A Nested Dissection Parallel Direct Solver for Simulations of 3D DC/AC Resistivity Measurements

Maciej Paszyński[1,2]
David Pardo[2], Carlos Torres-Verdín[2]

(1) Department of Computer Science,
AGH University of Science and Technology, Kraków, Poland
e-mail: paszynsk@agh.edu.pl
home.agh.edu.pl/~paszynsk

(2) Department of Petroleum and Geosystem Engineering,
The University of Texas at Austin

OUTLINE

- Formulation of the resistivity measurement simulation model problem
- Sequential algorithm
- Parallel algorithm
- Scalability of the parallel solver
- Parallel solver details
- Conclusions and future work (inversions)

FORMULATION OF THE MODEL PROBLEM

$$\begin{cases} \nabla \times \mathbf{H} = \left(\boldsymbol{\sigma} + j\,\omega\,\boldsymbol{\varepsilon}\right)\mathbf{E} + \mathbf{J}^{\mathbf{imp}} & \text{(Ampere's Law)} \\ \nabla \times \mathbf{E} = -j\,\omega\,\boldsymbol{\mu}\,\mathbf{H} - \mathbf{M}^{\mathbf{imp}} & \text{(Faraday's Law)} \\ \nabla \cdot \left(\boldsymbol{\varepsilon}\,\mathbf{E}\right) = \rho & \text{(Gauss' Law of Electicity)} \\ \nabla \cdot \left(\boldsymbol{\mu}\,\mathbf{H}\right) = 0 & \text{(Gauss' Law of Magnetism)} \end{cases}$$

$\mathbf{H}$	magnetic field	ω	angular frequency
$\mathbf{E}$	electric field	ρ	electric charge distribution
$\boldsymbol{\varepsilon}$	diaelectric permittivity	$\mathbf{J}^{\mathbf{imp}}$	impressed electric current
$\boldsymbol{\mu}$	magnetic permeability	$\mathbf{M}^{\mathbf{imp}}$	impressed magnetic current
$\boldsymbol{\sigma}$	electrical conductivity		

FORMULATION OF THE MODEL PROBLEM

$$\begin{cases} \nabla \times \mathbf{H} = \boldsymbol{\sigma}\,\mathbf{E} + \mathbf{J}^{\mathbf{imp}} & \text{(Ampere's Law)} \\ \nabla \times \mathbf{E} = 0 & \text{(Faraday's Law)} \\ \nabla \cdot \left(\boldsymbol{\varepsilon}\,\mathbf{E}\right) = \rho & \text{(Gauss' Law of Electicity)} \\ \nabla \cdot \left(\boldsymbol{\mu}\,\mathbf{H}\right) = 0 & \text{(Gauss' Law of Magnetism)} \end{cases}$$

$\mathbf{H}$ magnetic field

$\mathbf{E}$ electric field

$\boldsymbol{\varepsilon}$ diaelectric permittivity

$\boldsymbol{\mu}$ magnetic permeability

$\omega = 0$ DC formulation

ρ electric charge distribution

$\mathbf{J}^{\mathbf{imp}}$ impressed electric current

FORMULATION OF THE MODEL PROBLEM

Taking the curl of the Ampere's law and utilizing the Gauss' Electric law we obtain the **conductive media equation**

$$\nabla \cdot \left(\boldsymbol{\sigma} \nabla u \right) = -\nabla \cdot \mathbf{J}$$

where u scalar potential such that $\mathbf{E} = -\nabla u$

Variational formulation

Find $u \in u_D + H_D^1(\Omega)$ such that

$$\left\langle \nabla v, \boldsymbol{\sigma} \nabla u \right\rangle_{L^2(\Omega)} = \left\langle v, \nabla \cdot \mathbf{J} \right\rangle_{L^2(\Omega)} + \left\langle v, h \right\rangle_{L^2(\Gamma_N)} \quad \forall v \in H_D^1(\Omega)$$

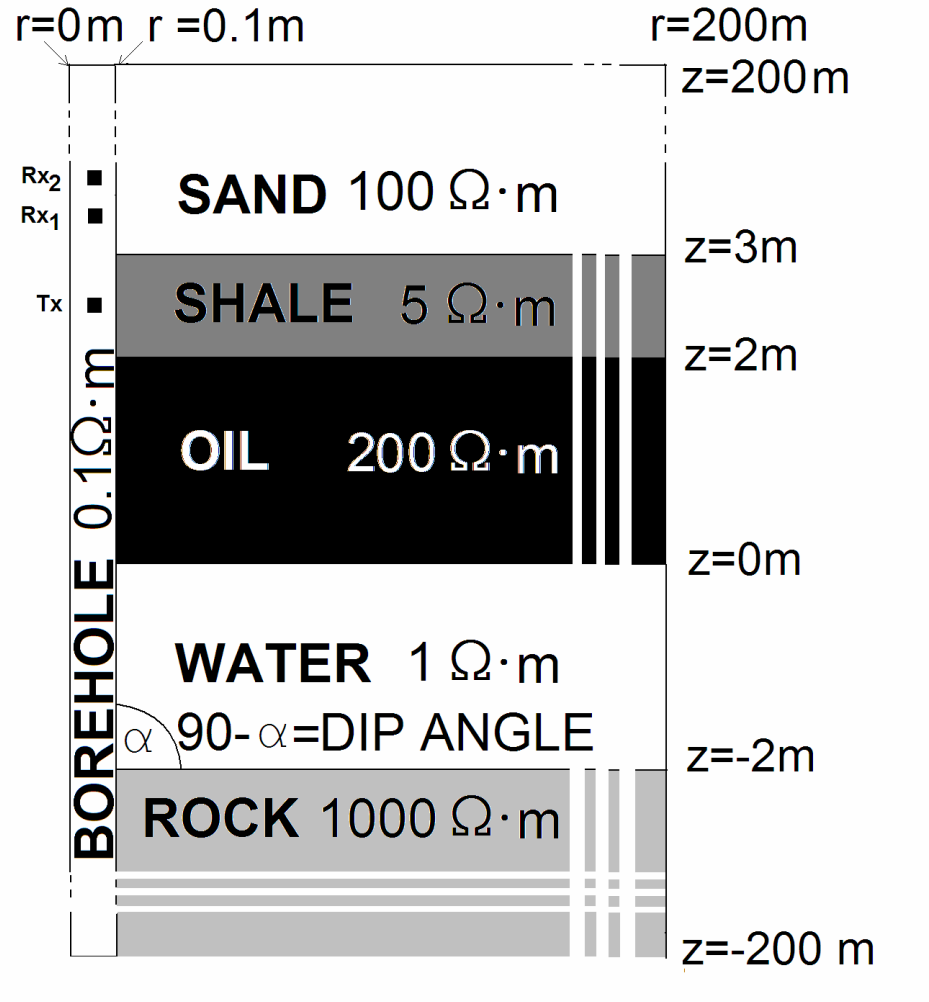
u_D lift of essential Dirichlet b.c.

$h = \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \nabla u$ prescribed flux on Γ_N

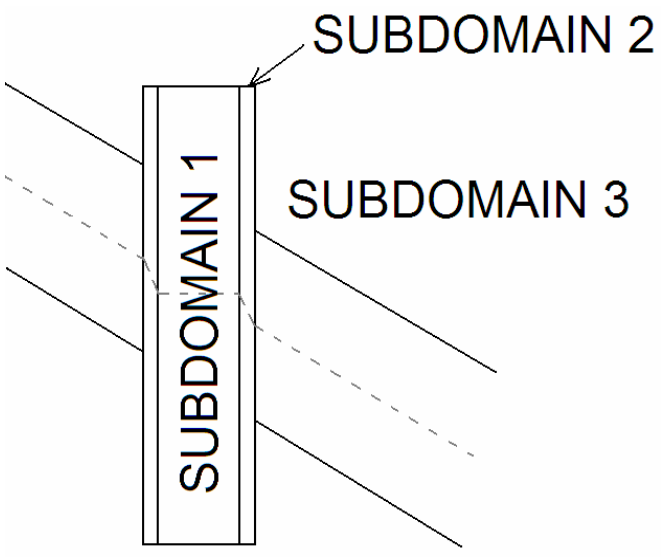
$$H_D^1(\Omega) = \left\{ u \in H^1(\Omega) : u \big|_{\Gamma_D} = 0 \right\}$$

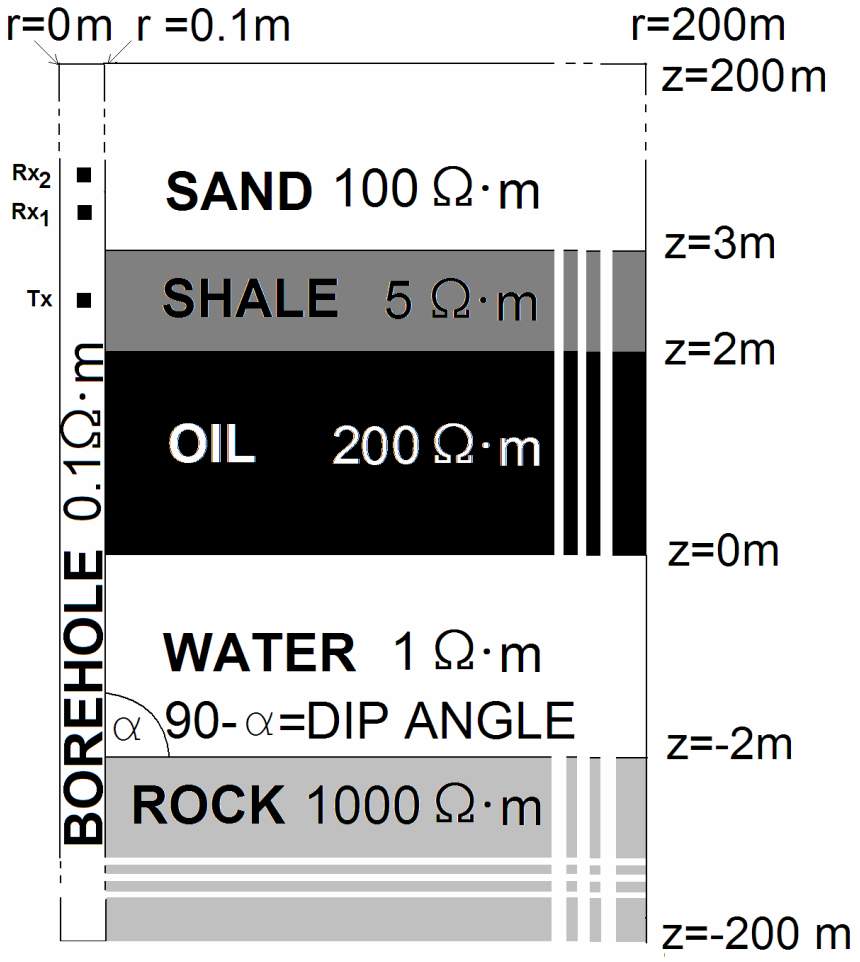
FORMULATION OF THE MODEL PROBLEM

- 1 transmitter
- 2 receiver electrodes
- transmitter electrode modeled by the impressed electric current $\mathbf{J}^{imp}$
- five different layers in the formation with resistivities 100 Ω·m (sand) 5 Ω·m (shale) 200 Ω·m (oil) 1 Ω·m (water) 1000 Ω·m (rock)
- borehole with resistivity 0.1 Ω·m
- 0 Dirichlet b.c. on the external boundary of the domain
- 0 Neumann b.c. on the axis of symmetry



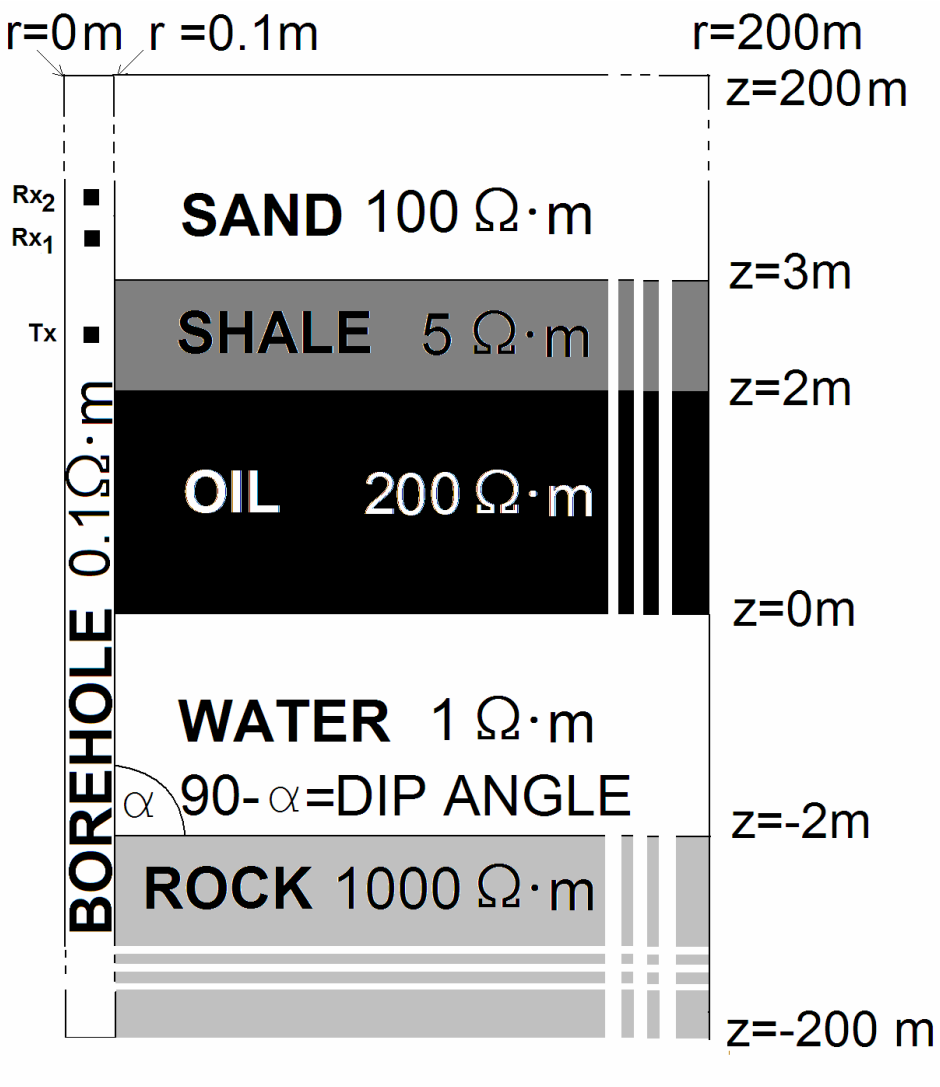
FORMULATION OF THE MODEL PROBLEM

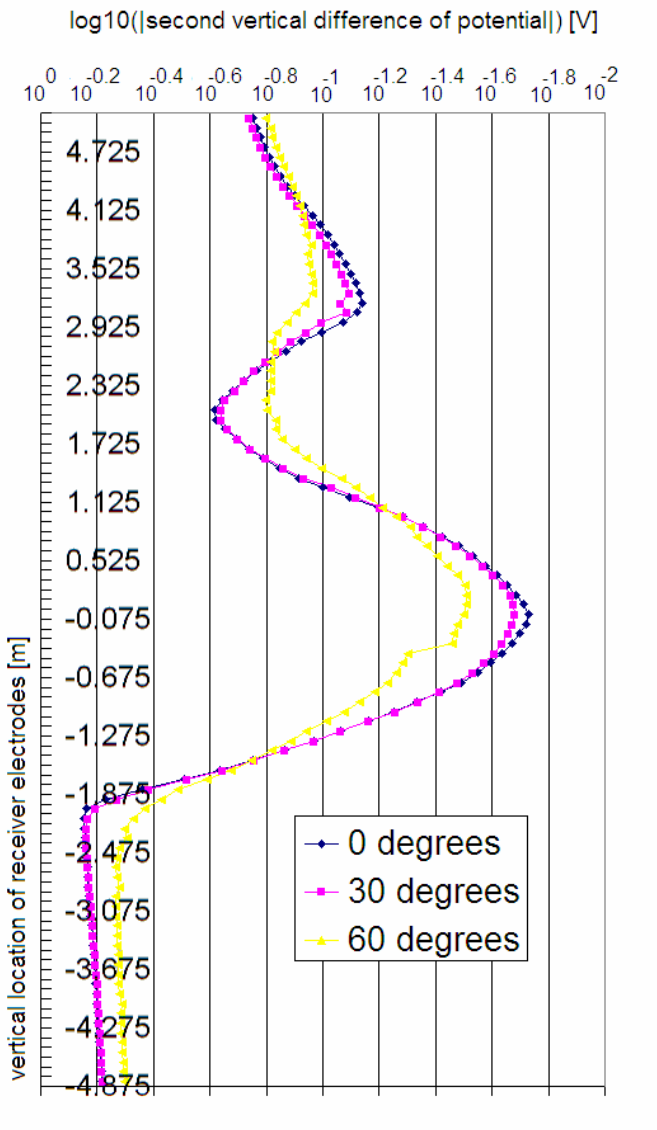




We utilize a 2D self-adaptive goal-oriented hp-adaptive strategy combined with a Fourier series expansion in a non-orthogonal system of coordinates

FORMULATION OF THE MODEL PROBLEM





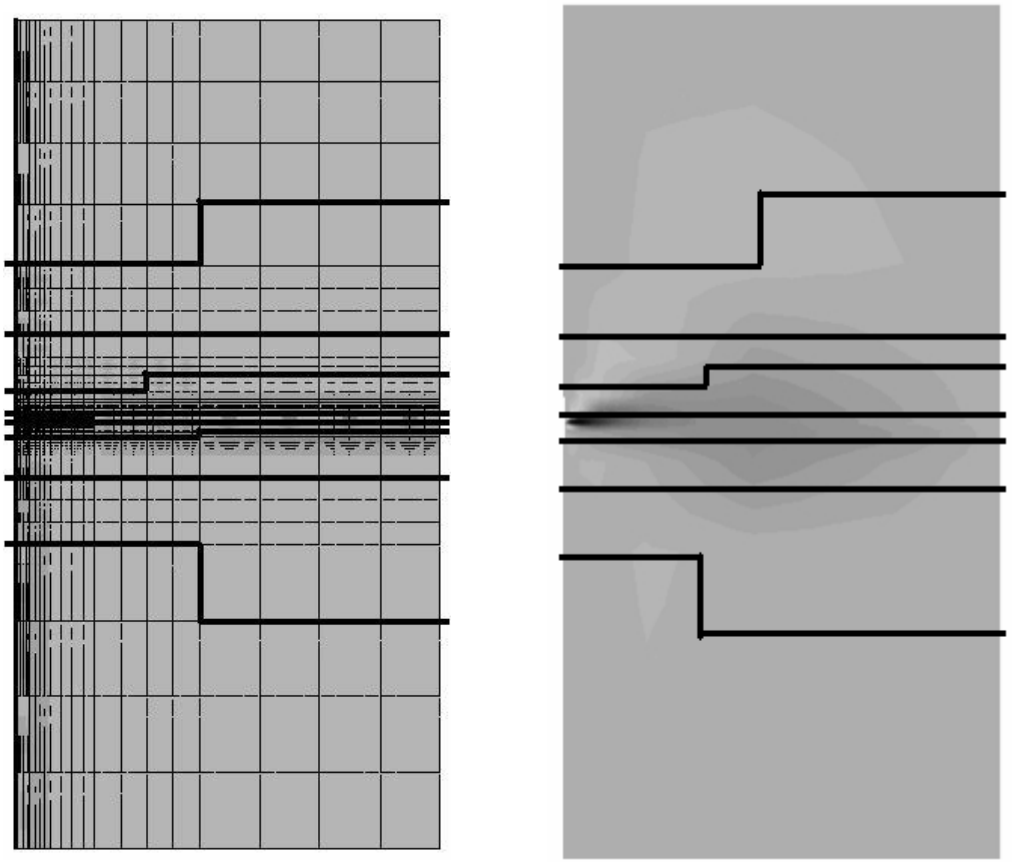
SEQUENTIAL ALGORITHM

```
Loop over electrode locations
  Iterations of the goal-oriented self-adaptive hp FEM
    Solve the problem over the coarse mesh
    Solve the problem over the fine mesh
    Compute relative error estimations over finite elements
    If maximum relative error < required accuracy then exit
    Make decisions about optimal refinements
    Perform optimal refinements
  End
  Store second vertical difference of potential
    at receiver electrodes
End
```

PARALLEL ALGORITHM

Each processor is assigned to a set of finite elements.
Each node from the interface is assigned to multiple processor owners.

Local copy of the entire data structure is stored on every processor.
But each processor performs computations only on assigned set of finite elements.
Only local solution degrees of freedom are stored to save memory.



PARALLEL ALGORITHM

Redistribute the computational mesh between processors

Loop over electrode locations

  Iterations of the goal-oriented self-adaptive hp FEM

    Solve the coarse mesh problem **by parallel solver**

    Solve the fine mesh problem **by parallel solver**

    Compute **relative error** estimations over finite elements

    Compute global maximum relative error (mpi_allreduce)

    If maximum **relative error** < required accuracy **then** exit

    Make **local** decisions about optimal refinements

    Broadcast required refinements

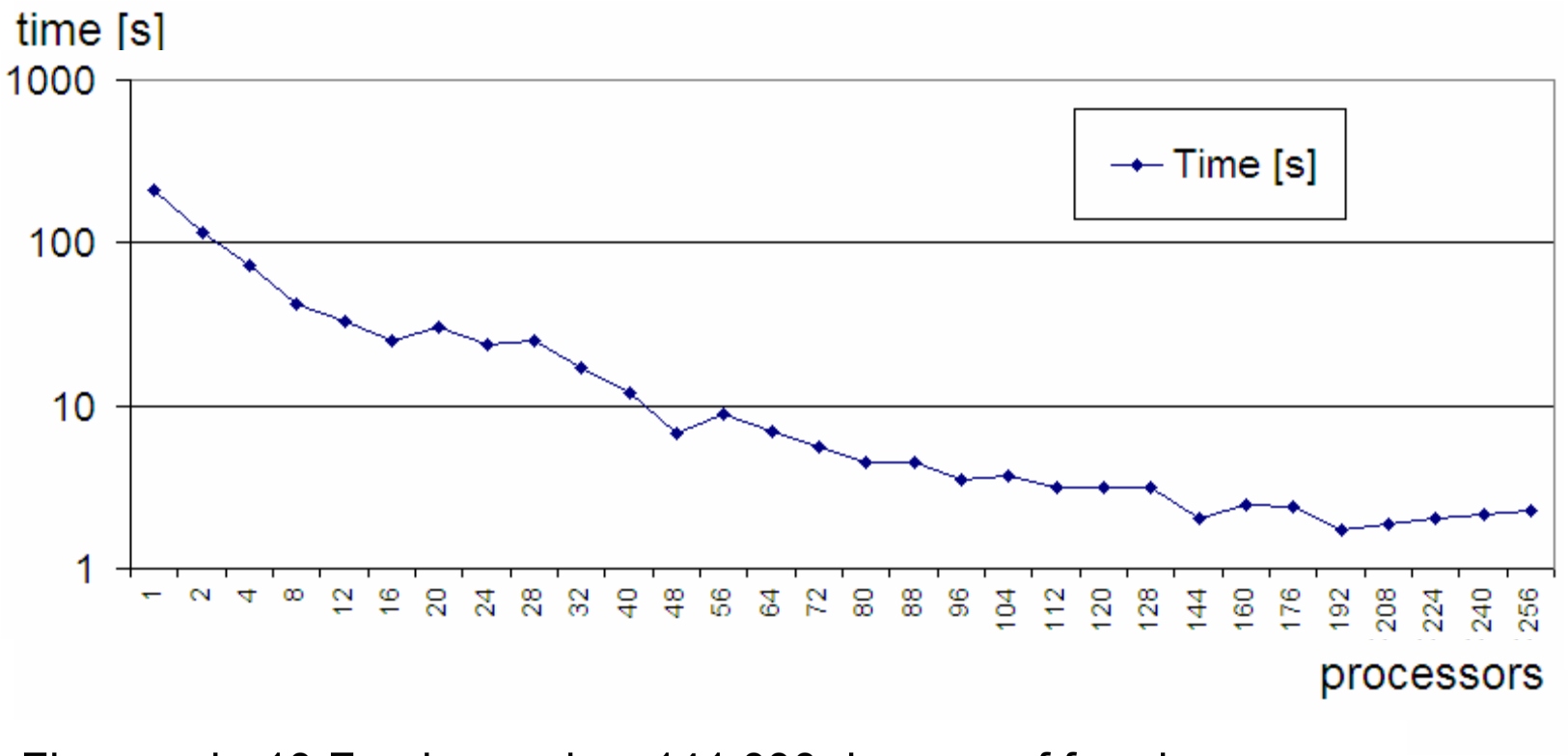
    Perform optimal refinements **on the whole local mesh**

  End

  Store second vertical difference of potential
    at receiver electrodes

    (requires communication to gather distributed solution)

SCALABILITY OF THE PARALLEL SOLVER

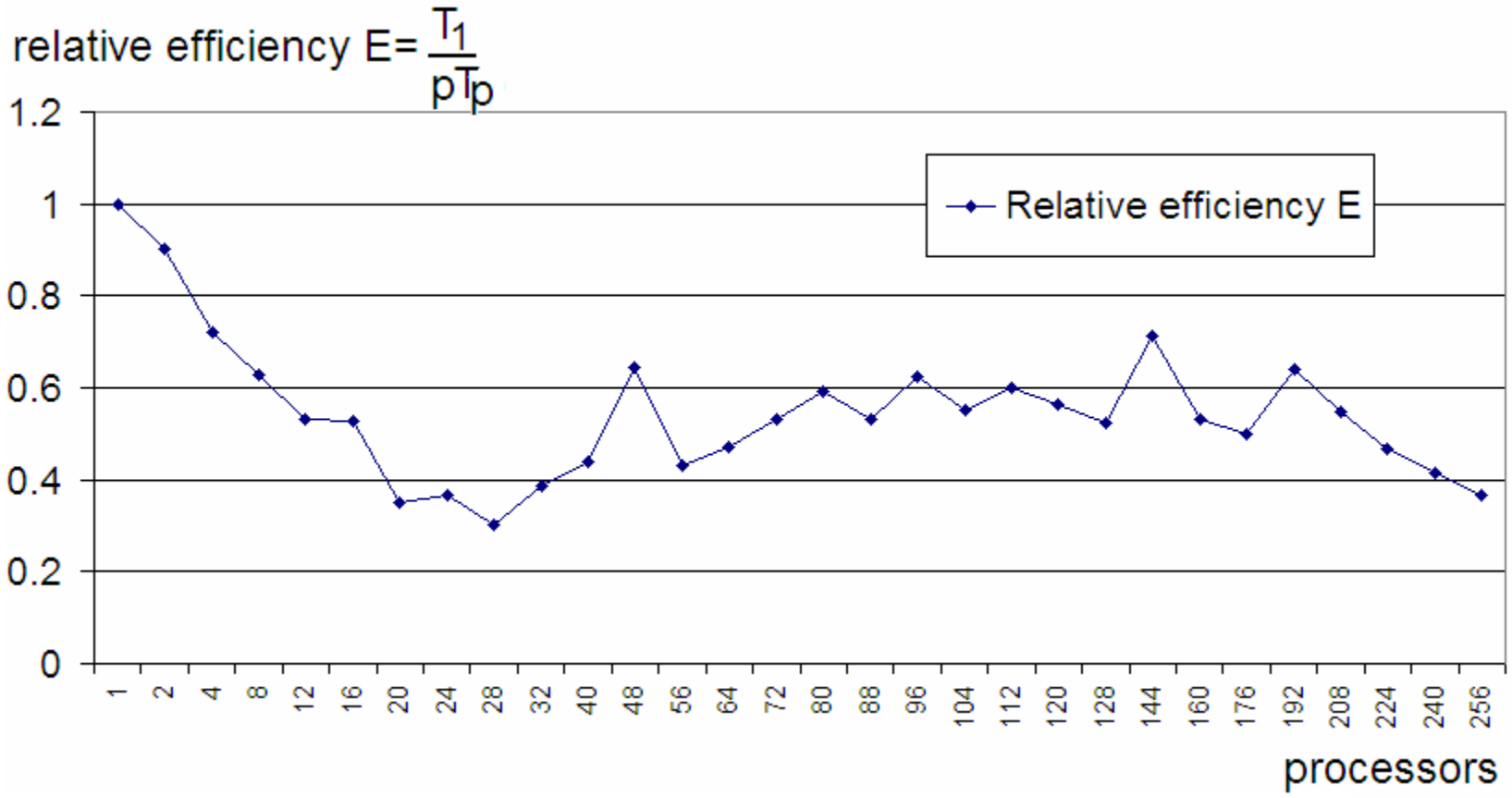


Fine mesh, 10 Fourier modes, 141 000 degrees of freedom

211 seconds on 1 processor (per single electrode location)

1.75 seconds on 192 processors (per single electrode location)

SCALABILITY OF THE PARALLEL SOLVER

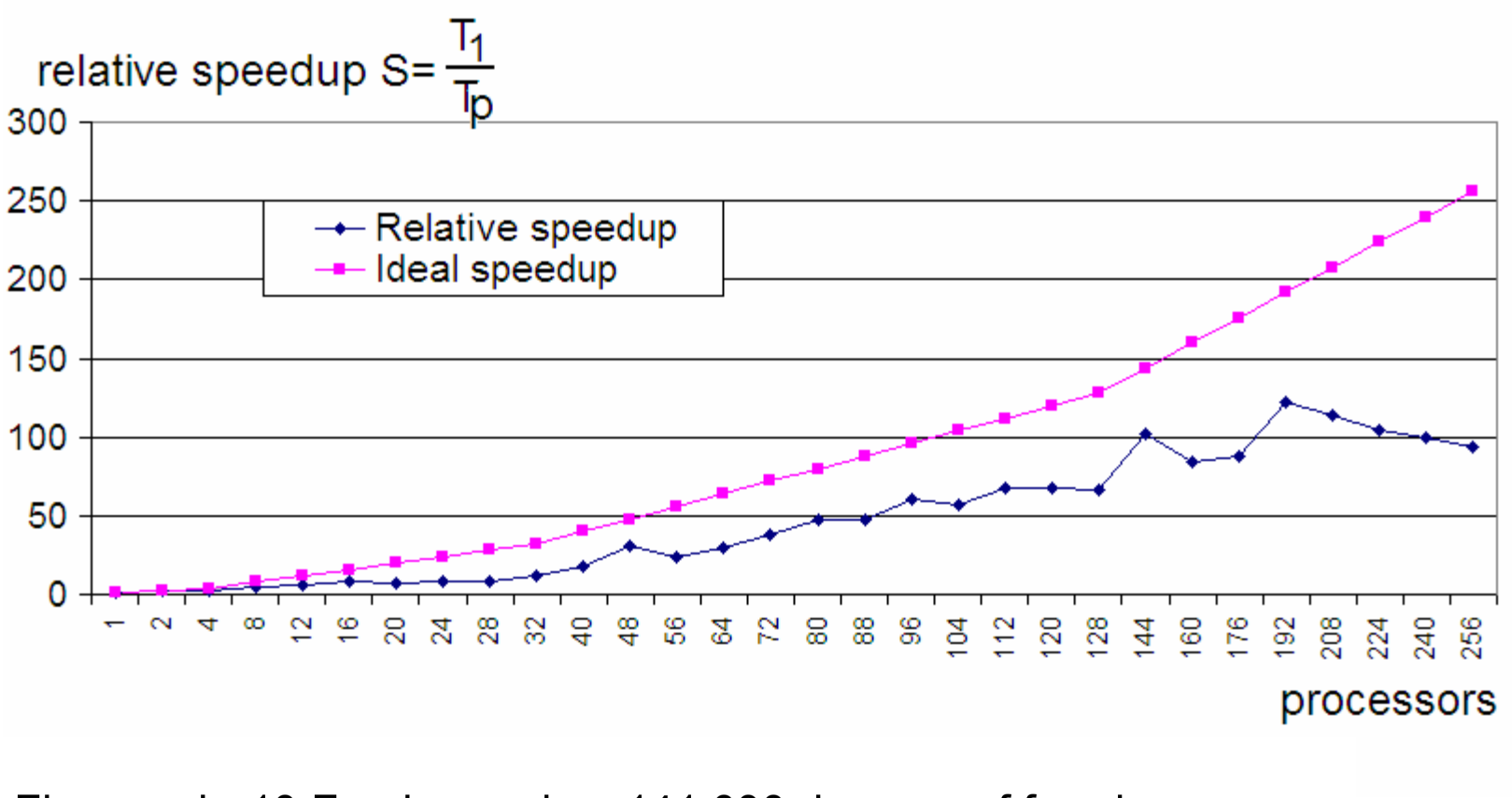


Fine mesh, 10 Fourier modes, 141 000 degrees of freedom

211 seconds on 1 processor (per single electrode location)

1.75 seconds on 192 processors (per single electrode location)

SCALABILITY OF THE PARALLEL SOLVER

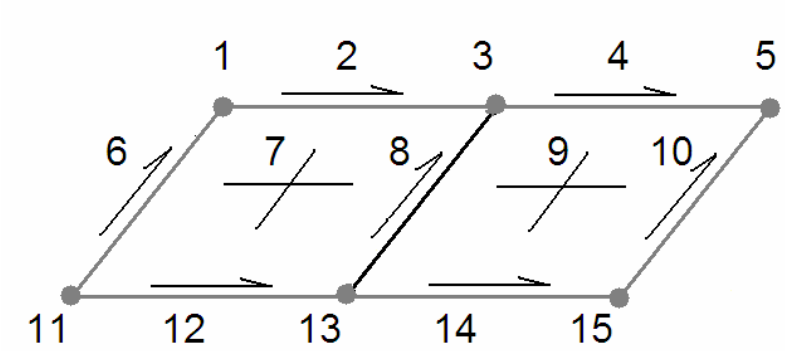


Fine mesh, 10 Fourier modes, 141 000 degrees of freedom

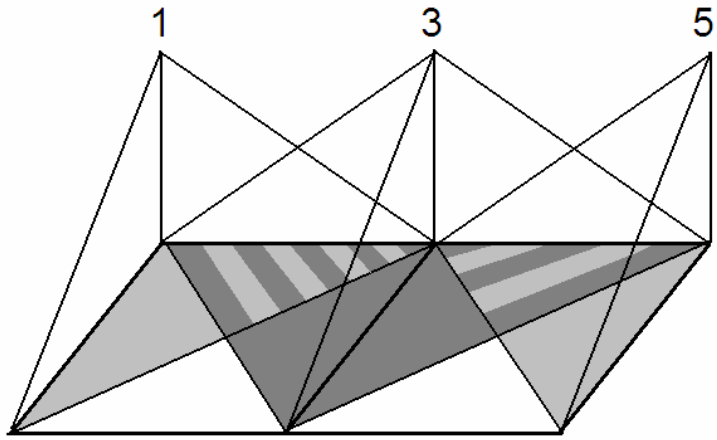
211 seconds on 1 processor (per single electrode location)

1.75 seconds on 192 processors (per single electrode location)

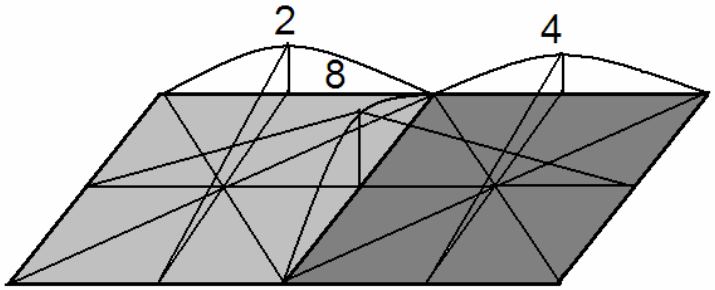
PARALLEL SOLVER DETAILS



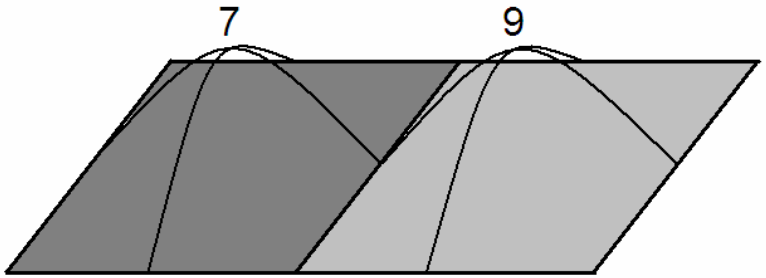
(a)



(b)



(c)

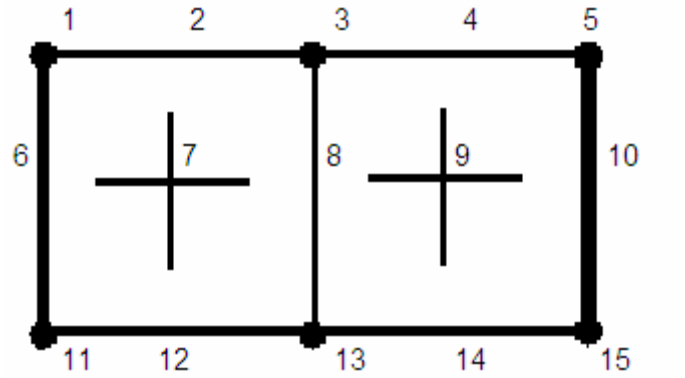


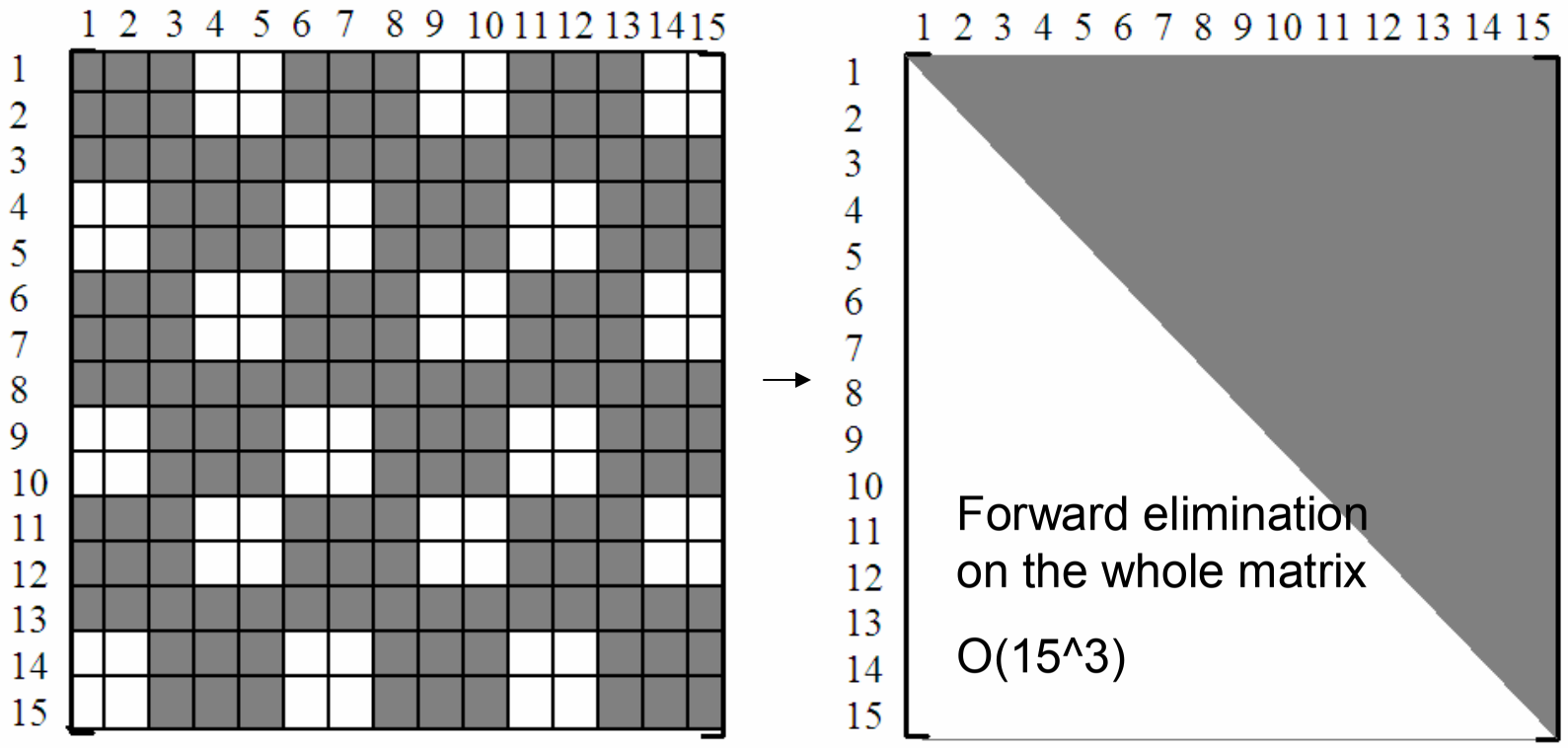
(d)

$$b\left(e_{hp}^{i}, e_{hp}^{j}\right) = \left\langle \nabla e_{hp}^{i}, \boldsymbol{\sigma} \nabla e_{hp}^{j} \right\rangle_{L^2(\Omega)}$$

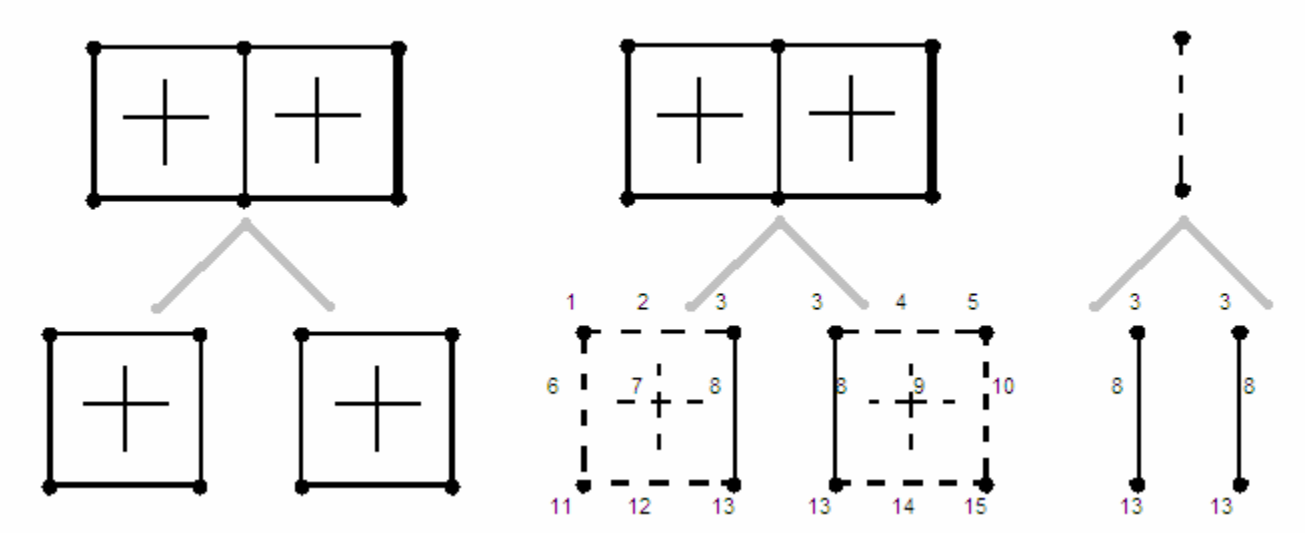
$$l\left(e_{hp}^{j}\right) = \left\langle e_{hp}^{j}, \nabla \cdot \mathbf{J} \right\rangle_{L^2(\Omega)} + \left\langle e_{hp}^{j}, h \right\rangle_{L^2(\Gamma_N)}$$

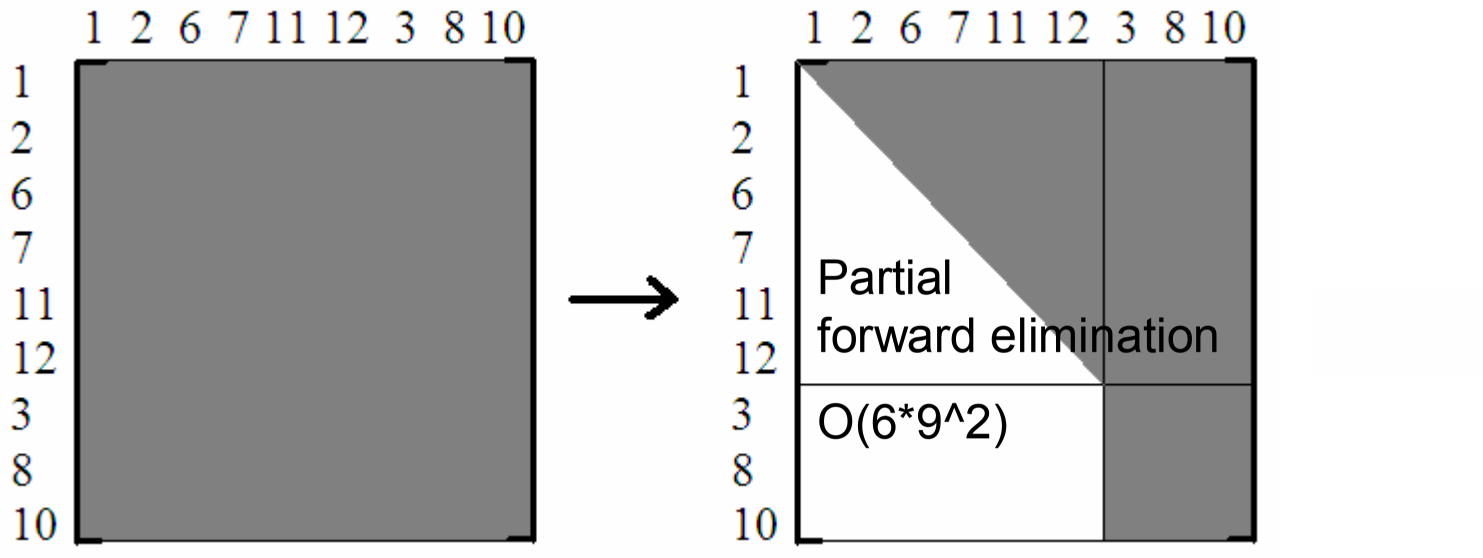
PARALLEL SOLVER DETAILS



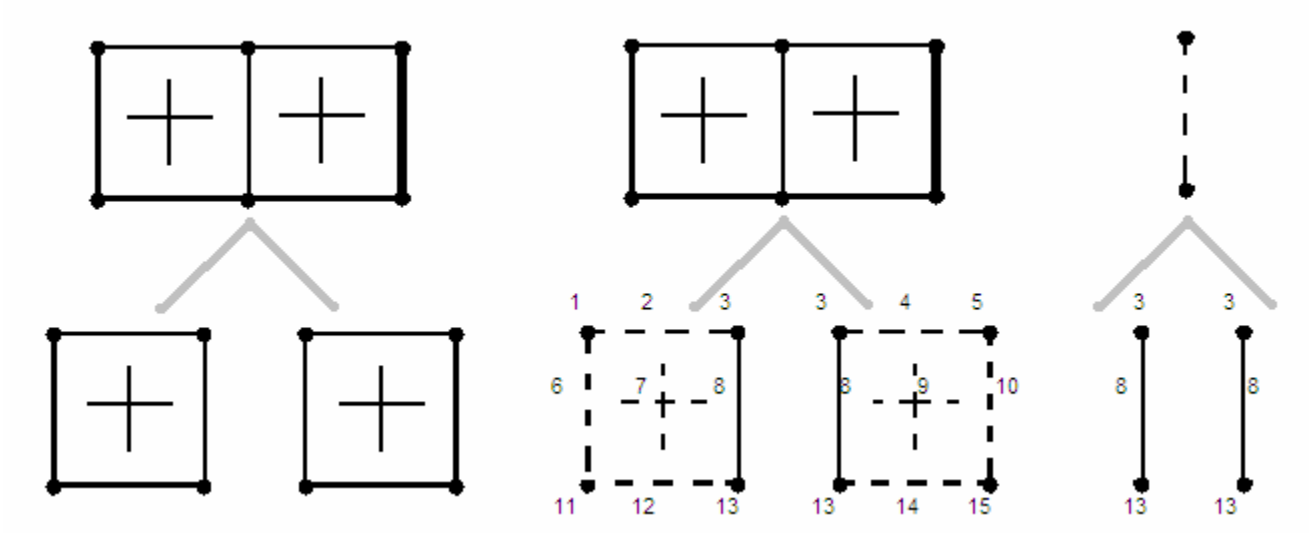


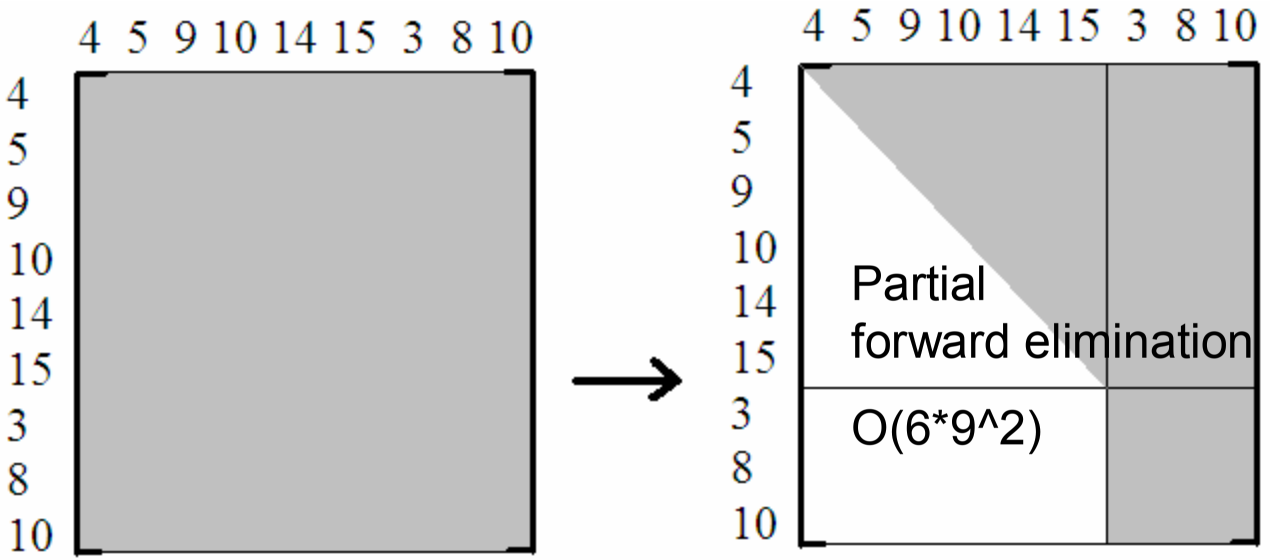
PARALLEL SOLVER DETAILS



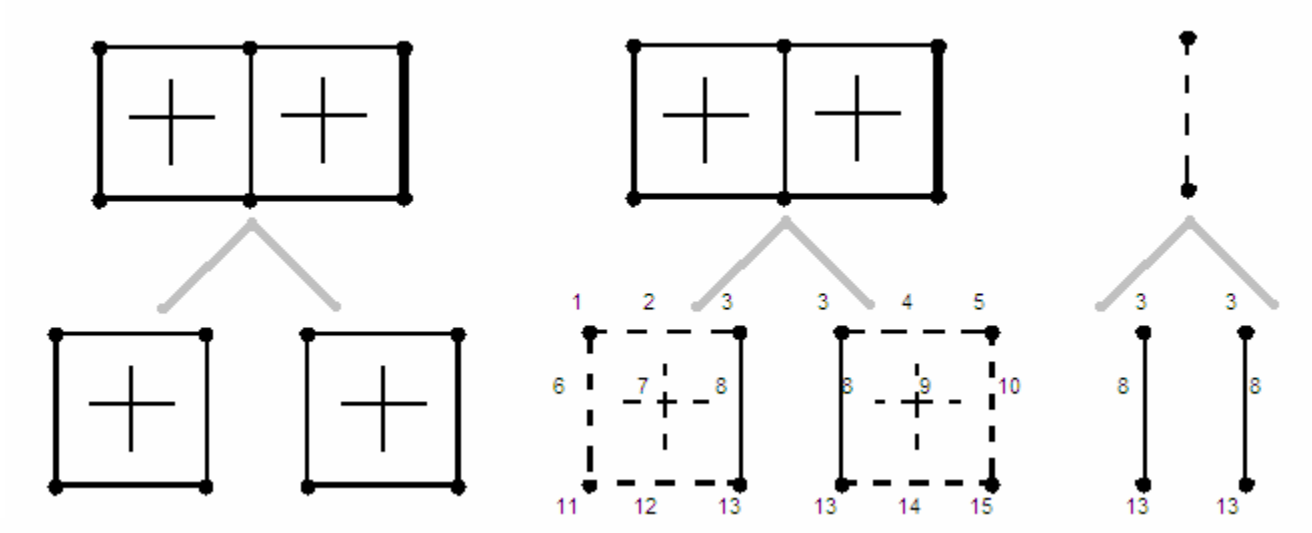


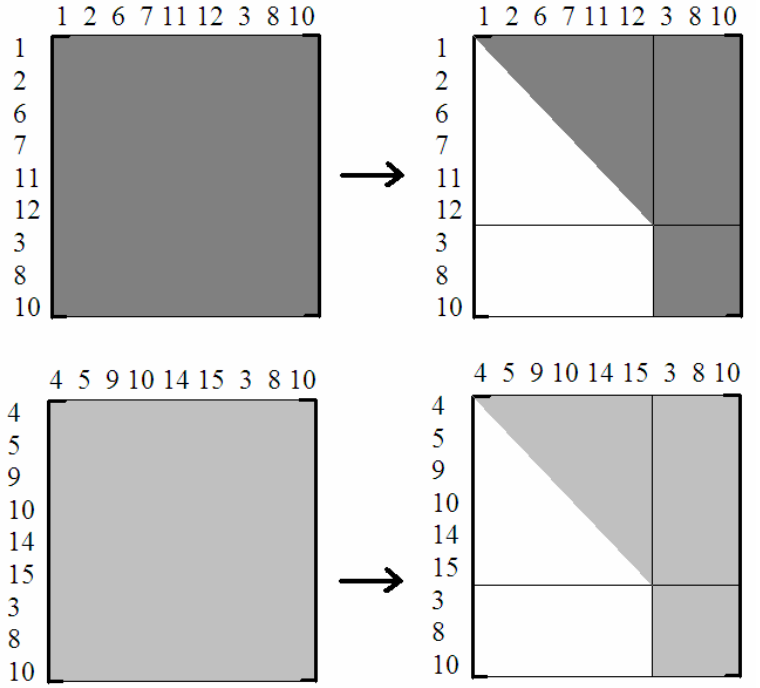
PARALLEL SOLVER DETAILS





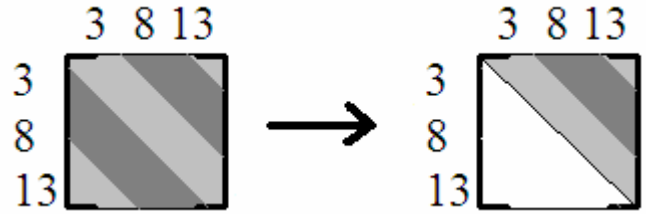
PARALLEL SOLVER DETAILS





Full forward elimination

O(3^3)

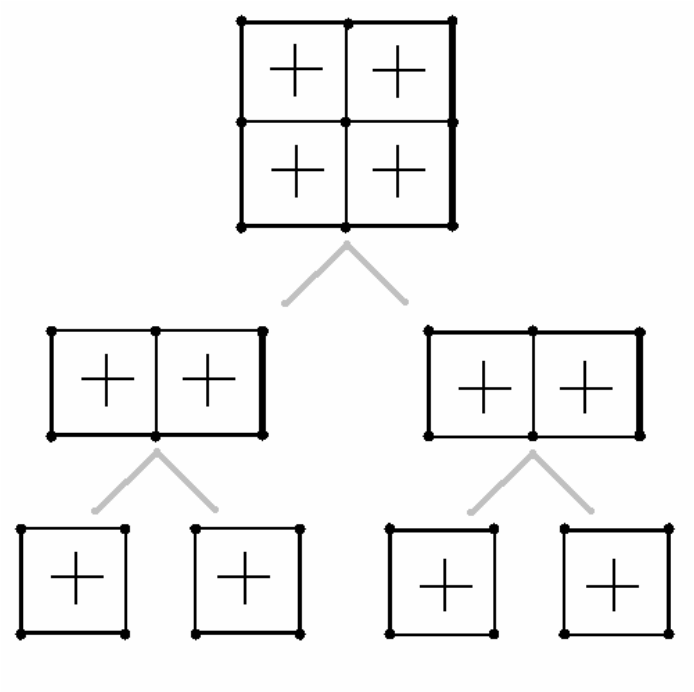


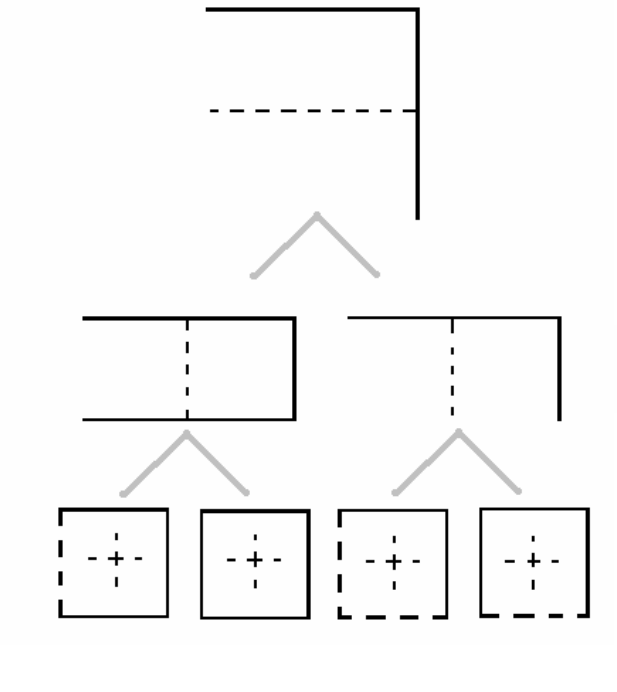
PARALLEL SOLVER DETAILS

Forward elimination over the whole matrix
15^3 = 3375 operations

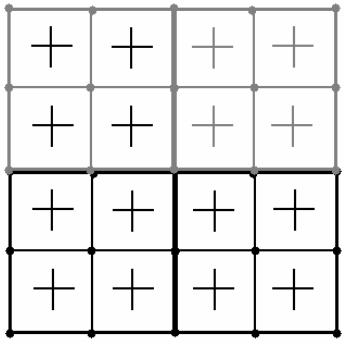
Partial forward eliminations over elements
6*9^2 + 6*9^2 + 3^3 = 486 + 486 + 27 = 999

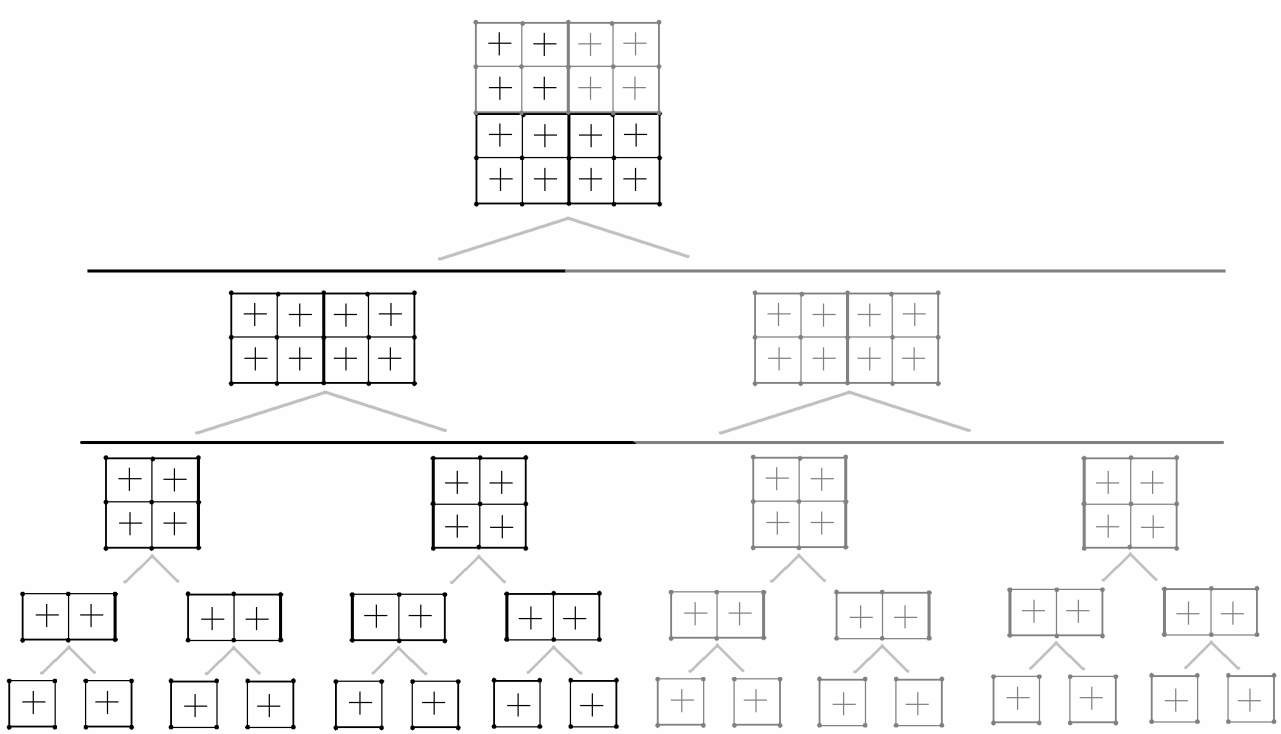
The idea is generalized into the whole domain





PARALLEL SOLVER DETAILS



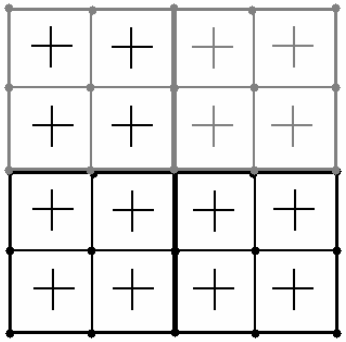


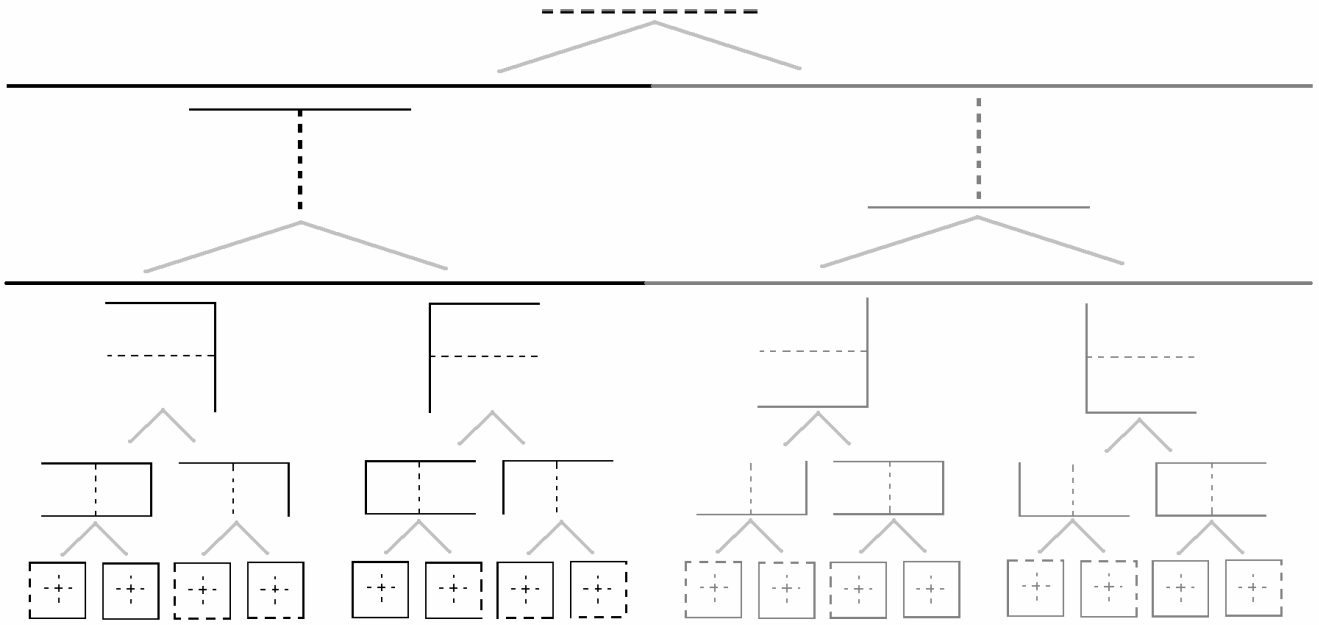
Level of subdomains

Level of initial mesh elements

Level of refinement trees

PARALLEL SOLVER DETAILS





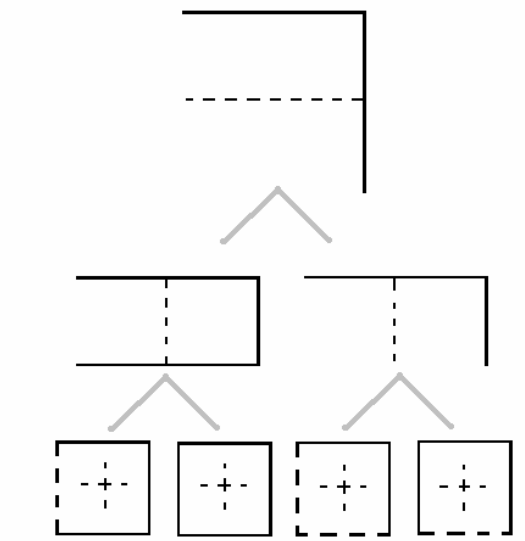
Level of subdomains

Level of initial mesh elements

Level of refinement trees

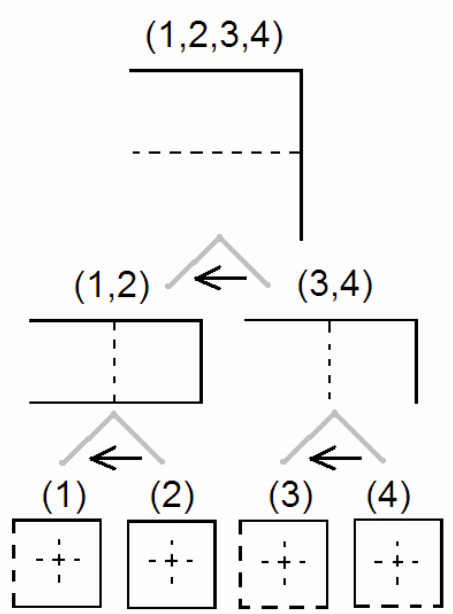
SOLVER RECURSIVE ALGORITHM

```
matrix function recursive_solver(tree_node)
if tree_node has no son nodes then
  eliminate leaf element stiffness matrix
    internal nodes
  return Schur complement sub-matrix
else if tree_node has son nodes then
  do for each son
     son_matrix = recursive_solver(tree_node_son)
     merge son_matrix into new_matrix
  enddo
  decide which unknowns of new_matrix can be eliminated
  perform partial forward elimination on new_matrix
  return Schur complement sub-matrix
endif
```



PARALLEL SOLVER RECURSIVE ALGORITHM

```
matrix function recursive_solver(tree_node)
if tree_node has no son nodes then
  eliminate leaf element stiffness matrix
    internal nodes
  return Schur complement sub-matrix
else if tree_node has son nodes then
  do for each son
     if son node is assign to current processor
       son_matrix = recursive_solver(tree_node_son)
     if current processor k is the first processor in procesors group
        RECEIVE son_matrix from processor 2k+1
        merge son_matrix into new_matrix
     else SEND son_matrix to the first processor in processors group
  enddo
  decide which unknowns of new_matrix can be eliminated
  perform partial forward elimination on new_matrix
  return Schur complement sub-matrix
```



CONCLUSIONS AND FUTURE WORK

- A new parallel direct solver has been developed.
- The solver recursively utilizes the Schur complement pattern to eliminate fully agregated degrees of freedom on every level of the elimination tree.
- The solver recursively travels the refinement trees, the tree of initial mesh elements as well as the tree of sub-domains.
- The parallel version of the solver provides over 60% relative efficiency on 200 processors.
- The solver is able to reduce the solution time of 141000 degrees of freedom problem from 211 seconds to 1.75 seconds on 192 processors.
- The single logging position can be solved within 2 seconds on 200 processors.
- The future work will involve application of the parallel solver to the inverse problem modeling.