Multigrid Improvements to CITCOM

Sarfraz Ahmad Nadeem

NAG Ltd

24 September

Outline

- CITCOM dCSE
- CITCOM package
- CITCOM learning curve
- Sample applications
- Governing equations
- Test problems / results
- What next?
- Conclusion

CITCOM dCSE

- Proposed by University of Durham
  - Department of Earth Sciences
    - Dr Jeroen van Hunen as PI
  - School of Engineering
    - Dr Charles E Augarde as Co-Investigator
- CITCOM Package
  - Parallel finite element code
  - Written in C with MPI based parallelisation
- Original developers:
  - Louis Moresi (author of original 2D/3D finite element code)
  - Shijie Zhong (parallelised and added Multigrid solver)
  - PI’s contribution over a number of years

Project Breakdown

- 12 Months full time one person
- On 80% basis translates to 15 months
  - Started on 1st January 2008
  - To end on 31st March 2010
- Consists of 3 phases
  - Initial Project Study
    - **Until end of April 2009**
  - Multigrid Cycles
    - **Until end of September 2009**
  - Mesh Refinement
    - **Until end of March 2010**

CITCOM Characteristics

- Solves for
  - ☐ Stokes flow with large viscosity contrasts
  - ☐ Heat advection/diffusion
  - ☐ Pure advection of composition using a tracer method
  - ☐ Employs Cartesian coordinates system
  - ☐ In two & three dimension
- Relies on
  - ☐ Linear velocity and constant pressure shape functions
  - ☐ Full multigrid method for Stokes flow
  - ☐ Uzawa algorithm to apply incompressibility

Source Code

- In the main CITCOM package
  - 1 Makefile, 29 source code files and 7 header files
  - More than 25,000 source code line
- Some code for post processing
  - In five sub directories
    - 1 Makefile and 2 source code files in each sub directories
    - Header files are used from main CITCOM source
    - Calls to a number of *functions* from main CITCOM source
- Documentation
  - Some comments within code
  - Useful notes from PI

Learning Curve

- Due to limited documentation, following been the learning tools
  - Code browsing
    - To read/understand code itself and comments
  - Use of **Doxygen** (to generate documentation from source/comments)
    - "**Call**" and "**Call by**" graphs been of particular help
  - Use of **eTrace** package
    - It gives function call tree starting from "main()"
      - Good for serial code
      - Duplicates function calls for parallel code; one call for each process
  - Meetings with PI
  - Internet
    - Google
    - Altavista

Building Blocks

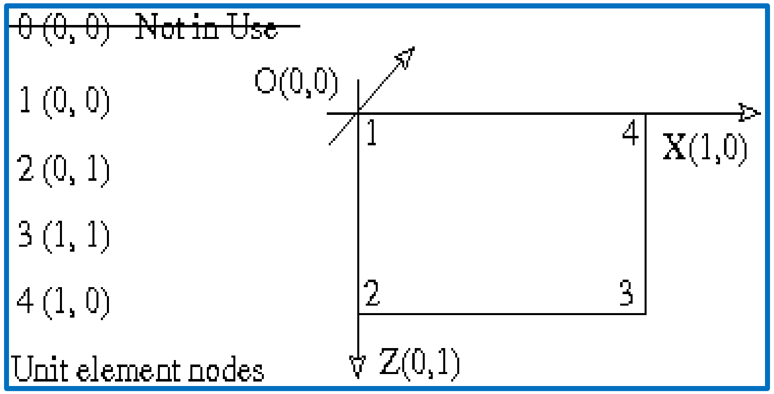
- Built on structured finite elements
  - Rectangular / Square elements in 2D
  - Brick / Cubic elements in 3D
- Z-axis is taken +ve in downward direction
- Although C code, zero locations in arrays are not used
  - Instead arrays been allocated an extra unit of memory
  - For most arrays, a couple of extra units of memory are allocated
- Most counter begins at 1 (one), not 0 (zero), e.g.
  - Local node numbering for each element starts at origin 1(0,0,0)
- Local node numbering for each element is counter clockwise

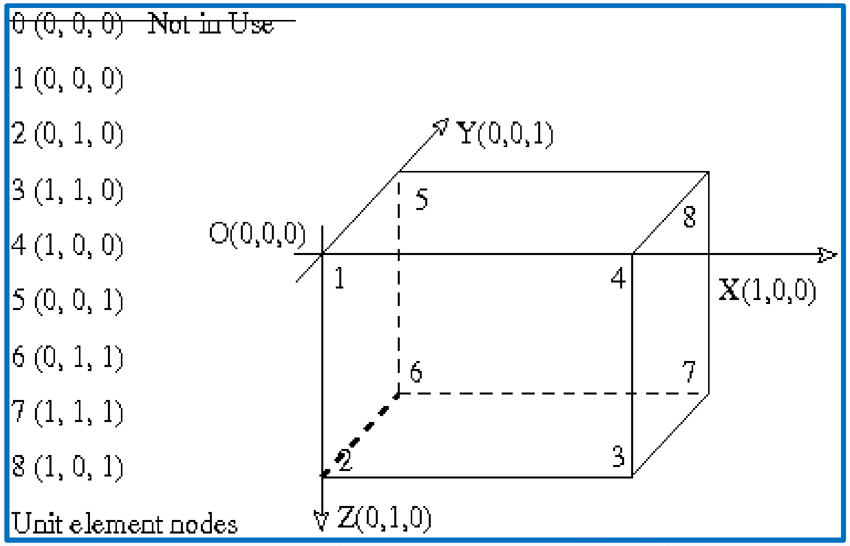
Mesh Elements in 2D / 3D

2D: Starting at origin, node numbering and orientation is counter clockwise

3D: Starting at origin, node numbering and orientation is counter clockwise spiral

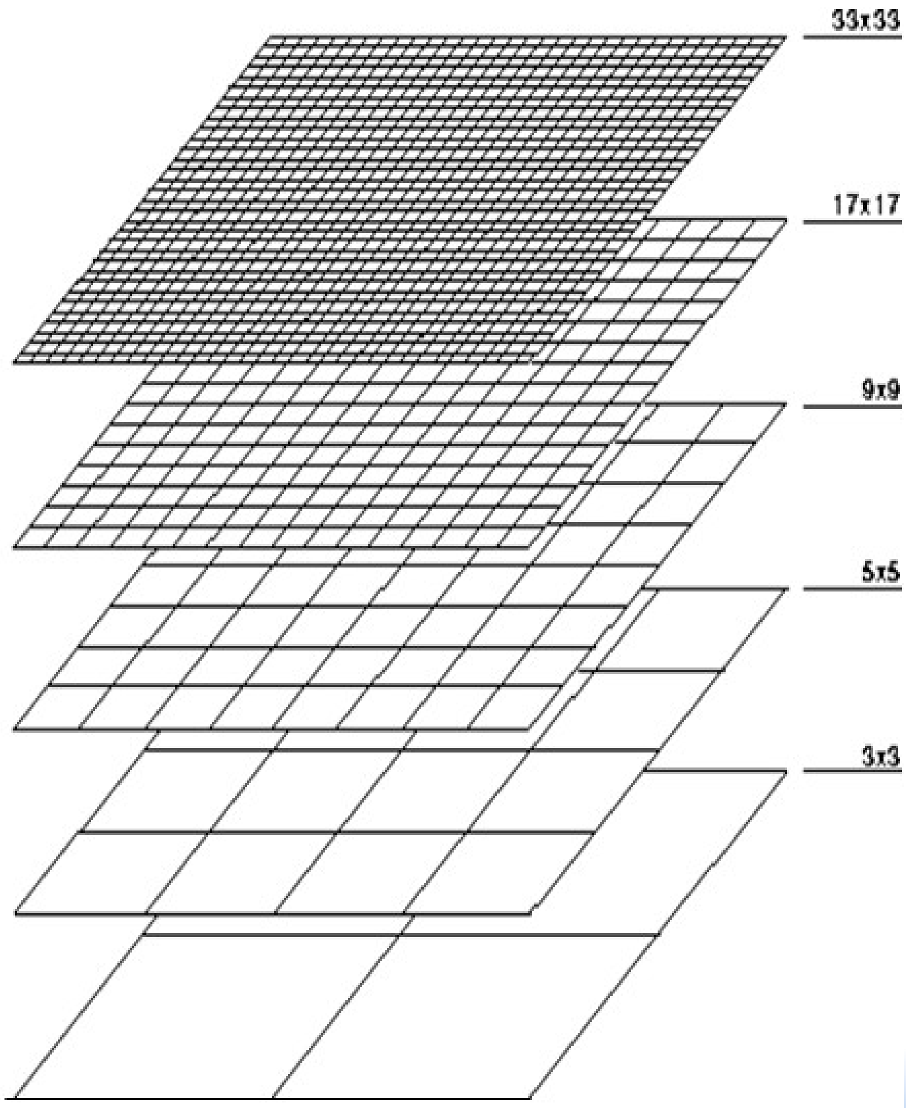
(front to back)





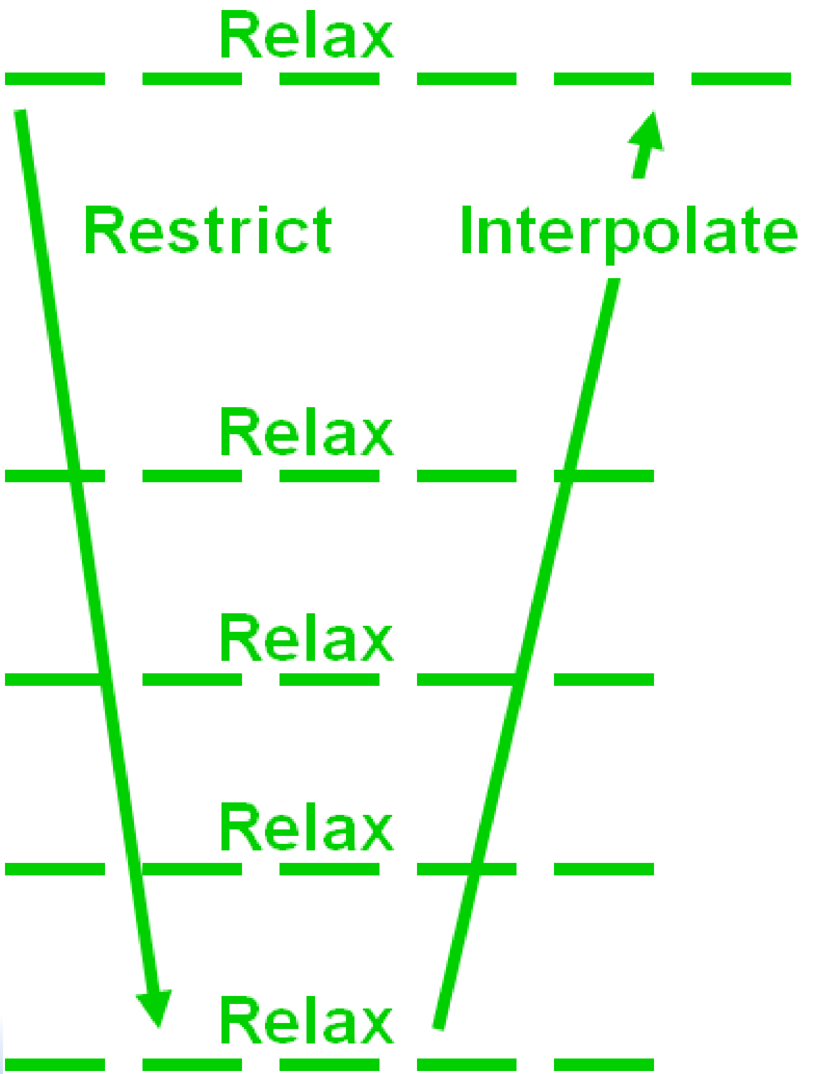
Multigrids

- Here 5 levels, each with different number of elements
  - Just 4 elements / 9 nodes at coarsest level
  - 1024 elements / 1089 nodes at finest level
- CITCOM allows up to 12 levels



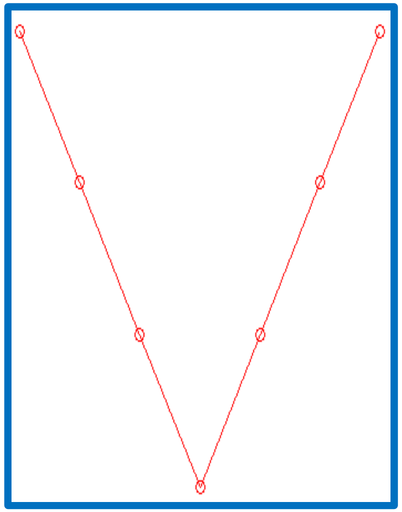
Multigrids Sudo Procedure

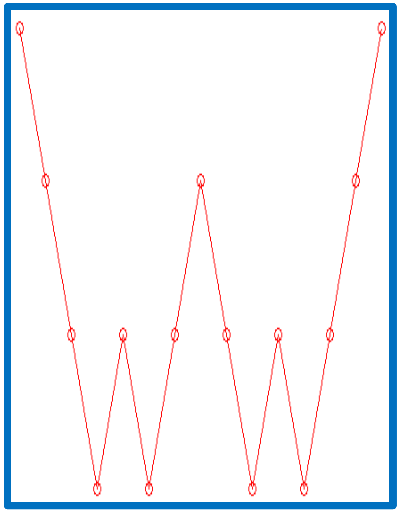
- Relax translate to an iterative solve
  - CG at coarsest level
  - GS everywhere else
- Restriction transforms vector to next coarse level
  - RHS, residual
- Prolongation (Interpolation) transform vector to next higher level
  - Velocity

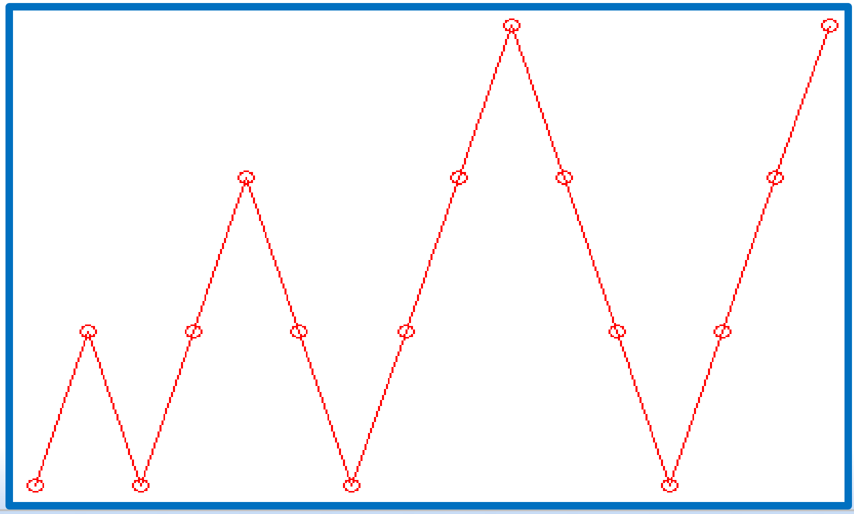


Popular Multigrid Schemes

- V-cycle, W-cycle and FMG(V) schemes
- Circles represents Smoothing/Correction / Relaxation
  - Iterative solve by CG / GS
- Lines represent Restriction/Prolongation(Interpolation)
  - RHS, residual
  - Velocity





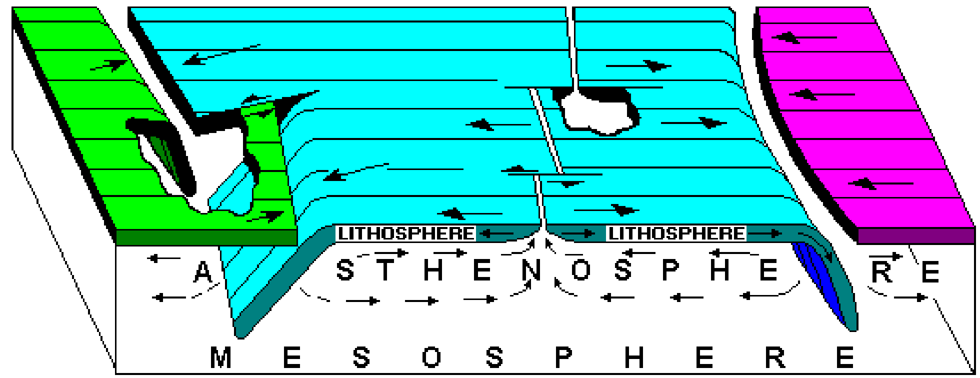


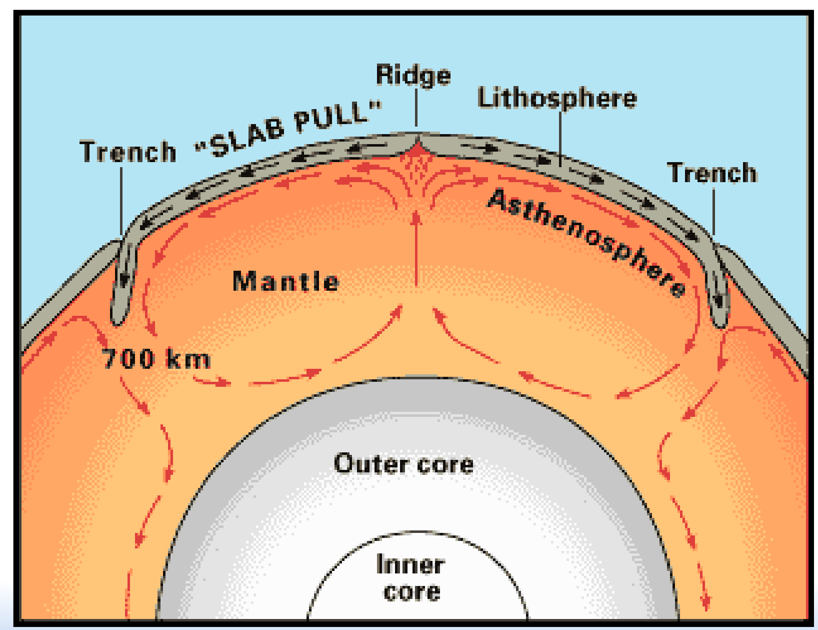
Multigrids Implemented in CITCOM

- Multigrid V-cycle & W-cycle schemes
  - These are most efficient schemes but may struggle in case of hard to solve problems
- FMG schemes (V- & W-cycles)
  - These schemes have the potential to overcome problems where V-cycle / W-cycle might fail
- V-cycles are efficient than W-cycles
  - In both of the above cases
- V- & W-cycles are efficient than corresponding FMG (V- & W-cycle) schemes respectively

General Applications

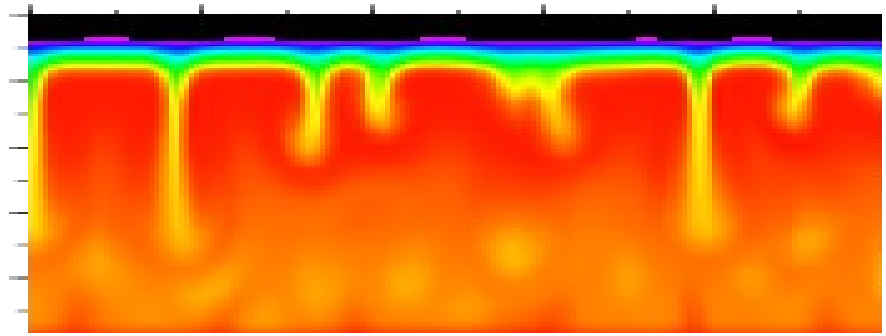
- A variety of dynamical problems related to the Earth's mantle and lithosphere:
  - ☐ Mantle convection
  - ☐ Subduction zones
  - ☐ Mantle plumes
  - ☐ Continental breakup
  - ☐ Thermal evolution of the Earth



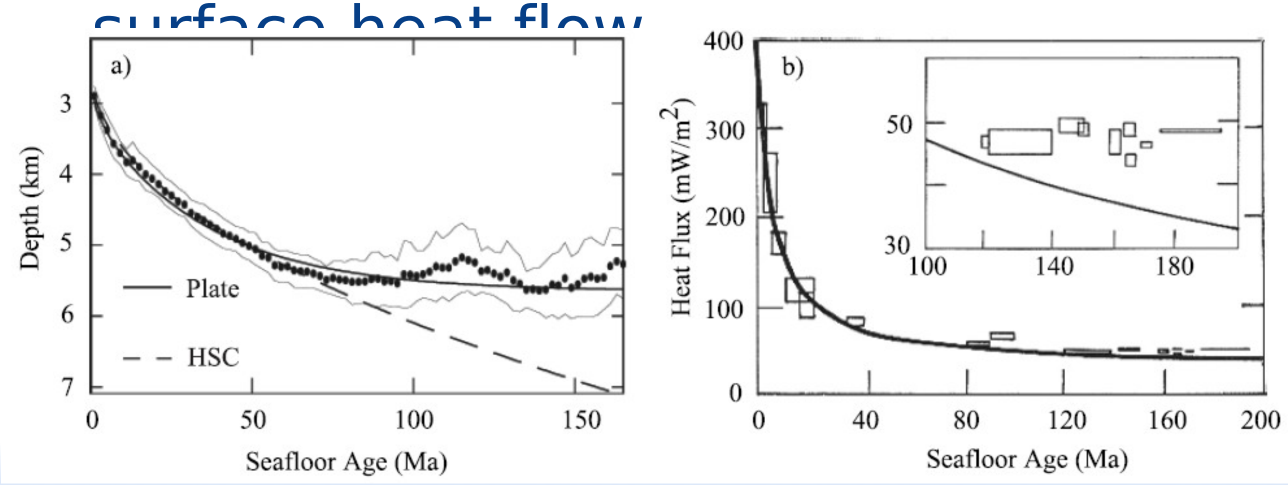


Lithospheric Thinning

- Oceanic lithosphere grows by conduction
- But at age > 70 M yrs, its base starts to ‘drip off’
- This might explain the observed flattening of the seafloor and surface heat flow



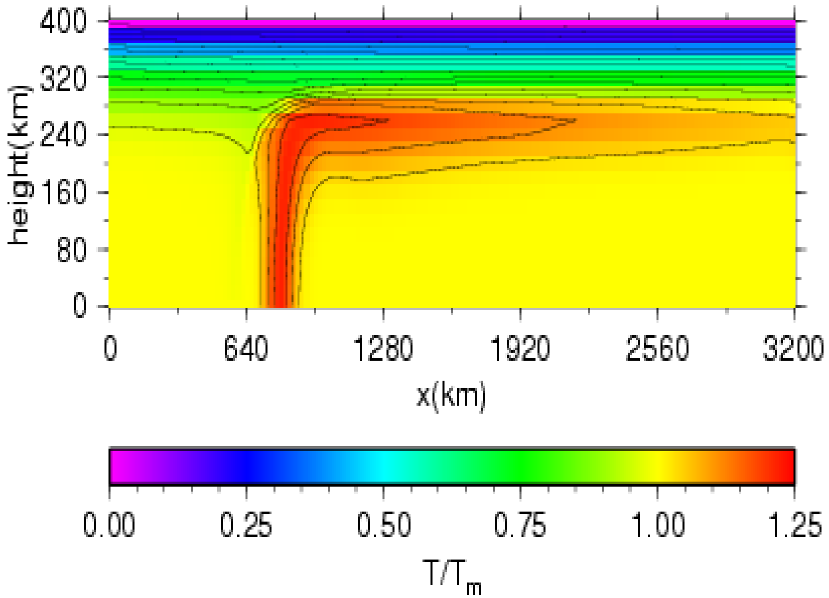
Simple illustration of CITCOM calculation



Observed topography and heatflow of Pacific seafloor (Huang & Zhong, 2005)

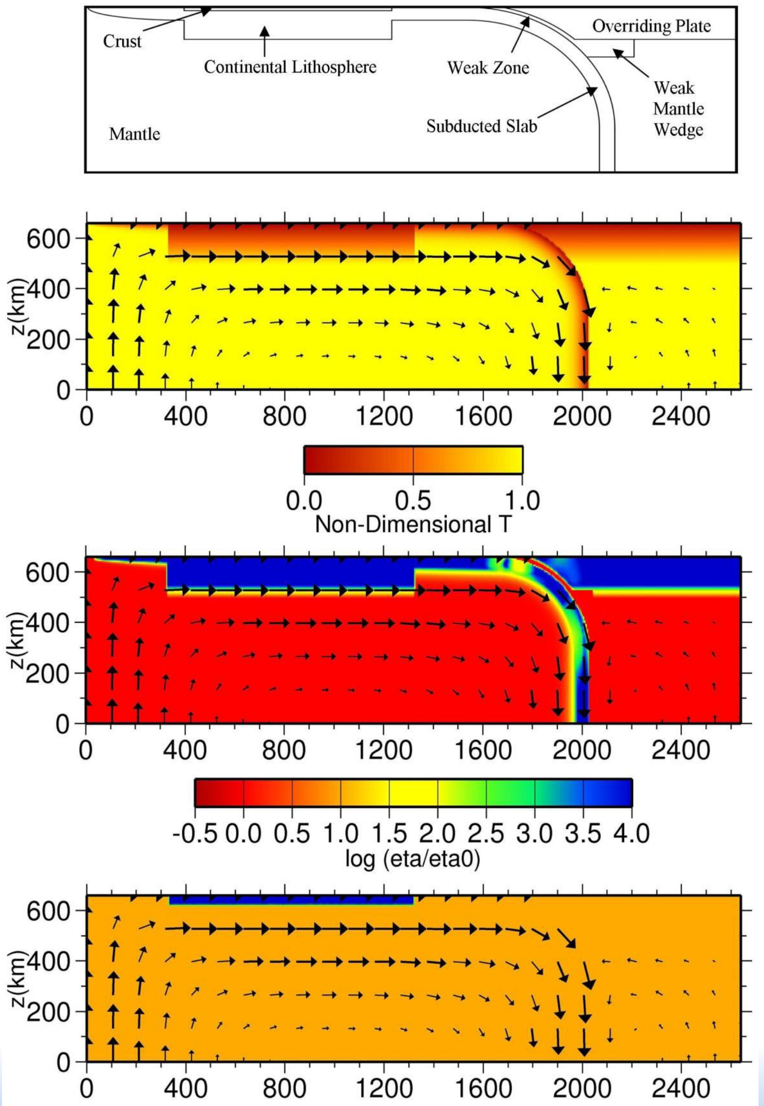
Mantle Plumes

- Most volcanism at plate boundaries (mid-ocean ridges and subduction zones)
- Some significant ‘intraplate’ volcanism (e.g. Hawaii) explained by mantle plumes
- Mantle plumes are hot upwellings from base of mantle (3000 km depth).
- When hitting lithosphere they melt partially to give volcanic activity.

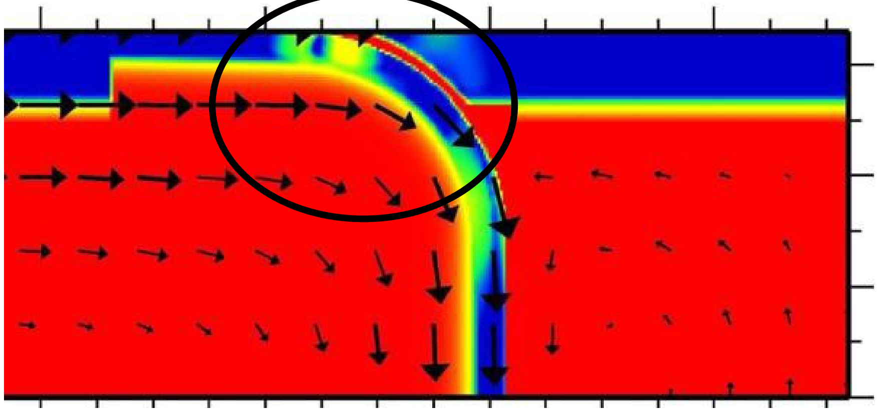


Subduction Zones

- Subducting plates (slabs) drive the movement of tectonic plates: main force to drive plate tectonics
- Subduction zones are also the location where most of the continental crust seems to be formed.
- Understanding dynamics of subduction essential for Earth's evolution



Numerical Challenges



- Modelling lithospheric plates requires large viscosity contrasts (10^4 – 10^6) in very narrow bands (shear zones)
- Solving this with multigrid is difficult, because the coarse levels don’t ‘see’ the narrow, low-viscosity bands
  - This explains why V & W face difficulties in contrast to FMG(V & W)
- Possible solutions(?):
  - Better multigrid algorithms (improved smoothing, AMG)
  - Strong local mesh refinement

Governing Equations

- Governing equations can be described as conservation equations for
  - Mass
  - Momentum
  - Energy
  - Composition
- Symbols have their usual meanings

$$\nabla \cdot \vec{v} = 0$$

$$-\nabla \tau + \nabla p = \Delta \rho g \hat{z}$$

$$\frac{\partial T}{\partial t} + \left(\vec{v} \cdot \nabla\right) T = \nabla^2 T + H$$

$$\frac{\partial C}{\partial t} + \left(\vec{v} \cdot \nabla\right) C = 0$$

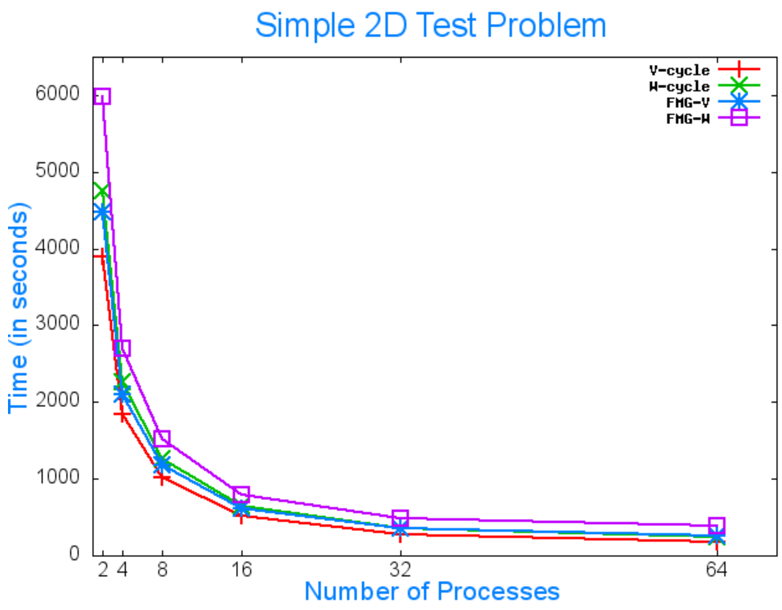
Discrete Linear System

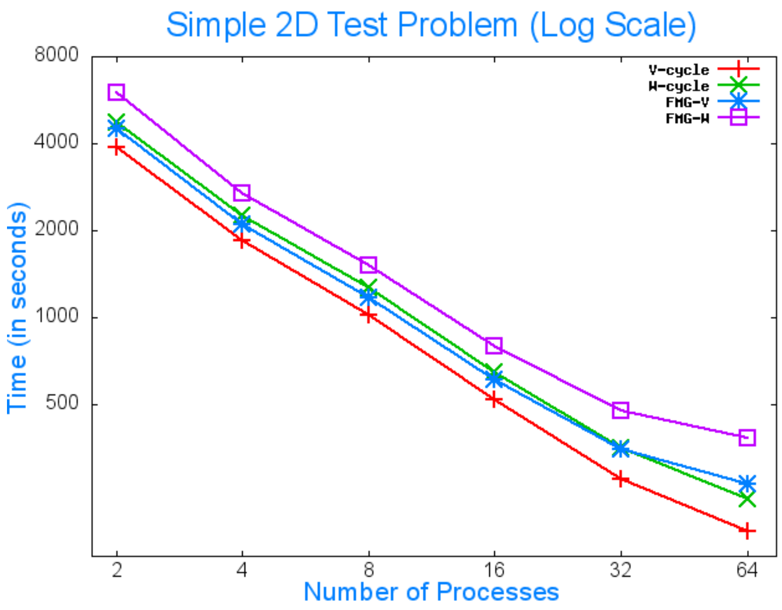
- Governing equations can be written in discrete form as
  - $Au + Bp = f$
  - $B^T u = 0$
- This yields system of linear equations
- Finite elements used are bi-linear in nature
- This system is solved using
  - Iterative MG method for Stokes equations (first two equations on previous slide)
  - Explicit forward integration for Temperature
  - Tracer method for composition

Simple 2D Test Problem

Number of Processes	Time (in seconds)			
	V-cycle	W-cycle	FMG-V	FMG-W
2	3902	4754	4487	5987
4	1851	2264	2104	2695
8	1026	1266	1177	1515
16	523	647	613	799
32	278	354	352	479
64	182	236	265	384

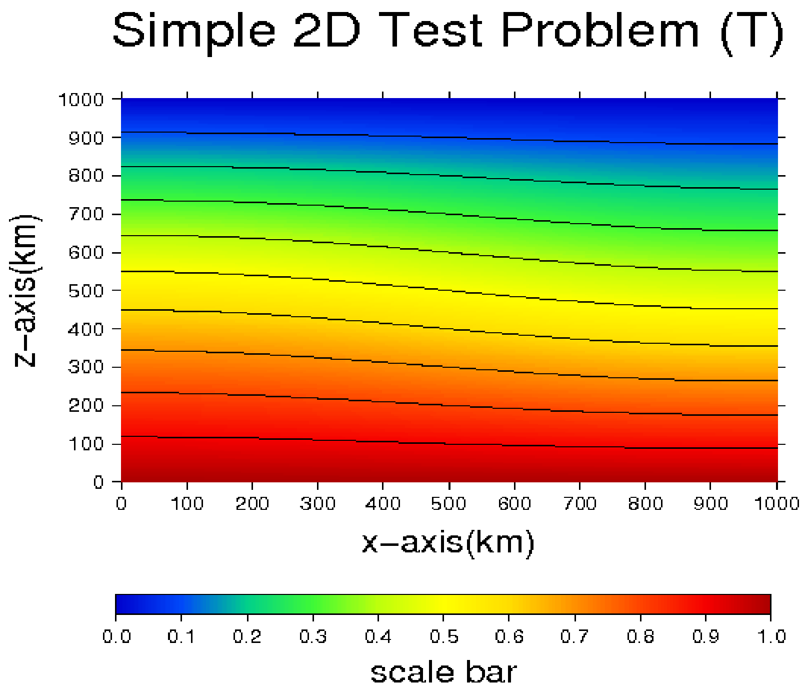
Simple 2D Test Problem

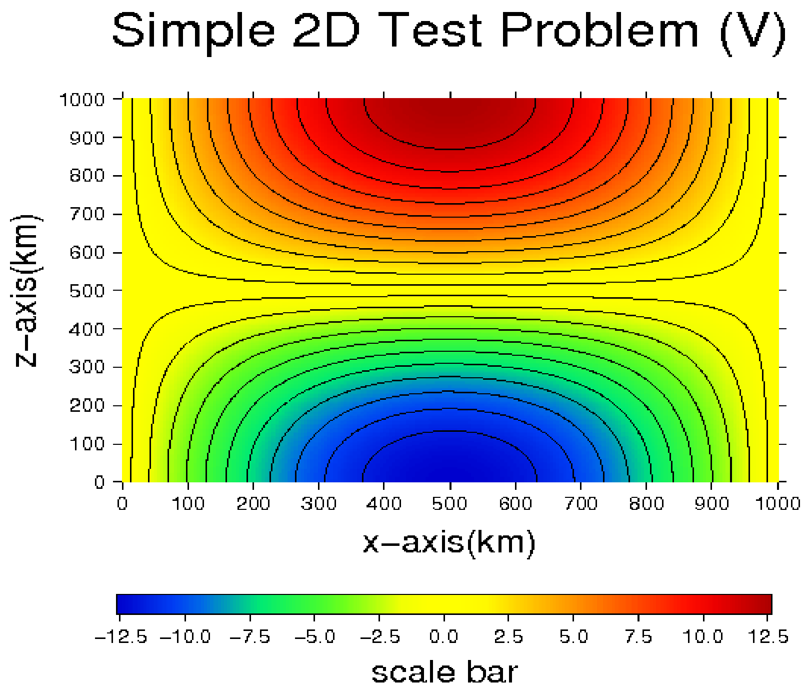




- Due to memory limitations (8GB per node)
  - One core per node for 2 MPI processes jobs is used
  - Two cores per node for 4 MPI processes jobs are used
- Four cores per node are used for all other

Simple 2D Test Problem





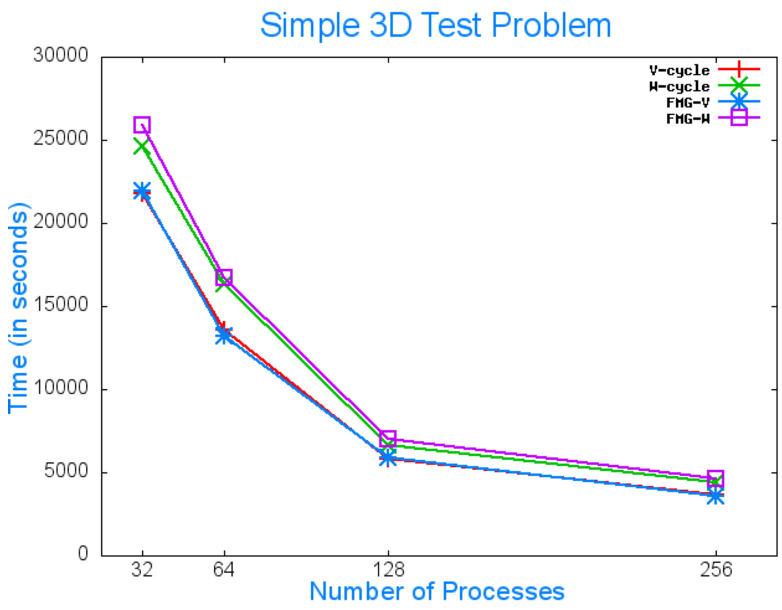
- Problem size
  - Initial mgunits (elements): 128 X 128 = 16,384
  - Global number of elements: 2048 X 2048 = 4,194,304
  - Global number of nodes: 2049 X 2049 X 1 = 4,198,401

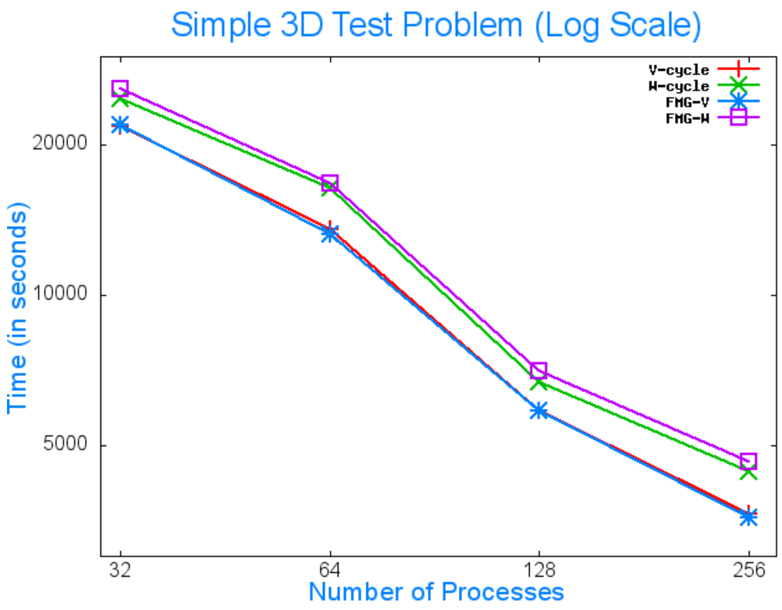
Simple 3D Test Problem

Number of Processes	Time (in seconds)			
	V-cycle	W-cycle	FMG-V	FMG-W
32	21826	24656	21935	25907
64	13548	16354	13194	16736
128	5851	6674	5869	7039
256	3635	4420	3586	4641

- Problem Size
  - Initial mgunits (elements): 32 X 16 X 32 = 16,384
  - Global number of elements: 512 X 256 X 512 = 67,239,936
  - Global number of nodes: 513 X 257 X 513 = 67,634,433

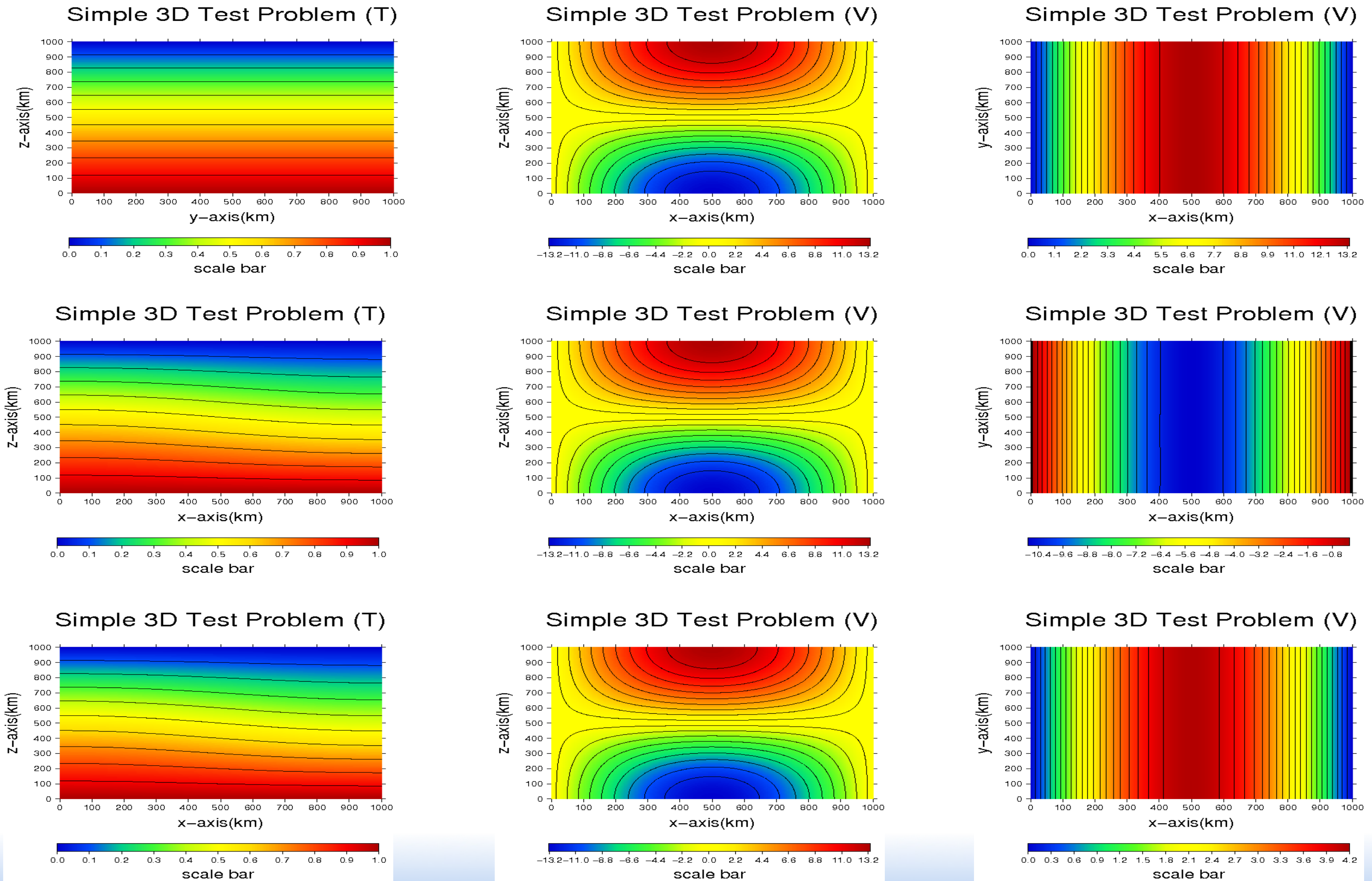
Simple 3D Test Problem





- Due to memory limitation (8GB per node)
  - One core per node for 32 MPI processes job is used
  - Two cores per node for 64 MPI processes job are used
- Four cores per node are used for all other jobs

Simple 3D Test Problem

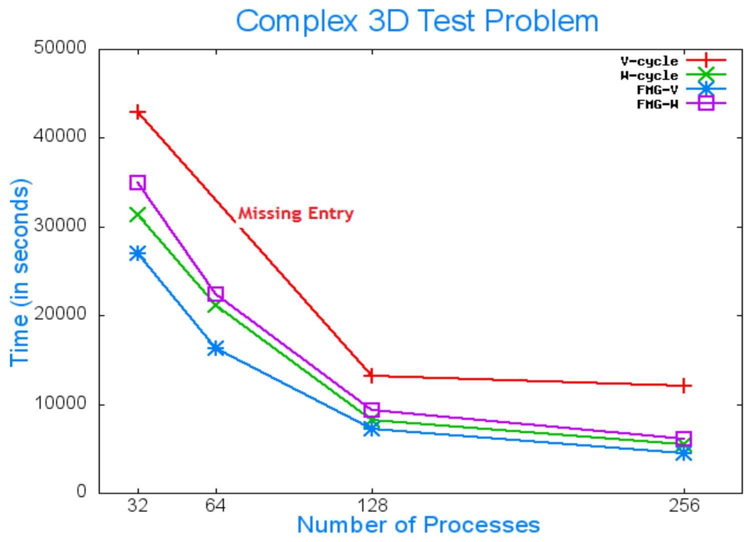


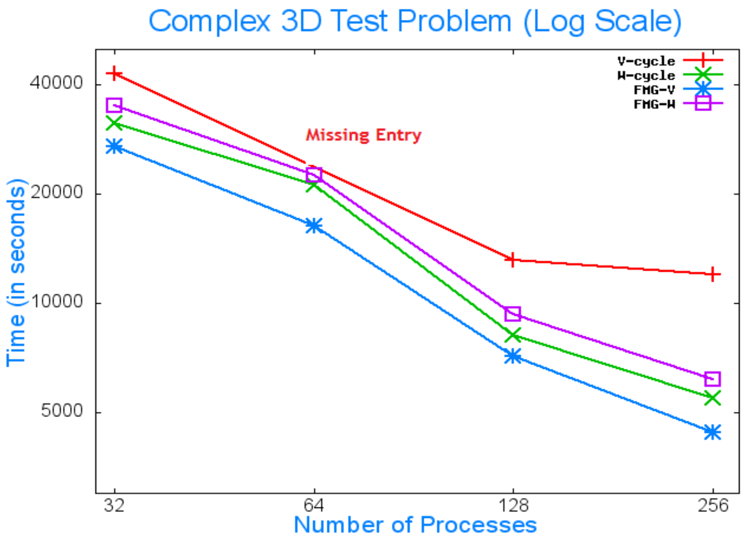
Complex 3D Test Problem

Number of Processes	Time (in seconds)			
	V-cycle	W-cycle	FMG-V	FMG-W
32	42960	31386	26940	34940
64	*	21205	16305	22440
128	13167	8147	7166	9306
256	12031	5469	4423	6150

- Extrapolated* from 88 to 100 steps (49116)
  - 88 iterations time: 43222 seconds

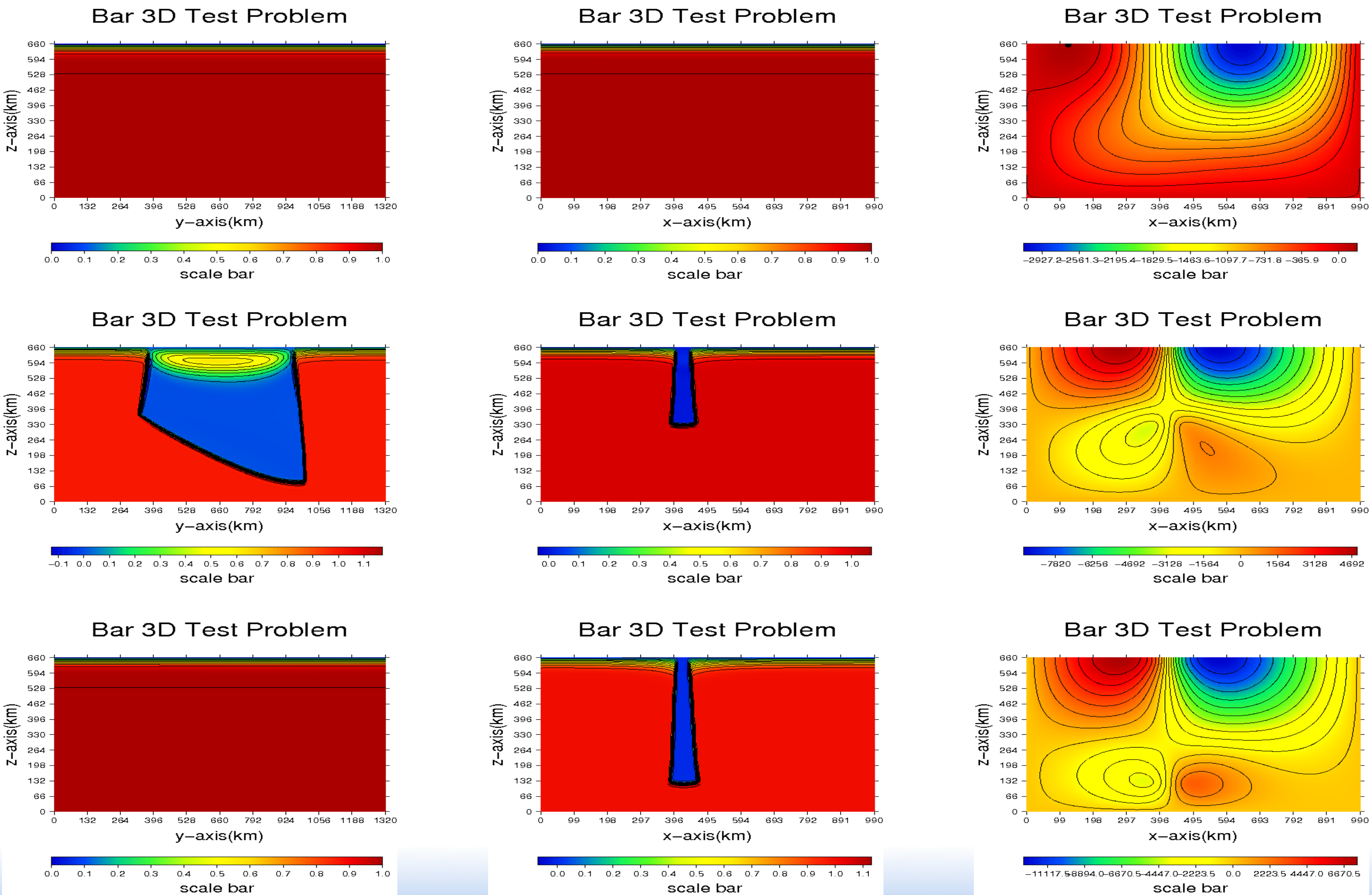
Complex (Bar) 3D Test Problem





- V-cycle failed to complete 100 time steps within 12 hours for 64 MPI processes job
  - Maximum queue time on HECToR is 12 hours
  - This is not understood given that 32 MPI processes job managed to complete 100 steps within 12 hours

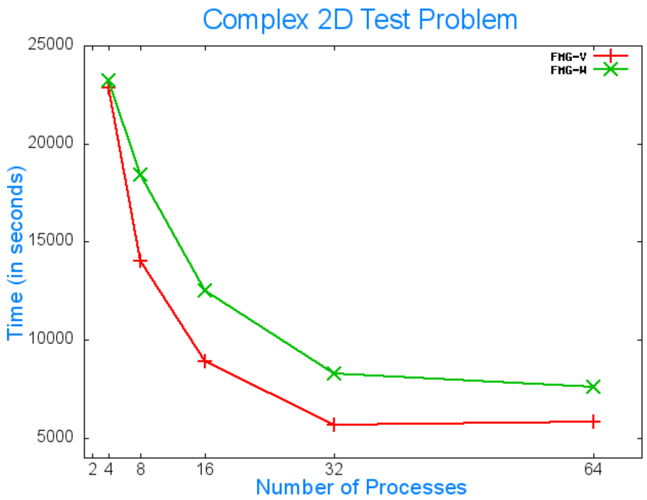
Complex (Bar) 3D Test Problem

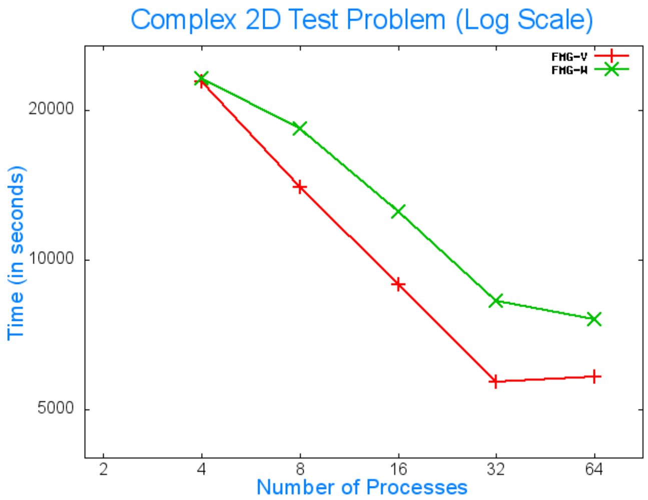


Complex 2D Test Problem

Number of Processes	Time (in seconds)			
	V-cycle	W-cycle	FMG-V	FMG-W
2	~	~	-	-
4	~	~	22883	23238
8	~	~	14005	18396
16	~	~	8934	12493
32	~	~	5676	8286
64	~	~	5815	7600

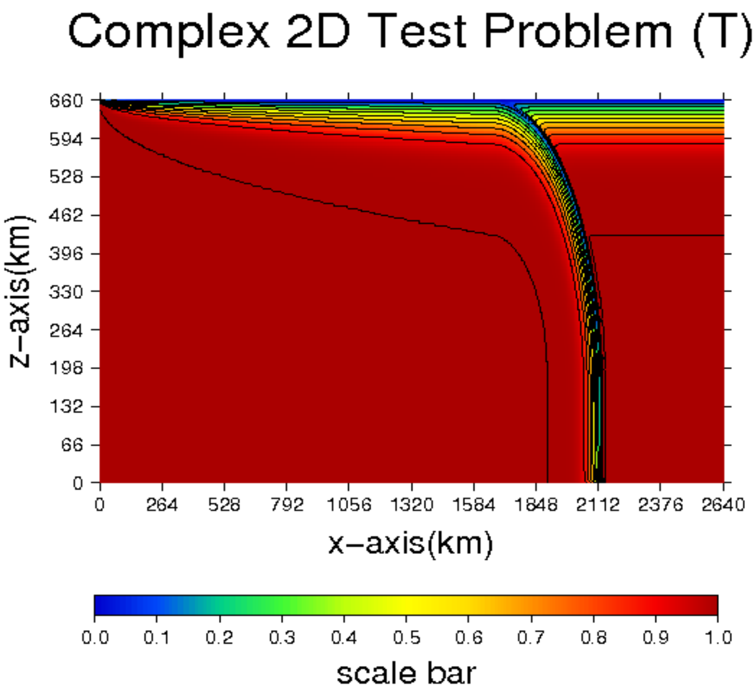
Complex 2D Test Problem

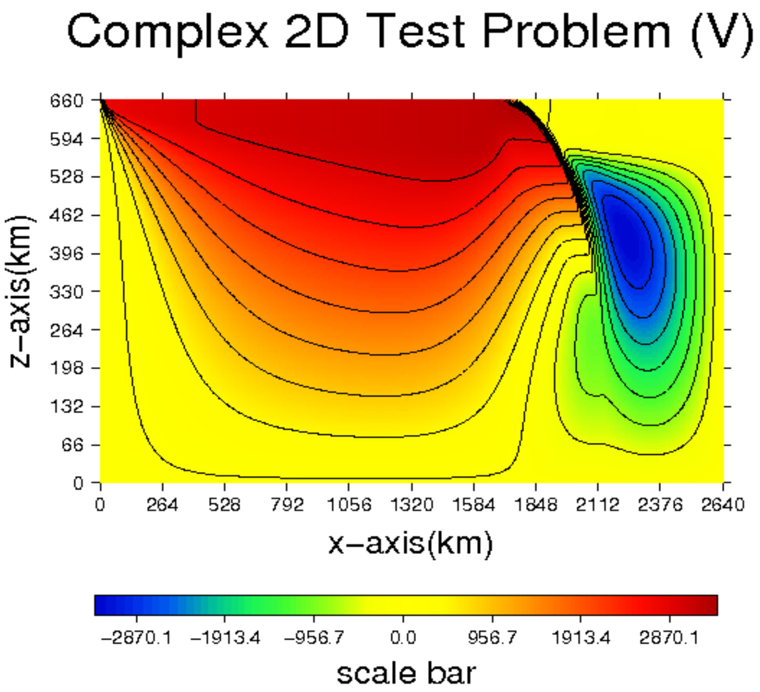




- V-cycle & W-cycle failed to achieve any results
- FMG(V) performed poorly for 64 processes job
  - Problem size per MPI process too small
- FMG(W) is the successful scheme in this

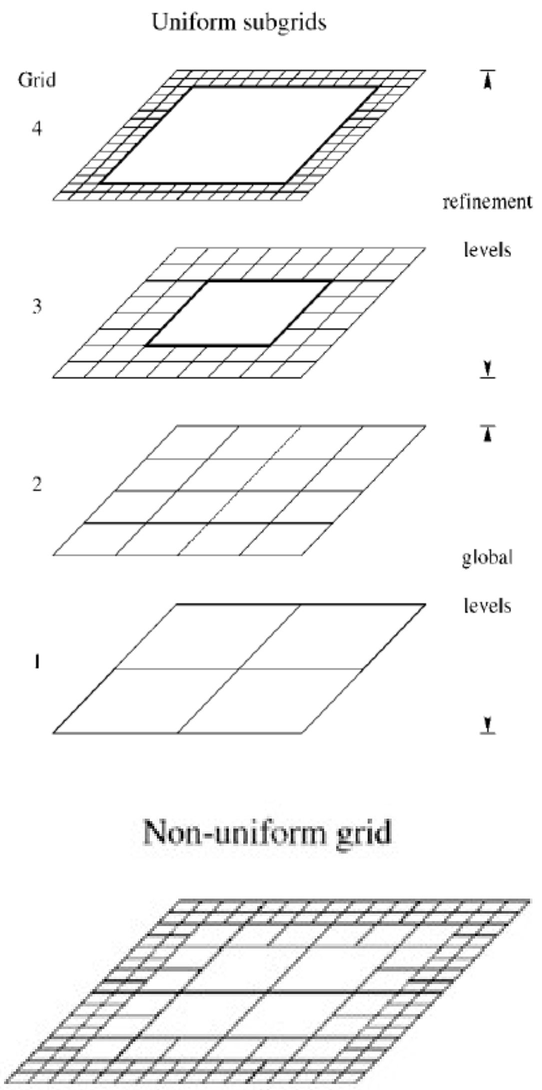
Complex 2D Test Problem





What Next? Local Mesh Refinement

- Aimed to help large velocity/viscosity gradients
- Might introduce more complexity
- Could require extra work by introducing
  - Ghost nodal point
  - Extra book keeping
- Potential to lead to load imbalance



What Next? Prolongation and Restriction

- These help transform info across mesh levels
- Prolongation could be achieved by interpolation
- Restriction could be achieved by averaging
  - Arithmetic averaging
    - $f = \frac{1}{2} (g + h)$
  - Geometric averaging
    - $f = \sqrt{gh}$
  - Harmonic averaging
    - $1/f = 1/g + 1/h$
- These may give significantly different rate

Conclusion

- Success so far
  - Four Multigrid schemes are available
  - Option of efficient schemes for not so hard problems
  - Option of FMG schemes for hard to solve problems
- Difficulties
  - Learning curve was quite steep
- Predictions for next phase
  - Local mesh refinements and improved prolongation and restriction expected to improve Multigrids performance and capability of handling hard to solve problems

Conclusion

- Success so far
  - Four Multigrid schemes are available
  - Option of efficient schemes for not so hard problems
  - Option of FMG schemes for hard to solve problems
- Difficulties
  - Learning curve was quite steep
- Predictions for next phase
  - Local mesh refinements and improved prolongation and restriction expected to improve Multigrids performance and capability of handling hard to solve problems