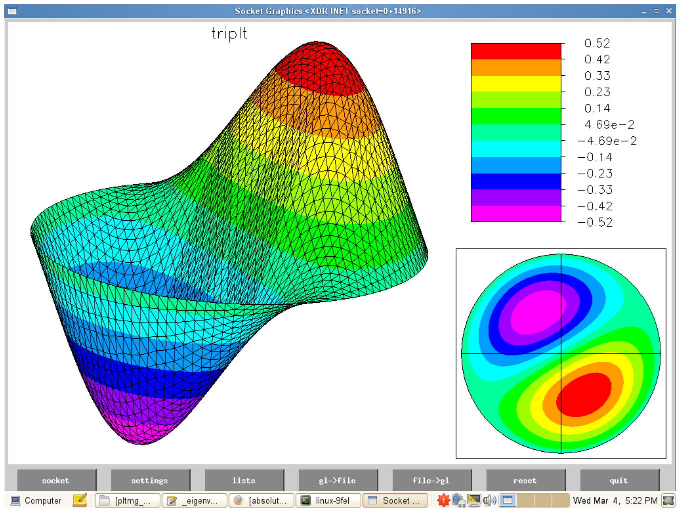


Analogous to resonance on a drum head.

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.

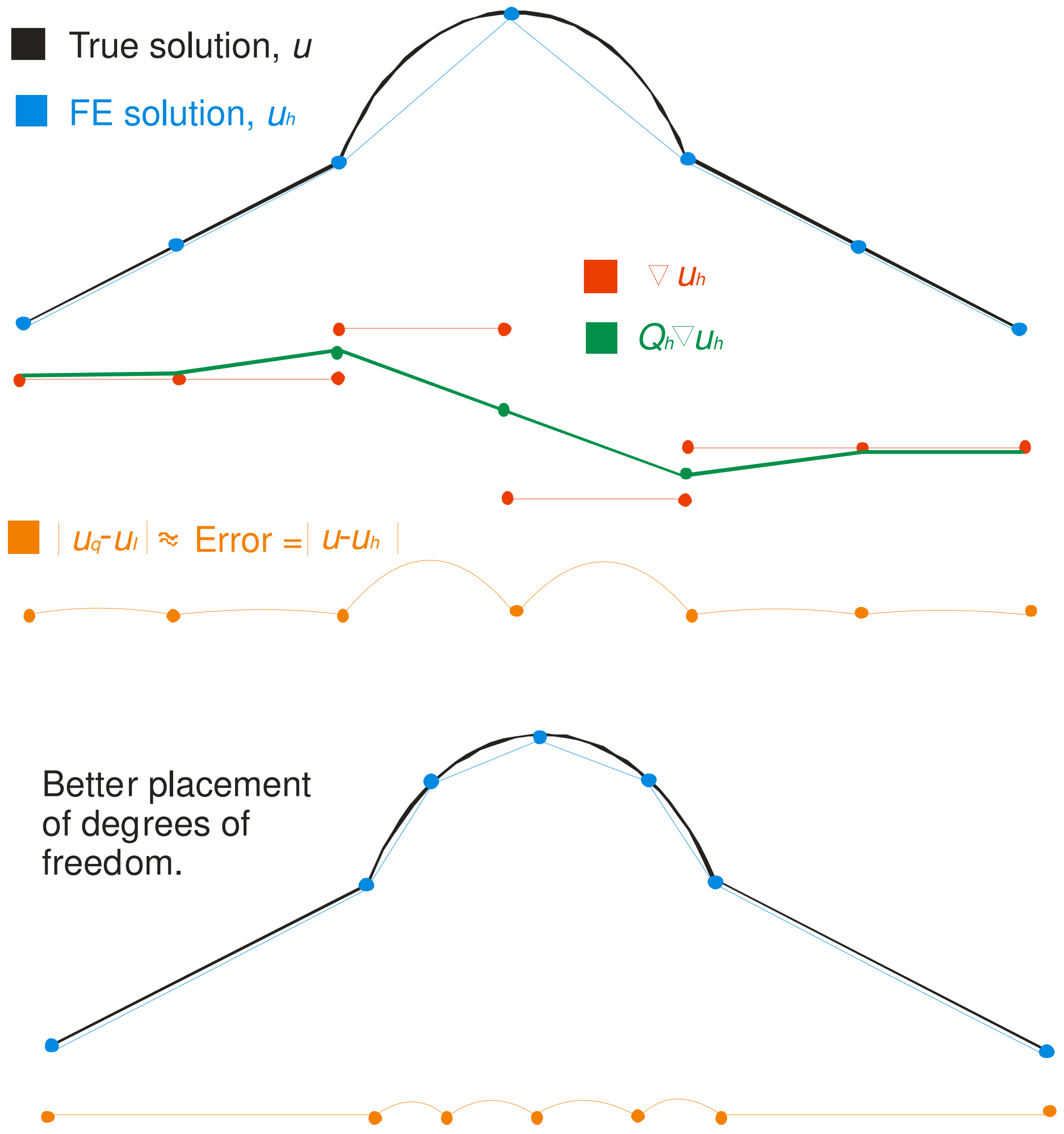
$$-\Delta u + \lambda u = 0$$





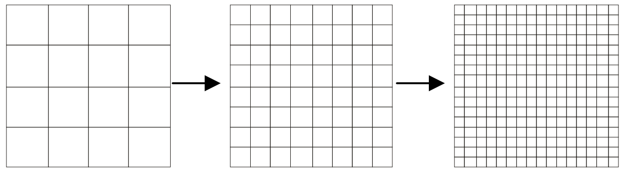
Deotte and Bank [7]

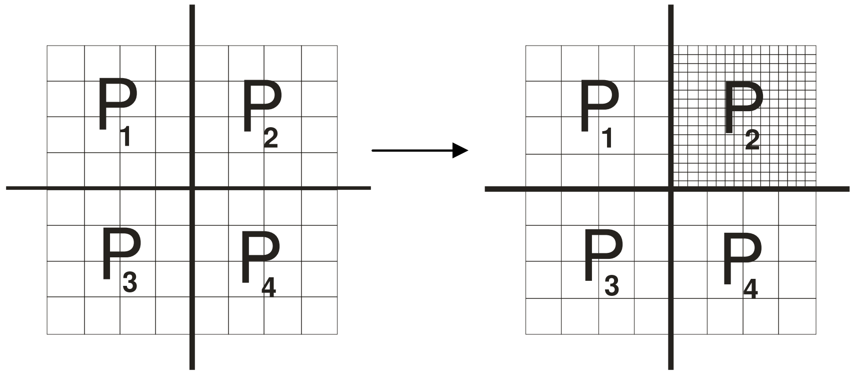
Place degrees of freedom where they are needed.


True solution, u
FE solution, u_h
∇u_h
$Q_h \nabla u_h$
$|u_q - u_l| \approx$ Error $= |u - u_h|$
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

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

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

$$\left\| \nabla (u - u_h) \right\|_{0,\Omega} \approx \left\| \nabla \left(u_q - u_I \right) \right\|_{0,\Omega}$$

where u_q is the quadratic interpolant of u_h and u_I the linear interpolant both using the recovered gradient $S^m Q_h \nabla u_h$

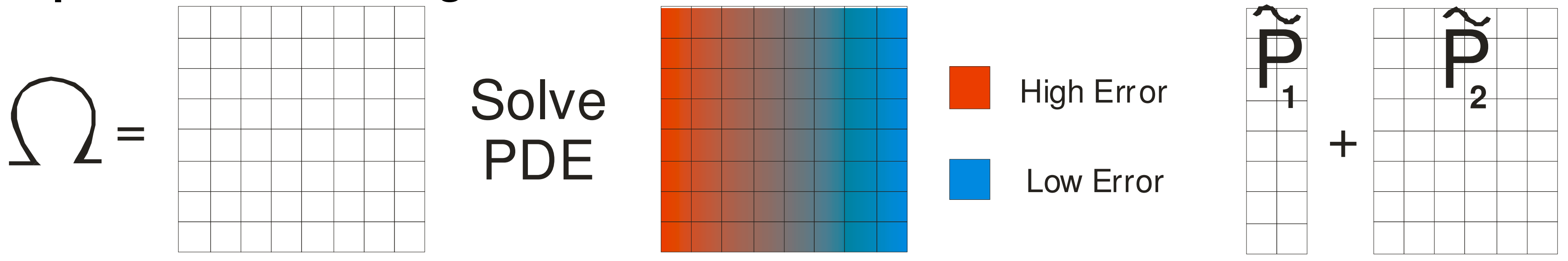
See Bank and Xu [8]

Bank-Holst Paradigm

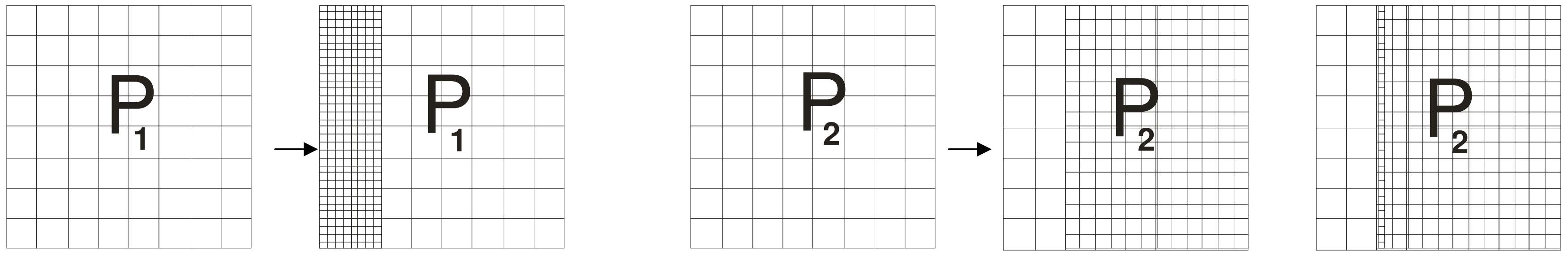
- **Step 1: Load Balancing**. We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements or gridpoints.
- **Step 2: Adaptive Meshing**. 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]

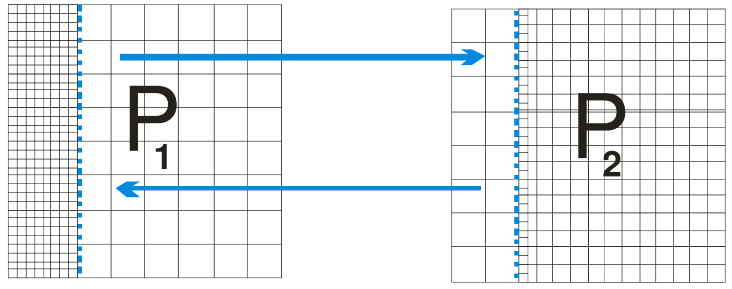
Step 1: Load Balancing



Step 2: Adaptive Meshing



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

[1] A. Toselli and O. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, Springer (2005) pp. 217-230

[2] A.K.Aziz and I. Babuska, *Part I, survey lectures on the mathematical foundations of the finite element method*, in The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, Academic Press, New York, (1972), pp. 1-362

[3] C. Deotte and M. Holst, *Finite Element Project*, (2007)

[4] A. Toselli and O. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, Springer (2005) pp. 35-54, 131-192

[5] C. Deotte and S. Baden, *Domain Decomposition Project*, (2008)

[6] T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, Springer (2008) pp. 270-275

[7] C. Deotte and R. Bank, *Recursive Spectral Bisection Project*, (2009)

[8] R. Bank and J. Xu, *Asymptotically exact a posteriori error estimators, part II: General unstructured grids*, SIAM J. Numerical Analysis (2003)

[9] R Bank and M Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. on Scientific Computing, 22 (2000)

[10] R Bank and J Ovall. *Dual Functions for A Parallel Adaptive Method*, SIAM J. on Scientific Computing, (2000).

Thank You.