PARALLEL FULLY AUTOMATIC HP-ADAPTIVE CODES FOR ACOUSTICS AND ELECTROMAGNETICS

Principal investigator:

Leszek Demkowicz (ICES, UT Austin)

Team:

Maciej Paszynski (ICES, UT Austin)

Jason Kurtz (ICES, UT Austin)

Collaborators:

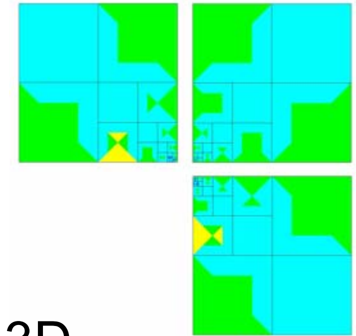
Waldemar Rachowicz (Cracow University of Technology)

Timothy Walsh (Sandia National Laboratories)

David Pardo (ICES, UT Austin)

Dong Xue (ICES, UT Austin)

Kent Milfeld (TACC, UT Austin)



INTRODUCTION

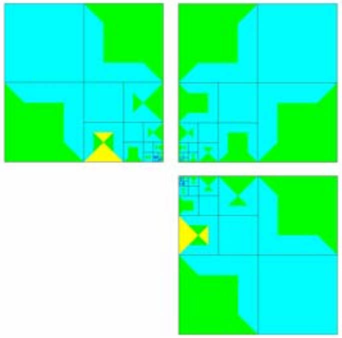
We are working on:

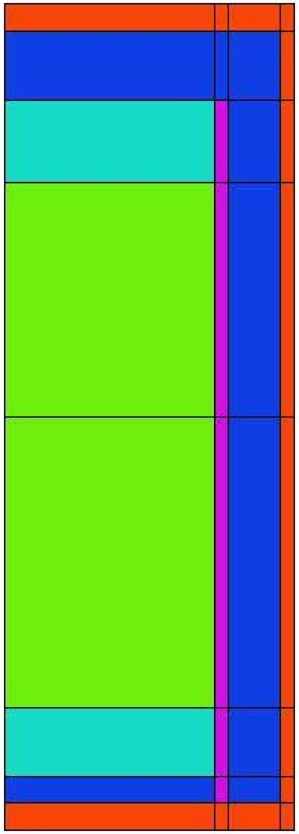
- Parallel fully automatic hp-adaptive finite element 2D and 3D codes
- The code automatically produces a sequence of optimal meshes with global exponential convergence rate
- Currently, we have running 2D version of the code for the Laplace equation
- All stages of the code are fully parallel
- The code will be soon extended to solve 3D Helmholtz and time harmonic Maxwell equations

The work is driven by 3 Challenging Applications:

- Simulation of EM waves in the human head
- Calculation of the Radar Cross-sections (3D scattering problems)
- Simulation of Logging While Drilling EM measuring devices

ORTHOTROPIC HEAT EQUATION

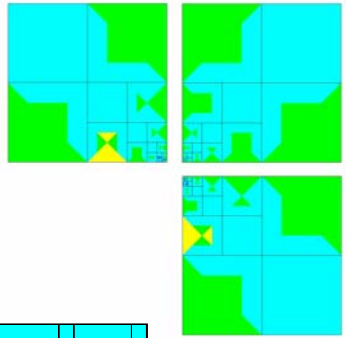




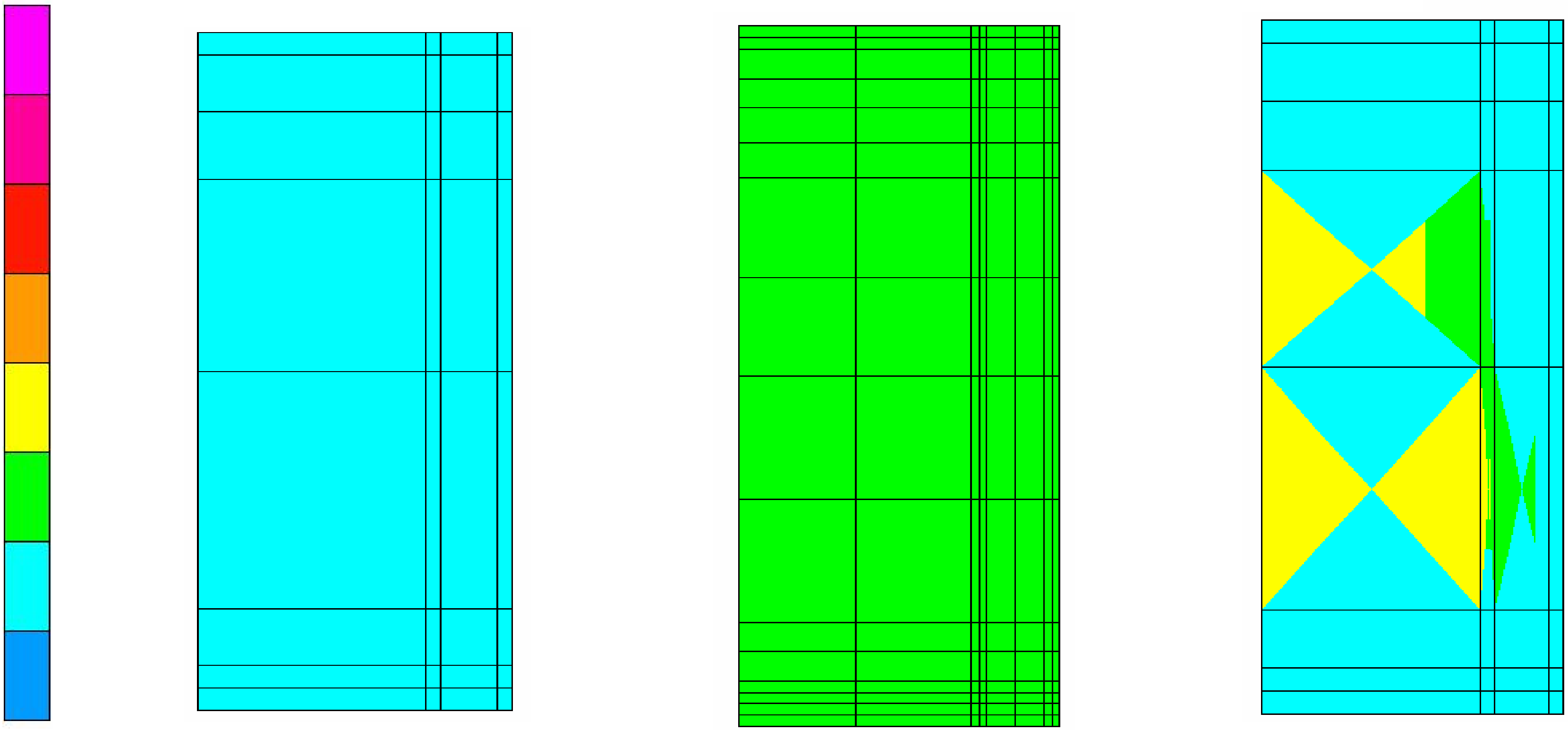
$$\nabla\left(K\nabla u\right) = f$$

$$K = K^{(k)} = \begin{bmatrix} K_x^{(k)} & 0 \\ 0 & K_y^{(k)} \end{bmatrix}$$

- 5 materials, some orthotropic some not
- large $O(10^5)$ jumps in material data generate singularities
- requires anisotropic refinements



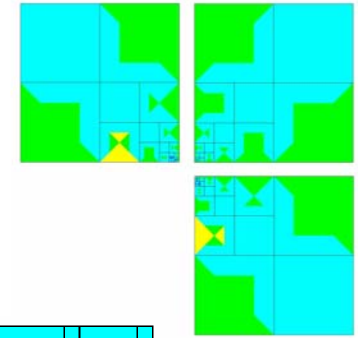
COARSE MESH, FINE MESH AND OPTIMAL MESH



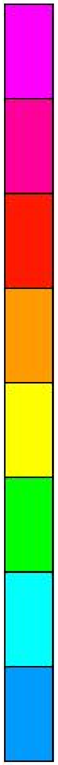
Initial mesh = coarse mesh
for the 1st step
of the iteration

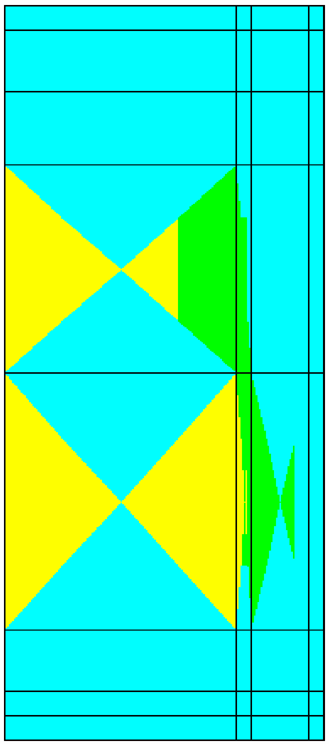
Fine mesh

Optimal mesh

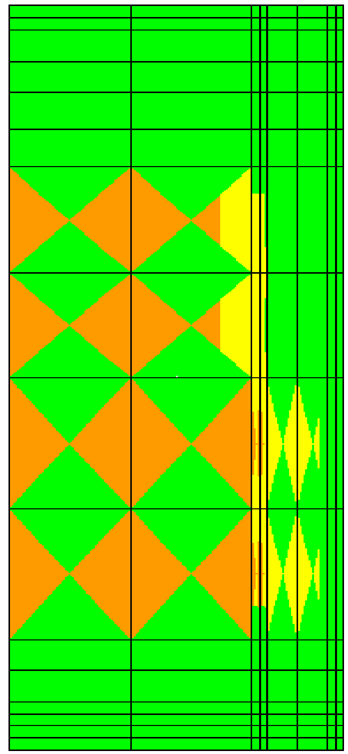


COARSE MESH, FINE MESH AND OPTIMAL MESH

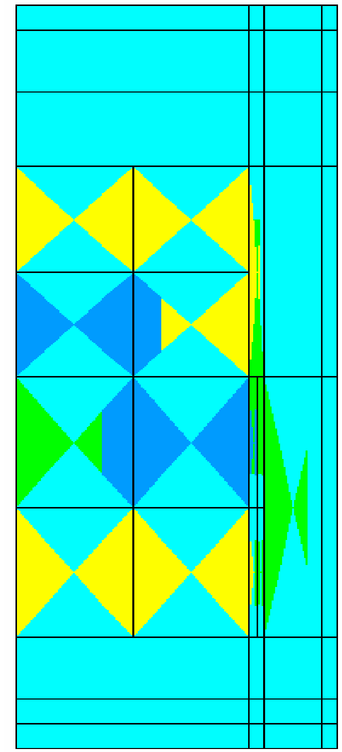




Optimal mesh = coarse mesh for the 2nd step of the iteration



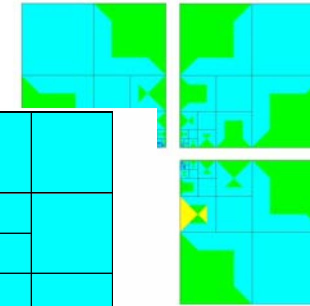
Fine mesh



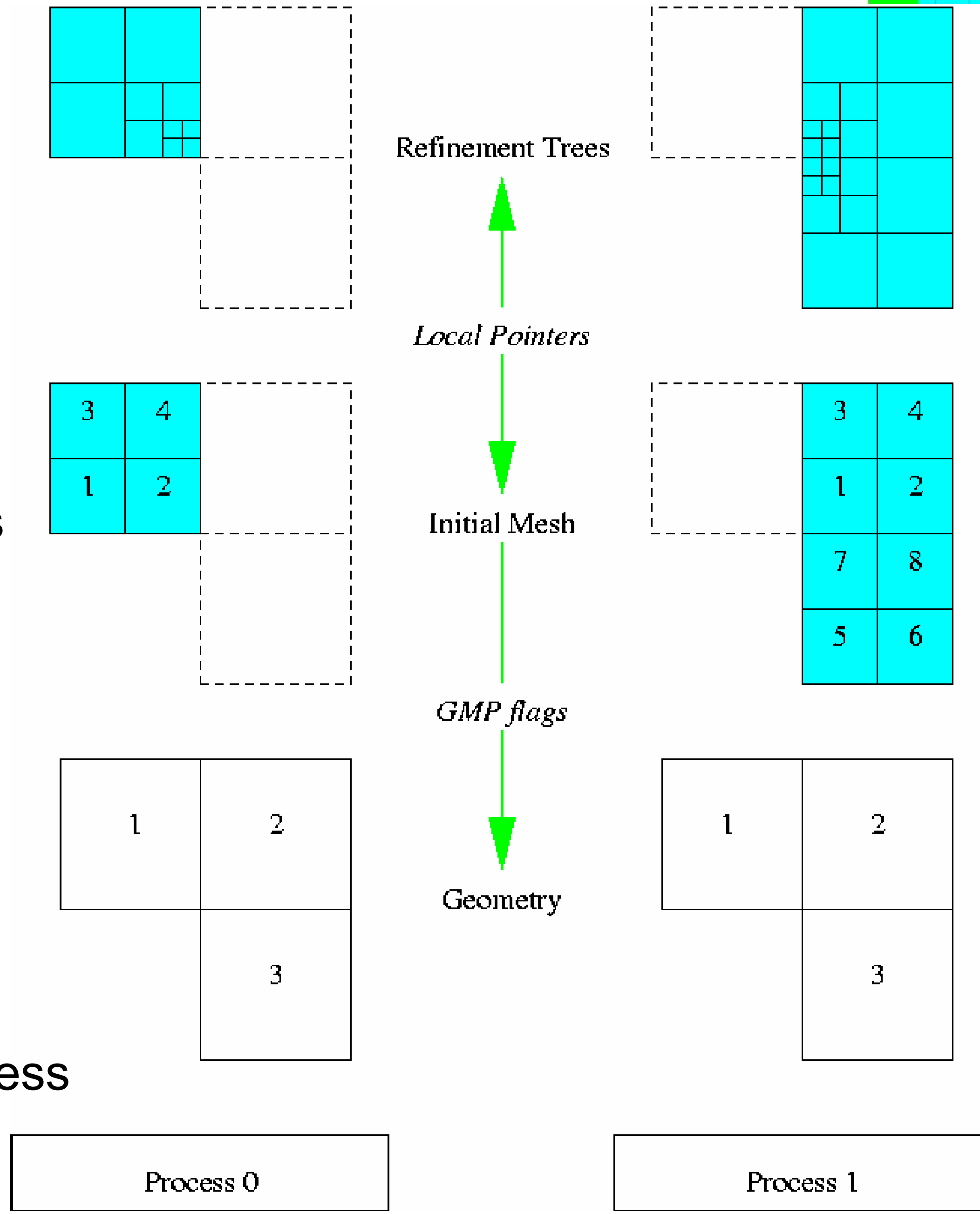
Optimal mesh

PARALLEL DATA STRUCTURES

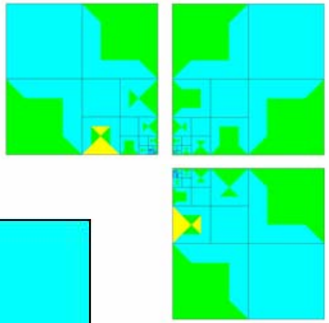
PARALLEL DATA STRUCTURES

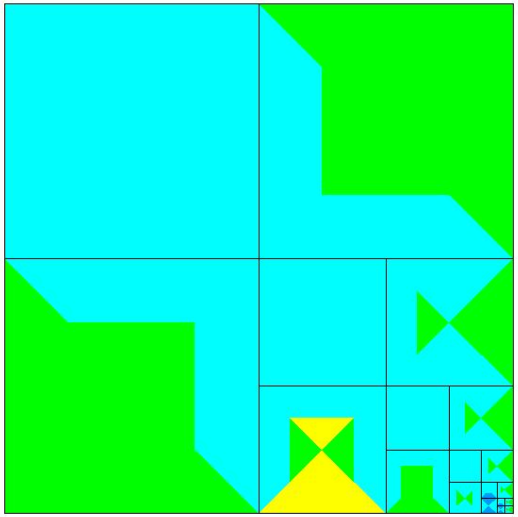


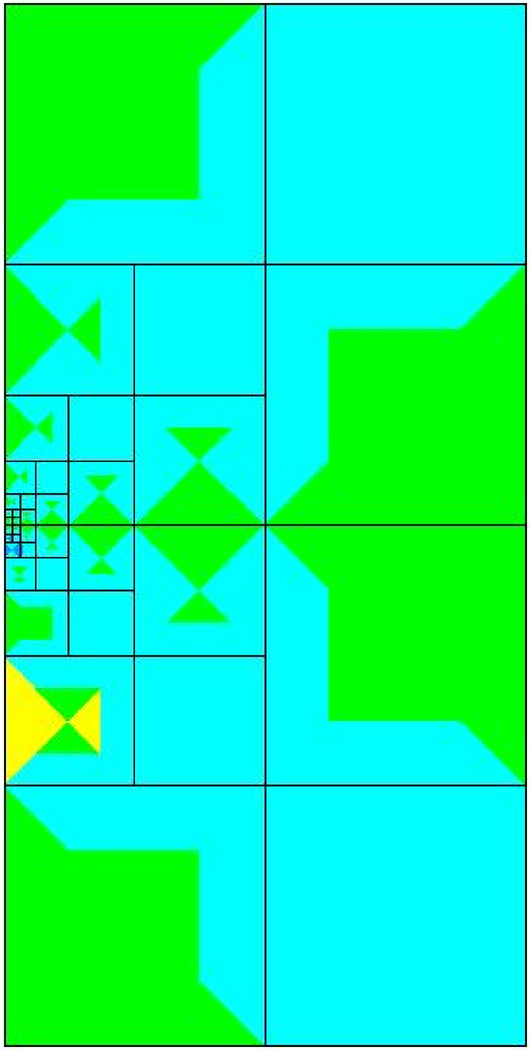
- Refinements trees are grown vertically from the initial mesh on each process
- Each process generates initial mesh elements in only a portion of the global geometry
- Identical copies of global geometry are stored on each process



DATA MIGRATION

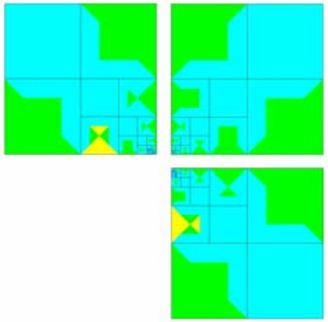


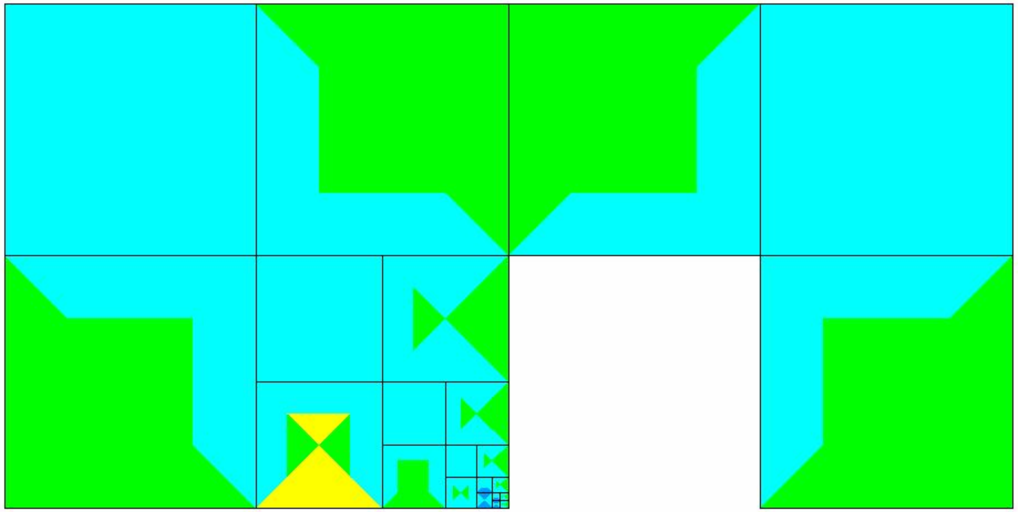


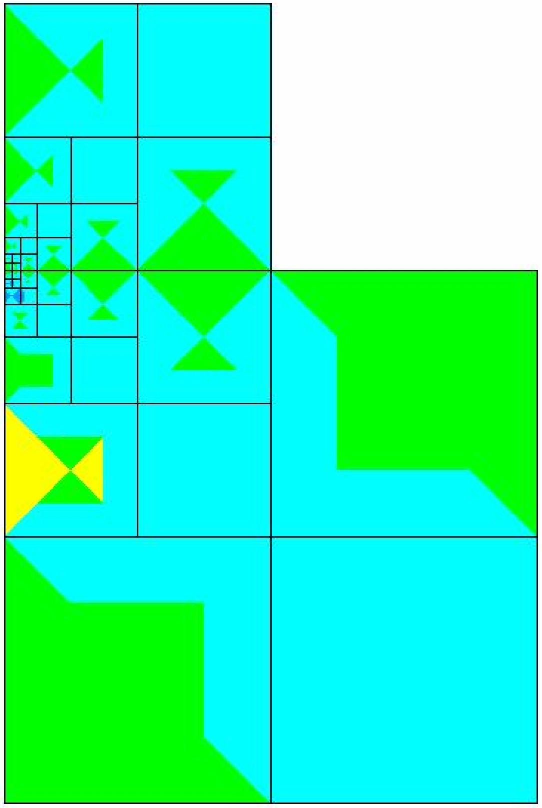


- Load balancing performed by ZOLTAN library
- ZOLTAN provides 6 different domain decomposition algorithms

DATA MIGRATION

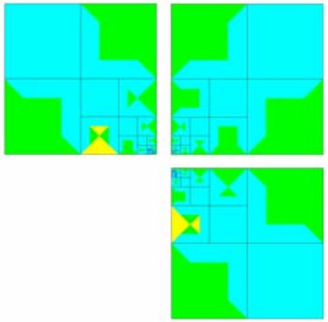






•Initial mesh elements together with refinements trees migrate through subdomains

PARALLEL DIRECT SOLVER



PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

Both the coarse and fine mesh problems

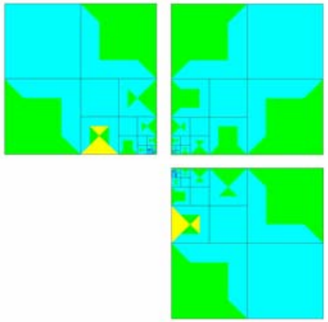
are solved using the parallel frontal solver

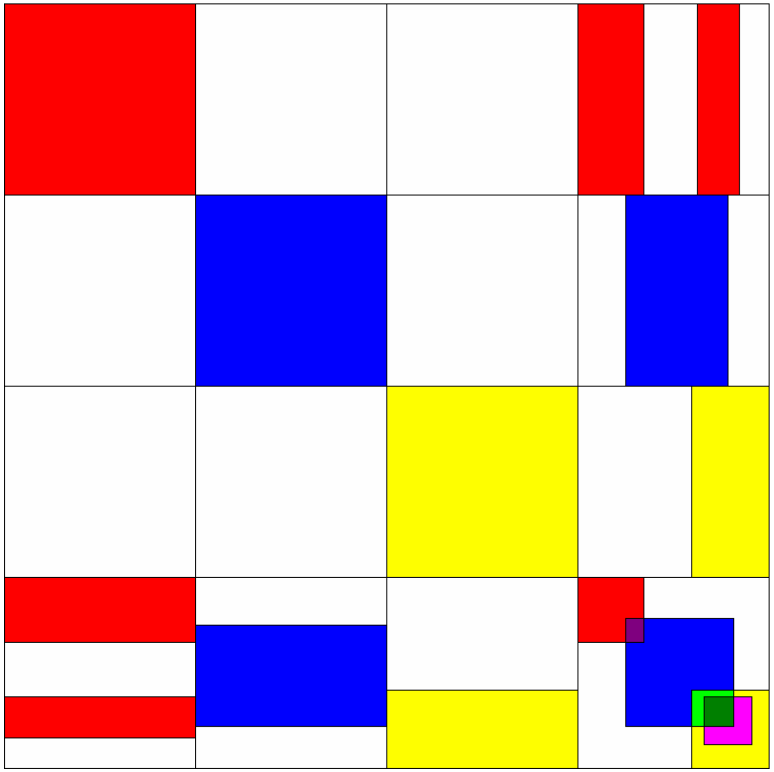
- Frontal solver = extension of the Gaussian elimination
- Assembling + Elimination performed together

  on the frontal submatrix of the global matrix

Domain decomposition approach

PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

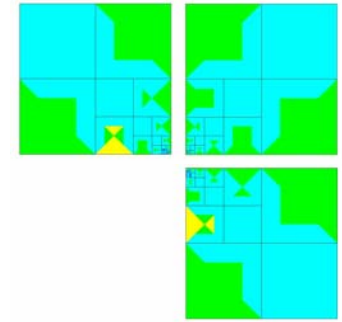


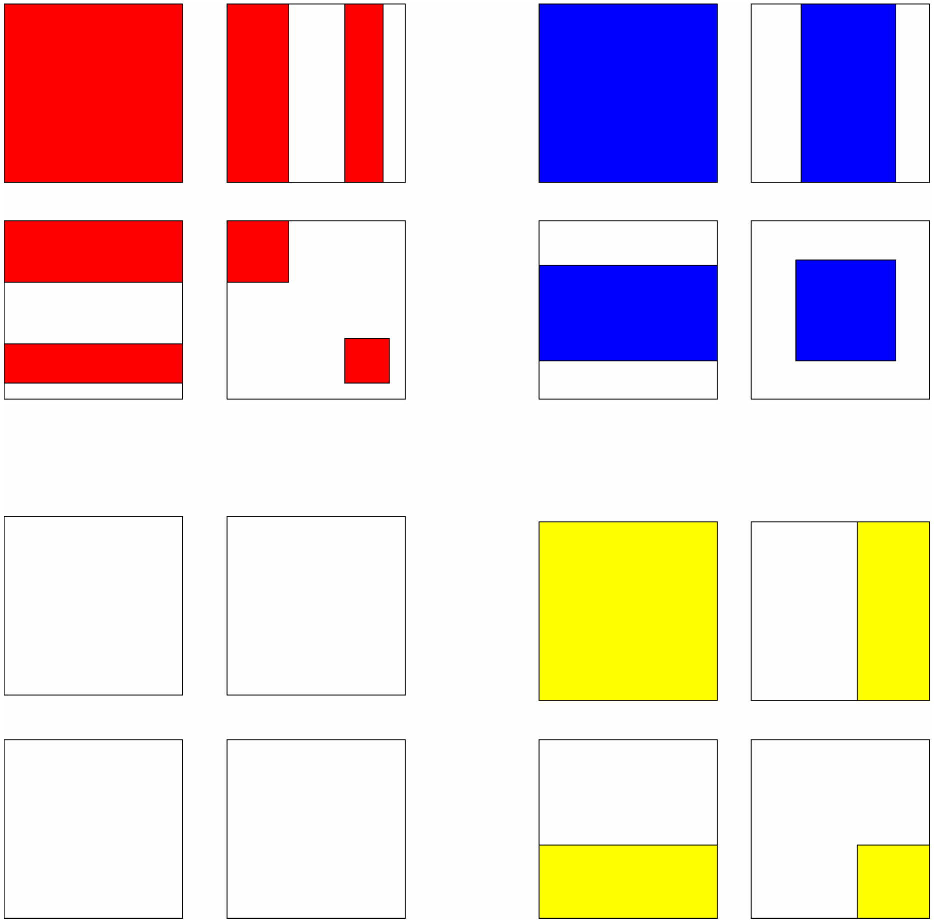


$$\begin{bmatrix} A_1 & & .. & B_1 \\ & A_2 & & B_2 \\ & & .. & .. \\ & & A_p & B_p \\ C_1 & C_2 & .. \; C_p & A_s \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_p \\ x_s \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ .. \\ b_p \\ b_s \end{bmatrix}$$

Global matrix

PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

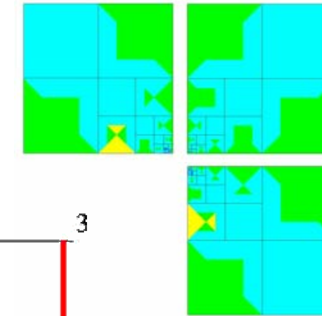




$$P_i A^{(i)} P_i^T \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & A_i & P_{ib} B_i P_{ib}^T \\ & & & \ddots \\ & & P_{bi} C_i P_{bi}^T & P_{bb} A_s^i P_{bb}^T \end{bmatrix}$$

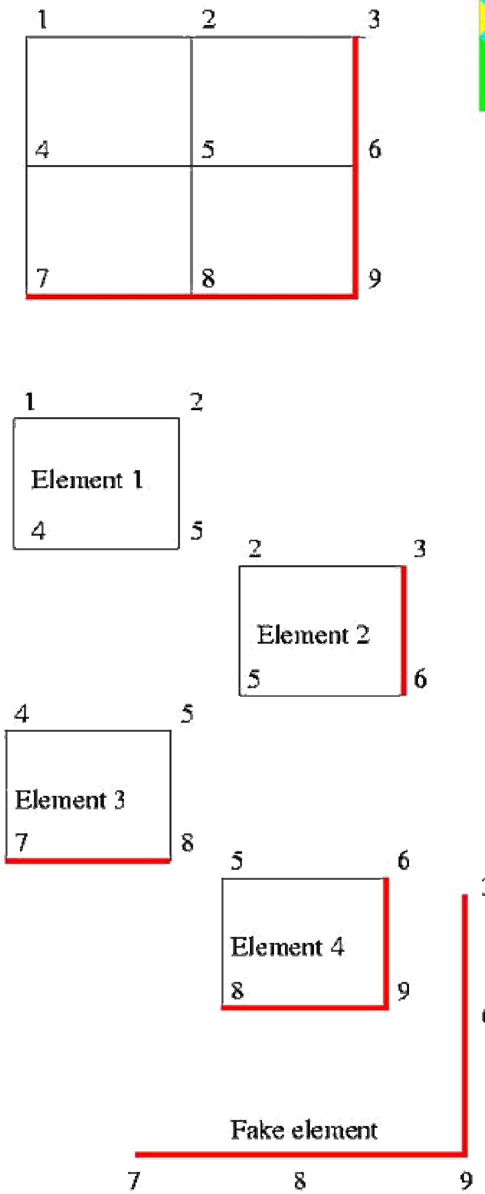
$$\begin{bmatrix} A_i & B_i \\ C_i & A_s^i \end{bmatrix} \begin{bmatrix} x_i \\ x_s^i \end{bmatrix} = \begin{bmatrix} b_i \\ b_s^i \end{bmatrix}$$

Distribution of the global matrix into processors

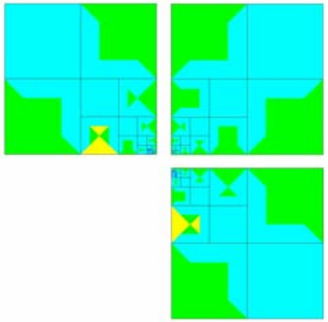


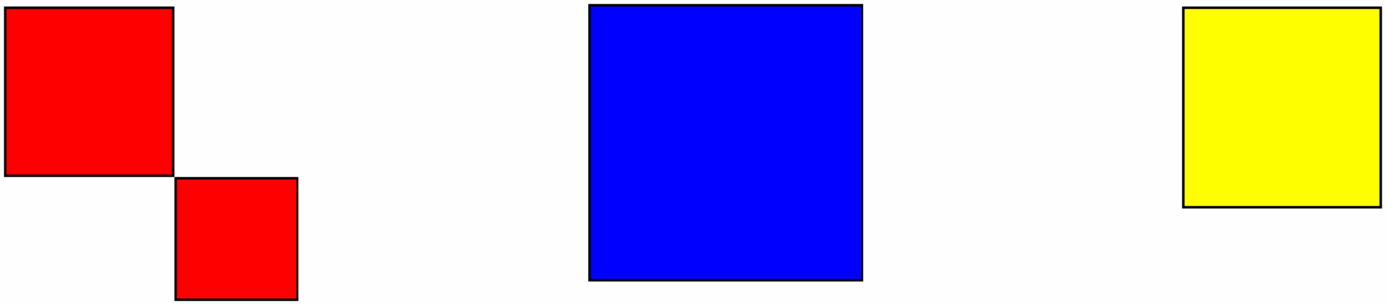
PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

1. Run the forward elimination stage with fake elements over each subdomain

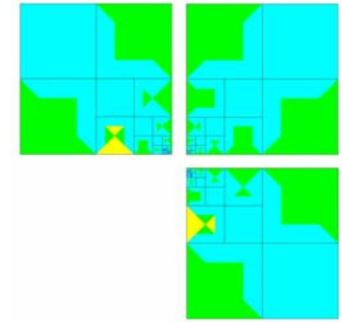


PARALLEL FRONTAL SOLVER
WITH FAKE ELEMENTS



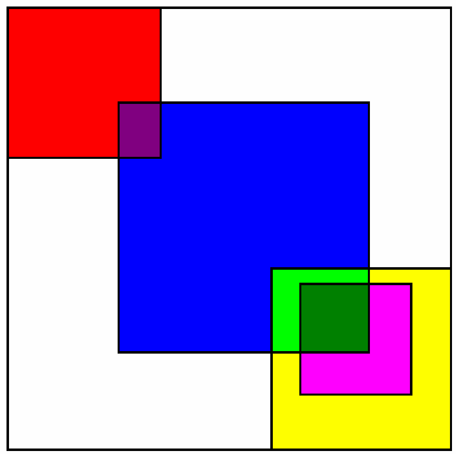


After the forward elimination with fake elements,
frontal matrices contains
contributions to the interface problem A_s^{i*}



PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

2. Formulate the interface problem

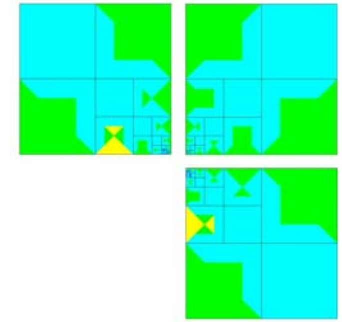


$$\hat{A}\hat{x} = \hat{b}$$

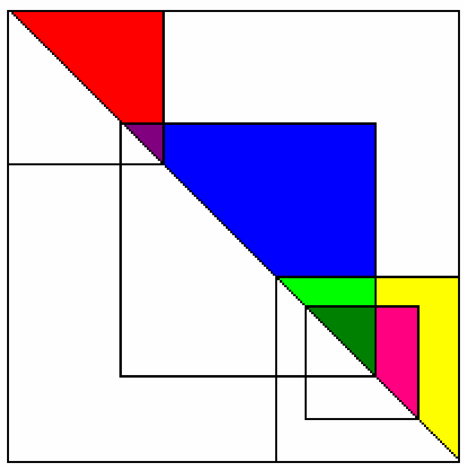
$$\hat{A} = \sum_{i=1}^{p} P A_s^{(i)^*} P^T$$

$$\hat{b} = \sum_{i=1}^{p} P b_s^{(i)^*} P^T$$

PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS



3. Solve the interface problem



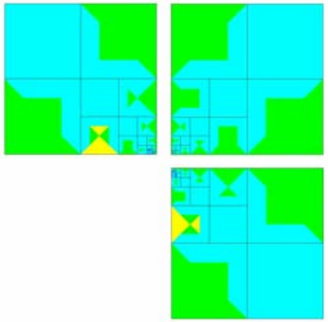
$$\hat{A}\hat{x} = \hat{b}$$

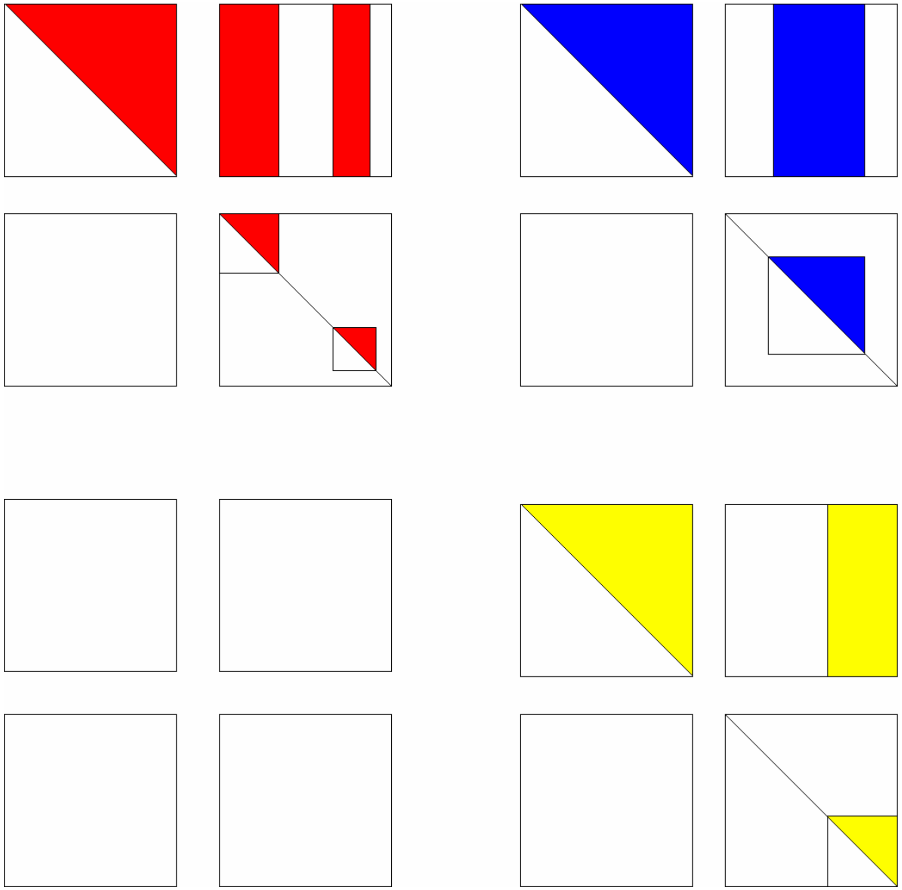
$$\hat{A} = \sum_{i=1}^{p} P A_s^{(i)^*} P^T$$

$$\hat{b} = \sum_{i=1}^{p} P b_s^{(i)^*} P^T$$

4. Broadcast the solution together with upper triangular form of the interface problem matrix

PARALLEL FRONTAL SOLVER WITH FAKE ELEMENTS

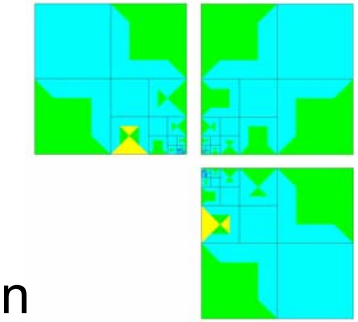




The backward substitution can be finally run in parallel, over each subdomain

PARALLEL MESH REFINEMENTS
AND
MESH RECONCILIATION

PARALLEL MESH REFINEMENTS



The mesh refinements algorithm is running on each subdomain separately

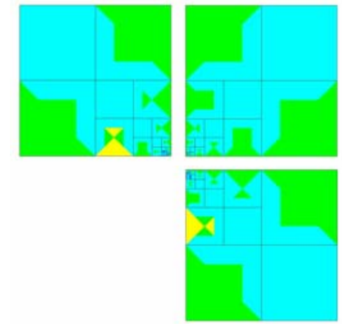
1-irregularity rule is enforced

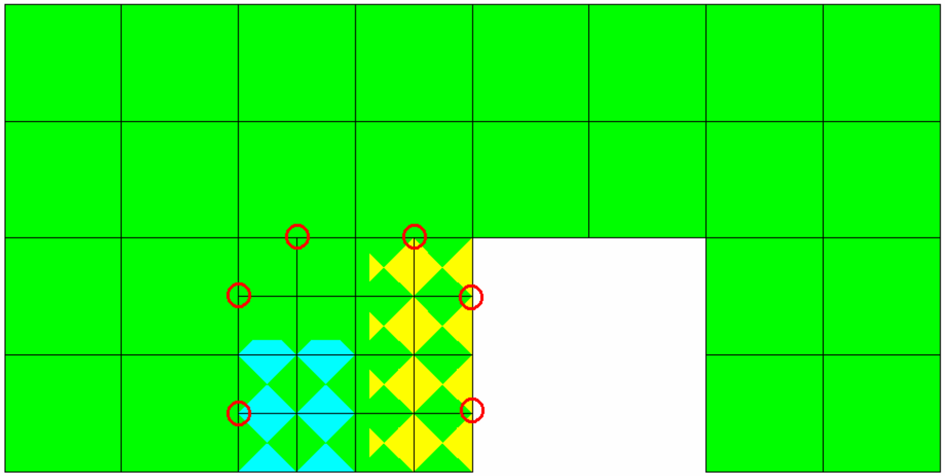
The rule is telling that edge of given element can be broken only once, without breaking neighboring elements

Nodes situated on the global interface are treated at the same way as internal nodes

After parallel mesh refinements it is necessary to run the mesh reconciliation algorithm

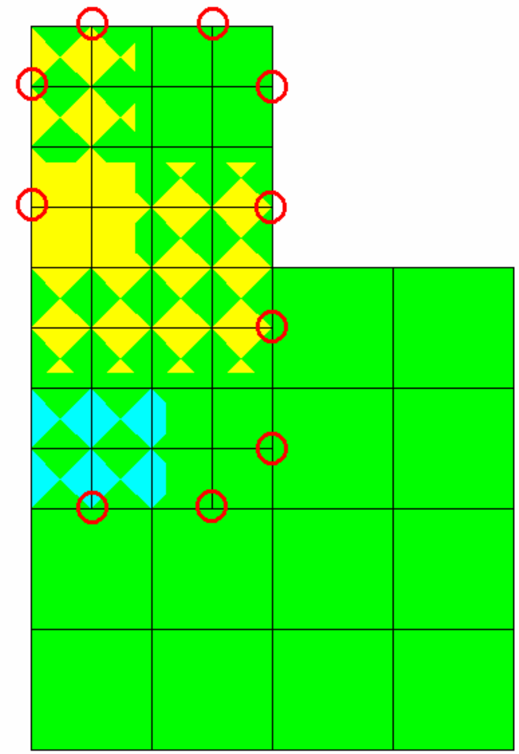
PARALLEL MESH REFINEMENTS EXAMPLE



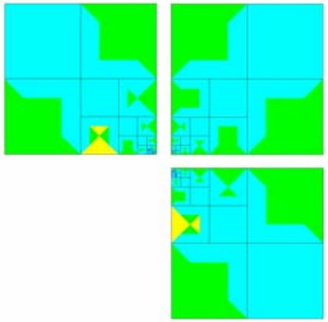


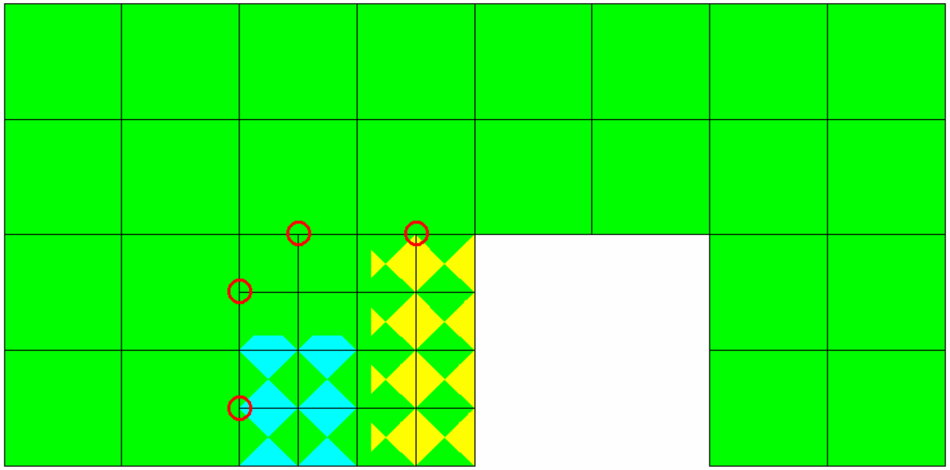
Constrained nodes

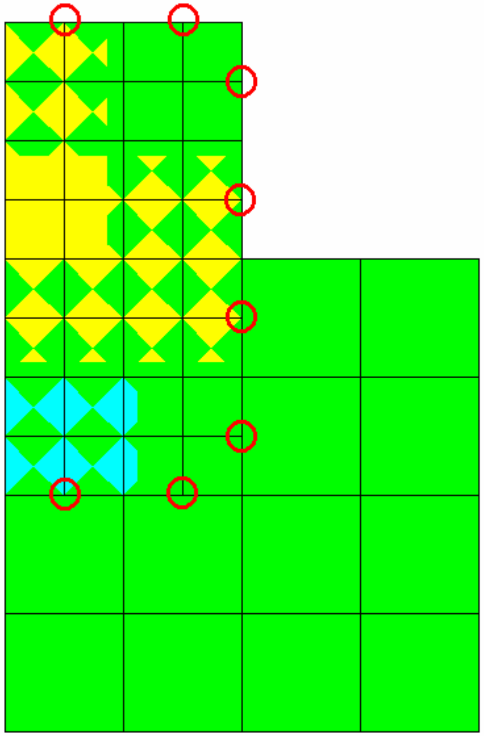
Fine mesh



PARALLEL MESH REFINEMENTS EXAMPLE



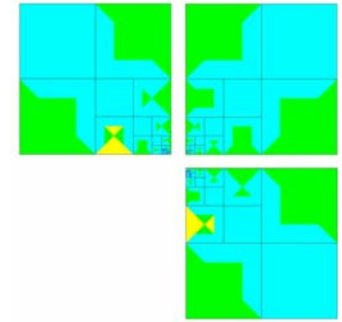


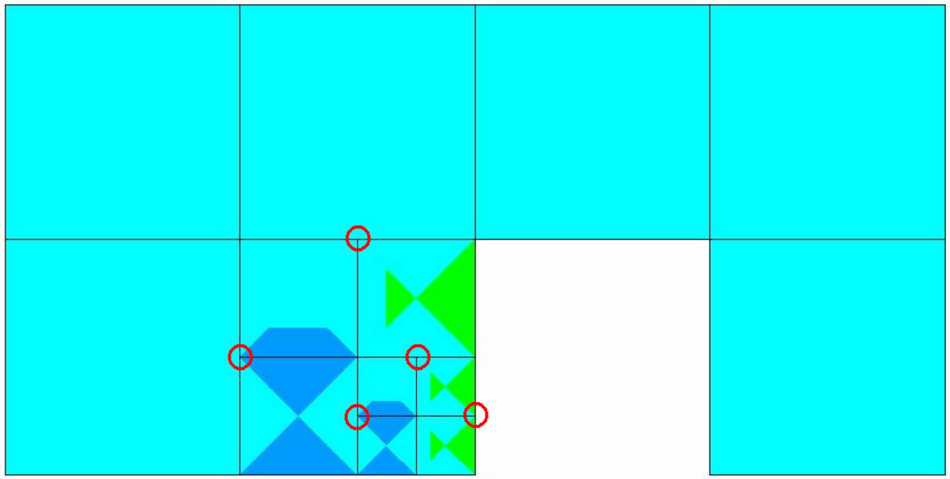


Constrained nodes

Fine mesh

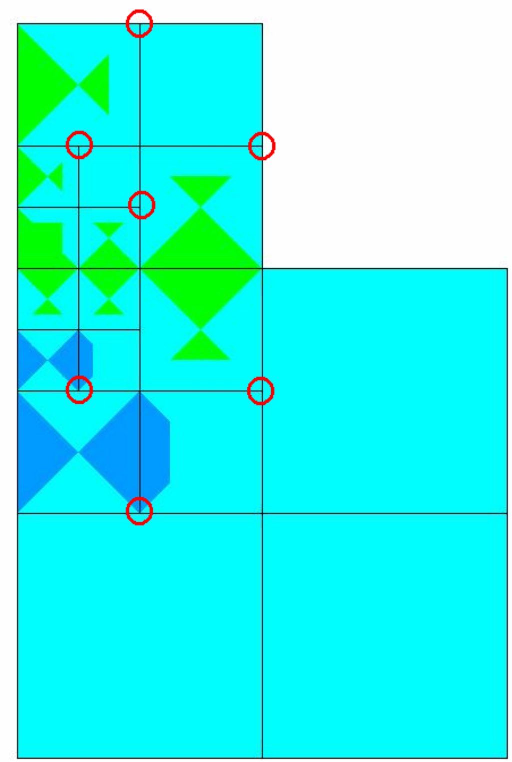
PARALLEL MESH REFINEMENTS EXAMPLE



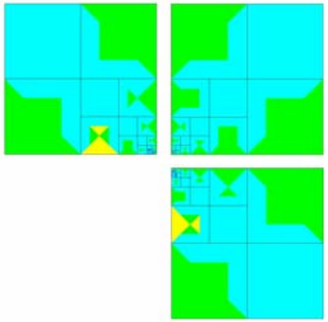


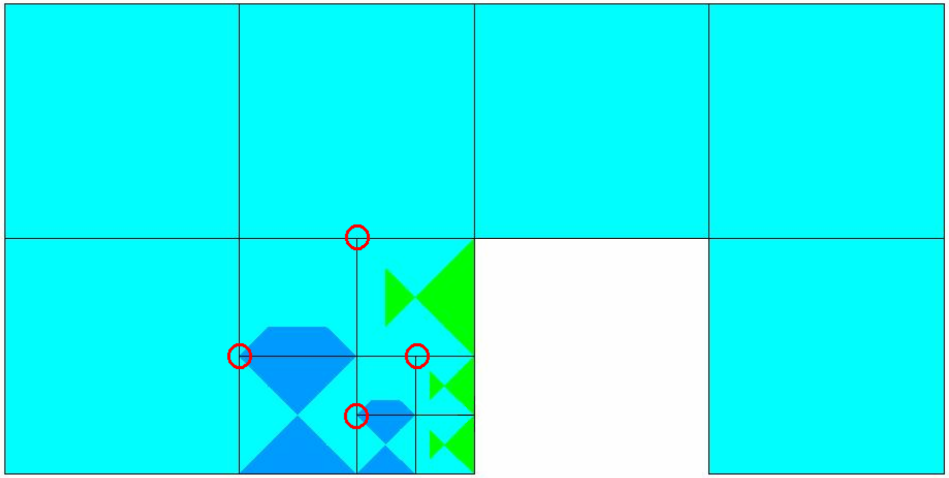
Constrained nodes

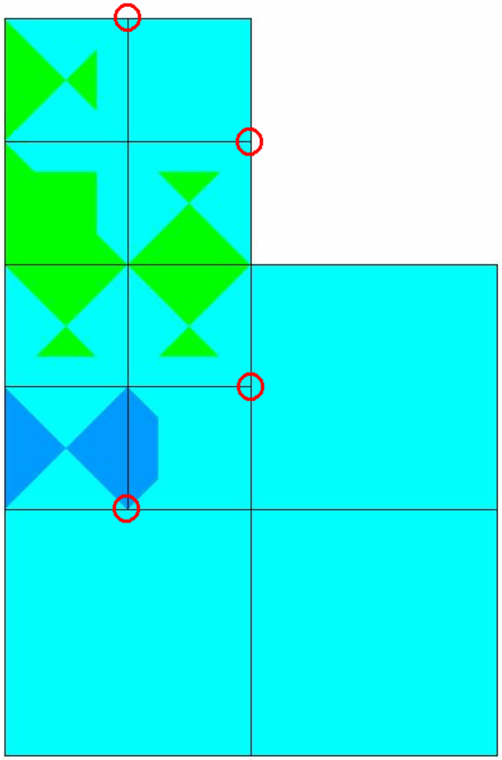
Optimal mesh



PARALLEL MESH REFINEMENTS EXAMPLE



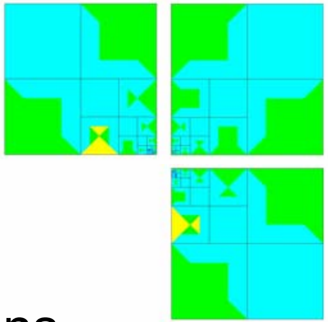




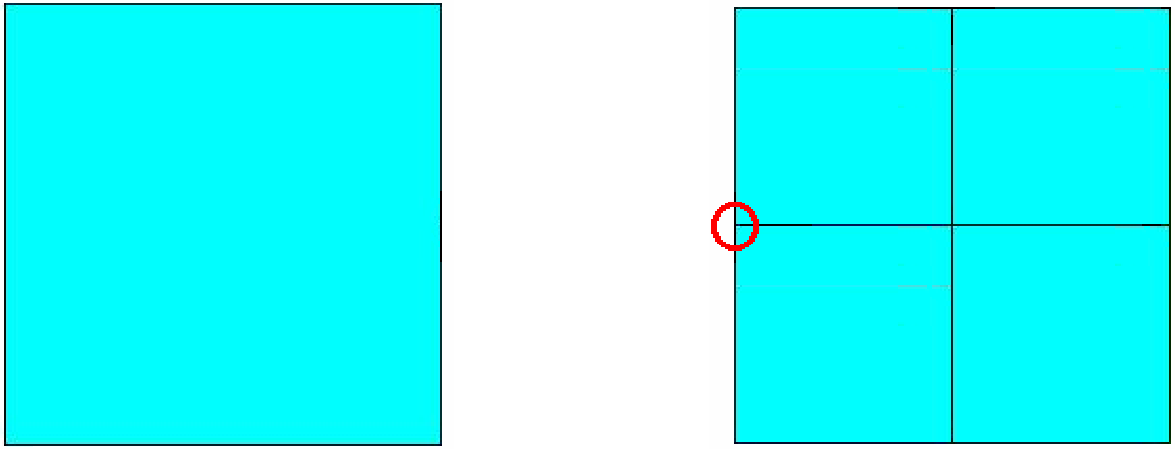
Constrained nodes

Optimal mesh

PARALLEL MESH RECONCILIATION
ADJACENCY CASE



Two adjacent elements from neighboring subdomains

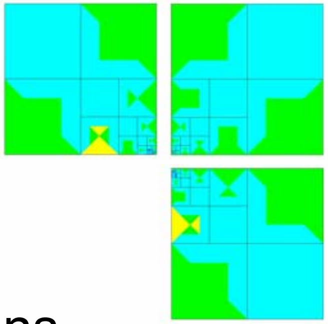


The first is not refined, the second one is refined

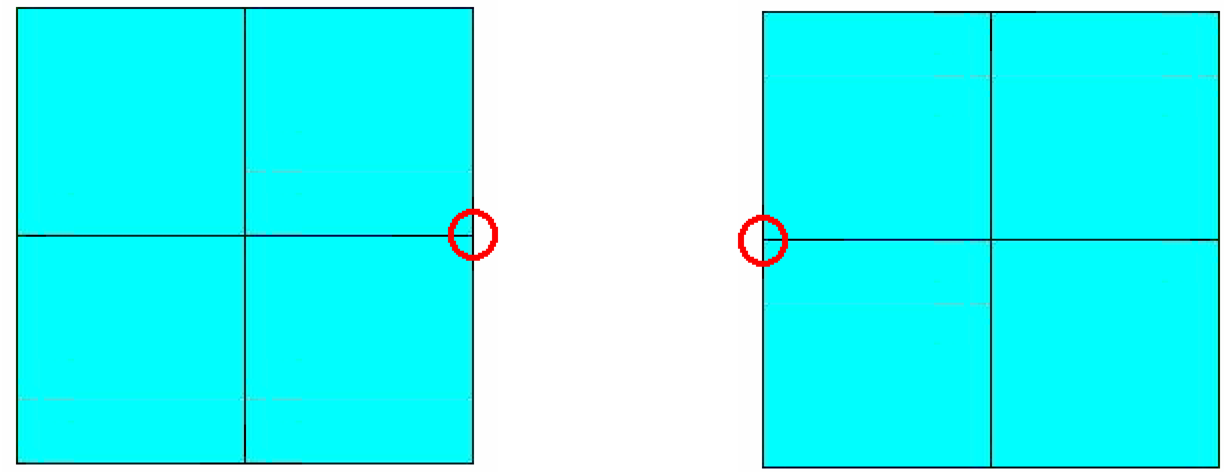
Create constrained node on the interface edge

(in order to have the same number of degrees of freedom)

PARALLEL MESH RECONCILIATION ADJACENCY CASE



Two adjacent elements from neighboring subdomains

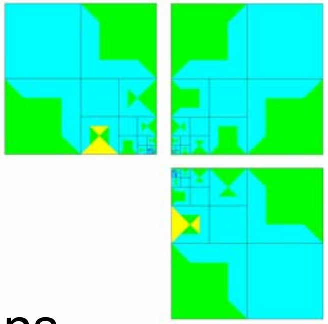


Both refined

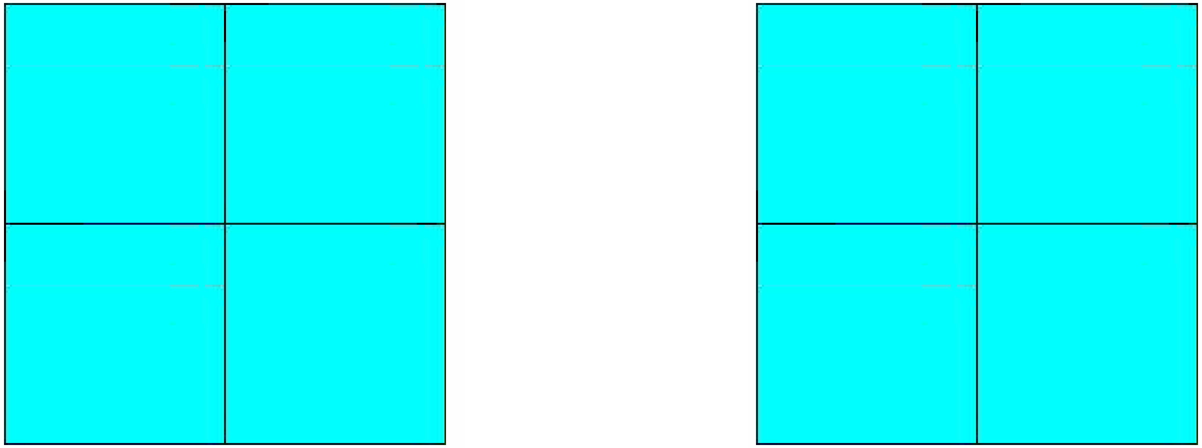
Create constrained nodes on the interface edges

Exchange constrained nodes data between subdomains

PARALLEL MESH RECONCILIATION
ADJACENCY CASE

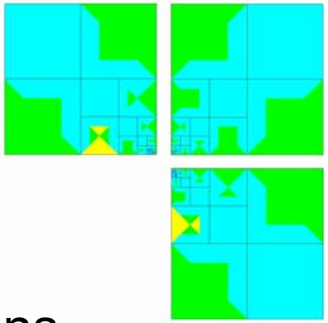


Two adjacent elements from neighboring subdomains

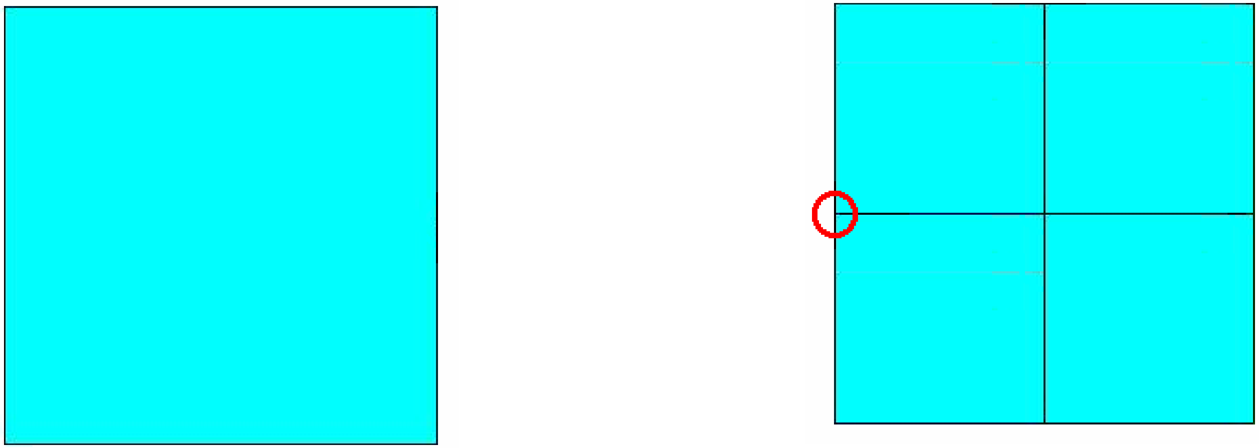


Remove interface constrained nodes situated at the same place on both subdomains

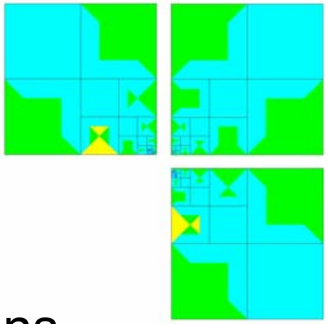
PARALLEL MESH RECONCILIATION ADJACENCY CASE



Two adjacent elements from neighboring subdomains

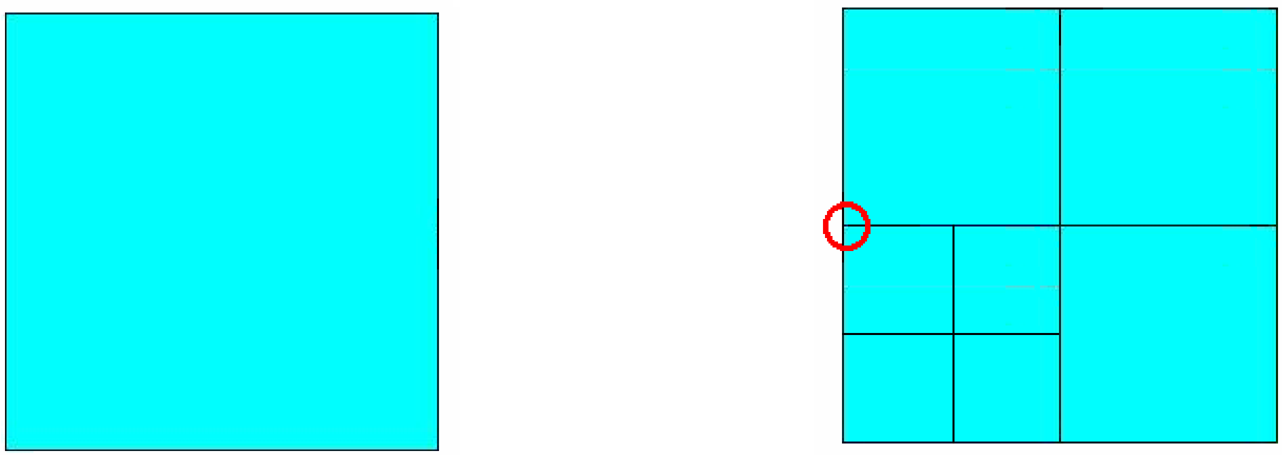


The first one is not refined, the second one is refined



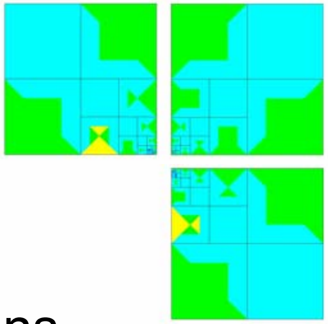
PARALLEL MESH RECONCILIATION ADJACENCY CASE

Two adjacent elements from neighboring subdomains

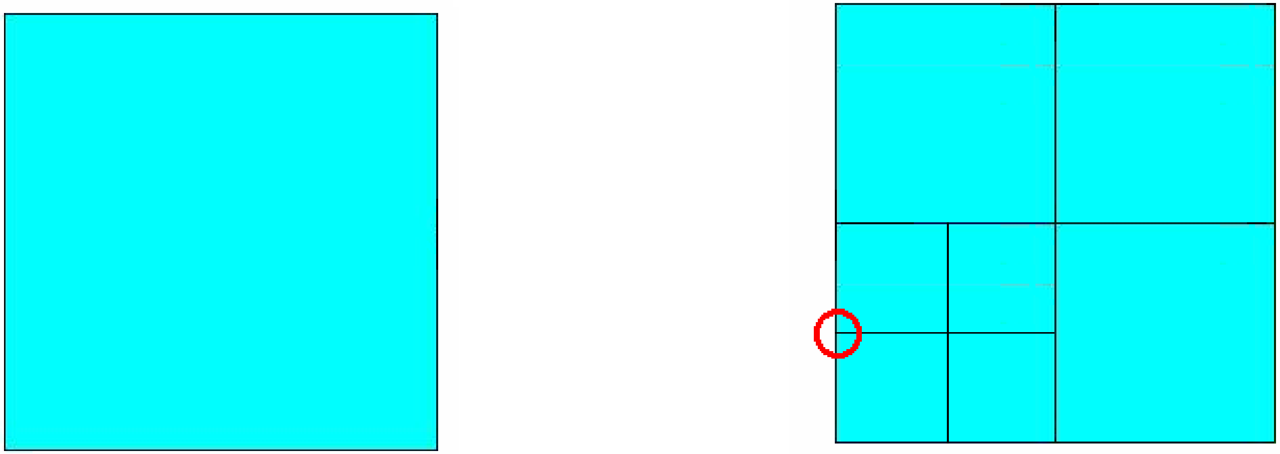


Second one is refined once again

PARALLEL MESH RECONCILIATION
ADJACENCY CASE



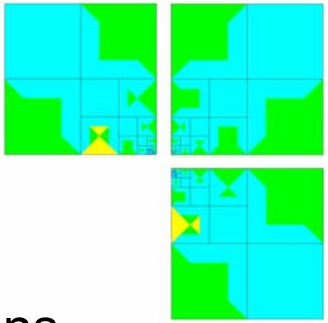
Two adjacent elements from neighboring subdomains



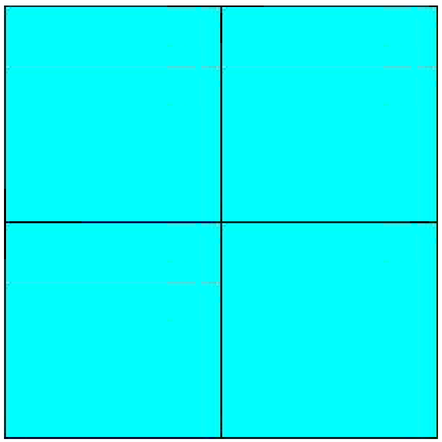
Remove constrained node

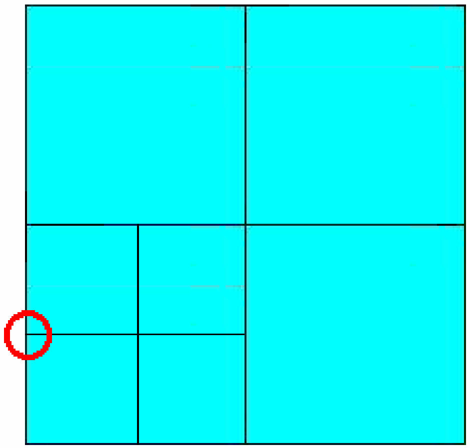
Create constrained node

PARALLEL MESH RECONCILIATION ADJACENCY CASE



Two adjacent elements from neighboring subdomains

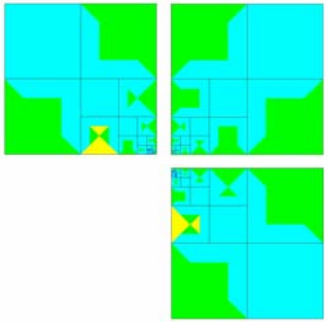




Exchange refinement trees between subdomains

Break the element

PARALLEL MESH REFINEMENTS SUMMARY

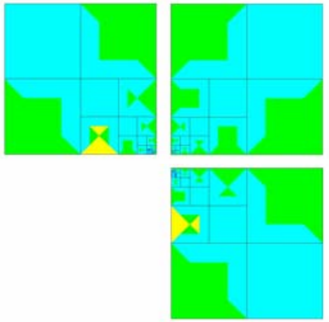


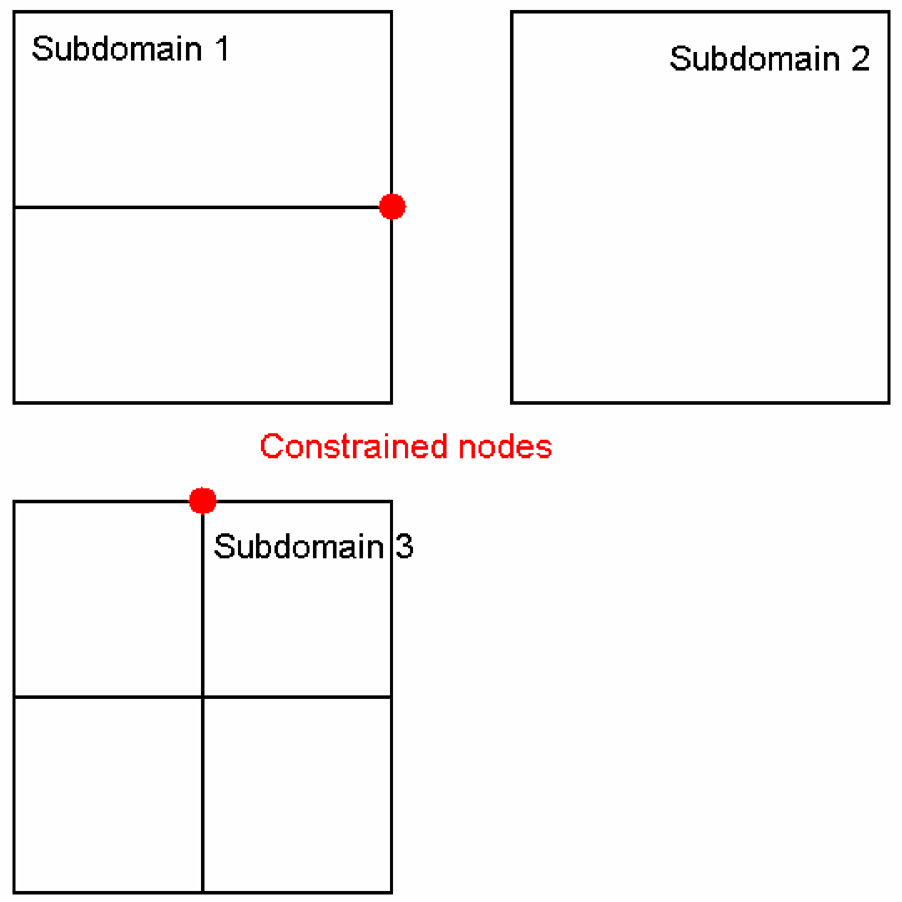
We can summarize our algorithm in the following stages

1. Parallel mesh refinements
2. Exchange information about interface edge refinement trees, constrained nodes and orders of approximation along the interface
3. Mesh reconciliation

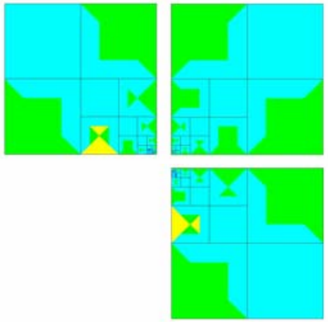
The repetition of stages 2 and 3 may be required if some of the interface edges were modified during the last iteration.

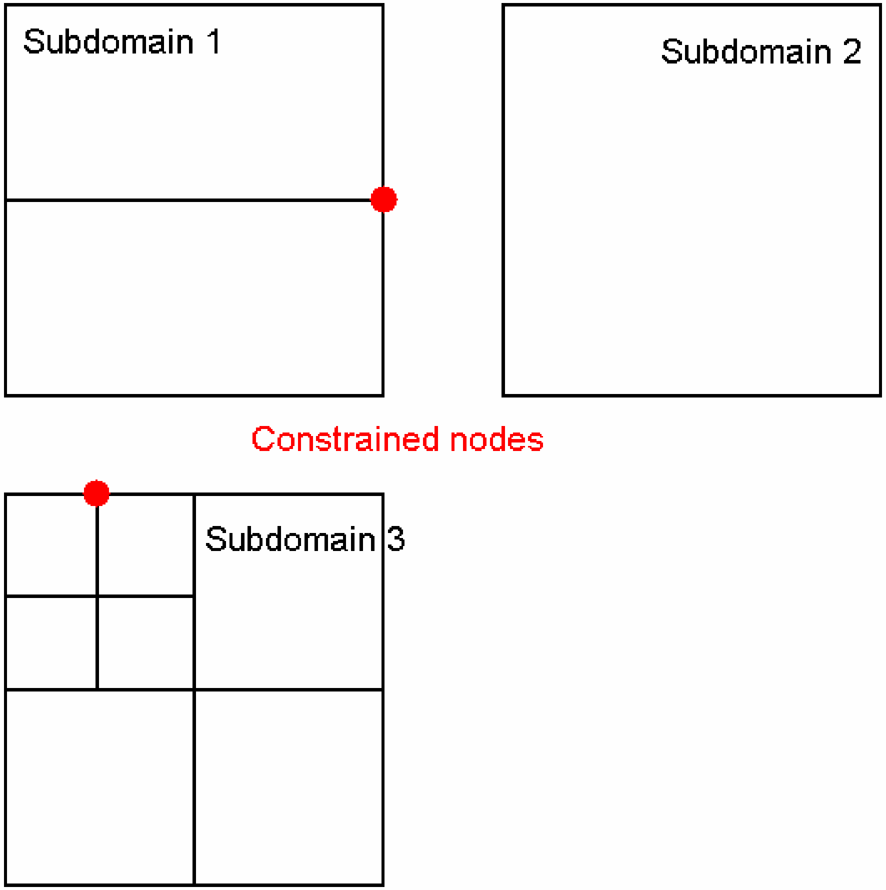
PARALLEL MESH REFINEMENTS SUMMARY



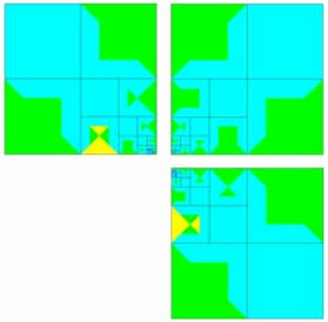


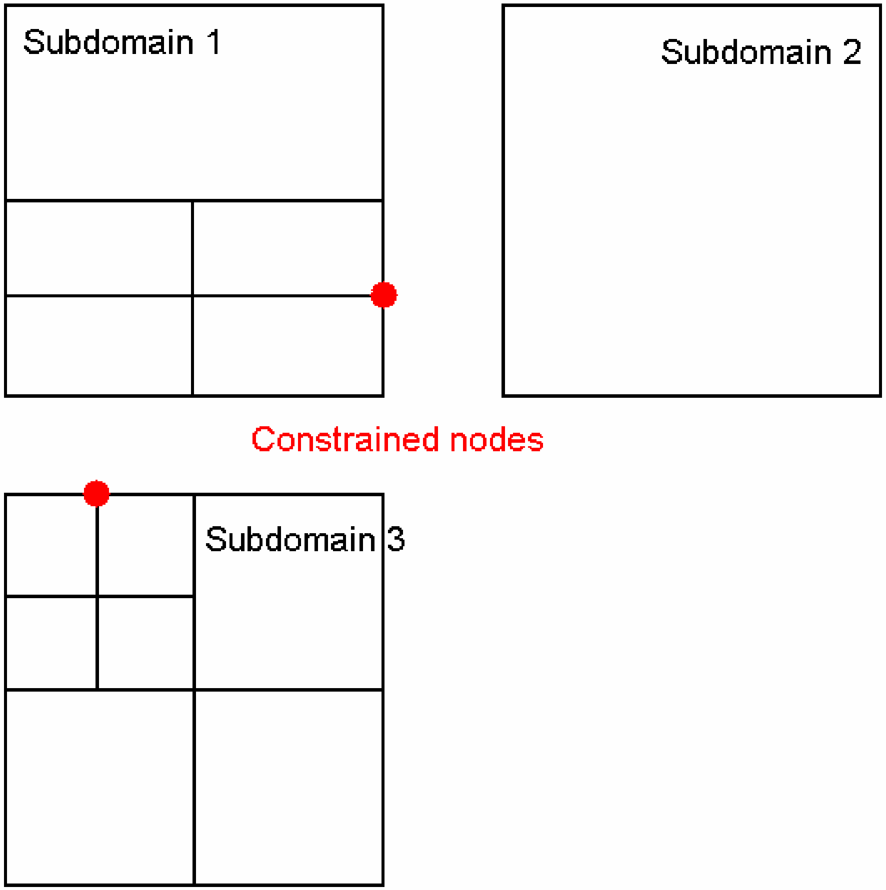
PARALLEL MESH REFINEMENTS SUMMARY

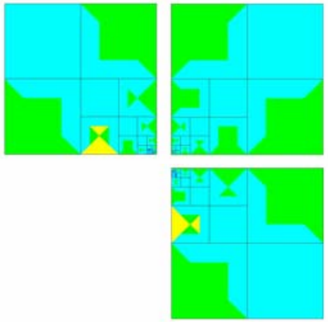




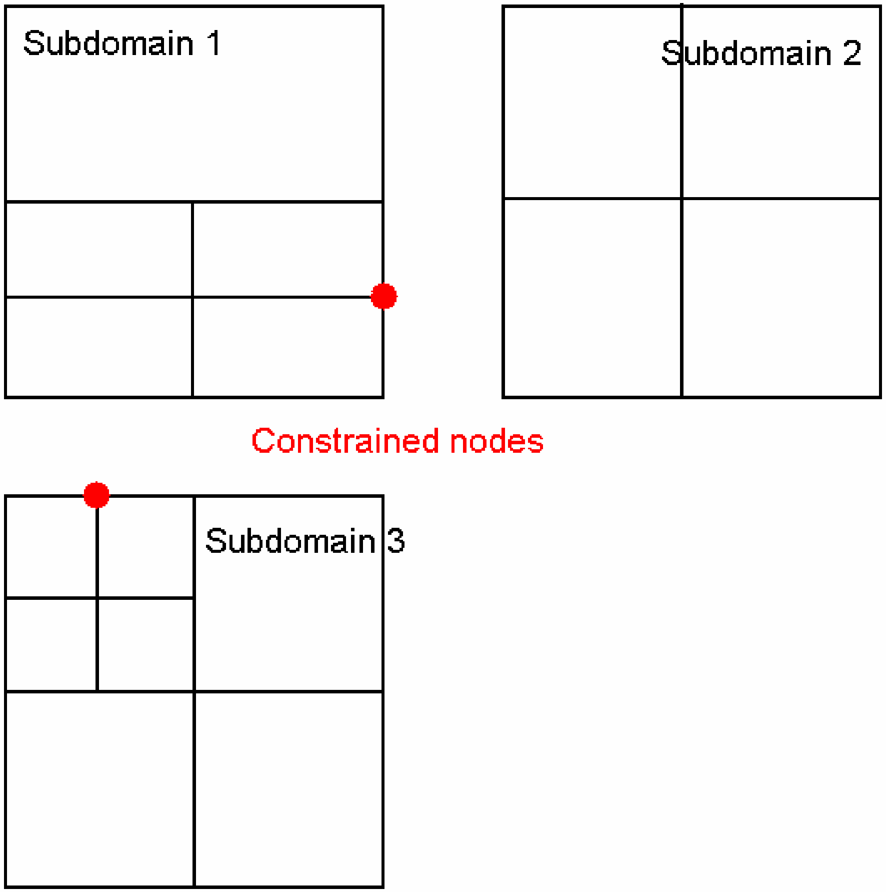
PARALLEL MESH REFINEMENTS SUMMARY



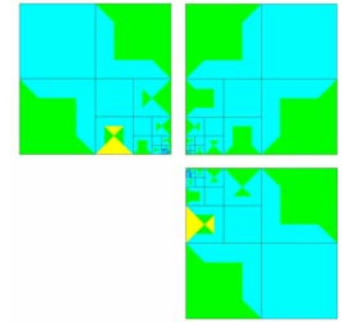




PARALLEL MESH REFINEMENTS SUMMARY



COMMUNICATION STRATEGY



COMMUNICATION STRATEGY

There are many points in our parallel fully automatic hp-adaptive algorithm, where communication between processors is required.

These include:

a) data migration during load balancing

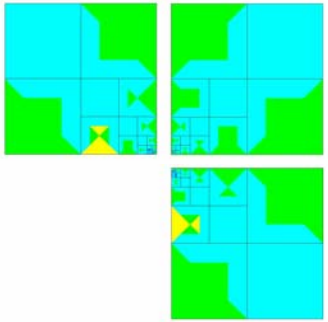
b) global denumeration of interface nodes

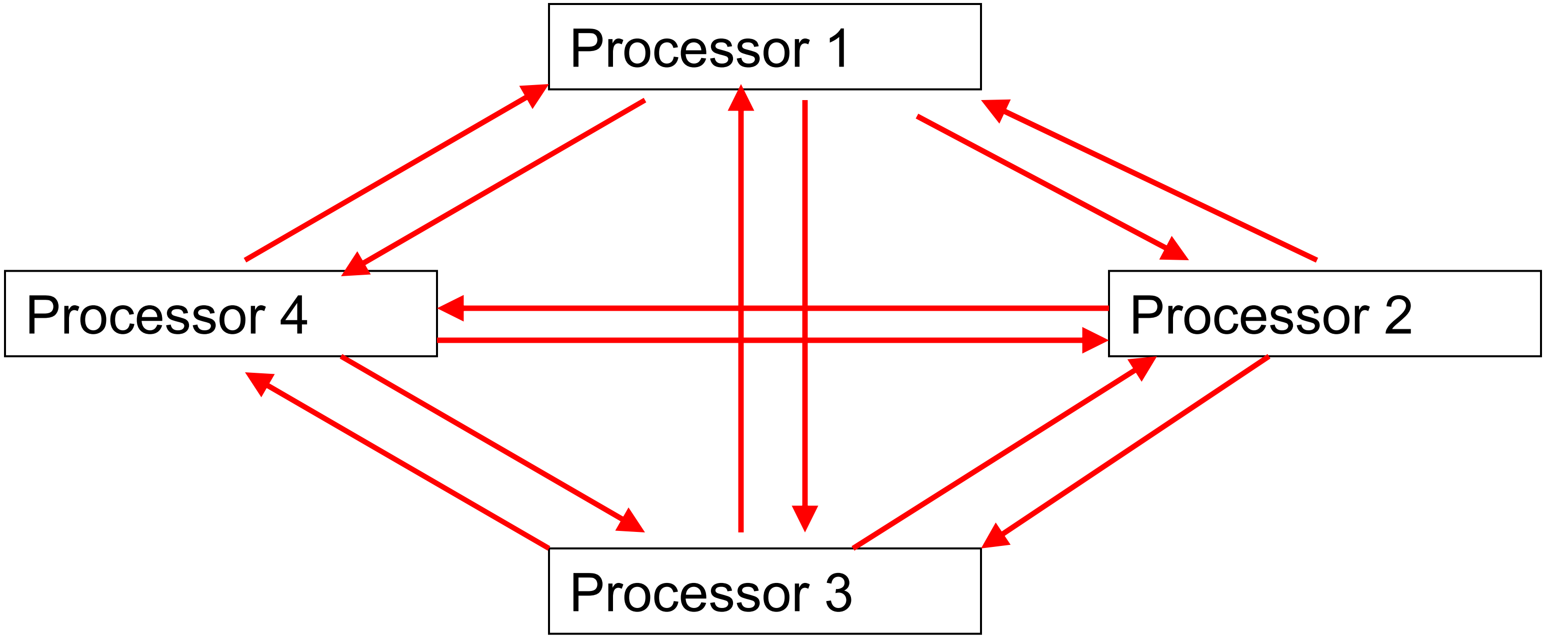
c) formulation of the wire frame problem

d) exchanging optimal refinement information over the interface

All of these points may be reduced to the problem of exchanging data between neighboring subdomains.

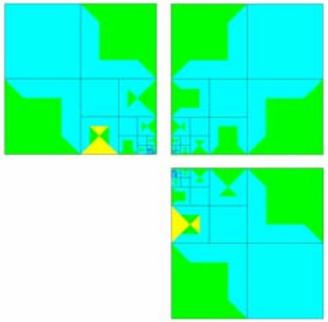
COMMUNICATION CYCLES



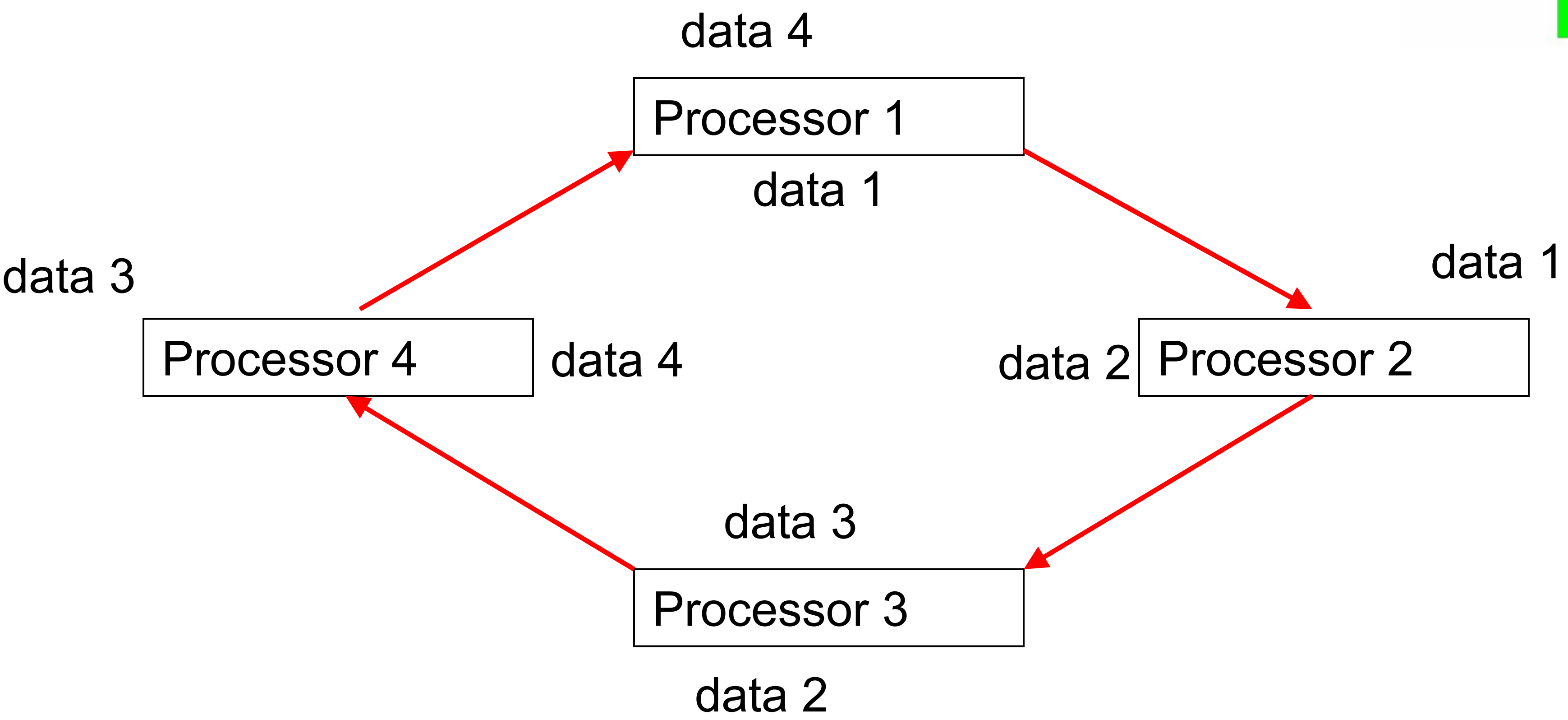


When data need to be exchanged between k processors,
the number of communication channels in direct communication scheme

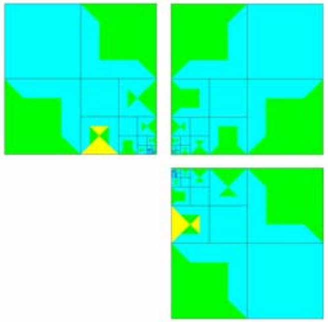
is large, equal to the number of edges in k-clique graph $\sum_{k=1}^{N-1} 2k$



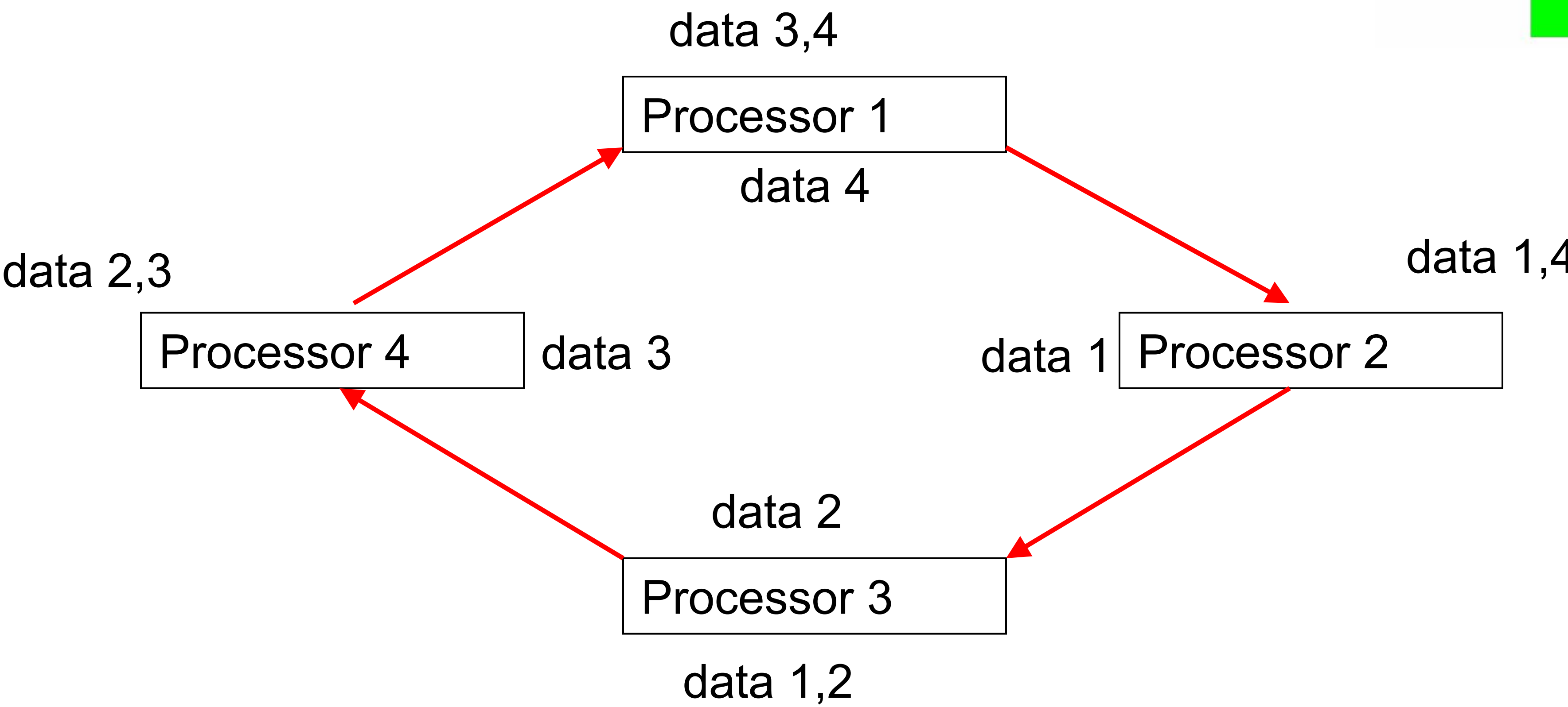
COMMUNICATION CYCLES



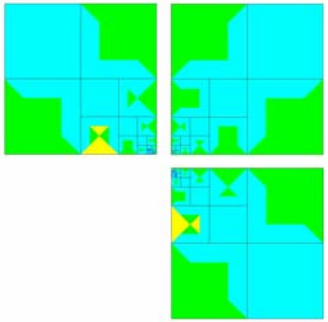
In order to reduce the number of communication channels, data are sent using cyclic communication scheme (k-1 communication cycles in k-clique graph)



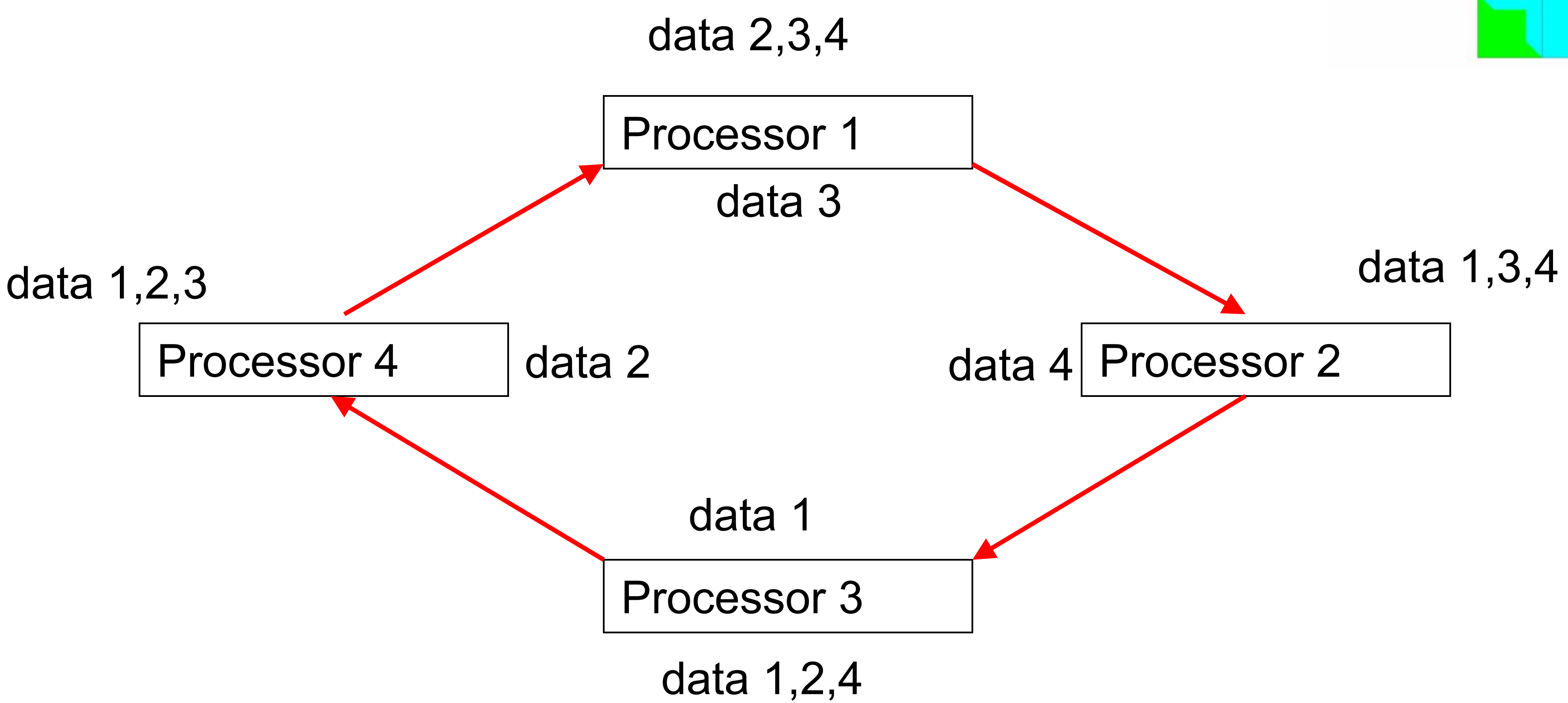
COMMUNICATION CYCLES



In order to reduce the number of communication channels, data are sent using cyclic communication scheme (k-1 communication cycles in k-clique graph)

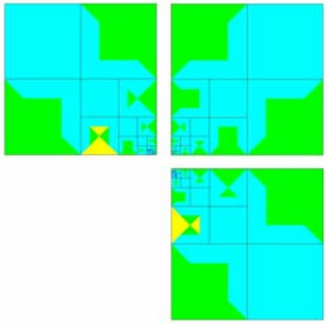


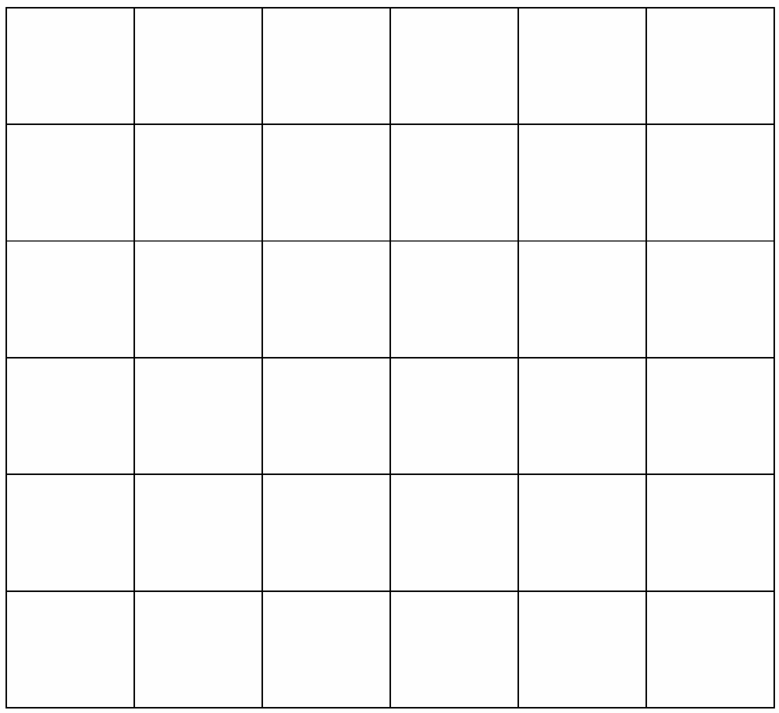
COMMUNICATION CYCLES



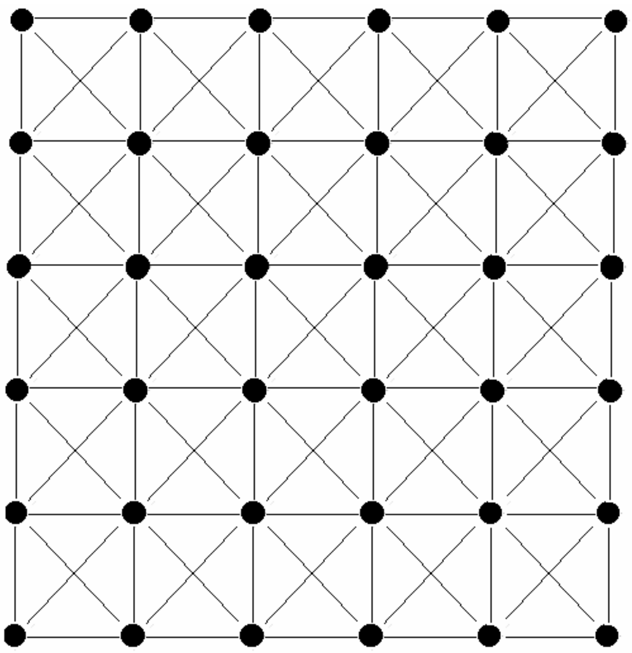
In order to reduce the number of communication channels, data are sent using cyclic communication scheme (k-1 communication cycles in k-clique graph)

GRAPH REPRESENTATION OF FINITE ELEMENT MESH





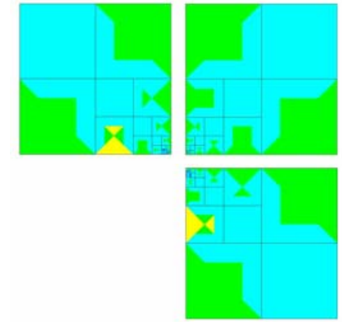
Computational domain divided into 36 subdomains



Its graph representation G=(V,E)

Node = subdomain

Edge = adjacency

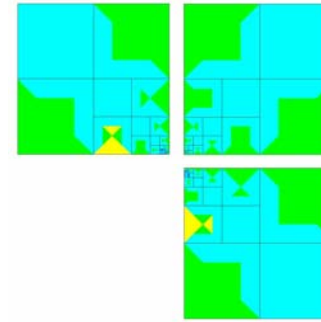


OPTIMAL COMMUNICATION CYCLES

The idea is to find covering of the graph representing computational domain by optimal set of communication cycles, where each processor in the cycle performs the following operations:

```
prepare(buffer)
do i=1,number of processors in the cycle
  send(buffer, i+1 )
  recv(buffer, i-1 )
  process(buffer)
enddo
```

GRAPH COLORING ALGORITHM



$$c : V \rightarrow \{1, 2\} :$$

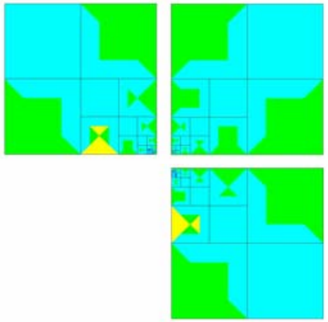
$$\forall v \in V \quad \exists < v_0, ..., v_k >: v_0 = u_0, v_k = v,$$

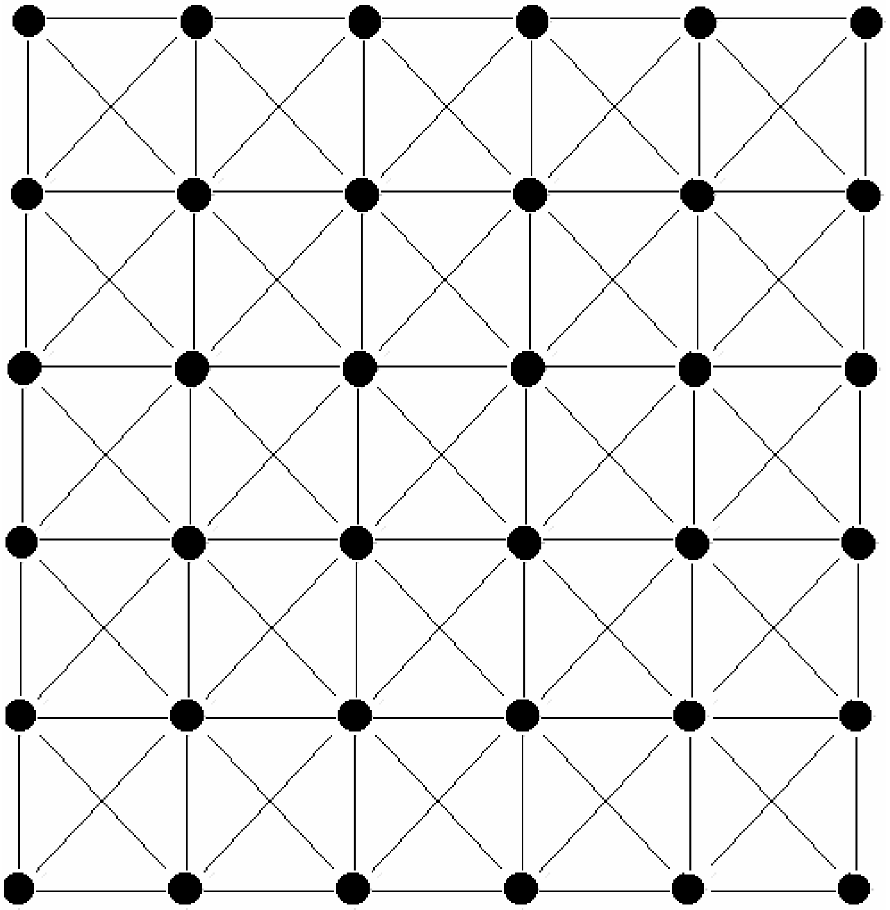
$$v_i \neq v_j, i \neq j, \quad c(v_{i-1}) \neq c(v_i) \quad \forall i = 1, ..., k,$$

$$< v_0, ..., v_k >= \min_{< w_0, ..., w_k >: w_0 = u_0, w_k = v}$$

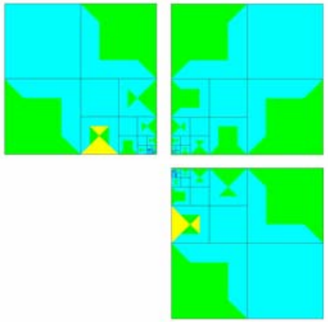
The graph coloring algorithm creates layers around one selected node of the graph

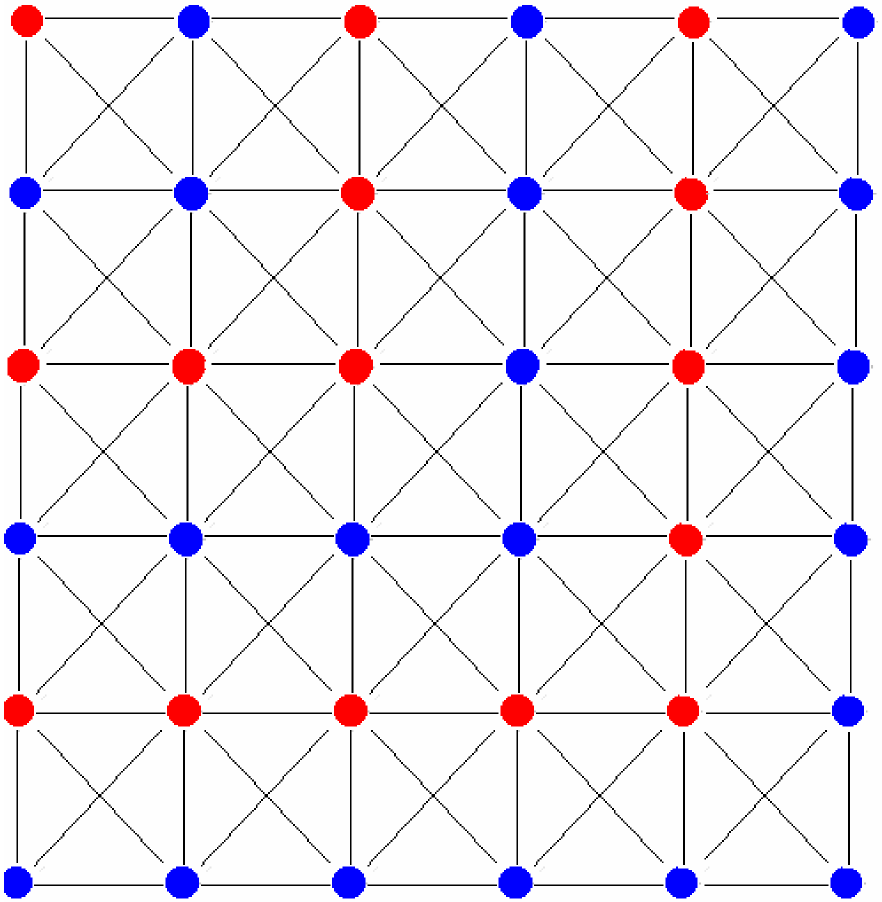
GRAPH COLORING ALGORITHM

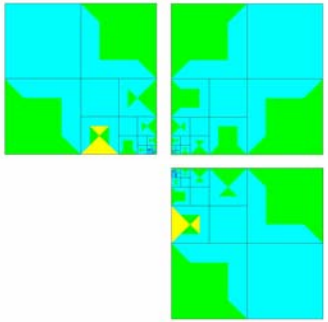




GRAPH COLORING ALGORITHM







GRAPH COLORING ALGORITHM

We define layers $L_i \subset V$ as compact sets of graph nodes colored by the same color

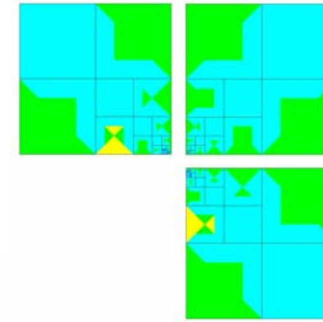
$$L_i \subset V :$$

$$\forall u, w \in L_i \,, \quad c(u) = c(w)$$

$$\exists < v_0, \ldots, v_k > \subset V :$$

$$v_0 = u, v_k = w, c(v_i) = c(u) \quad \forall i$$

GRAPH COLORING ALGORITHM



Two sets of communication cycles: ”odd”

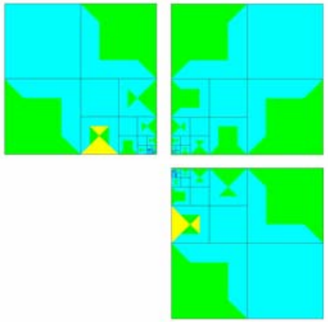
$$CC_i^1 = L_{2i-1} \cup L_{2i}$$

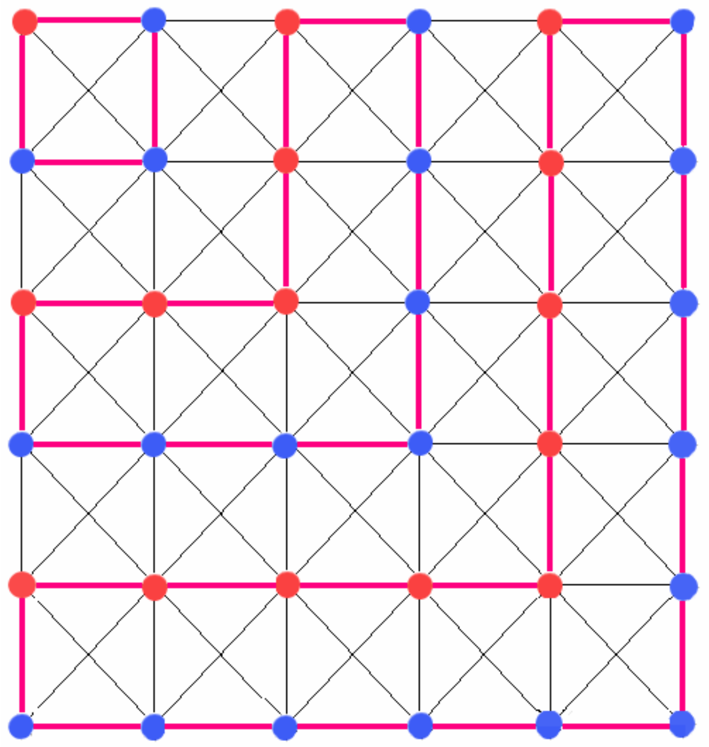
and ”even”

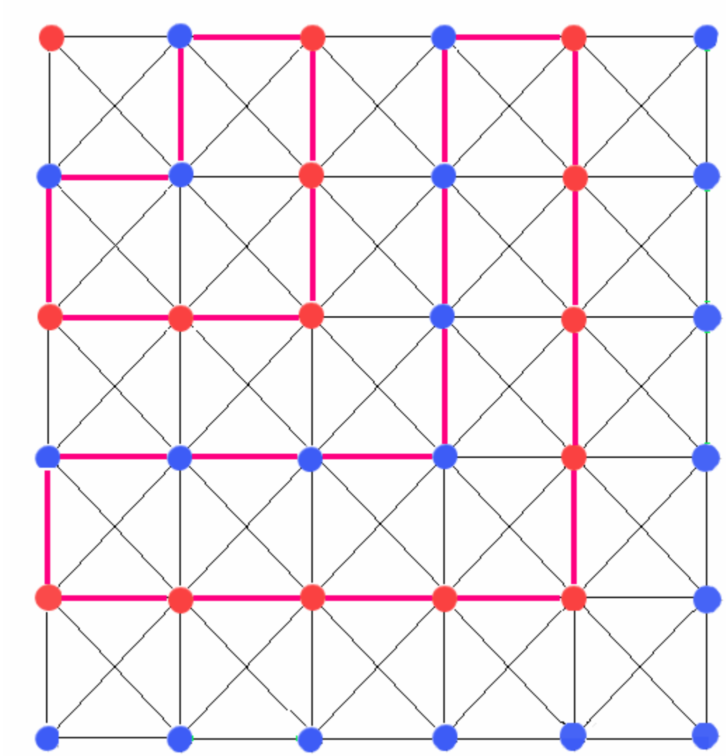
$$CC_i^2 = L_{2i} \cup L_{2i+1}$$

(for all admissible i) are painted over the nodes

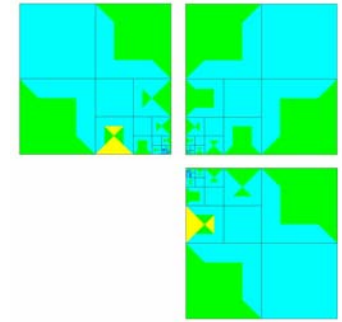
SET OF OPTIMAL COMMUNICATION CYCLES







GRAPH COLORING ALGORITHM



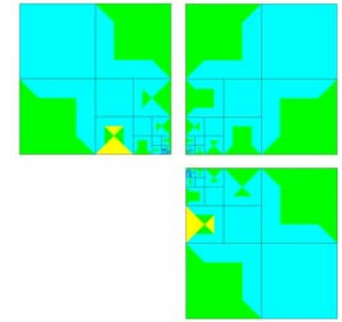
The graph coloring algoritm can be ran recursively, over each communication cycle graph

$$G_j^i = \left(V_j^i, E_j^i\right)$$
$$V_j^i = CC_j^i$$
$$E_j^i = \{\{u, v\} : u, v \in V_j^i, \{u, v\} \in E\}$$

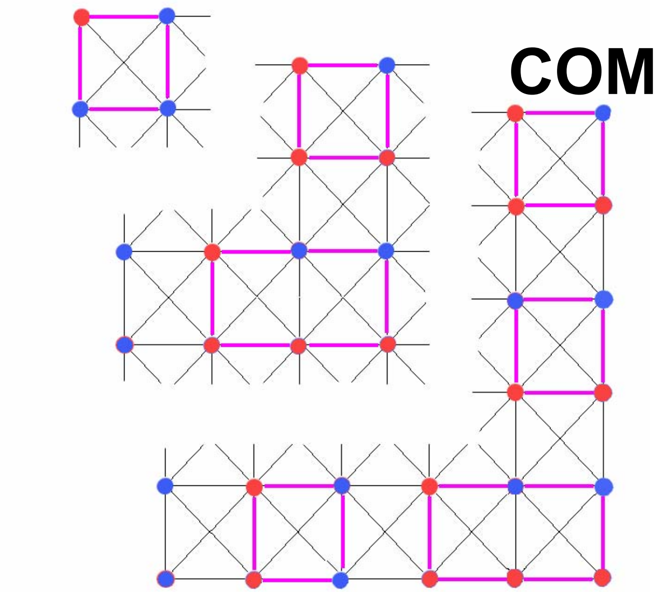
Two new sets of communication cycles are created from each communication cycle.
This could be done unless the total communication cost will be optimal

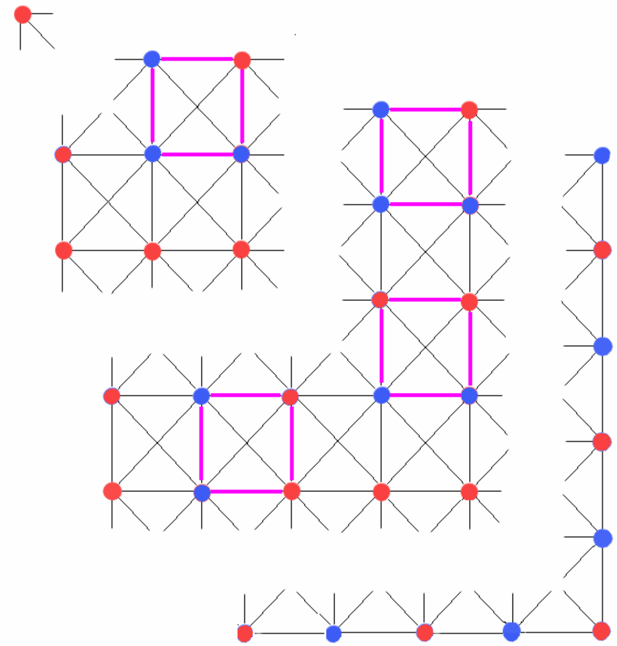
$$\text{communication cost} = \sum_i \left(\max_j \#CC_j^i\right)$$

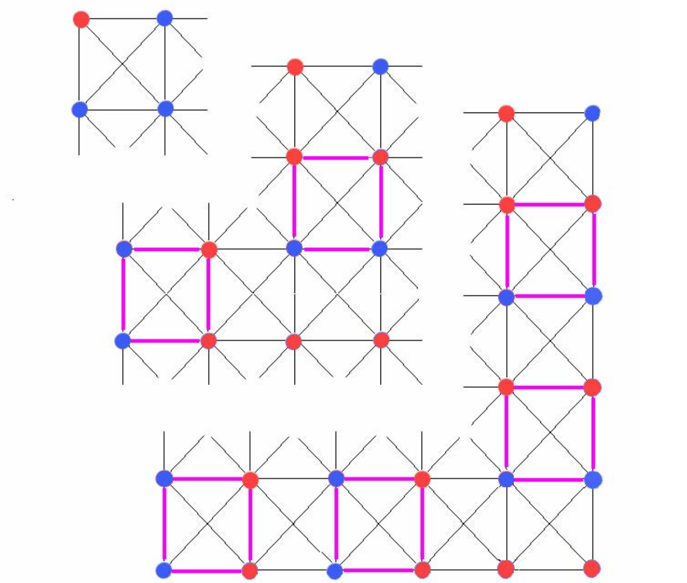
$\#CC_j^i$ is the number of vertices in the j-th communication cycle from i-th set of communication cycles.

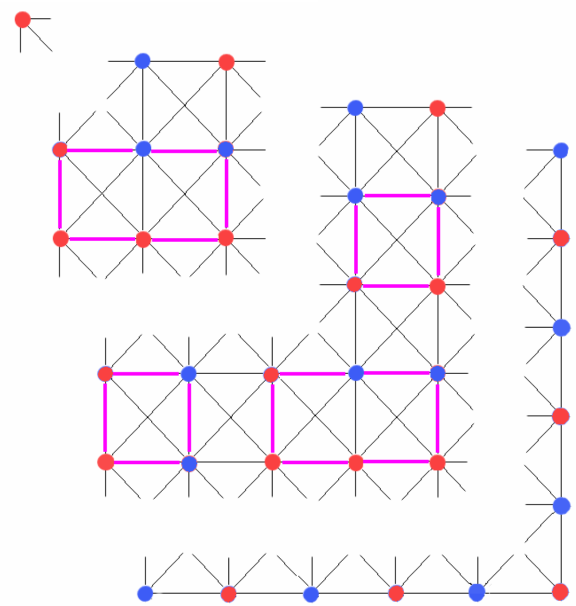


SET OF OPTIMAL COMMUNICATION CYCLES

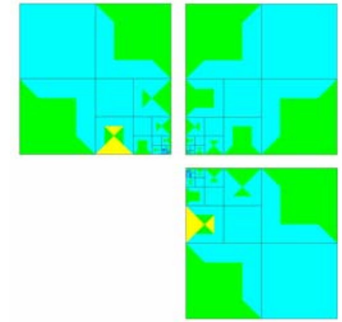








OPTIMAL COMMUNICATION CYCLES



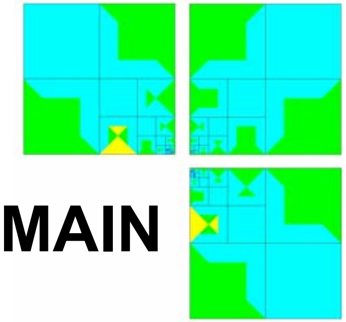
The algorithm is the following

1. All communication cycles from the first set of communication cycles are performed.
2. All communication cycles from the second set of communication cycles are performed.
3. …
4. All communication cycles from the last set of communication cycles are performed

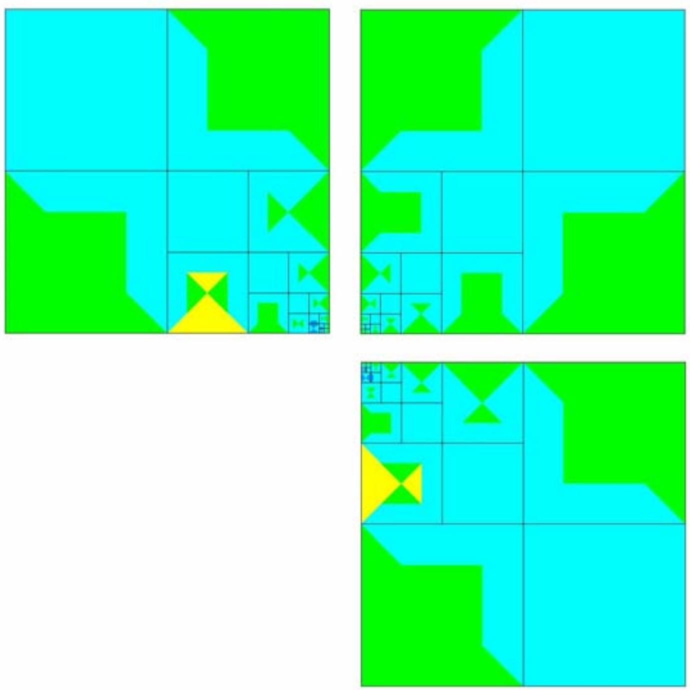
The algorithm allows us to reduce

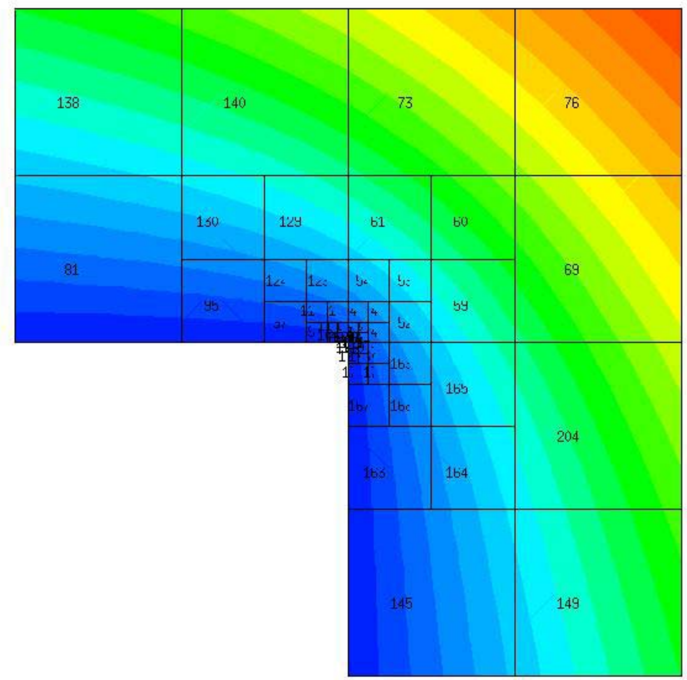
- Number of communication channels
- Total communication time
  (since all communication cycles from one set of communication cycles can be performed **at the same time**)
  (assuming there are enough interprocessor connections)

RESULTS



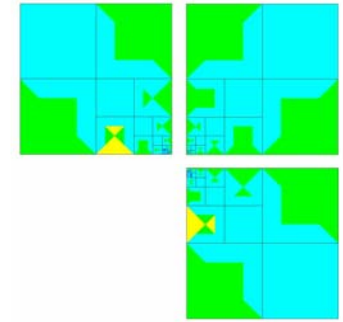
RESULTS
THE LAPLACE EQUATION OVER L-SHAPE DOMAIN



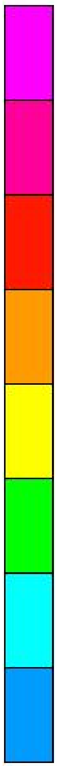


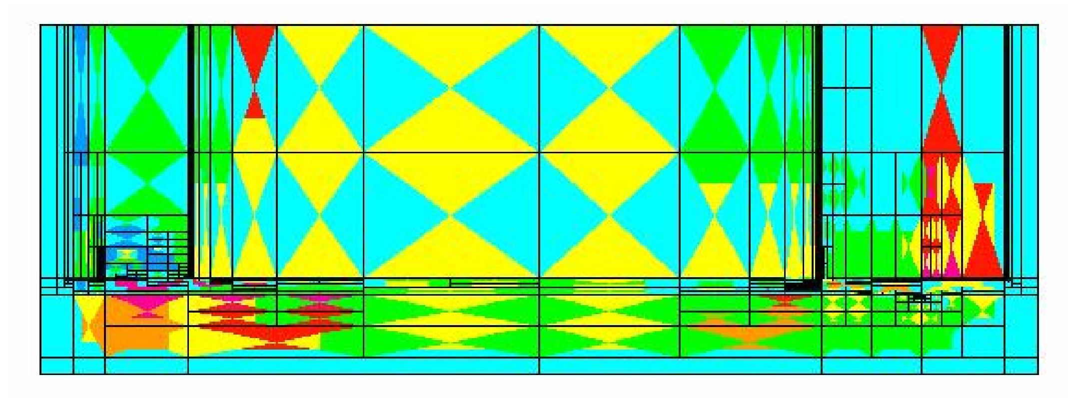
Optimal mesh obtained after parallel iterations over 3 subdomains.

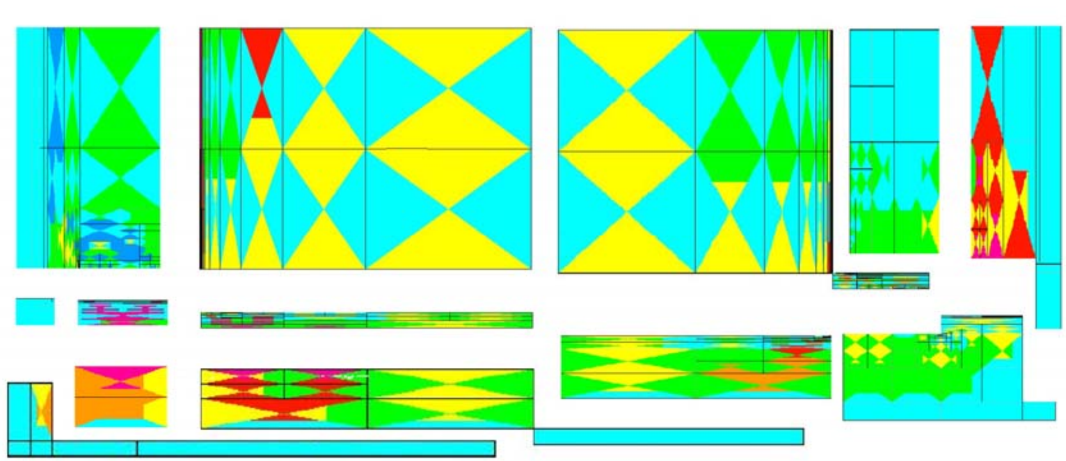
Exponential convergence is obtained to the accuracy of 1 % relative error.



RESULTS
THE BATTERY PROBLEM

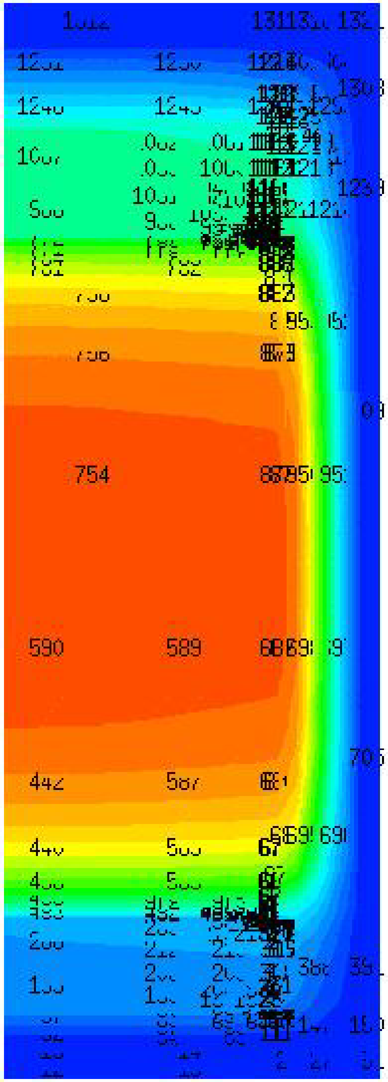


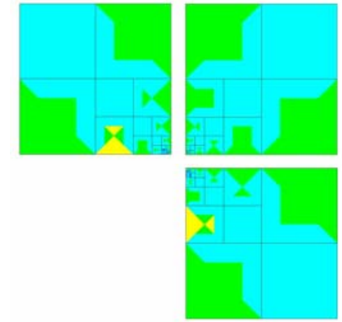


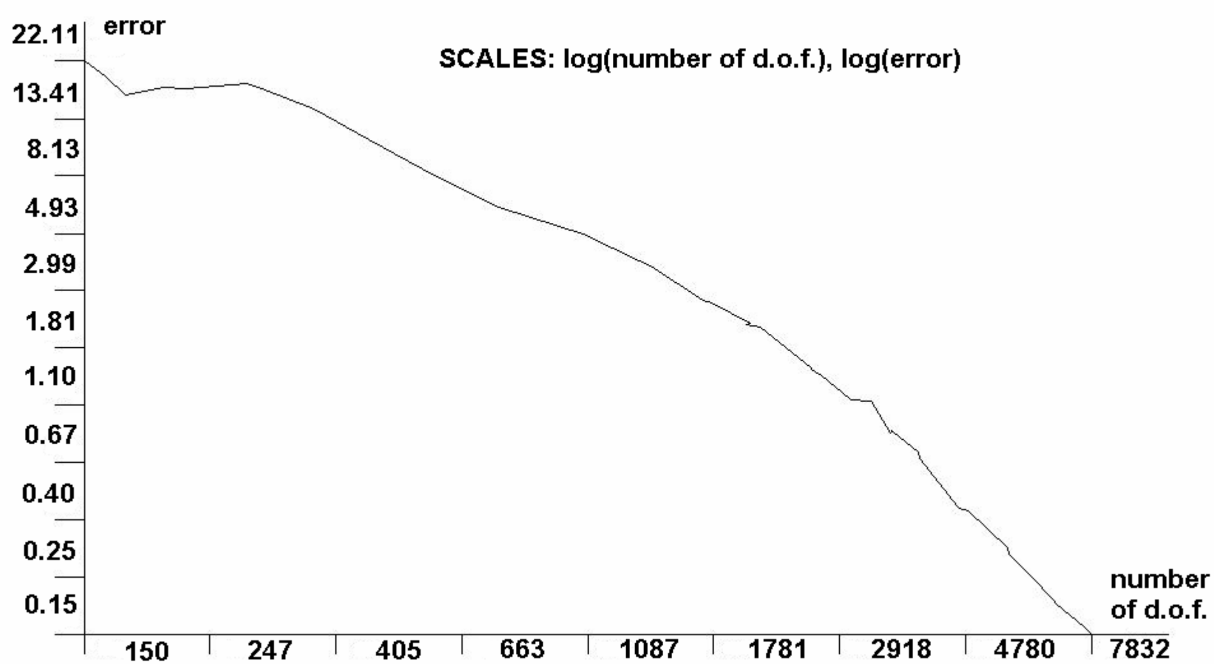


Optimal mesh obtained after parallel iterations over 15 subdomains giving the accuracy of 0.1 % relative error.

RESULTS
THE BATTERY PROBLEM

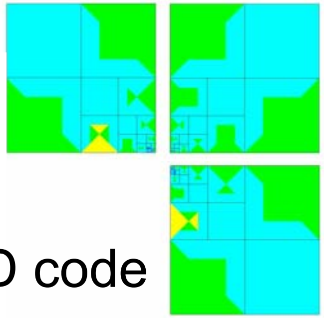






The solution with the accuracy of 0.1% relative error.

Exponential convergence curve for the parallel execution (16 processors)



CONCLUSIONS

We have developed the parallel fully automatic hp-adaptive 2D code for the Laplace equation, where

- Load balancing is performed by ZOLTAN library
- Both coarse and fine mesh problems are solved by the parallel frontal solver
- Mesh is refined fully in parallel

Future work will include:

- Implementation of parallel version of 3D code
- Extending the code to be able to solve 3D Helmholtz and time-harmonic Maxwell equations
- Parallel version of two grid solver
- Challenging applications:

  Simulation of EM waves in the human head

  Calculation of the Radar Cross-sections (3D scattering problems)

  Simulation of Logging While Drilling EM measuring devices