Some Recent Developments in Grid and Object Generation

Rainald Löhner
CFD Center, College of Sciences
George Mason University, Fairfax, VA, USA

www.cfd.gmu.edu/~rlohner

Outline

- Introductory Remarks
- Advancing Front / Delaunay
- Speed: Scalar, SMP, DMP
- Regular Point Distributions
- Separated Arbitrary Objects
- Conclusions and Outlook

Acknowledgements

- Jean Cabello
- Juan Cebral
- Fernando Camelli
- Ralf Tilch
- Romain Aubry
- Adrien Loiselle

Introductory Remarks

Worries About LES…

- Aerodynamics / Aeroacoustics: RANS → LES → DNS
  - Increase in Spatial/Temporal Order [4^{th}-6^{th} Order]
  - Increase in Nr. Of Gridpoints/DOF: $O(10^3\text{-}10^6)$ [MPI OK]
  - Increase in Nr. Of Timesteps: $O(10^3\text{-}10^5)$ [MPI Limiting]

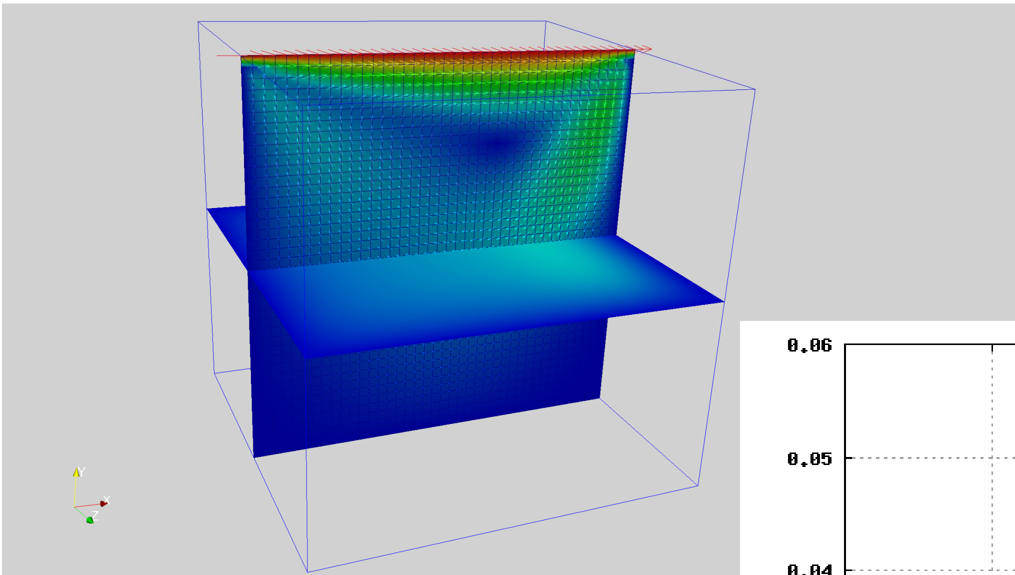
- Exascale Machines
  - Massively Parallel
  - Energy Consumption ~ Access To Memory
  - Motto: `Flops Are Free, Memory is Expensive'

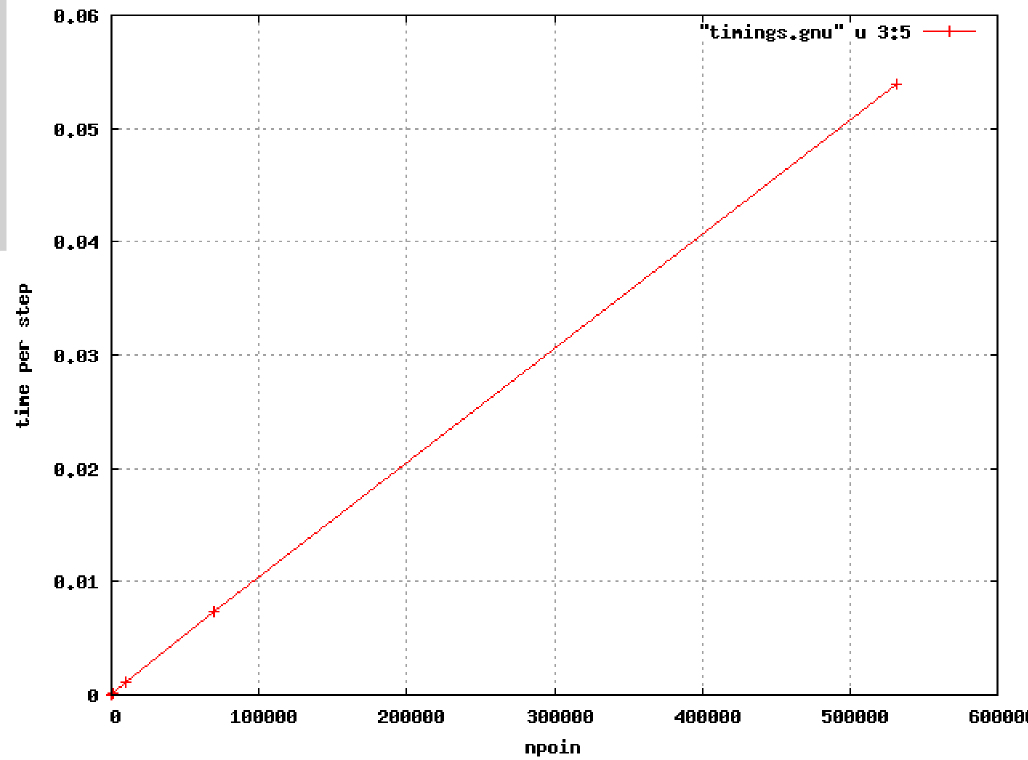
- ➔ Need **High Order Schemes With:**
  - **Minimal Memory Access**
  - **Minimal Transfer Between Domains/MPI Nodes**

Worries About Hardware…

- Advances in Hardware Uneven
- CPUs > RAM > Network
- ➔ Red Shift ➔ Hardware Crisis
- CPUs/GPUs: Transfer Rate to Memory Limit Performance
  - Can Predict Speed By Just Counting Memory Access
  - `FLOPS Are Free'
- Even Worse for MPI/Network Transfer

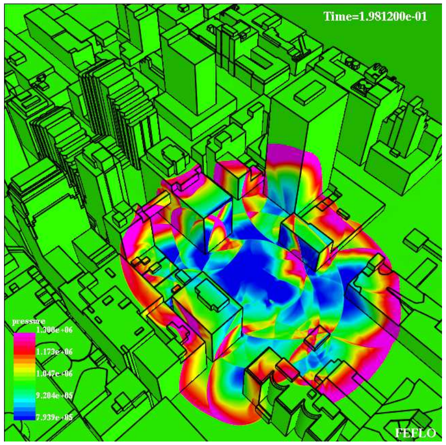
Lid-Driven Cavity, FDFLO Timings





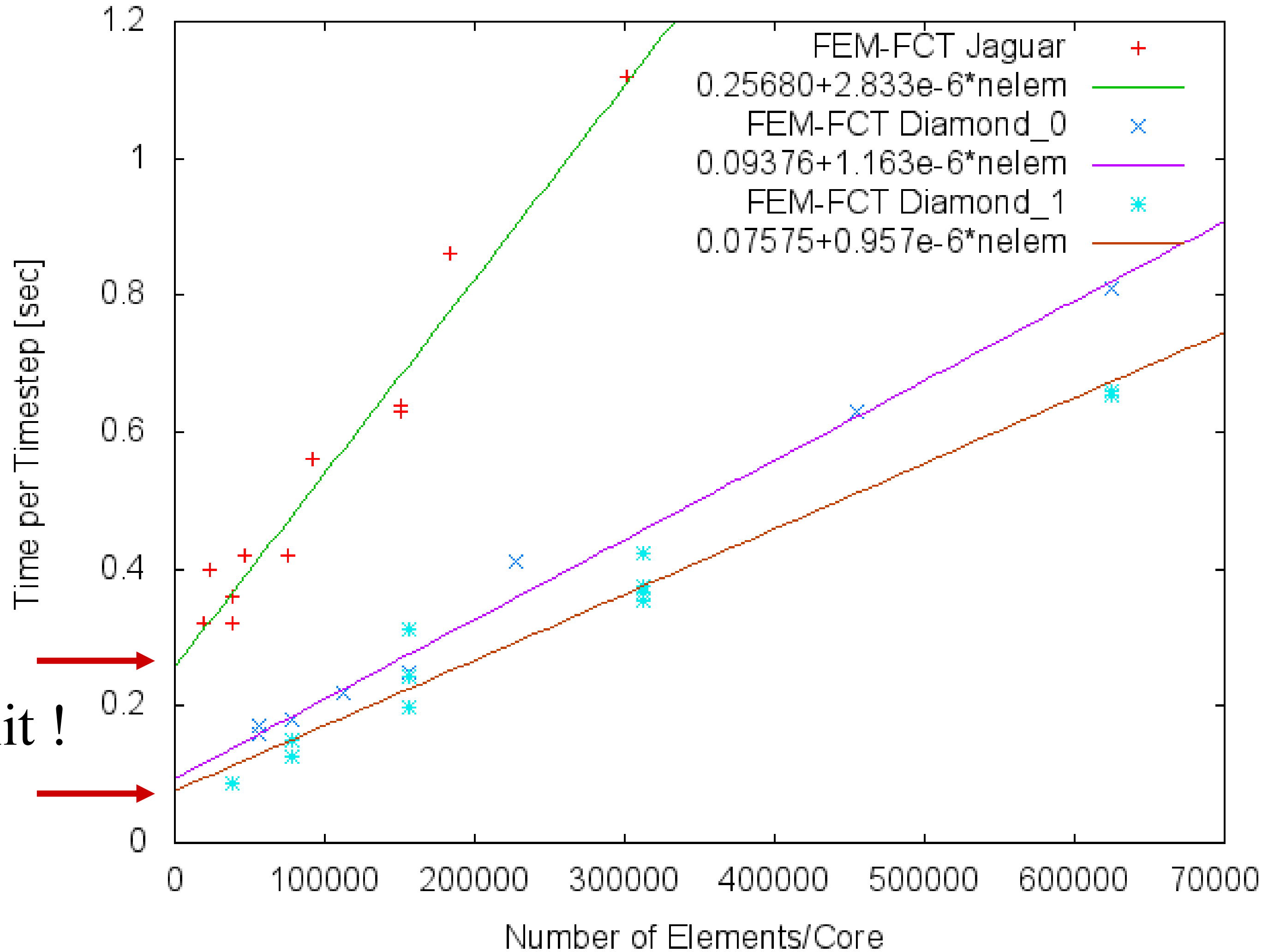
Laptop, 1 Core Running

FEFLO: FEM-FCT Timings



Nelem>1Bels
Nproc=O(50K)

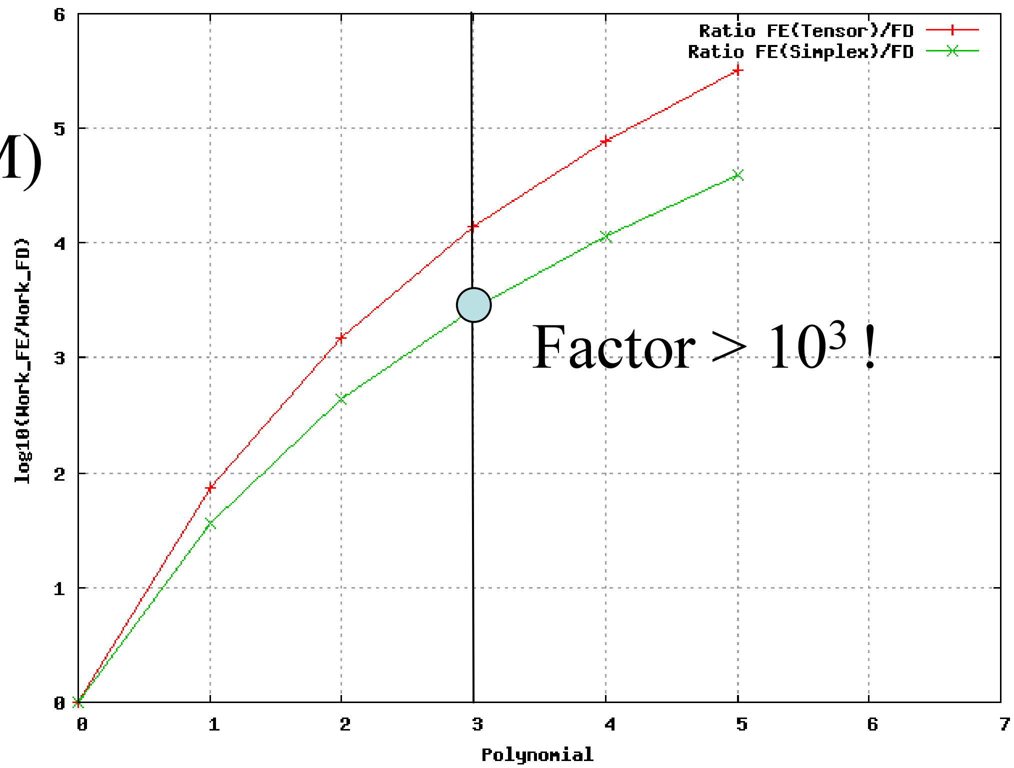
MPI Limit !



Worries About Numerics…

- Higher Order Schemes: Finite Difference Methods (FDMs) Orders of Magnitude Faster Than FEM/DGMs
  - ➔ Revisit FDMs

log(CPU_FEM/CPU_FDM)



Worries About Mesh Adaptation

- LES: Most Elements in LES Region(s)
- Need Isotropic Elements
- Lengthscale(s) Known

- ➔ Gains From Stretched Elements Limited

- (Perhaps…)

CFD Crisis

- If The Time To Do 1 Timestep Is Limited (MPI)
- If We Need > 10^{12} Elements
- If We Need > 10^{6} Timesteps
- ➔ 3-4 Weeks on 10^{6} Cores

- ➔ Windtunnel Serious Contender Again
  - 3-D Printing
  - Fast Machining/Instrumentation
  - Fast Evaluation/Sweep

Current CFD Leading Edge

- Only Partially in Aerospace
- Wind Engineering / Light Structures
  - Always LES + Complex FSI ➔ Long Duration
  - Large Domains (Upstream Influence, Wind)
- Automotive
  - Always LES, Fluid + Noise
- `Complex Physics'
  - Chemistry, Particles, FSI, …

Meshing

Overarching Motto:

No Mesh, No Run !

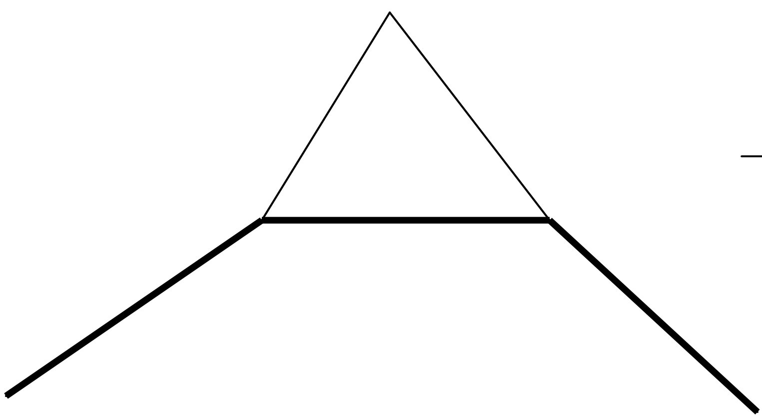
Unstructured Grids

- Advancing Front: One Element At A Time
  - Fill Empty Space
  - All Lengthscales Local
  - Loss of Mathematical Rigour: Face Crossings
  - Loss of Mathematical Rigour: Sweep and Retry
- Delaunay: One Point At A Time
  - Modify Exiting Grid (Cavity Operator)
  - Lengthscales Progressively Local
  - Loss of Mathematical Rigour: Accept Only Good Tets
  - Loss of Mathematical Rigour: Boundary Recovery

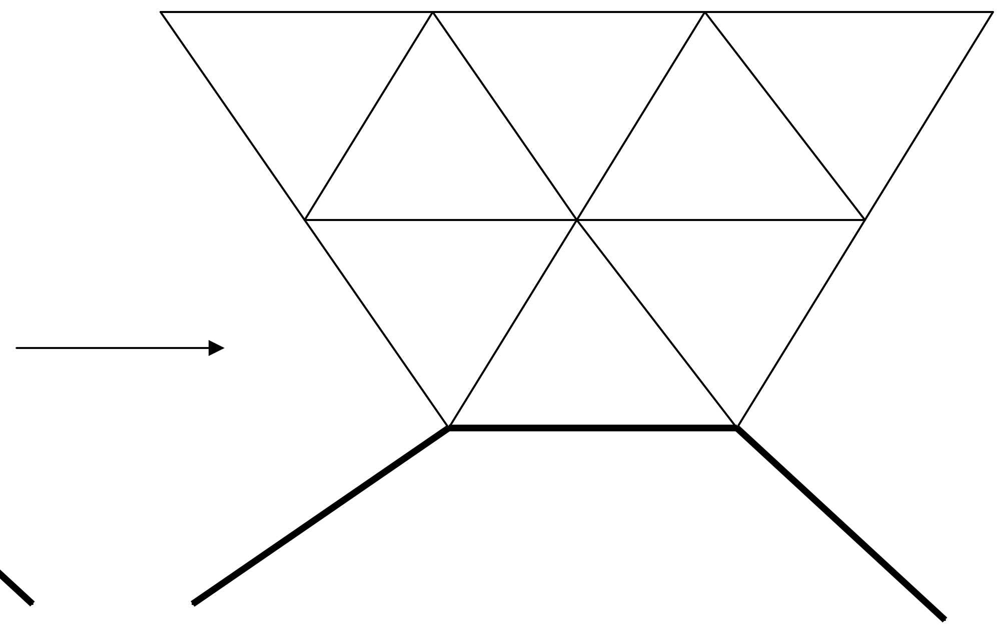
Advancing Front

- Increase Speed For:
- Scalar
- SMP (Shared Memory Parallel)
- DMP (Distributed Memory Parallel)

Introduce Multiple Elements ?



1 Element (Usual)



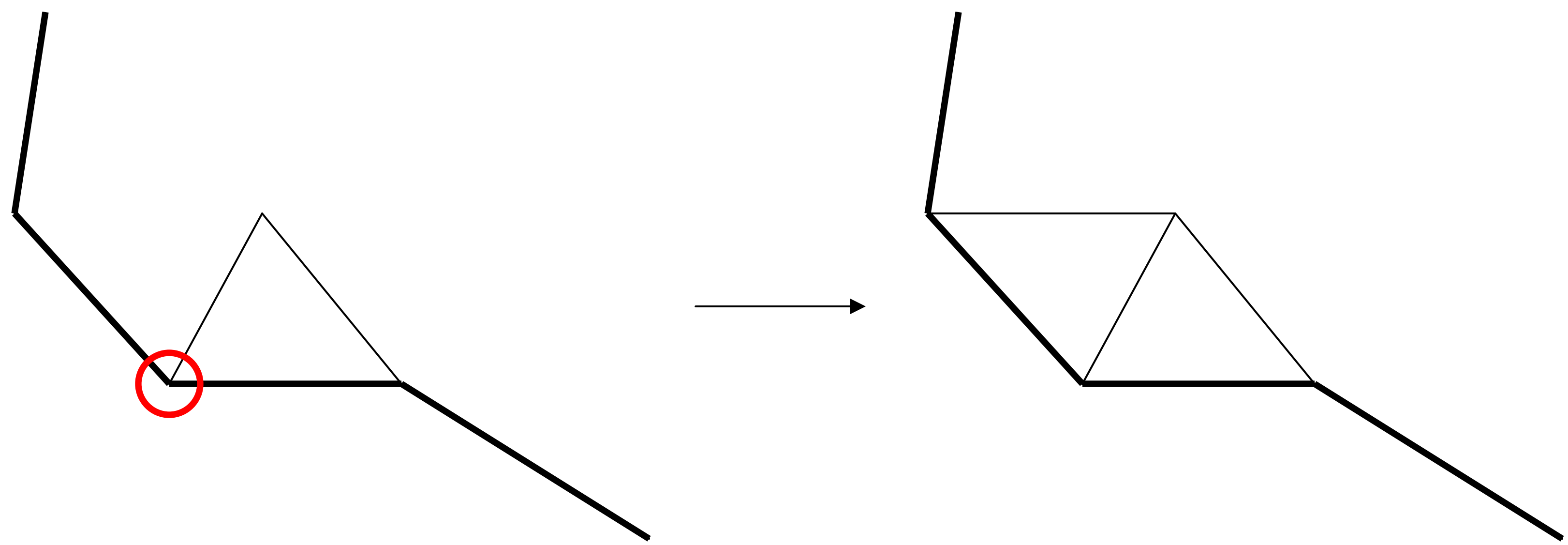
Multiple Elements (3, 8, …)

Introduce Multiple Elements ?

- Idea: Introduce New Element As Before
- If New Point:
  - See If Neighbourhood `Open' [Adjacent Faces]
  - See If Multiple Elements Possible
  - Add New Elements/Points/Faces
- Several `Multiple Element' Topologies Possible
- Easy to Add to Usual Advancing Front
- Status: Implemented, Initial Timings Inconclusive

Fast Gap Closure ?

- Idea: Exploit Adjacent Face Info/Topology



Fast Gap Closure ?

- Idea: Introduce New Element As Before
- See If Neighbourhood `Closed' [Adjacent Faces]
- See If Immediate Closing Possible (Quality)
- Easy to Add to Usual Advancing Front

- Status: Not Yet Implemented

Scaling Up…

Scaling Up (1)

- Current Runs: $O(10^3)$ Procs/Cores
  - ➔ Complete Mesh Still on Large-Memory Node
  - Mature (Scalar) Grid Generators ➔ Generate Mesh on Large-Memory Node
  - Splitting on Large-Memory Node
  - Field Solver(s): Distributed
  - Coupled Codes: Info Transfer Via: All to 1; 1 to All

Scaling Up (2)

- Foreseeable Future: $O(10^6)$ Procs/Cores ➔ Factor 10^3 ➔
  - Complete Mesh Will Not Fit on Any Node
  - Need **Parallel, Distributed Mesh Generation**
  - Need Parallel, Distributed Splitting/Repartitioning
  - Need Parallel, Distributed Info Transfer for Coupled Runs

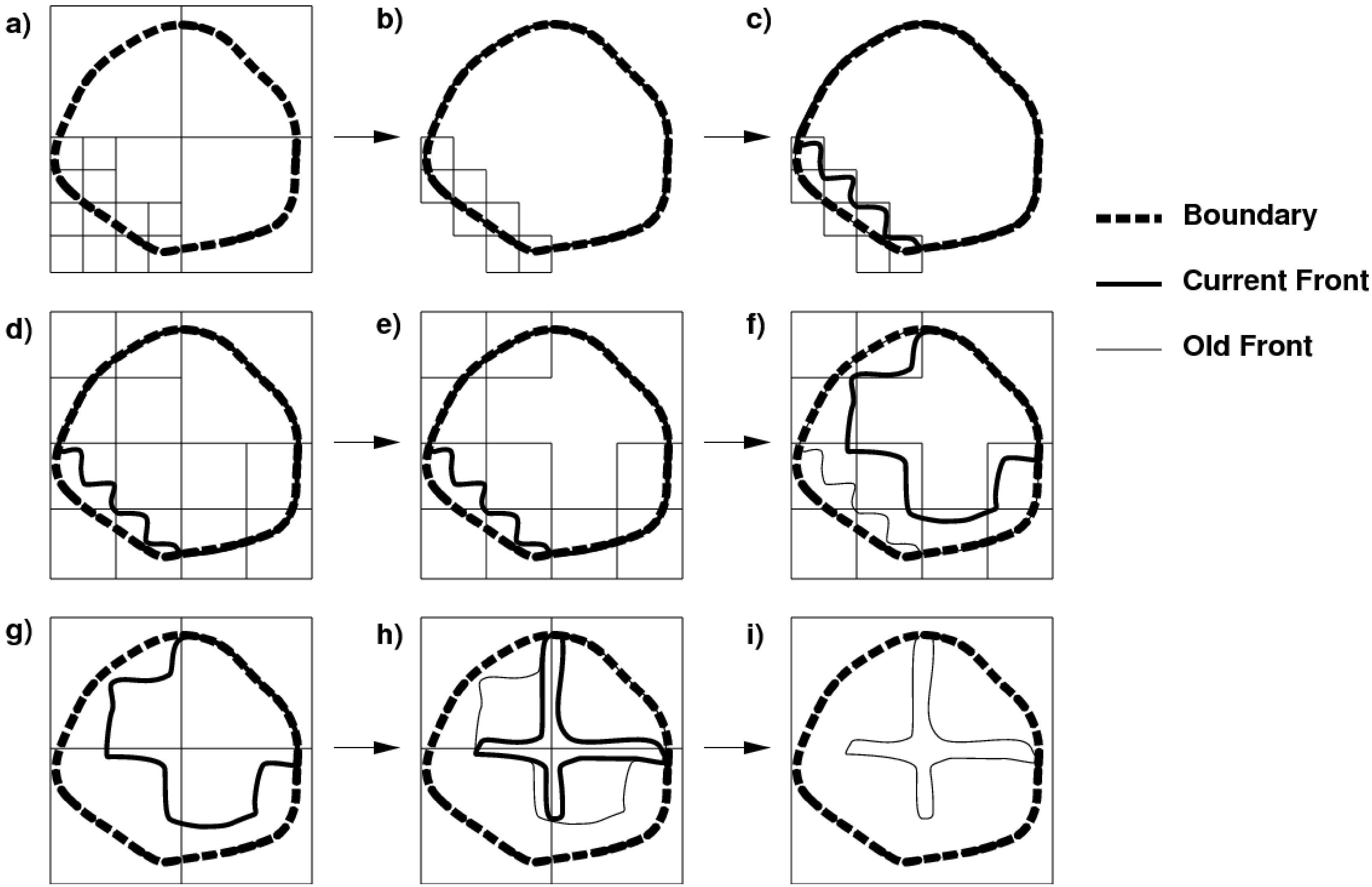
Unstructured Grid Generators

- Scalar
- Large Variation in OPS
- Parallel by `Distance‘

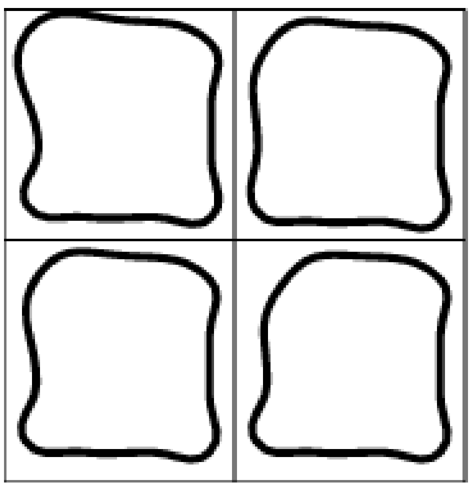
- ➔ Porting to Parallel Machine:
  – Break Up Problem Into Pieces
  – Each Processor a Piece
  – Interprocessor Transfer of Info
  – Assemble

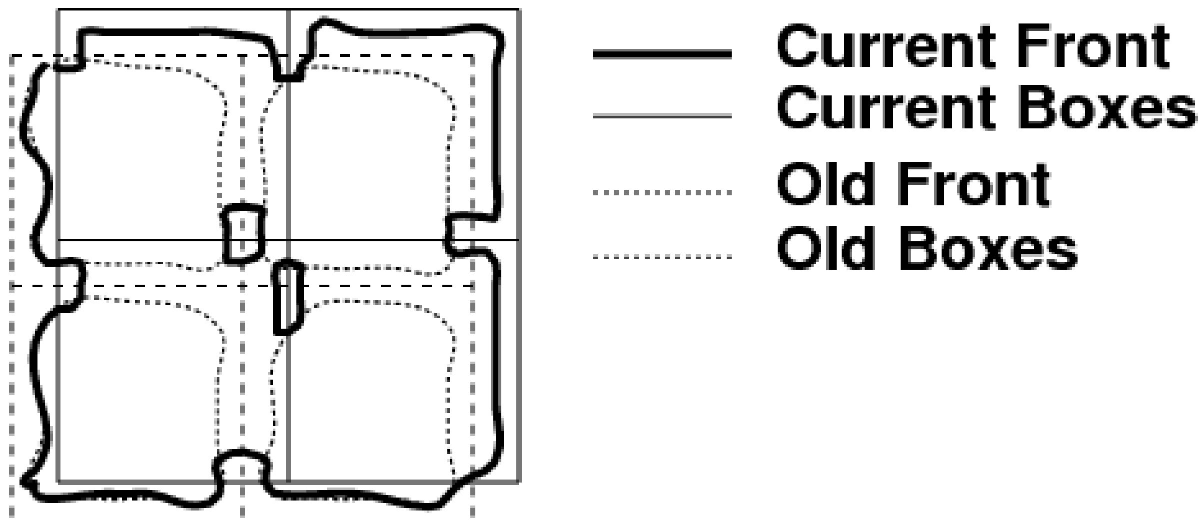
SMP Grid Generation

SMP-Parallel Advancing Front (2000)



Shift + Mesh

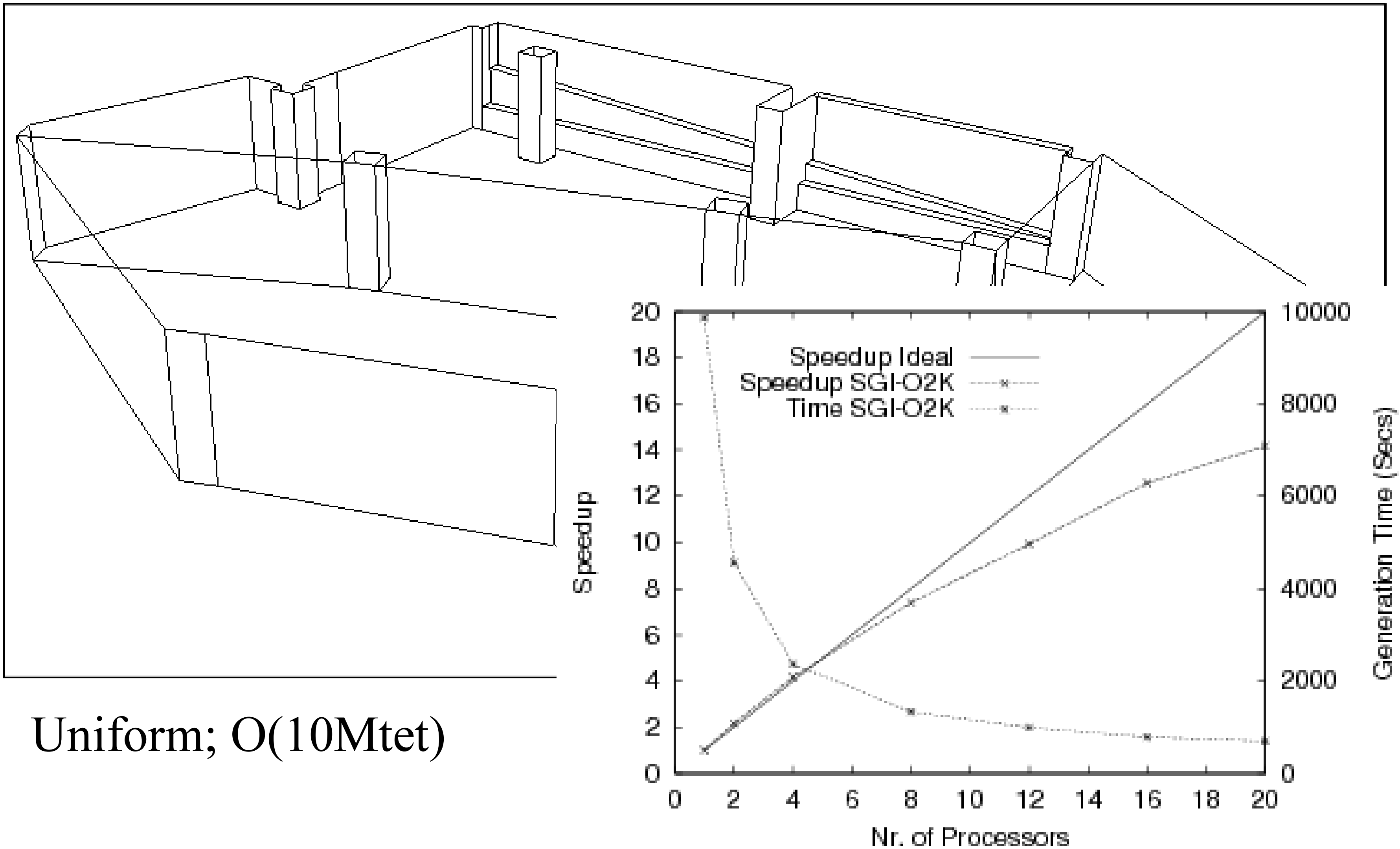




Parallel Advancing Front

- WHILE: Active Faces Left:
  - Build Octree of Active Faces
  - Retain Octants With Faces Generating Small Elements
  - (Shift) + Mesh Octants in Parallel
  - Reassemble Remaining Faces
- END WHILE

Garage (2000)



Uniform; O(10Mtet)

Shared-Memory Parallel GridGen

- Working, BUT:
- Scalability Limited [nprol=32]
  - Front-Based Parallelism
- Mesh Size Limited [RAM]
  - 10^9 Elements > 250 Gbytes RAM [!]

- ➔ Need Distributed Memory Parallel GGen
- ➔ Need Domain-Based Parallelism

DMP Grid Generation

Massively Parallel Mesh Generation

- Key Requirements:
- Very Large Meshes ➔ Distributed Memory ➔
- Simple, Effective Definition of Domain to be Gridded
  - In/Out Problem
- Parallel Load Balancing
  - During Generation: Advancing Front
  - Post-Generation: Smoothing, Diagonal Swapping

DMP Parallel Mesh Generation

- Löhner, Camberos, Merriam, Shostko [1992 (!), 1995]
  - Advancing Front; Domain Splitting of Background Grid [2/3-D]
- deCougny, Shepard, Ozturan [1994, 1995]
  - Modified Octree, [2/3-D]
- Okusanya, Peraire [1996, 1997]
  - Delaunay [2-D, 3-D]
- Christochoides, Chew, Nave [1997, 2003, 2005]
  - Domain Decomposition, 2-D
- Weatherhill, Hassan, Tremel [2006]
  - Delauney [3-D]
- Ivanov, Andrae, Kudryavtsev [2006]
  - Domain Decomposition

Massively Parallel Mesh Generation

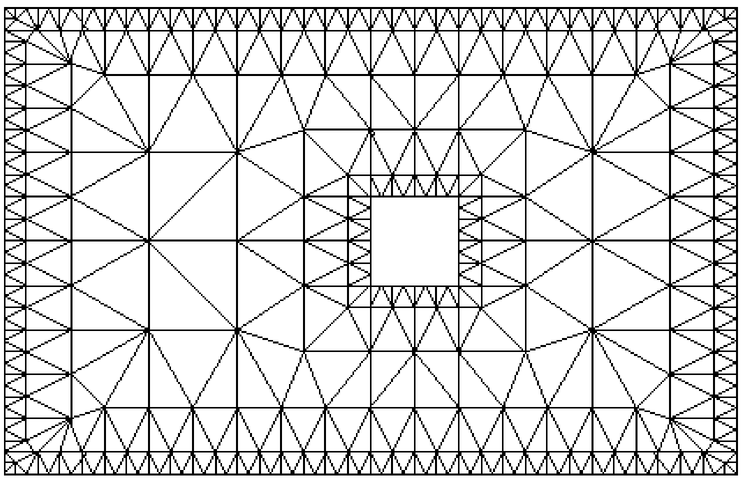
- Shortcomings of 1st Generation ParGen:

- No Simple, Effective Definition of Domain to be Gridded
- ➔ <u>Solution:</u> Domain-Defining Grid (DDG)

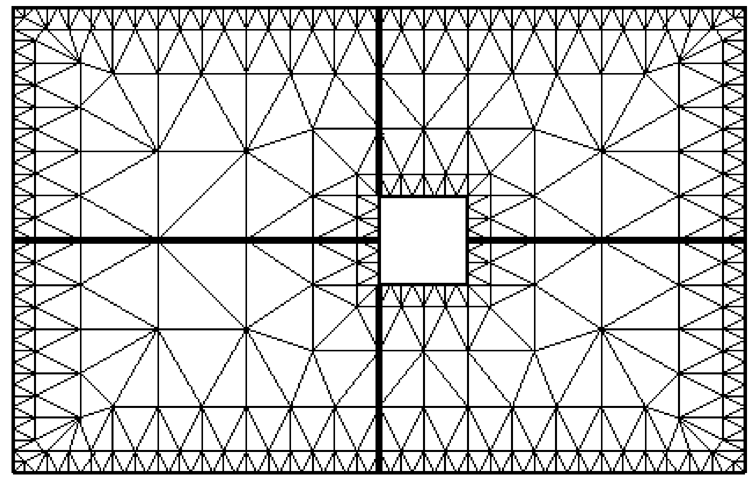
- Loss of Parallel Load Balancing After 1st Pass (`log-Trap')
- ➔ <u>Solution:</u> Re-Splitting of DDG

Domain Defining Grid (DDG)

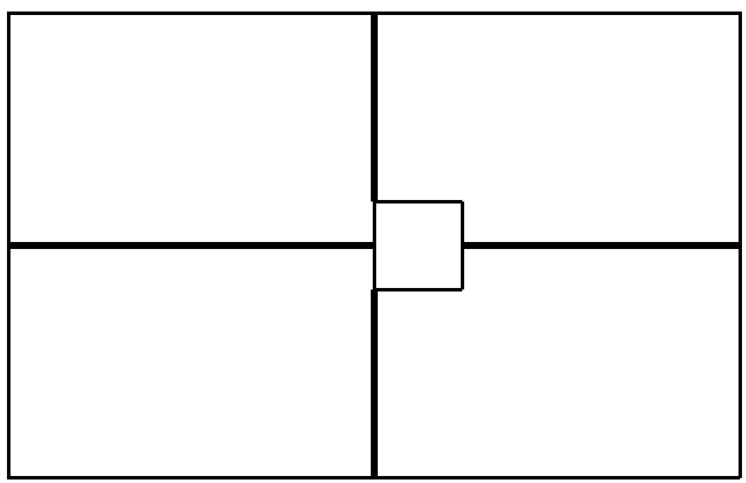
Coarse Mesh (To Define Space to be Gridded)

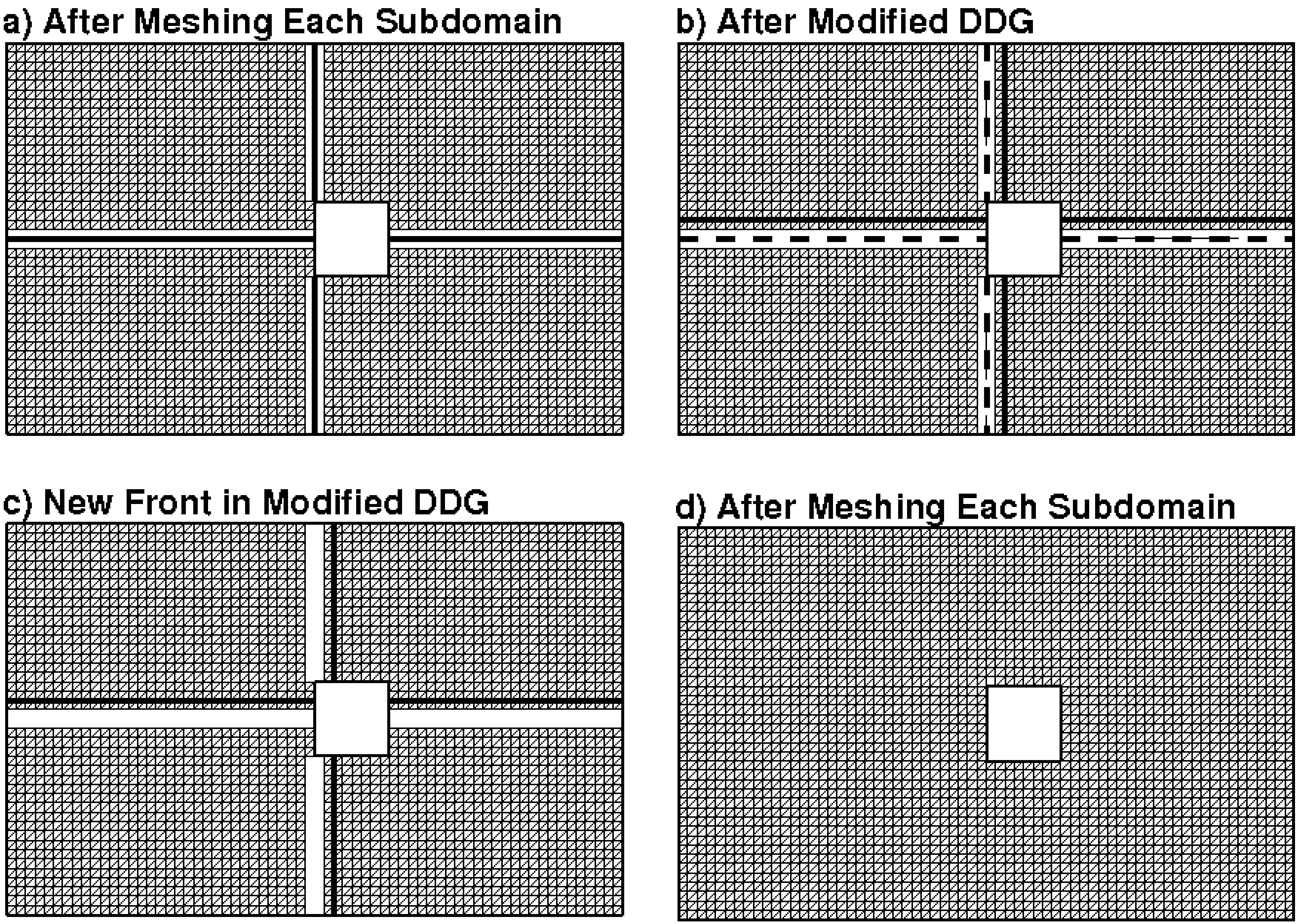


Load Balancing for Fine Mesh Generation



After Load Balancing




a) After Meshing Each Subdomain
b) After Modified DDG
c) New Front in Modified DDG
d) After Meshing Each Subdomain

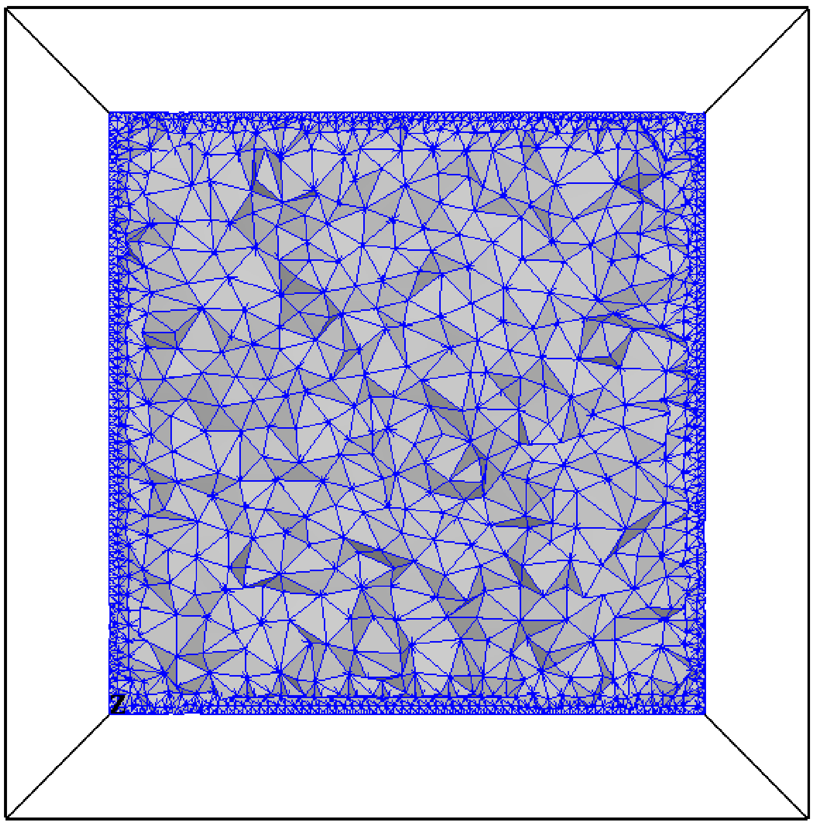
Massively Parallel Mesh Generation

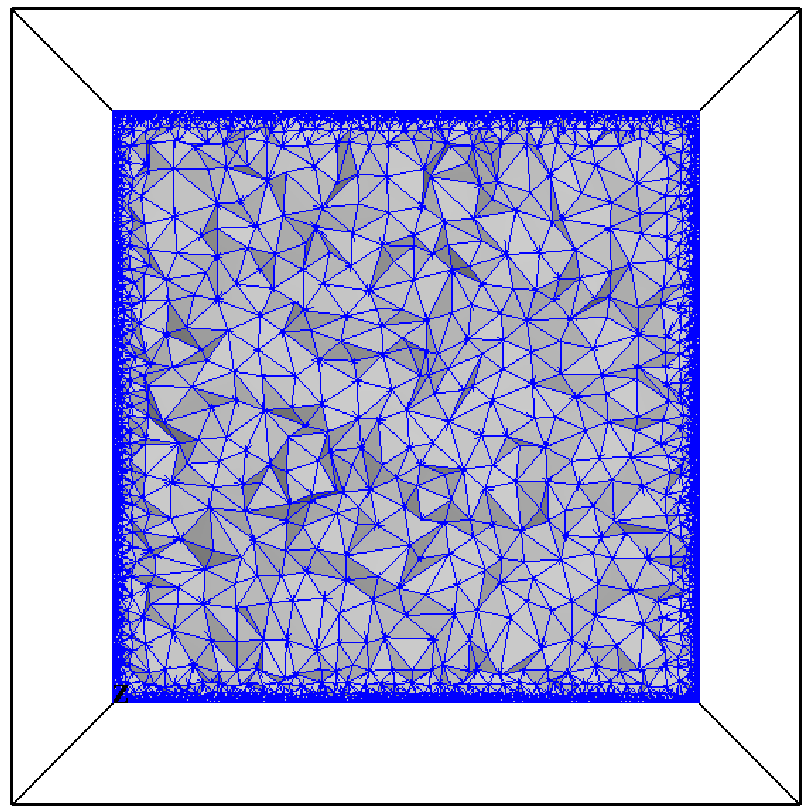
- Step 1: Define the Domain to be Gridded
- Easiest Form of Definition: Mesh
- BUT: Need Fine Surface Mesh ➔
- Use Advancing Front With Fine (Required) Surface Mesh
- Generate Maximum Possible Element Size in Domain
- ➔ `Domain Defining Grid' (DDG)
- Split DDG to Distribute Work

Massively Parallel Mesh Generation

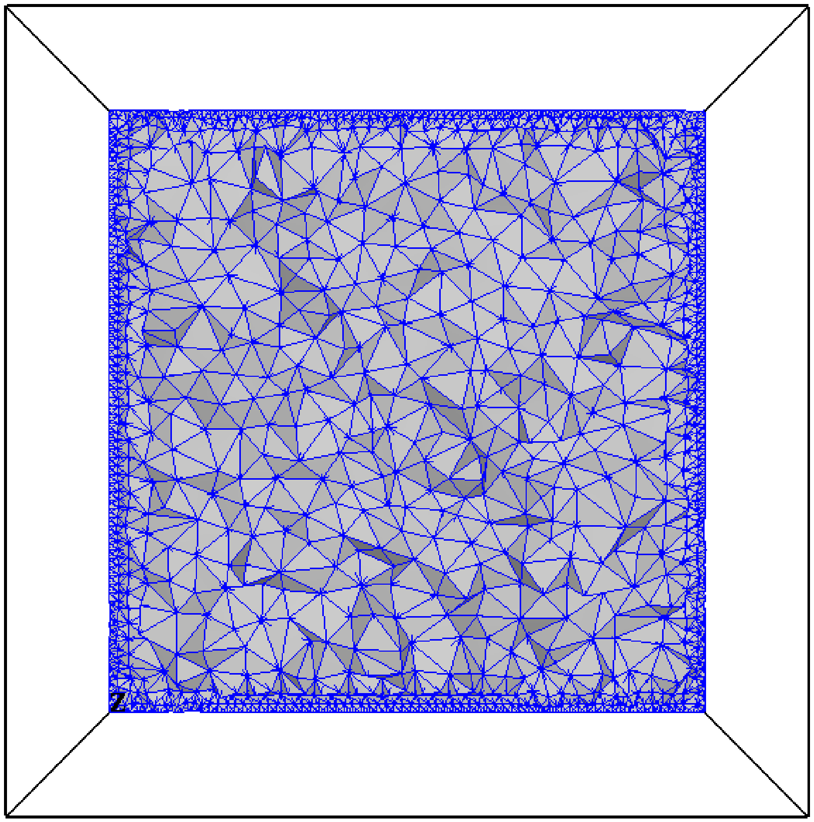
- Suppose: Volume Mesh of 10^9 Points
- ➔ Surface Mesh of 10^6 Points
- ➔ DDG of $O(10^6)$ Points
- ➔ Can Fit Surface Mesh and DDG Into Each Processor

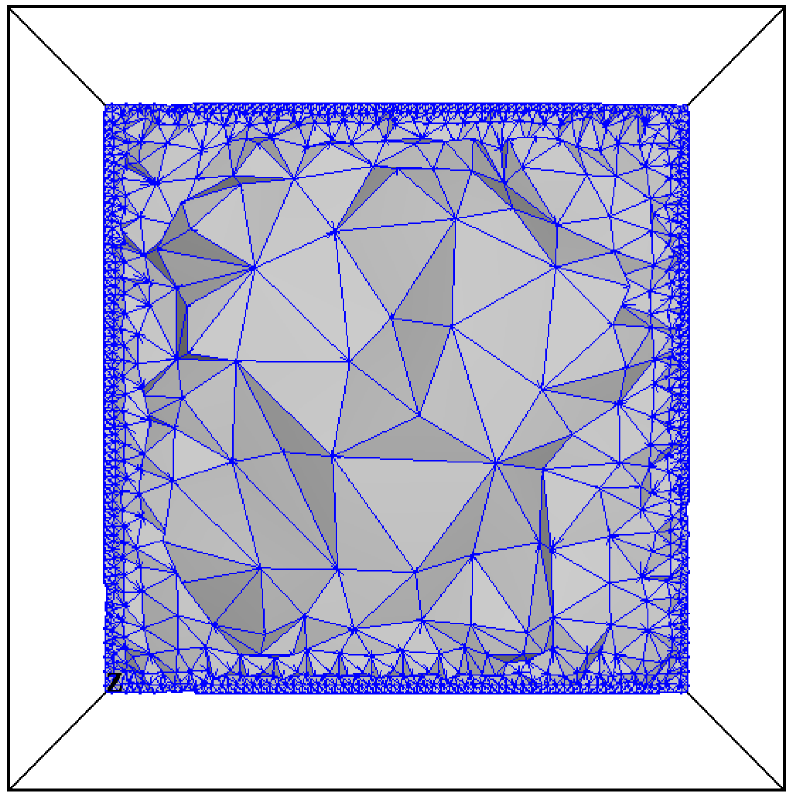
Cube: Coarsest Possible Mesh



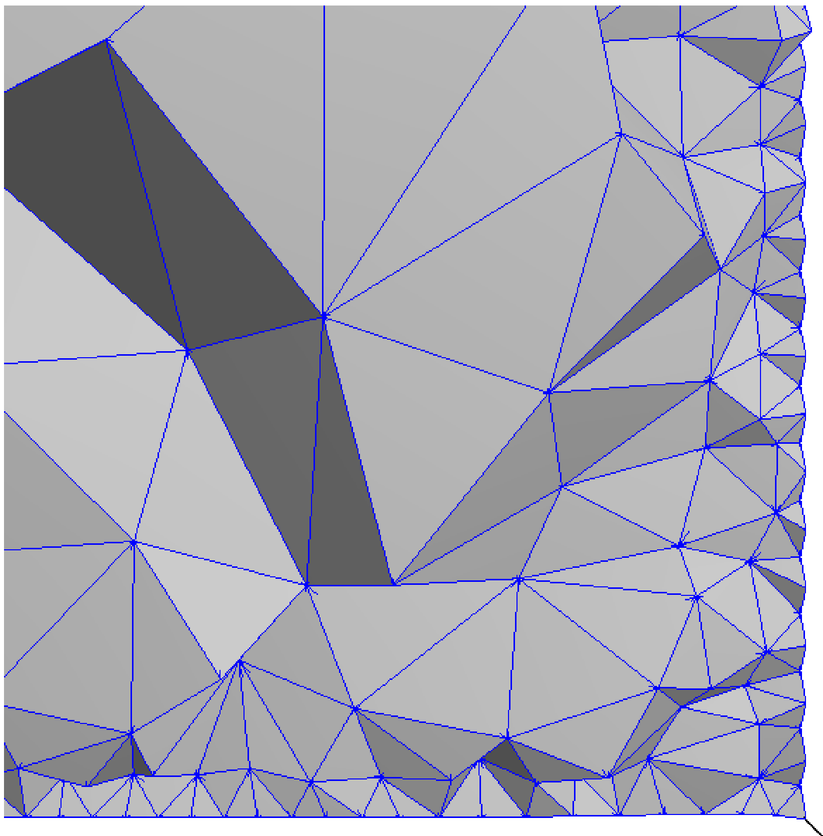


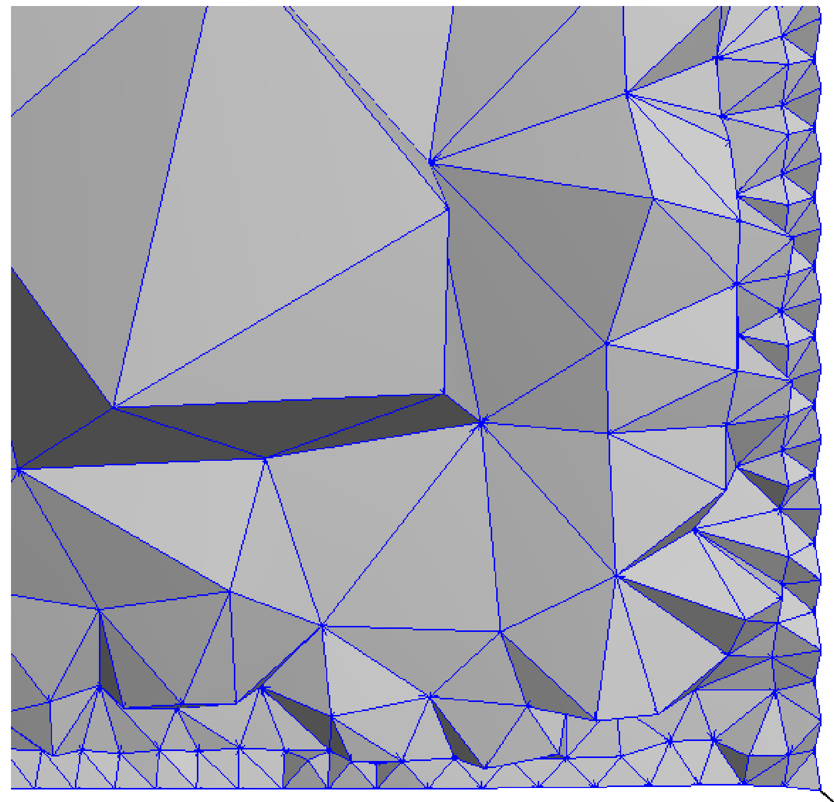
Cube: Coarsest Possible Mesh





Cube: Coarsest Possible Mesh





Massively Parallel Mesh Generation

- Step 2: Split DDG to Distribute Work

- Can Use any Domain Decomposition Technique
  - Even Scalar Ones
- Used Here:
  - Advancing Front
  - Recursive (Coord/Moment) Bisection
  - Peano-Morton-Hilbert Space Filling Curve

- Result: Each Processor Has Surface Faces, DDG+SplitInfo

Massively Parallel Mesh Generation

- Step 3: Generate the Mesh in Parallel
- Done in Passes; In Each Pass:
  - Filter Surface Faces Required + Mesh
  - Send Active Faces to Proper Processors/Domains
  - Store Mesh in Processor/Domain (Distributed)
- Result: Each Processor Has Mesh It Generated
- Pass 1: Take Initial Splitting of DDG
- Pass 2,…: Add 1-2 Layers From Neighbours With Higher/Lower Domain Nr.
  - Integer / Size / Distance…

Massively Parallel Mesh Generation

- Step 4: Redistribute the Mesh

- Assign Element to Domain Based on Lowest Point-Nr. In DDG Domain/Region
- Send to Required Processors
  - Needs Colouring Scheme
- Remove Duplicate Points
  - Use Octree Search

- Result: Each Processor Has Mesh It Generated

Massively Parallel Mesh Generation

- Step 5: Improve the Mesh
- Fix the Points That Are at the Boundary of the Mesh
  - True Boundary
  - Boundary of Domain/Processor
- Perform Edge Collapse/Smoothing/Diagonal Swap/…
- Step 5a: Redistribute the Mesh Again and Redo Step 5
- Result: Each Processor Has Mesh It Generated, Improved

Massively Parallel Mesh Generation

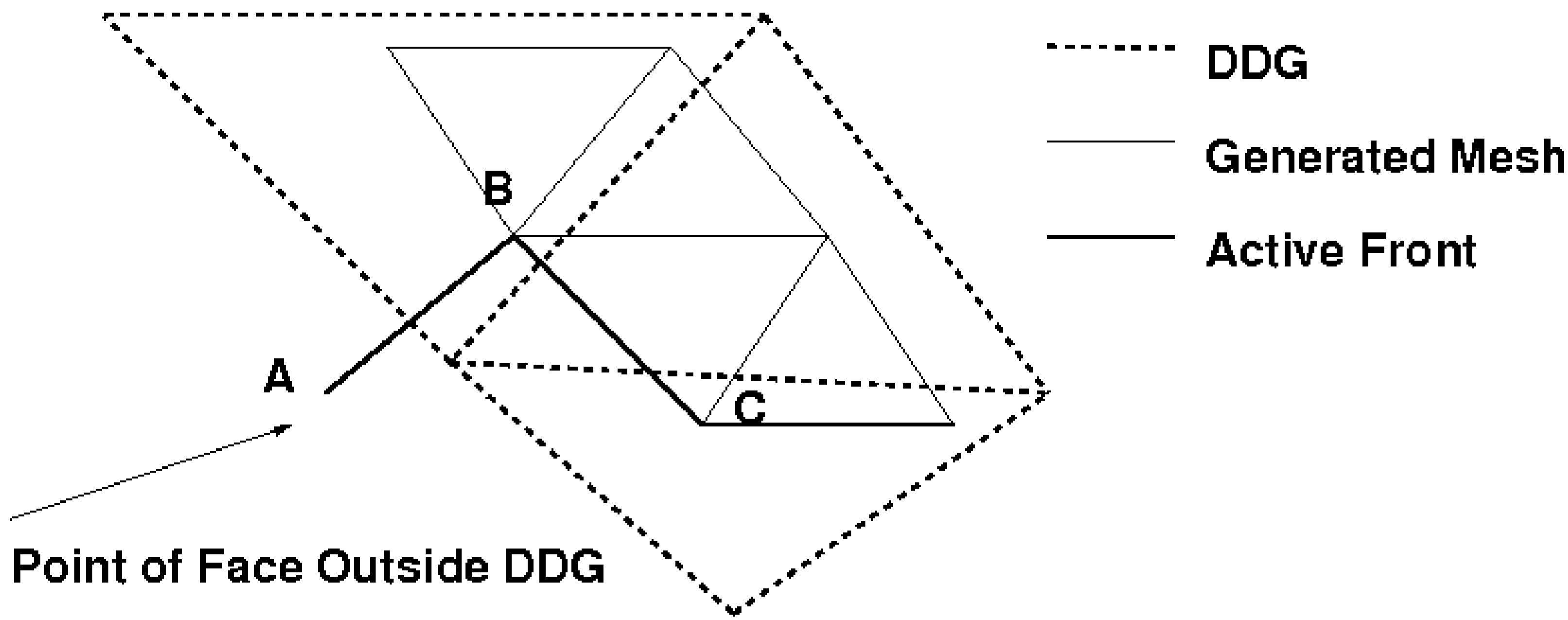
- Step 6: Output the Mesh

- Identify the Boundary Points in this Domain
- Output the Mesh in Parallel

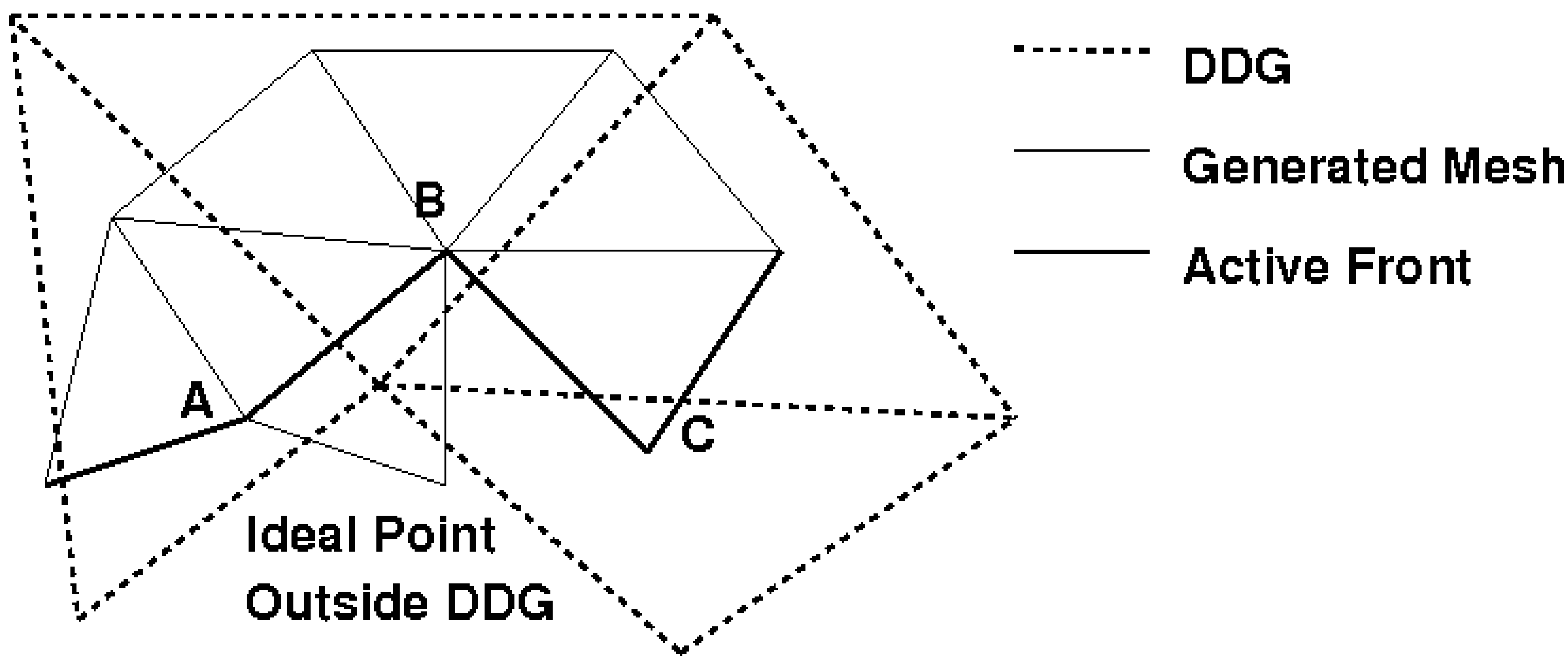
Changes to Usual Advancing Front

- If Point of Face Outside DDG: Mark + Skip
- If Ideal (New) Point Outside DDG: Mark + Skip
- If Ideal (New) Point Too Close to DDG Boundary: Skip
- If Side Of New Element Crosses DDG Boundary: Skip
  - Imposed Via NN Search

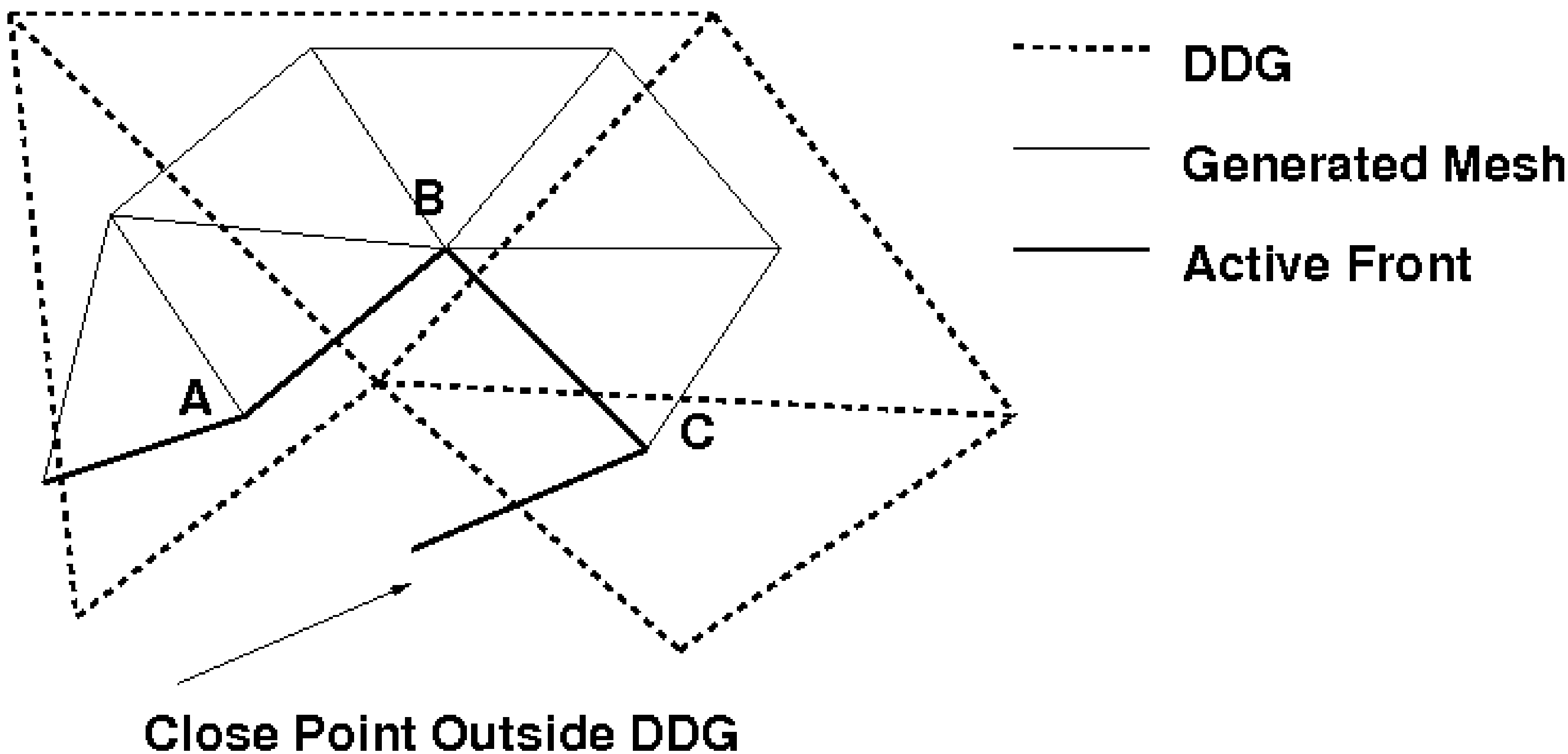
Point of Face Outside DDG



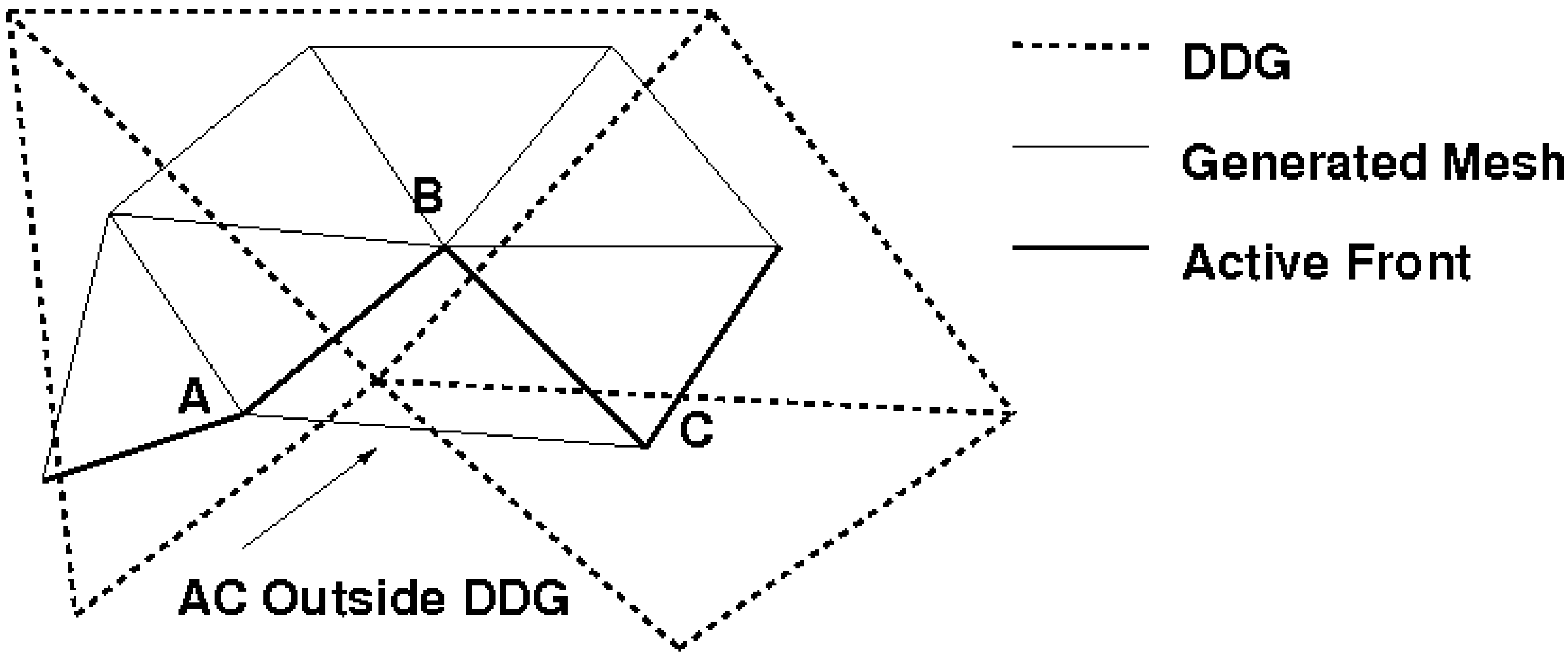
Ideal Point Outside DDG



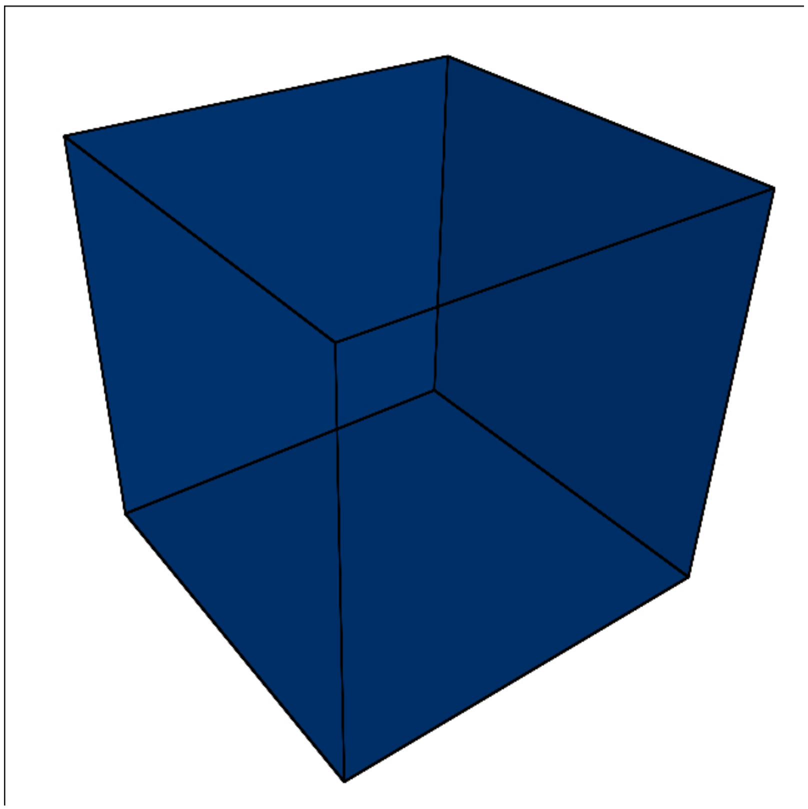
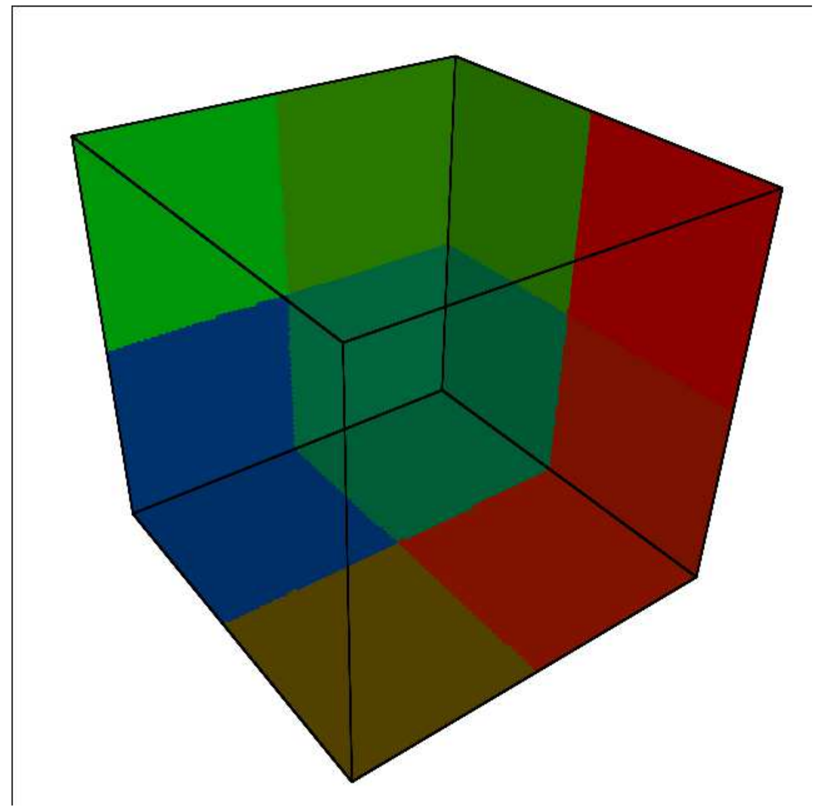
Close Point Outside DDG



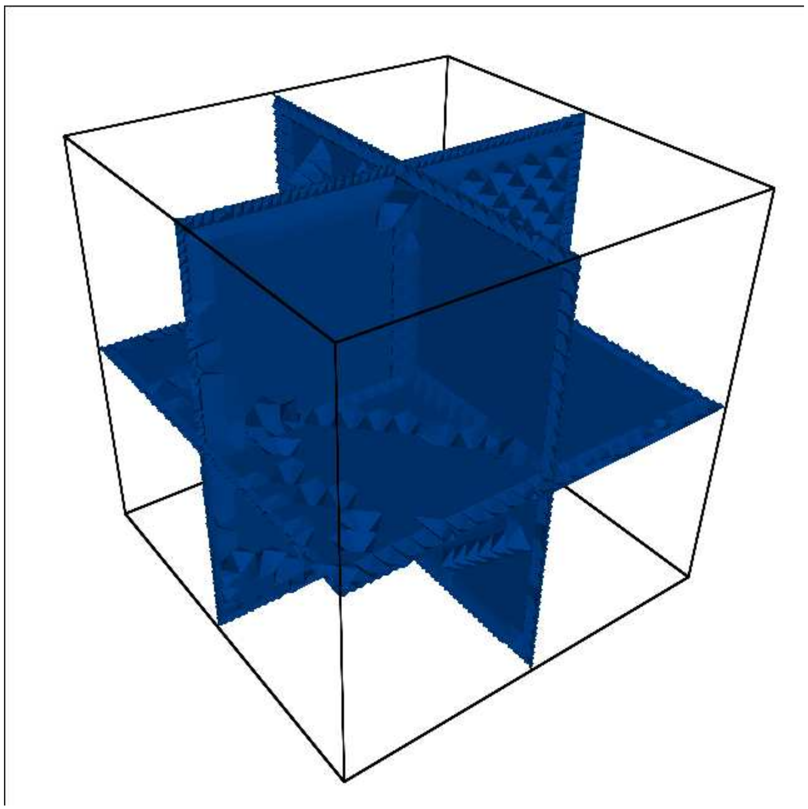
Edge of New Element Crossing DDG Boundary

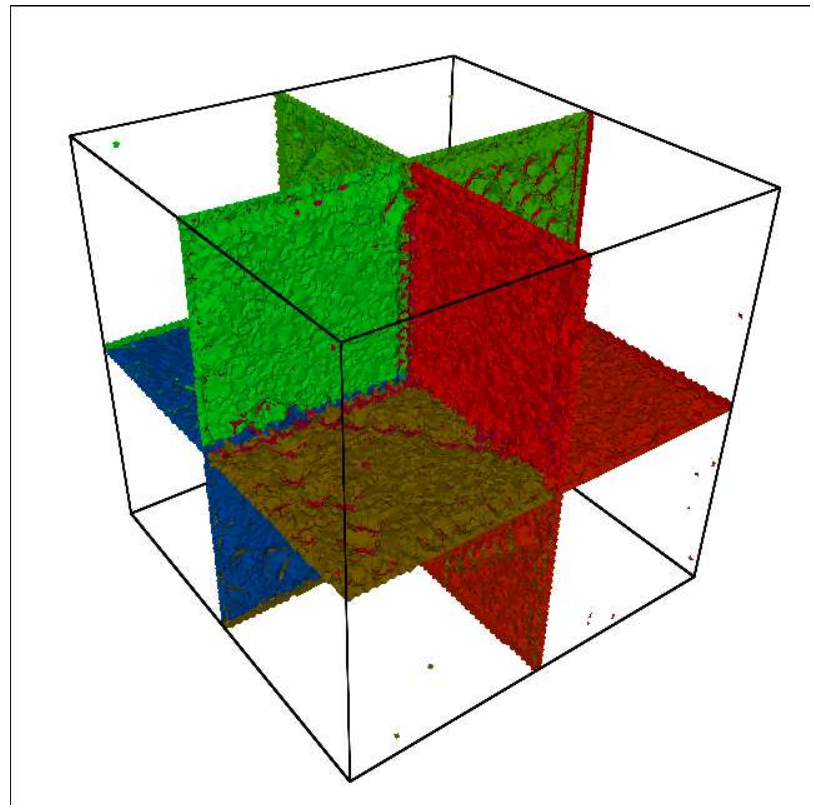


Cube: Initial Front + DDG Allocation

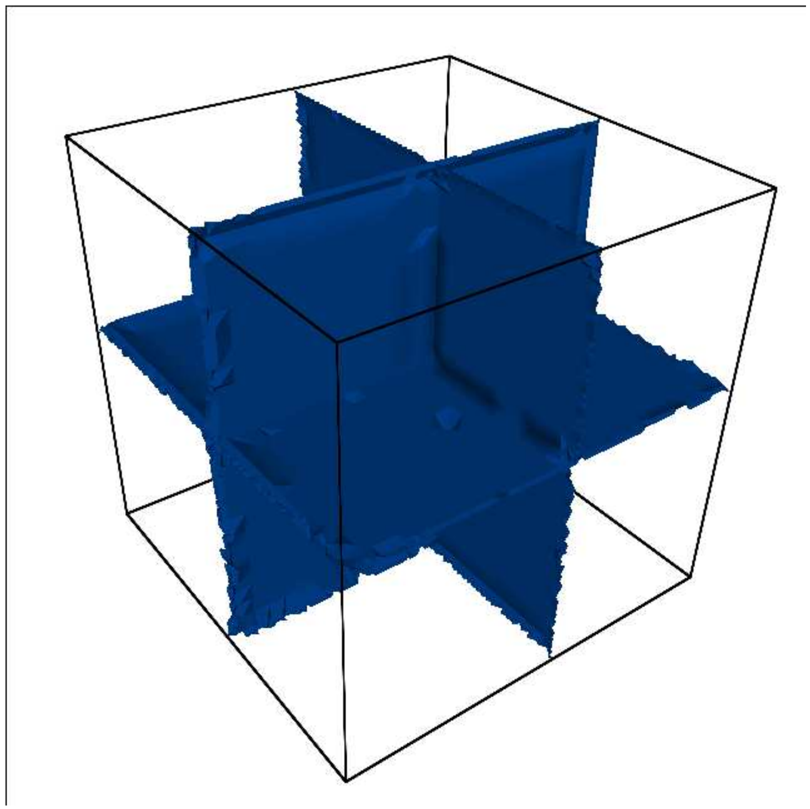
 

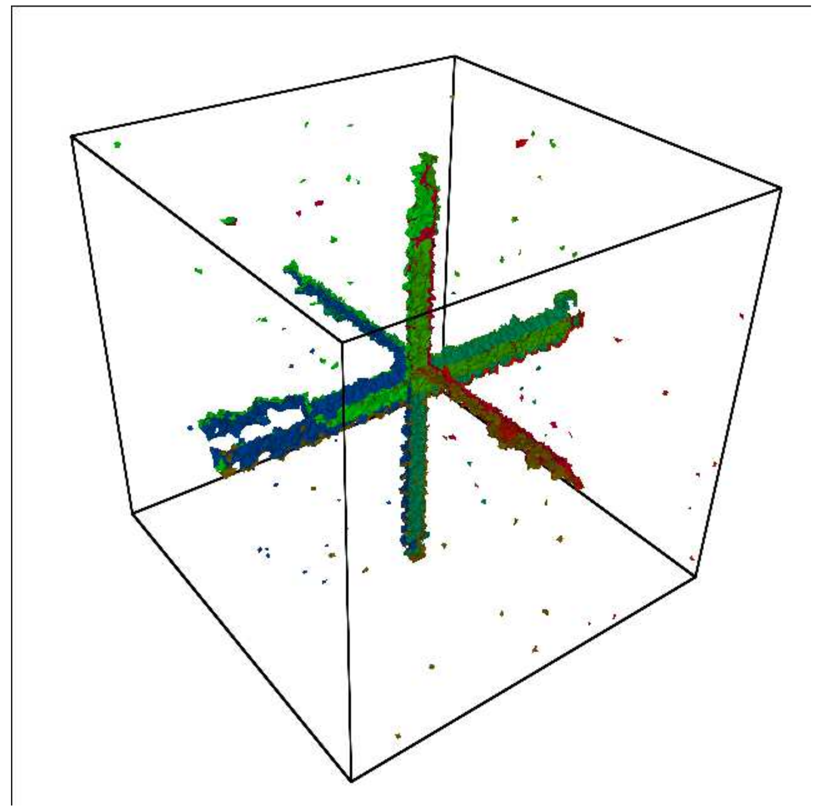
Cube: DDG + Front After Pass 1



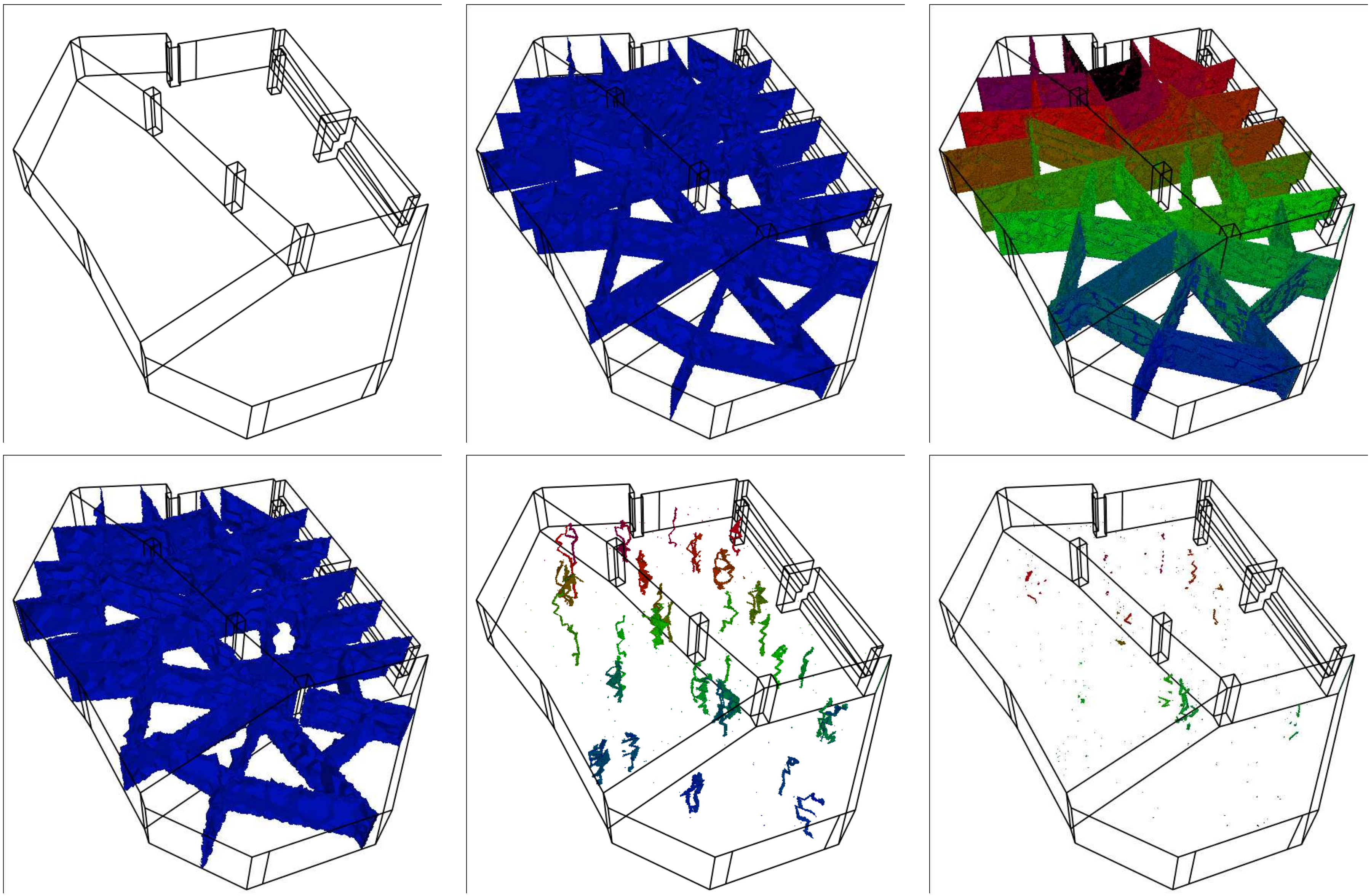


Cube: DDG + Front After Pass 2





GARAGE: Version 2



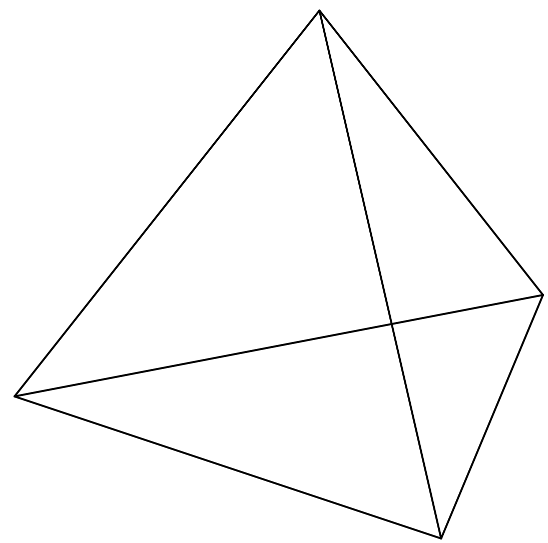
Garage: Timings

Machine	nproc	nprol	ncore	nelem	Time [sec]	AbsSpeed [Mels/sec]	RelSpeed [Kels/sec/core]
Xeon(1)	1	8	8	121 M	2293	0.052	6.54
SGI ITL	8	1	8	121 M	1605	0.075	9.42
SGI ITL	8	8	64	121 M	516	0.234	3.66
Cray AMD	8	1	8	121 M	2512	0.048	6.02
Cray AMD	16	1	16	121 M	1954	0.062	3.87
Cray AMD	64	1	64	972 M	6048	0.160	2.51
SGI ITL	32	1	32	121 M	646	0.187	5.85
SGI ITL	32	4	128	121 M	346	0.349	2.73
SGI ITL	64	8	512	1010 M	2504	0.403	0.79

Optimal Space-Filling Tets

Optimal Space-Filling Tetrahedra (1)

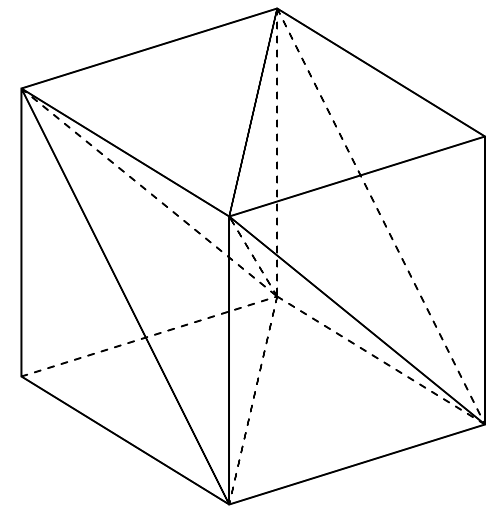
- Optimal (Equilateral) Tetrahedron: α=70.52^o
- ➔ **<u>NOT</u>** Space-Filling
- ➔ Which is Best ?

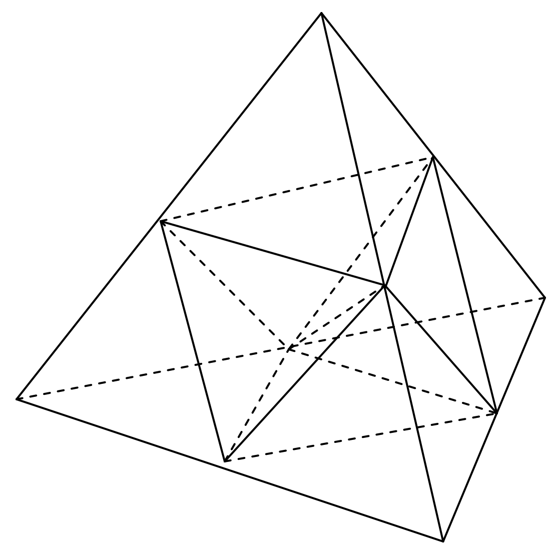


- Sommerville 1923
- Senechal 1981
- Naylor 1999
- ….

Optimal Space-Filling Tetrahedra (2)

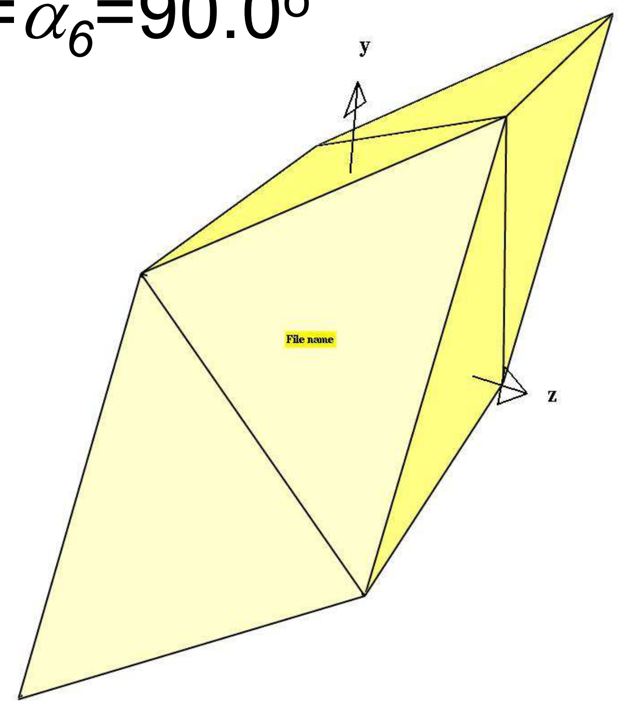
- Desired: Space-Filling With
  - max(min) angle(s), max(min) angle(s)
  - max(side)-min(side) --> min
- Approach 1:
  - Take Parallelepiped (Hexahedron)
  - Deform With Affine Transformation
    - Keeping Faces Parallel
  - Measure Quality of Tetrahedra
  - Run Through Optimizer
- Approach 2:
  - Invoke h-Refinement Argument
    - Parent = Children
  - Deform
  - Measure Quality of Tetrahedra
  - Run Through Optimizer





Optimal Space-Filling Tetrahedra (3)

- Same Result for Both Approaches: **ISOTET, BCC Lattice**
- l_{min}=1.0, l_{max}=1.157
  $\alpha_1=\alpha_2=\alpha_3=\alpha_4=60.0^o$, $\alpha_5=\alpha_6=90.0^o$



Optimal Space-Filling Tetrahedra (4)

- Graded ISOTET, BCC Lattice Grids:
- Even H-Refined Grids Have Very Good Angles
  α_{min}=O(30.0^o), α_{max}= O(90.0^o)

- Unlike Graded Grids From Cartesian Point Distributions

Incorporation in General Ggen

- Key Idea: Replace `Ideal' Point Position by Closest
  - Isotet Point
  - Cartesian Point

- For Graded Grids:
  - Take: h_{min}= min(isotropic element size)
  - Given Current h(**x**): $h = h_{min}\,2^{\alpha} \Rightarrow \alpha \Rightarrow h_c = h_{min}\,2^{\,int(\alpha)}$

- ➔ Very Simple Change

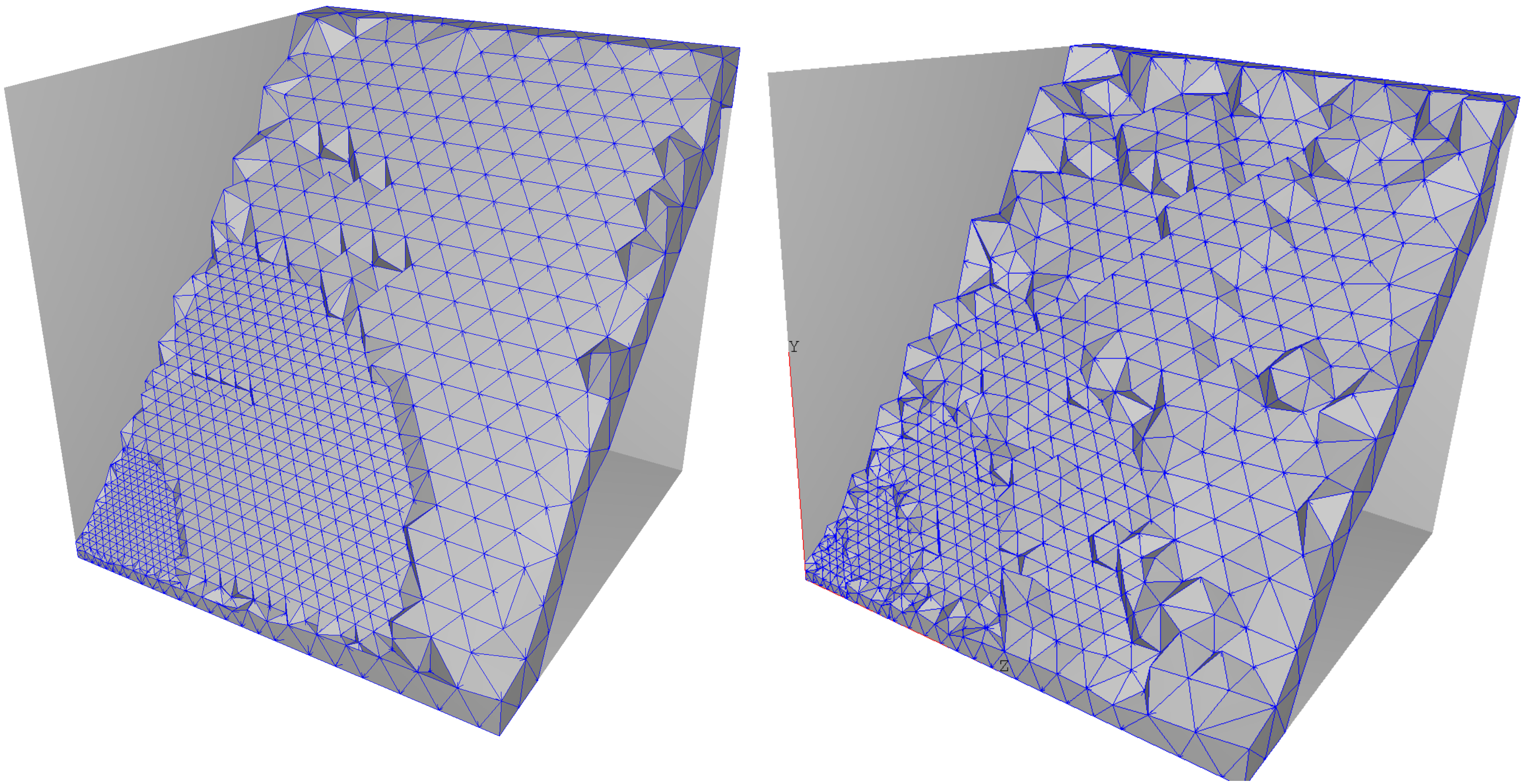
Comparison for Graded Meshes

- Cartesian/Isotet Placement Will Require More Points
  - Increase in Mesh Size Always By a Factor of 2
- Worst Case Scenario: 8:1 (!)
  - Take Box; Corner 1: h_{min} ; Rest: 1.999 h_{min}
- Grading in 1-D: 4:3 (!)
  - Take Box; Corner 1: h_{min} ; Rest: $h= h_{min} (1+ \alpha x)$
- Realistic Grids: Somewhere Between …

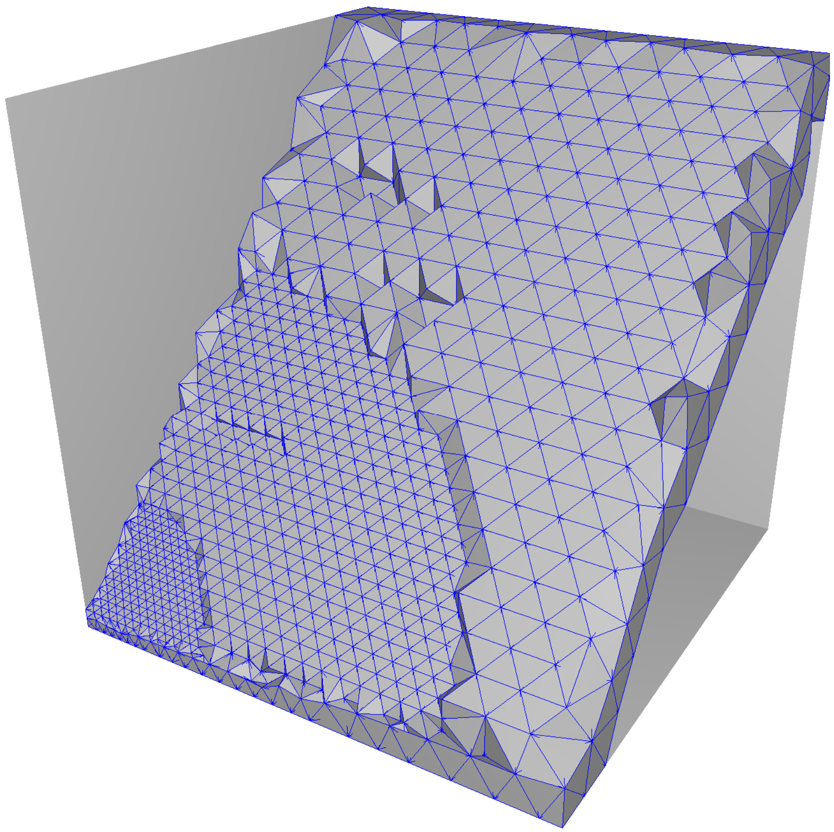
Regular Point Distributions

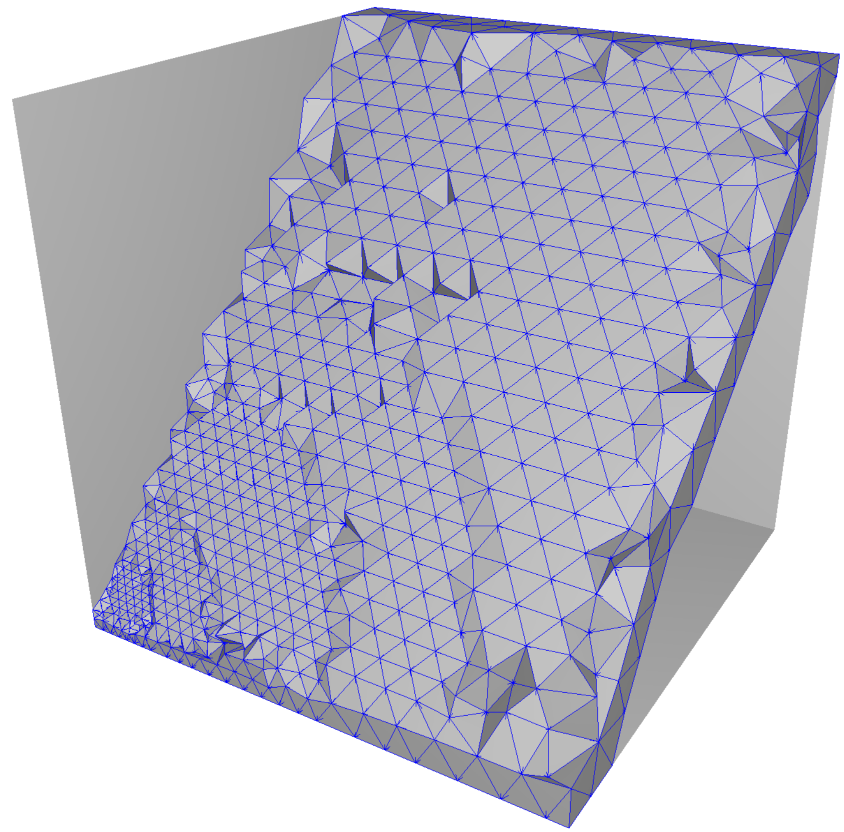
- Reflections Reported at 1:2 Transitions
- ➔ Attempt `Softer' Transitions
- Increase Factor: 1.5
- Increase Factor: 1.414 [=sqrt(2)]
  - `Double Every 2nd Jump'
- Both Possible
- Same Procedure As Before:
  - Find Ideal Point
  - Get Closest Cartesian/Isotet/BCC Point & Check

cincr=1.5

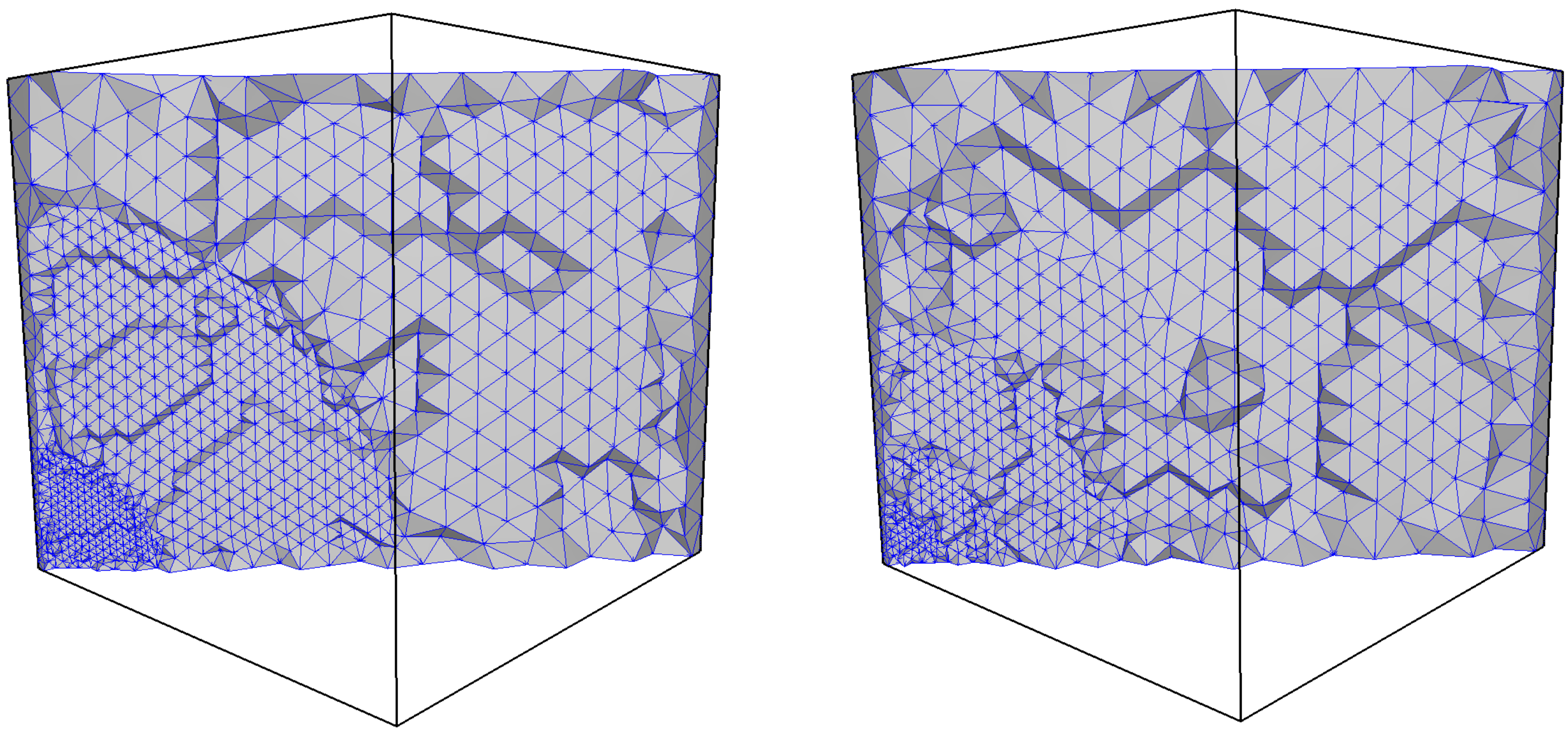


cincr=1.4

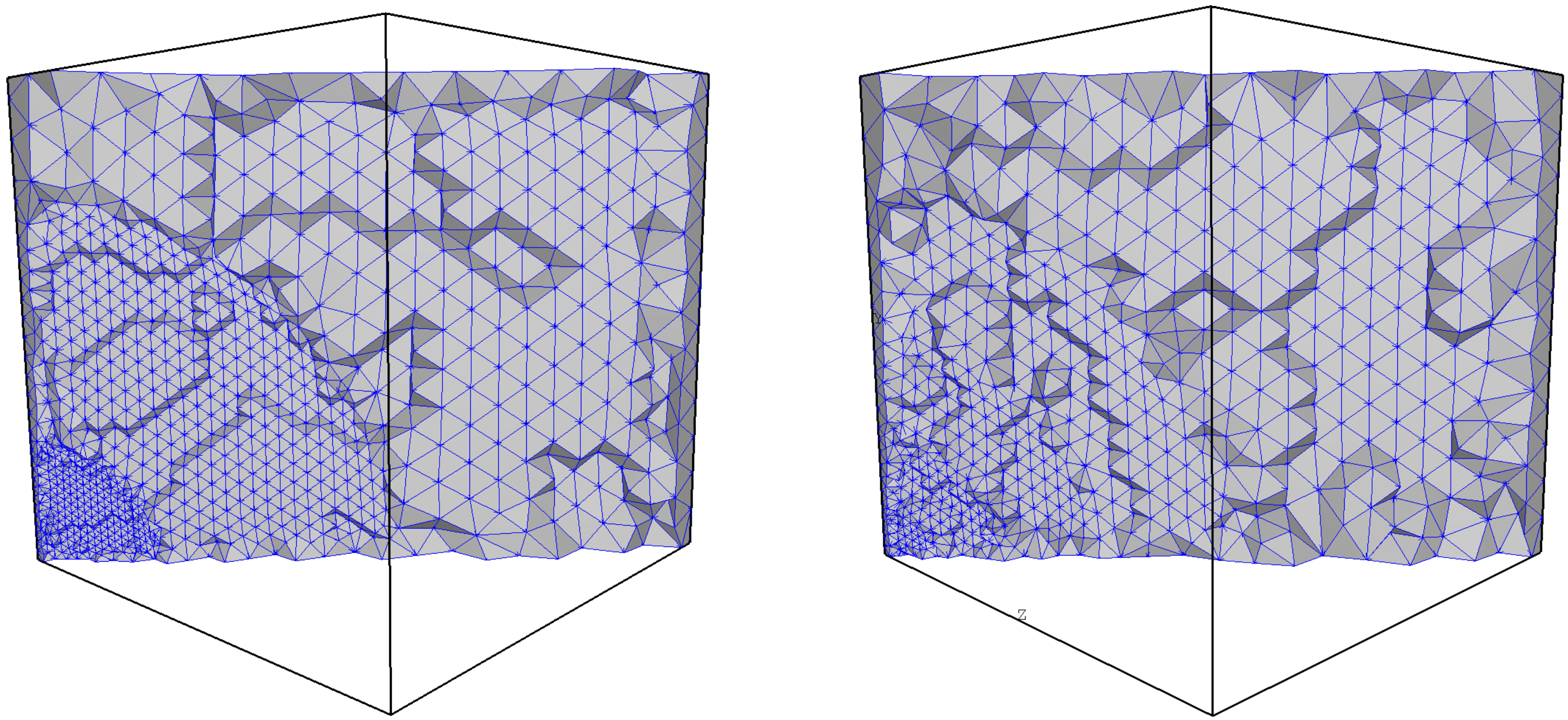




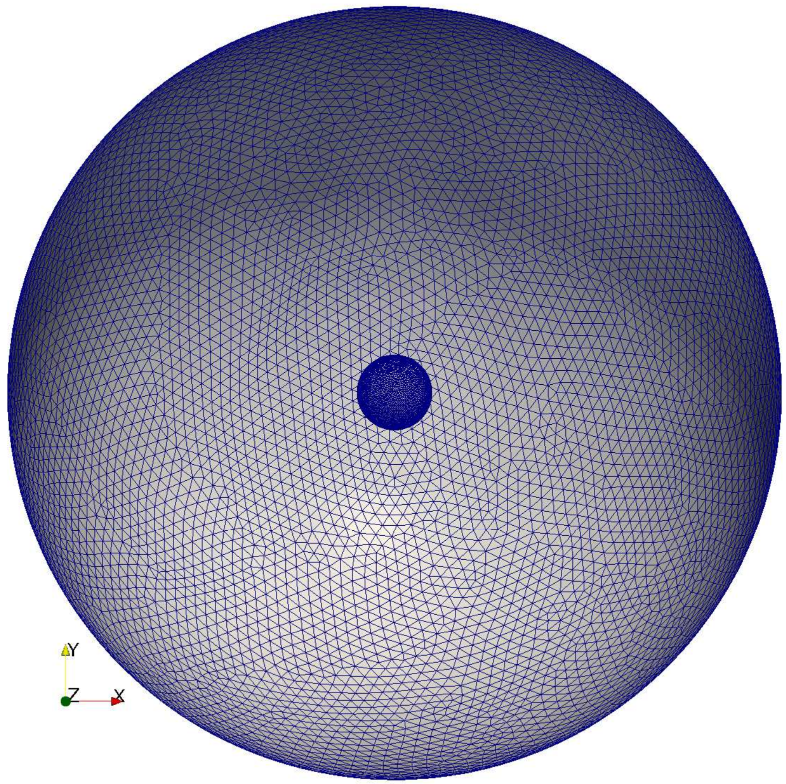
cincr=1.5

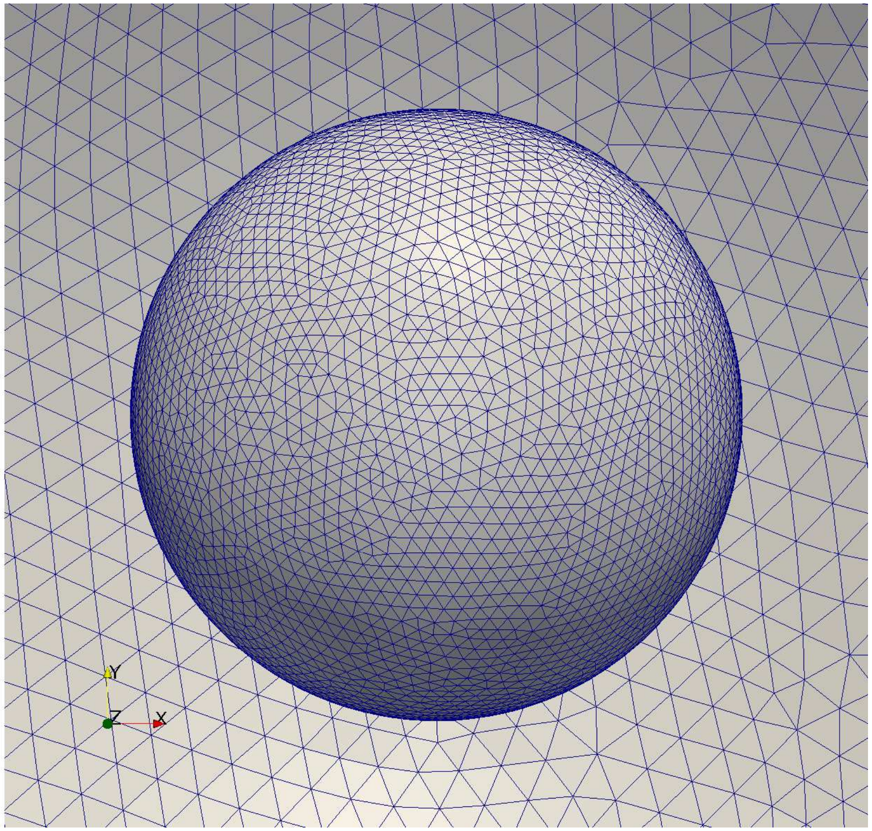


cincr=1.4

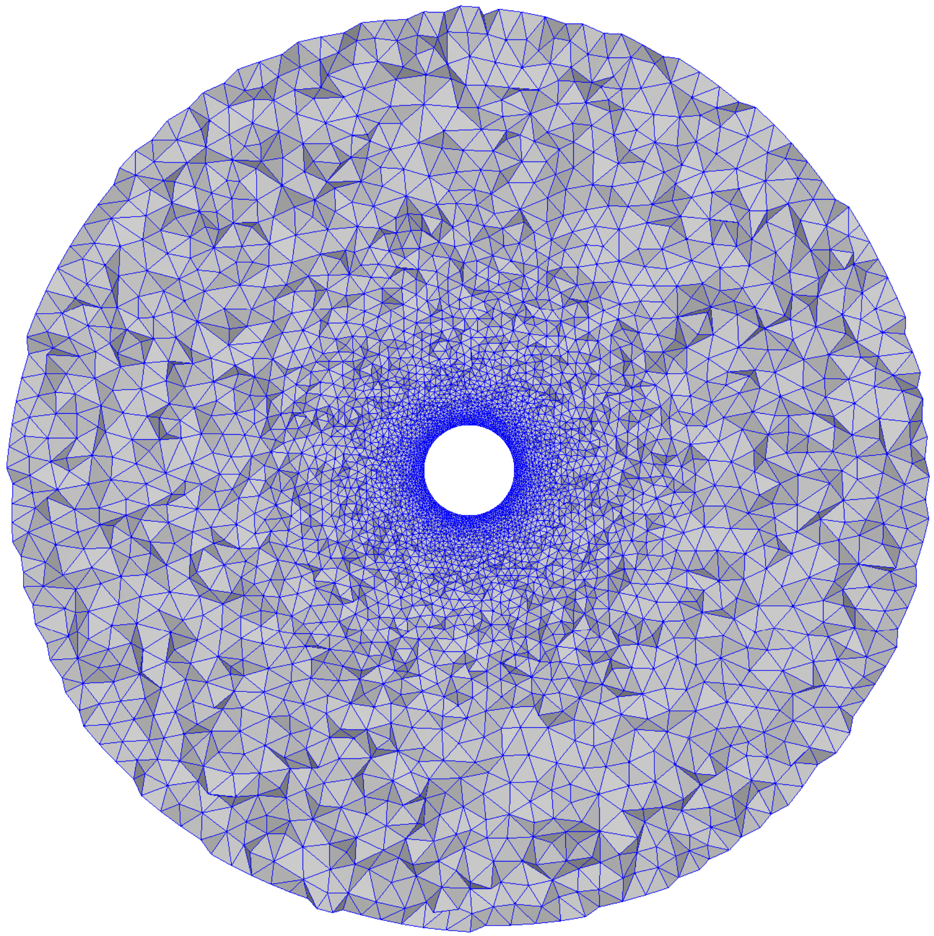


Sphere in Sphere

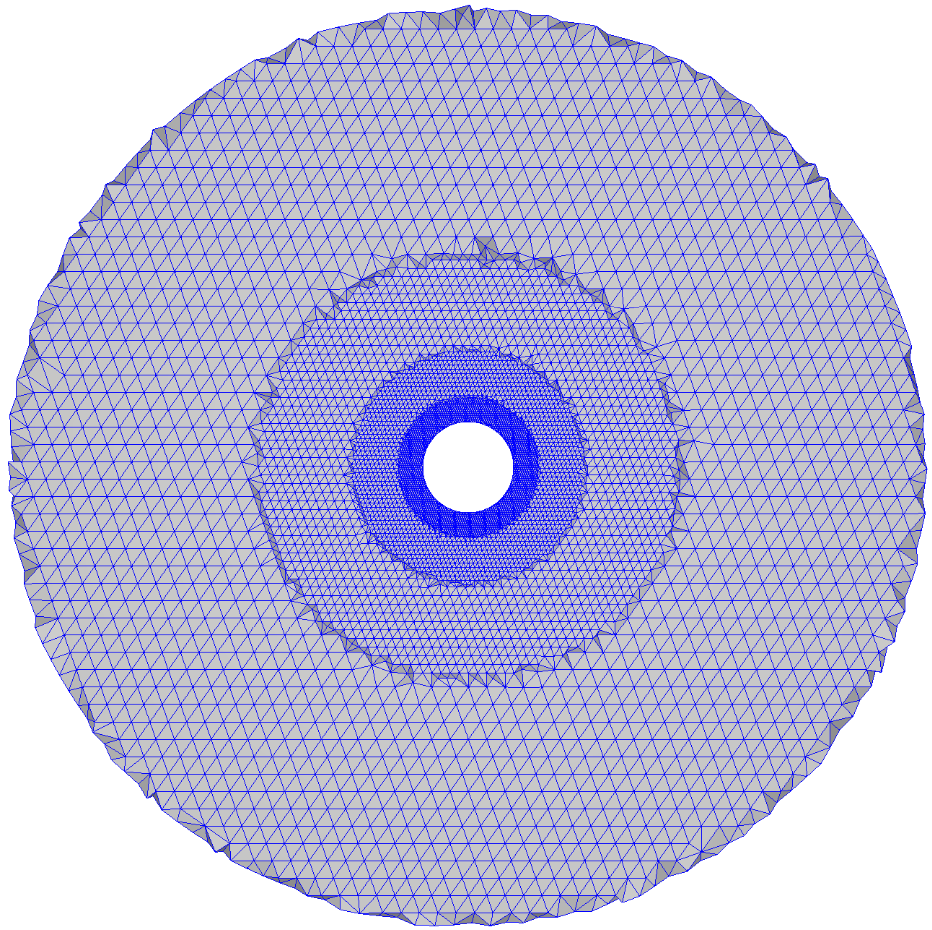




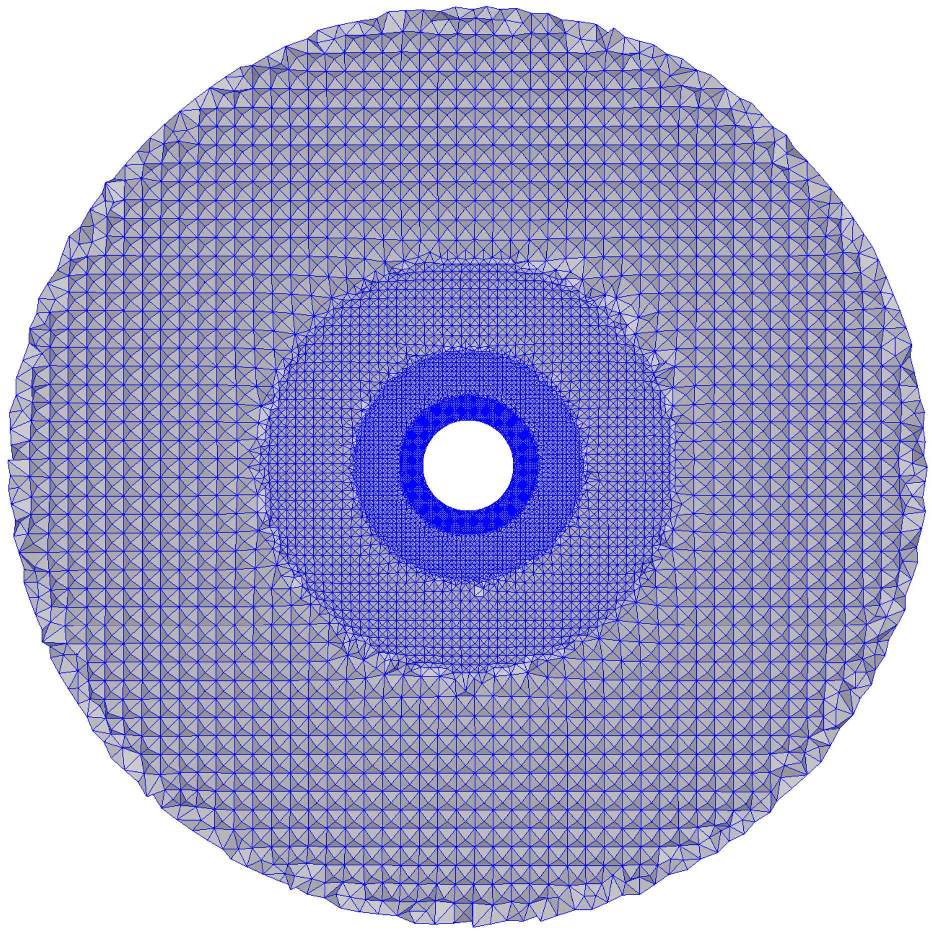
ncart=0



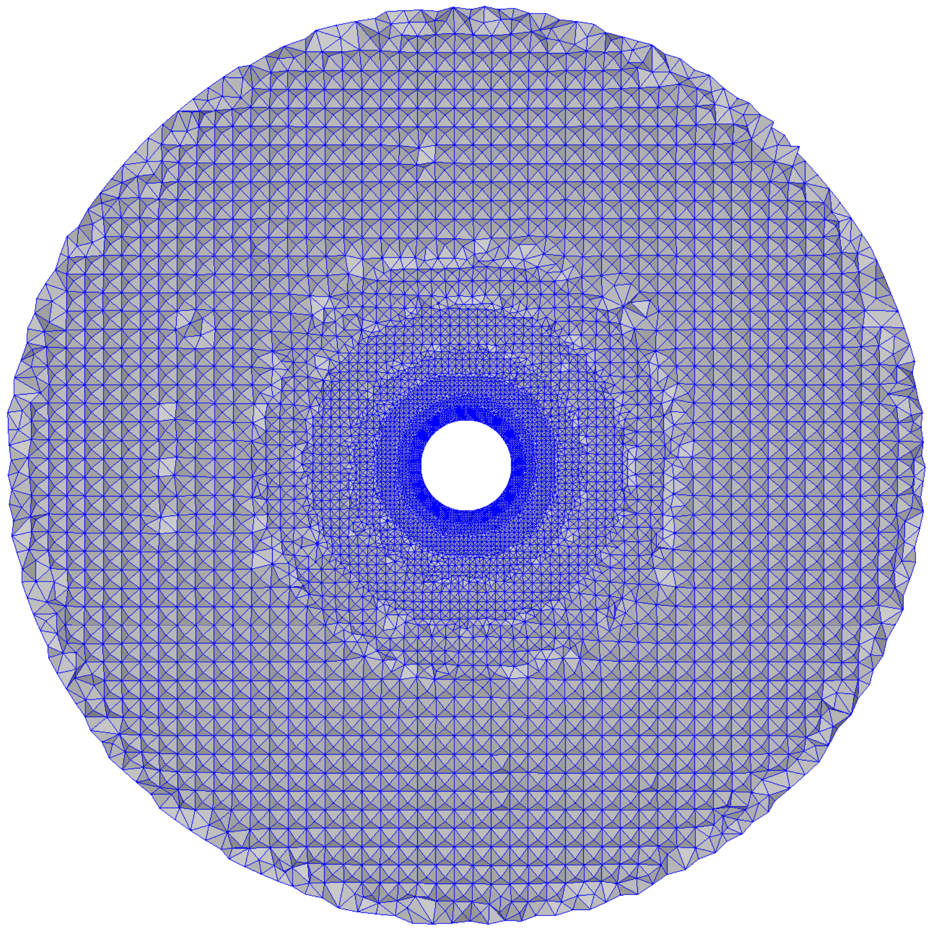
ncart=10



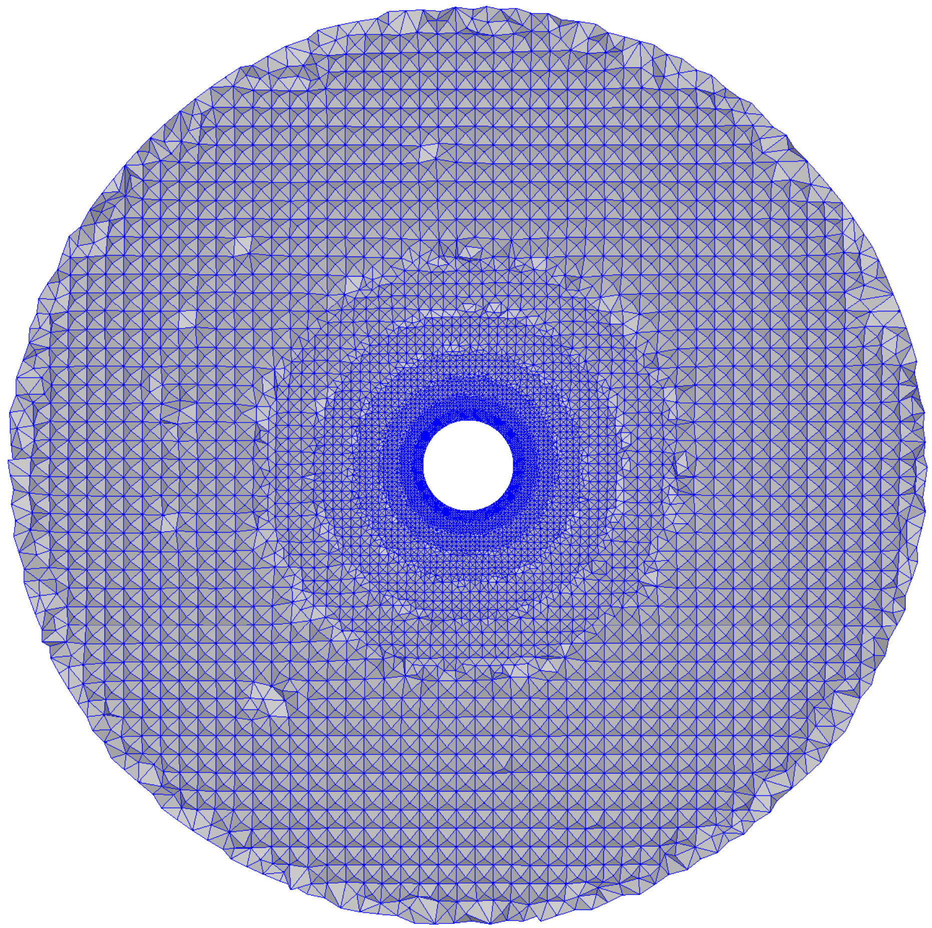
ncart=11



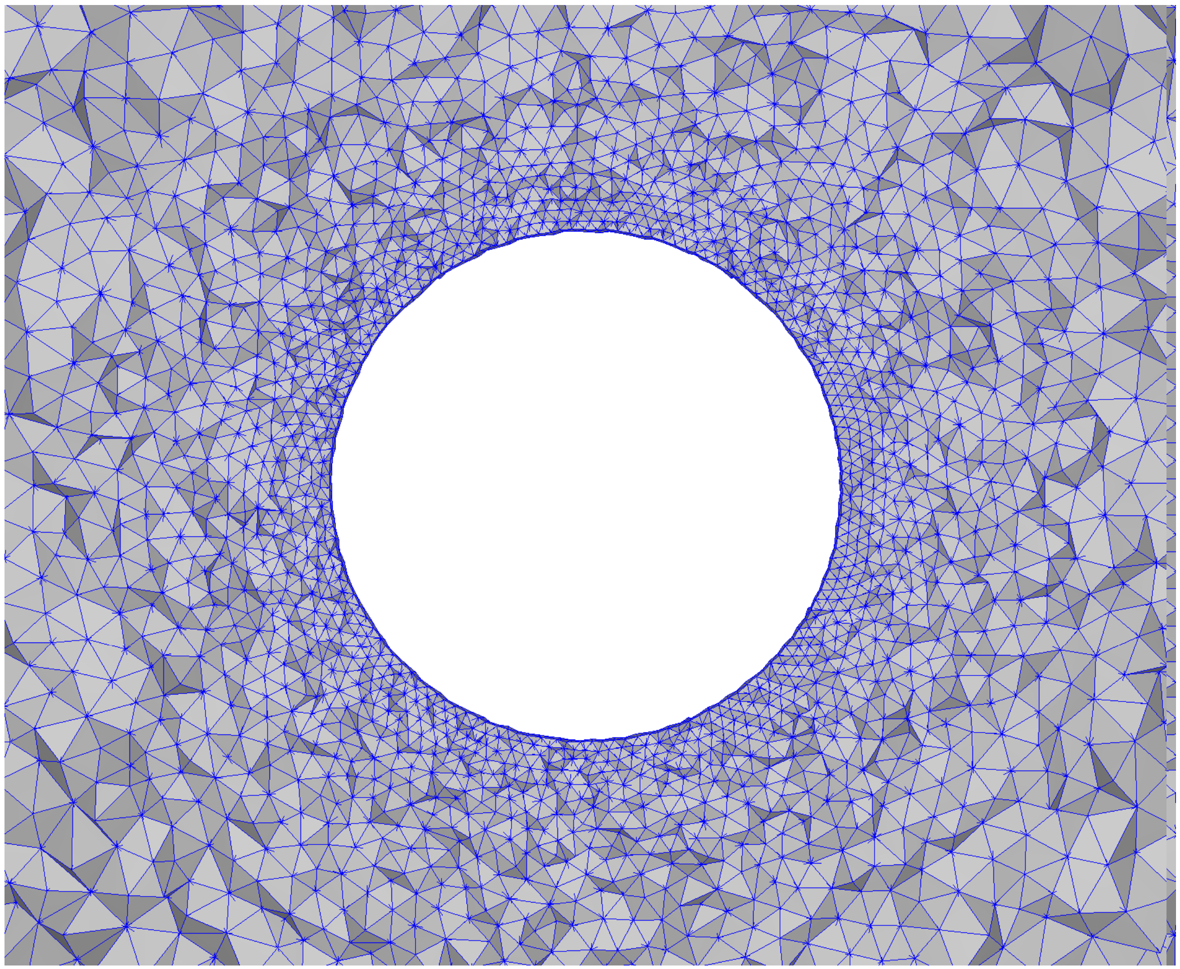
ncart=15



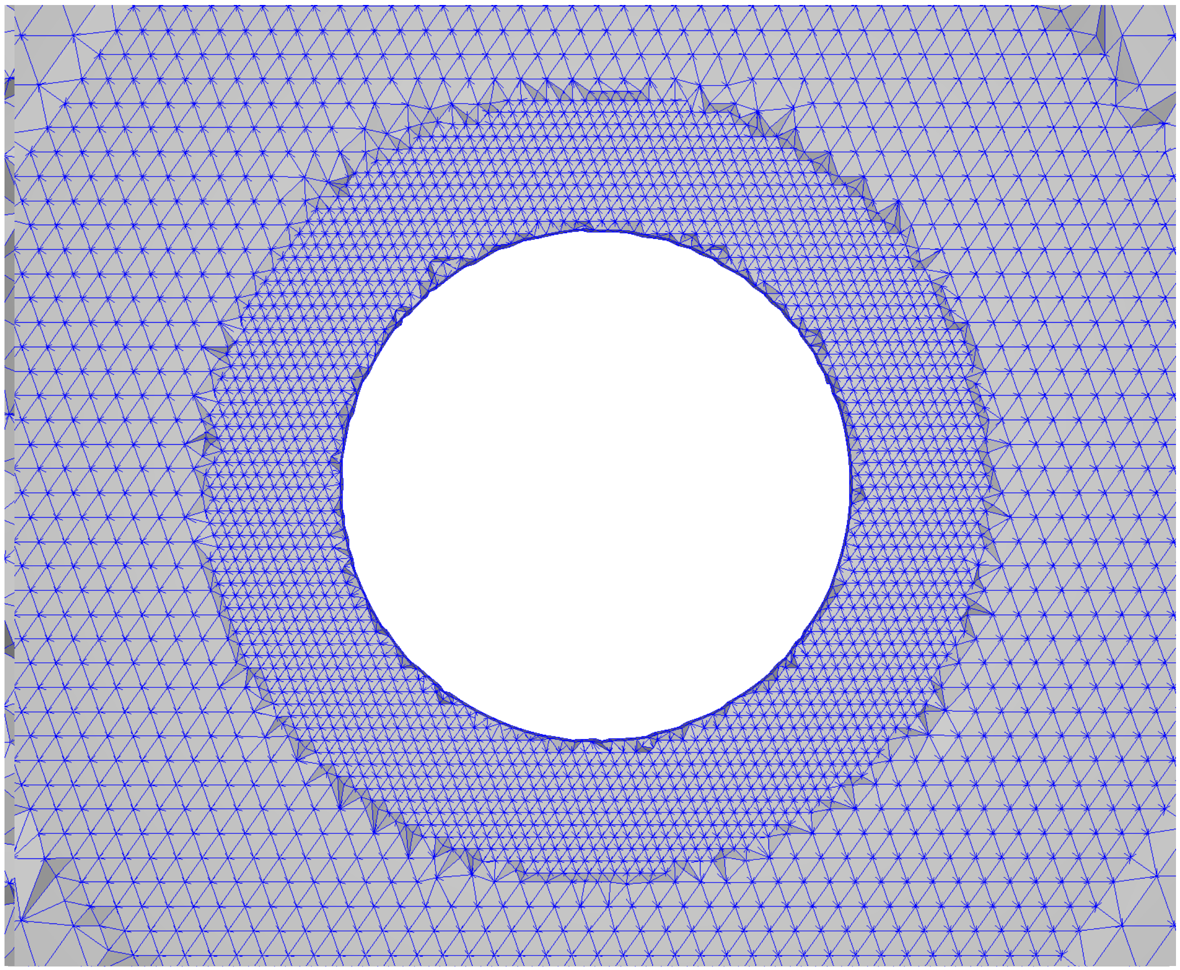
ncart=16



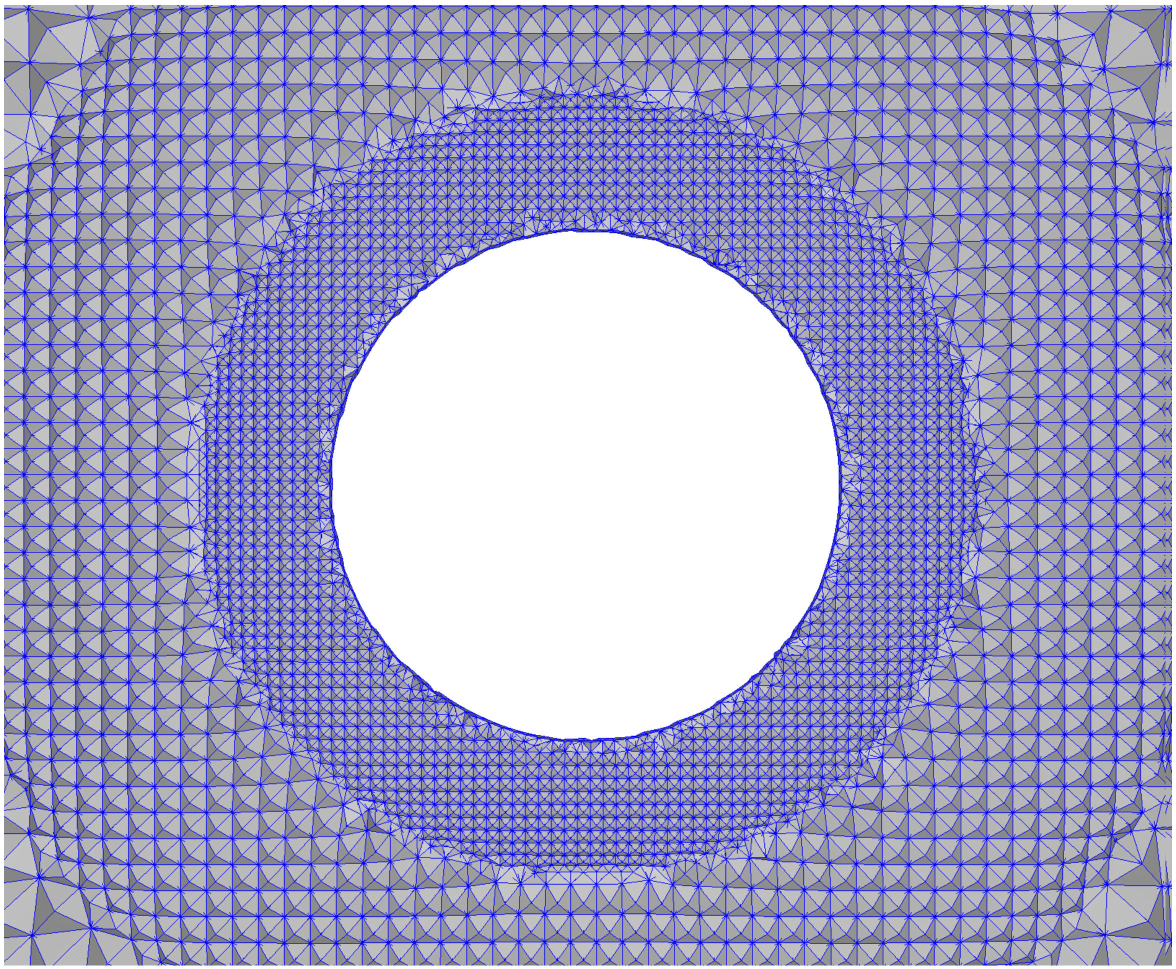
Closeup: ncart=0



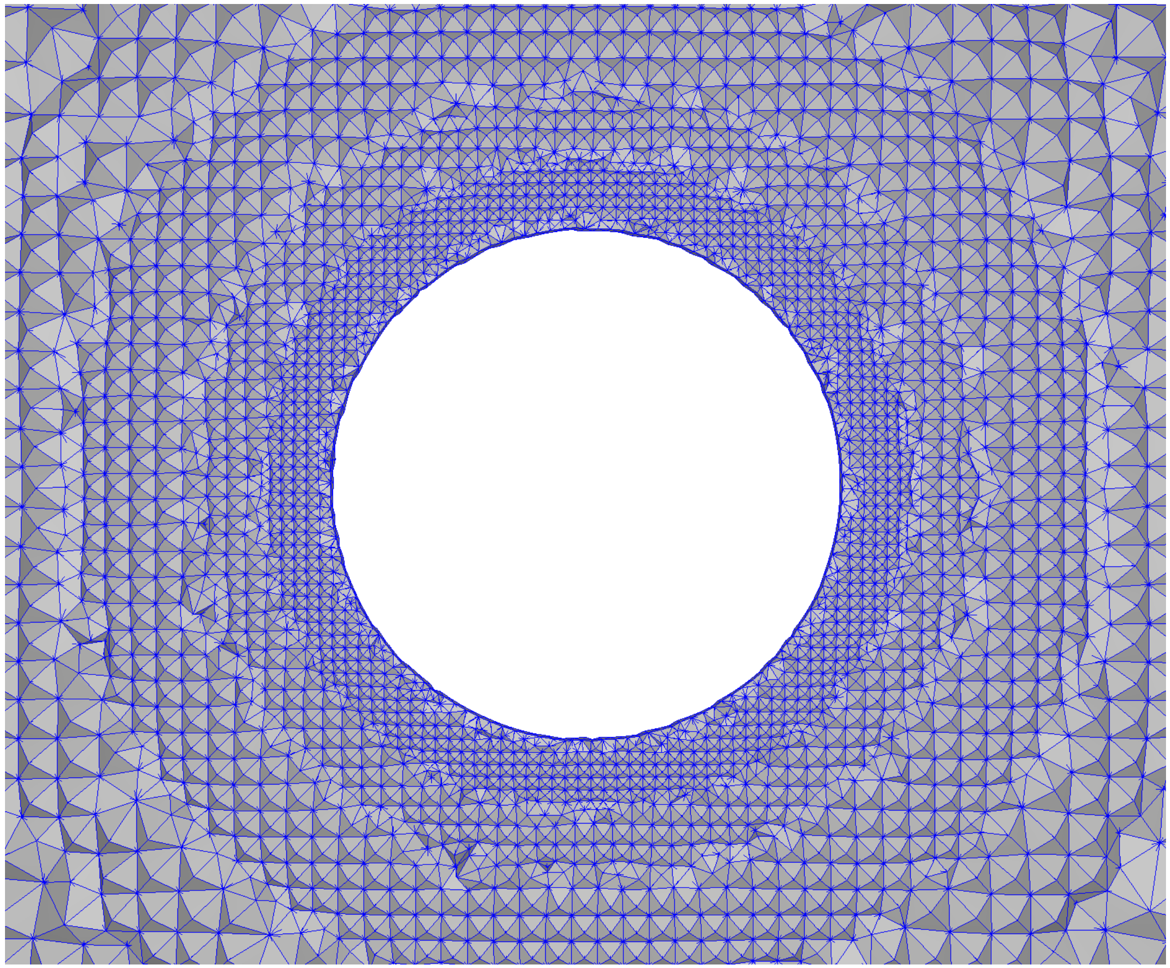
Closeup: ncart=10



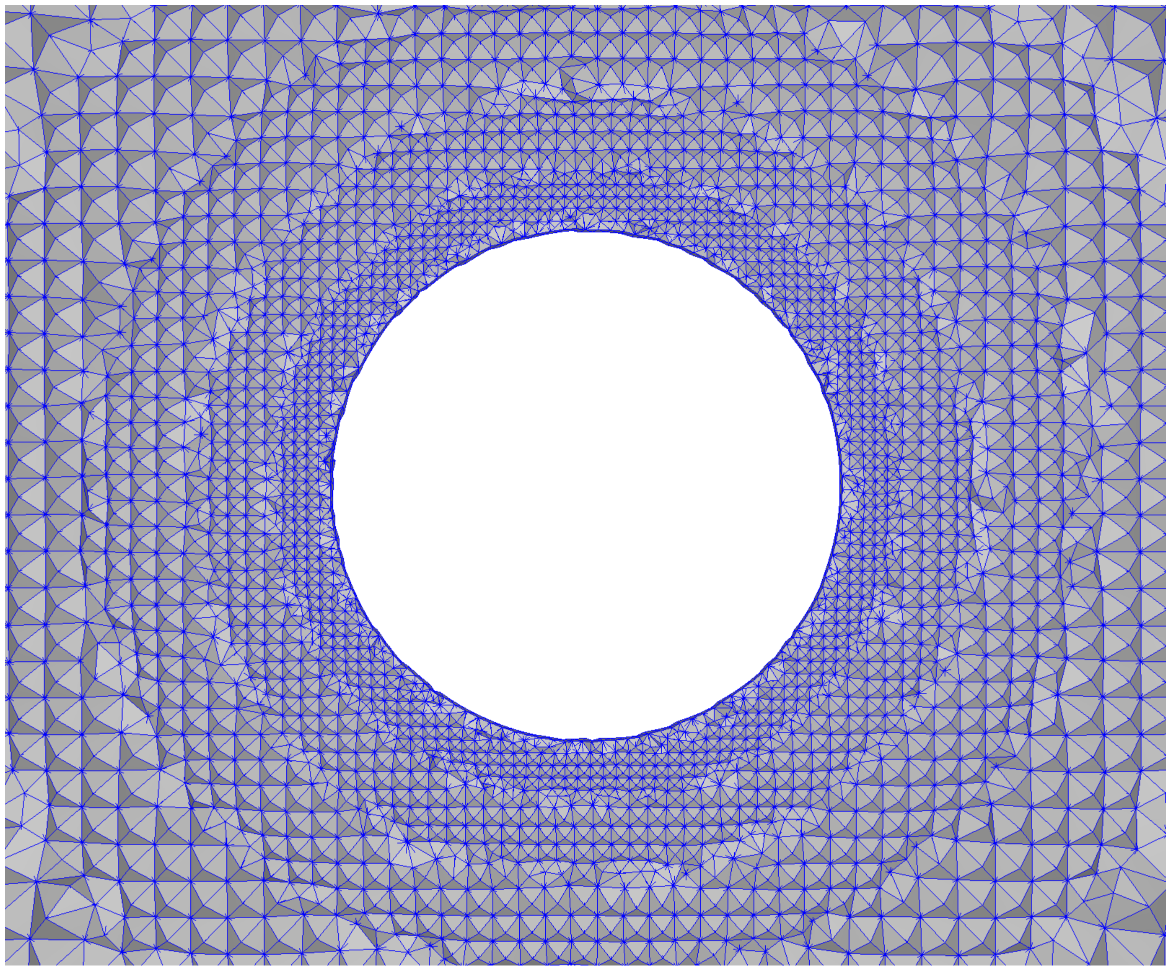
Closeup: ncart=11



Closeup: ncart=15

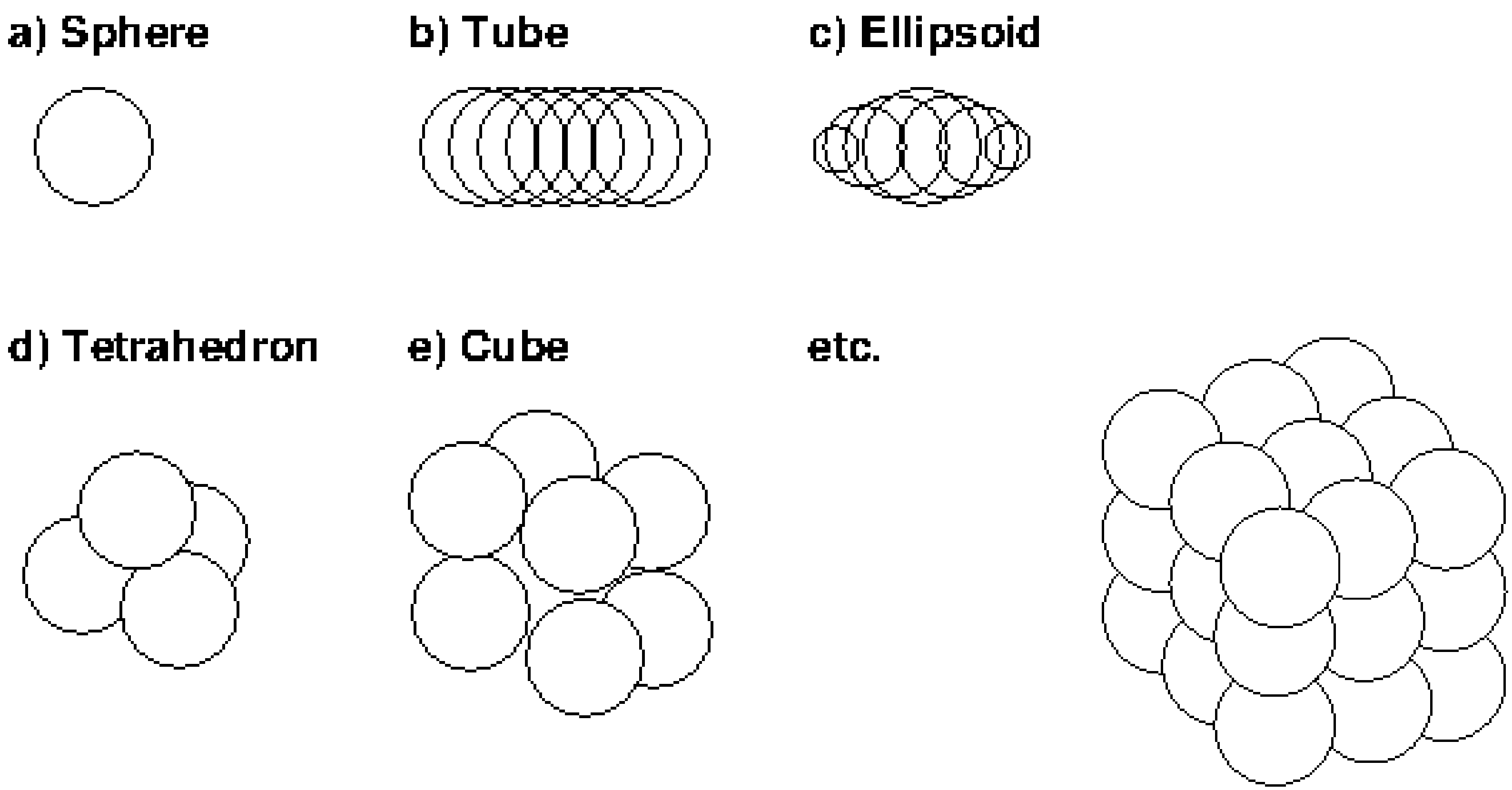


Closeup: ncart=16



Advancing Front Object Generation

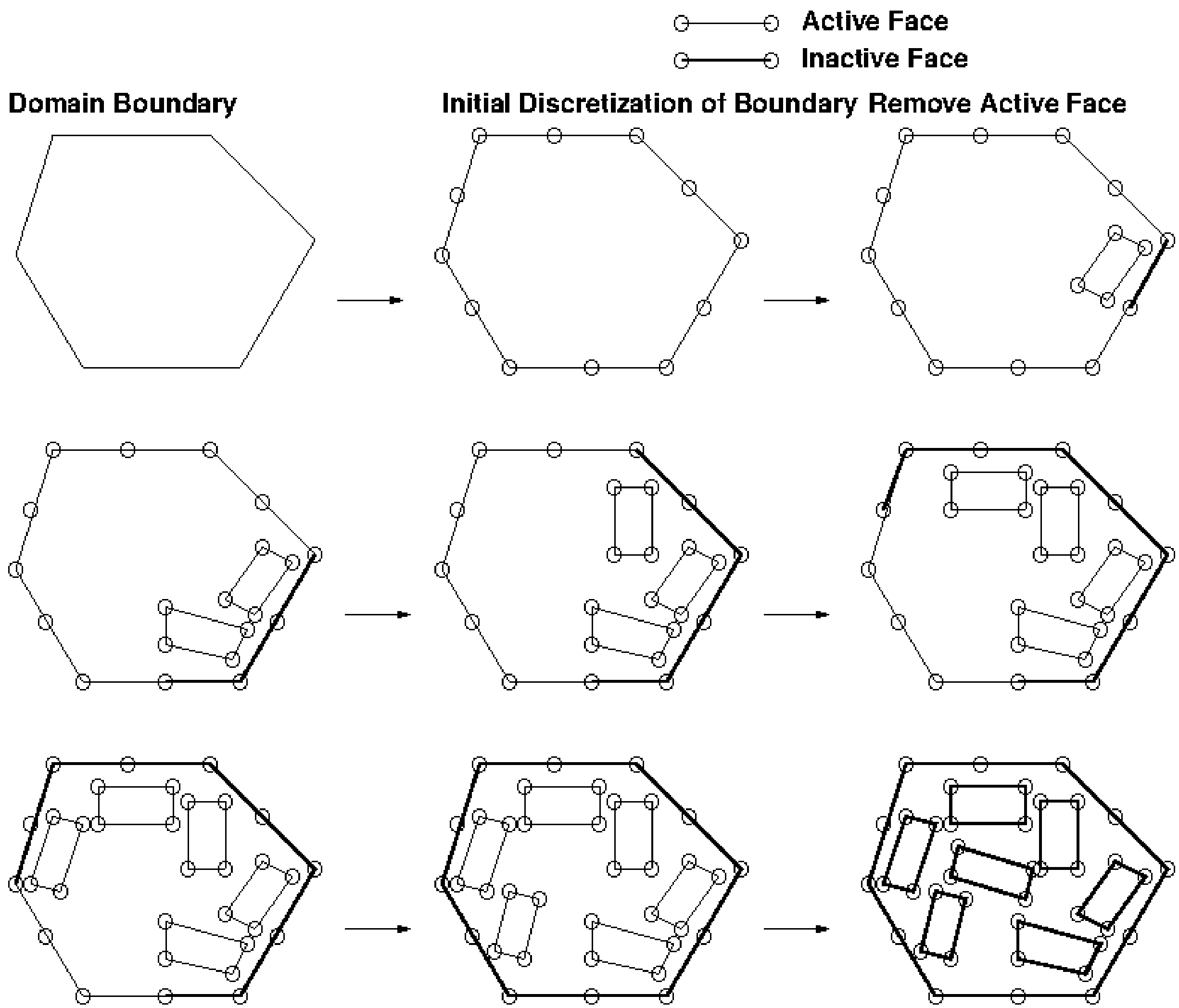
Option 1: Objects From Spheres



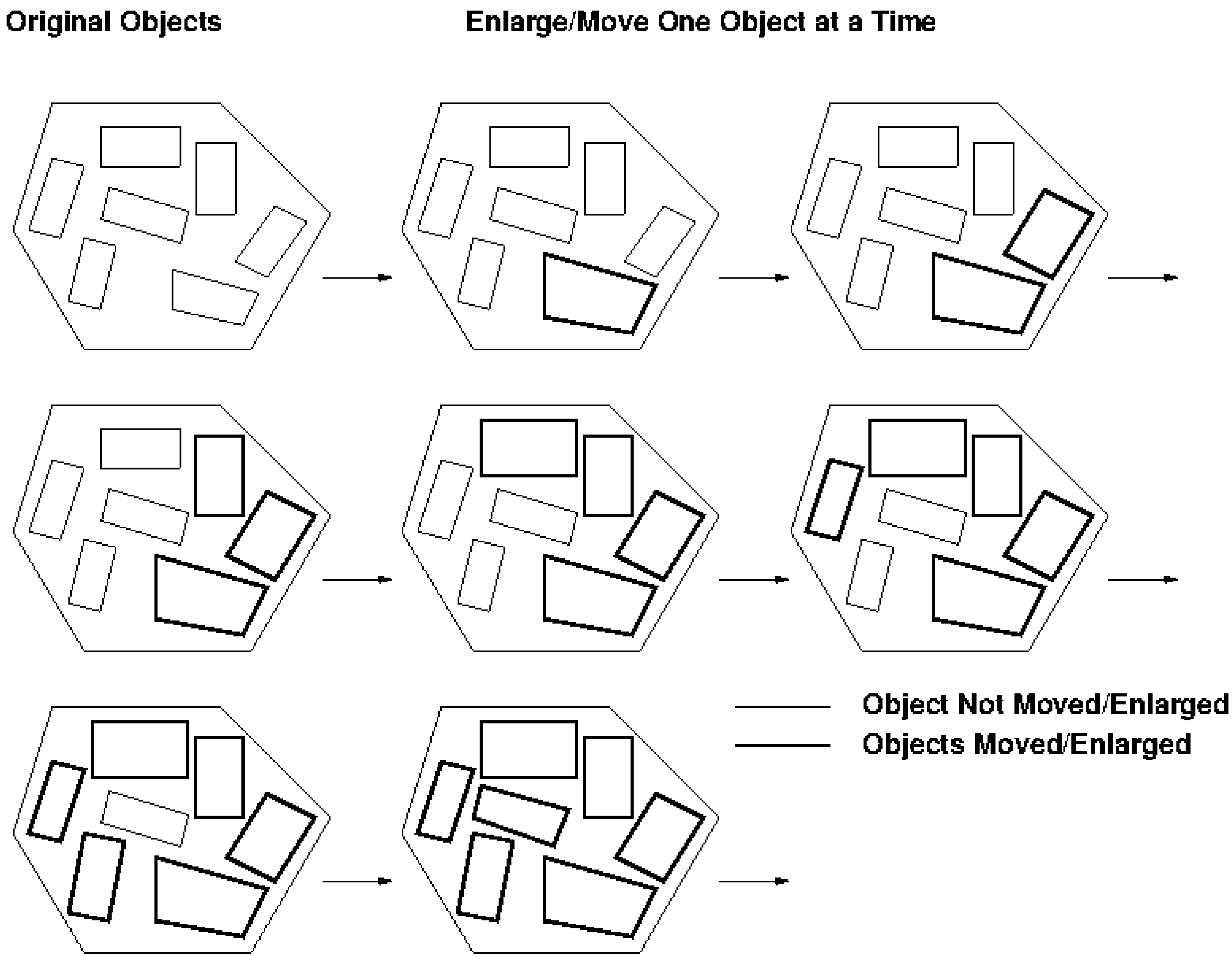
Option 2: Use Triangulated Objects

- Higher Degree of Accuracy / Realism
- Can Re-Use Large Portion of Advancing Front Tet-Meshing Techniques
  - Close Faces
  - Crossing Checks
  - …

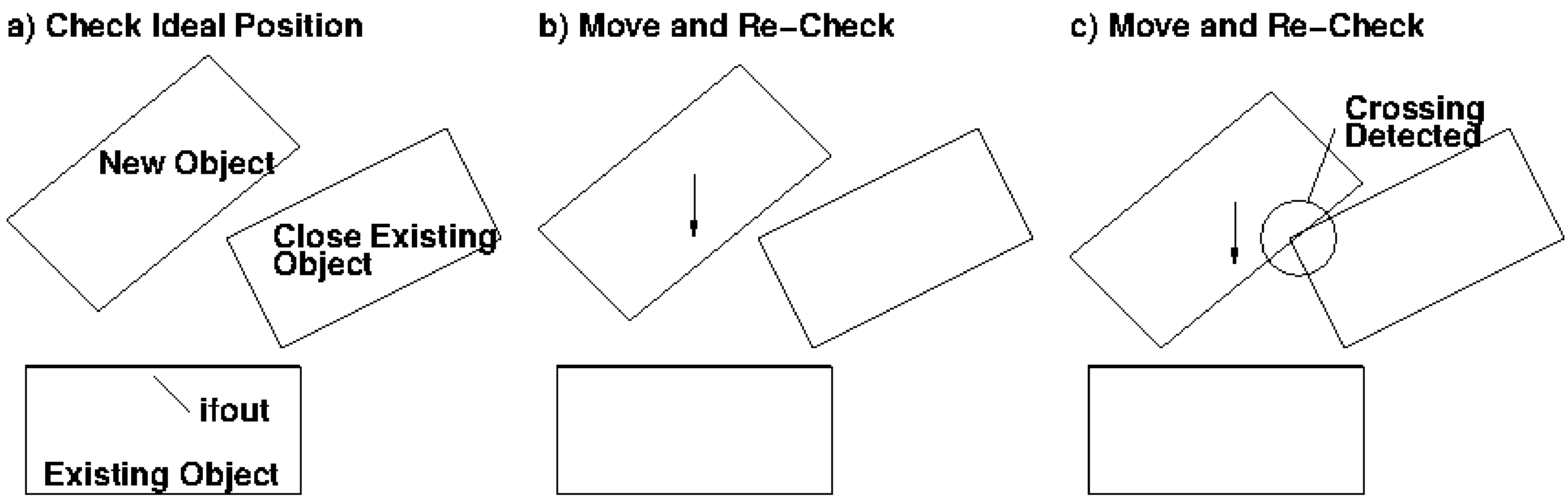
Advancing Front Object Generation

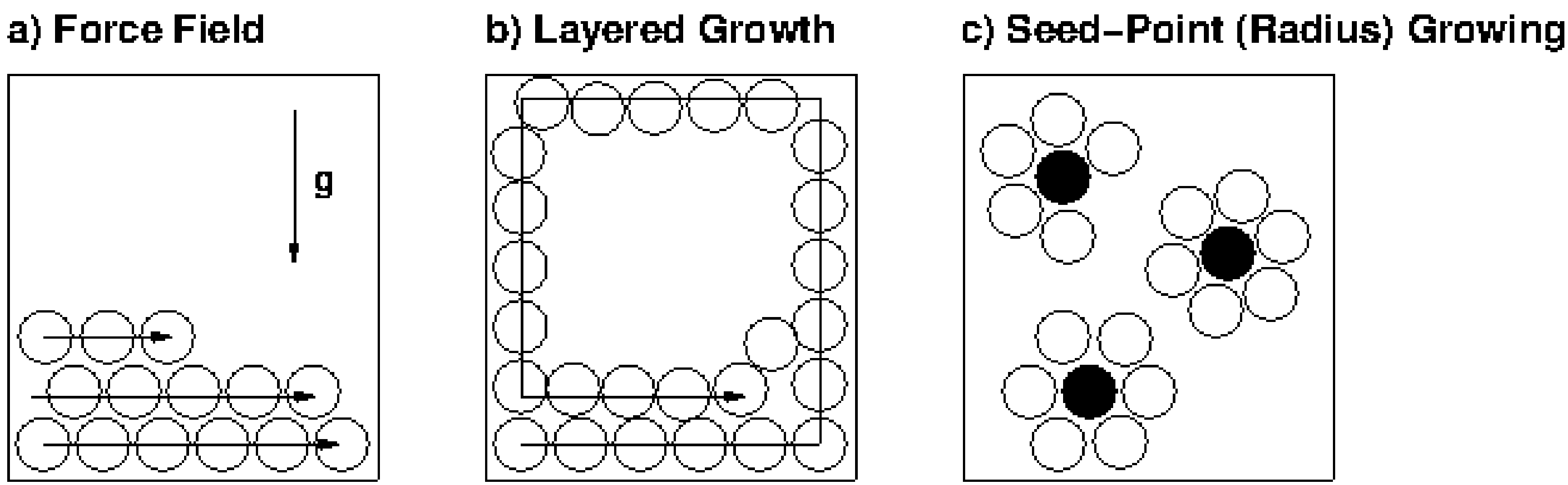


Move and Enlarge

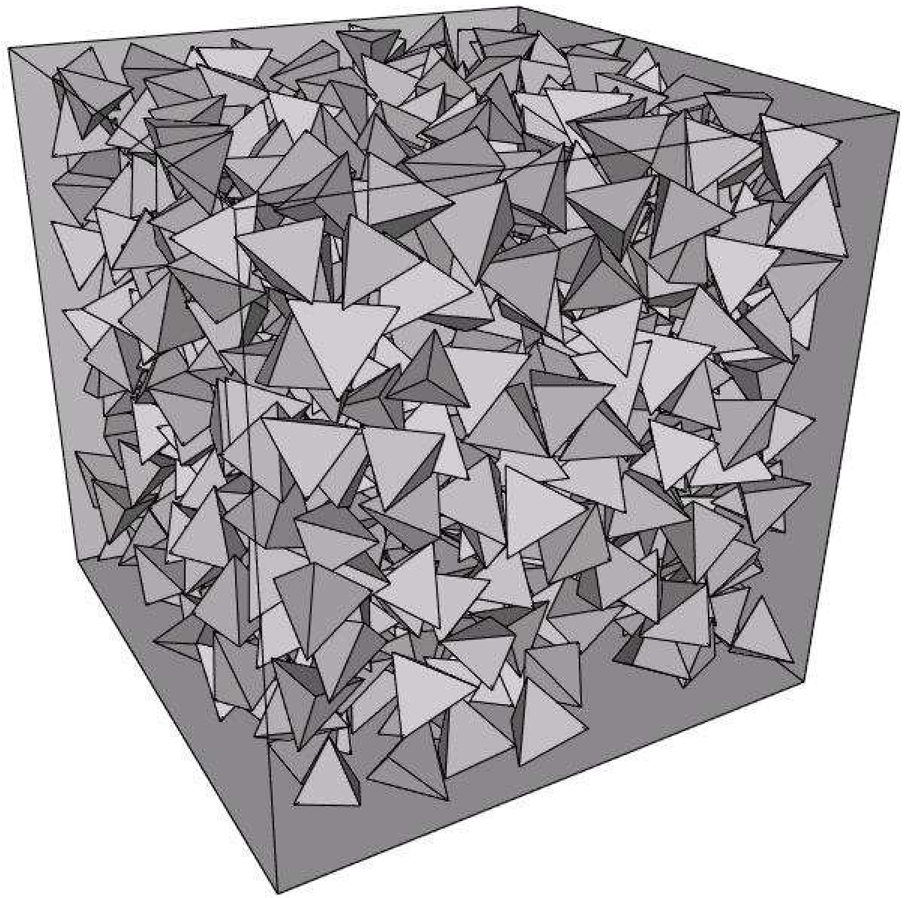


Move To Object

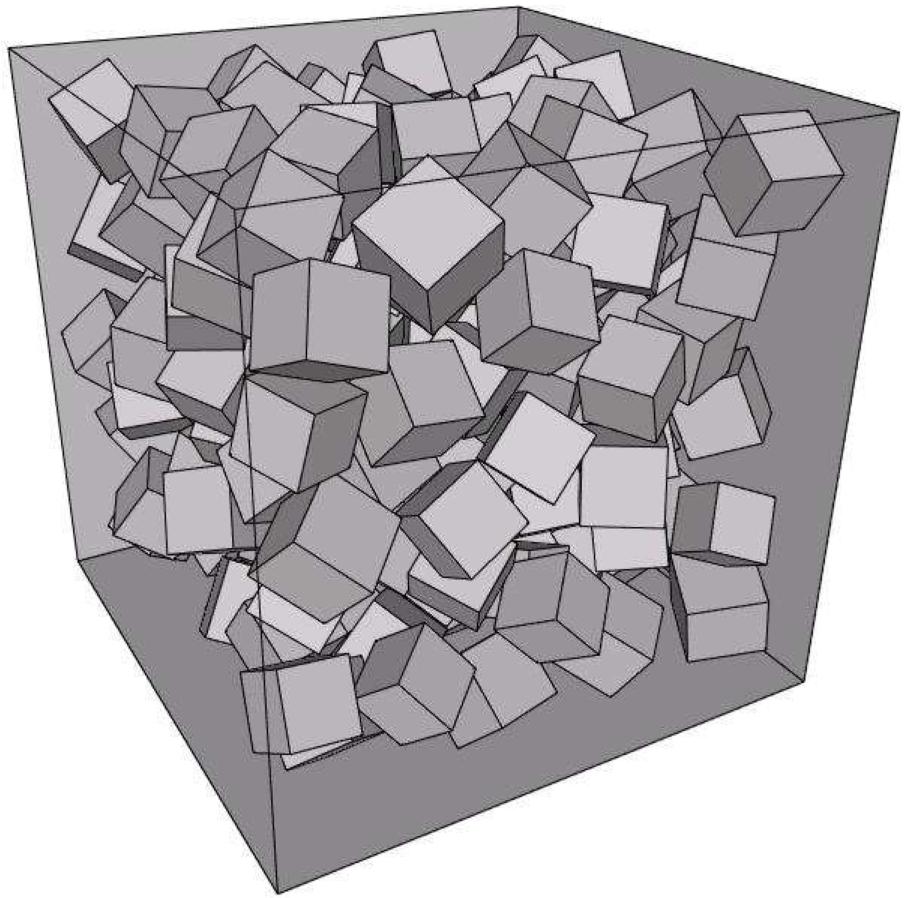



a) Force Field
g
b) Layered Growth
c) Seed-Point (Radius) Growing

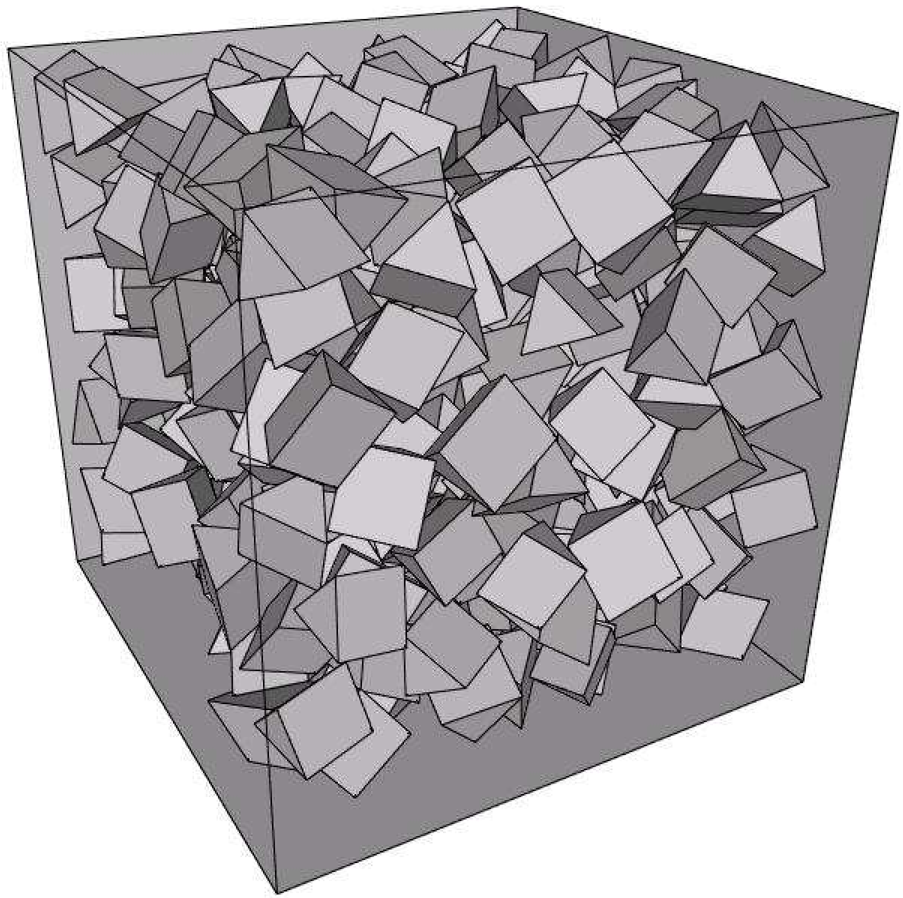
Tetrahedra of Different Sizes



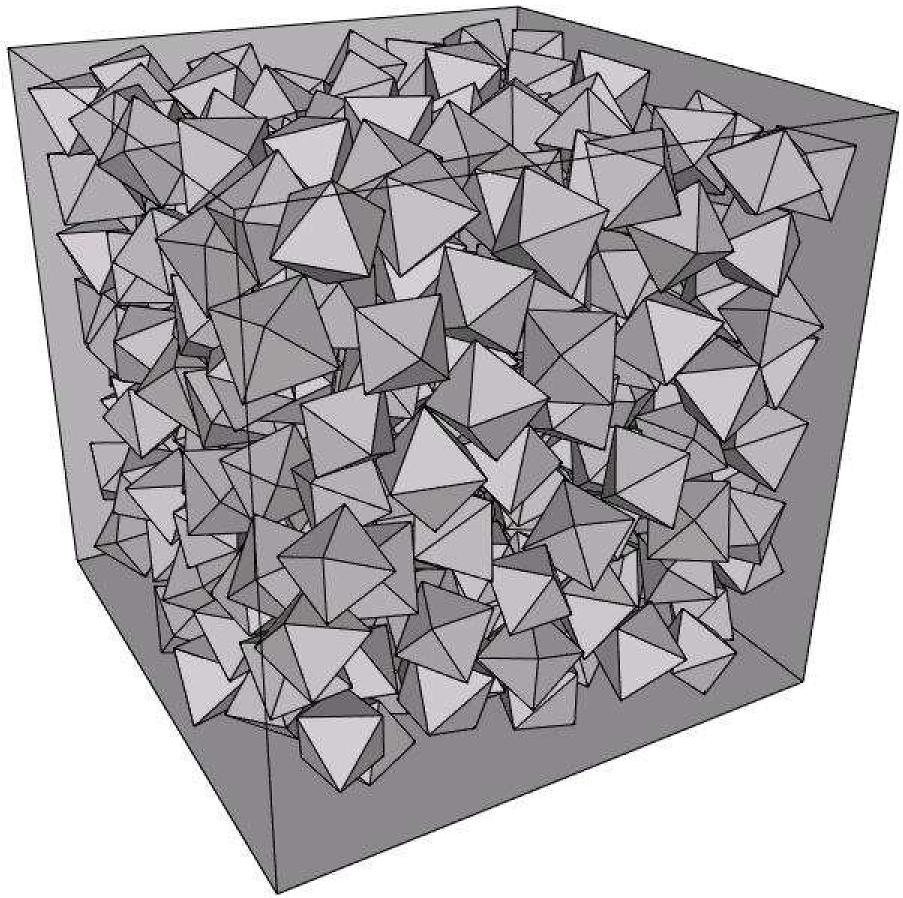
Cubes



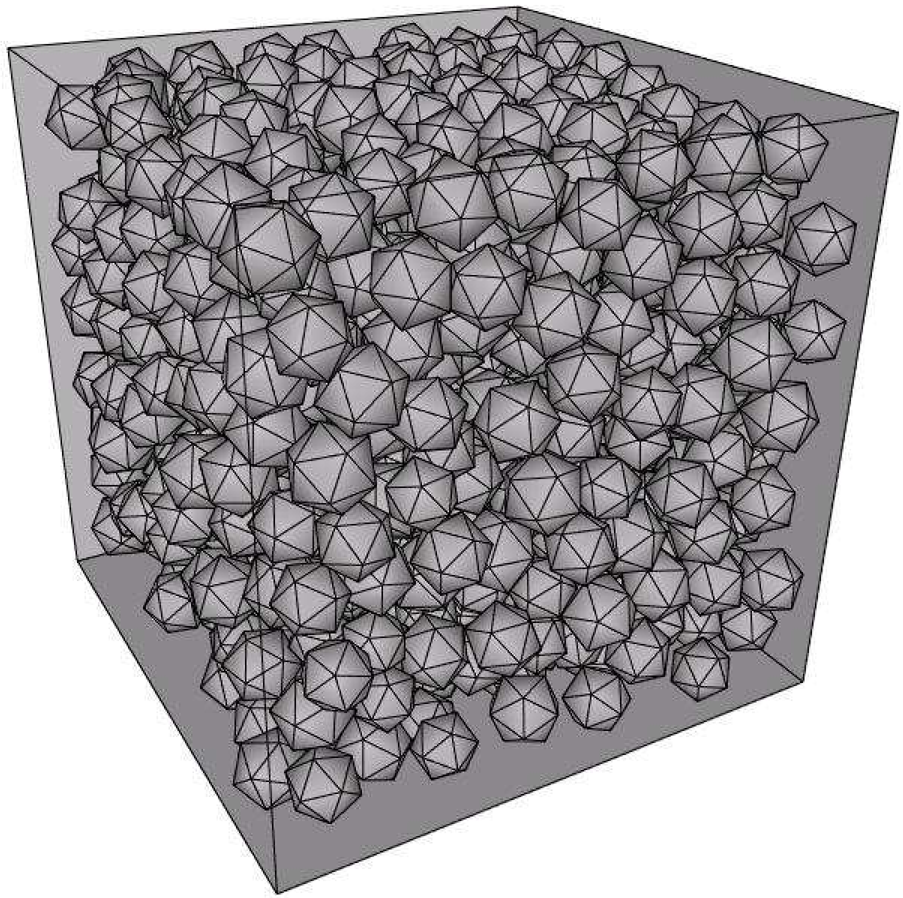
Prisms



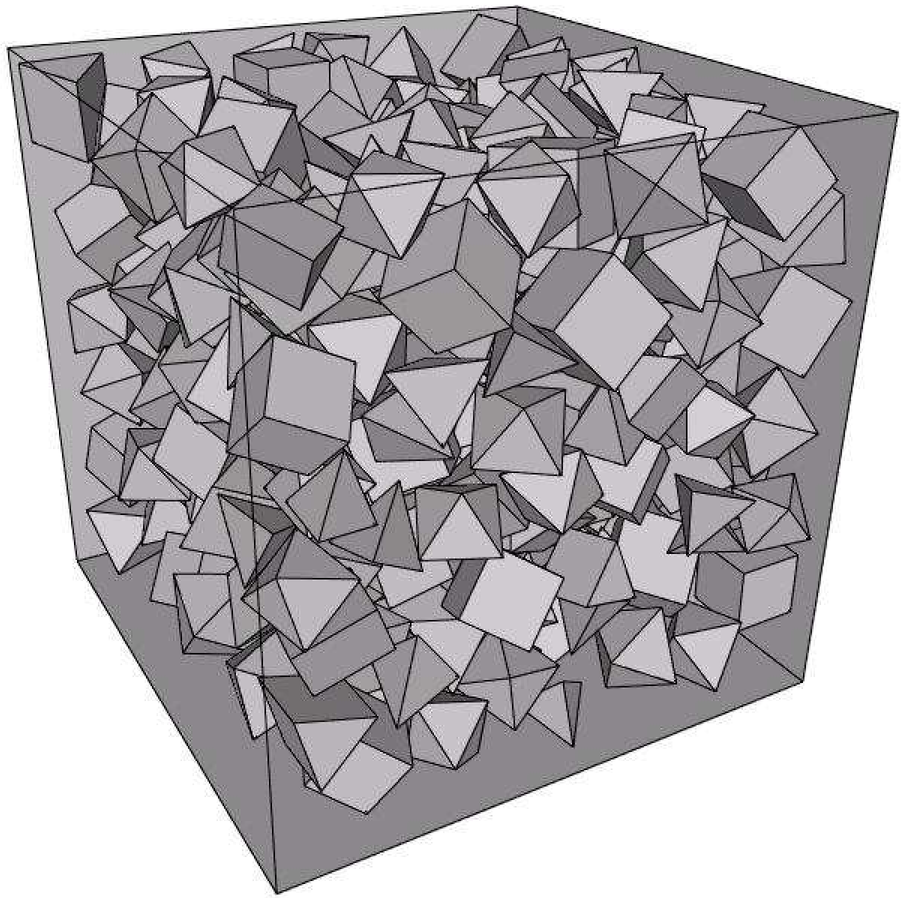
Octahedra



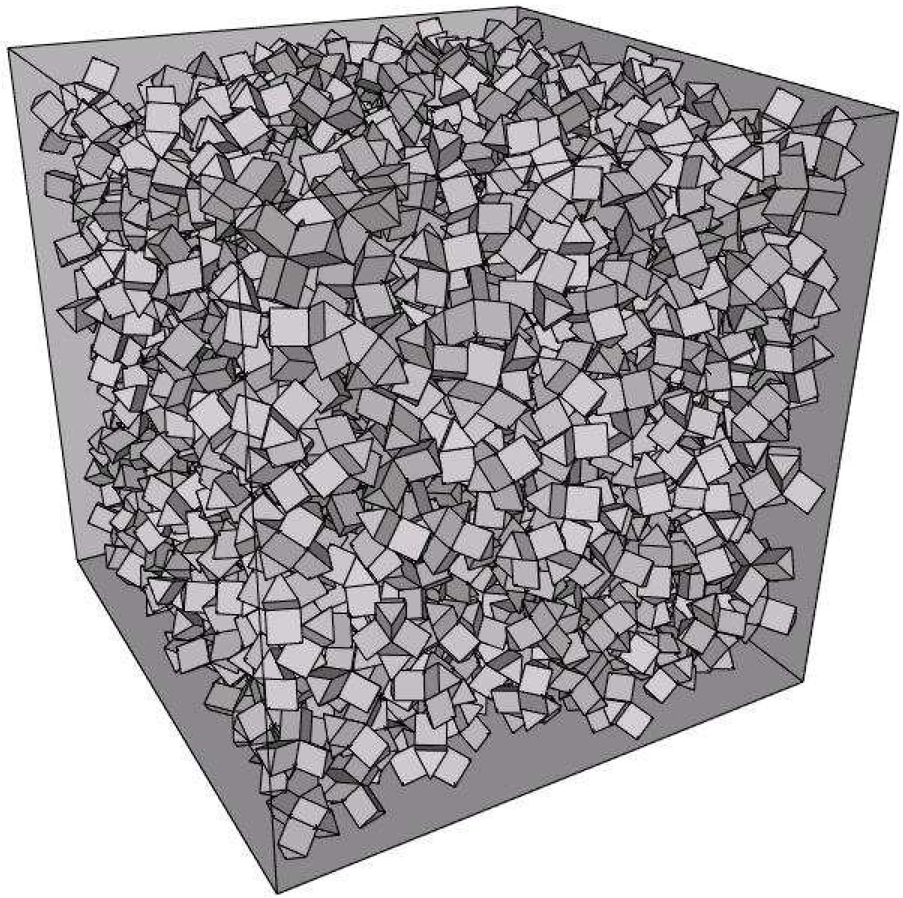
Icosahedra



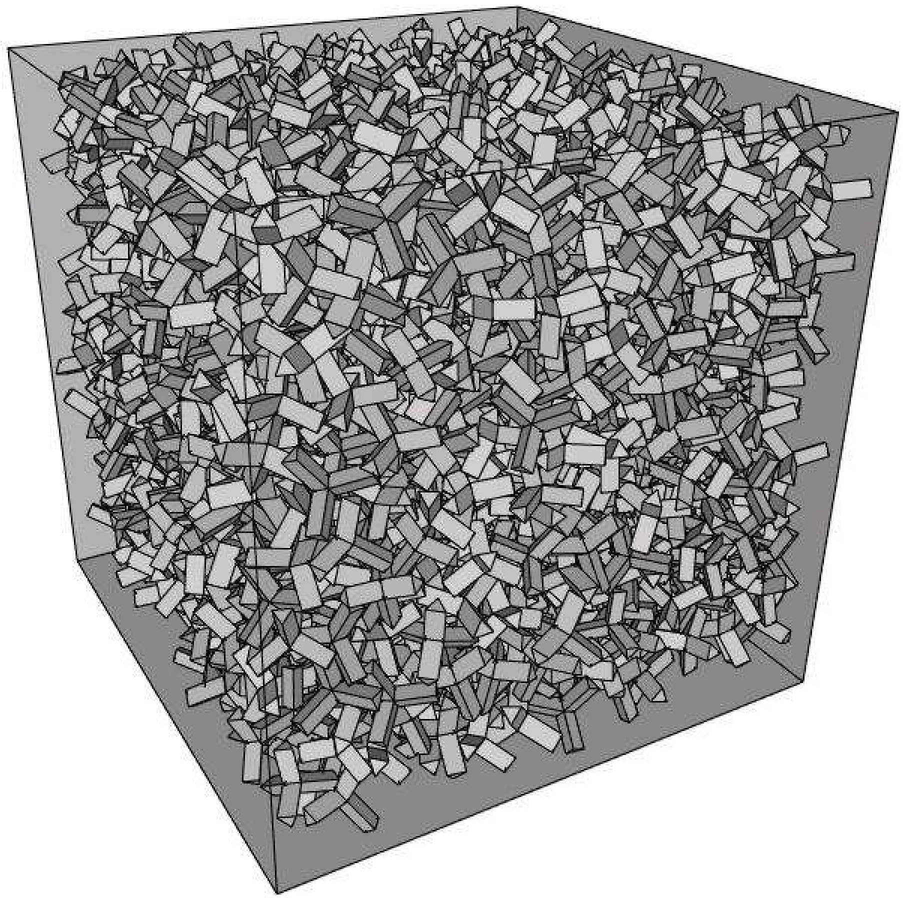
Mix



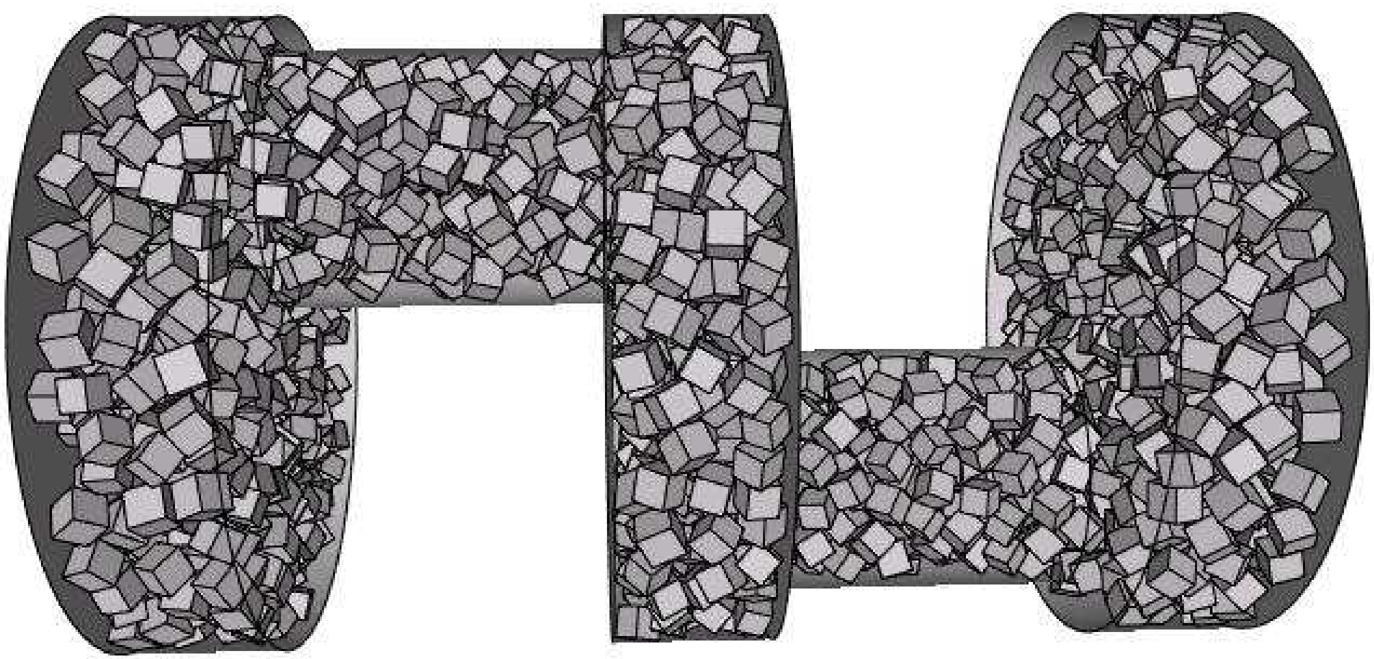
Short Tetrapods



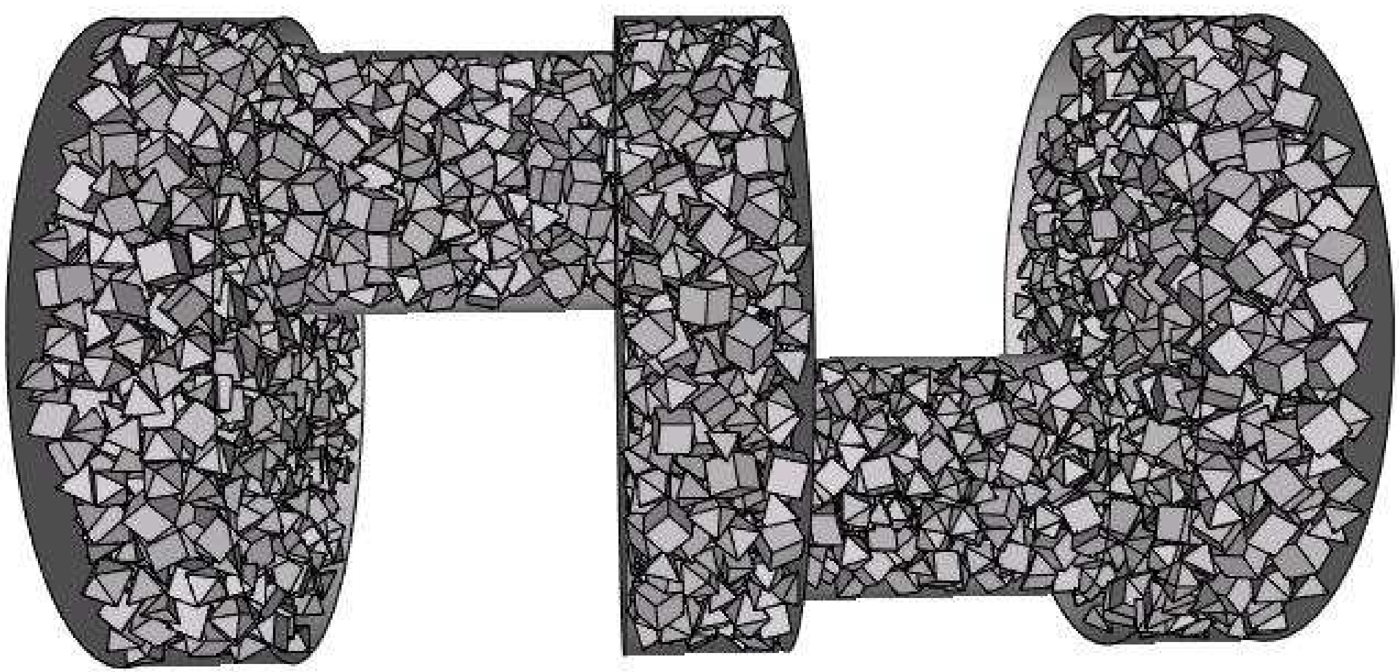
Long Tetrapods



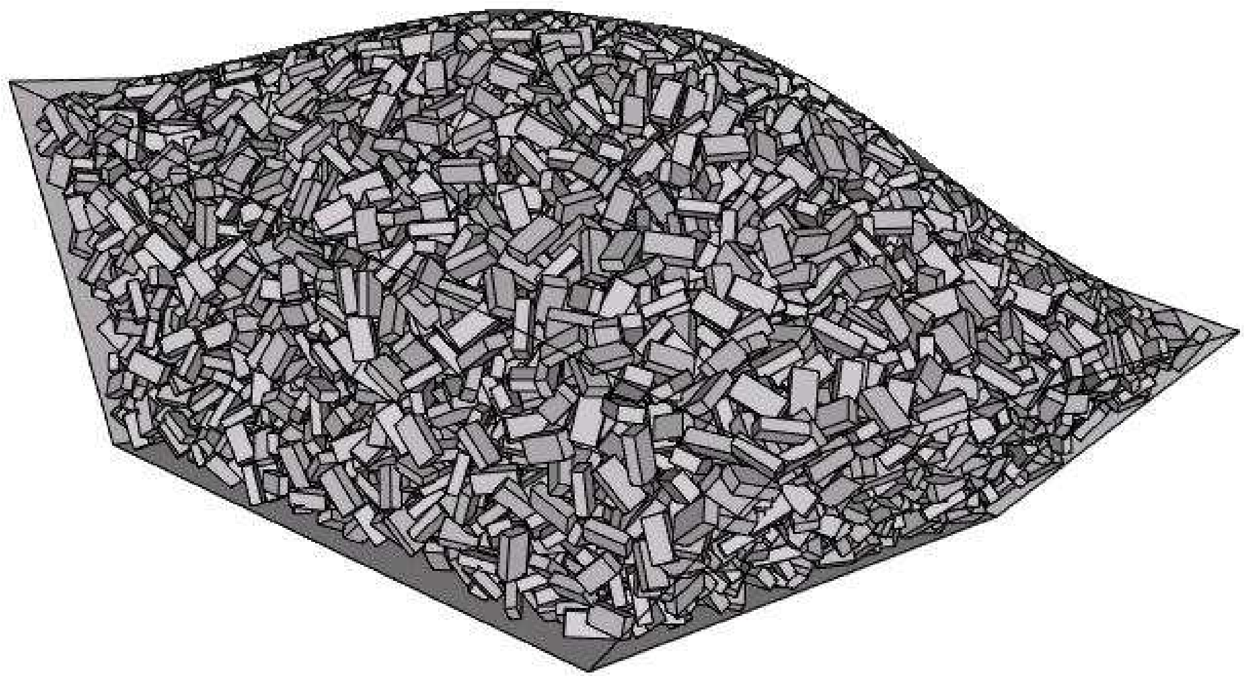
Mechanical Part, Cubes



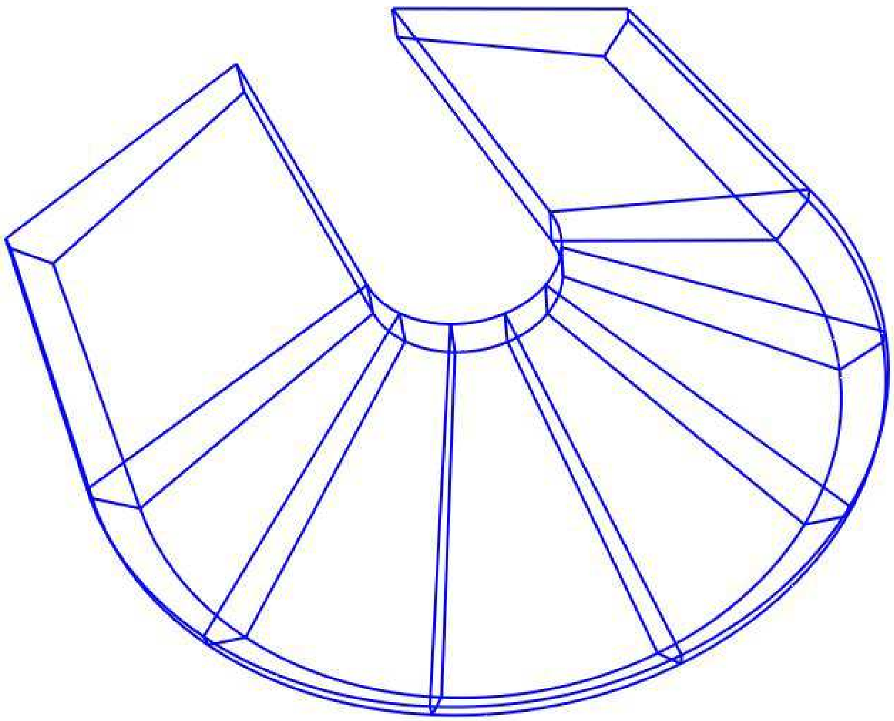
Mechanical Part, Mix

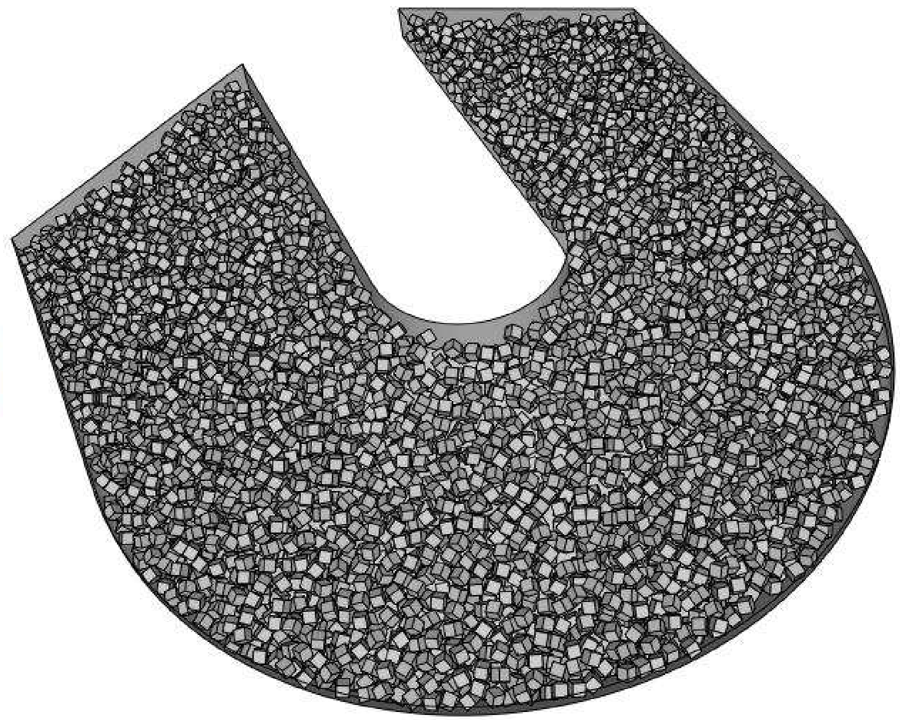


Truckload of Bricks

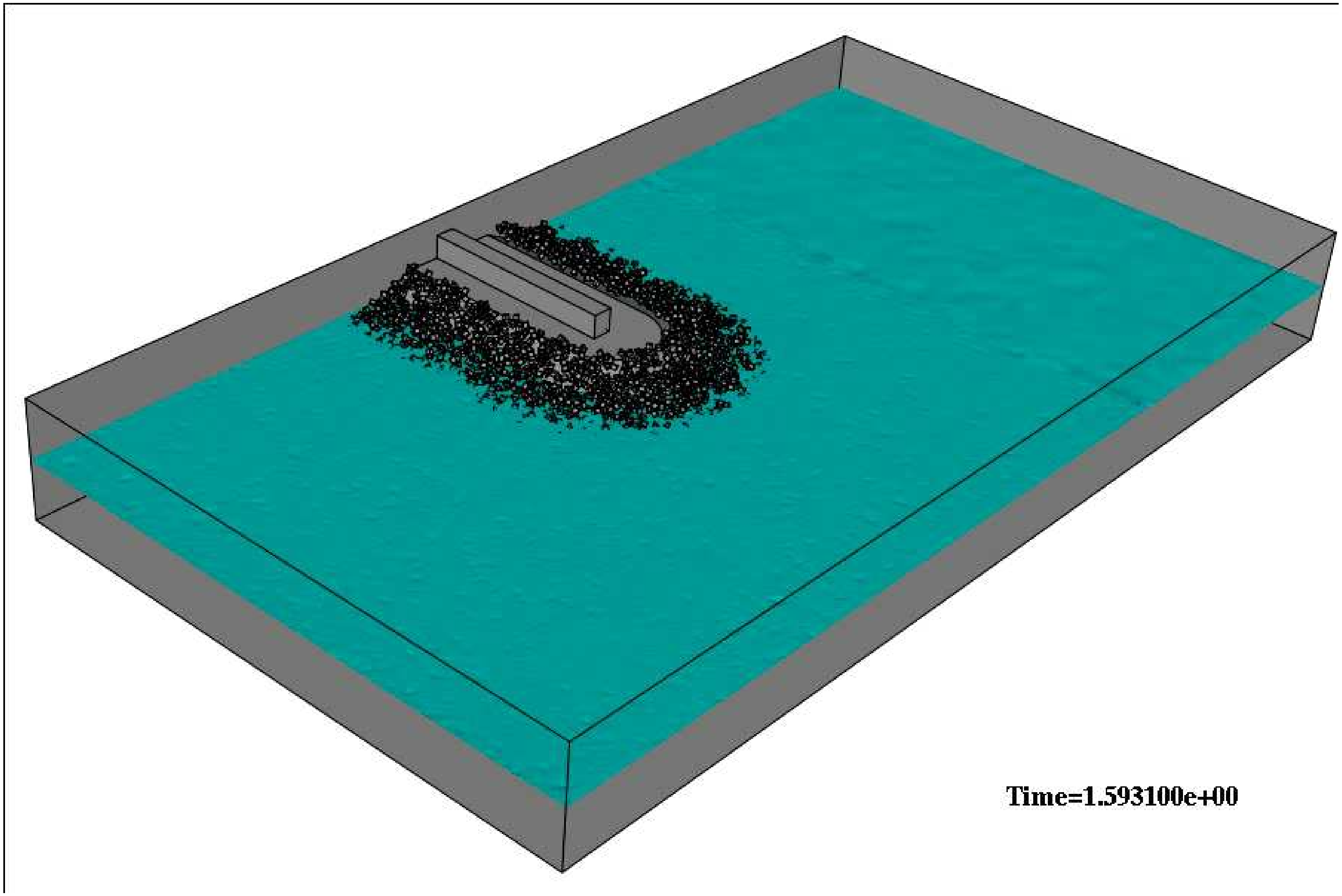


Breakwater





Breakwater



Conclusions and Outlook (1)

- CFD Crisis:
  - Successful Parallelization in Space
  - No Parallelization in Time (Physics ?)
  - LES: Most Points in Isotropic Element Regions
- Advancing Front:
  - Attempts to Improve Scalar Speed
- Parallel (Distributed, MPI) GridGen Working
  - Key Ideas: Domain Defining Grid, Parallel Advancing Front, …
  - Good Scalability Once Mesh Size > 1 Mel/Core

Conclusions and Outlook (2)

- Parallel Domain Decomposition Working
  - Key Ideas: Bounding Boxes, Octrees, Additional Layers, …
- Current Efforts
  - Multimaterial
  - Link to BL Modules
  - Faster Generation of DDGs
  - Smaller DDGs (Planar Surfaces)

Conclusions and Outlook (3)

- Optimal Space-Filling Tets
  - Exploit in Solver
- Arbitrary Object Generation
  - Working
  - Any Mix of Objects Possible
  - Interesting for Basic Physics Studies