APPLIED MATHEMATICS
NAVAL POSTGRADUATE SCHOOL

ADAPTIVE MESH REFINEMENT

for non-conforming discontinuous Galerkin method

Michal A. Kopera

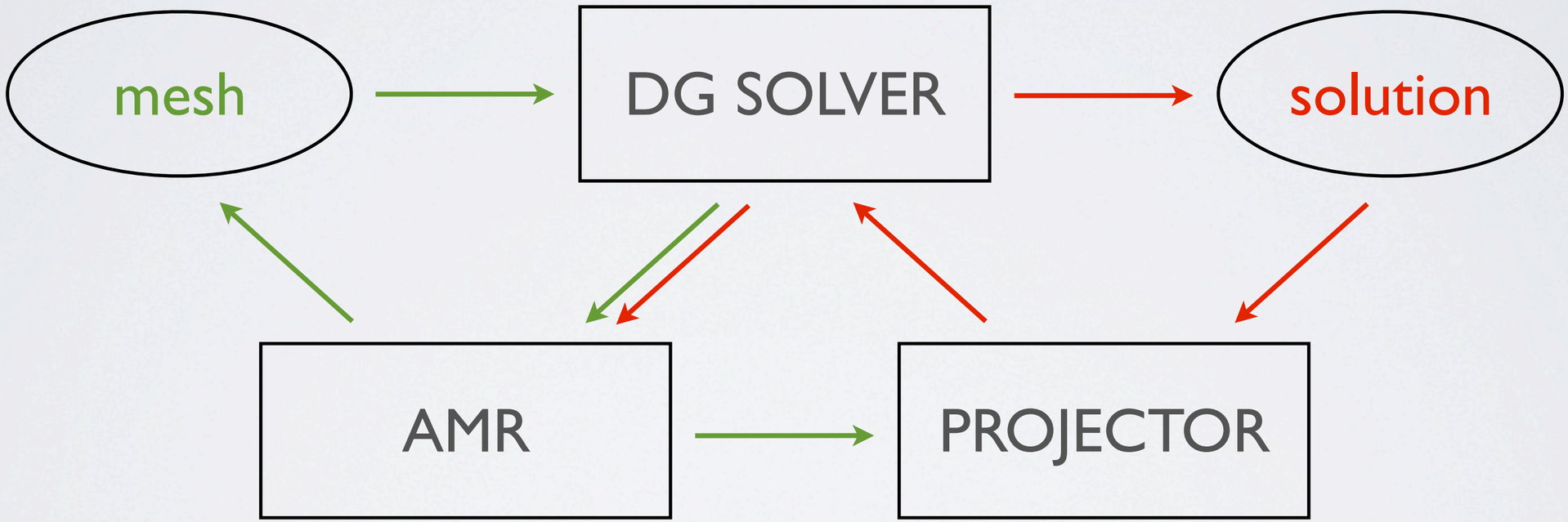
Gene Golub SIAM Summer School 2012

How does **AMR** work?

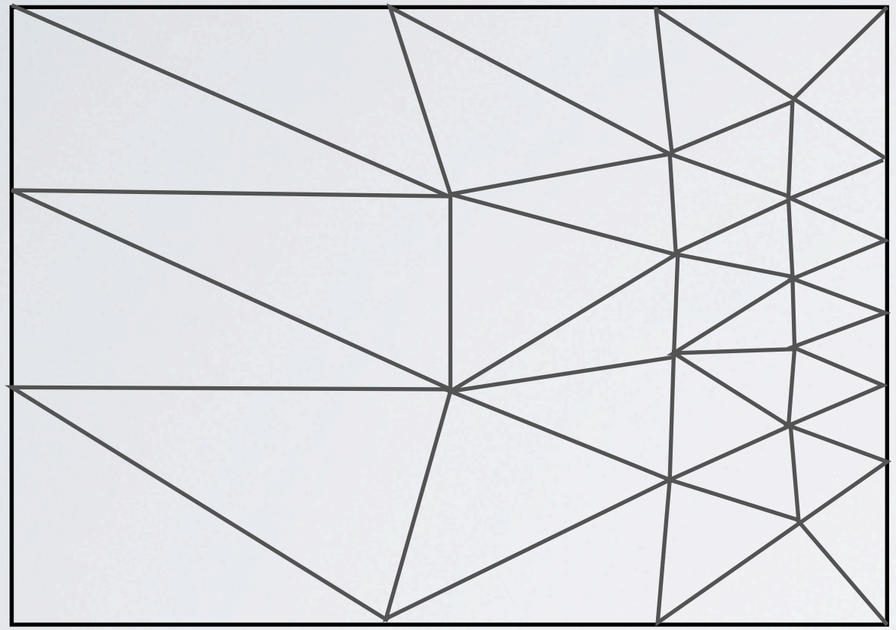
static mesh:

How does **AMR** work?

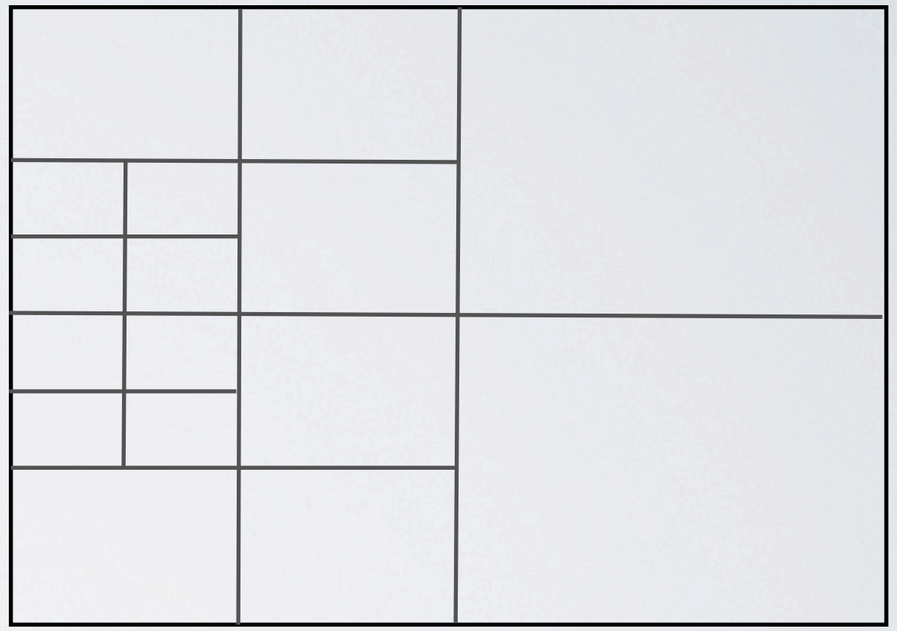
dynamic mesh:



Conforming vs. non-conforming



mesh



CONFORMING

Each edge is shared by at most two elements

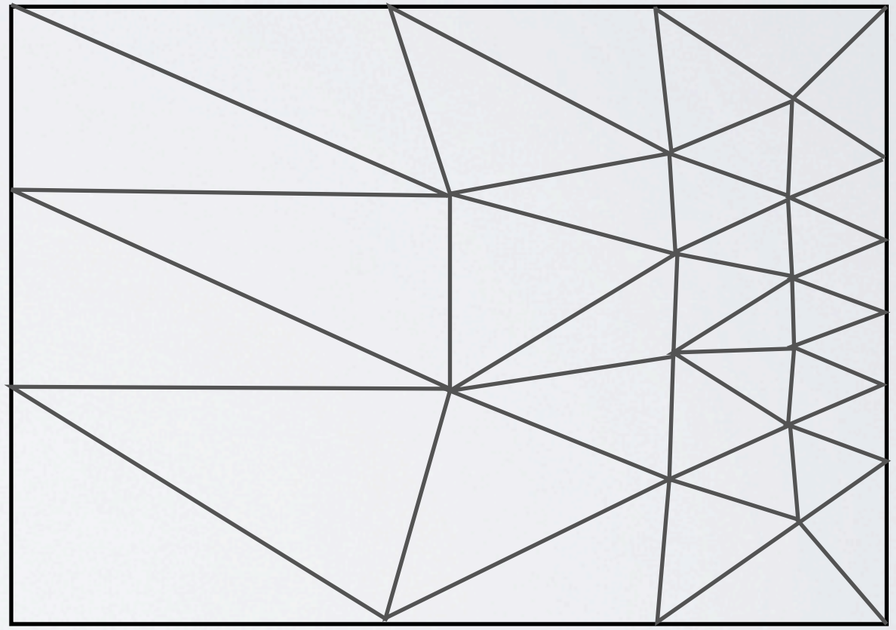
NON-CONFORMING

An edge can be shared by more than two elements

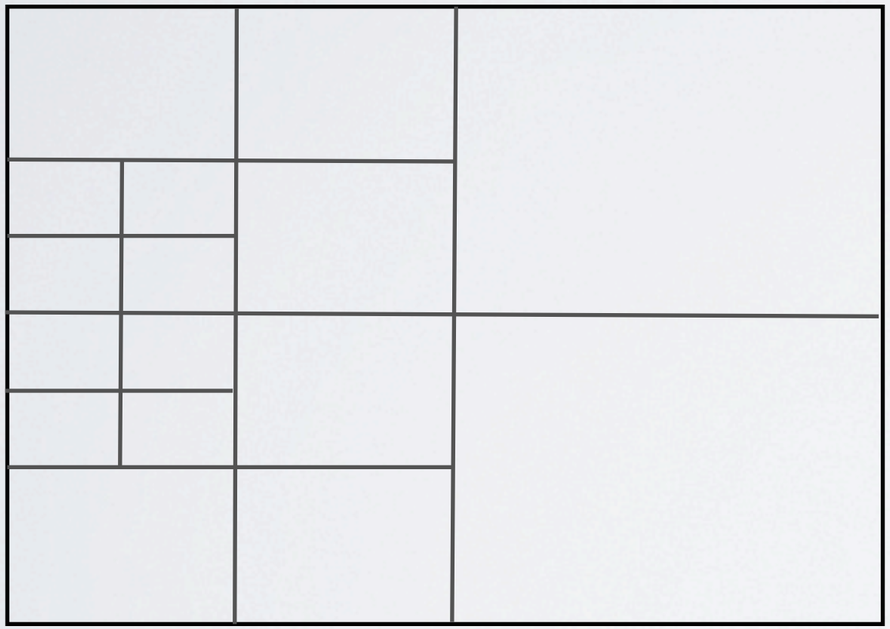
Conforming vs. non-conforming AMR

CONFORMING

- *No changes to flux computation*
- *Possibly more complicated re-meshing strategies*
- *Possibly more difficult projection between meshes*



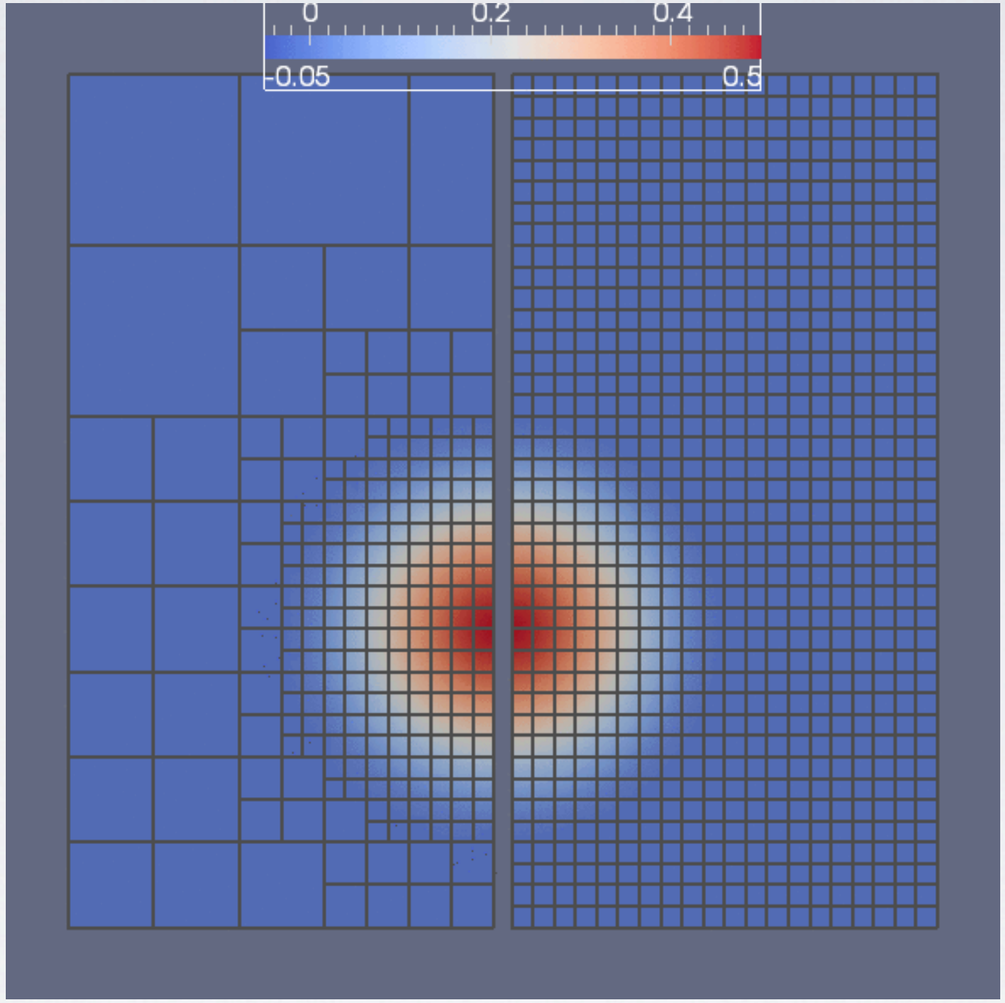
Conforming vs. non-conforming AMR



NON-CONFORMING

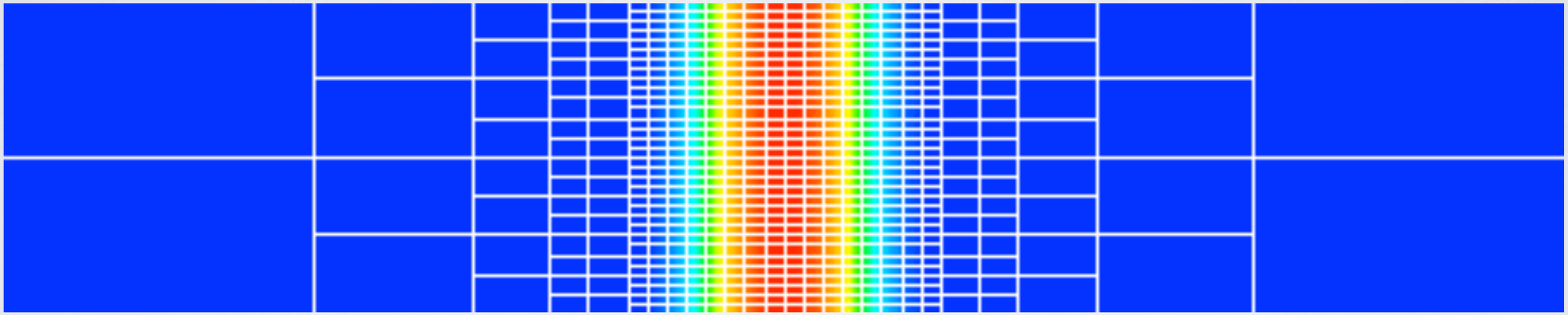
- *Solver needs to handle the non-conforming flux computation*
- *Straightforward re-meshing strategy*
- *Can take advantage of the tree structure*

Examples of non-conforming AMR



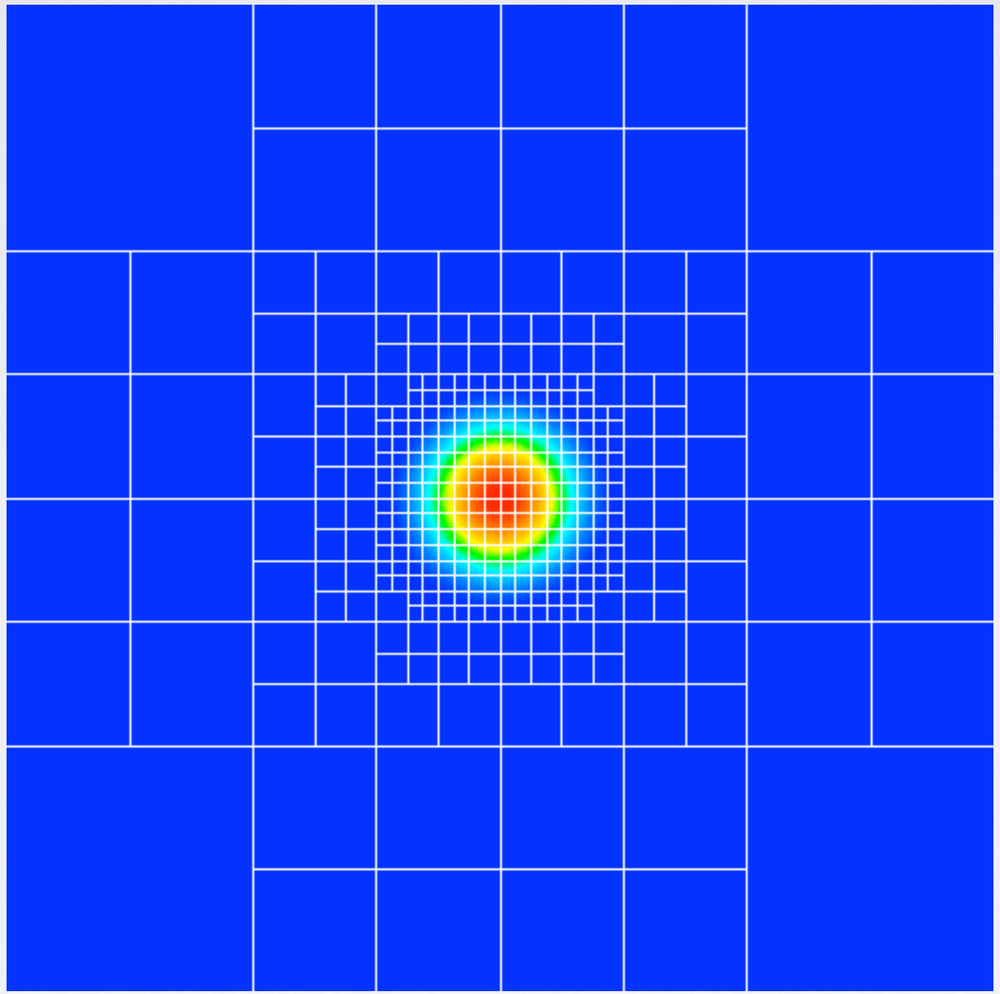
Rising thermal bubble

Examples of non-conforming AMR



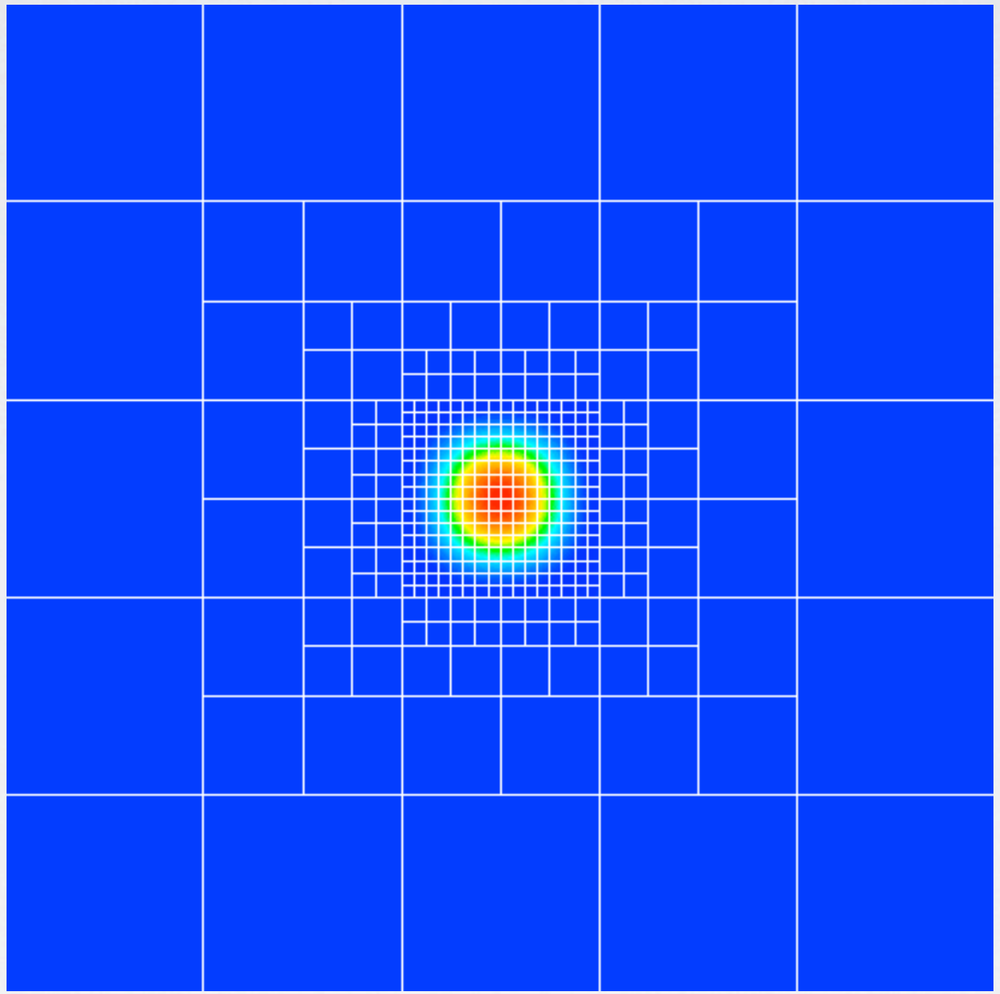
Shallow Water Equations
1D wave with 1D bathymetry

Examples of non-conforming AMR

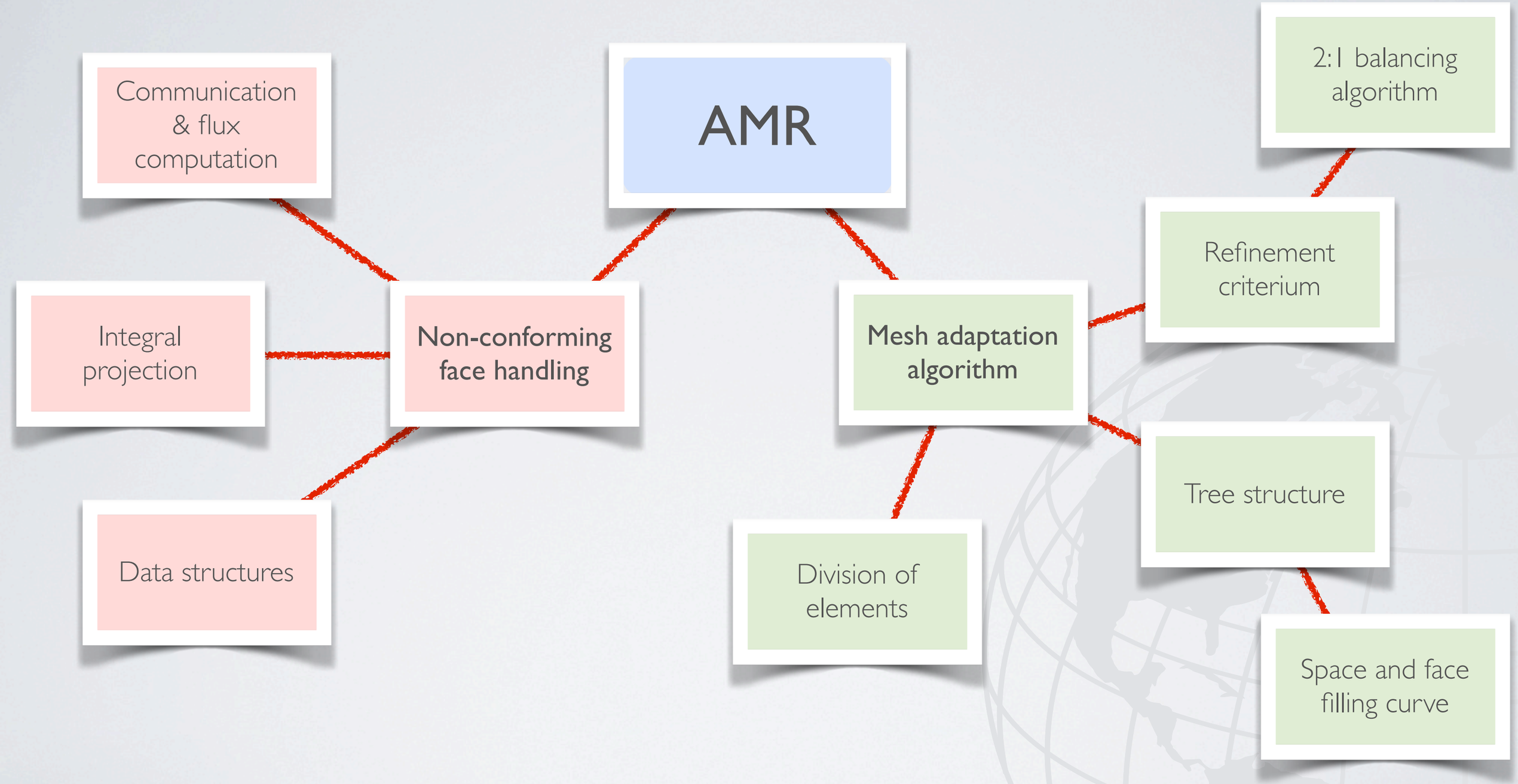


Shallow Water Equations
2D Wave

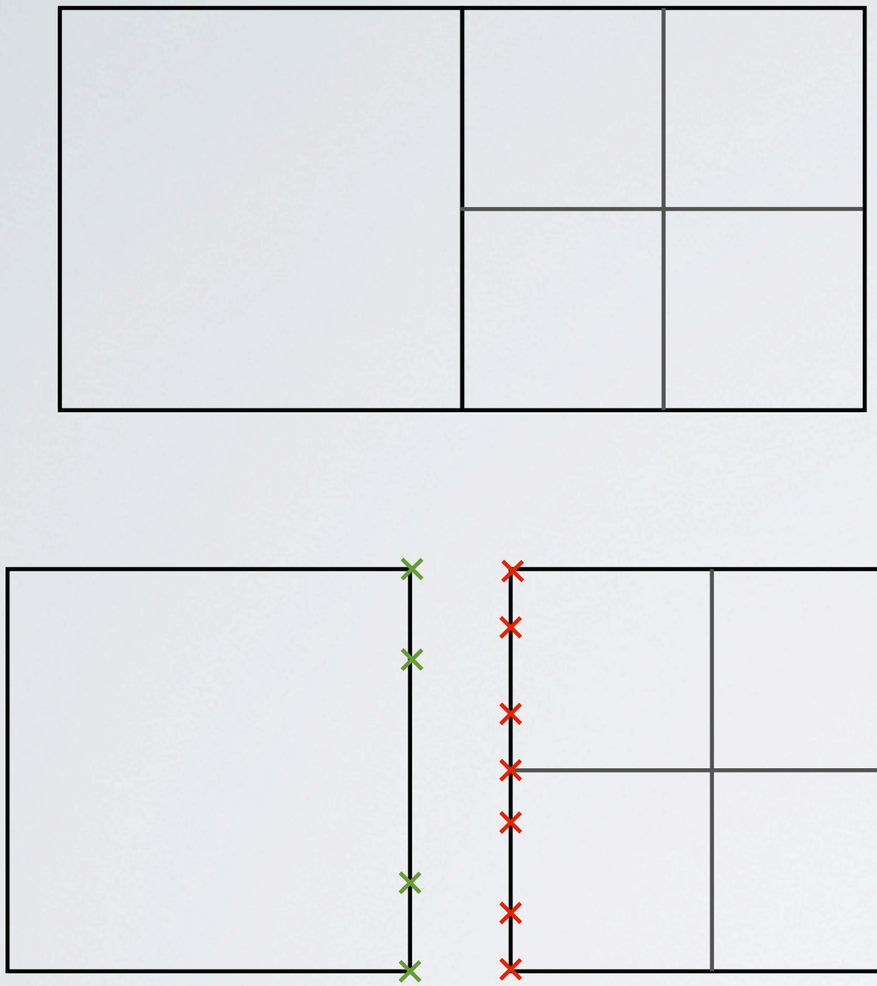
Examples of non-conforming AMR

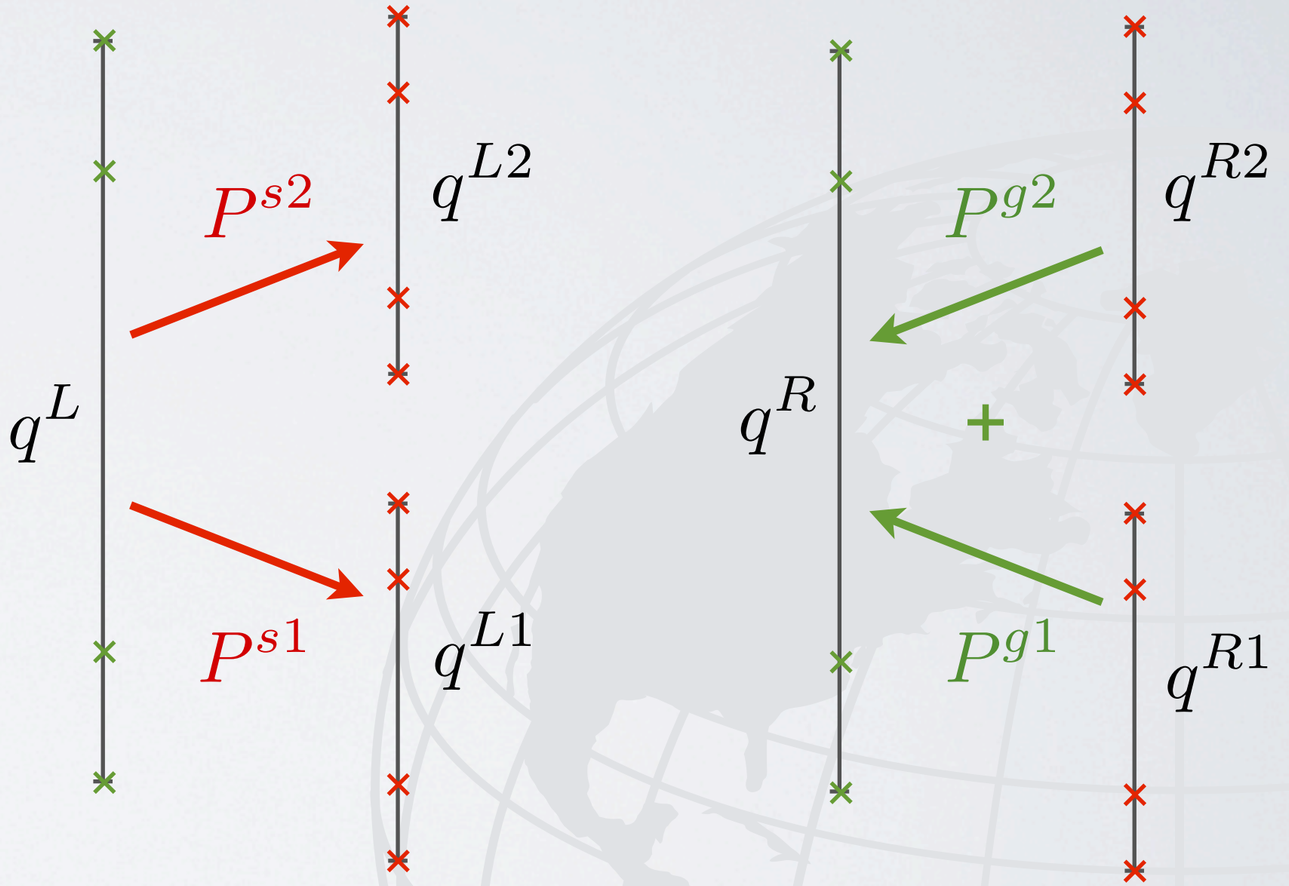


Shallow Water Equations
2D Wave with 2D bathymetry


AMR
Communication & flux computation
Integral projection
Non-conforming face handling
Data structures
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

Non-conforming face handling





Integral projection

Let us define the space for both parent and child faces:

$$\xi, z^{(k)} \in [-1, 1], \quad k = 1, 2$$

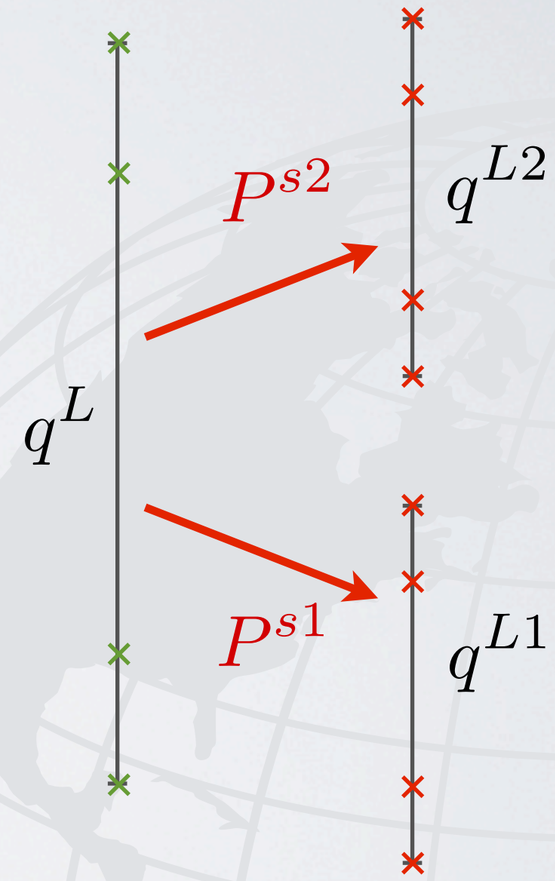
with mappings

$$z^{(k)} = \frac{\xi - o^{(k)}}{s}, \qquad \xi = s \cdot z^{(k)} + o^{(k)}.$$

Expanding variables yields

$$q^L(\xi) = \sum_{j=0}^{N} q_j^L \psi_j(\xi) \quad \Rightarrow \quad q^L(z^{(k)}) = \sum_{j=0}^{N} q_j^L \psi_j(s \cdot z^{(k)} + o^{(k)}),$$

$$q^{Lk}(z^{(k)}) = \sum_{j=0}^{N} q_j^{Lk} \psi_j(z^{(k)}).$$



Integral projection

For each children face we require

$$\int_{-1}^{1} \left(q^{Lk}(z^{(k)}) - q^{L}(s \cdot z^{(k)} + o^{(k)}) \right) \psi_i(z^{(k)}) \; dz^{(k)} = 0$$

Substitution of expansions gives

$$\int_{-1}^{1} \left(\sum_{j=0}^{N} q_j^{Lk} \psi_j(z^{(k)}) - \sum_{j=0}^{N} q_j^{L} \psi_j(s \cdot z^{(k)} + o^{(k)}) \right) \psi_i(z^{(k)}) \; dz^{(k)} = 0$$

Reorganizing the terms yields

$$\sum_{j=0}^{N} \int_{-1}^{1} \psi_j(z^{(k)}) \psi_i(z^{(k)}) dz^{(k)} \; q_j^{Lk} - \sum_{j=0}^{N} \int_{-1}^{1} \psi_j(s \cdot z^{(k)} + o^{(k)}) \psi_i(z^{(k)}) dz^{(k)} \; q_j^{L} = 0$$

Integral projection

We notice that:

$$\sum_{j=0}^{N} \boxed{\int_{-1}^{1} \psi_j(z^{(k)})\psi_i(z^{(k)})dz^{(k)}} q_j^{Lk} - \sum_{j=0}^{N} \boxed{\int_{-1}^{1} \psi_j(s \cdot z^{(k)} + o^{(k)})\psi_i(z^{(k)})dz^{(k)}} q_j^{L} = 0$$

M_{ij} S_{ij}

Therefore:

$$M_{ij} q_j^{Lk} - S_{ij}^{(k)} q_j^{L} = 0 \quad \Rightarrow \quad q_i^{Lk} = \boxed{M_{il}^{-1} S_{lj}^{(k)}} q_j^{L}$$

P_{ij}^{sk}

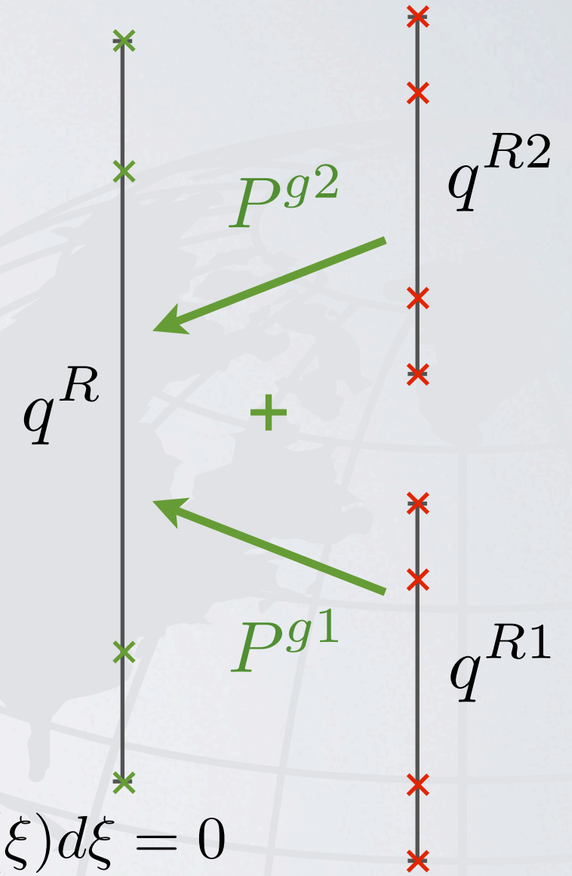
Integral projection

Let

$$\tilde{q}^R(\xi) = \begin{cases} q^{R1}(z^{(1)}) = q^{R1}\left(\frac{\xi - o^{(1)}}{s}\right) & for \quad -1 \le \xi \le 0^-, \\ q^{R2}(z^{(2)}) = q^{R2}\left(\frac{\xi - o^{(2)}}{s}\right) & for \quad 0^+ \le \xi \le 1. \end{cases}$$

We require that

$$\int_{-1}^{1} \left(q^R(\xi) - \tilde{q}^R(\xi)\right) \psi_i(\xi) d\xi = 0$$



Splitting into two integrals gives:

$$\int_{-1}^{0} \left(q^R(\xi) - q^{R1}\left(\frac{\xi - o^{(1)}}{s}\right)\right) \psi_i(\xi) d\xi + \int_{0}^{1} \left(q^R(\xi) - q^{R2}\left(\frac{\xi - o^{(2)}}{s}\right)\right) \psi_i(\xi) d\xi = 0$$

$$\int_{-1}^{0}\left(q^{R}(\xi)-q^{R1}\left(\frac{\xi-o^{(1)}}{s}\right)\right)\psi_i(\xi)d\xi+\int_{0}^{1}\left(q^{R}(\xi)-q^{R2}\left(\frac{\xi-o^{(2)}}{s}\right)\right)\psi_i(\xi)d\xi=0$$

Expanding the variables gives:

$$\sum_{j=0}^{N}\left(\int_{-1}^{0}\psi_i(\xi)\psi_j(\xi)d\xi\right)q_j^{R}-\sum_{j=0}^{N}\left(\int_{-1}^{0}\psi_i(\xi)\psi_j\left(\frac{\xi-o^{(1)}}{s}\right)d\xi\right)q_j^{R1}+$$

$$+\sum_{j=0}^{N}\left(\int_{0}^{1}\psi_i(\xi)\psi_j(\xi)d\xi\right)q_j^{R}-\sum_{j=0}^{N}\left(\int_{0}^{1}\psi_i(\xi)\psi_j\left(\frac{\xi-o^{(2)}}{s}\right)d\xi\right)q_j^{R2}=0,$$

$$\left(\int_{-1}^{1}\psi_i(\xi)\psi_j(\xi)d\xi\right)q_j^{R}$$

$$M_{ij}$$

$$M_{ij}q_j^R - \left(\int_{-1}^{0} \psi_i(\xi)\psi_j\left(\frac{\xi - o^{(1)}}{s}\right)d\xi\right) q_j^{R1} - \left(\int_{0}^{1} \psi_i(\xi)\psi_j\left(\frac{\xi - o^{(2)}}{s}\right)d\xi\right) q_j^{R2} = 0$$

Introducing the variable change $\xi = s \cdot z + o^{(k)}, \quad d\xi = s \cdot dz$

we get:

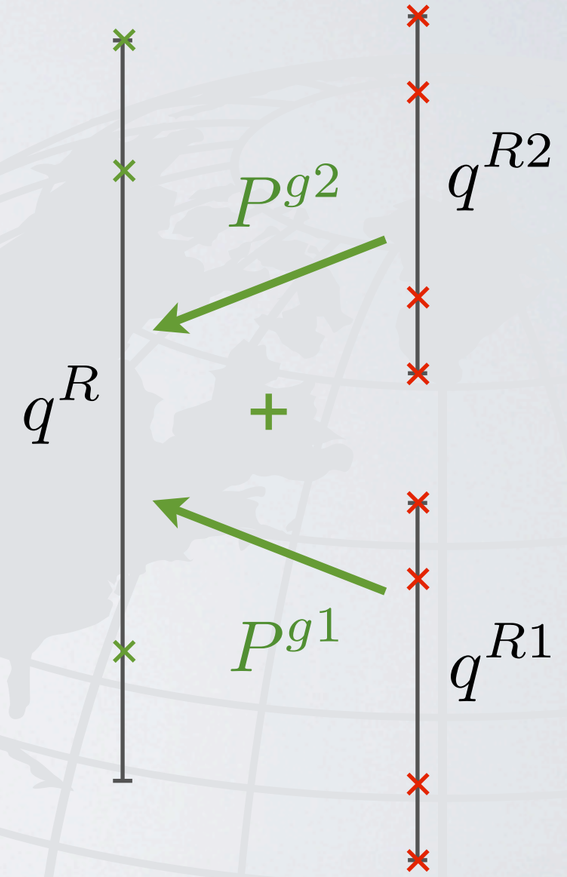
$$M_{ij}q_j^R - s\sum_{k=1}^{2}\boxed{\left(\int_{-1}^{1} \psi_i(s \cdot z + o^{(k)})\psi_j(z)dz\right)} q_j^{Rk} = 0.$$

$$S_{ij}^{(k)^T}$$

Therefore

$$q_i^R = \sum_{k=1}^{2}\boxed{s\mathbf{M}_{il}^{-1}\mathbf{S}_{lj}^{(k)^T}} q_j^{Rk}.$$

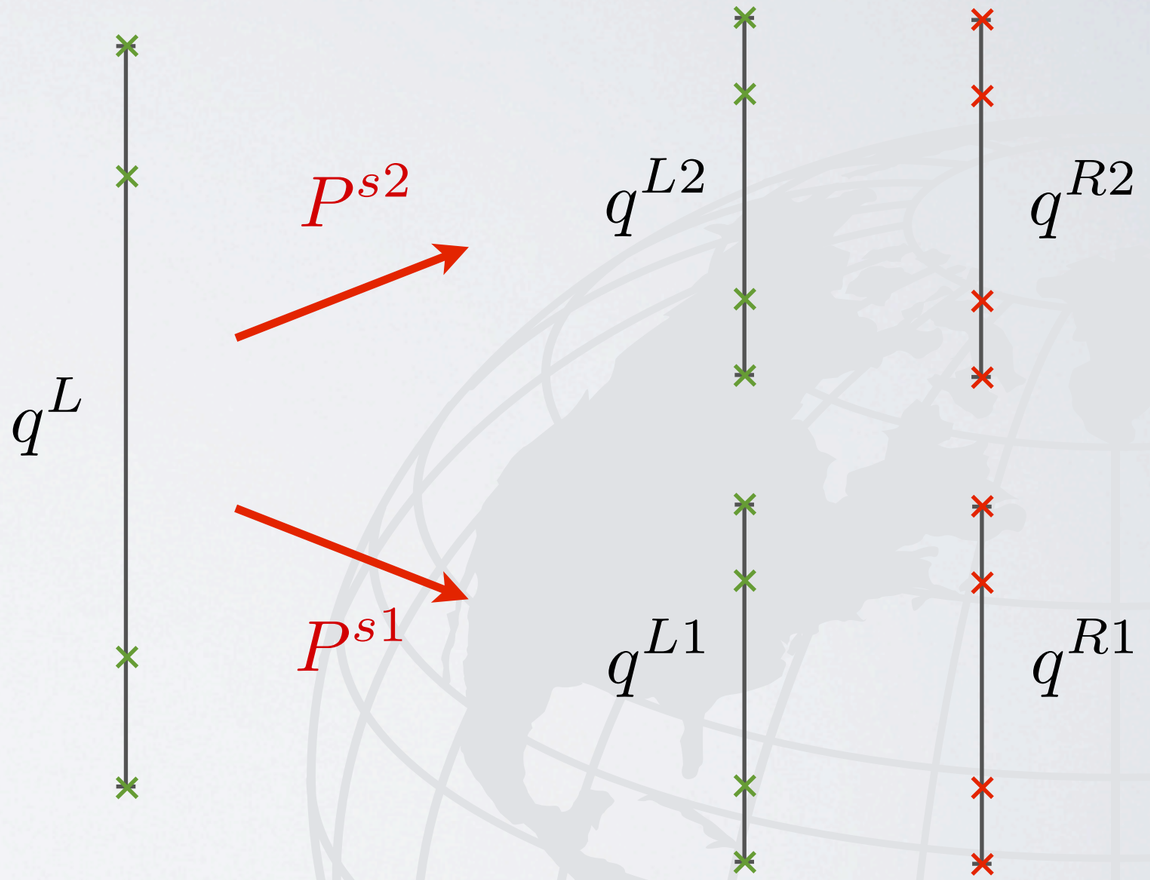
$$P_{ij}^{gk}$$



How to compute flux?

1) *Scatter data from the parent edge to children edges*

Communication & flux computation

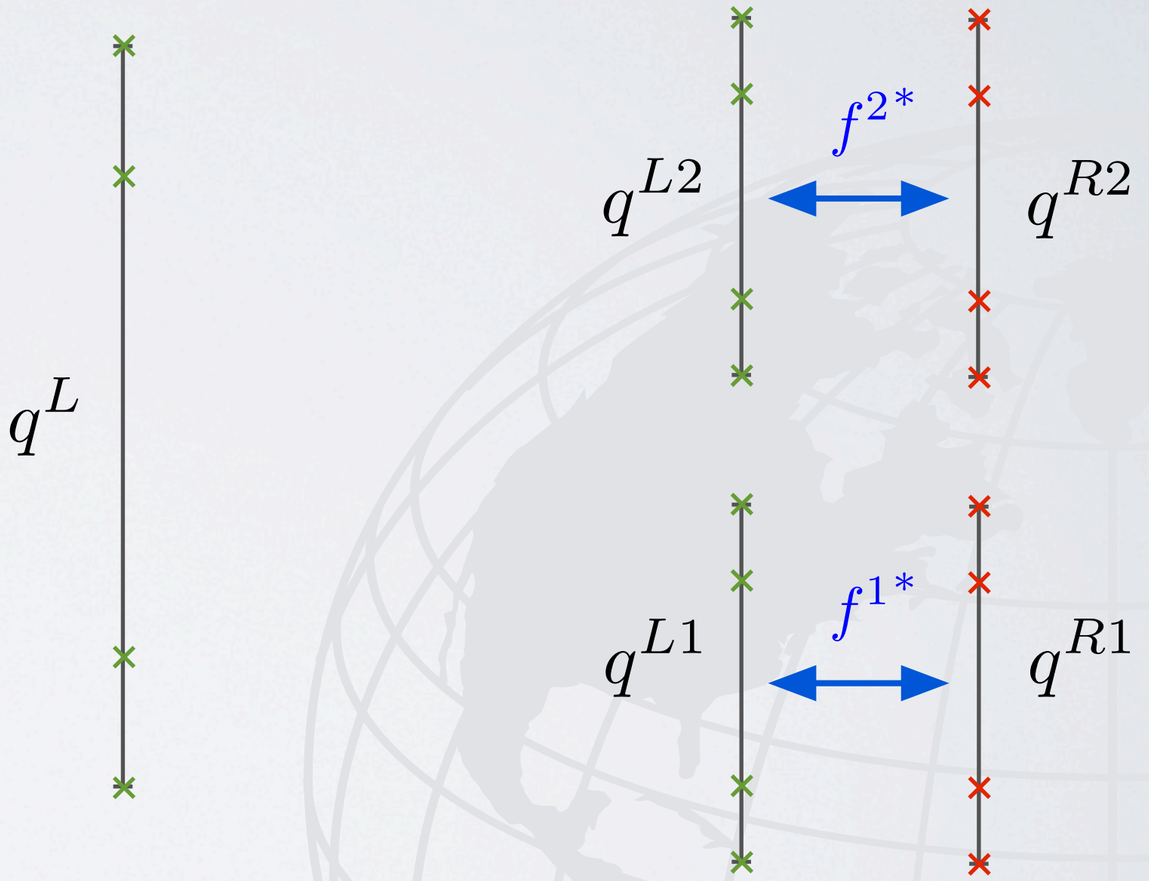


How to compute flux?

1) *Scatter data from the parent edge to children edges*

2) *Compute flux on children edges like in a conforming case*

Communication & flux computation



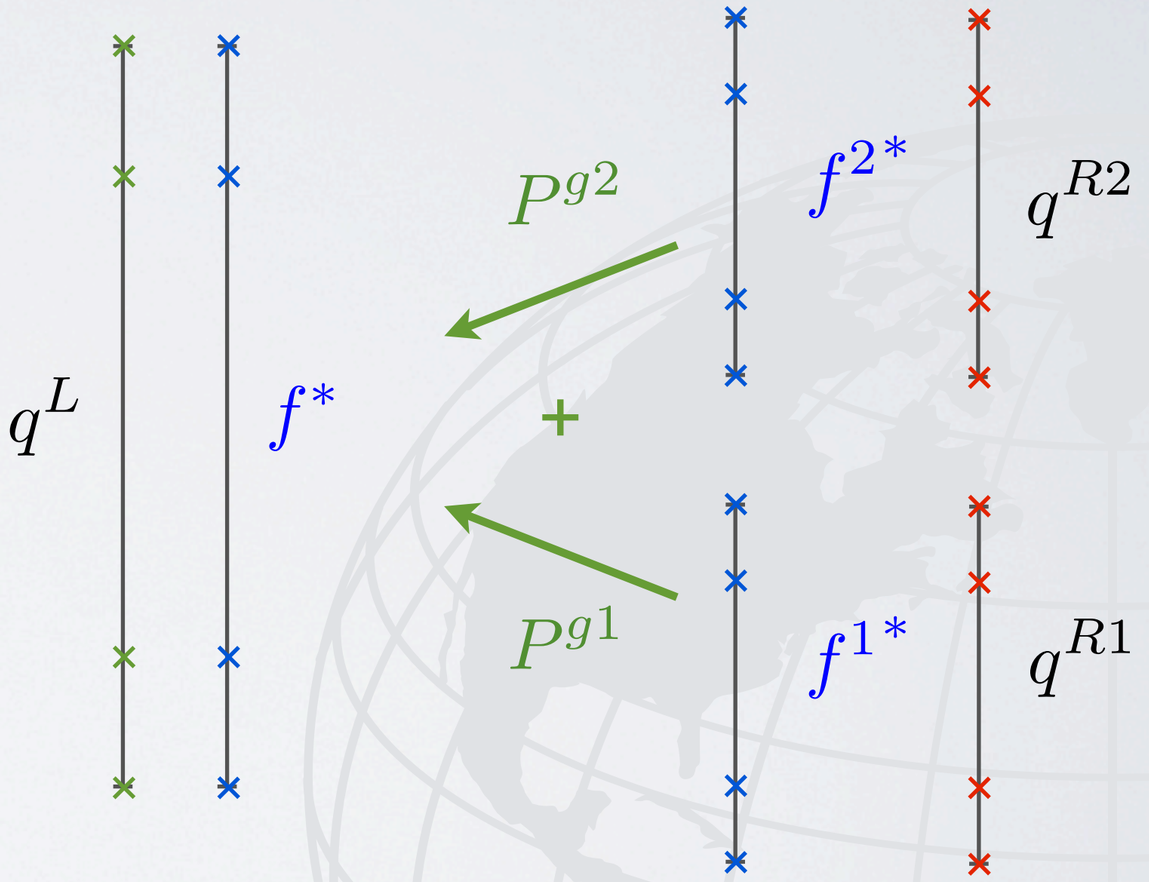
How to compute flux?

1) *Scatter data from the parent edge to children edges*

2) *Compute flux on children edges like in a conforming case*

3) *Gather fluxes from children edges to the parent edge*

Communication & flux computation



How to compute flux?

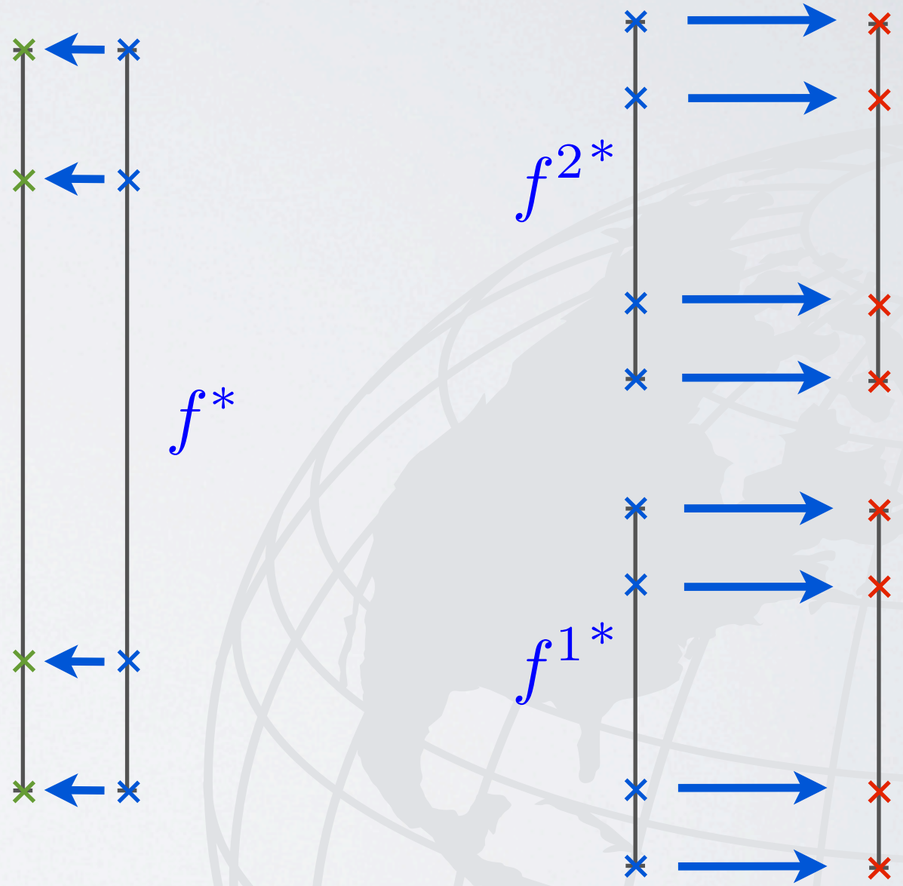
Communication & flux computation

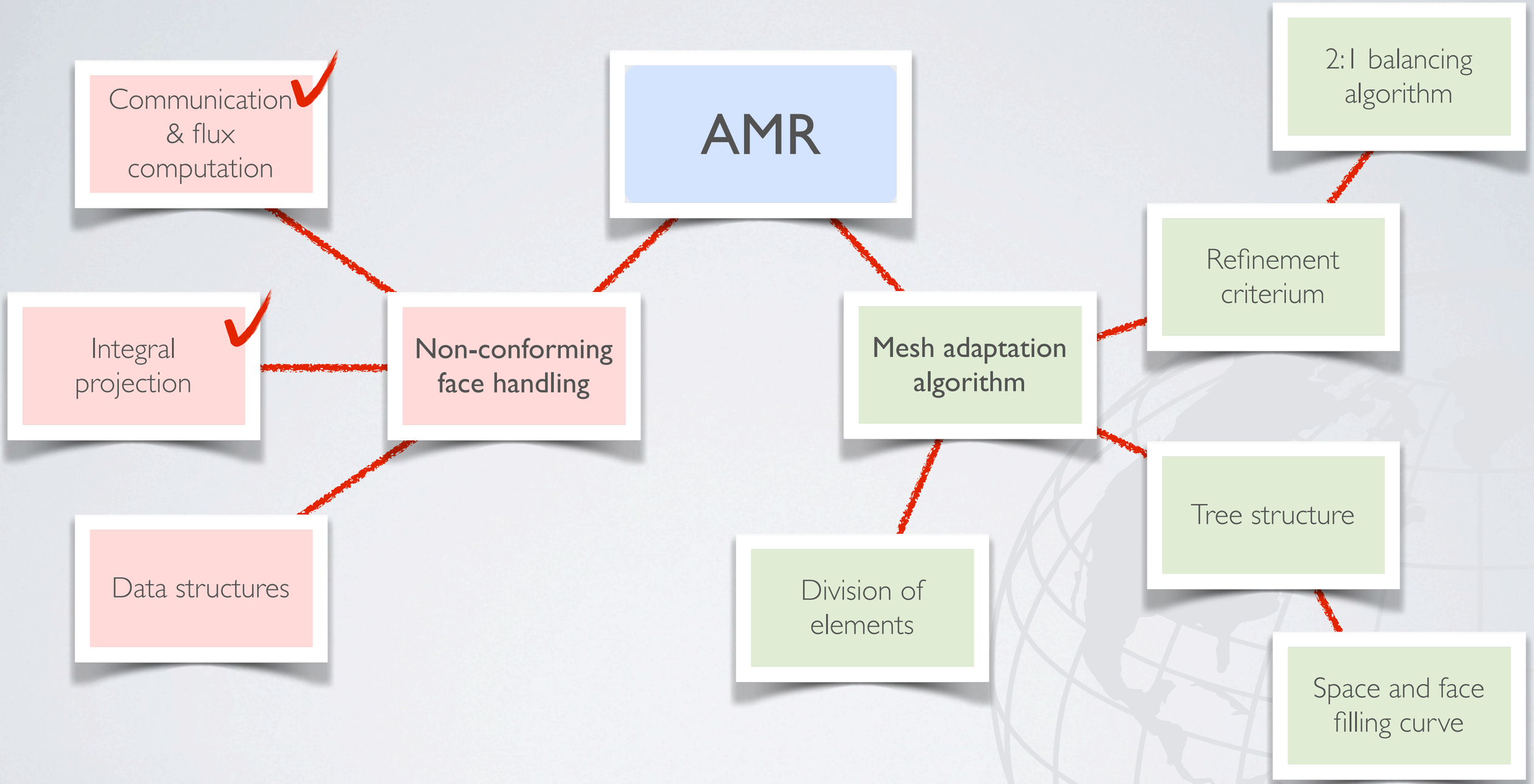
1) *Scatter data from the parent edge to children edges*

2) *Compute flux on children edges like in a conforming case*

3) *Gather fluxes from children edges to the parent edge*

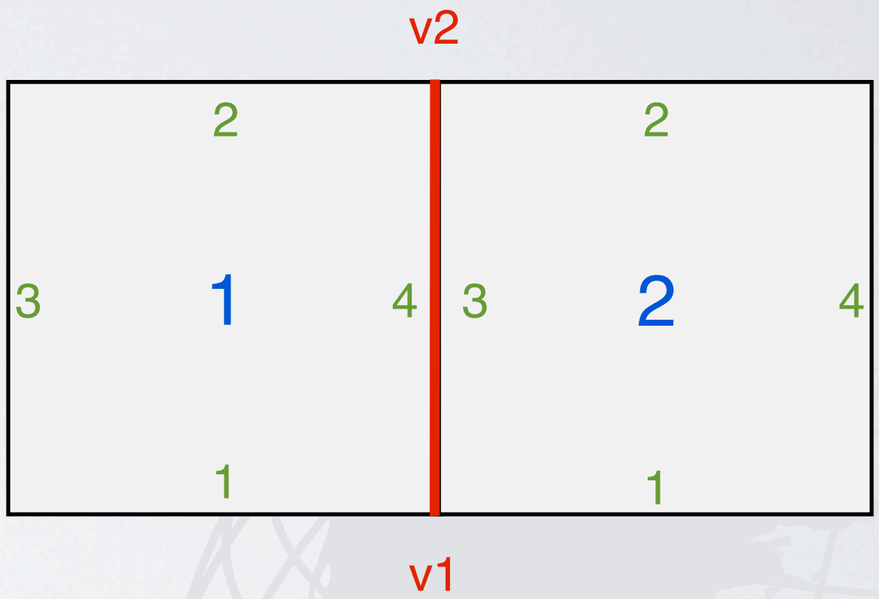
4) *Apply fluxes like in a conforming case*




AMR
Communication & flux computation
Integral projection
Non-conforming face handling
Data structures
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

Data structures

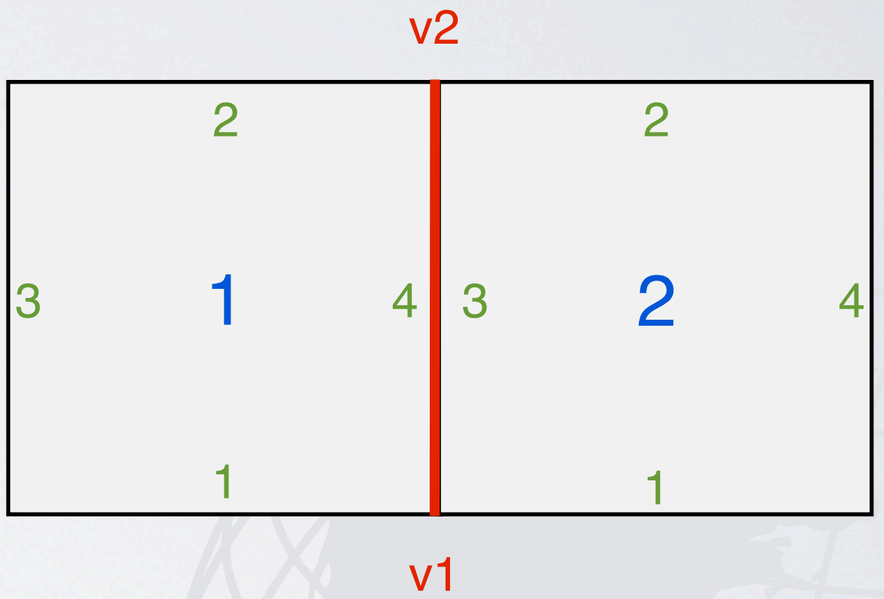
int face(6,nfaceT)
face(1,iface) = v1
face(2,iface) = v2
face(3,iface) = s1 [=4]
face(4,iface) = s2 [=3]
face(5,iface) = e1 [=1]
face(6,iface) = e2 [=2]



Data structures

```
int face(11,nfaceT)
face(1,iface) = v1
face(2,iface) = v2
face(3,iface) = s1 [=4]
face(4,iface) = s2 [=3]
face(5,iface) = e11 [=1]
face(6,iface) = e21 [=2]
-------------------------------
face(7,iface) = e12 [=0]
face(8,iface) = e22 [=0]
face(9,iface) = type [0,1,2]
face(10,iface) = BC_type
face(11,iface) = ngb [periodic BC]

int facepa(nfaceT)
facepa(iface) = [0,1]
```

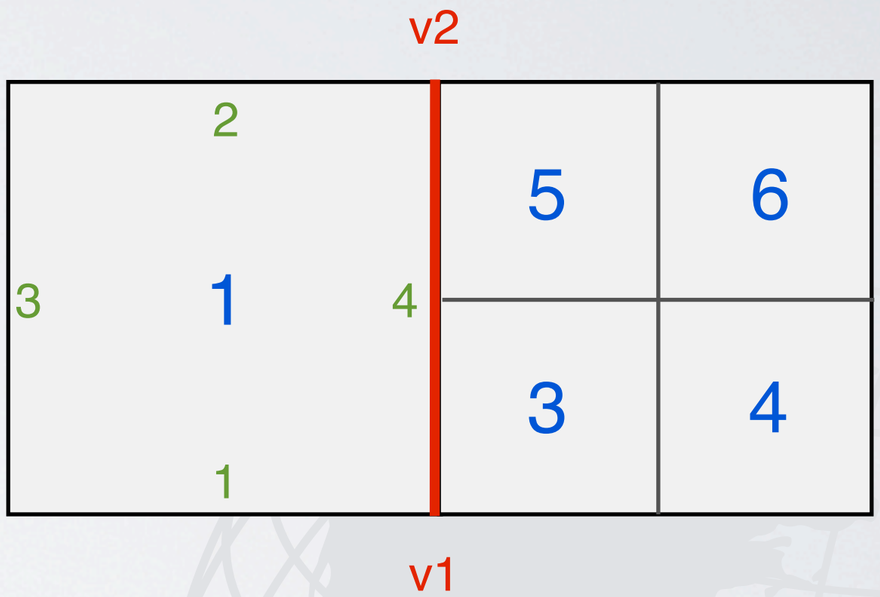


type = 0

Data structures

int face(11,nfaceT)
face(1,iface) = v1
face(2,iface) = v2
face(3,iface) = s1 [=4]
face(4,iface) = s2 [=3]
face(5,iface) = e11 [=1]
face(6,iface) = e21 [=3]
---
face(7,iface) = e12 [=0]
face(8,iface) = e22 [=5]
face(9,iface) = type [0,1,2]
face(10,iface) = BC_type
face(11,iface) = ngb [periodic BC]

int facepa(nfaceT)
facepa(iface) = [0,1]

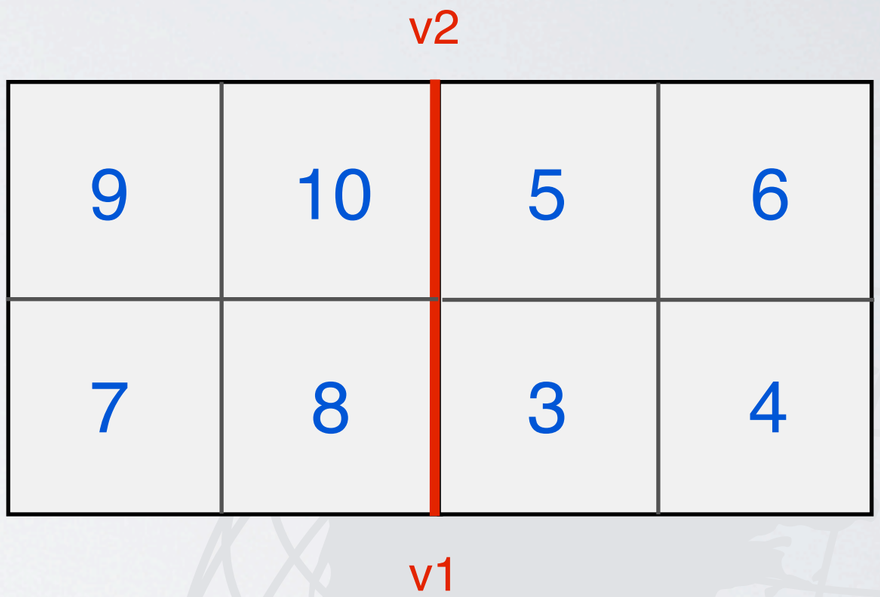


type = 1

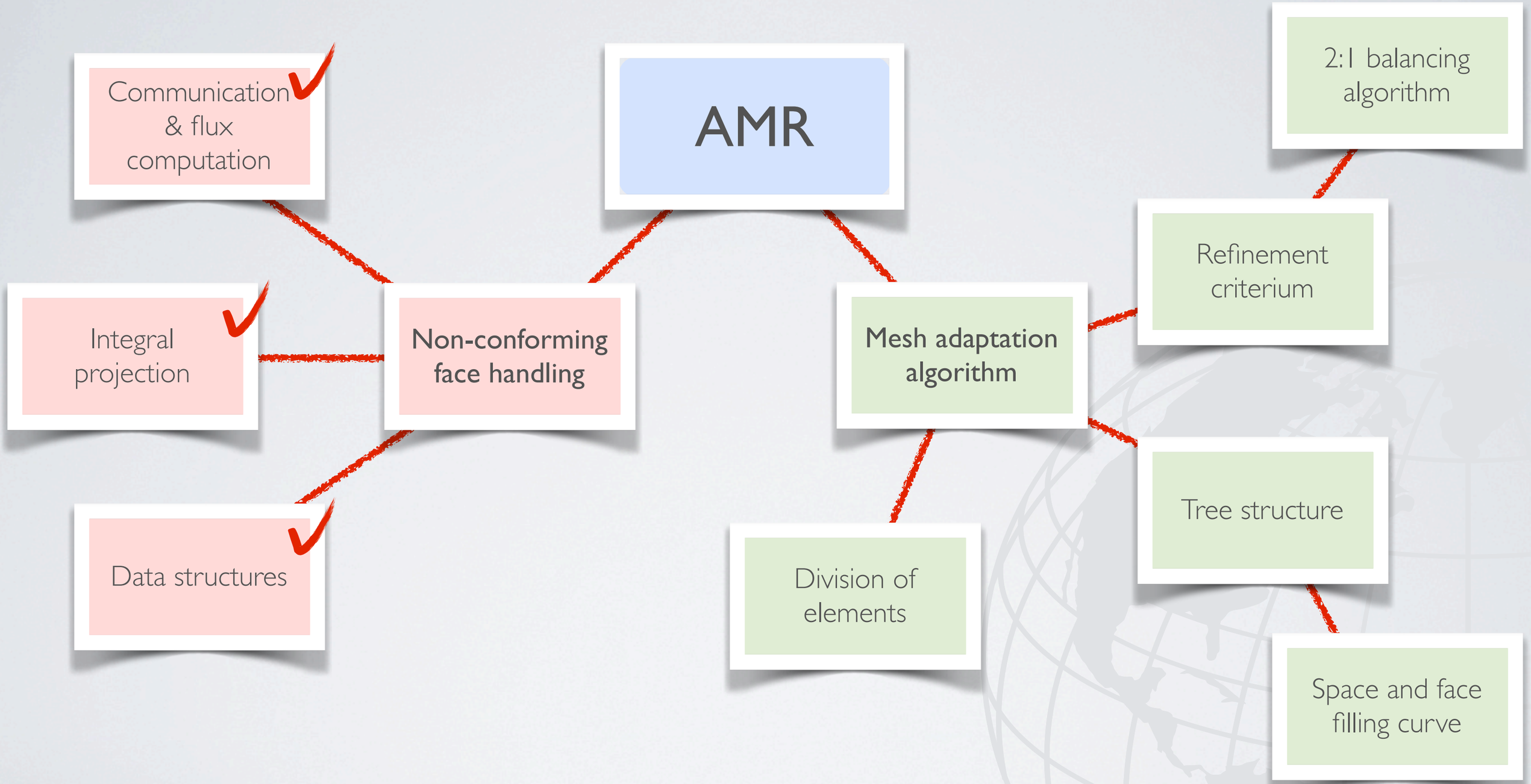
Data structures

```
int face(11,nfaceT)
face(1,iface) = v1
face(2,iface) = v2
face(3,iface) = s1 [=4]
face(4,iface) = s2 [=3]
face(5,iface) = e11 [=8]
face(6,iface) = e21 [=3]
-------------------------------
face(7,iface) = e12 [=10]
face(8,iface) = e22 [=5]
face(9,iface) = type [0,1,2]
face(10,iface) = BC_type
face(11,iface) = ngb [periodic BC]

int facepa(nfaceT)
facepa(iface) = [0,1]
```



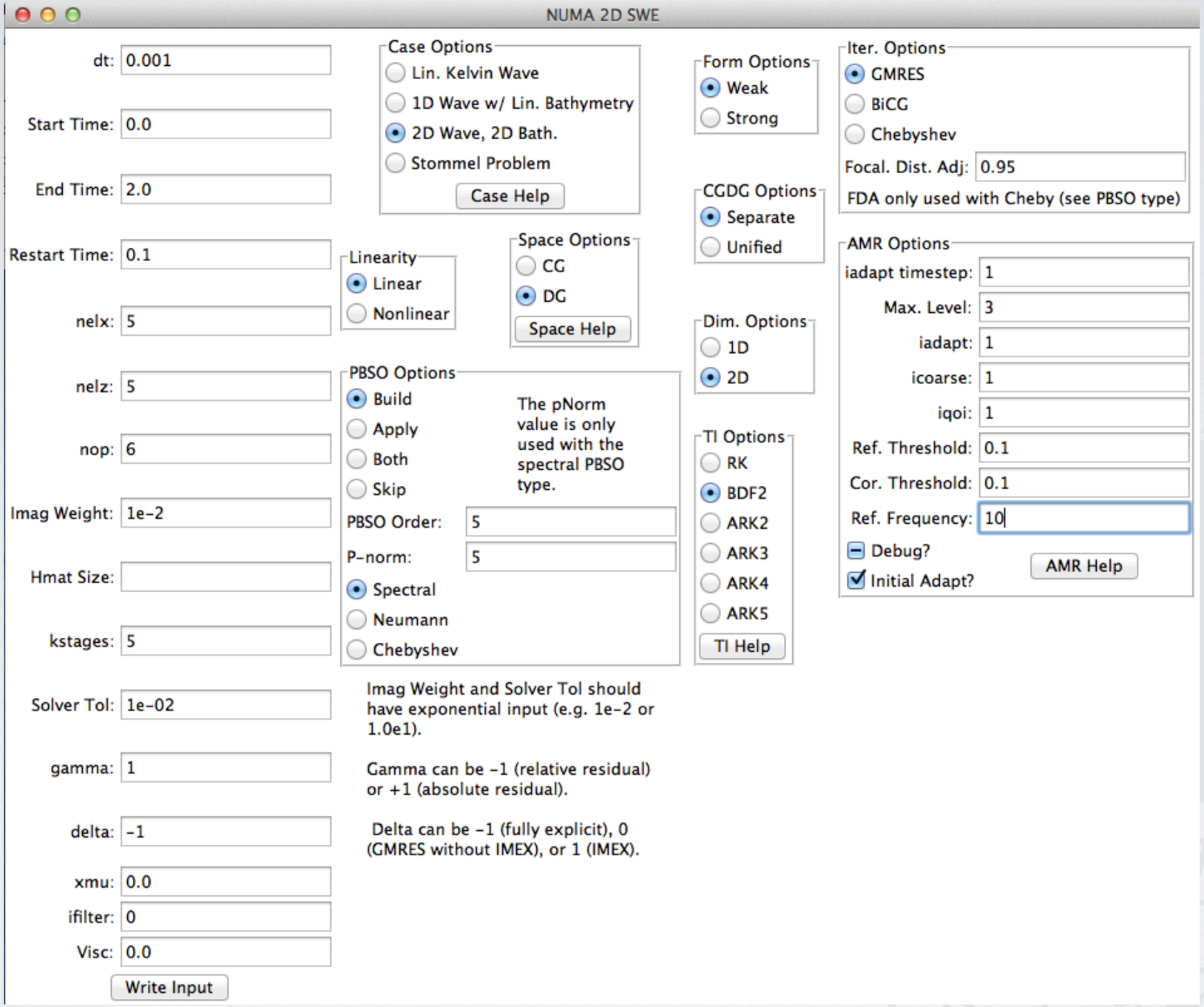
type = 2


AMR
Communication & flux computation
Integral projection
Data structures
Non-conforming face handling
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

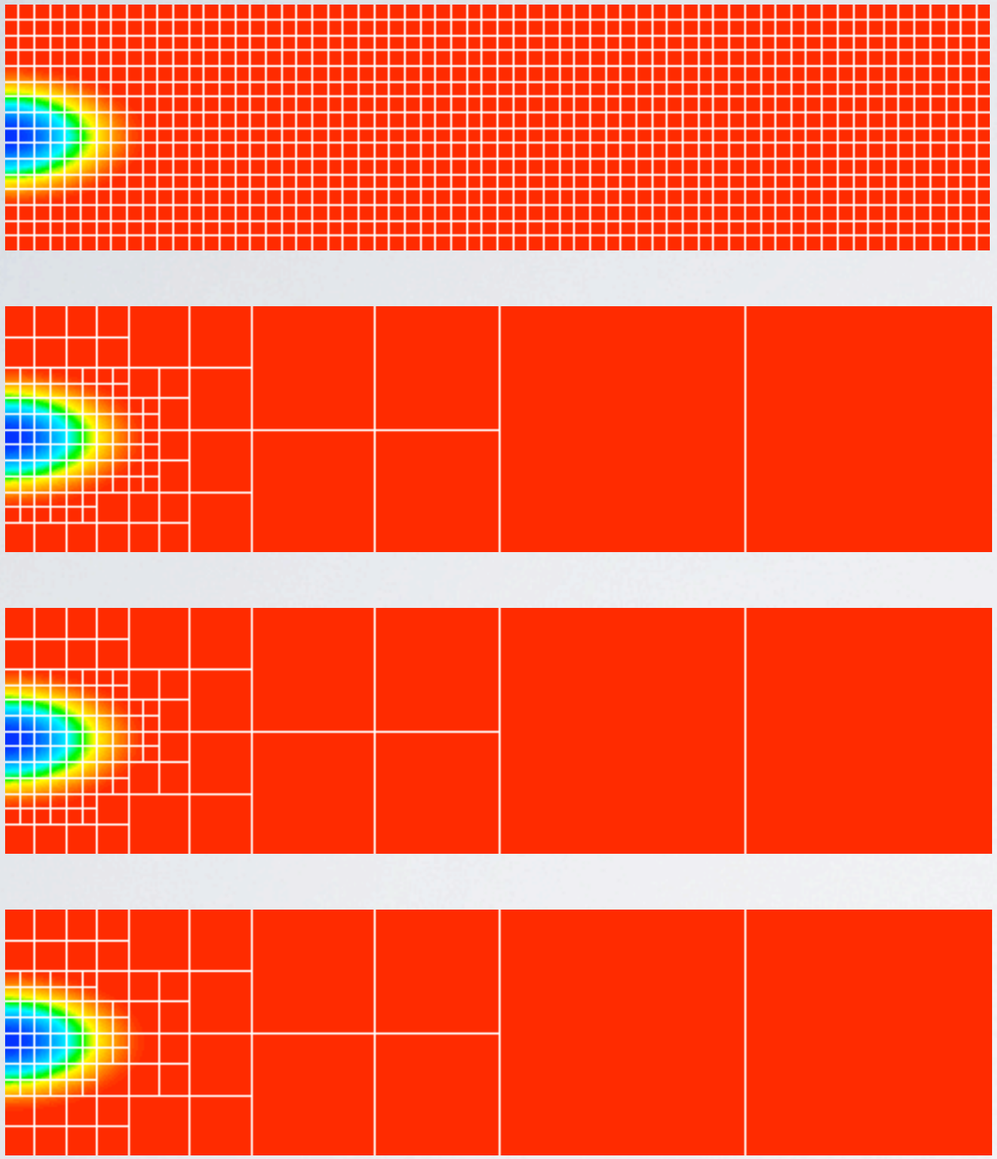
AMR is as good as the ***refinement criterium!***

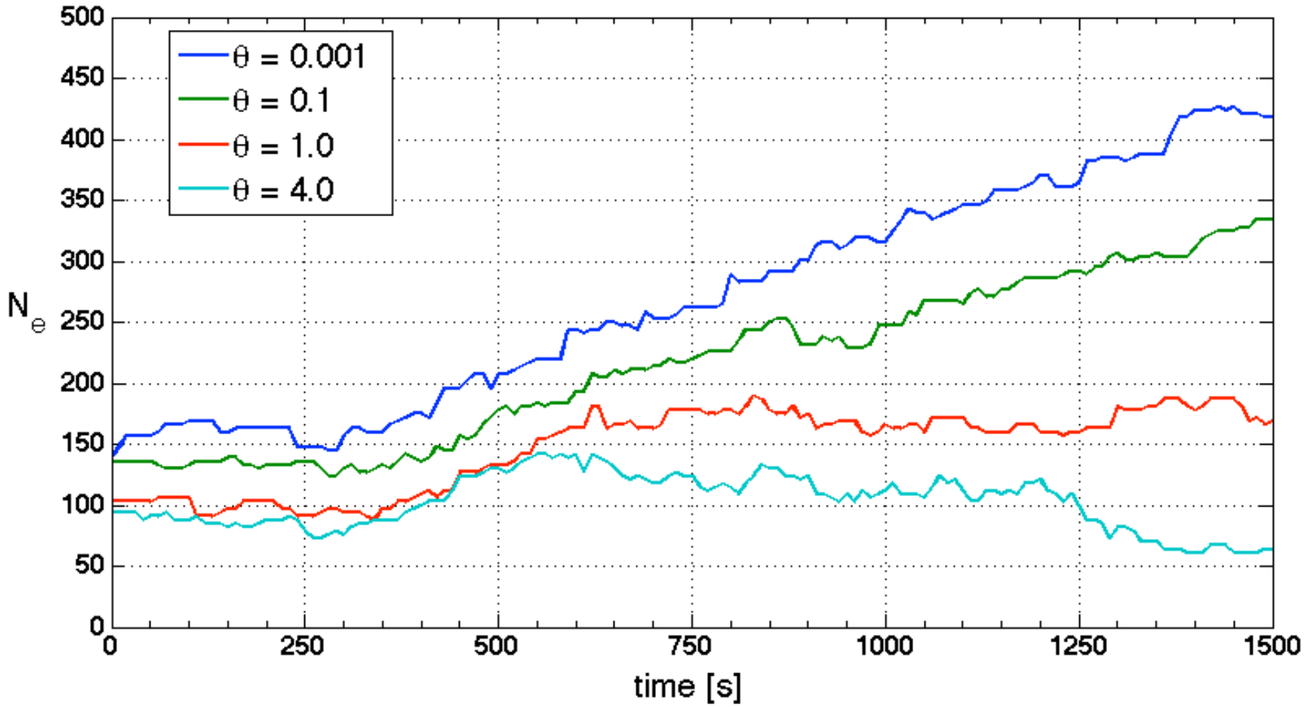
Refinement criterium

- *Criterium needs to be* ***element local***
  - *solution variable*
  - *derived quantity*
  - *mathematical error estimation - residual based*


NUMA 2D SWE
dt: 0.001
Start Time: 0.0
End Time: 2.0
Restart Time: 0.1
nelx: 5
nelz: 5
nop: 6
Imag Weight: 1e-2
Hmat Size:
kstages: 5
Solver Tol: 1e-02
gamma: 1
delta: -1
xmu: 0.0
ifilter: 0
Visc: 0.0
Write Input
Case Options
Lin. Kelvin Wave
1D Wave w/ Lin. Bathymetry
2D Wave, 2D Bath.
Stommel Problem
Case Help
Linearity
Linear
Nonlinear
Space Options
CG
DG
Space Help
PBSO Options
Build
Apply
Both
Skip
The pNorm value is only used with the spectral PBSO type.
PBSO Order: 5
P-norm: 5
Spectral
Neumann
Chebyshev
Imag Weight and Solver Tol should have exponential input (e.g. 1e-2 or 1.0e1).
Gamma can be -1 (relative residual) or +1 (absolute residual).
Delta can be -1 (fully explicit), 0 (GMRES without IMEX), or 1 (IMEX).
Form Options
Weak
Strong
CGDG Options
Separate
Unified
Dim. Options
1D
2D
TI Options
RK
BDF2
ARK2
ARK3
ARK4
ARK5
TI Help
Iter. Options
GMRES
BiCG
Chebyshev
Focal. Dist. Adj: 0.95
FDA only used with Cheby (see PBSO type)
AMR Options
iadapt timestep: 1
Max. Level: 3
iadapt: 1
icoarse: 1
iqoi: 1
Ref. Threshold: 0.1
Cor. Threshold: 0.1
Ref. Frequency: 10
Debug?
Initial Adapt?
AMR Help

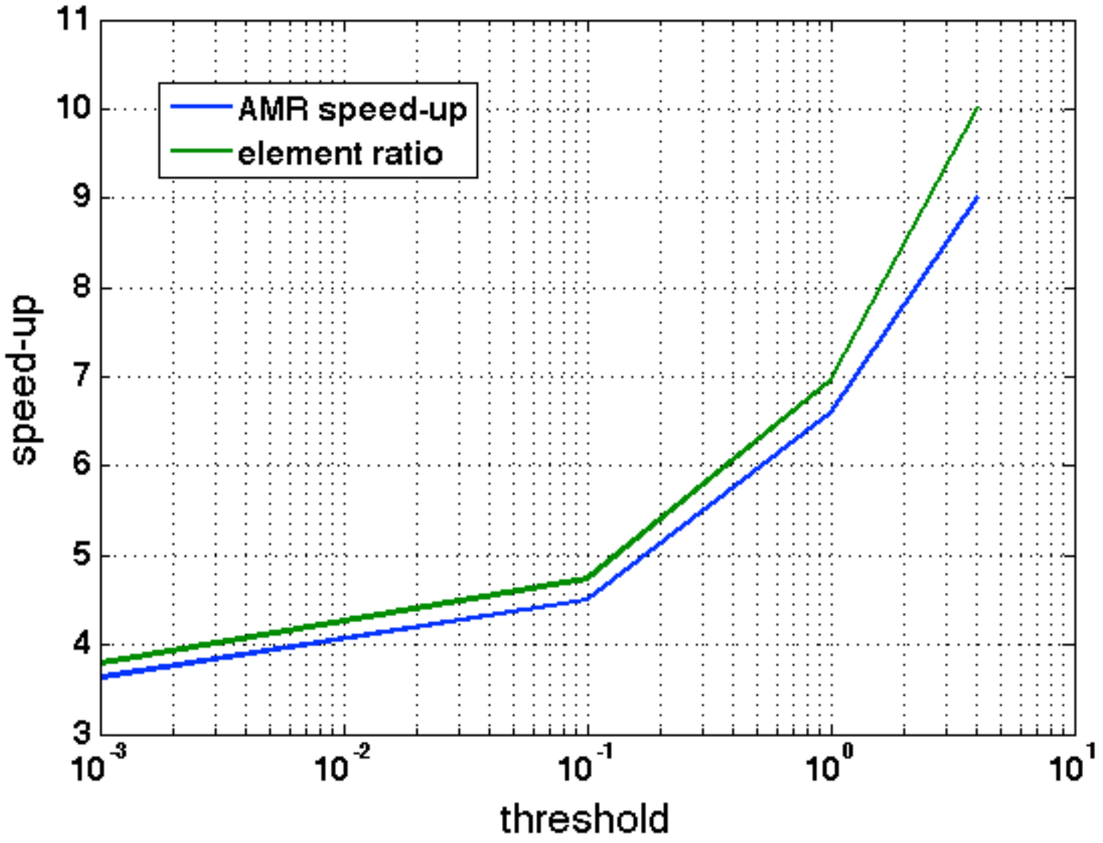
Refinement criterium

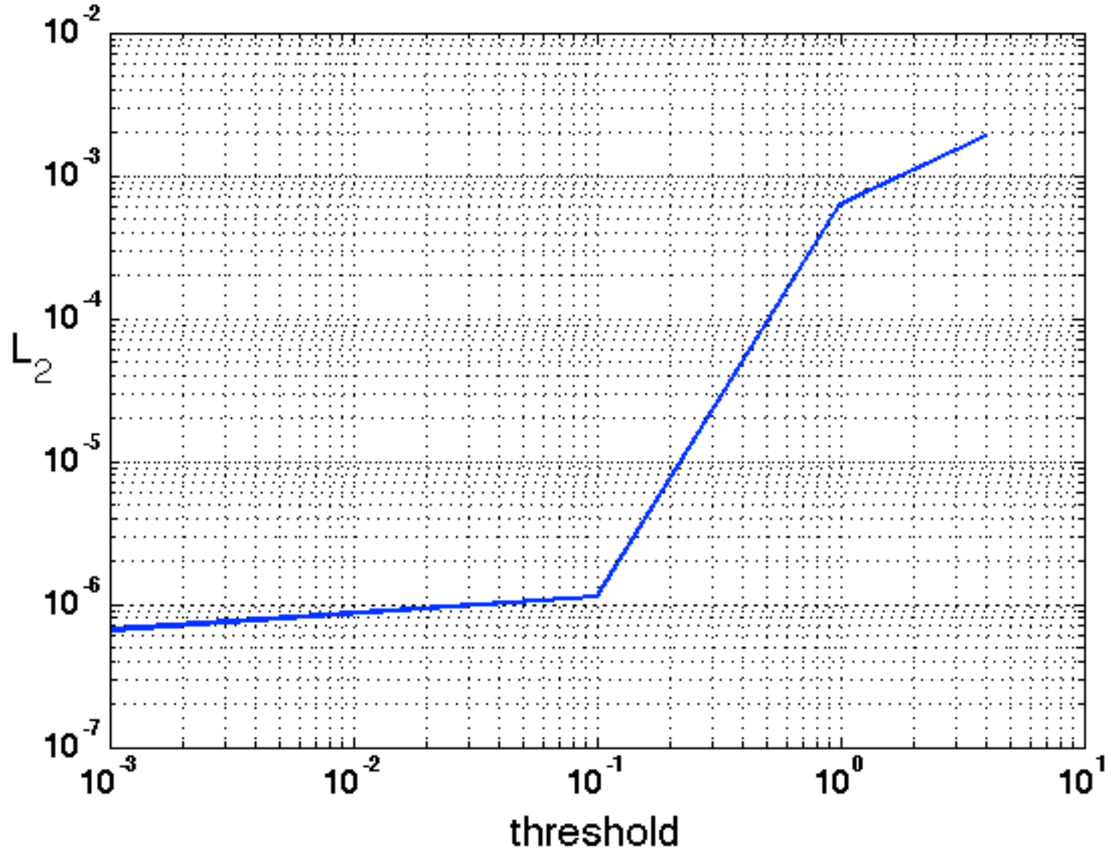


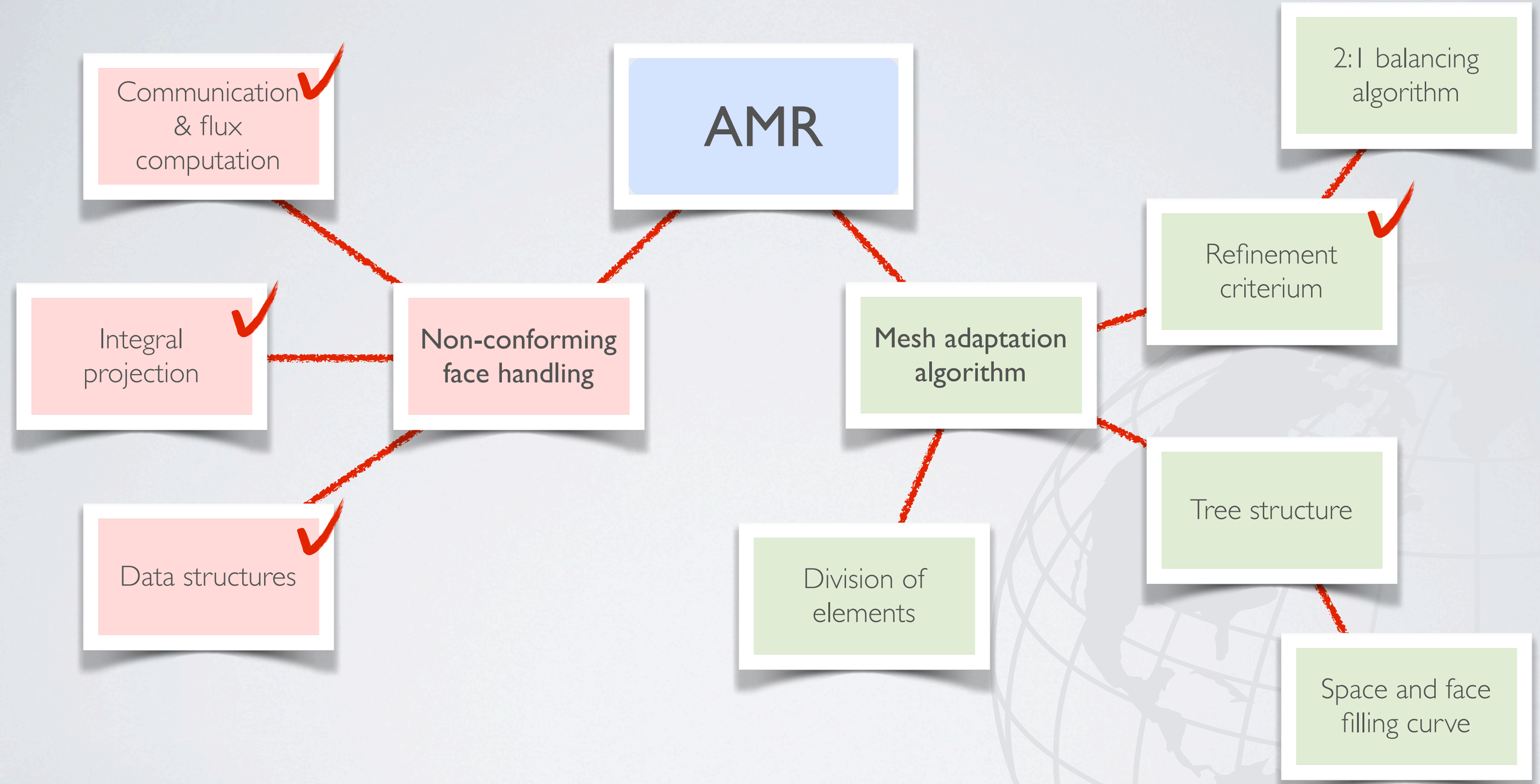


*What are the **benefits and costs**?*

Refinement criterium



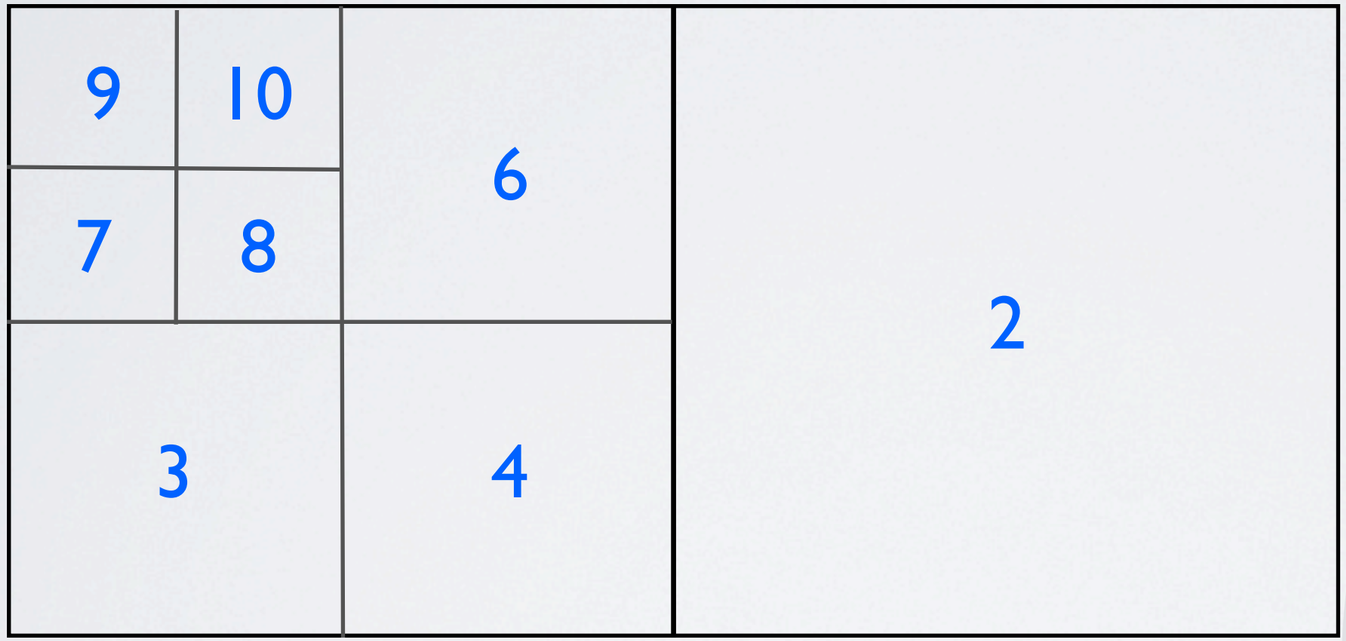



AMR
Communication & flux computation
Integral projection
Data structures
Non-conforming face handling
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

2:1 balancing algorithm

Example 1:

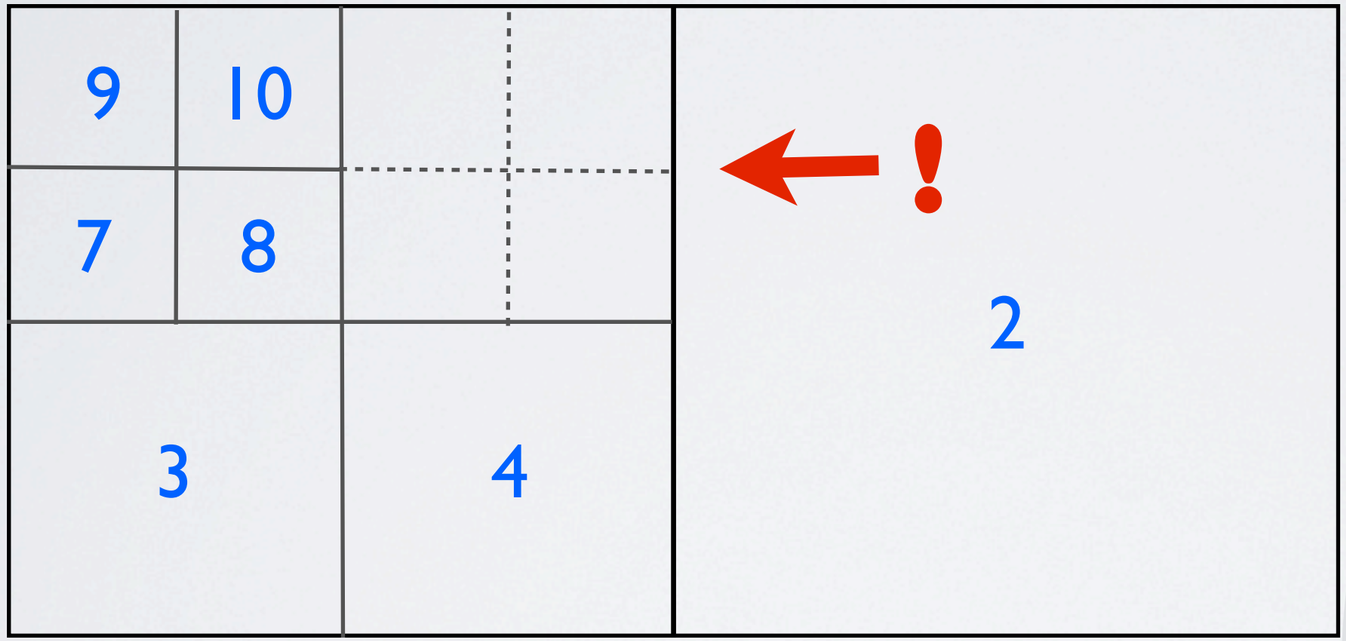
Say we want to refine element 6



ref_list = [6]

2:1 balancing algorithm

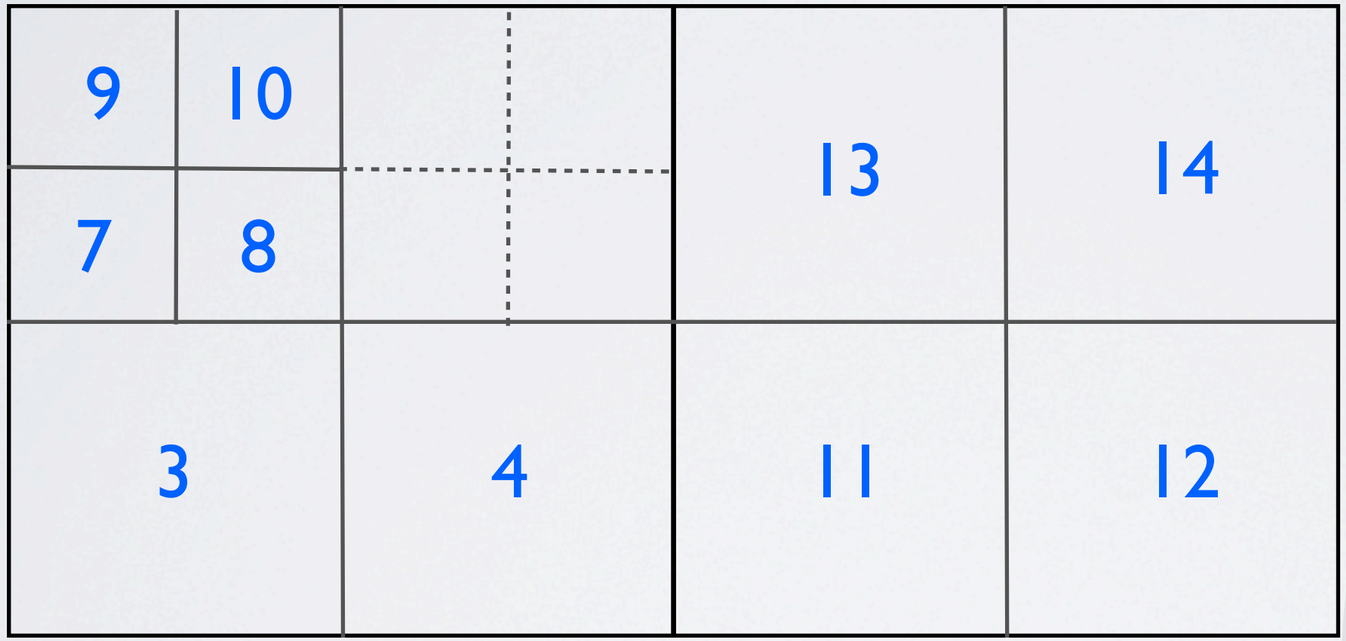
There is a conflict with element 2



ref_list = [6]

2:1 balancing algorithm

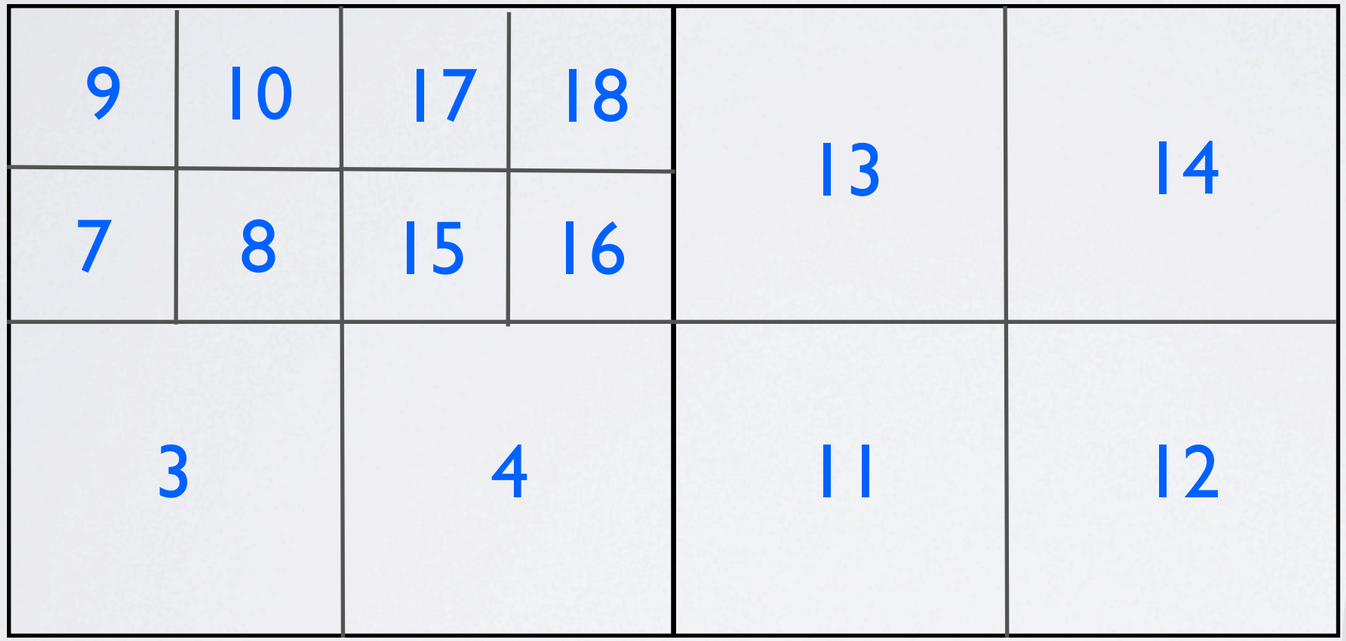
We need to refine 2 first to remove the conflict



ref_list = [2, 6]

2:1 balancing algorithm

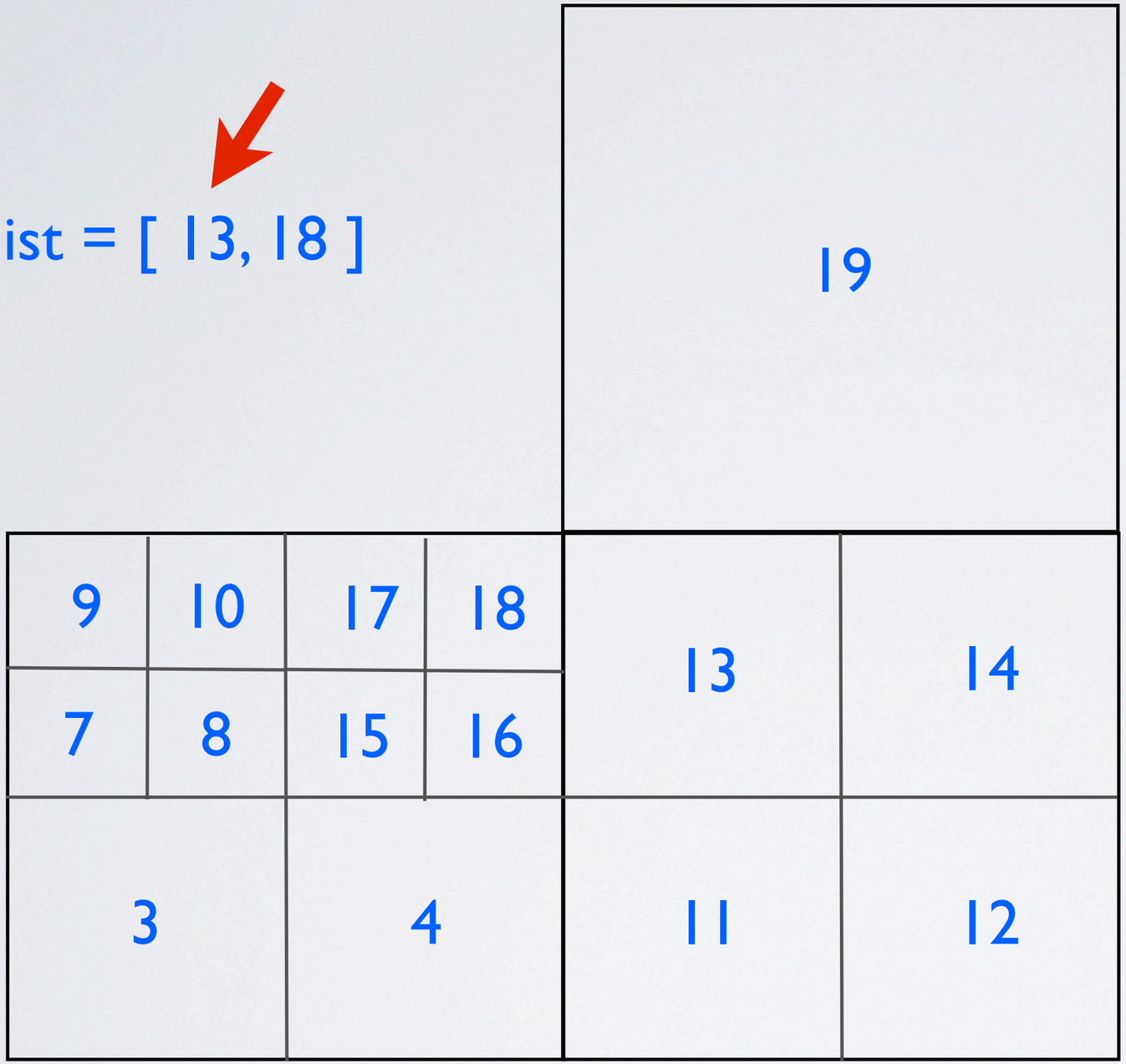
Only then we can proceed with refining 6



ref_list = [2, 6]

2:1 balancing algorithm

ref_list = [13, 18]

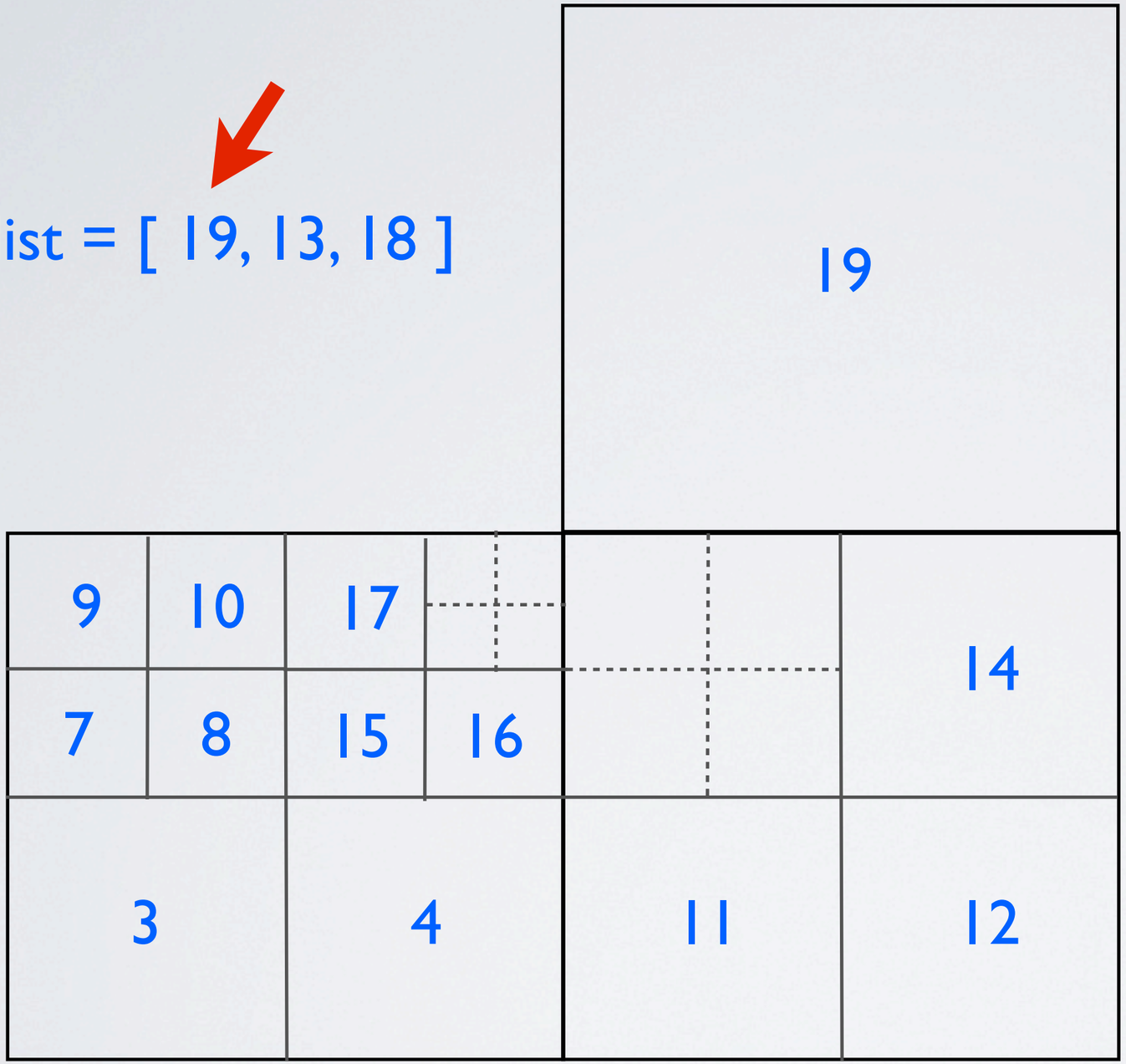


Example 2:

If we would like to refine 18 now, there would be a conflict with element 13.
We need to refine 13 first.

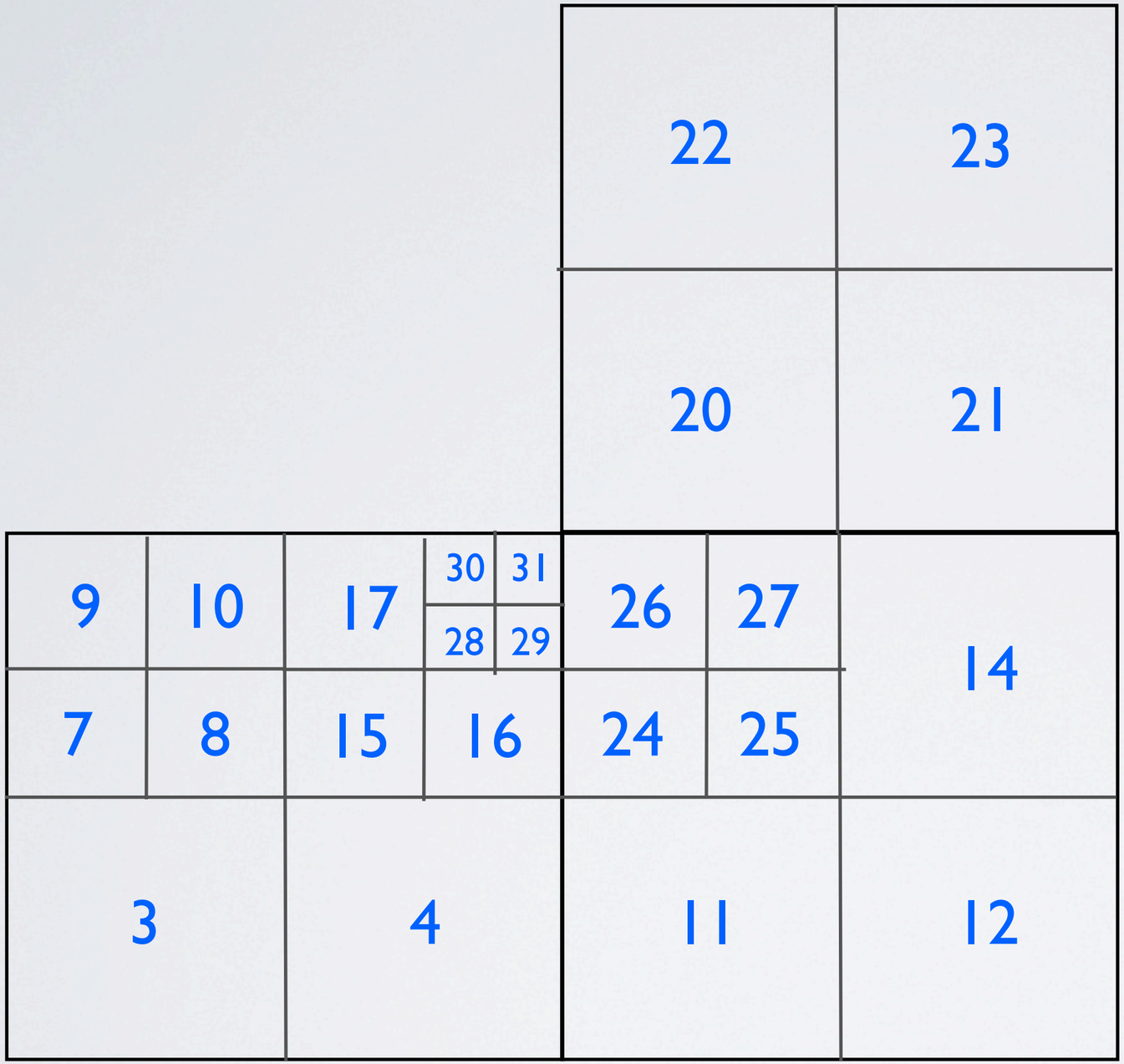
2:1 balancing algorithm

ref_list = [19, 13, 18]



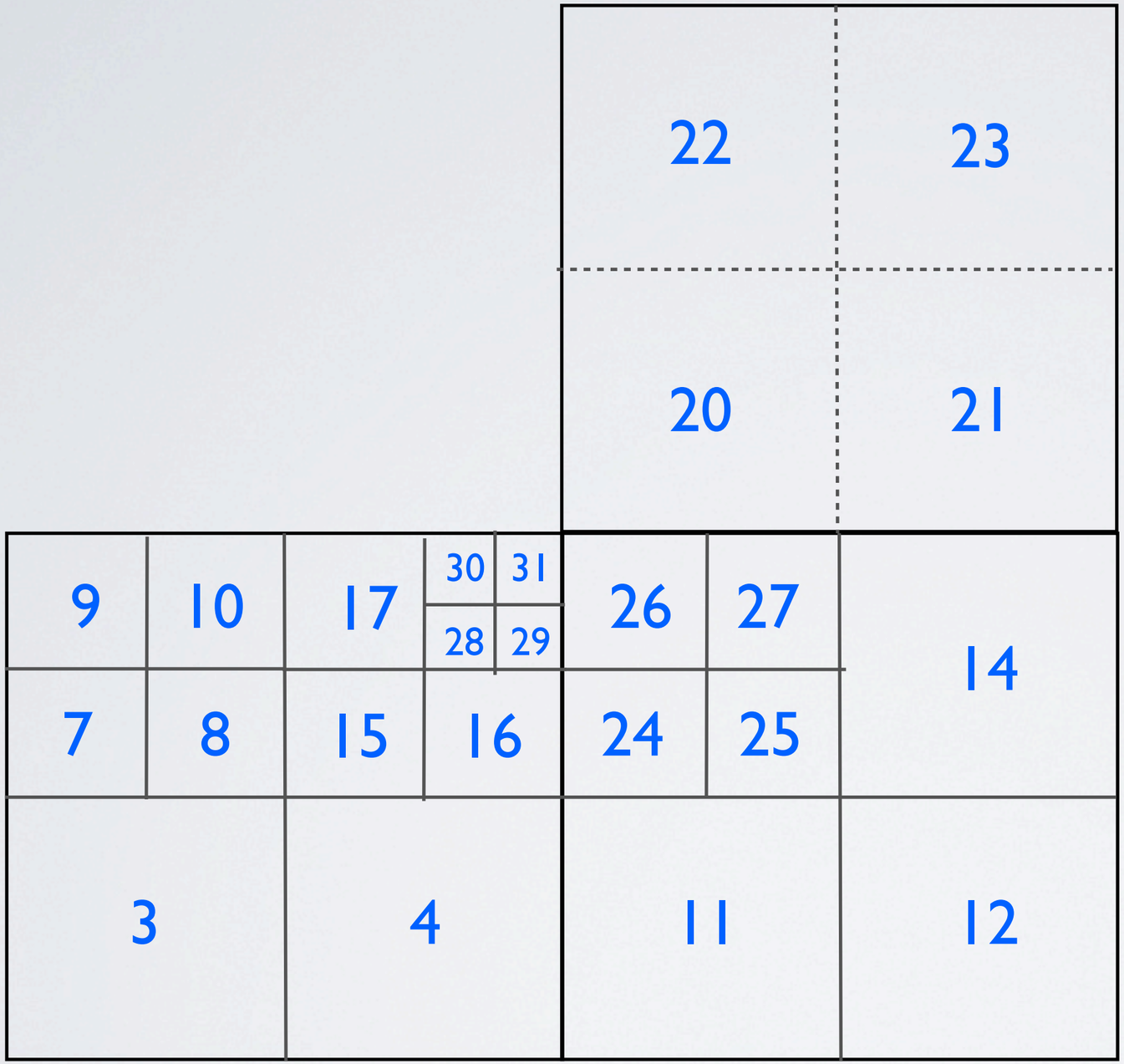
Refining 13, however, creates a conflict with 19.
Before refining 13 we need to refine 19.

2:1 balancing algorithm



After refining 19, 13 and 18 we obtain a 2:1 balanced mesh.

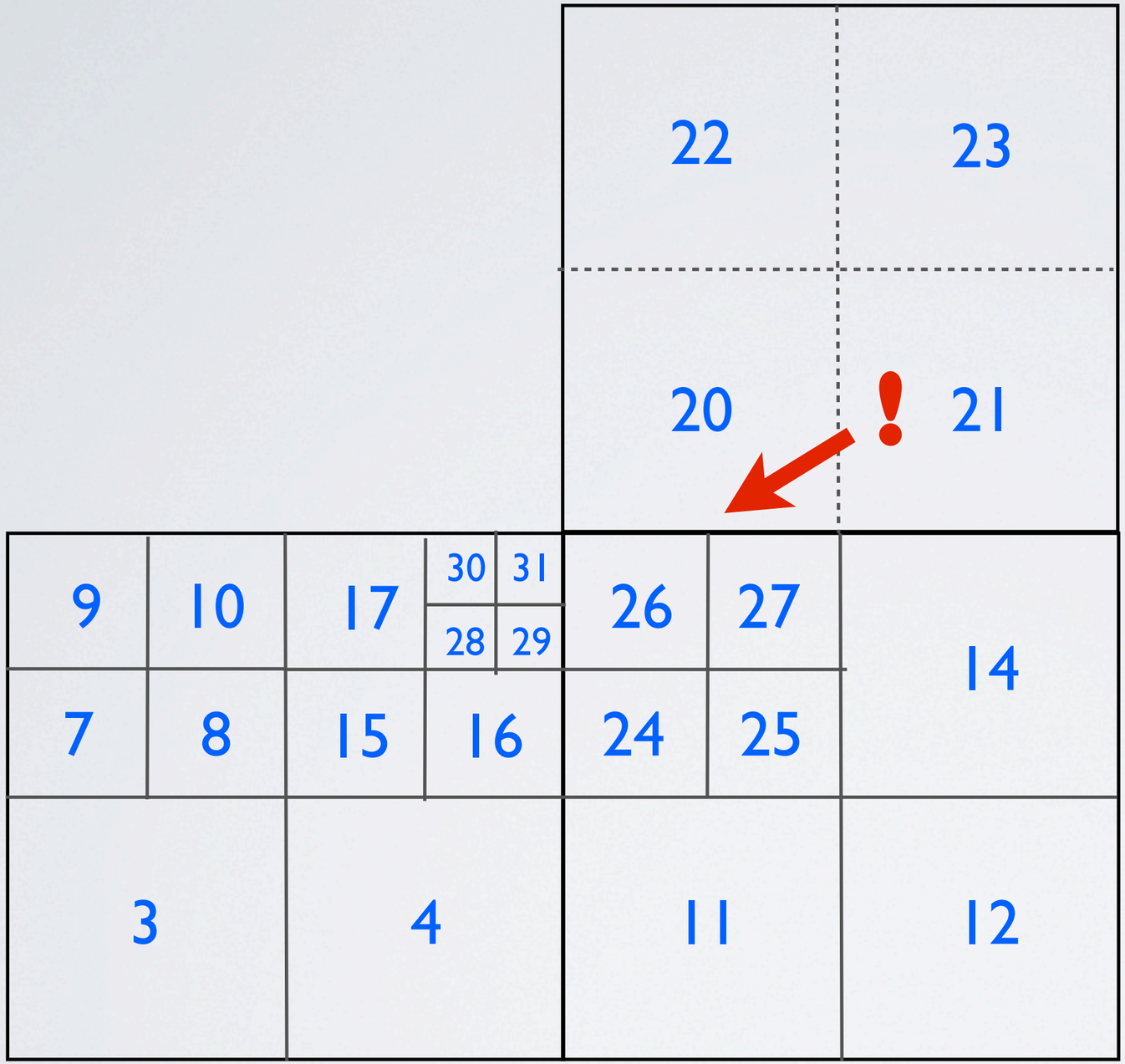
2:1 balancing algorithm



Example 3:

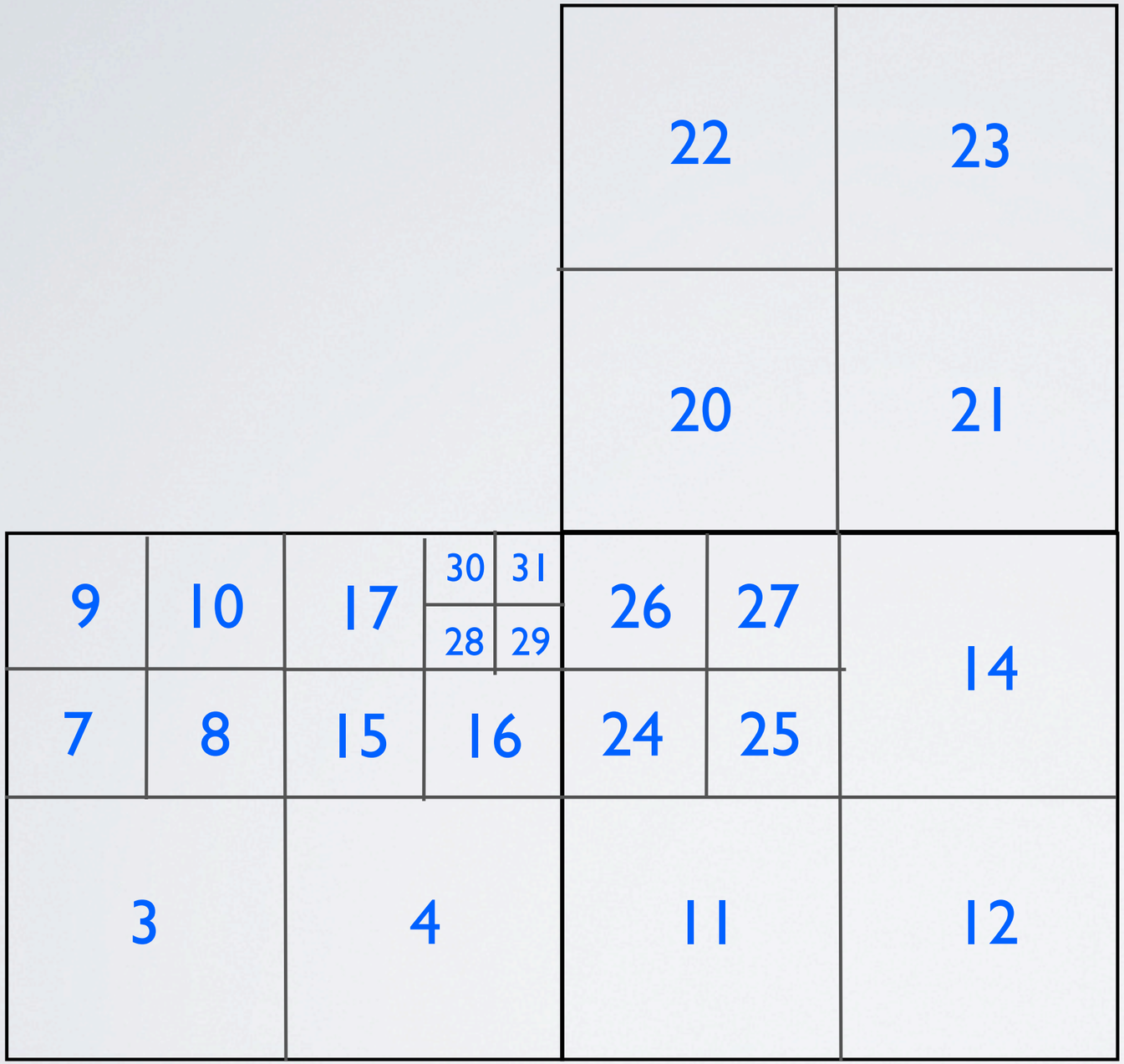
Marking algorithm requests to coarse elements 20 - 23 back into 19.

2:1 balancing algorithm

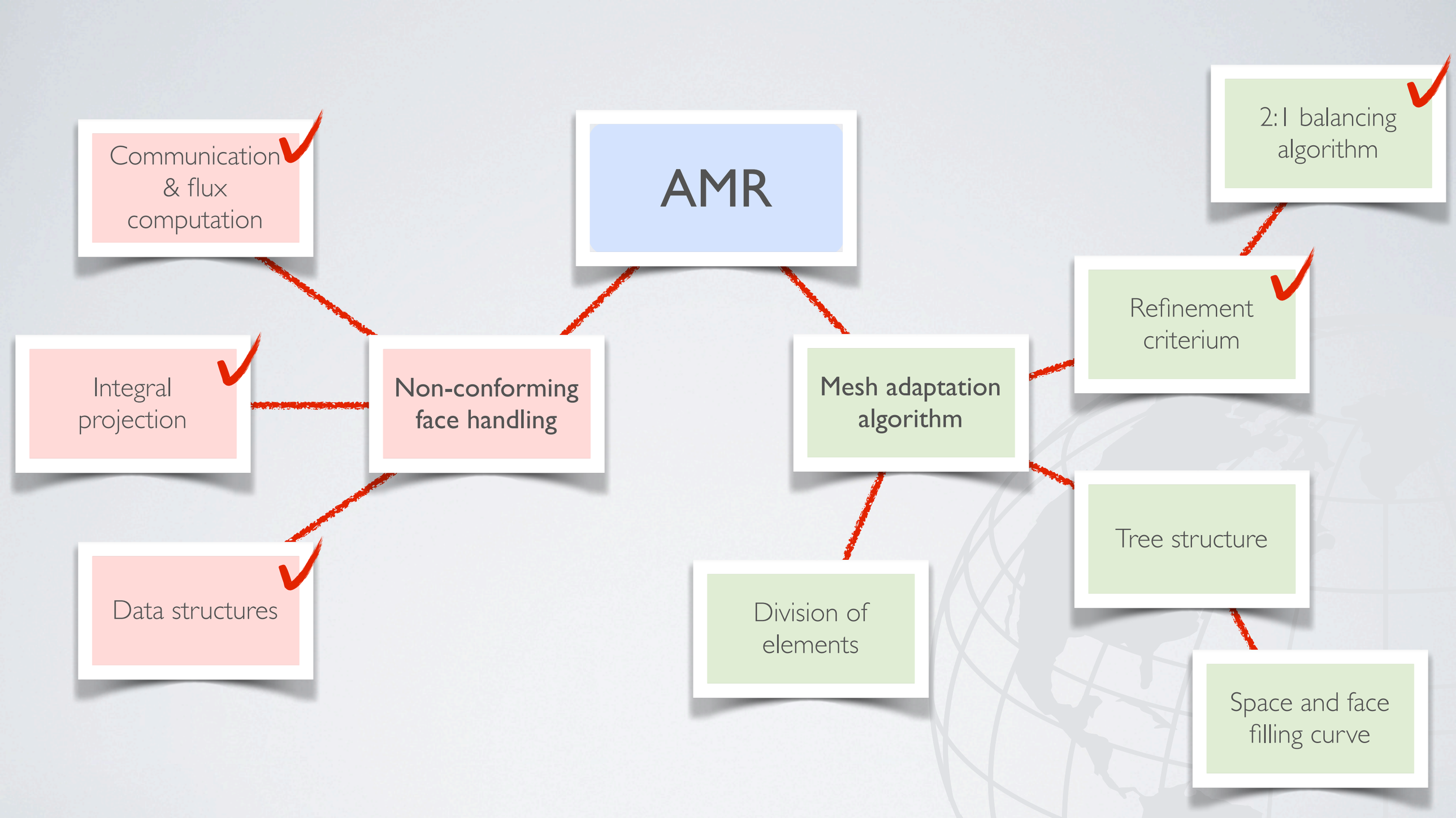


That would cause a conflict with elements 26 and 27.

2:1 balancing algorithm



Elements 20-23 do not coarse to avoid conflict.

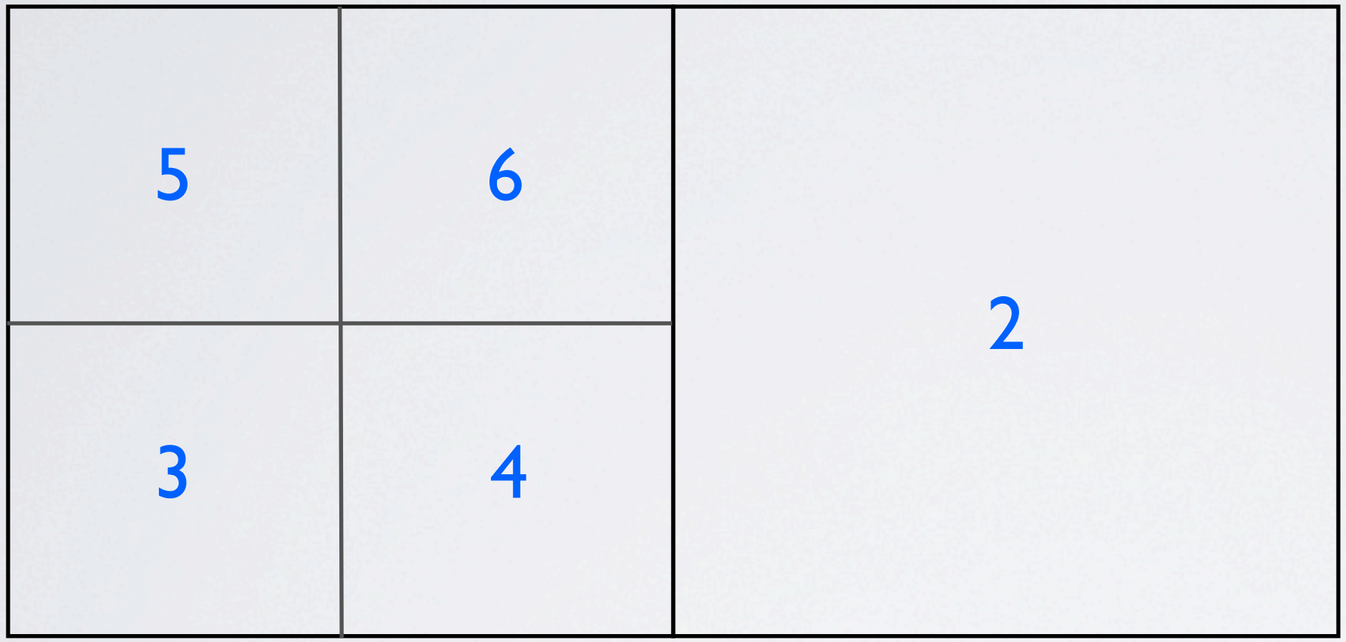

AMR
Communication & flux computation
Integral projection
Data structures
Non-conforming face handling
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

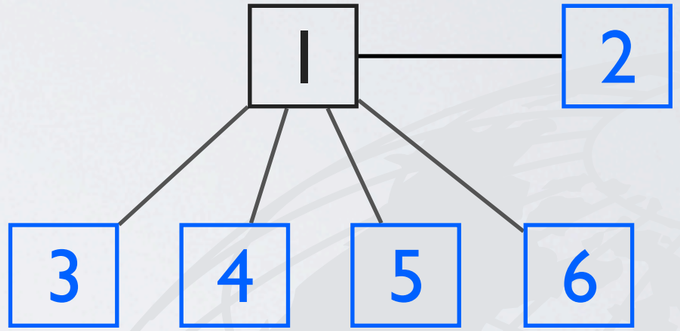
Mesh adaptation algorithm

Level 0 grid

Mesh adaptation algorithm

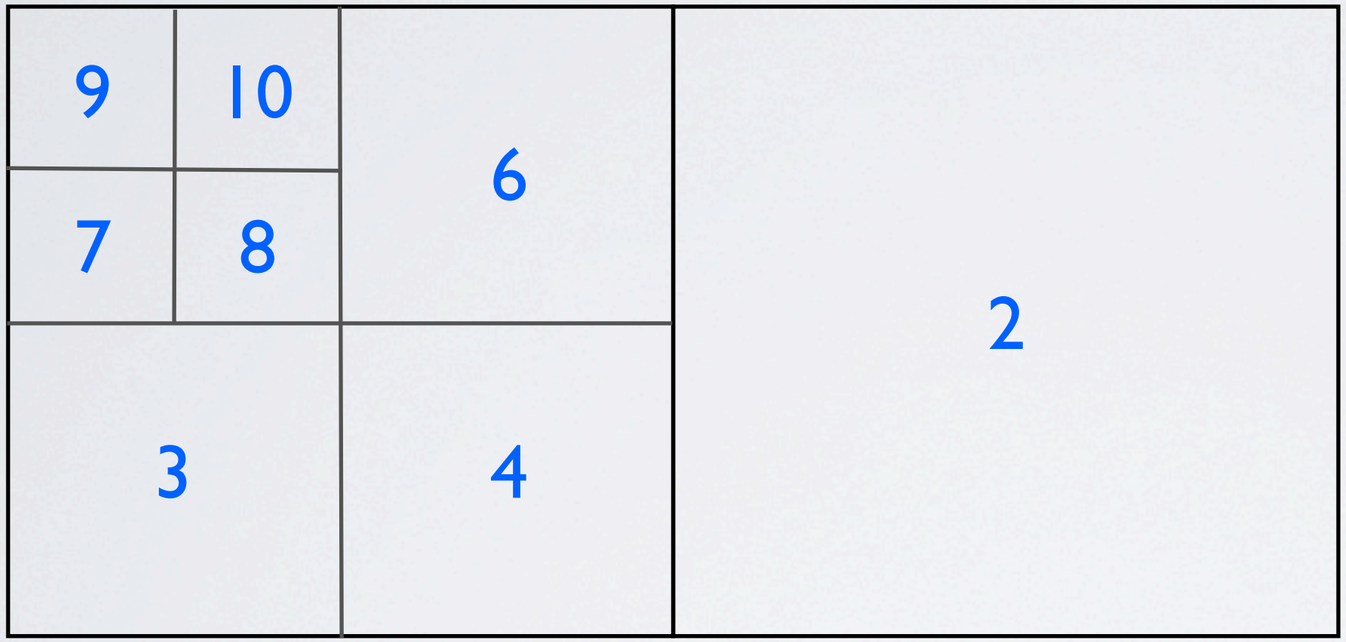
Level 1 grid

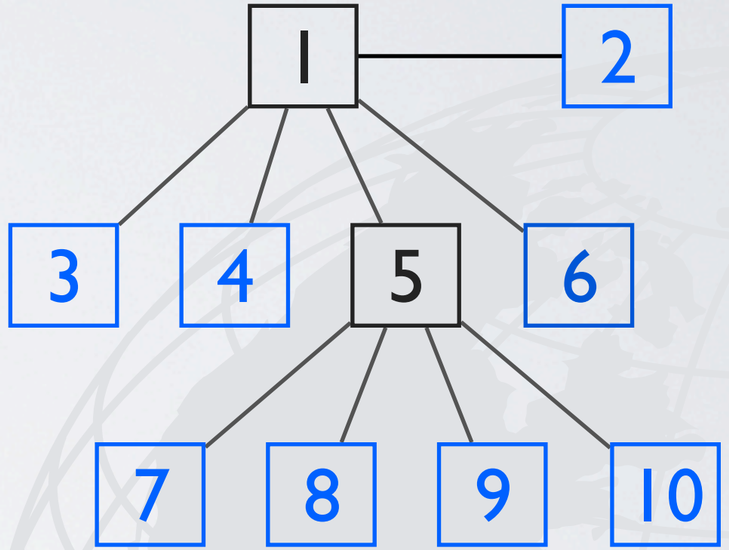




Mesh adaptation algorithm

Level 2 grid

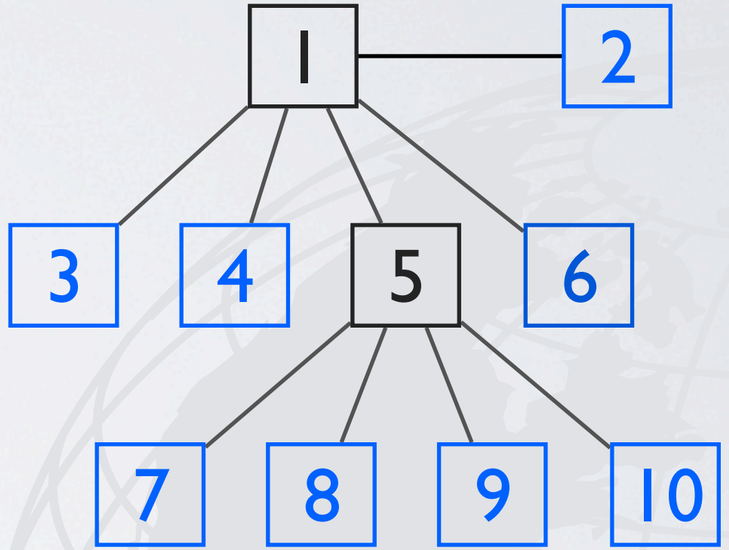




Children array

```
int tc(4,nelemT)
tc(:,1) = [3 4 5 6]
tc(:,2) = [0 0 0 0]
tc(:,3) = [0 0 0 0]
tc(:,4) = [0 0 0 0]
tc(:,5) = [7 8 9 10]
tc(:,6) = [0 0 0 0]
tc(:,7) = [0 0 0 0]
tc(:,8) = [0 0 0 0]
tc(:,9) = [0 0 0 0]
tc(:,10) = [0 0 0 0]
```

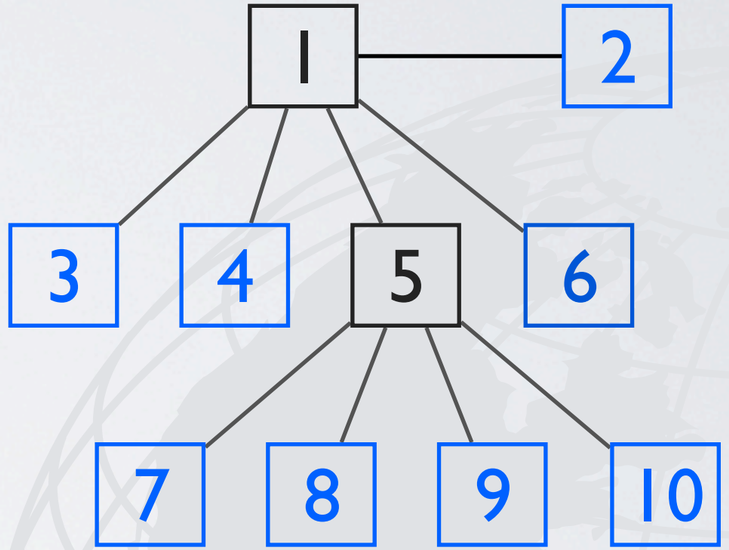
Tree structure



Parent array

```
int tp(nelemT)
tp(1) = 0
tp(2) = 0
tp(3) = 1
tp(4) = 1
tp(5) = 1
tp(6) = 1
tp(7) = 5
tp(8) = 5
tp(9) = 5
tp(10) = 5
```

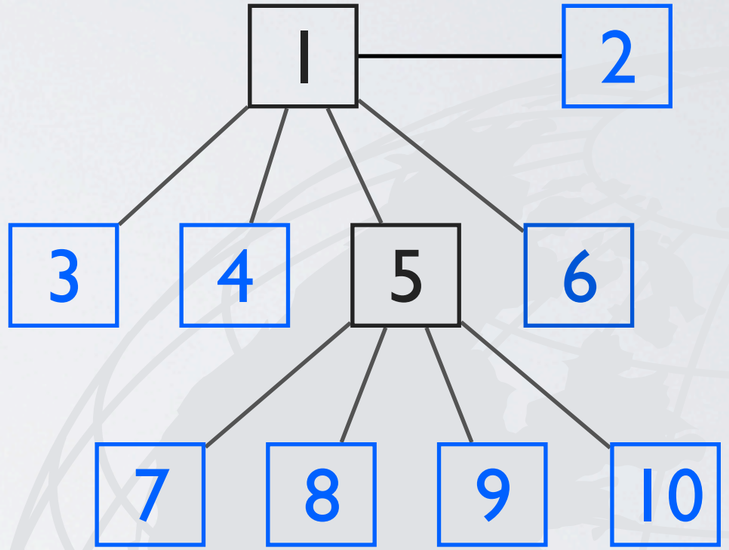
Tree structure



Status array

```
int tm(nelemT)
tm(1) = 0
tm(2) = 1
tm(3) = 1
tm(4) = 1
tm(5) = 0
tm(6) = 1
tm(7) = 1
tm(8) = 1
tm(9) = 1
tm(10) = 1
```

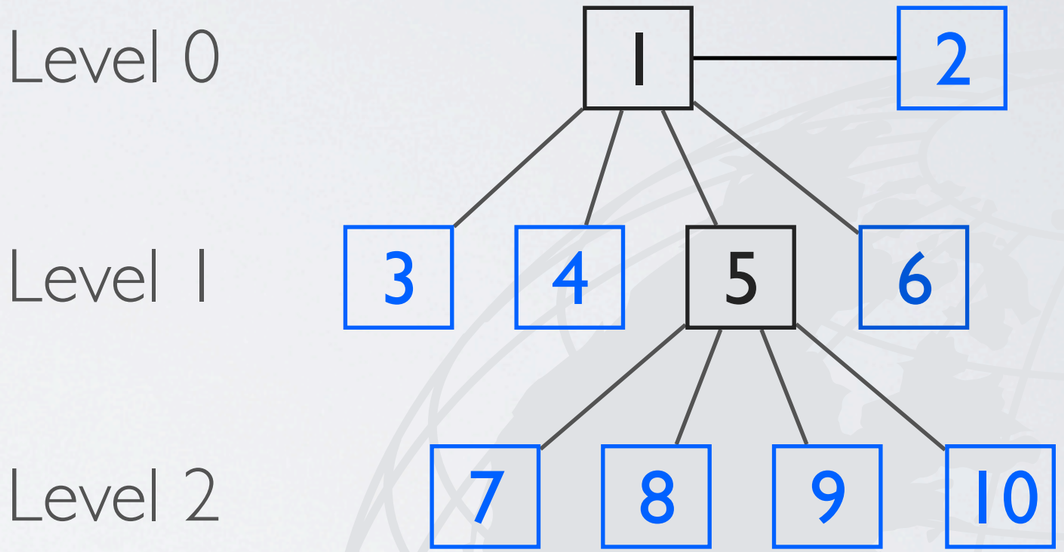
Tree structure



Level array

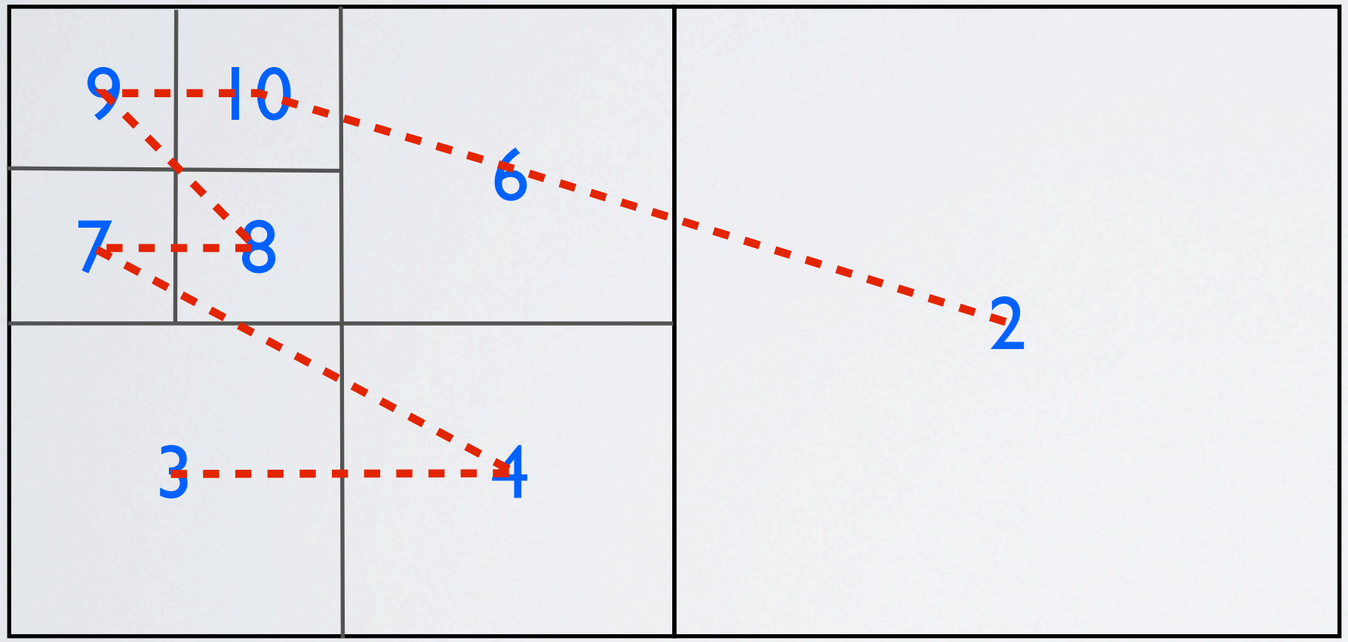
```
int tl(nelemT)
tl(1) = 0
tl(2) = 0
tl(3) = 1
tl(4) = 1
tl(5) = 1
tl(6) = 1
tl(7) = 2
tl(8) = 2
tl(9) = 2
tl(10) = 2
```

Tree structure

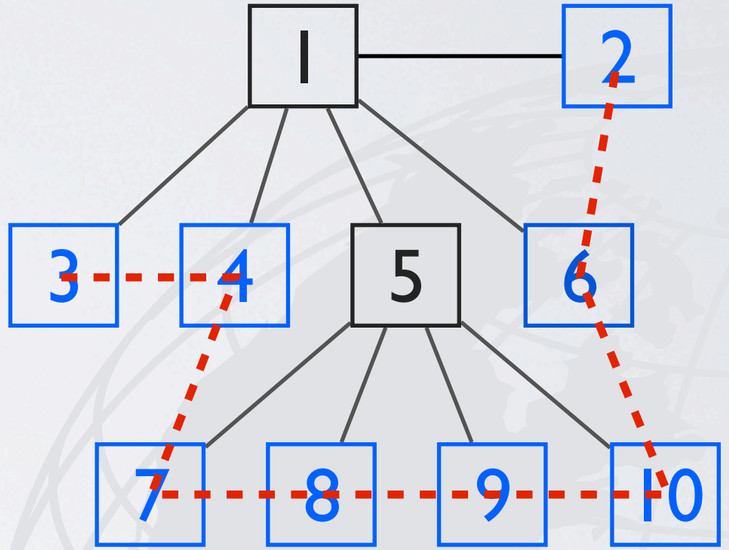


SFC array

```
int nsfc = 8
int sfc(nsfc)
sfc(:) = [3 4 7 8 9 10 6 2]
```



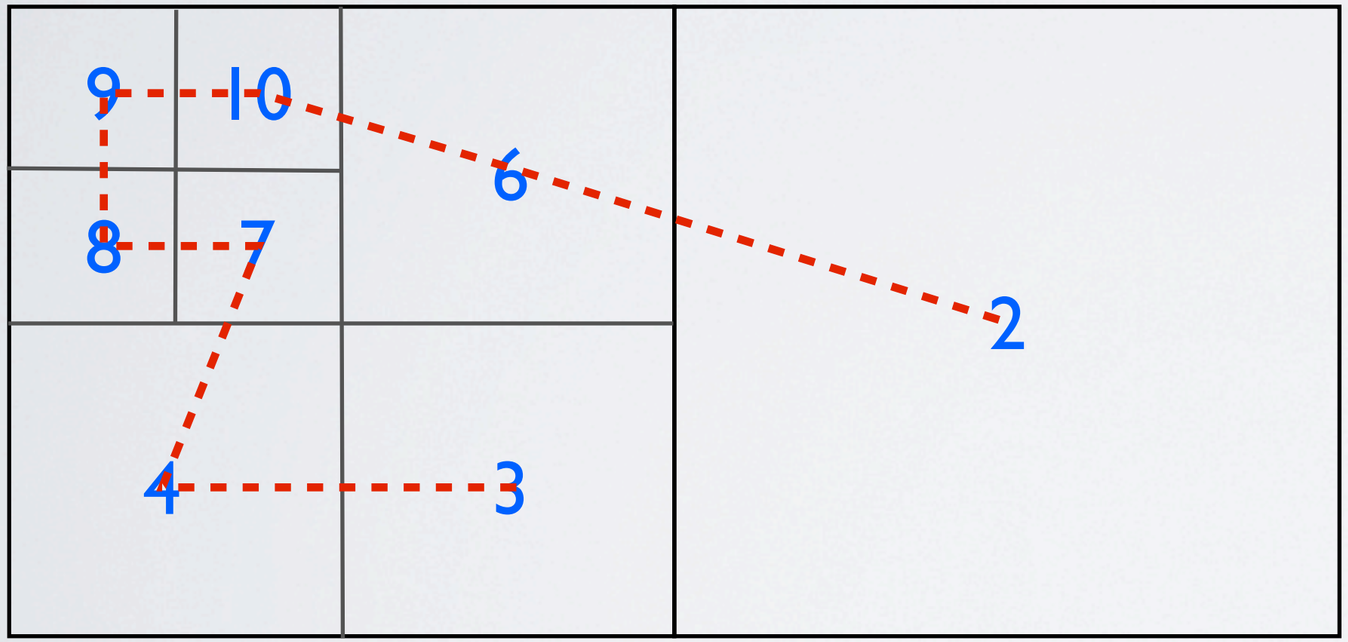
Space filling curve



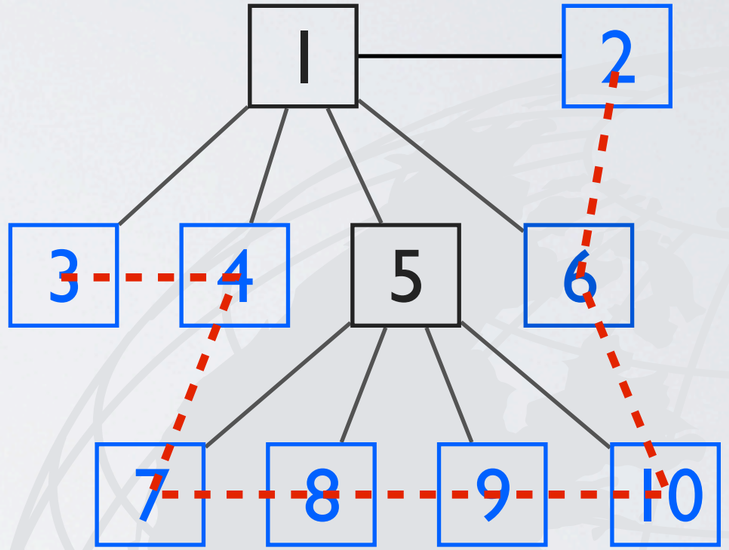
Traverses all active elements

SFC array

```
int nsfc = 8
int sfc(nsfc)
sfc(:) = [3 4 7 8 9 10 6 2]
```



Space filling curve



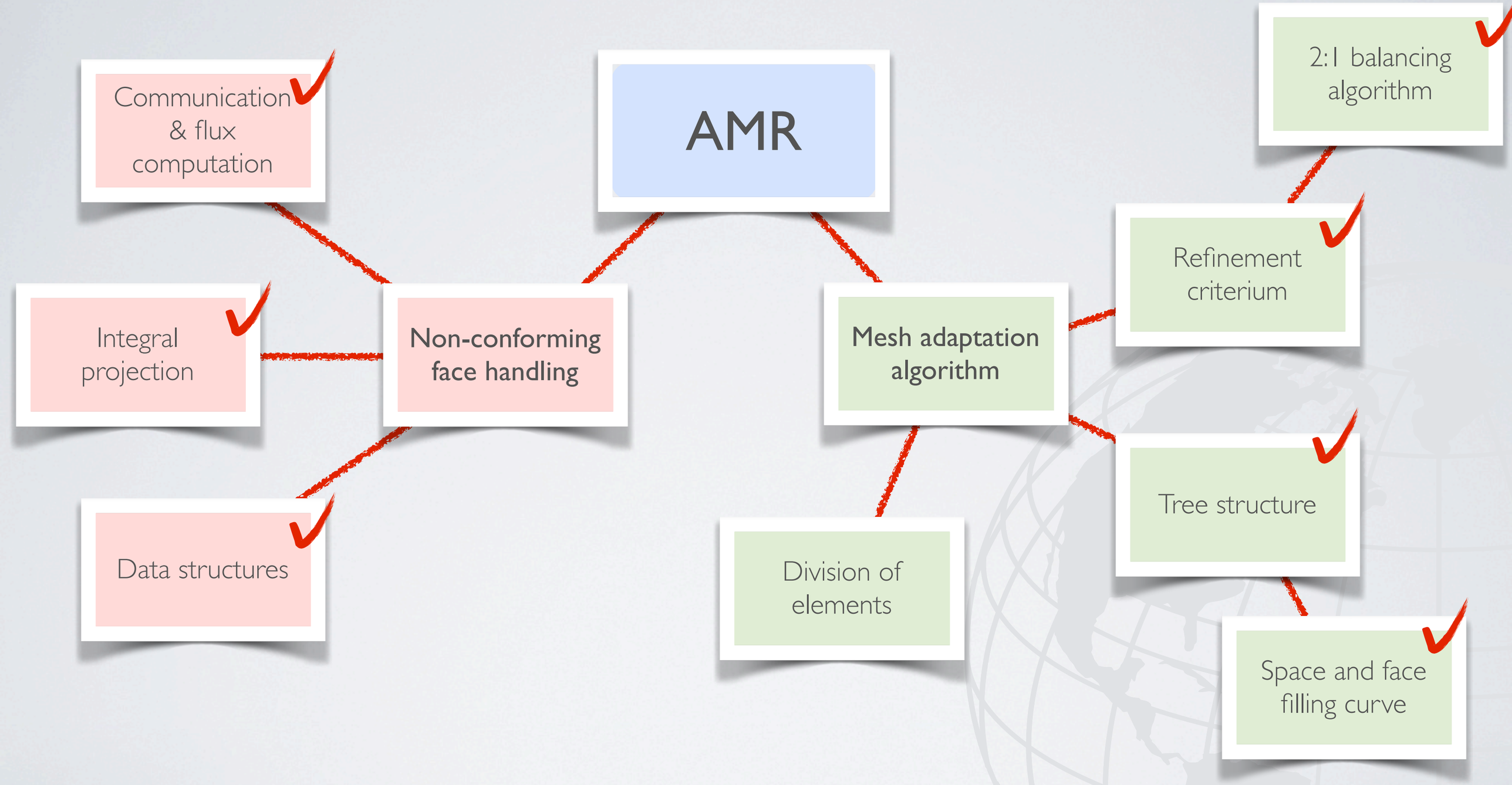
Traverses all active elements

FFC array

Face filling curve

```
int nffc
int ffc(nface)
j = 0
for i=1:nface
    if facepa(i)==1
        j = j+1
        ffc(j) = i
    end
end
```

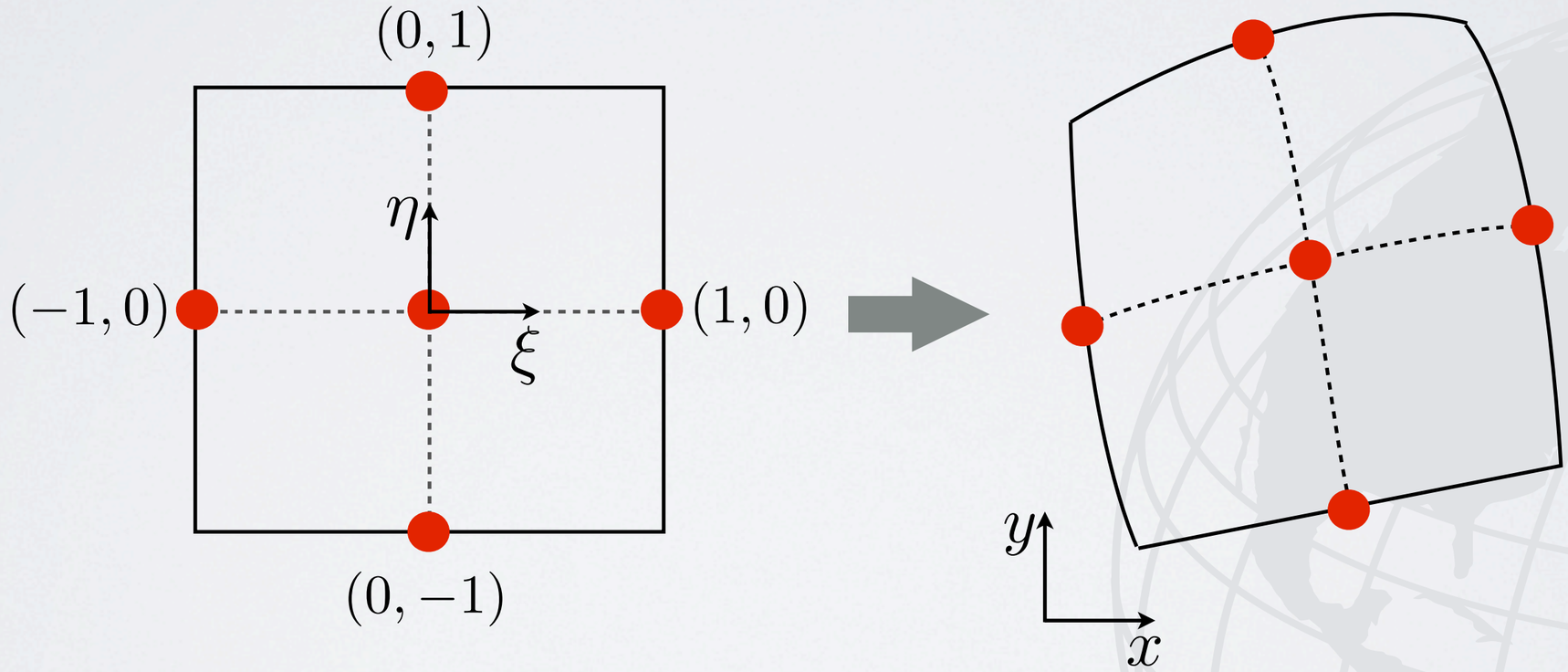
Traverses all active faces


AMR
Communication & flux computation
Integral projection
Non-conforming face handling
Data structures
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve

$$x = \sum_{i=1}^{M_N} x_i L_i(\xi, \eta),$$

$$y = \sum_{i=1}^{M_N} y_i L_i(\xi, \eta).$$

Division of elements



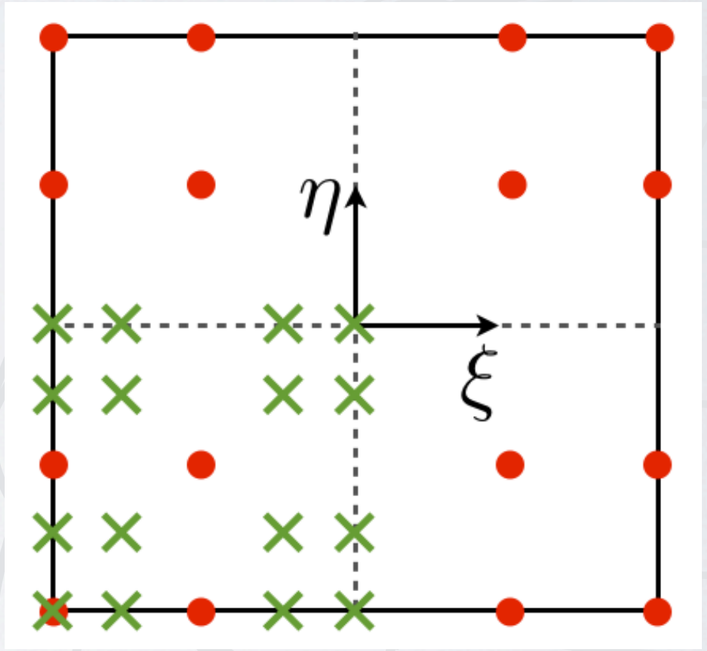
$$x = \sum_{j=1}^{M_N} x_j L_j(\xi, \eta), \quad y = \sum_{j=1}^{M_N} y_j L_j(\xi, \eta).$$

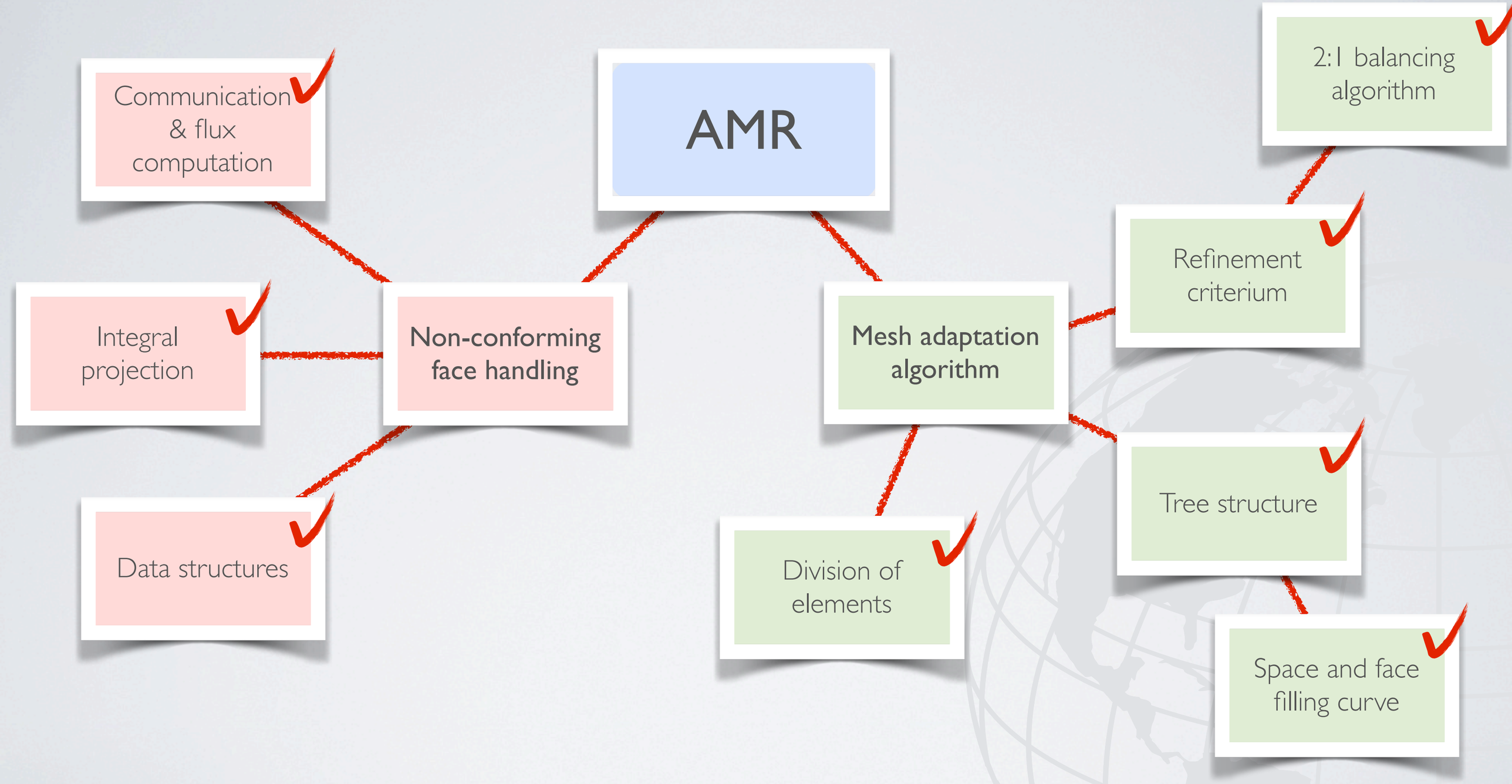
Division of elements

$$\xi_i^{c(k)} = s \cdot \xi_i^p - o_\xi^{(k)},$$
$$\eta_i^{c(k)} = s \cdot \eta_i^p - o_\eta^{(k)},$$

$$x_i^{c(k)} = \sum_{j=1}^{M_N} x_j L_j(\xi_i^{c(k)}, \eta_i^{c(k)}) = \mathbf{L}_{ij}^{c(k)} x_j^p,$$

$$y_i^{c(k)} = \sum_{j=1}^{M_N} y_j L_j(\xi_i^{c(k)}, \eta_i^{c(k)}) = \mathbf{L}_{ij}^{c(k)} y_j^p.$$




AMR
Communication & flux computation
Integral projection
Non-conforming face handling
Data structures
Mesh adaptation algorithm
Division of elements
2:1 balancing algorithm
Refinement criterium
Tree structure
Space and face filling curve