A Strategy for Optimal Solvers in PyLith

Matthew Knepley[1] Peter Brune[2]

[1] Mathematics and Computer Science Division
Argonne National Laboratory

[2] Computer Science Department
University of Chicago

AMR Tutorial Workshop
Boulder, CO
Oct 24–27, 2007

Questions

"A very popular error – having the courage of one's convictions: Rather it is a matter of having the courage for an attack upon one's convictions."
— Friedrich Wilhelm Nietzsche

Outline

Why Optimal Algorithms?

- The more powerful the computer,
  the greater the importance of optimality
- Example:
  - Suppose Alg_1 solves a problem in time CN^2, where N is the input size
  - Suppose Alg_2 solves the same problem in time CN
  - Suppose Alg_1 and Alg_2 are able to use 10,000 processors
- In constant time compared to serial,
  - Alg1 can run a problem 100X larger
  - Alg2 can run a problem 10,000X larger
- Alternatively, filling the machine's memory,
  - Alg1 requires 100X time
  - Alg2 runs in constant time

Linear Convergence

Convergence to $||r|| < 10^{-9}||b||$ using GMRES(30)/ILU

Elements	Iterations
128	10
256	17
512	24
1024	34
2048	67
4096	116
8192	167
16384	329
32768	558
65536	920
131072	1730

Linear Convergence

Convergence to $||r|| < 10^{-9}||b||$ using GMRES(30)/MG

Elements	Iterations
128	5
256	7
512	6
1024	7
2048	6
4096	7
8192	6
16384	7
32768	6
65536	7
131072	6

Outline

AMG

Why not use AMG?

AMG

Why not use AMG?

- Of course we will try AMG

AMG

Why not use AMG?

- Of course we will try AMG
  - BoomerAMG, ML, SAMG, ASA

AMG

Why not use AMG?

- Of course we will try AMG
  - BoomerAMG, ML, SAMG, ASA
- Problems with vector character

AMG

Why not use AMG?

- Of course we will try AMG
  - BoomerAMG, ML, SAMG, ASA
- Problems with vector character
- Geometric aspects to the problem

AMG

Why not use AMG?

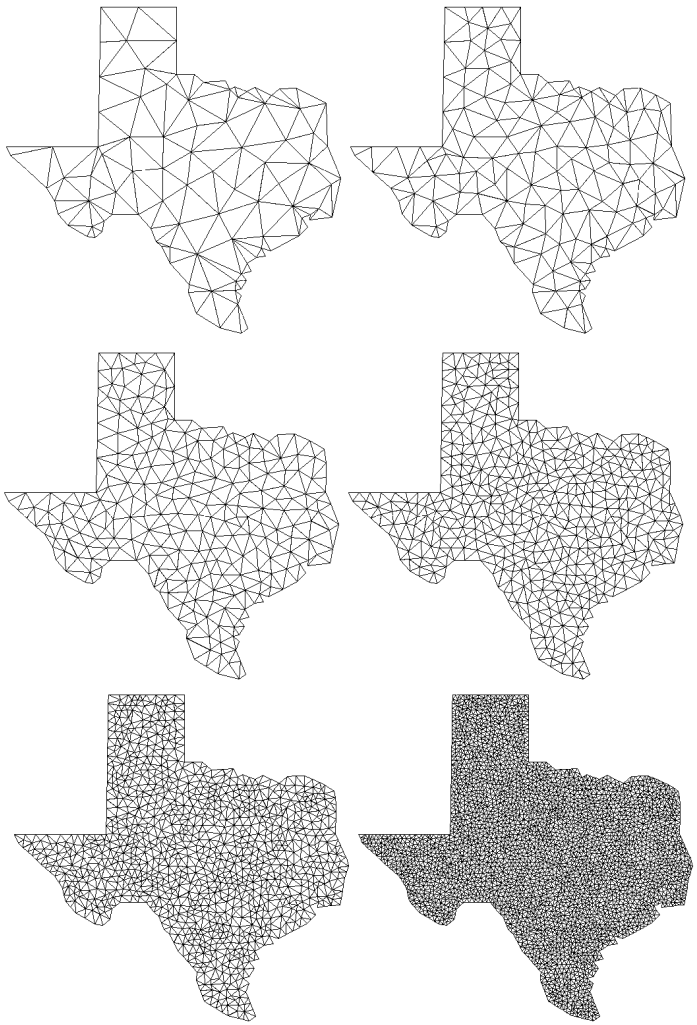
- Of course we will try AMG
  - BoomerAMG, ML, SAMG, ASA
- Problems with vector character
- Geometric aspects to the problem
  - Material property variation

AMG

Why not use AMG?

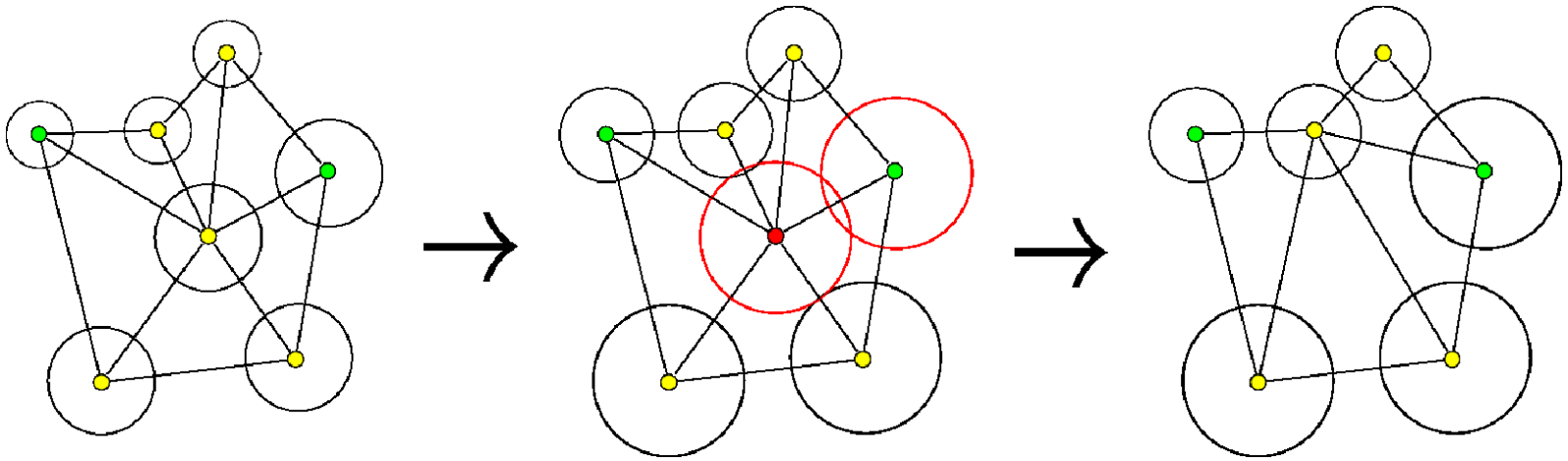
- Of course we will try AMG
  - BoomerAMG, ML, SAMG, ASA
- Problems with vector character
- Geometric aspects to the problem
  - Material property variation
  - Faults

Coarsening



- Users want to control the mesh
- Developed efficient, topological coarsening
  - Miller, Talmor, Teng algorithm
- Provably well-shaped hierarchy

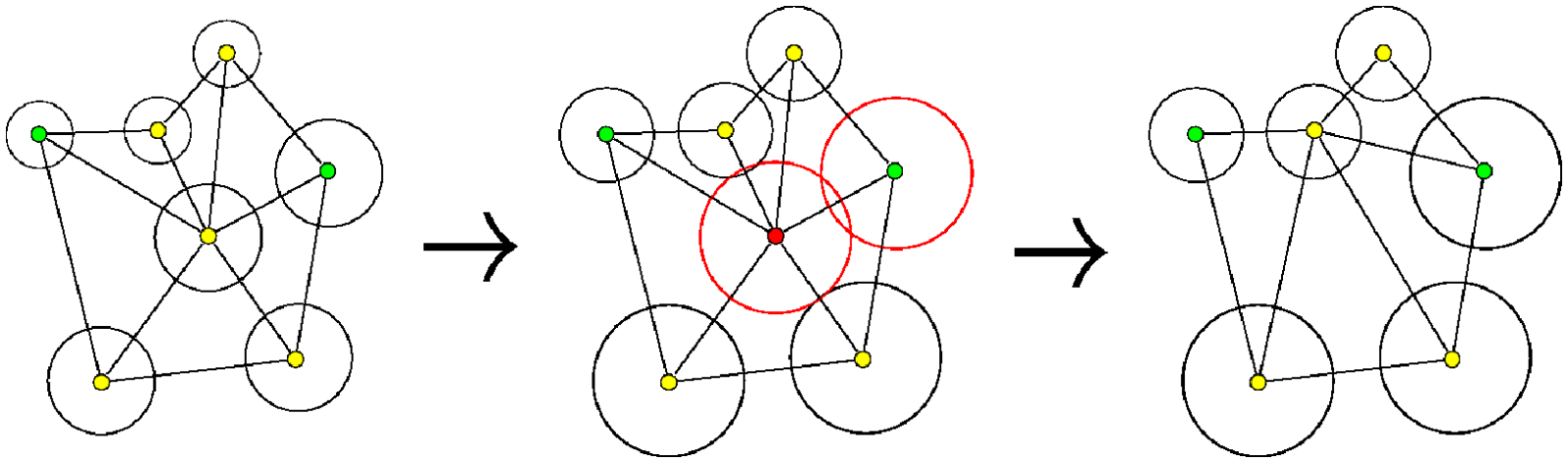
Miller-Talmor-Teng Algorithm



Simple Coarsening

1. Compute a spacing function f for the mesh (Koebe)

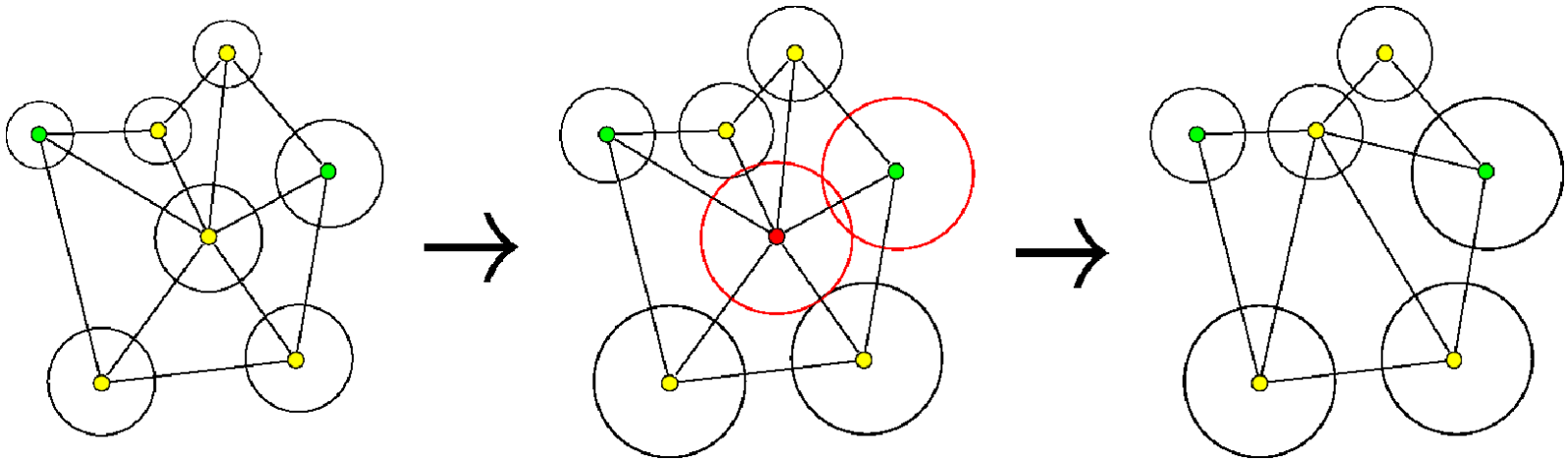
Miller-Talmor-Teng Algorithm



Simple Coarsening

1. Compute a spacing function f for the mesh (Koebe)
2. Scale f by a factor $C > 1$

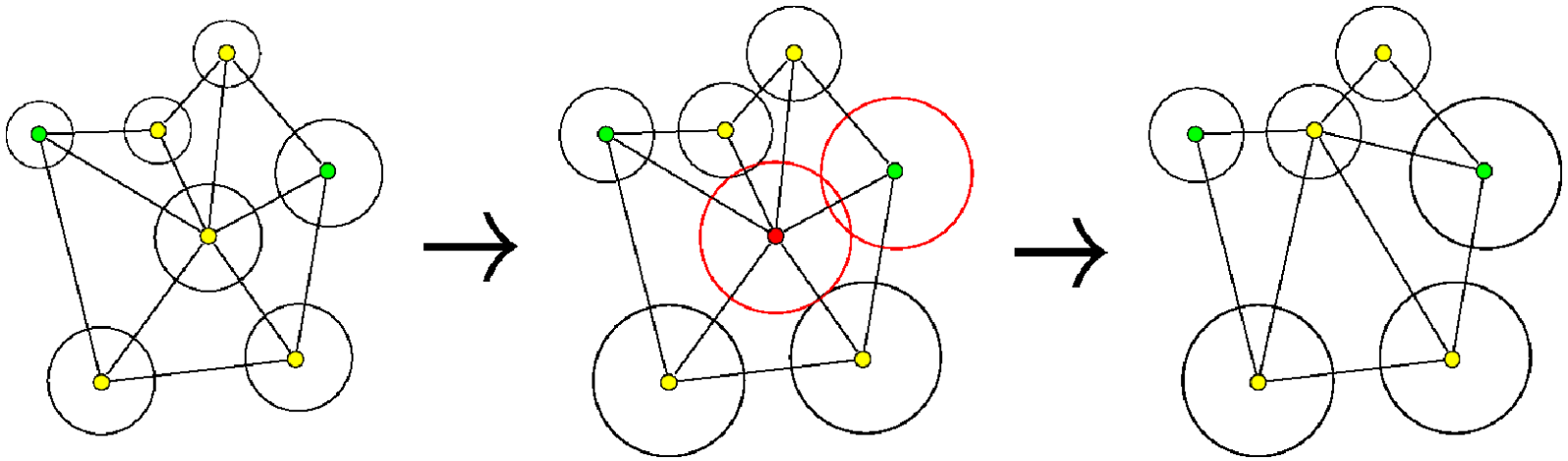
Miller-Talmor-Teng Algorithm



Simple Coarsening

1. Compute a spacing function f for the mesh (Koebe)
2. Scale f by a factor $C > 1$
3. Choose a maximal independent set of vertices for new f

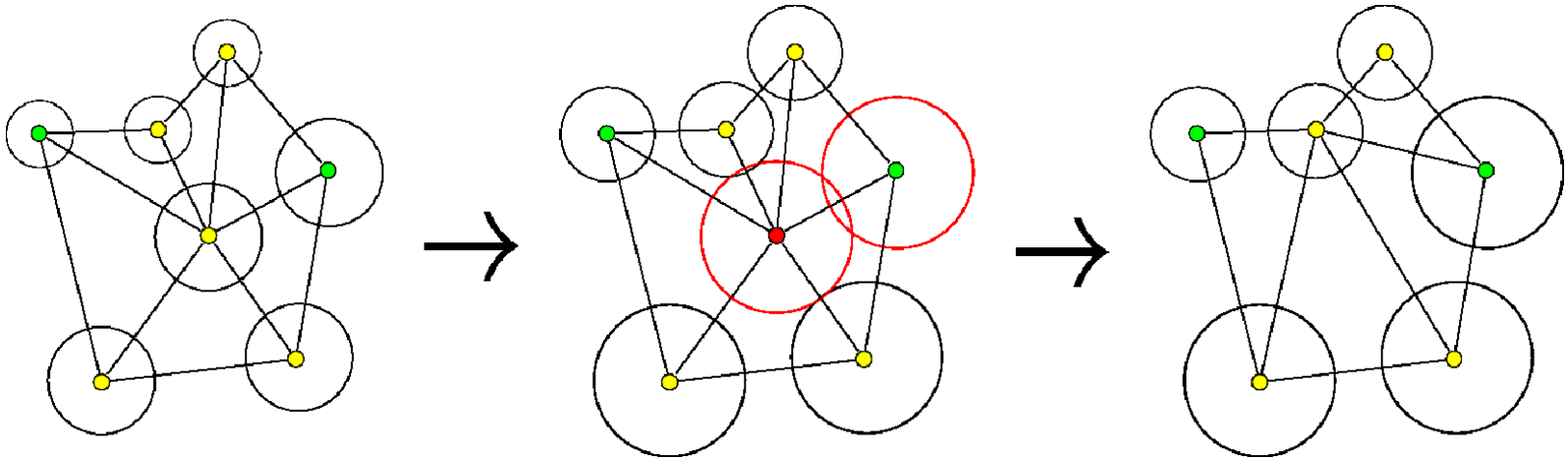
Miller-Talmor-Teng Algorithm



Simple Coarsening

1. Compute a spacing function f for the mesh (Koebe)
2. Scale f by a factor $C > 1$
3. Choose a maximal independent set of vertices for new f
4. Retriangulate

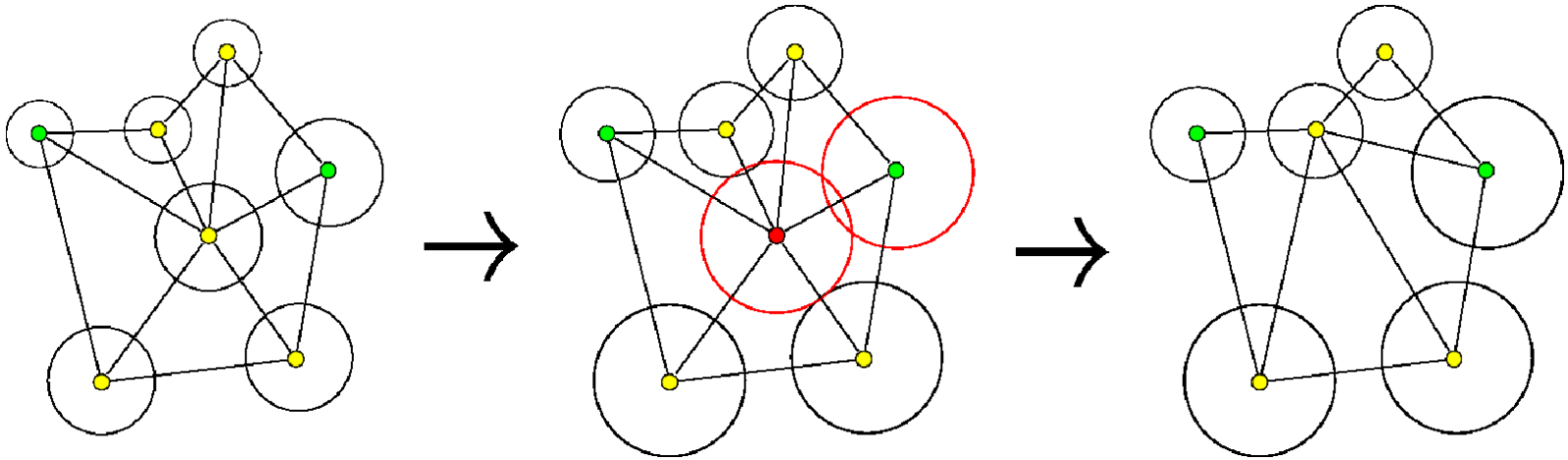
Miller-Talmor-Teng Algorithm



Caveats

1. Must generate coarsest grid in hierarchy first

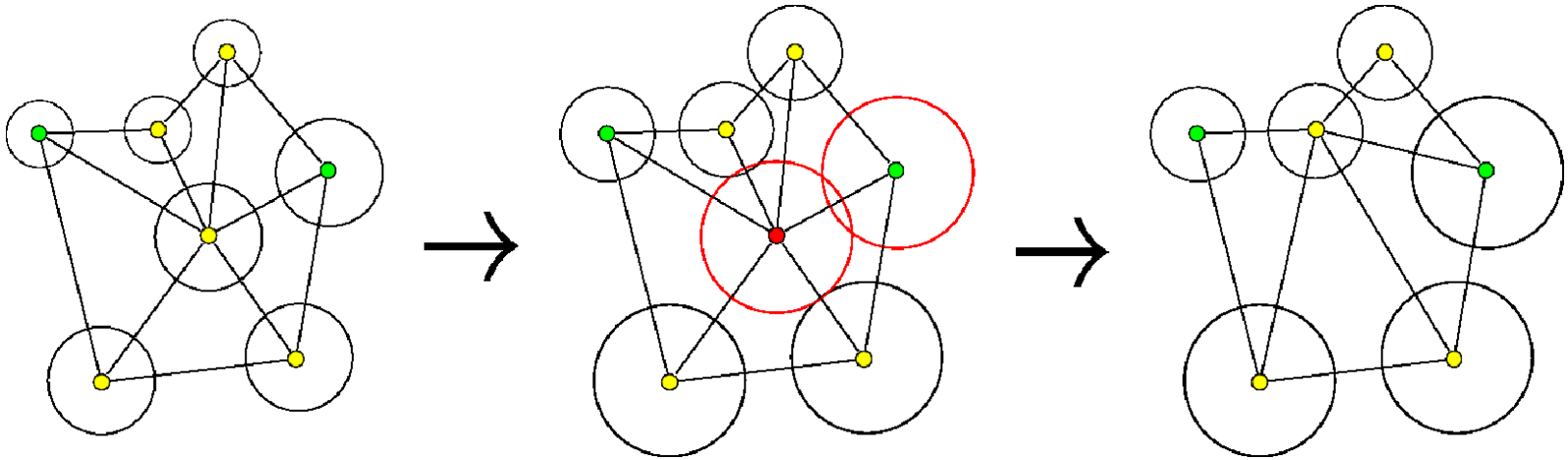
Miller-Talmor-Teng Algorithm



Caveats

1. Must generate coarsest grid in hierarchy first
2. Must choose boundary vertices first (and protect boundary)

Miller-Talmor-Teng Algorithm

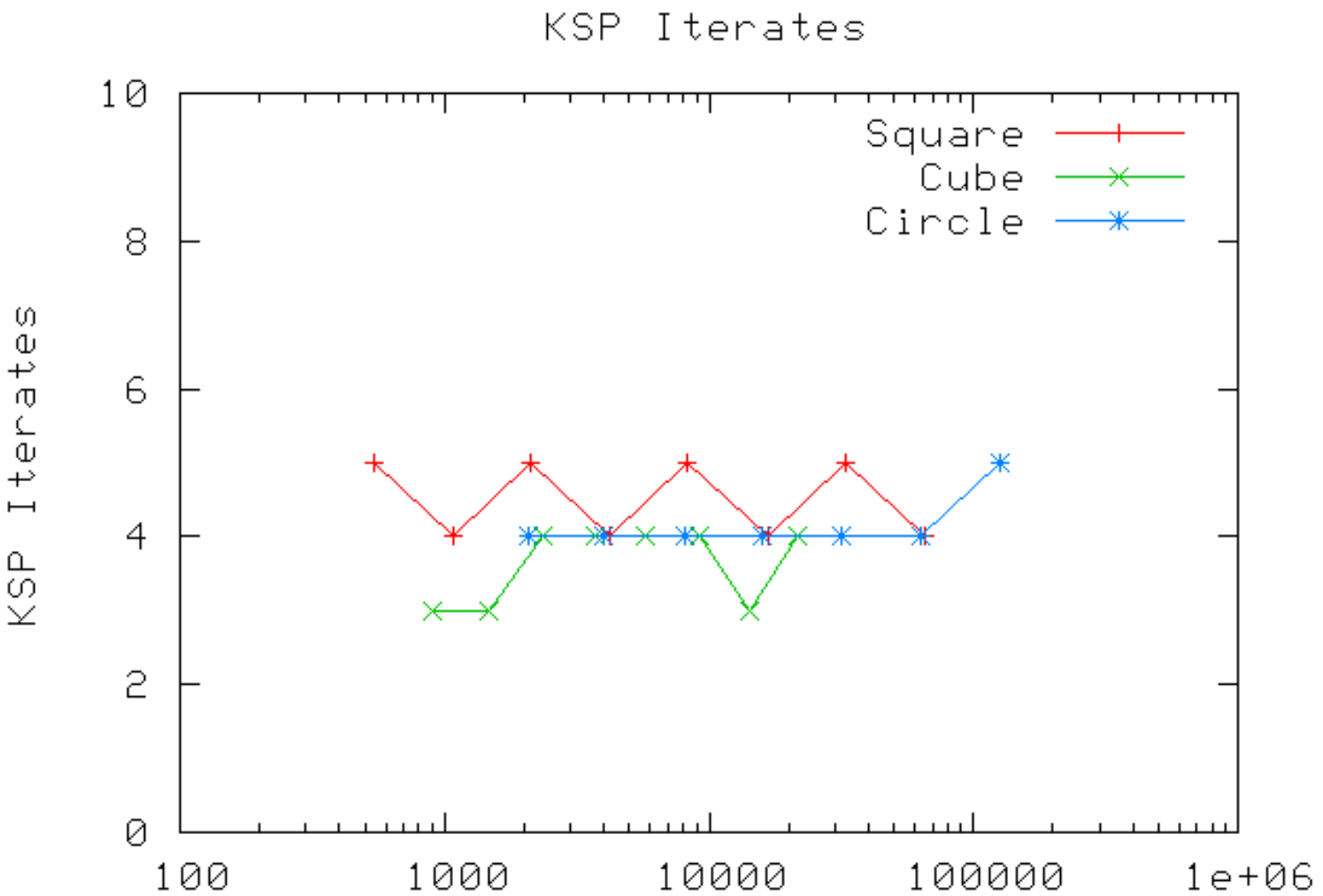


Caveats

1. Must generate coarsest grid in hierarchy first
2. Must choose boundary vertices first (and protect boundary)
3. Must account for boundary geometry

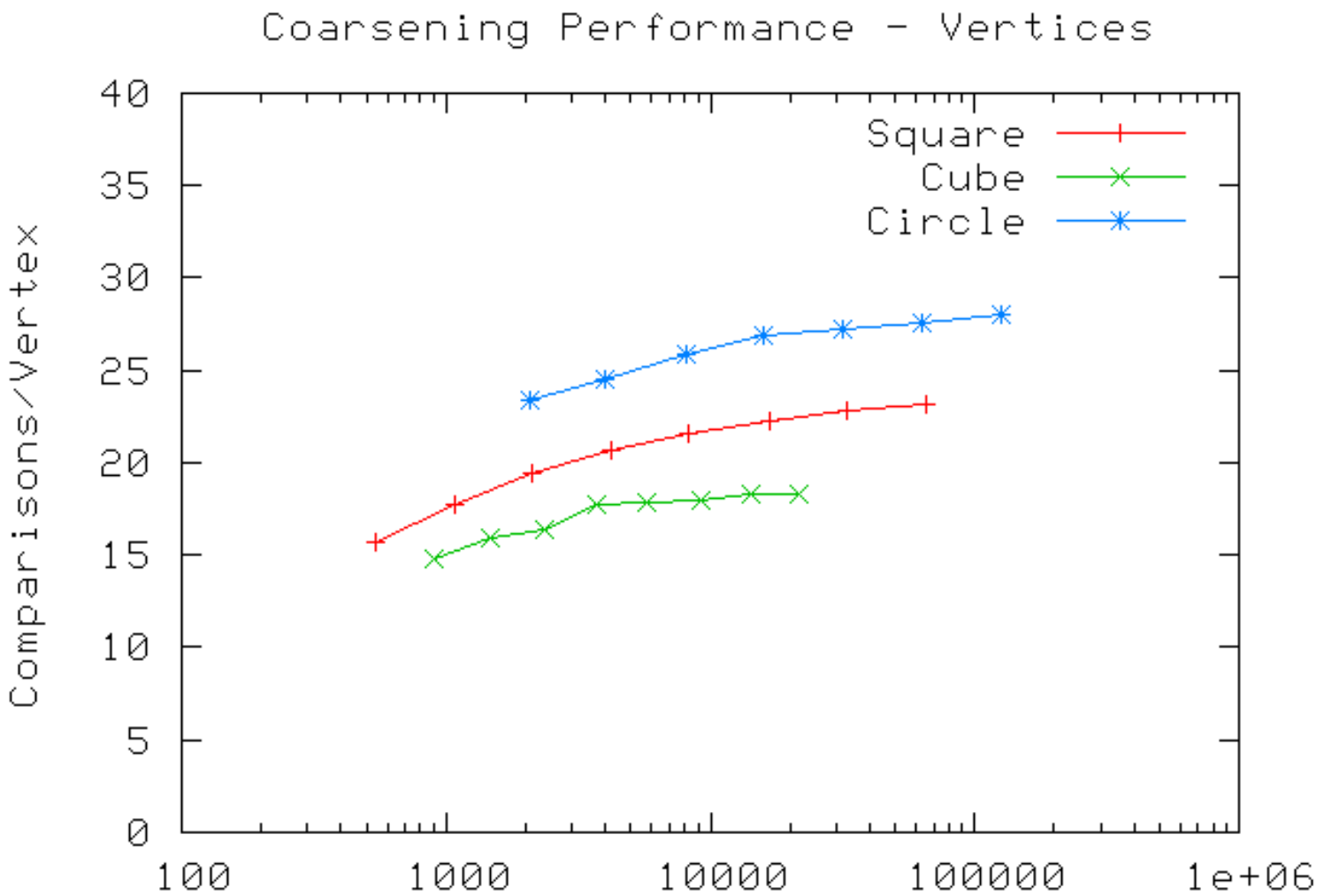
GMG Performance

For simple domains, everything works as expected:
Linear solver iterates are constant as system size increases:



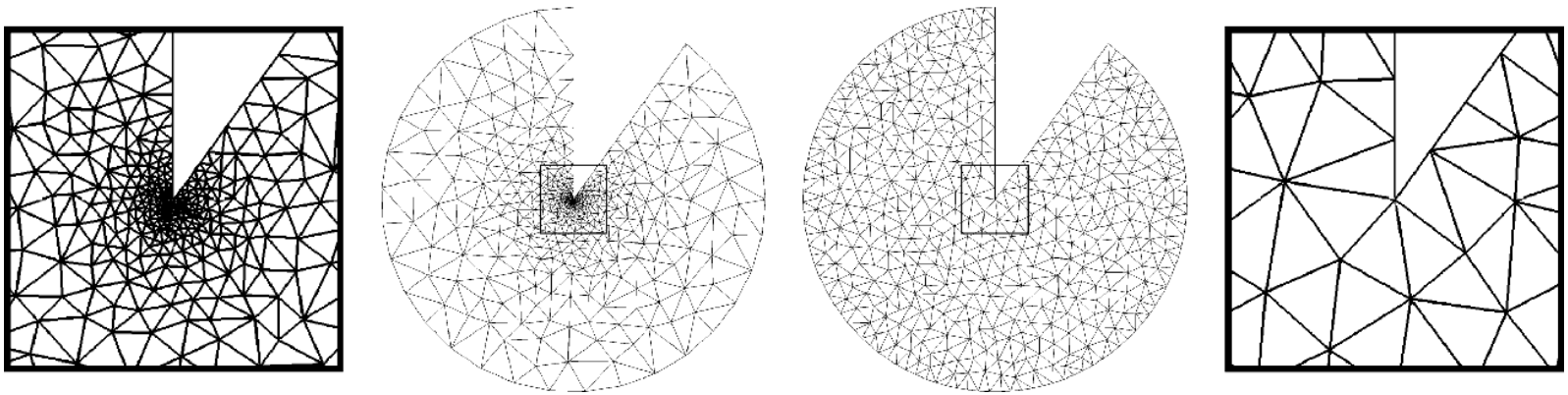
GMG Performance

For simple domains, everything works as expected:
Work to build the preconditioner is constant as system size increases:



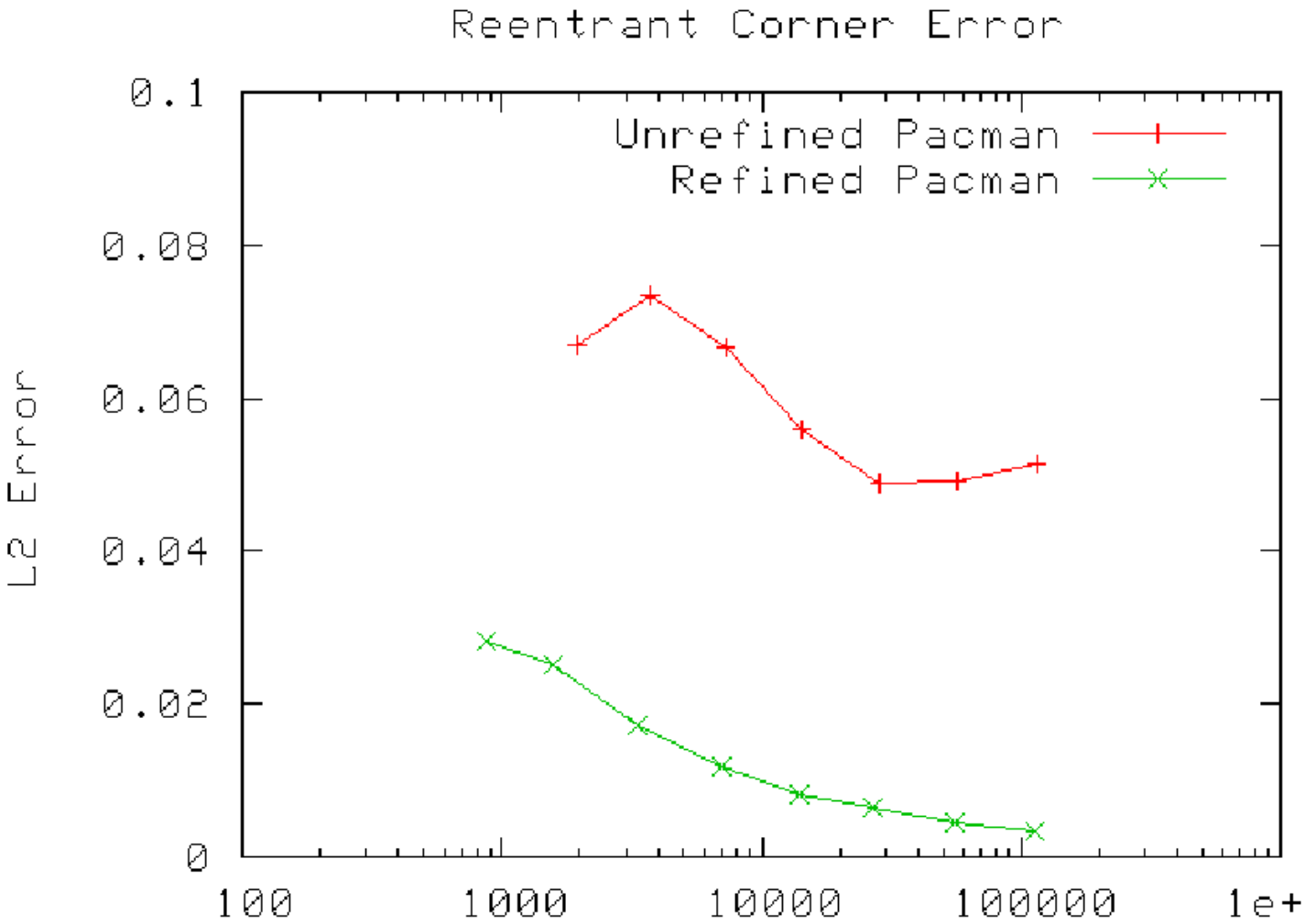
Reentrant Problems

- Reentrant corners need nonnuiform refinement to maintain accuracy
- Coarsening preserves accuracy in MG without user intervention



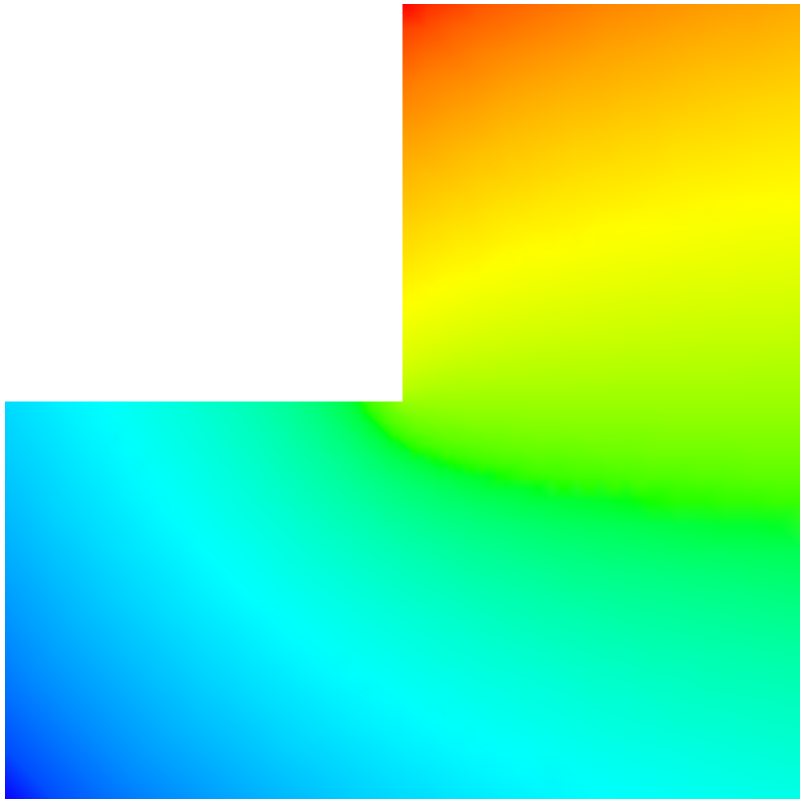
Reentrant Problems

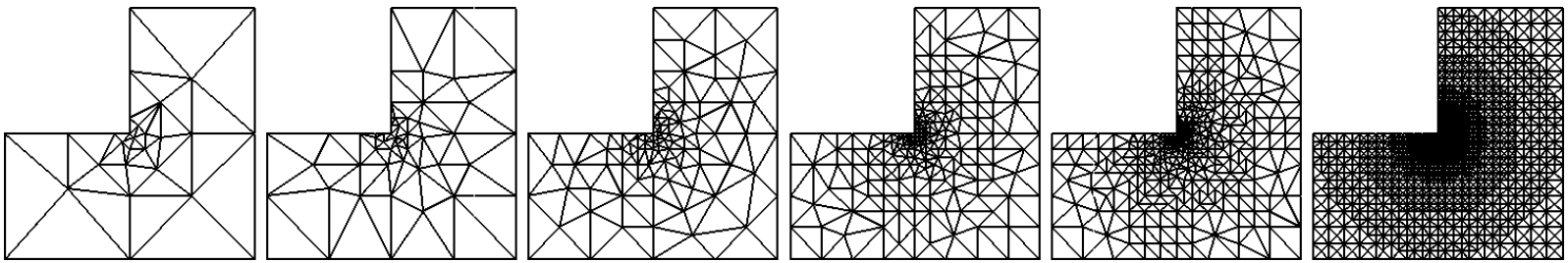
- Reentrant corners need nonnuiform refinement to maintain accuracy
- Coarsening preserves accuracy in MG without user intervention



Reentrant Problems

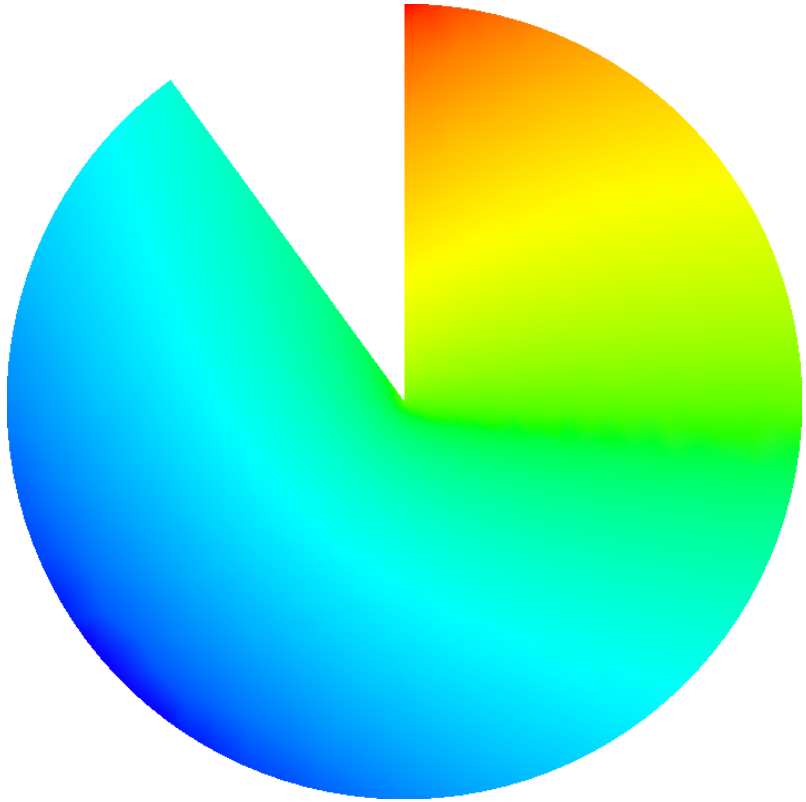
Exact Solution for reentrant problem: $u(x, y) = r^{\frac{2}{3}} sin(\frac{2}{3}\theta)$

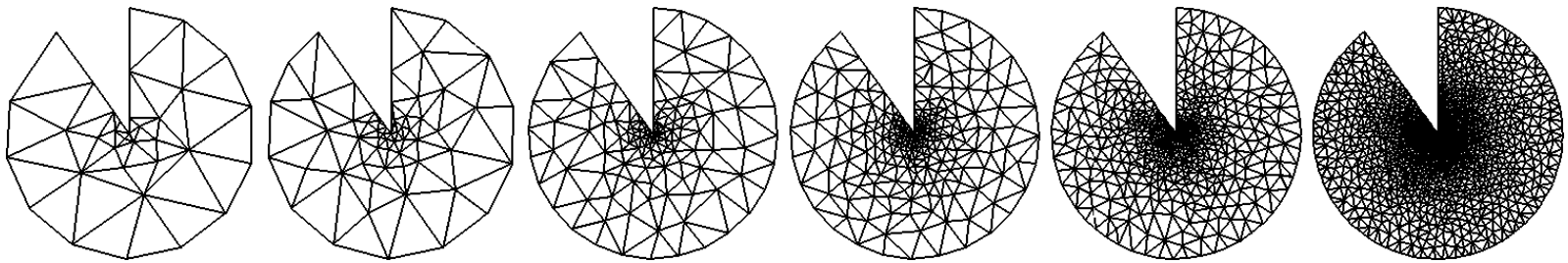




Reentrant Problems

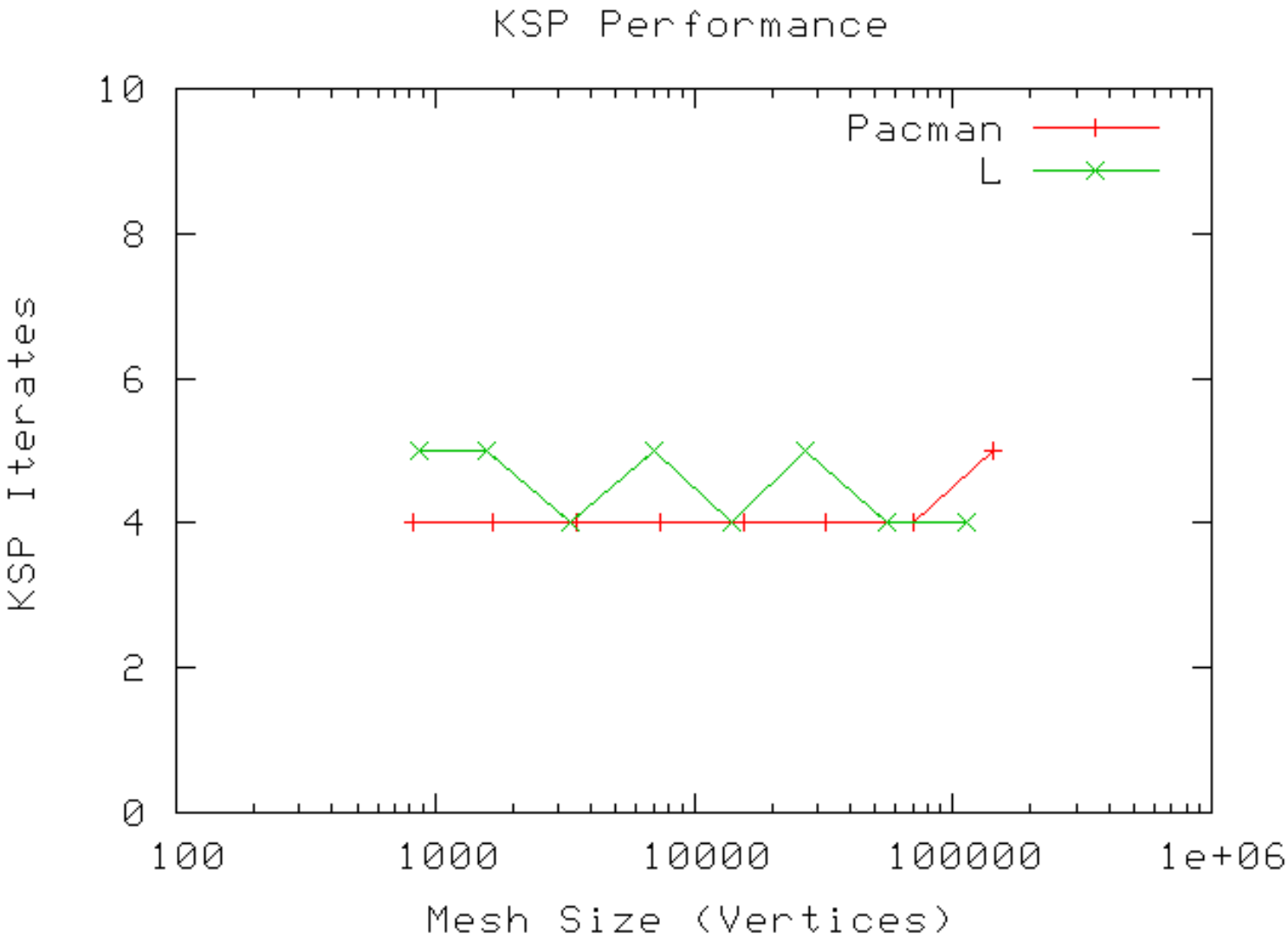
Exact Solution for reentrant problem: $u(x, y) = r^{\frac{2}{3}} sin(\frac{2}{3}\theta)$





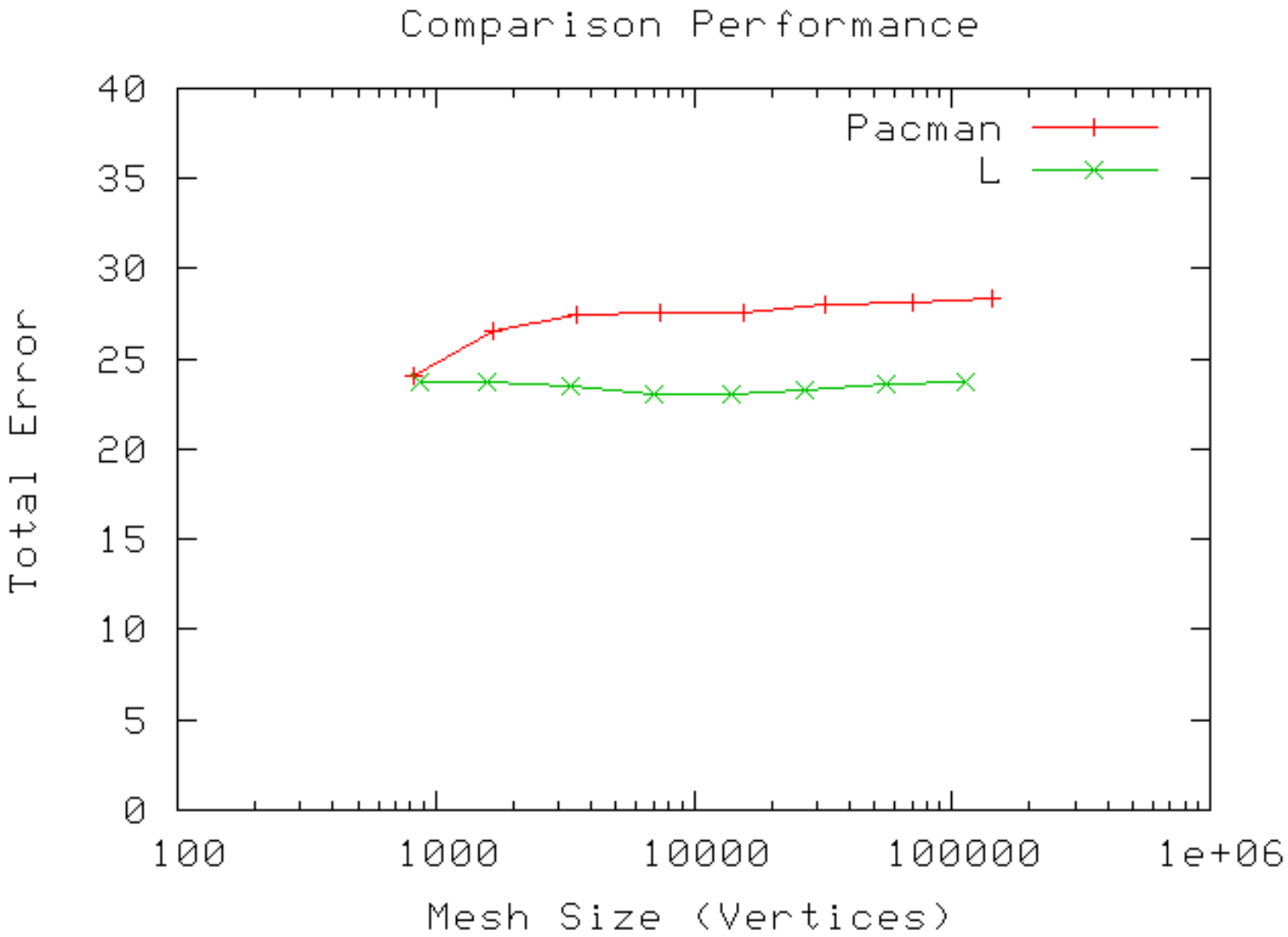
GMG Performance

Linear solver iterates are constant as system size increases:



GMG Performance

Work to build the preconditioner is constant as system size increases:



Outline

Difficulties

We would like to couple

- dynamic rupture
- quasi-static relaxation

Difficulties

We would like to couple

- dynamic rupture
- quasi-static relaxation

However, we must cope with

- different length scales
- different time scales
- solving full algebraic system

Length Scales

We assume that

- we can produce adequate meshes for each subproblem
- these meshes will change slowly during the simulation
- each subproblem admits an efficient, multilevel solver

Length Scales

We assume that

- we can produce adequate meshes for each subproblem
- these meshes will change slowly during the simulation
- each subproblem admits an efficient, multilevel solver

Our proposed solution is then

1. Create a single, adapted mesh

Length Scales

We assume that

- we can produce adequate meshes for each subproblem
- these meshes will change slowly during the simulation
- each subproblem admits an efficient, multilevel solver

Our proposed solution is then

1. Create a single, adapted mesh
2. For multiple coarse hierarchies
   - Can quickly coarsen where solution is smooth

Length Scales

We assume that

- we can produce adequate meshes for each subproblem
- these meshes will change slowly during the simulation
- each subproblem admits an efficient, multilevel solver

Our proposed solution is then

1. Create a single, adapted mesh
2. For multiple coarse hierarchies
   - Can quickly coarsen where solution is smooth
3. Solve each subproblem on its own hierarchy

Time Scales

We assume that

- we can separate time scales *a priori*
- change slowly over the simulation

Time Scales

We assume that

- we can separate time scales *a priori*
- change slowly over the simulation

Our proposed solution is then

1. to formulate the problem as spacetime FEM

Time Scales

We assume that

- we can separate time scales *a priori*
- change slowly over the simulation

Our proposed solution is then

1. to formulate the problem as spacetime FEM
2. use Multi-Adaptive Galerkin timestepping

Multi-Adaptive Galerkin Methods for ODEs I,
Anders Logg, SISC, **24**, pp. 1879–1902, 2003.

Algebraic Solution

We assume that

- each subproblem has an efficient, scalable solver

Algebraic Solution

We assume that

- each subproblem has an efficient, scalable solver

Our solution is then

- use Jacobian-free Newton

Algebraic Solution

We assume that

- each subproblem has an efficient, scalable solver

Our solution is then

- use Jacobian-free Newton
- use *physics-based* (Schwartz) preconditioning

Algebraic Solution

We assume that

- each subproblem has an efficient, scalable solver

Our solution is then

- use Jacobian-free Newton
- use *physics-based* (Schwartz) preconditioning
- could also do this with FAS

Algebraic Solution

We assume that

- each subproblem has an efficient, scalable solver

Our solution is then

- use Jacobian-free Newton
- use *physics-based* (Schwartz) preconditioning
- could also do this with FAS

Jacobian-free Newton-Krylov methods: a survey of approaches and applications, D.A. Knoll and D.E. Keyes, JCP, **193**, 2, pp. 357–397, 2004.

Outline

PETSc Work

- New boundary protection scheme
  - Better preservation of embedded boundaries
- Replace mesh generator call with flips
  - Do not need point insertion

PyLith Work

- Replace KSP solver with DMMG
  - Enables GMG
  - Enables nonlinear solver
- Fix MG interpolation for cohesive elements
  - Collocation
  - Weighted average in a ball