Recent Progress in High Accuracy and High Performance Linear Algebra Algorithms

or

Throwing Away Information For Fun and Profit

James Demmel
Mathematics and Computer Science
UC Berkeley

MSRI Corporate Affiliates Colloquium
4 May 1999

Supported by NSF and DOE

What is Numerical Linear Algebra?

- **Solving**
  - $A \cdot x = b$
  - $A \cdot x = \lambda \cdot x$
  - $A^T \cdot A \cdot x = \sigma^2 \cdot x$
- **Many mathematical modeling problems reduce to these problems**
- **Good libraries available**
  - **LAPACK, ScaLAPACK, NAG, IMSL, HSL, ...**
- **Why is it still hard after all these years?**
  - **Problems get bigger**
    - **FE modeling: 39M unknowns, A sparse**
    - **COBE data set: 1M unknowns, A dense**
  - **Parallel machines harder to program efficiently**
    - **Moving one word of data between processors can cost up to O(1000) arithmetic operations**
  - **Accuracy requirement higher**
    - **Find a few correct digits of an eigenvalue $\sim 10^{-70}$, without using > 70-digit arithmetic**
- **Need new algorithms**

Notation

- n = **dimension of** A**, used measure size, cost**
- $\|x\|$ = **norm of vector** $x = \sqrt{\sum_i x_i^2}$
- $\|A\|$ = **norm of matrix** $A \approx$ **largest entry in** A

How do we improve algorithms?

- **Algorithm must depend on structure of each problem**
  - **Solve $Ax = b$ for general A:**
    Costs $O(n^3)$, using Gaussian elimination (GE)
  - **Solve $Ax = b$, assuming it comes from modeling heat flow, elastic response, etc.:**
    Costs $O(n)$, using Multigrid
- **Depending on structure**
  - **We throw away a lot of information to have less to store and compute**
  - **Information = words of data needed to describe original problem and intermediate results**
  - **Valid if problem "compressible"**
  - **Many examples ...**

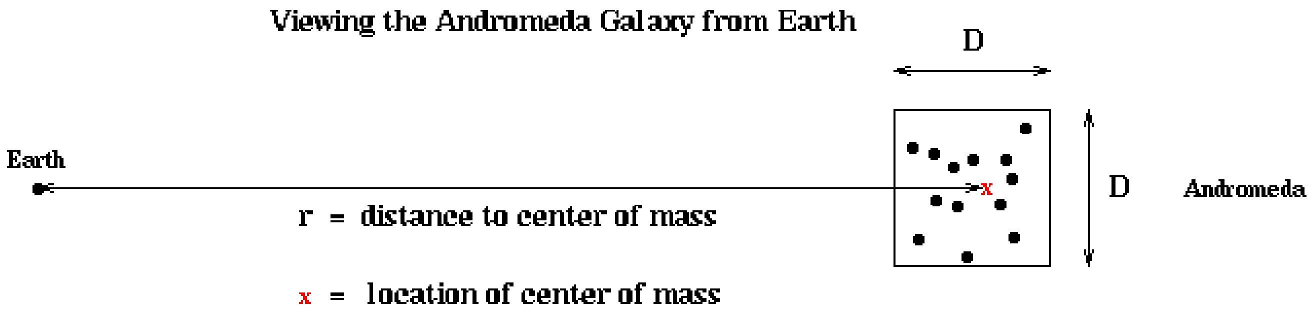
Computing Gravitational force on n particles

for $i = 1$ **to** n

force on $i = \sum_{j \neq i}$ **force on** i **due to** j

$= \sum_{j \neq i} G \cdot m_i \cdot m_j \cdot v(i \to j)/\mathbf{dist}(i \text{ to } j)^2$

- Cost $= O(n^2)$



- Replace cluster of particles by Center of Mass, Total Mass
- Barnes-Hut: Cost $= O(n \cdot \log n)$
- Fast Multipole Method (Greengard/Rokhlin): Cost $= O(n)$
- Revolutionized Astrophysical, Electrostatic computations
  - Warren/Salmon: 17M particles in 1 day vs. 18 years
  - Modeling E-Beam lithography devices using PBody `www.cs.berkeley.edu/~davidb/Pbody.html`
- Not entirely new (Newton) `www.cs.berkeley.edu/~demmel/cs267_Spr99`

Examples to Discuss in Detail

---

1. **Solving sparse $Ax = b$ in parallel using Gaussian Elimination**
   - **Pivoting traditionally used for numerical stability**
   - **Pivoting $\Rightarrow$ lots of communication $\Rightarrow$ not scalable**
   - **Don't pivot (throw away precision) to be scalable, but be stable anyway!**
   - **Up to 8 Gflops on 512 processor Cray T3E**
2. **Finding tiny eigenvalues of symmetric A accurately**
   - **Old algorithms would need arbitrarily high precision, depending on A, to get tiny eigenvalues accurately**
   - **New algorithm always uses 16 digit arithmetic**
     - **Hundreds of times faster**
   - **Throw away digits old algorithm would need to keep**
3. **Solving sparse $Ax = b$ arising from FE models**
   - **Use multigrid**
   - **Starting with finest mesh, throw away mesh points to create coarse representations of problem**
   - **Solve with $n = 39M$ on 960 processors with 50% parallel efficiency**

A Scalable Parallel Implementation of Sparse Gaussian Elimination

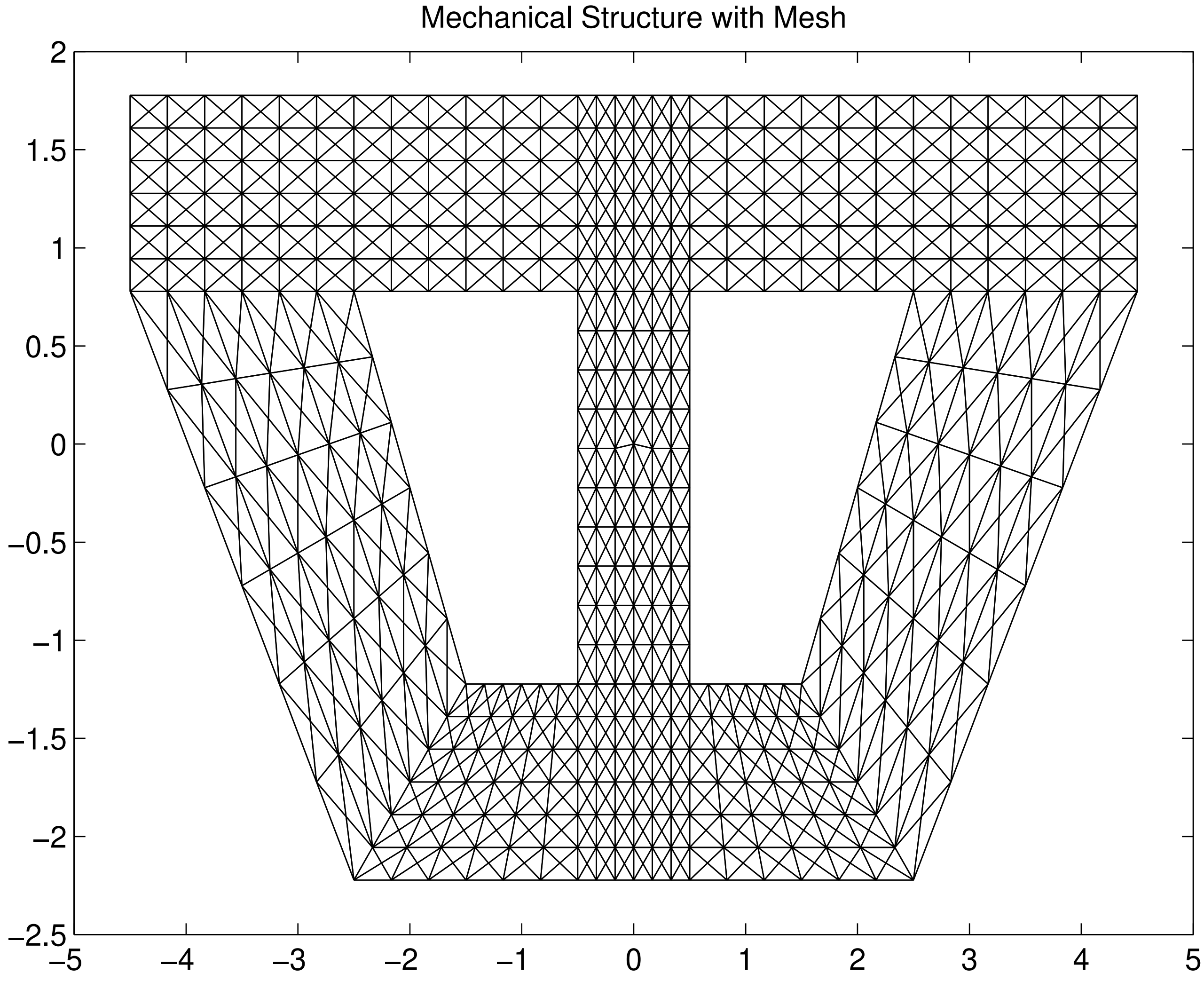
---

With Xiaoye Li, NERSC

- Challenges of implementing sparse Gaussian Elimination (GE) efficiently
- Conventional Wisdom:
  Use dynamic pivoting for stability
  - OK on sequential machines
    `www.cs.berkeley.edu/~xiaoye/superlu.html`
  - OK on small SMPs machines
    `www.cs.berkeley.edu/~xiaoye/superlu_mt.html`
  - Too slow on large machines with high latency
- New algorithm: GE with Static Pivoting (GESP)
  - Risk throwing away digits that might be needed for stability, and clean up later
- Numerical Results

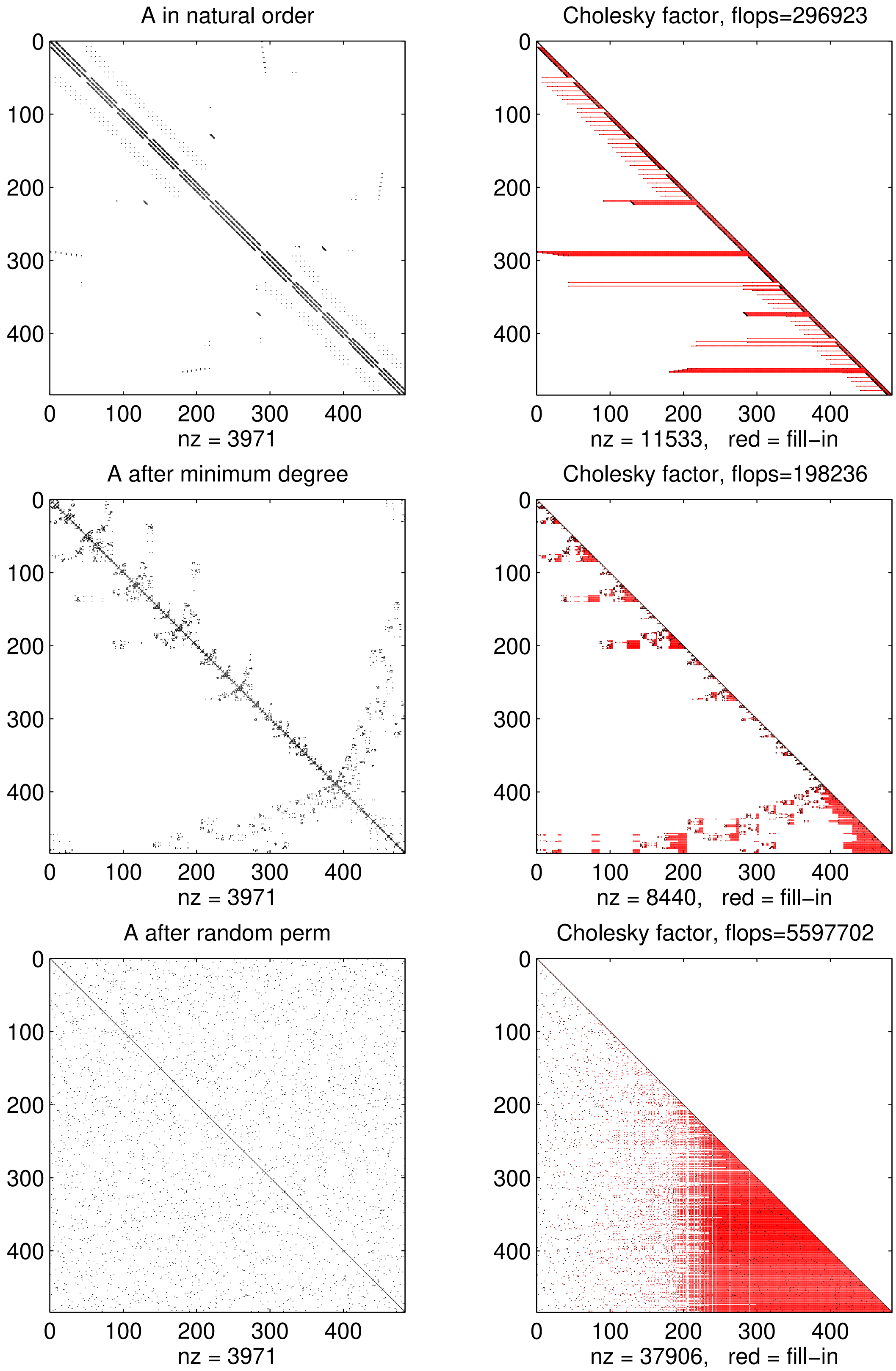
An Easier Problem – Sparse Cholesky

- Applies when A symmetric, positive definite
- Choose a permutation P and compute $PAP^T = LL^T$
  - L is lower triangular
- Choose P to minimize the work and memory for computing L



- Each row/column represented by node in mesh
- Each nonzero of A represented by an edge in mesh

Choosing a Permutation P



Challenges of Sparse Cholesky

- Need to store, operate on nonzeros only
  - Complicated data structures
  - Avoid spending more time figuring out which arithmetic operations to do than doing them
- Ideal choice of permutation P will
  - Minimize work
  - Minimize space
  - Maximize parallelism
- Numerically stable for any P (easier than general A)
- Hard to pick P
  - NP complete
  - Lots of heuristics
- Overall Algorithm
  1. Choose P
  2. Precompute data structure, data layout, and communication schedule for L
  3. Compute L
- Good implementations available
  - Rothberg/Schreiber, Gupta/Kumar, ...

Pivoting for Stability

- **Another constraint on permutation P for general A**
- **Goal of Gaussian Elimination:**
  Factor $A = LU$, L lower, U upper triangular
- **Consider $A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix} = LU$ in 3-digit arithmetic**
- **Get each entry of L and U exactly, except $U(2,2)$:**

$$L = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{bmatrix}$$

  instead of $U(2,2) = -9999.$
- **Threw away one digit in 4th place of $U(2,2)$ - an important one!**
- **Unfortunately $L \cdot U = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 0 \end{bmatrix}$, far from A**
  - **Numerical Instability**
  - **Get completely wrong solution to $Ax = b$**
- **Conventional Cure: Dynamic Pivoting**
  - **Swap rows of A**
  - **At each step, put large number on diagonal**
  - **Compute $PA = LU$**

Cost of Pivoting

- **Dynamic Pivoting harder than Cholesky**
  - **Don't know pivot order until runtime**
  - **Data structure changes dynamically during factorization**
  - **Can't precompute data structure, data layout, and communication schedule for L and U**
  - **Need frequent fine grain communication to pivot**
  - **Expensive!**
- **Our approach: Static Pivoting**
  - **Statically pivot *before* numerical factorization, to precompute data structure, data layout, communication schedule**
  - **Compute L and U without pivoting**
  - **Risk throwing away digits needed for stability!**
  - **GESP - Gaussian Elimination with Static Pivoting**

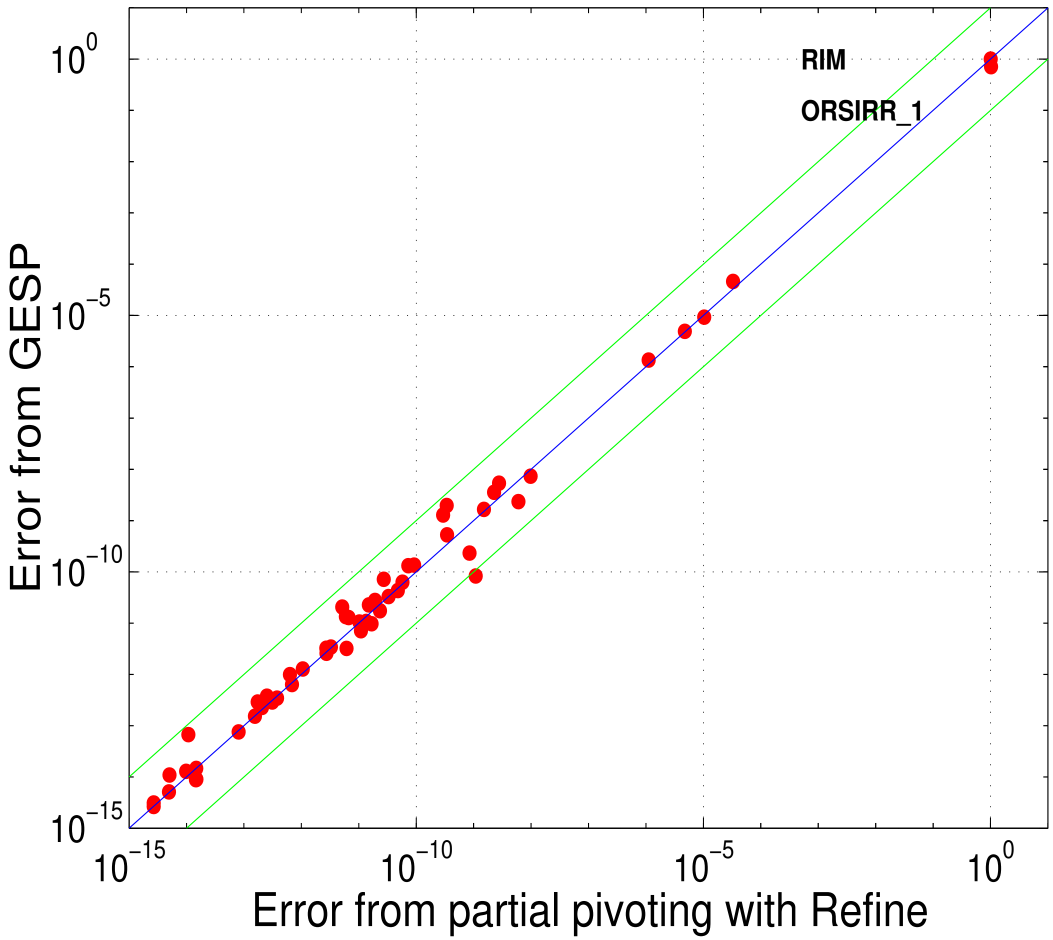
GESP Algorithm

1. **Equilibrate:** $A := D_r \cdot A \cdot D_c$
   - **Make rows and columns of** A **similar in norm**
2. **Pivot rows to maximize diagonal:** $A := P_r \cdot A$.
   - **Weighted bipartite matching on Graph(A) to maximize diagonal of** $P_r \cdot A$ $(\Pi_i |A_{ii}|)$
   - **Duff/Koster 1998**
3. **Permute to minimize work, space; maximize parallelism**
   - $A := P_c^T \cdot A \cdot P_c$
4. **Precompute data structure, data layout, communication schedule**
5. **Factorize** $A = L \cdot U$
   - **Increase tiny pivots to** $\sqrt{\epsilon}\|A\|$
6. **Iterative refinement (with extra precision)**

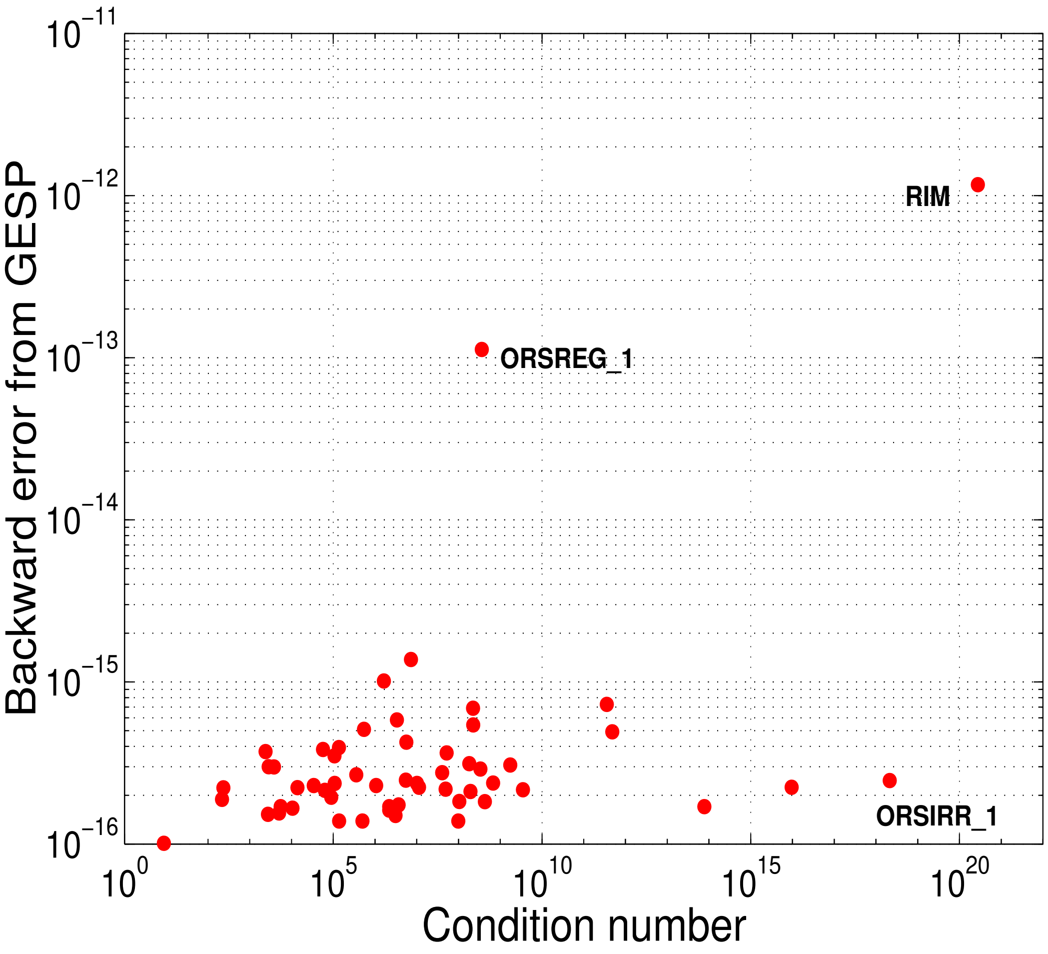
Application Domains

Discipline	Matrices
fluid flow, CFD	af23560, bbmat, bramley1, bramley2, ex11, fidapm11, garon2, graham1, inv-extrusion-1, lnsp3937, lns_3937, mixing-tank, raefsky3, rma10, venkat01, wu
fluid mechanics	goodwin, rim
circuit simulation	add32, gre_1107, jpwh_991, memplus, onetone1, onetone2, twotone
device simulation	wang3, wang4, ecl32
chemical engineering	extr1, hydr1, lhr01, radfr1, rdist1, rdist2, rdist3a, west2021
petroleum engineering	orsirr_1, orsreg_1, sherman3, sherman4, sherman5
finite element PDE	av4408, av11924
stiff ODE	fs_541_2
Olmstead flow model	olm5000
aeroelasticity	tols4000
reservoir modelling	pores_2
crystal growth simulation	cry10000
power flow modelling	gemat11
dielectric waveguide	dw8192 (eigenproblem)
astrophysics	mcfe
plasma physics	utm5940
demography	psmigr_1
economics	mahindas, orani678

The Error $\frac{||x_{true}-x||_\infty}{||x||_\infty}$



The Sparse Backward Error $\max_i \frac{|A \cdot x - b|_i}{(|A| \cdot |x| + |b|)_i}$



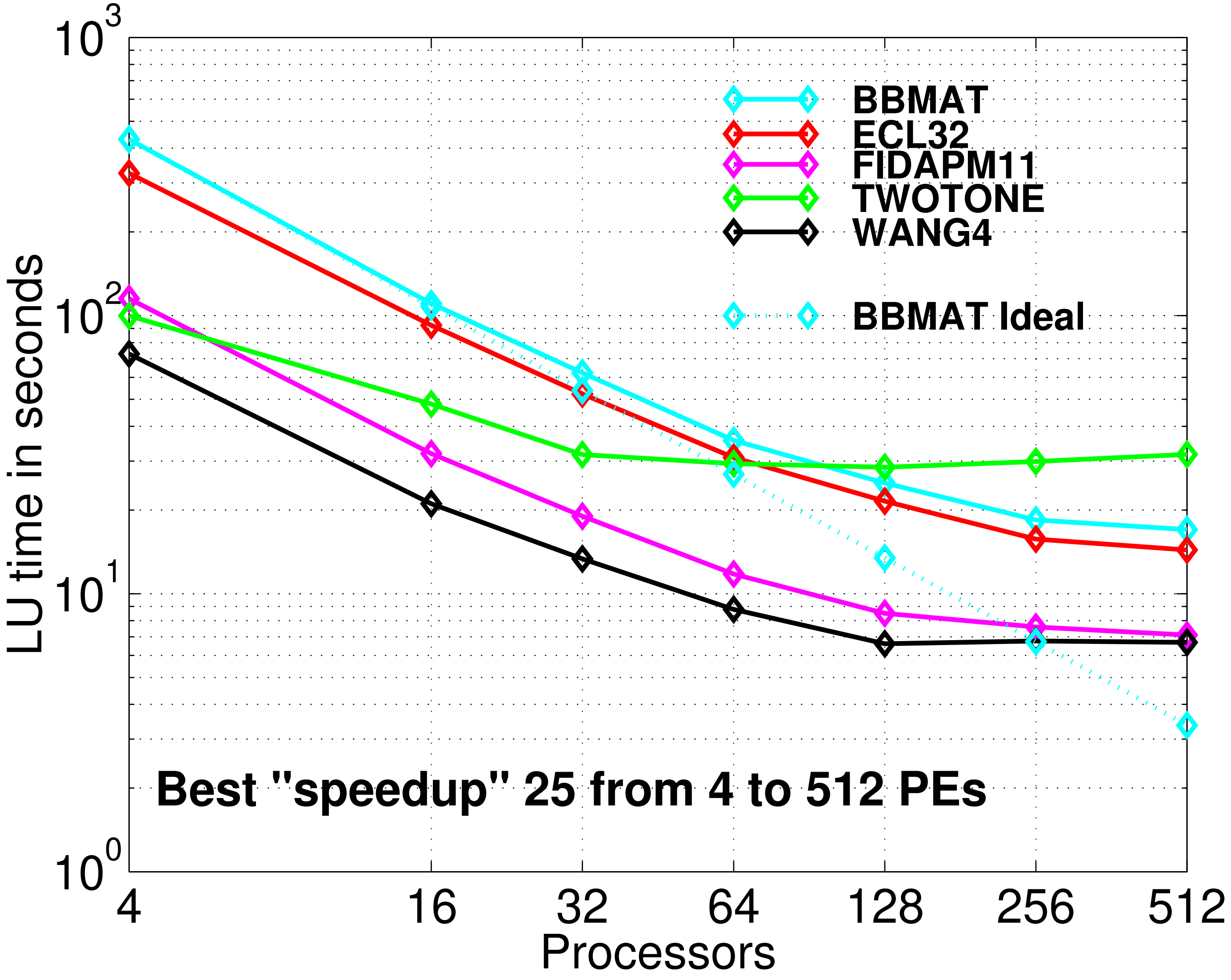
Parallel Performance

- Portable implementation with MPI
  Tested on Cray T3E, ASCI Blue, IBM SP2, Berkeley NOW

- Matrices and their best performance on 512 node T3E-900

	Discipline	Order	$nnz(A)$	$nnz(L+U)$ (10^6)	Flops (10^9)	Mflops
BBMAT	Fluid flow	38744	1771722	49.1	4.3	2493
ECL32	Device sim.	51993	380415	73.5	120.4	8419
FIDAPM11	Fluid flow	22294	623554	23.0	17.9	2291
TWOTONE	Circuit sim.	120750	1224224	22.6	8.7	297
WANG4	Device sim.	26068	177196	27.7	35.3	5542

LU Time on Cray T3E-900



LU Time on Cray T3E-900

	Symbolic	Numeric							
		P=4	16	32	64	128	256	512	Mflops
BBMAT	11.8	430.8	110.5	62.3	35.6	25.8	18.4	17.0	2493
ECL32	14.0	325.0	92.3	52.2	30.8	21.5	15.7	14.3	8419
FIDAPM11	4.1	115.1	31.9	19.0	11.77	8.5	7.6	7.1	2291
TWOTONE	6.6	99.8	48.1	31.6	29.4	28.5	29.9	31.7	297
WANG4	4.2	72.8	21.1	13.3	8.8	6.6	6.76	6.59	5542

- Best "speedup" 25, from 4 to 512 processors

One Application

- Quantum chemistry: Calculate electron impact ionization cross-sections [Baertschy/McCurdy, LBNL]

- Complex, non-Hermitian systems
  - Of order 209,764 on 16 PE Cray T3E, in 2 minutes
  - Of order 736,164 on 24 PE ASCI Blue, in 21 minutes

Conclusions and Future Work for GESP

- OK to risk throwing away digits conventional pivoting would have kept!
  - Minimize risk by static pivoting, iterative refinement,...
- Payoff is much higher performance, with tiny risk of failure
- `www.nersc.gov/~xiaoye/SuperLU`
- Will release software soon!
- Compare to related work
  - SPOOLES (Ashcraft/Grimes/Pierce/Wah, 1997)
  - MUMPS (Amestoy/Duff/L/Excellent, 1998)
  - S* (Fu/Jiao/Yang 1996)
- Future work
  - Further lower risk of instability
    - Sherman-Morrison for tiny pivots
    - Judicious use of extra precision
  - Improve performance
    - Improved use of EDAGs in scheduling
    - Switch to dense code at end
    - Parallelize symbolic part
  - Incomplete (ILU) version

Computing Tiny Eigenvalues of A Accurately

James	Demmel	UC Berkeley
Ming	Gu	UCLA
Stan	Eisenstat	Yale
Ivan	Slapničar	Split
Krešimir	Veselić	Hagen
Zlatko	Drmač	Zagreb

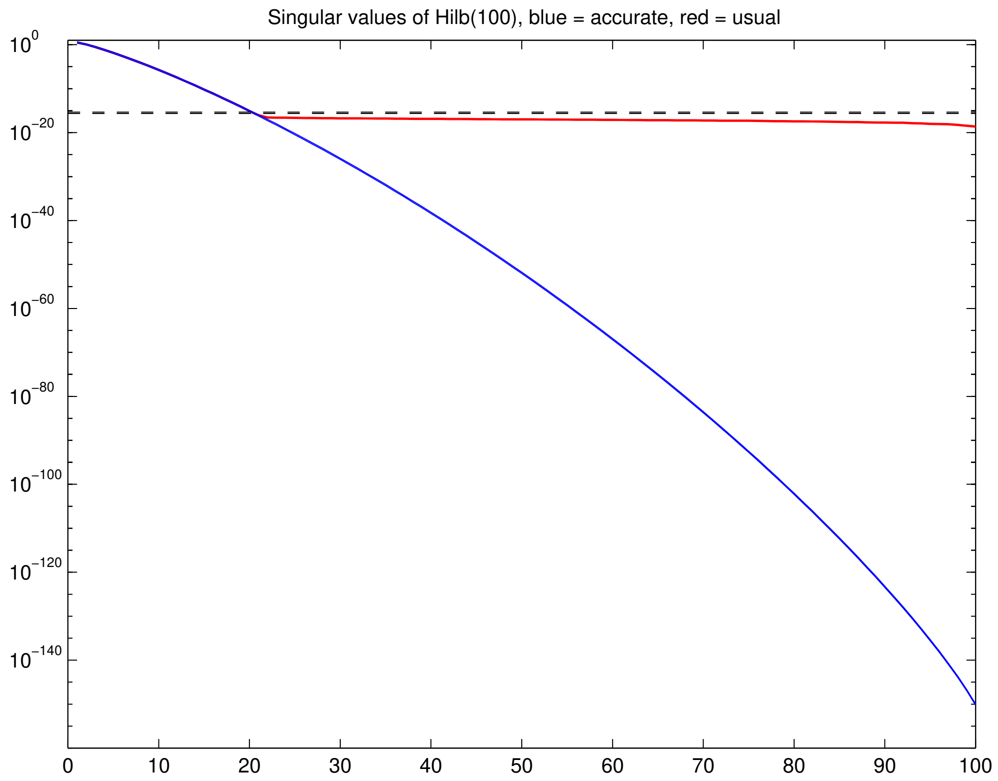
- $Ax_i = \lambda_i x_i$ for i = 1,2, ... , n
- Assume A symmetric and positive semidefinite
  - $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$
  - Applies to Singular Value Decomposition (SVD)
- Same as computing $A = Q\Lambda Q^T$ where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \quad \text{and} \quad Q = [x_1, x_2, \cdots, x_n]$$

  - All x_i orthonormal $\Leftrightarrow$ Q orthogonal
    $\Leftrightarrow$ $\|Qz\| = \|z\|$ for all z
- Goal: Compute all λ_i with correct leading digits, even when $\lambda_i \ll \lambda_1$

Example: 100 by 100 Hilbert Matrix
$H(i,j) = 1/(i+j-1)$

- **Eigenvalues range from 1 down to 10^{-150}**
- **Old algorithm, New Algorithm, both in 16 digits**



- **D = number of digits from λ_1 to λ_n (here $D = 150$)**
- **Cost of Old algorithm = $O(n^3 D^2)$**
- **Cost of New algorithm = $O(n^3)$, independent of D**
  - **Run in double, not bignums as in Mathematica**
  - **New hundreds of times faster than Old**
- **When does it work? Not for all matrices ...**
- **Why bother?**

Why do we want tiny eigenvalues accurately?

1. When they are determined accurately by the data
   - Hilbert: $H(i,j) = 1/(i+j-1)$
   - Cauchy: $C(i,j) = 1/(x_i + y_j)$
   - Changing x_i or y_j in d-th digit changes each eigenvalue in d-th digit (for Hilbert, similar for Cauchy)
2. In some applications, tiny eigenvalues matter most
   - Quantum mechanics: want lowest energy levels
   - Elasticity: want lowest frequencies of vibration
   - Large eigenvalues are artifacts of discretization

Sources of error in Old Algorithms

- **Compute sequence**

  $A = Q_1 \cdot A_1 \cdot Q_1^T = Q_2 \cdot A_2 \cdot Q_2^T = \cdots = Q_i \cdot A_i \cdot Q_i^T \to Q \cdot \Lambda \cdot Q^T$

  - **Each Q_i orthogonal**
  - A_i **converges to** $\Lambda = \mathbf{diag}(\lambda_1, ..., \lambda_n)$

- **At step i, rows of A_i and Q_i get updated**
- **Assume each operation rounded to 16 digits**
- **Computed A_i = True A_i + error,**
  where $\|$error$\| = O(10^{-16}\|A\|)$.
- **Theorem (Weyl, Wilkinson):**
  Computed λ_i = True $\lambda_i \pm O(10^{-16}\|A\|)$
- **Example: If $\|A\| \approx 1$ and we want $\lambda_n \approx 10^{-150}$ to 16 digits, we need to compute with 150+16=166 digits**

Error in New Algorithm

- **Abandon orthogonal Q_i and instead compute**

  $A = L{\cdot}D{\cdot}U \equiv L_1{\cdot}D_1{\cdot}U_1 = L_2{\cdot}D_2{\cdot}U_2 = \cdots = L_i{\cdot}D_i{\cdot}U_i \rightarrow Q{\cdot}\Lambda{\cdot}Q^T$

- **Step 1:** $A = L \cdot D \cdot U$
  - L **and** U **"triangular", and** D **diagonal**
  - L **and** U **close to orthogonal:** $\|Lz\| \approx \|z\|$ **for all** z
  - **Looks like Gaussian elimination with complete pivoting (identical if no roundoff)**
  - **Actual algorithm depends on structure of** A **(not just Gaussian Elimination!)**
  - L, D **and** U **must all be accurate to 16 digits**

- **Step 2: Compute sequence of** $L_i \cdot D_i \cdot U_i$
  - **Variant of Jacobi's Method**
  - **Independent of structure of** A

- **Theorem (DGESVD): Keeping just** d **digits in each operation, and throwing away the rest, yields the first** d **digits of each eigenvalue**

How do we compute $A = L \cdot D \cdot U$ for a Hilbert (or Cauchy) Matrix?

- **How can we lose accuracy in computing?**
  - OK to multiply, divide, add positive numbers
  - OK to subtract exact numbers (initial data)
  - Cancellation when subtracting approximate results dangerous:

$$\begin{array}{rl} & .12345\text{xxx} \\ - & .12345\text{yyy} \\ \hline & .00000\text{zzz} \end{array}$$

- **Cauchy:** $C(i,j) = 1/(x_i + y_j)$
- **det**$(C) = \Pi_{i<j}(x_j - x_i)(y_j - y_i)/\,\Pi_{i,j}(x_i + y_j)$
  - No bad cancellation $\Rightarrow$ good to 16 digits
- **Linear Algebra Fact: Each entry of L, D, U is a quotient of determinants**
- **Change inner loop of Gaussian Elimination from**

$$C(i,j) = C(i,j) - C(i,k) * C(k,j)/C(i,i)$$

  to

$$C(i,j) = C(i,j) * (x_i - x_k) * (y_j - y_k)/(x_k + y_j)/(x_i + y_k)$$

- **Each entry of L, D, U accurate to 16 digits!**

For which matrices is high accuracy attainable?

- **Cauchy, Vandermonde, related matrices from DSP**
- **Sparse matrices (depending on sparsity pattern)**
- **Discretized ODES and PDES**
  - **Sturm-Liouville ODE:** $(p(x)y')' + q(x)y = \lambda r(x)y$
  - **Discretize usual way: second centered differences**
  - **Get tridiagonal** $Tx = \lambda x$**; Don't form it!**
  - **Rewrite as** $GG^T y = \lambda y$ **where** $G = D_1 \cdot Z \cdot D_2$
    - D_1 **and** D_2 **are diagonal, depend on data**
    - Z **is an incidence matrix of discretization: combinatorial only**
  - **Theorem (Poincarè,DGESVD):**
    Can compute $G = LDU$ **very accurately**
  - **Conjecture: Take a PDE, form a mesh, and discretize. The accuracy with which we can compute eigenvalues is limited by the geometry of the mesh, not the coefficients of the PDE**
  - **Similar results by Vavasis for solving** $Ax = b$
- **General result:** $\exists$ **poly(**n**) time algorithm for very accurate tiny eigenvalues of** A **if each minor of** A **can be factored as** $\Pi_{i=1}^{M} p_i/q_i$ **where** M **=poly(**n**) and** p_i **and** q_j **are polynomials of poly(**n**) size.**

Conclusions and Future Work on Accurate Eigenvalues

- Possible to compute tiny eigenvalues accurately much faster than using conventional algorithm with high precision
- Motivated by applications where tiny eigenvalues more important than large ones
- Technical Reports
  - `www.cs.berkeley.edu/~demmel/DGESVD.ps.gz`
    to appear in Lin. Alg. Appl.
  - `www.cs.berkeley.edu/~demmel/cauchy.ps.gz`
    to appear in SIAM. J. Mat. Anal.
- Some algorithms will appear in LAPACK
- Future work
  - Generalizations of Vandermonde, ...
  - More general discretizations of ODEs and PDEs

Parallel Multigrid for Finite Element Problems on Irregular Meshes

---

With Mark Adams and Bob Taylor, Berkeley

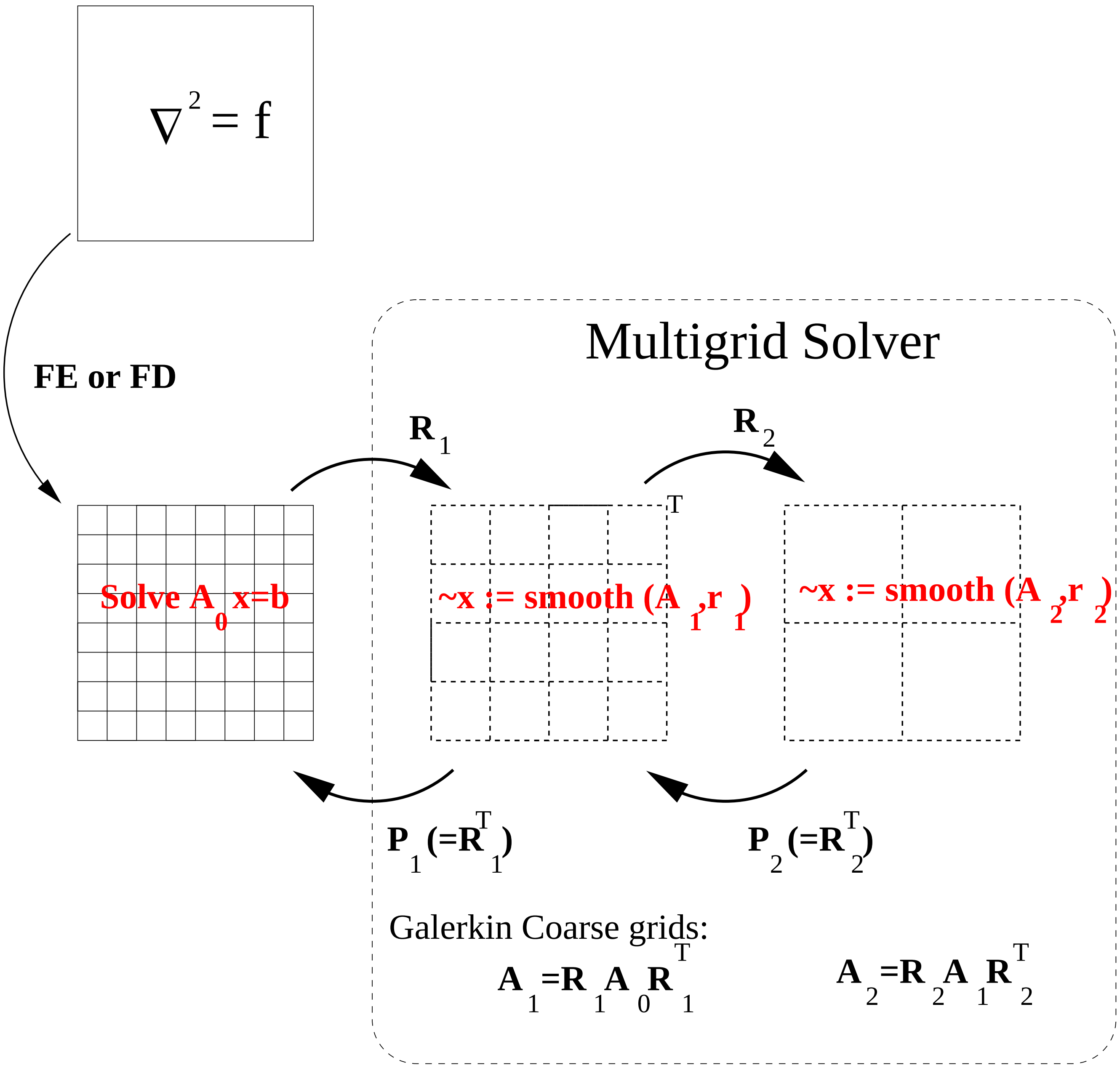
- Exploit Structure!
- Review multigrid on regular meshes
  - Throw away information to get approximations on coarse meshes
- Challenges for irregular meshes
  - Coarse mesh generation from fine mesh
  - Parallelization
- Solver architecture
- Numerical results
- Conclusions

Exploiting structure of A to solve $Ax = b$ faster

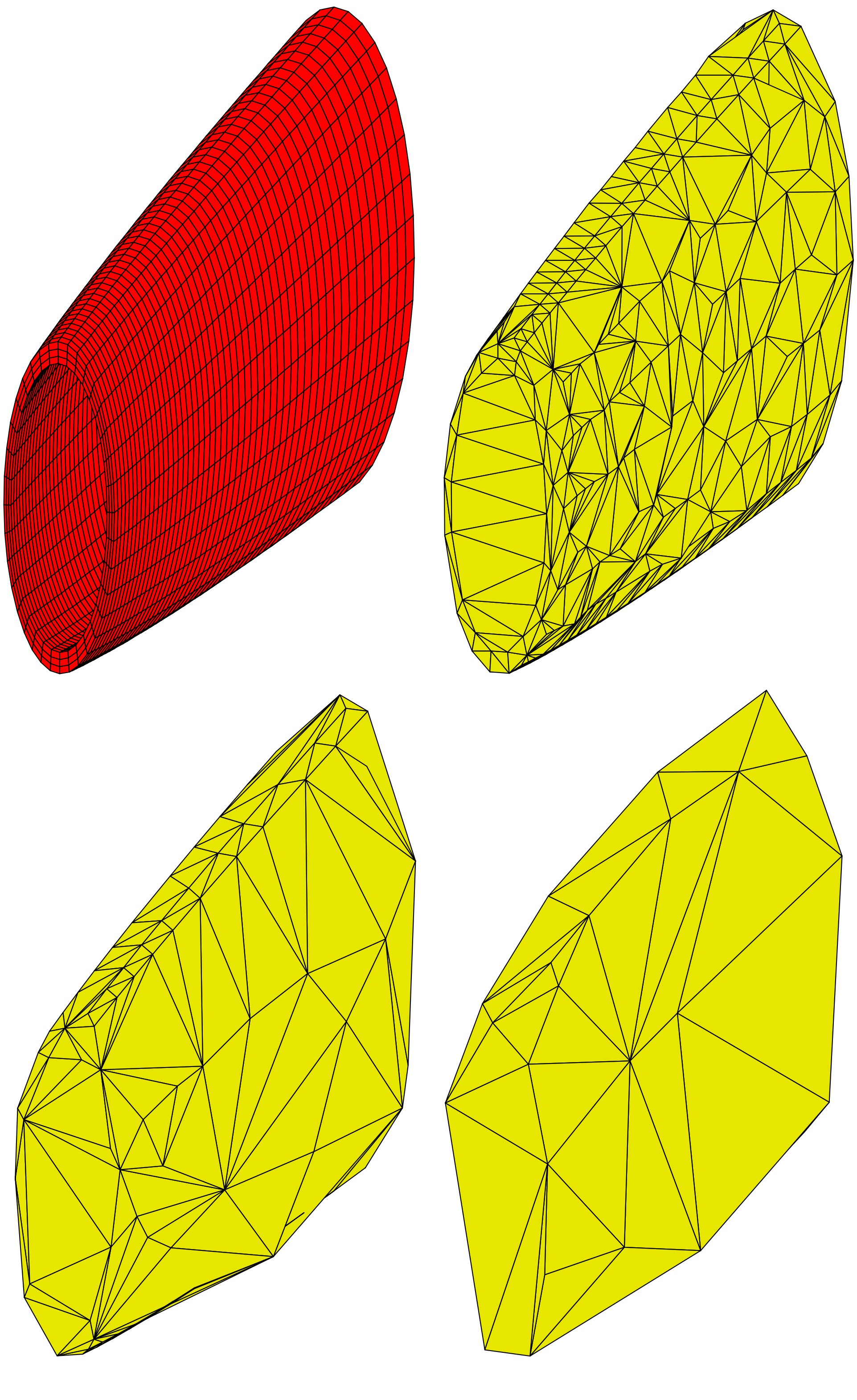
Algorithm	Serial Cost	"Parallel Cost"
Dense Gaussian Elimination	N^3	N
Band Gaussian Elimination	N^2	N
Conjugate Gradients	$N^{1.5}$	$N^{.5} \log N$
Multigrid	N	$\log^2 N$
Lower Bound	N	$\log N$

- Costs in $O()$ sense
- Parallel cost is for PRAM
  - As many processors as you like
  - Free communication
  - Unrealistic, "best case"

Multigrid on a structured mesh



Sample Fine Input Grid and Coarse Grids



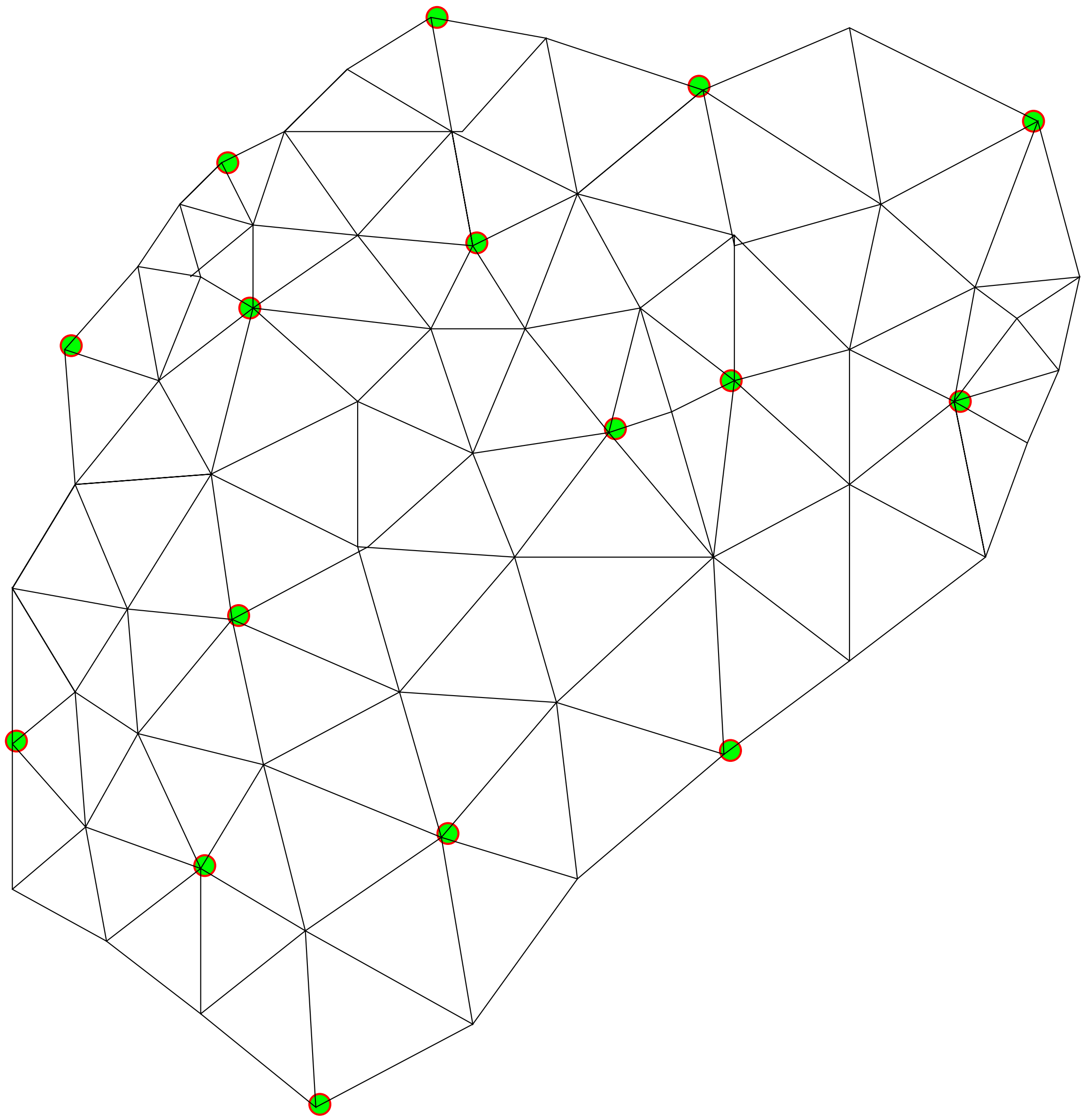
Algorithm Outline

- Based on Guillard (1992), Chan and Smith (1994)
- Mixed geometric/algebraic MG

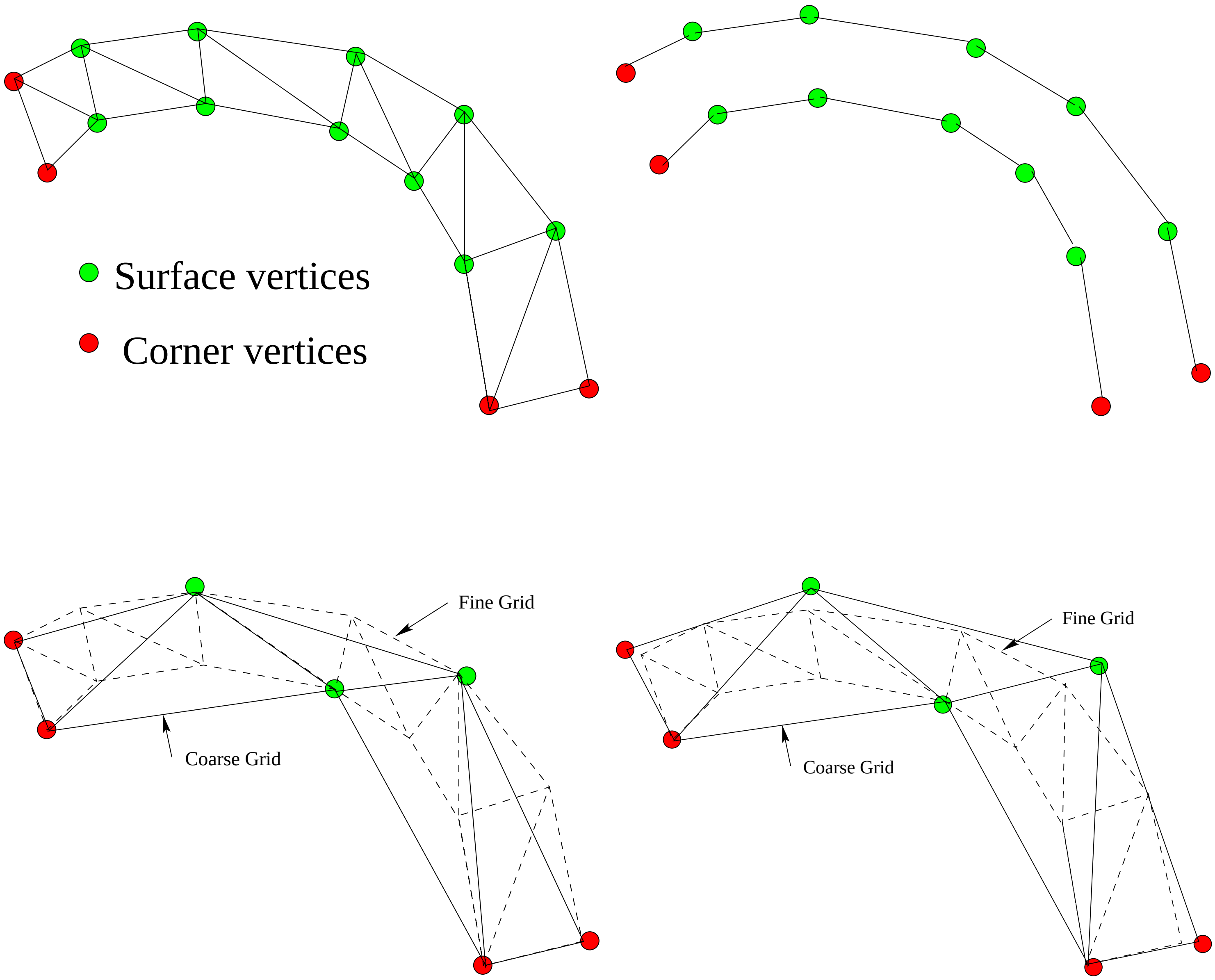
1. Produce coarse meshes - throw away details
   - Maximal Independent Sets to get vertex set
   - Heuristics to preserve geometry, interfaces in coarse meshes
   - Delaunay Mesh the vertex set
2. Restriction, Interpolation operators R and R^T
   - Use FE shape functions
3. Galerkin Coarse Grid Operators
   - $A_{coarse} = R \cdot A_{fine} \cdot R^T$
4. Solve using multigrid
   - Load Balance using ParMetis
   - Accommodates CLUMPs (Cluster of SMPs)
   - Coarse Grid Agglomeration
   - Use PETSc
5. Nonlinear problems
   - Control iterations using FEAP

Producing Coarse Grids

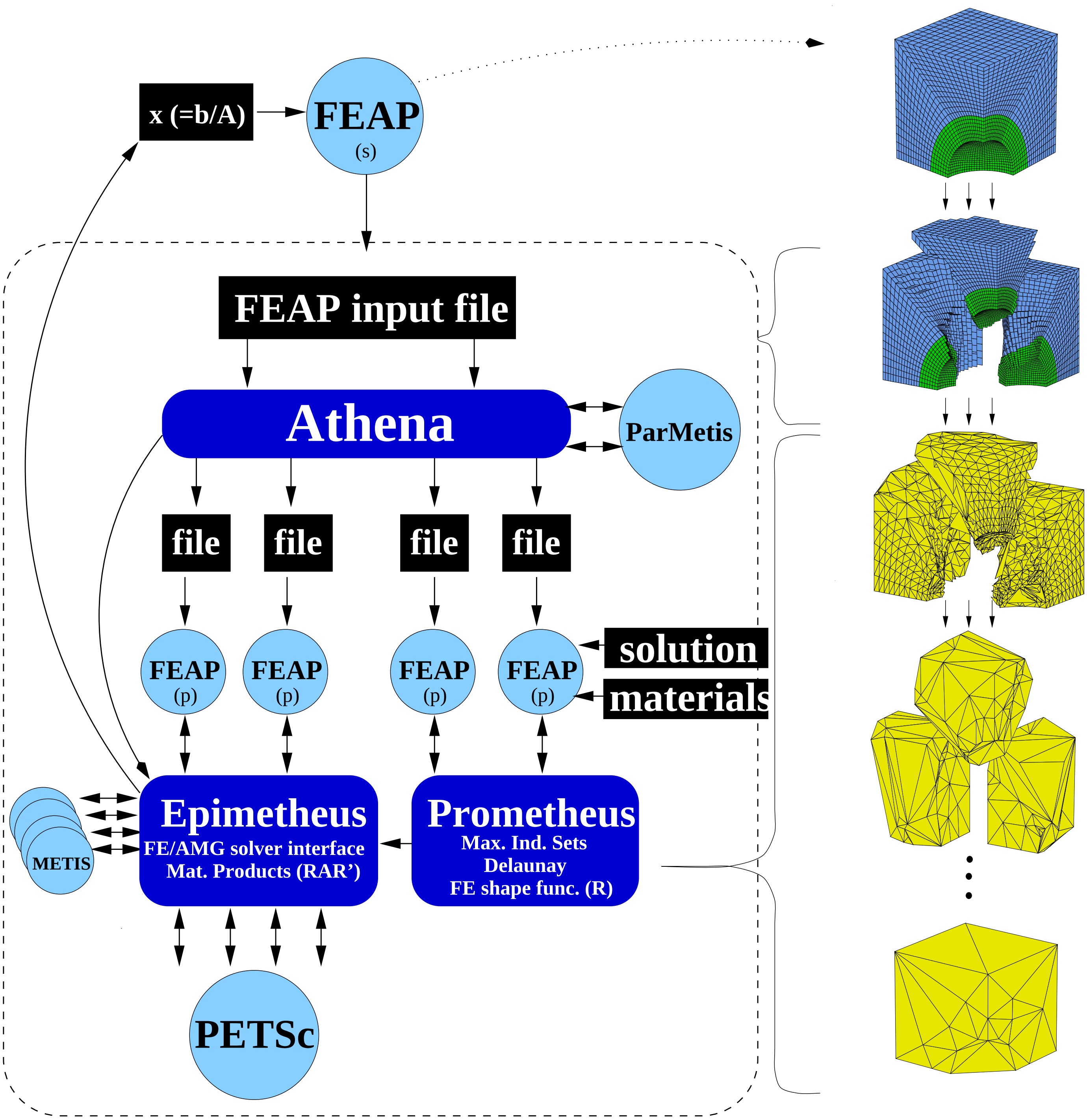
- **Goals**
  - Preserve geometry (corners, edges, faces)
  - Preserve boundaries (material discontinuities)
- **Basic Tool: Maximal Independent Set (MIS)**
  - No two nodes in MIS connected
  - Can't add any nodes to MIS



Producing Coarse Grids – Example

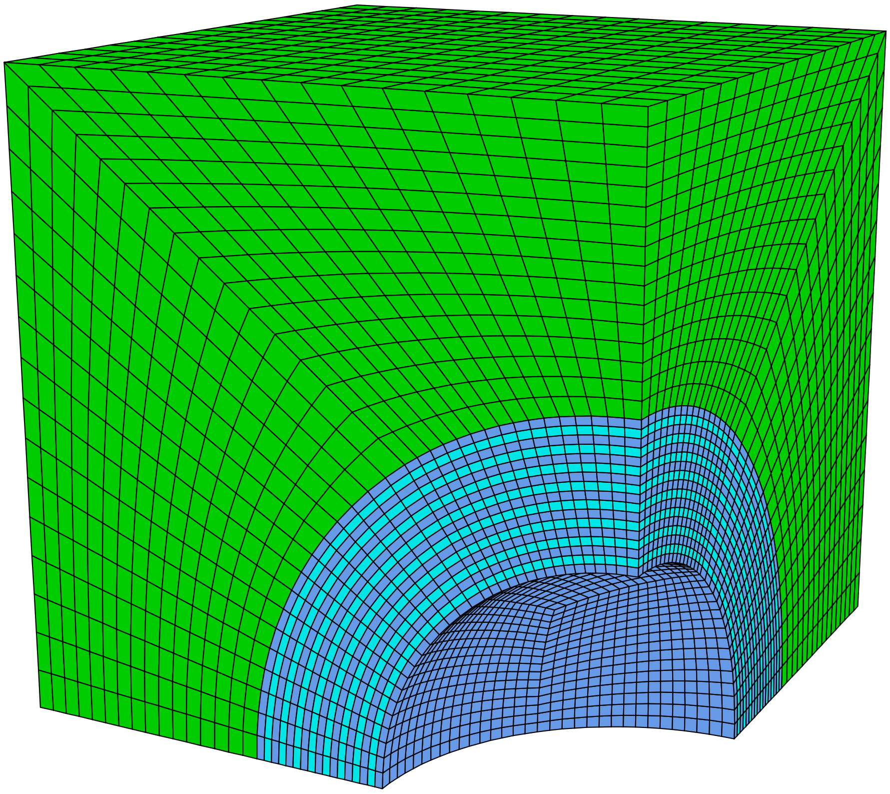


Software Architecture



Numerical Example - Set up

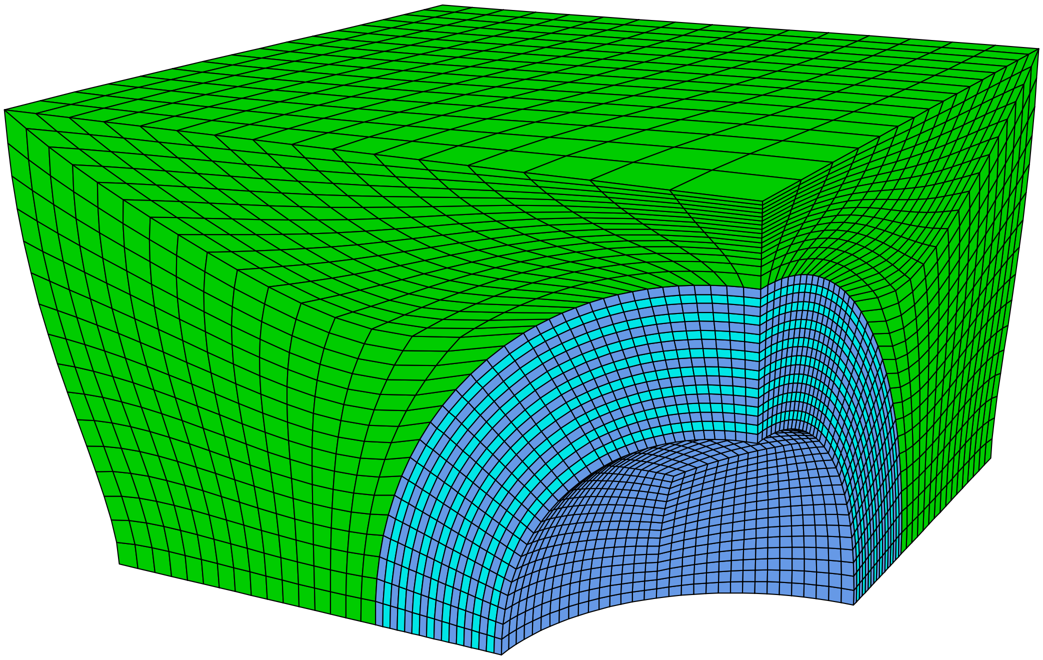
- **Parameterized mesh: 80 K to 39.2 M dof**
- **$\sim$ 40,000 dof/proc.: Cray T3E, IBM cluster of SMPs**
- **Stiff sphere: 17 thin stiff (steel) and soft (rubber) layers, embedded in soft material**



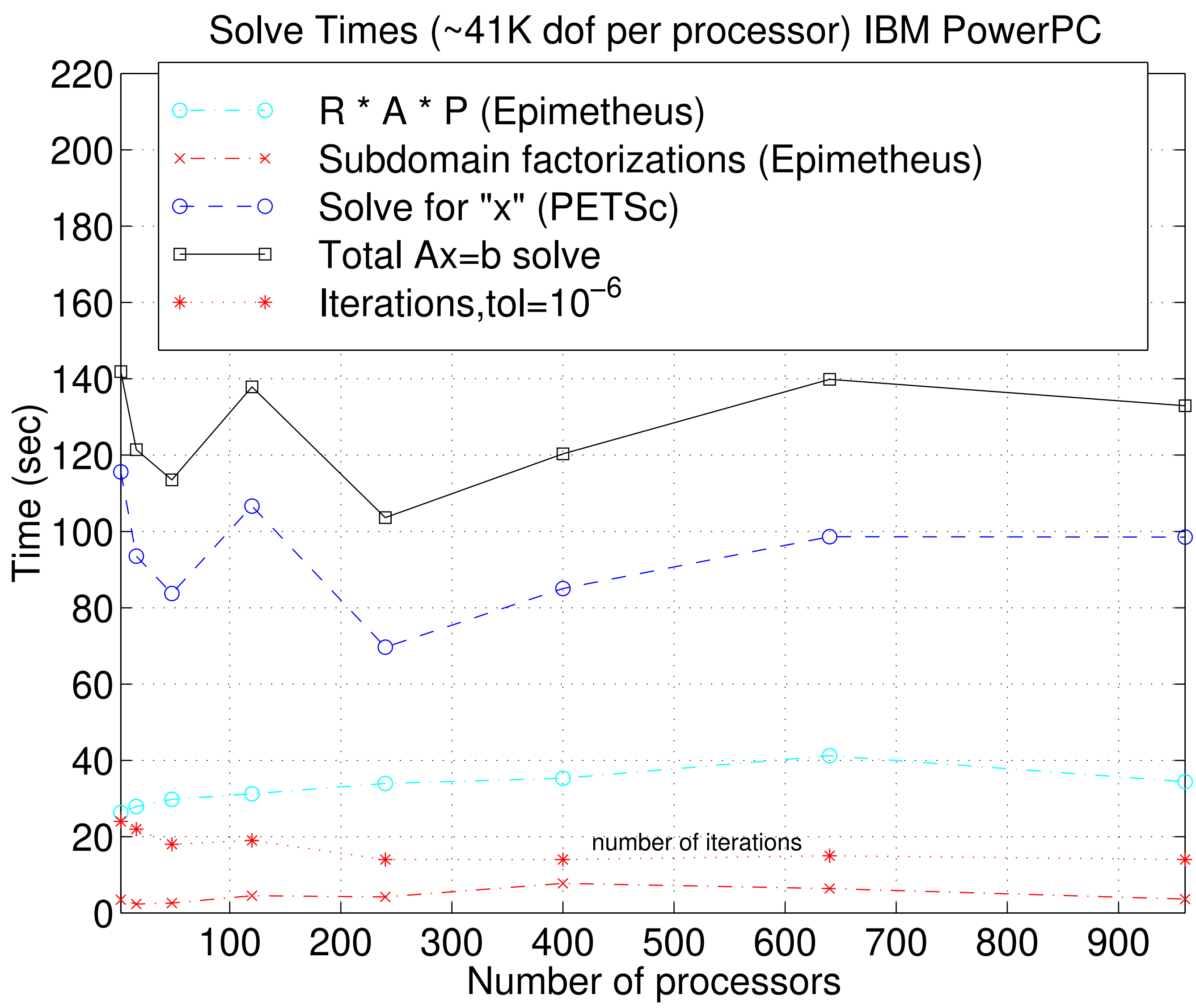
- **One octant modeled with symmetry boundary conditions and imposed displacement (crushing) boundary conditions**
- **Soft material: finite deformation, mixed**
  $E_{soft} = 10^{-4} E_{hard}$, $\nu = 0.49$
- **Stiff material: plasticity w/ kinematic hardening**
  $E_h = 30000$, $\sigma_y = 30 = 0.001 \cdot E_h$, $\nu = 0.3$

Numerical Example - Results

- **Compute deformation after crushing object**
- **Nonlinear - use Newton (FEAP)**
- **Linear solver: CG with multigrid preconditioner**
- **CG smoother: Block Jacobi preconditioner**



Numerical Example - Performance on IBM PowerPC Cluster



N. dof	80 K	622 K	2 M	5 K	10 K	16 K	26 M	39 M
N. Procs.	2	16	52	120	240	400	640	960
Iterations	24	22	18	19	14	14	15	14

Conclusions and Future Work on Multigrid

- Overall FE System called Athena
  - Fully nonlinear
- Multigrid solver called Prometheus
  - Requires only fine grid matrix and mesh
  - Everything else automatic
  - Available at `www.cs.berkeley.edu/~madams/prometheus`
  - Runs on Cray T3E, IBM Clusters, Berkeley NOW, Berkeley Millennium
- Future Work
  - More test cases, more users
  - Harden implementation
  - Add different coarsening algorithms
    * a la Vanek/Mandel/Brezina
    * Geometric alternatives
  - More elements (shells, beams), higher order
  - Indefinite systems from large deformation elasticity and plasticity

Overall Conclusions

- **Numerical Linear Algebra major activity in applied mathematics**
- **Uses tools from Applied Math, Pure Math, CS**
- **Constant stream of new application demands new algorithms**
- **Good algorithms store and manipulate only the data they really need, and throw away the rest**
- `www.cs.berkeley.edu/~demmel/MSRI99talk.ps`