Parallelization of the Mesh Refinement Algorithm of the FDEM Program Package

Torsten Adolph and Willi Schönauer

Forschungszentrum Karlsruhe
Institute for Scientific Computing
Karlsruhe, Germany

torsten.adolph@iwr.fzk.de
willi.schoenauer@iwr.fzk.de

http://www.fzk.de/iwr
http://www.rz.uni-karlsruhe.de/rz/docs/FDEM/Literatur

Motivation

Numerical solution of systems of Partial Differential Equations (PDEs)

- **Finite Difference Method (FDM)** ← **our method**
- **Finite Element Method (FEM)**
- **Finite Volume Method (FVM)**

Objectives

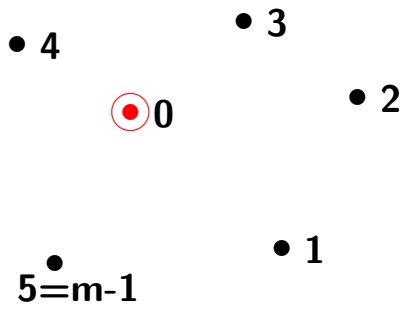
- **Elliptic and parabolic non-linear systems of PDEs**
- **2-D and 3-D with arbitrary geometry**
- **Arbitrary non-linear boundary conditions (BCs)**
- **Subdomains with different PDEs**
- **Robustness**
- **Black-box (PDEs/BCs and domain)**
- **Error estimate**
- **Order control/Mesh refinement** ← **this lecture**

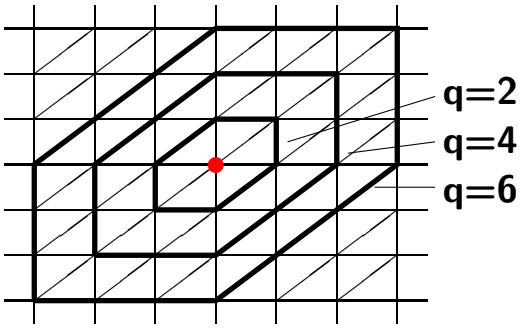
Difference formulas of order q on unstructured grid

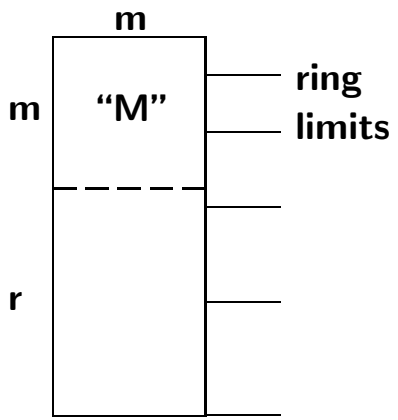
Polynomial approach of order q (m coefficients)

2-D: m = (q+1)·(q+2)/2

3-D: m = (q+1)·(q+2)·(q+3)/6







Influence polynomial $P_{q,i} = \begin{cases} 1, & \text{node i} \\ 0, & \text{other nodes} \end{cases} \longrightarrow u_d,\ u_{x,d},\ u_{y,d},\ u_{xx,d},\ u_{yy,d},\ u_{xy,d}$

Search for nodes in rings (up to order q+Δq) $\longrightarrow$ m+r nodes

Selection of m appropriate nodes

Crossing of ring limit only if $|\text{pivot}| < \varepsilon_{\text{pivot}}$

$\longrightarrow$ Key parameters Δq and $\varepsilon_{\text{pivot}}$

Discretization error estimate

e.g. for u_x: $\quad u_x = u_{x,d,q} + \bar{d}_{x,q} = u_{x,d,q+2} + \bar{d}_{x,q+2}$

$$\rightarrow \quad d_{x,q} = u_{x,d,q+2} - u_{x,d,q} \left\{ + \bar{d}_{x,q+2} \right\}$$

Error equation

$$Pu \equiv P(t, x, y, u, u_t, u_x, u_y, u_{xx}, u_{yy}, u_{xy})$$

Linearization by Newton-Raphson

Discretization with error estimates d_t, d_x, ... and linearization in d_t, d_x, ...

$$\rightarrow \quad \Delta u_d = \Delta u_{Pu} + \Delta u_{D_t} + \Delta u_{D_x} + \Delta u_{D_y} + \Delta u_{D_{xy}} = \quad \text{(level of solution)}$$

$$= Q_d^{-1} \cdot [(Pu)_d + D_t + \{D_x + D_y + D_{xy}\}] \quad \text{(level of equation)}$$

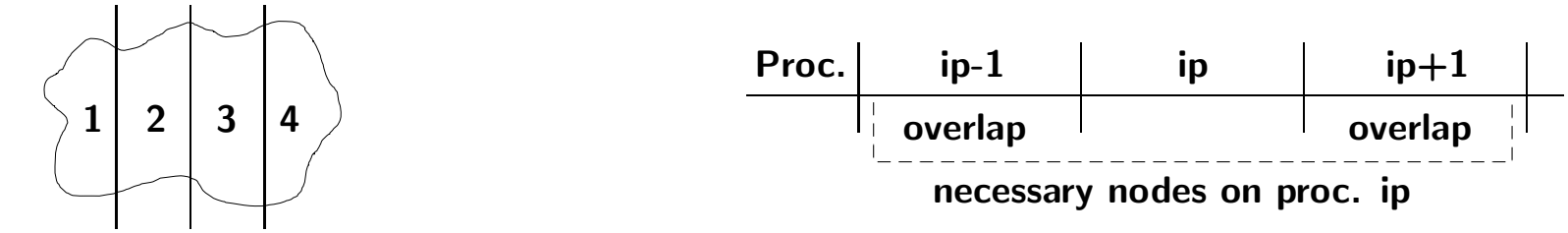
Only apply Newton correction Δu_{Pu}:

$$\rightarrow \quad Q_d \cdot \Delta u_{Pu} = (Pu)_d$$

Parallelization (distributed memory)

Sorting of the nodes by x-coordinate

Distribution of the nodes in np equal parts on np processors (1-D DD)



ip-1, ip+1: overlap processors of proc. ip

Elements are distributed by basket principle

Computation of the right hand side and of the matrix Q_d } **local (without communication)**

Refinement nodes and elements

$$tolg = tol \cdot \|u_d\| \cdot \frac{\|(Pu)_d\|}{\|\Delta u_{Pu}\|}$$

solution level tol $\overset{\text{transf.}}{\rightarrow}$ equation level $tolg$

Node becomes refinement node if

$\|D_x + D_y + D_{xy}\| > s_{grid} \cdot tolg$

holds for 1 solution component at least

Element becomes refinement element if at least one of its nodes is a refinement node

Element number	belongs to Ref. stage	refel			
		Stage 0	Stage 1	Stage 2	Stage 3
1	1	false	true	false	false
2	0	true	false	false	false
3	3	false	false	false	true
4	2	false	false	false	false
⋮	⋮	⋮	⋮	⋮	⋮
ne_l	0	true	false	false	false

refel is key for refinement

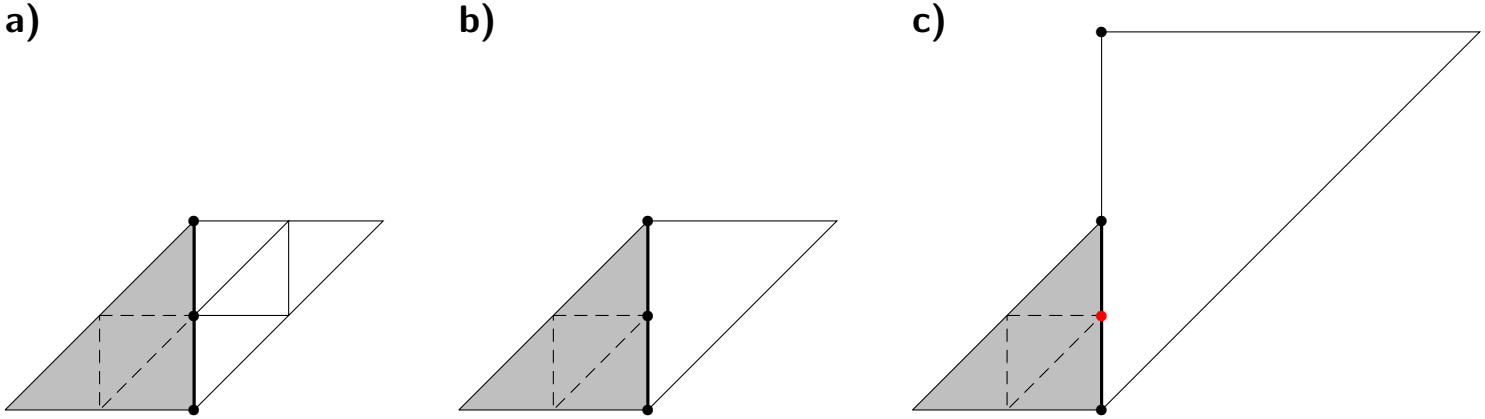
true: to be refined

induces refinement cascade

Refinement cascade I

Investigation of neighbour elements of refinement elements

a) **b)** **c)**

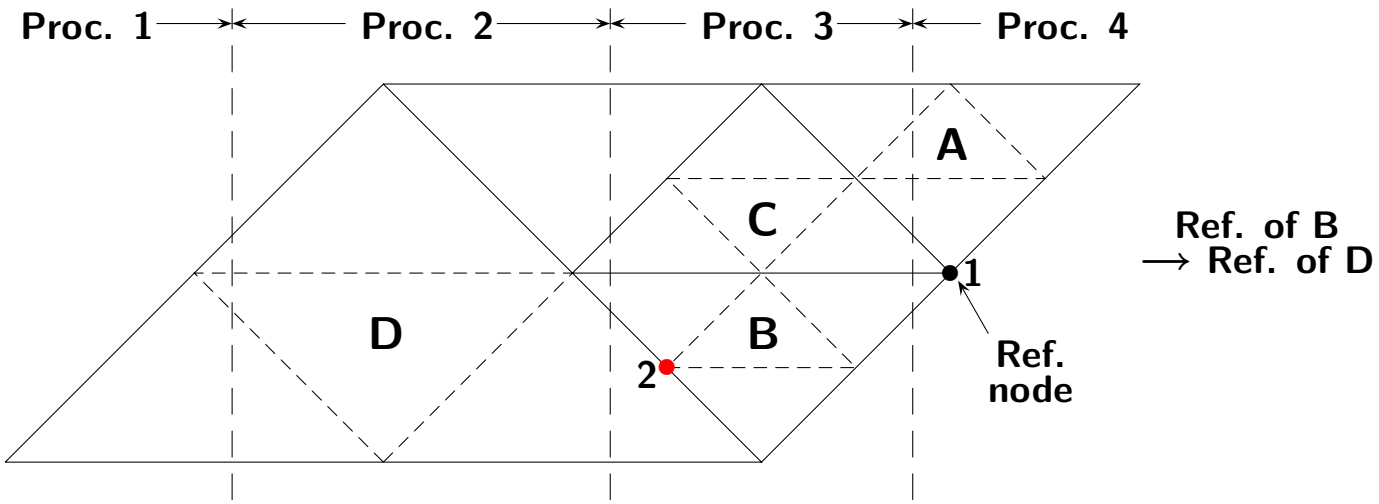


investigated element and current edge

Max. 3 nodes allowed on one edge because of fixed storage scheme

Or: Difference of the refinement stages of two neighboured elements ≤ 1

Refinement cascade II (distributed memory)



Refinement by processor that owns refinement element

Owner is processor that owns the leftmost node of the element

Accumulate all refinement elements because of the cascade

Send them to the corresponding owning processors (⟶ Entry in refel)

Next local refinement cascade

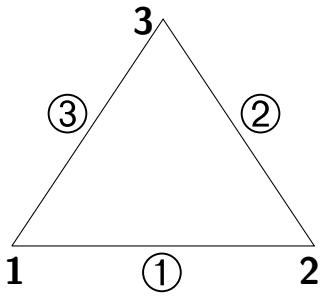
Finish if no new refinement elements occur at processor borders

Mesh refinement I

Largest elements (refinement stage 0) $\longrightarrow$ smallest elements

Array with refinement elements $\longrightarrow$ array with refinement edges

Refinement edge information consists of: **Node numbers of the endpoints (local)**
Element number (local)
Position within element
Number of neighbour elements at the edge
Neighbour element numbers (local)
new! $\longrightarrow$ (Node number of the mid-point)



Store refinement edge information in array (if edge not yet entered in edge list)

Problem: **Edges are halved or new nodes are generated, resp., by that processor that owns the edge**
Edge is owned by that processor that owns the leftmost endpoint of the edge
Unit "edge" does not exist, edge is only identified by endpoints
$\longrightarrow$ no edge numbers

Mesh refinement II

2 arrays for refinement edge information (local + overlap)

one column for each overlap processor

Send number of refinement edges for overlap processors to the right

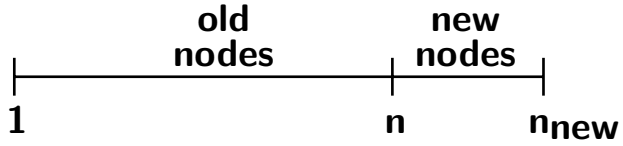
Send array with refinement edge information to the right

Receiving processor: Assign local node numbers 1-n_{cnt}

Send n_{cnt} to the right $\longrightarrow$ Number of nodes generated on the left side is known

Compute new global node numbers (append)

Send node array(s) back to original processor(s)



Mark edges for which the mid-point received a number

Give global node numbers for local edge mid-points

Assign coordinates, consistency order and function values to the nodes

Enter node number into the element list

Check if nodes are boundary nodes

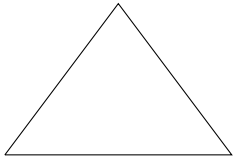
After last refinement stage: Redistribution of nodes and elements onto the processors with 1-D DD

Mesh refinement III

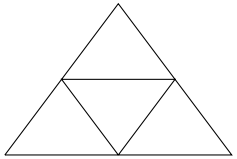
Separation of the elements (new element may "move" to neighbour processor!)

2-D: 4 triangles

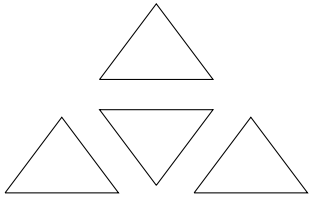
a)



b)

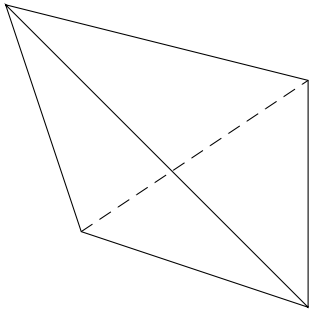


c)

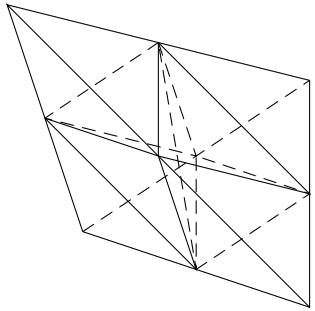


3-D: 8 tetrahedrons

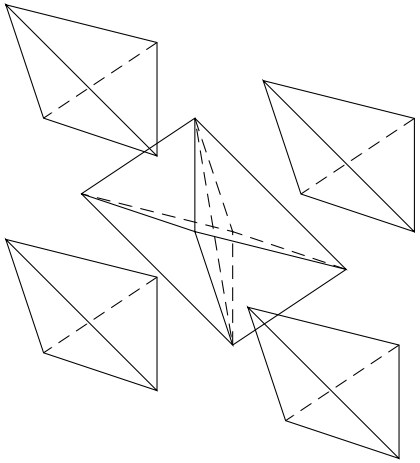
a)



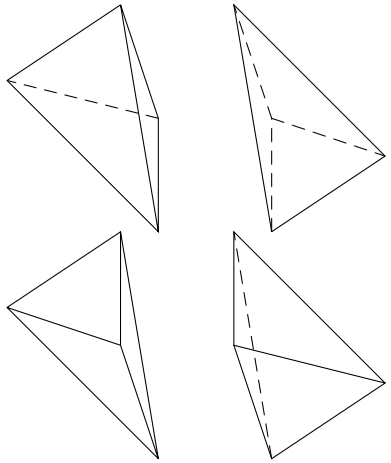
b)



c)

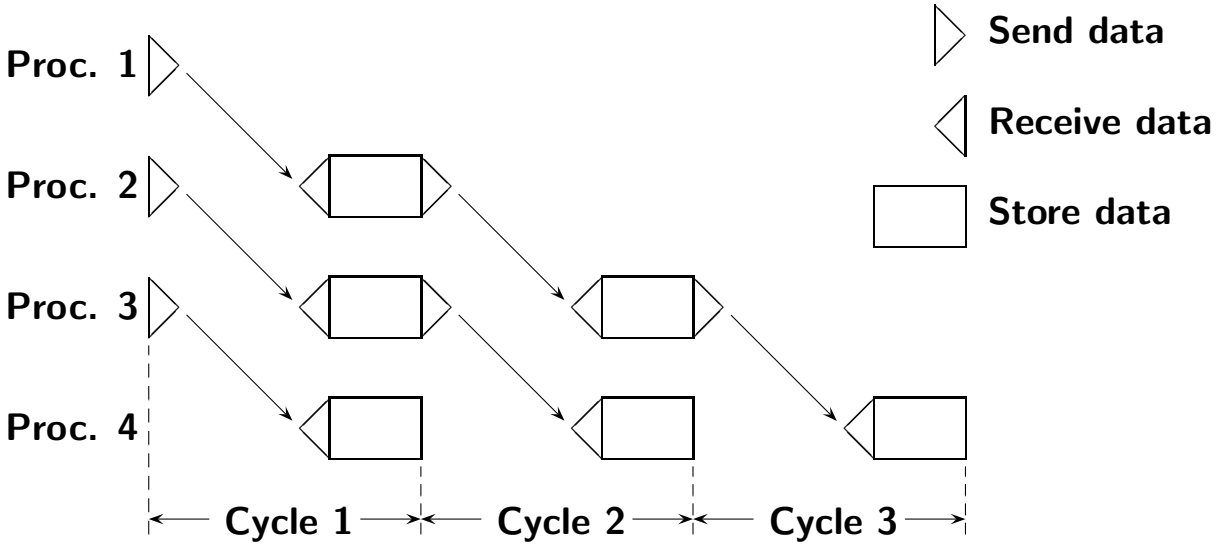


d)



Specific characteristic

Sending arrays to overlap processors: always first send number of units to send to the right/left (np-1 cycles)!



Processors are able to set "receives" correctly:

a) no hard buffer lengths necessary ⟶ saving of storage and time

b) no unnecessary communication ⟶ saving of time

Example

Scaling of the mesh refinement on a 4×1-domain (complete refinement)

Grid nr.	Dimension	Original grid Number of nodes	Original grid Number of elements	Refined grid Number of nodes	Refined grid Number of elements	No. of proc.	CPU time for ref.
1	512 × 128	65,536	129,794	260,865	519,176	1	1.95
2	1,024 × 256	262,144	521,730	1,046,017	2,086,920	4	1.93
3	2,048 × 512	1,048,576	2,092,034	4,189,185	8,368,136	16	2.04
4	4,096 × 1,024	4,194,304	8,378,370	16,766,977	33,513,480	64	2.13
5	8,192 × 2,048	16,777,216	33,533,954	67,088,385	134,135,816	256	2.20
6	16,384 × 4,096	67,108,864	134,176,770	268,394,497	536,707,080	1024	2.28

IBM Blue Gene/L, Jülich, Germany

700 MHz PowerPC 440 core, 2800 MFLOPS peak