The Parallel Performance of a Tightly Coupled 3D Magnetohydrodynamic Simulation

Lee Margetts, Mike Pettipher

Manchester Computing

THE UNIVERSITY of MANCHESTER

Lee.margetts@man.ac.uk

m.a.pettipher@man.ac.uk

Contents

- Overview
  - What is magnetohydrodynamics?
  - Solution strategy
  - The parallel implementation
- How cache use affects performance
  - Poor performance observations
  - The fix
- Convergence variability
  - How roundoff affects convergence
  - Reliability of the results
- Two case studies – 4 million unknowns on 256 processorss
  - A Navier Stokes case study
  - A magnetohydroynamics case study

Magnetohydrodynamics

- 3D Navier Stokes
  - Steady state
  - Primitive variables
  - Viscous incompressible fluid
- 3D Magnetohydrodynamics
  - Steady state
  - Primitive variables
  - Viscous incompressible fluid

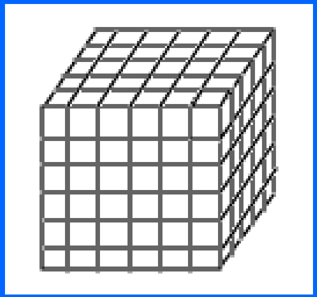
Electrically conducting fluid
Applied magnetic field

- Typical methods
  - Decouple the physics
  - Use different solvers – difficult to parallelise!
-
- Fully coupled, direct numerical solution (DNS)
  - DNS - Benchmark solutions for simpler algorithms

Solution Strategy

Discretisaton using the finite element method

- 20 node quadrilateral bricks



- MHD Degrees of Freedom – 60v 60B 8P

Solution carried out element by element

- Iterative solution algorithm BiCGStab(l)
- BiCGStab(l) necessary to deal with unsymmetric stiffness matrix

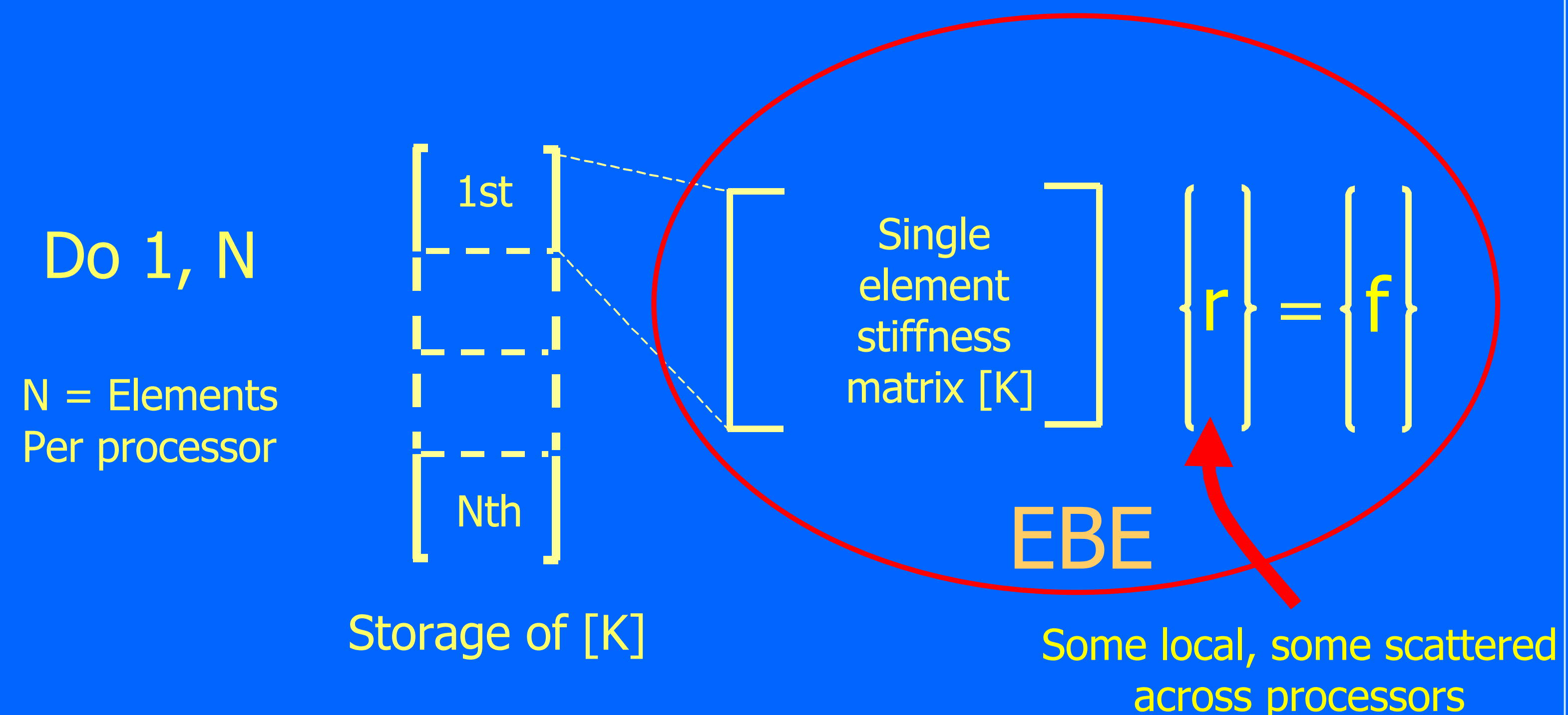
Leads naturally to a parallel implementation

- Simple distribution of elements across processors
- Equal number of equations per processor

Essence of the Parallel Implementation

For its set of elements, each processor

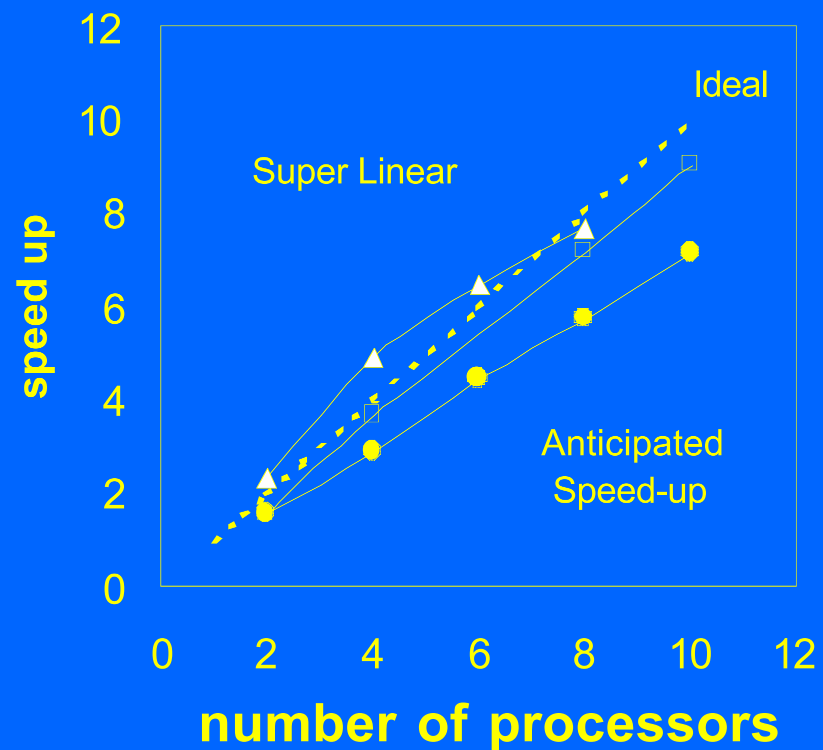
- Gathers the equations it needs (communication)
- Performs, EBE, matrix vector computations (Kr=f)
- Scatters the results (communication)



Preliminary Performance Findings 1

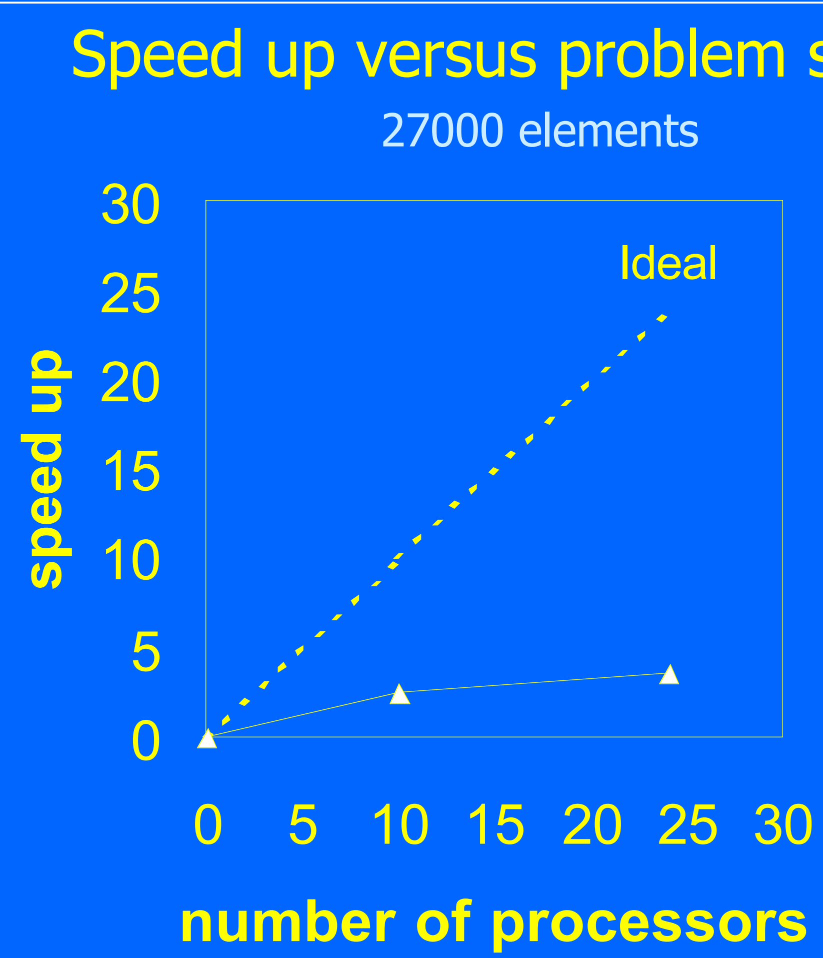
Speed up versus problem size

216, 1000, 1728 elements



Performance decreases as problem size increases

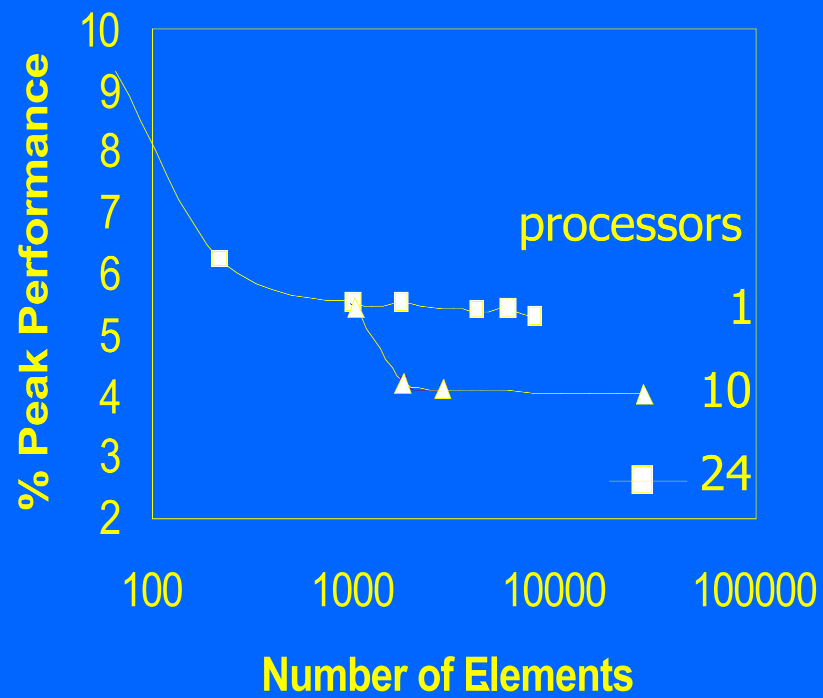
Preliminary Performance Findings 2



...and decreases

Preliminary Performance Findings 3

- %Peak performance also decreases with increasing problem size

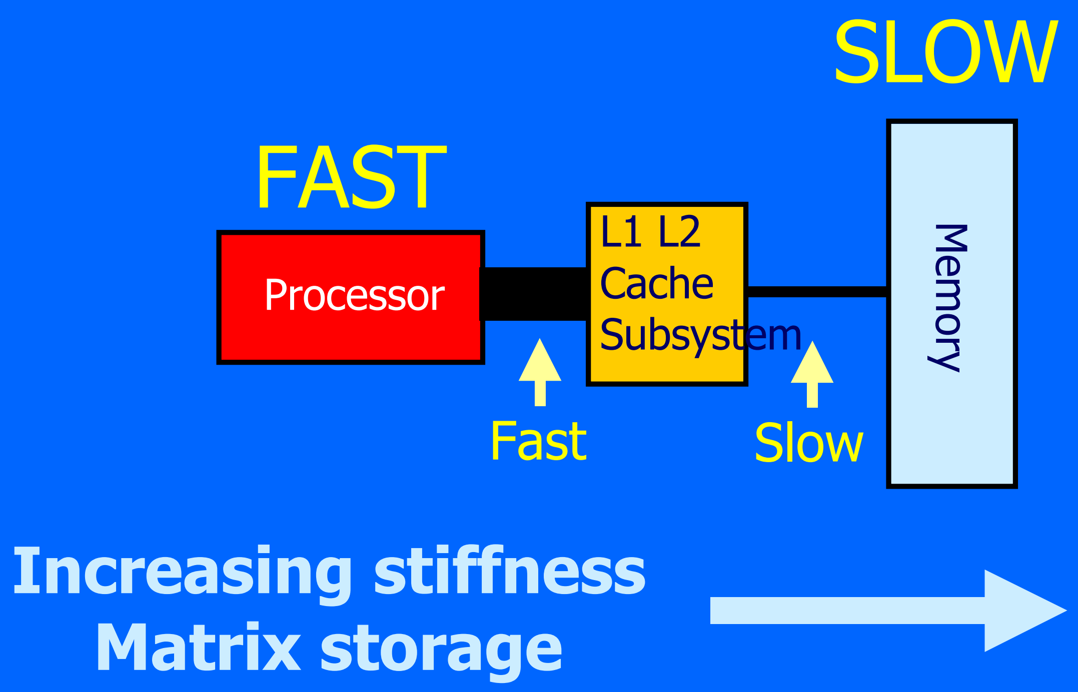


- And decreases with more processors
  - Single processor ~ 6% - 24 processors ~ 2-3%

Why is the performance poor?

It turns out the reason is mainly inefficient cache use

- Due to the storage required for N element stiffness matrices [K]



Which is made worse when the memory is shared with other users (Origin2000)

Sparse Element Stiffness Matrix

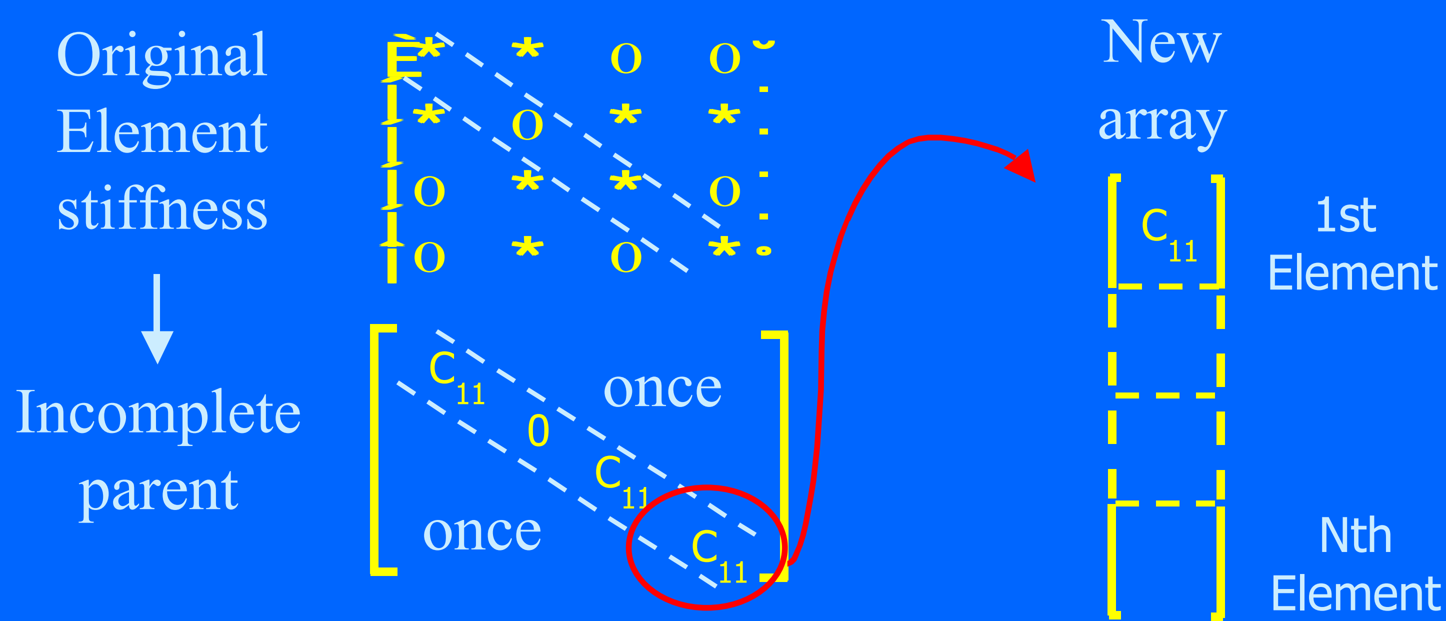
Naver Stokes Coupling terms

$$
\begin{bmatrix}
C_{11} & C_{12} & 0 & 0 & C_{15} & C_{16} & C_{17} \\
C_{21} & 0 & C_{23} & C_{24} & 0 & 0 & 0 \\
0 & C_{32} & C_{11} & 0 & C_{35} & C_{36} & C_{37} \\
0 & C_{42} & 0 & C_{11} & C_{45} & C_{46} & C_{47} \\
C_{51} & 0 & 0 & 0 & C_{55} & 0 & 0 \\
0 & 0 & C_{51} & 0 & 0 & C_{55} & 0 \\
0 & 0 & 0 & C_{51} & 0 & 0 & C_{55}
\end{bmatrix}
\begin{Bmatrix} u \\ p \\ v \\ w \\ B_x \\ B_y \\ B_z \end{Bmatrix}
=
\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}
$$

Coupling terms Magnetic diffusion

Reducing Element Stiffness Storage 1

Consider the Navier-Stokes part



For elements of identical shape and material property – duplication

- Store C_{11} once for each element
- Originally ~200 elements, now ~8000 elements fit in cache

Reducing Element Stiffness Storage 2

Consider the full Magnetohydrodyamics stiffness matrix

- There are 13 unique submatrices for each element
- Each submatrix has 400 entries
- Storage still a problem even if each element is identical

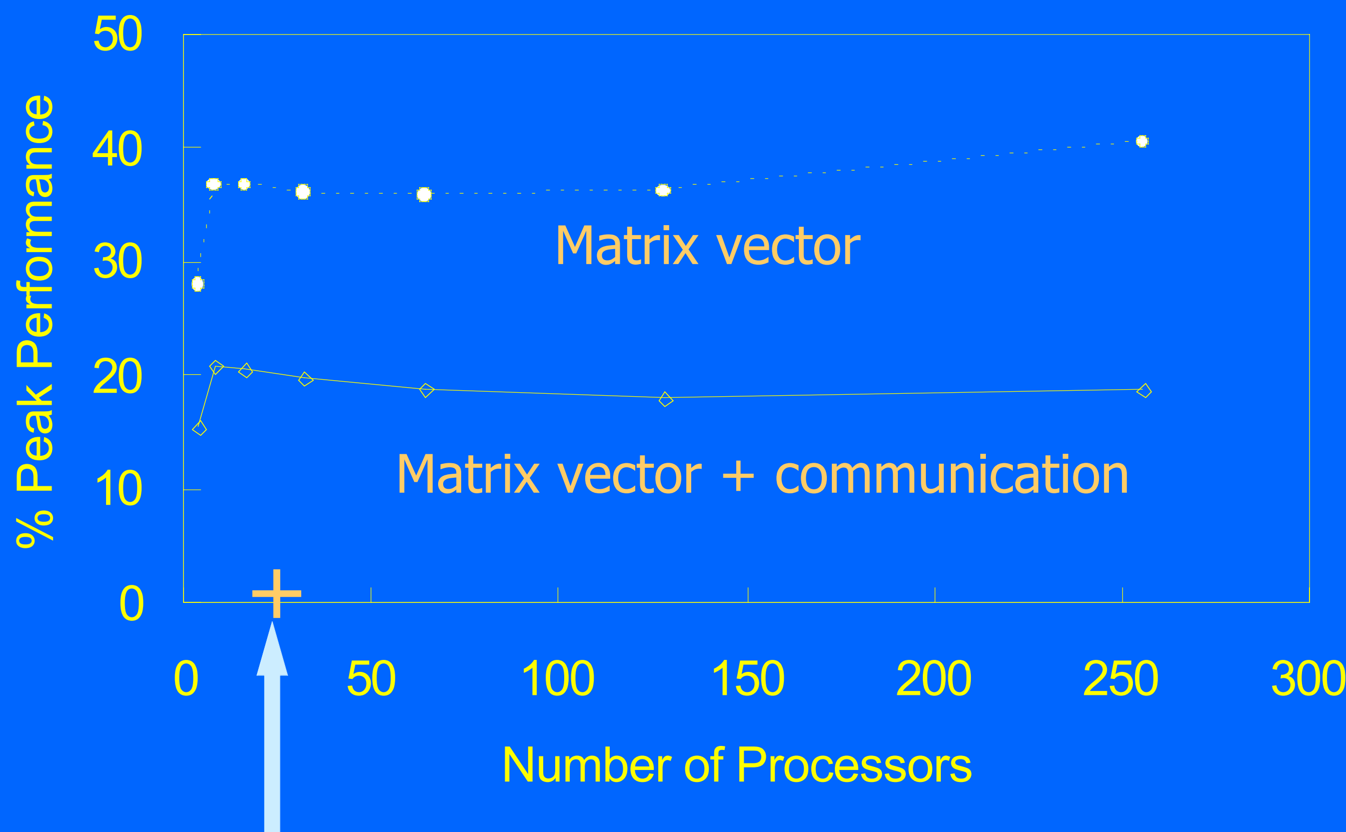
Break up the element matrix vector computation, replacing

```
do iel=1,nels_pp
   u=matmul(ke,x)
end do
```

→

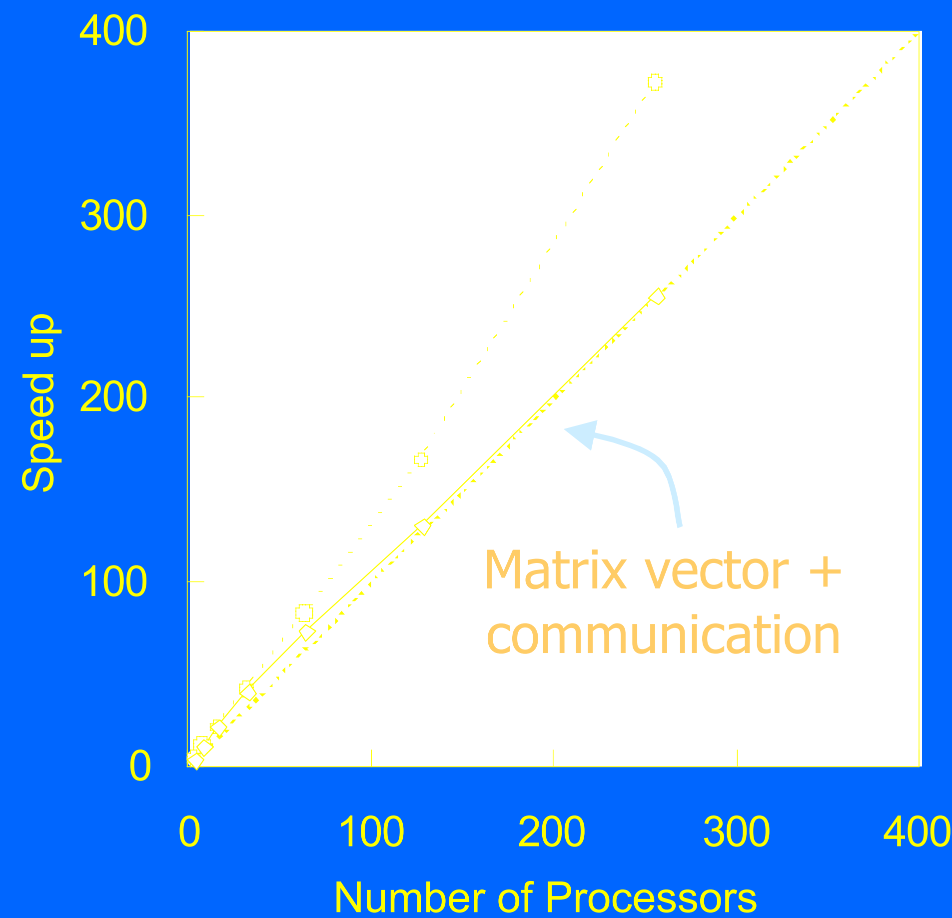
```
do iel=1,nels_pp
   u ′= matmul ( C11 , x ′ )
end do
do iel=1,nels_pp
   u ′= matmul ( C55 , x ′ )
end do
do iel=1,nels_pp
   u ′= matmul ( C15 , x ′ )
end do
```

Percentage Peak Performance



Previously ~2% 24 Processors
~300,000 unknowns

Speed up



Prediction:

10 day analysis will take less than 1 hour with 256 processors

Convergence Variability

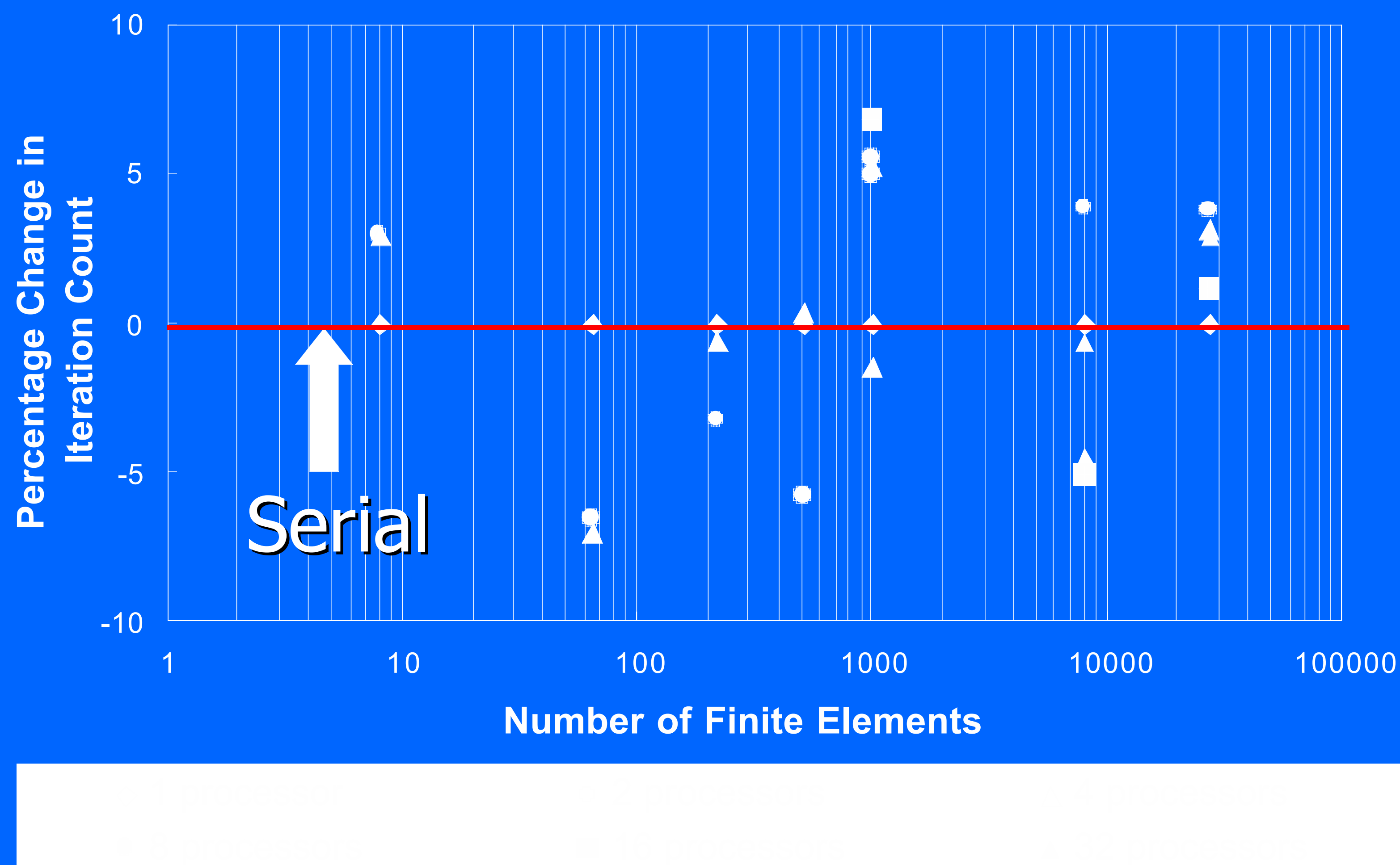
What does it mean?

- The number of iterations to a converged solution changes depending on whether the problem is solved in serial or parallel
- It also depends on the number of processors
- It can also varies from one run to the next (not commonly)

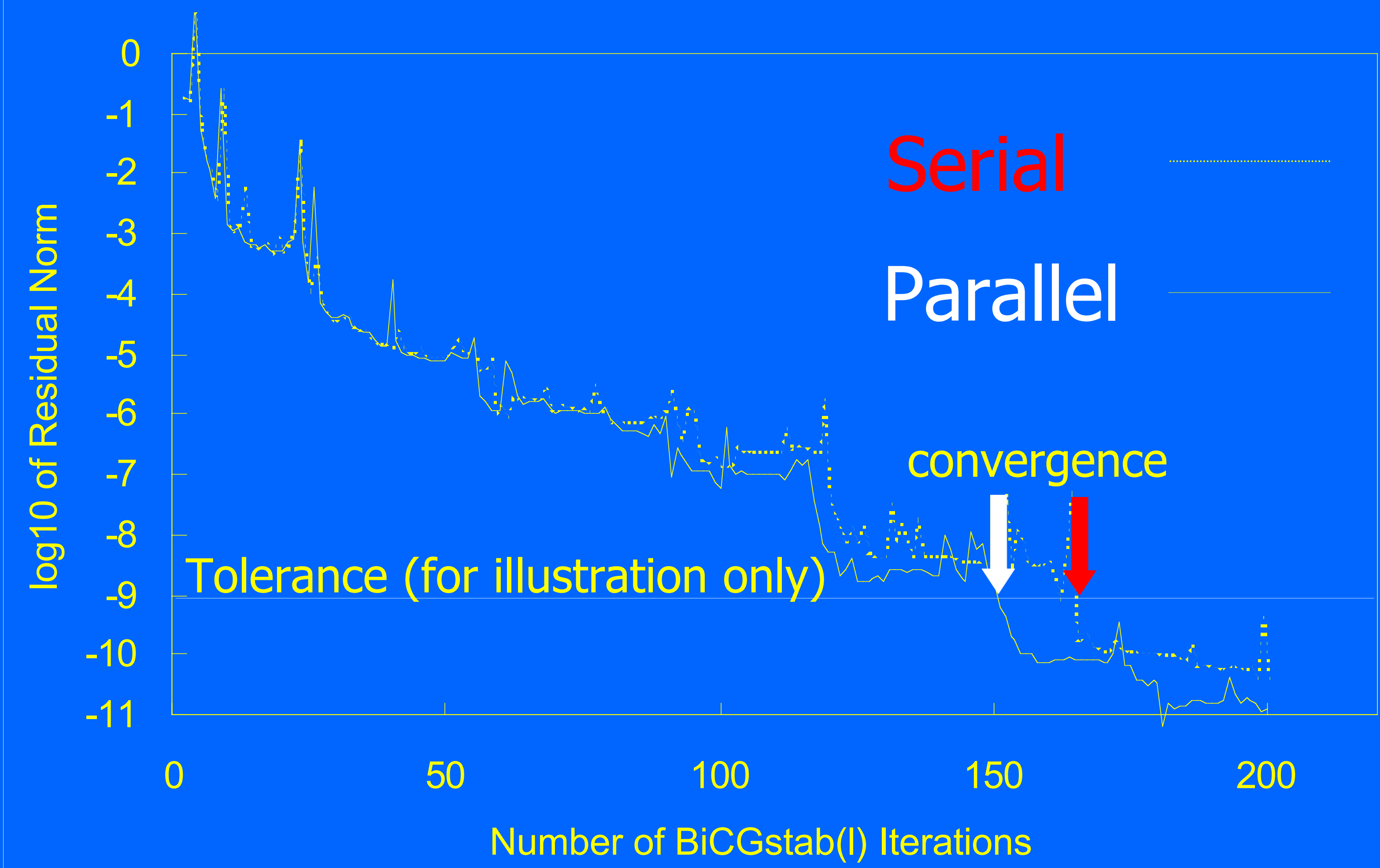
Reasons why it is disturbing?

- The algorithm has not changed
  - Only some details that seem unimportant!
- How do you check that the parallel program is working properly?
- Does this mean the answer is wrong?

Convergence Variability



A Typical Convergence History



Why is there variability?

Has the solution method changed? - No

- No - important
- The parallel algorithm is essentially the same as the serial
- Other parallel methods change the algorithm (Schwartz)
- In these, variation in iteration count is common

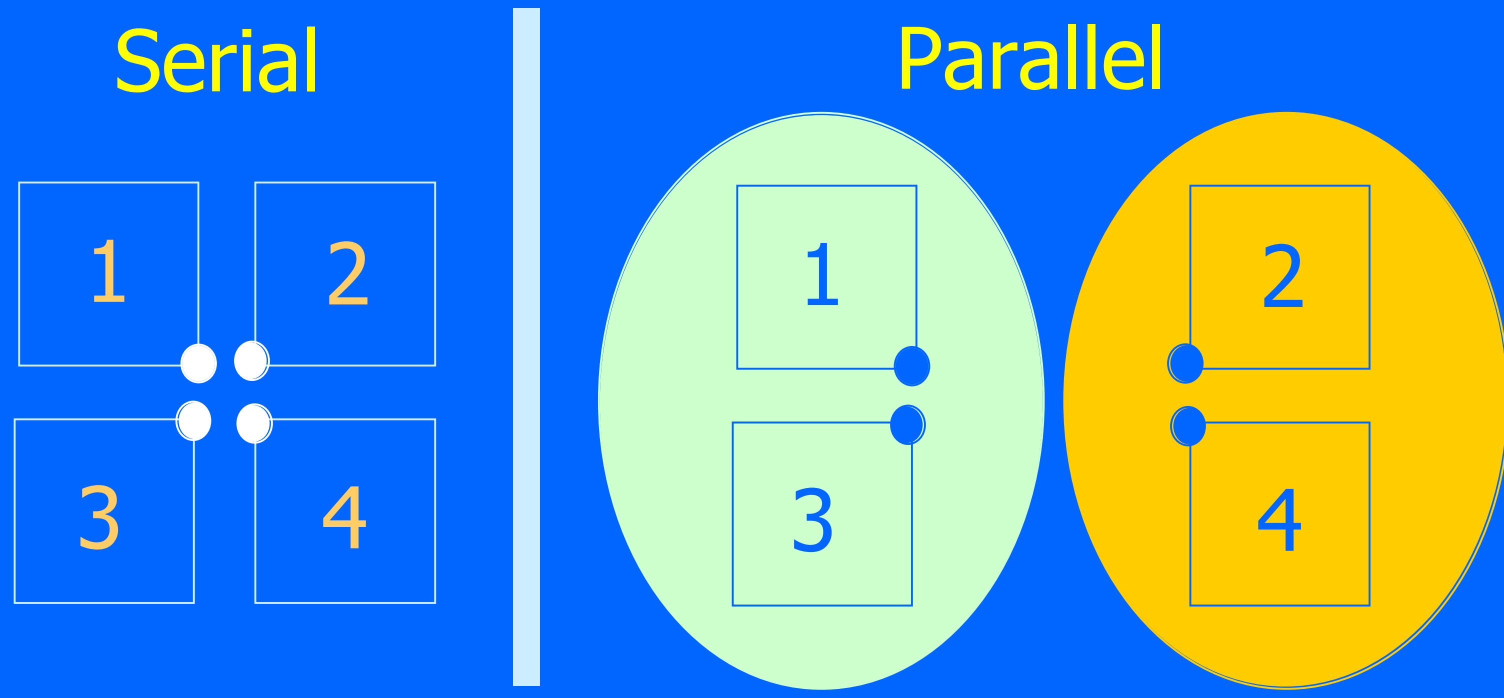
Is there a bug in the program? - No

- Parallel programs are notoriously difficult to debug
- There's no bug and the problem seems to be roundoff
- Even though we are using double precision!

Prove it.

- How?
- Simulate the parallel program with a serial one.
- Hand code the 'messages'

What's the difference? Summation Order



1+2+3+4 /= 1+3 + 2+4

Subtotals

...worse if 1>>3 etc

Convergence Variability

What happens to BiCGStab(l)?

- Over successive ‘external’ iterations
- The convergence paths diverge and converge
- The algorithm is self correcting, by design

Are the results reliable?

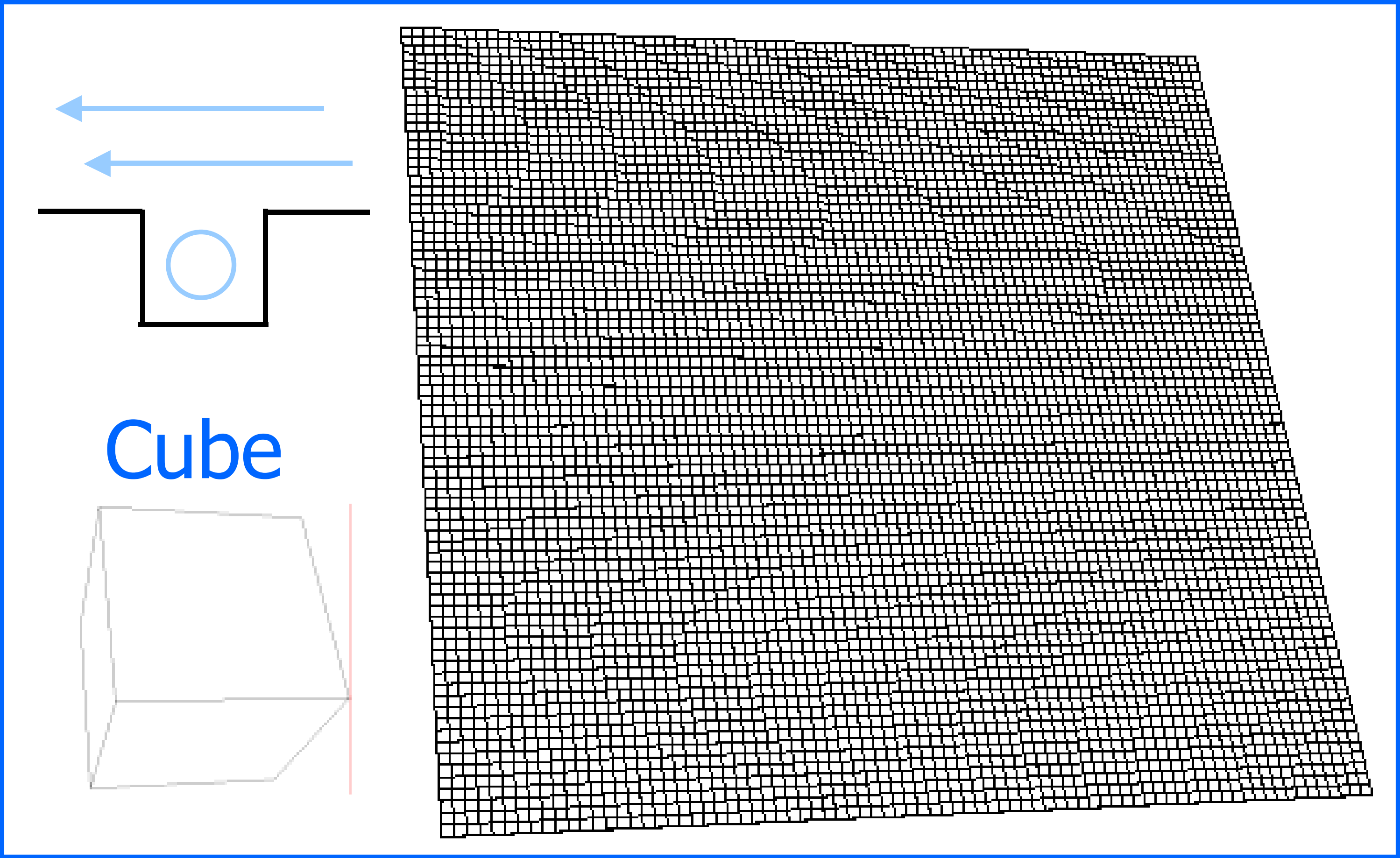
- Computation is terminated by a tolerance test
- Experience – results always agree within set tolerance
- Will they always agree? – mathematics problem!

What about other iterative algorithms?

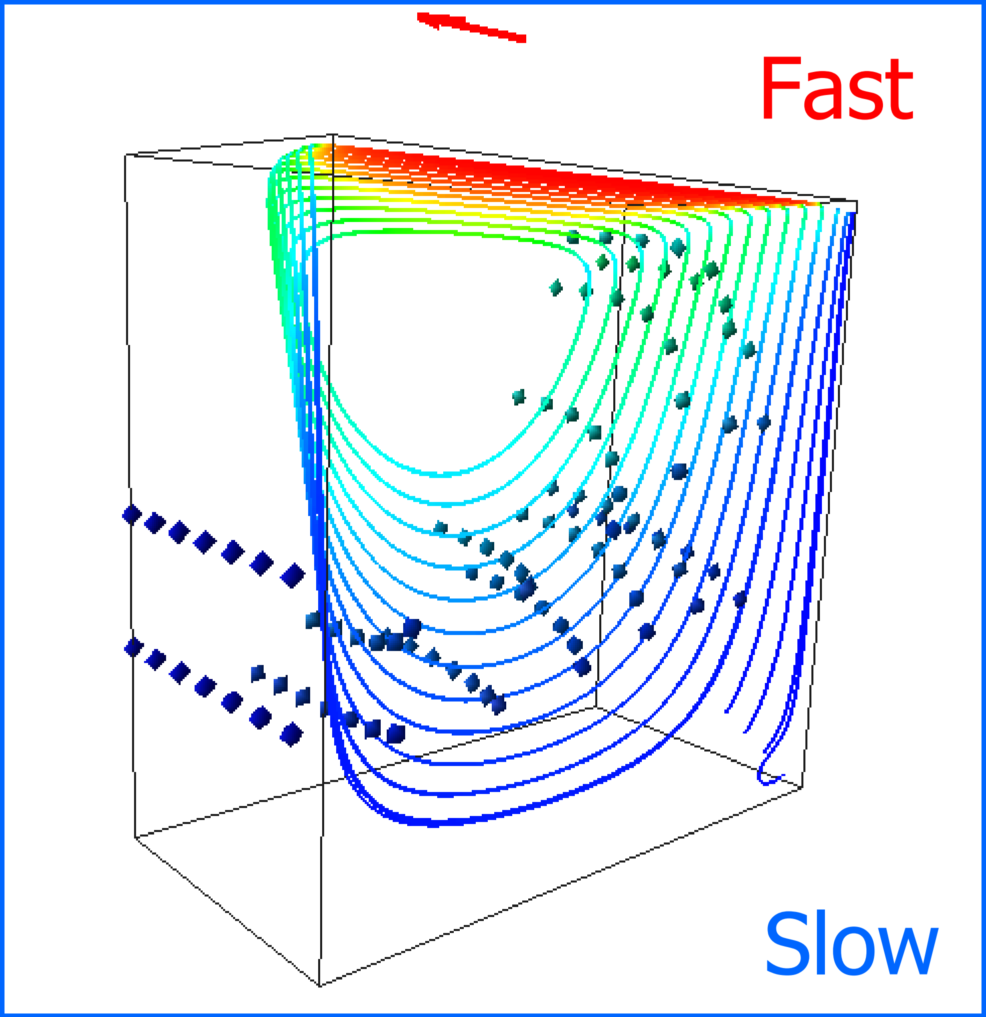
- Other experience with PCG
- May not be a problem for positive definite matrices
- Some variability with BIOT consolidation

Navier Stokes Test Problem

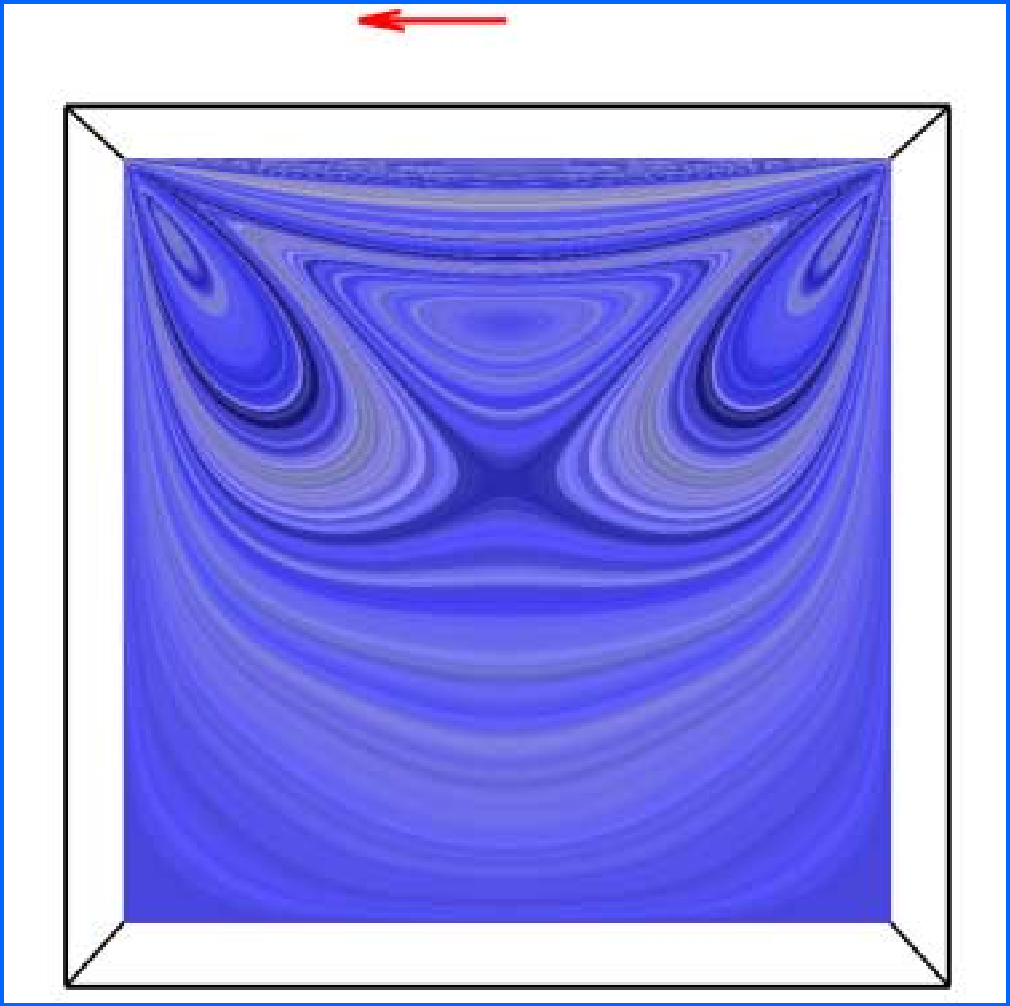
3D Cubic lid-driven cavity - 4 million equations



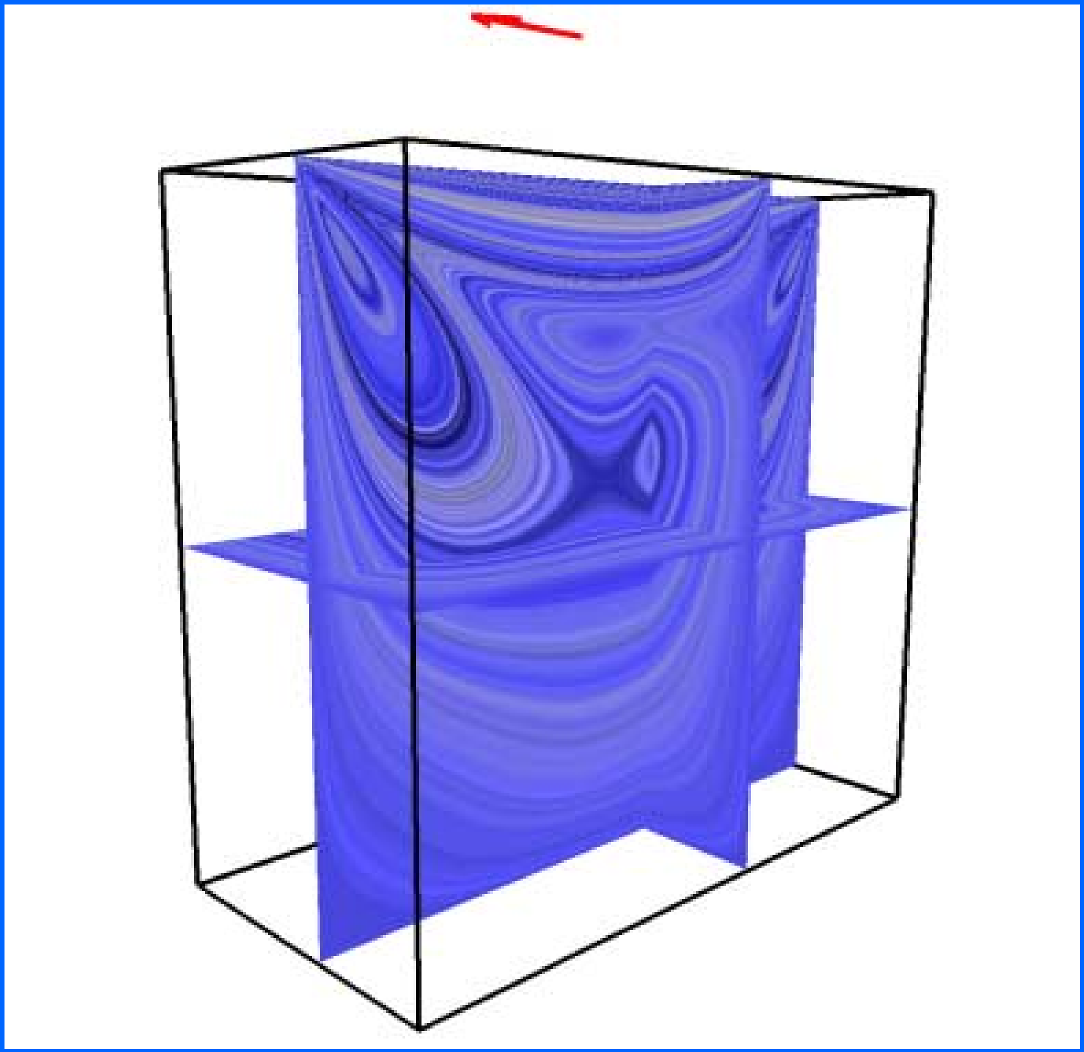
Streamlines



Velocity Magnitude 1



Velocity Magnitude 2



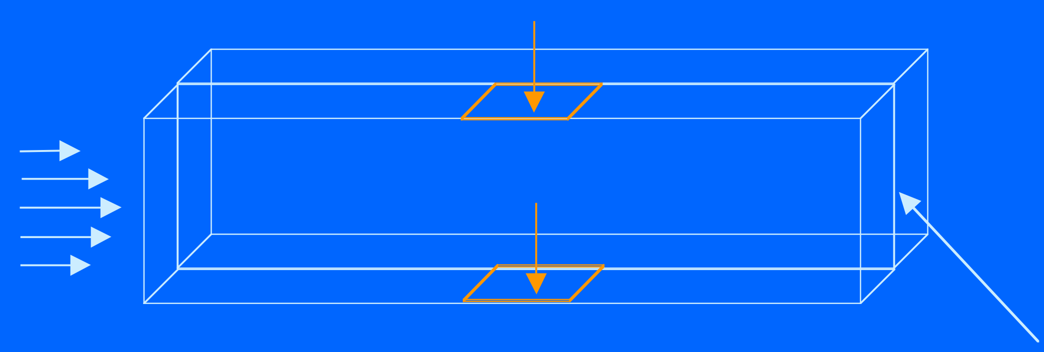
Performance vs Reynolds Number

Reynolds Number	Parallel Time Minutes	Serial Time Days	%Peak Perf.	Gflops
10	20	2-3	23	47
100	47	8-9	29	59
1000	180	>1 month	29	59

MHD Test Problem

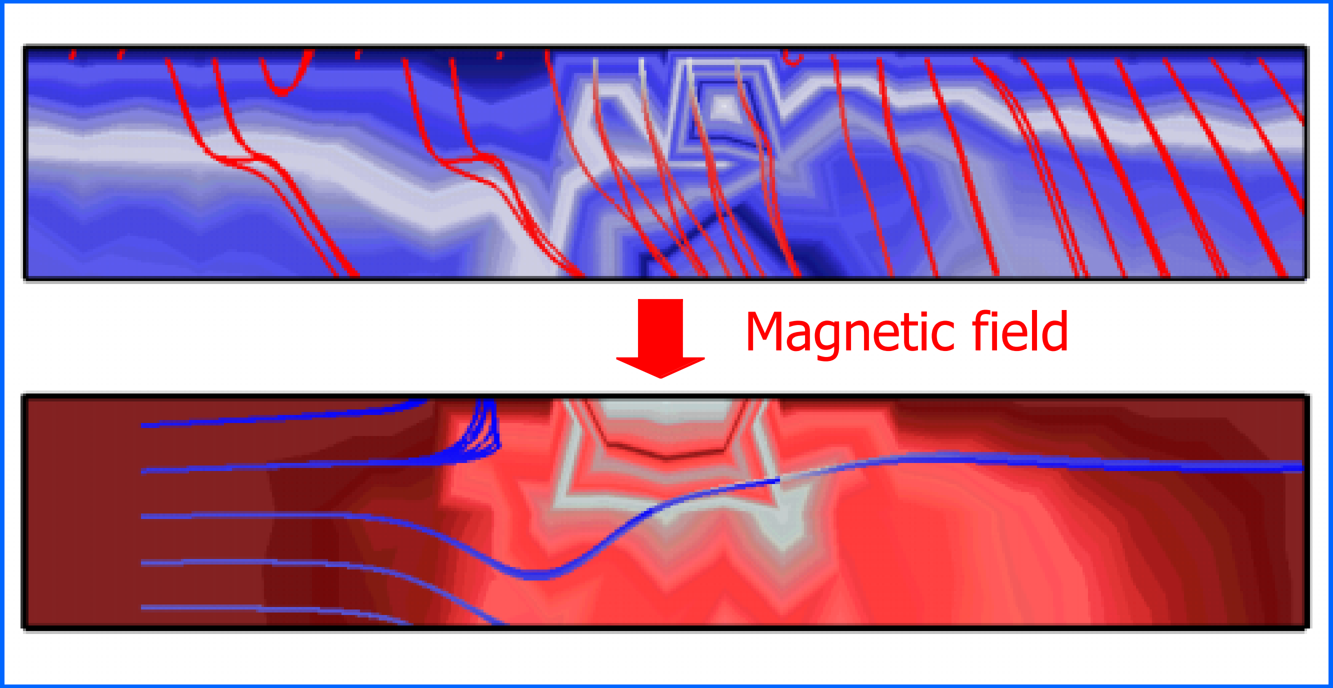
Flow through a rectangular duct

- Electrically insulating walls
- Uniform pressure gradient
- Externally applied magnetic field
- Only a small example ~ 4,000 equations



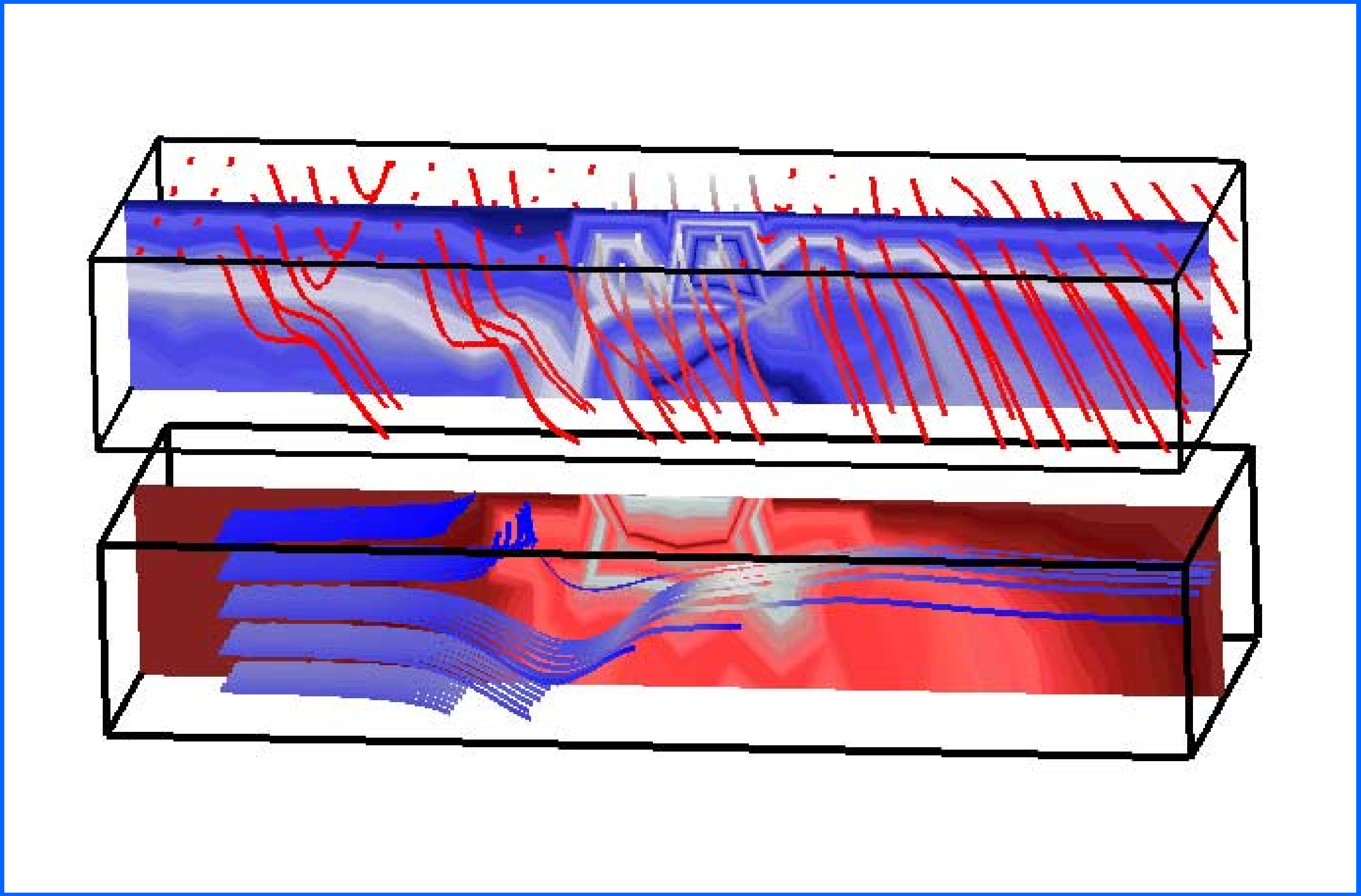
MHD Duct Flow 1

The fluid drags the magnetic field



The magnetic field distorts the fluid flow

MHD Duct Flow 2



Summary

Performance

- Modern parallel architectures – cache dominated
- Coupling physics has a storage penalty
- Scaling to large numbers of processors

Roundoff and convergence variability

- A ‘forgotten’ issue returns with parallel computation
- EBE method still gives the correct answer
- How trustworthy are other parallel algorithms?

Future work?

- Should be able to add ‘more physics’
- Mesh generation + visualisaton, perhaps now the bottleneck!

Acknowledgements

Thanks in particular go to

- Joanna Leng (MVC) for her help in visualising the huge datasets.
- Ian Smith – PhD supervisor.
- Current and previous MVC/CSAR staff for their contributions.
- EPSRC (UK research council) for funding the research.

The Parallel Performance of a Tightly Coupled 3D Magnetohydrodynamic Simulation

Lee Margetts, Mike Pettipher

Manchester Computing

THE UNIVERSITY of MANCHESTER

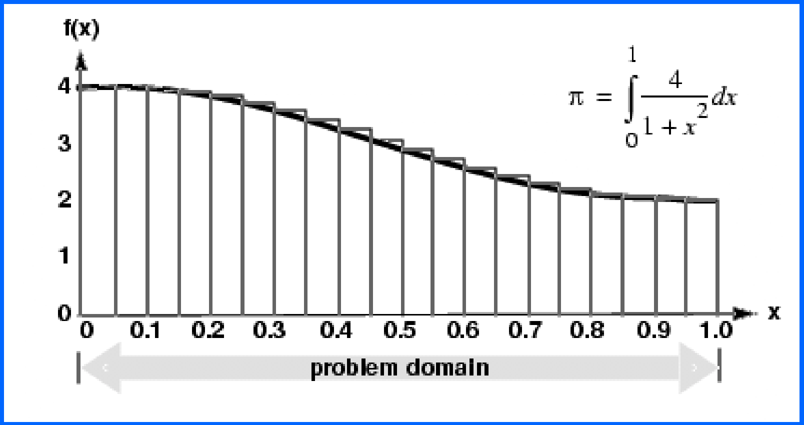
Lee.margetts@man.ac.uk

m.a.pettipher@man.ac.uk

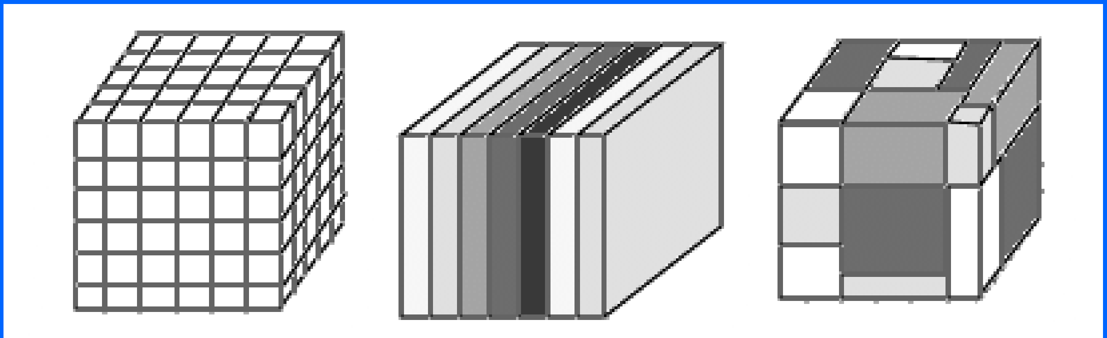
Domain Decomposition

Examples

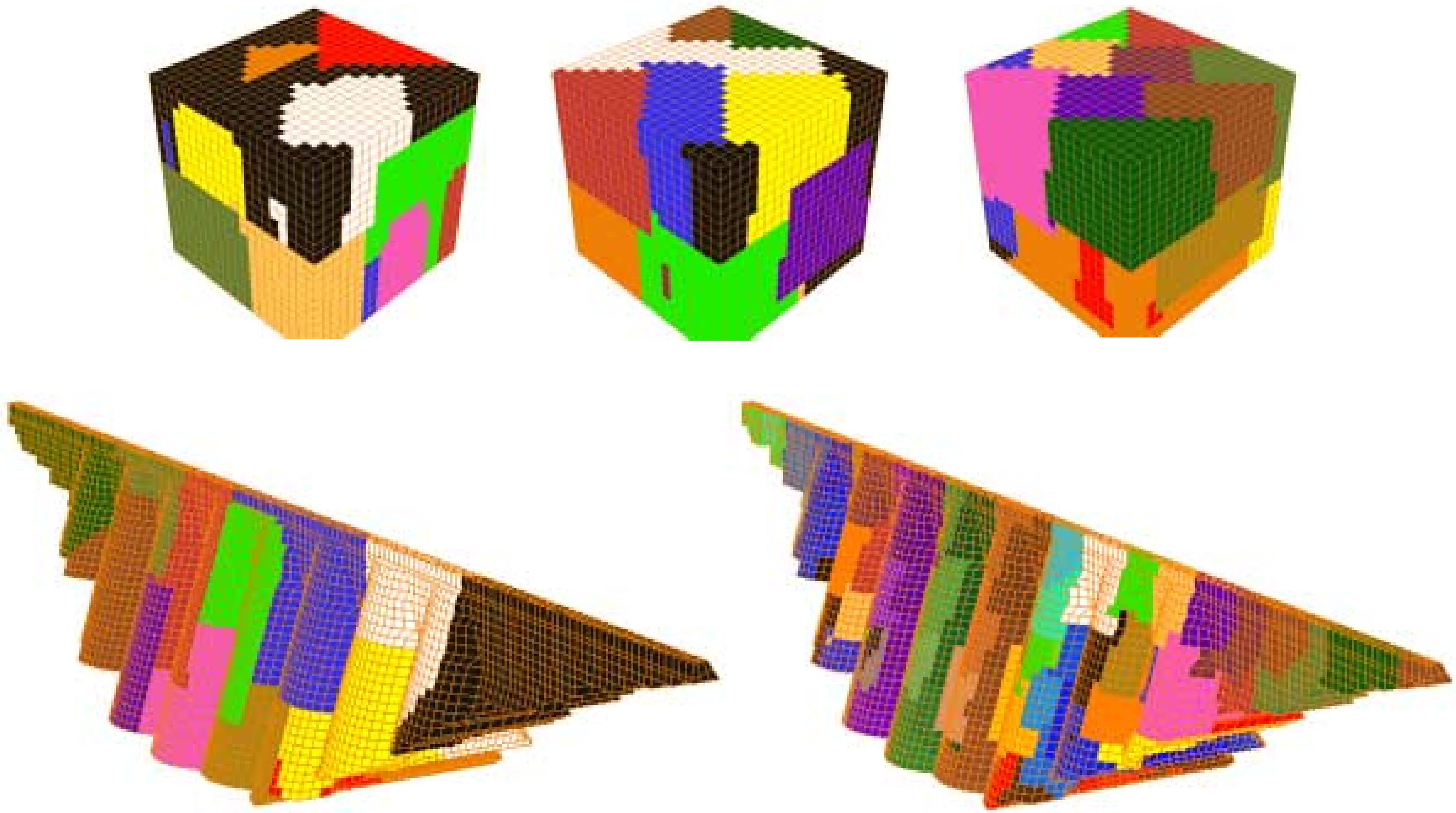
- Calculation of the area under a curve by numerical integration



- 3D grid problem



The 'Greedy' Algorithm



Research Objectives

Parallelisation of a suite of finite element programs

- Multi-purpose
  - Structural — Building design
  - Geomechanics — Tunneling/foundation design
  - Fluid Mechanics — Contaminant transport
  - Forced Vibrations — Earthquake engineering
  - Fully coupled "Multiphysics" — Magnetohydrodynamics
- General approach based on MPI

Virtual prototyping

- Real time interactive finite element analysis

Teaching

- Encouraging non-specialists to use parallel computers

