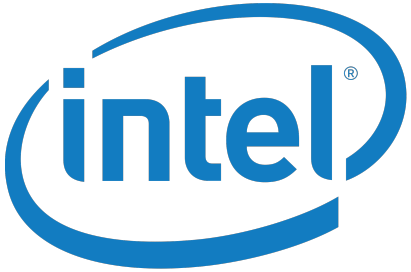
Imperial College London

Sistema FIEB SENAI
Federação das Indústrias do Estado da Bahia

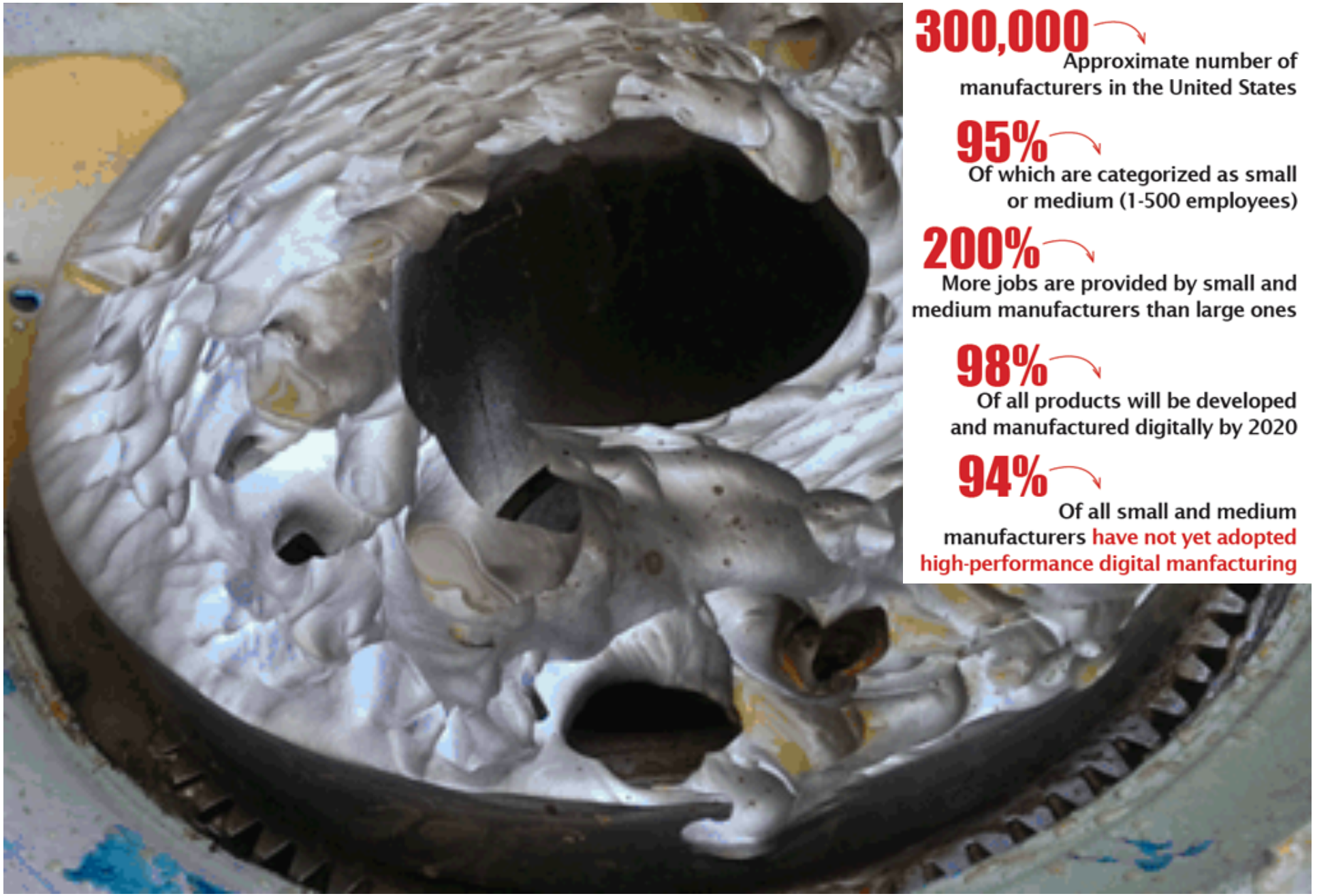
Big Data and the evolving many-core landscape: Are traditional low level programming models really enough?

Gerard Gorman, Imperial College London

Intel Parallel Computing Center
Open Performance portablE SeismiC Imaging



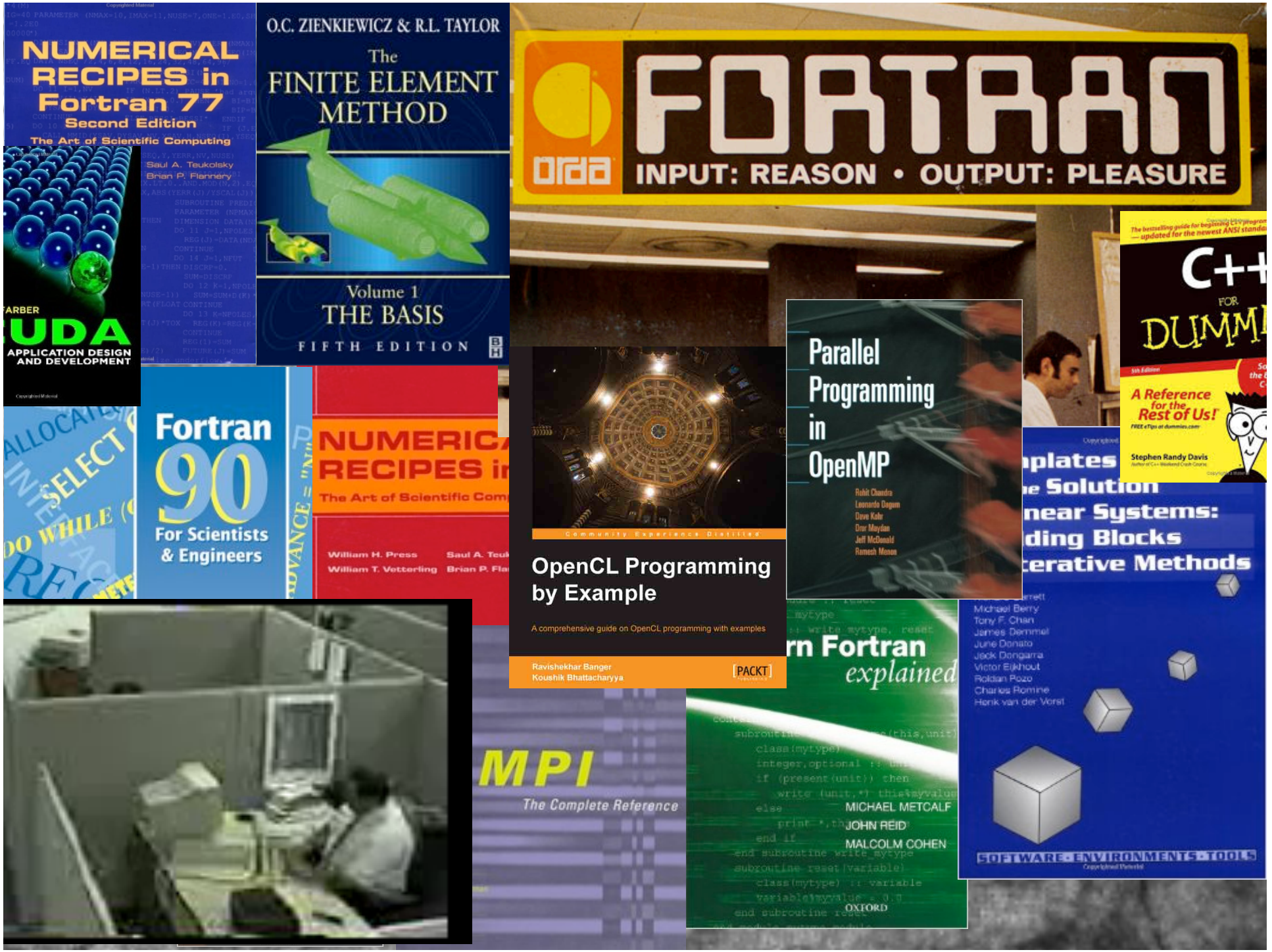
EPSRC
Pioneering research and skills



http://www.ncms.org/index.php/programs/digital-manufacturing-initiative/digital-manufacturing-sig/

HPC often compared to F1…
…the problem is that the comparison is accurate.

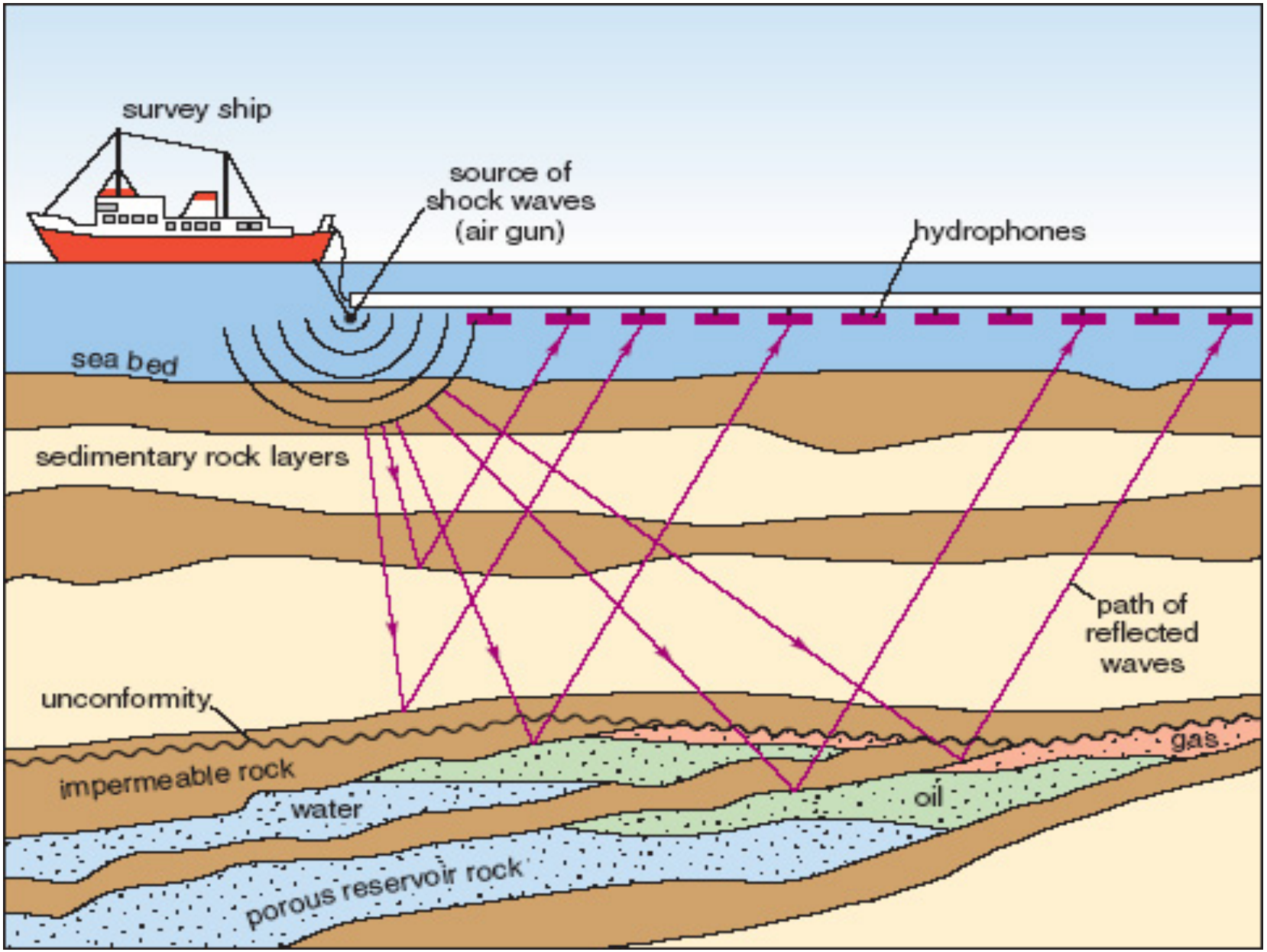
Original: https://www.youtube.com/watch?v=EGUZJVY-sHo


NUMERICAL RECIPES in Fortran 77
Second Edition
The Art of Scientific Computing
Saul A. Teukolsky
Brian P. Flannery
O.C. ZIENKIEWICZ & R.L. TAYLOR
The FINITE ELEMENT METHOD
Volume 1
THE BASIS
FIFTH EDITION
orda
FORTRAN
INPUT: REASON • OUTPUT: PLEASURE
APPLICATION DESIGN AND DEVELOPMENT
Fortran 90
For Scientists & Engineers
NUMERICAL RECIPES
The Art of Scientific Com
William H. Press
William T. Vetterling
Parallel Programming in OpenMP
Rohit Chandra
Leonardo Dagum
Dave Kohr
Dror Maydan
Jeff McDonald
Ramesh Menon
C++ FOR DUMMIES
A Reference for the Rest of Us!
Stephen Randy Davis
OpenCL Programming by Example
A comprehensive guide on OpenCL programming with examples
Ravishekhar Banger
Koushik Bhattacharyya
PACKT
MPI
The Complete Reference
Fortran explained
MICHAEL METCALF
JOHN REID
MALCOLM COHEN
OXFORD
Michael Berry
Tony F. Chan
James Demmel
June Donato
Jack Dongarra
Victor Eijkhout
Roldan Pozo
Charles Romine
Henk van der Vorst
SOFTWARE • ENVIRONMENTS • TOOLS

SC14 Registration Count 2014-Nov-10

Code	Description	Booked
NOTES	Tutorial Notes	
MF01	OpenACC: Productive, Portable Performance on Hy	44
MF02	Advanced OpenMP: Performance and 4.0 Features	37
MF03	Advanced MPI Programming	68
MF04	OpenCL: A Hands-On Introduction	39
MF05	Fault-tolerance for HPC: Theory and Practice	51
MF06	Parallel I/O In Practice	60
MF07	Debugging and Performance Tools for MPI and Ope	35
MF08	Node-Level Performance Engineering	47
MF09	Python in HPC	106
MH01	In Situ Data Analysis and Visualization with ParaVie	21
MH02	Parallel Programming with Charm++	15
MH03	InfiniBand and High-Speed Ethernet for Dummies	104
MH04	Introducing R: From Your Laptop to HPC and Big D	51
MH05	Effective HPC Visualization and Data Analysis using	36
MH06	Enhanced Campus Bridging via a Campus Data Se	24
MH07	OpenSHMEM: Tools for Productive PGAS Program	27
MH08	Designing and Using High-End Computing Systems	97
SF01	Large Scale Visualization with ParaView	41
SF02	Linear Algebra Libraries for High-Performance Com	31
SF03	From "Hello World" to Exascale Using x86, GPUs a	76
SF04	Parallel Programming in Modern Fortran	25
SF05	Efficient Parallel Debugging for MPI, Threads, and I	28
SF06	Hands-On Practical Hybrid Parallel Application Perf	27
SF07	A Hands-On Introduction to OpenMP	32
SF08	SciDB - Manage and Analyze Terabytes of Array Da	34
SF09	Parallel Computing 101	87
SF10	Programming the Xeon Phi	69
SH01	MPI+X - Hybrid Programming on Modern Compute	43

Specific example – oil&gas industry



Full Waveform Inversion

- Given a sound source, and an array of receivers, can we infer the subsurface?
- Yes - but it is computationally expensive:
  - Lots of data, terabytes per survey (this is a real Big Data problem!)
  - Dominant computational expense is running the wave model (aka **forward model**, and **propagator)** for each **shot** of data.
  - Regardless of how good the inversion algorithm is, the quality of the final subsurface image will be limited by the accuracy of the wave model used.

Opportunity & challenge I
Elastic wave equation (Improved model)

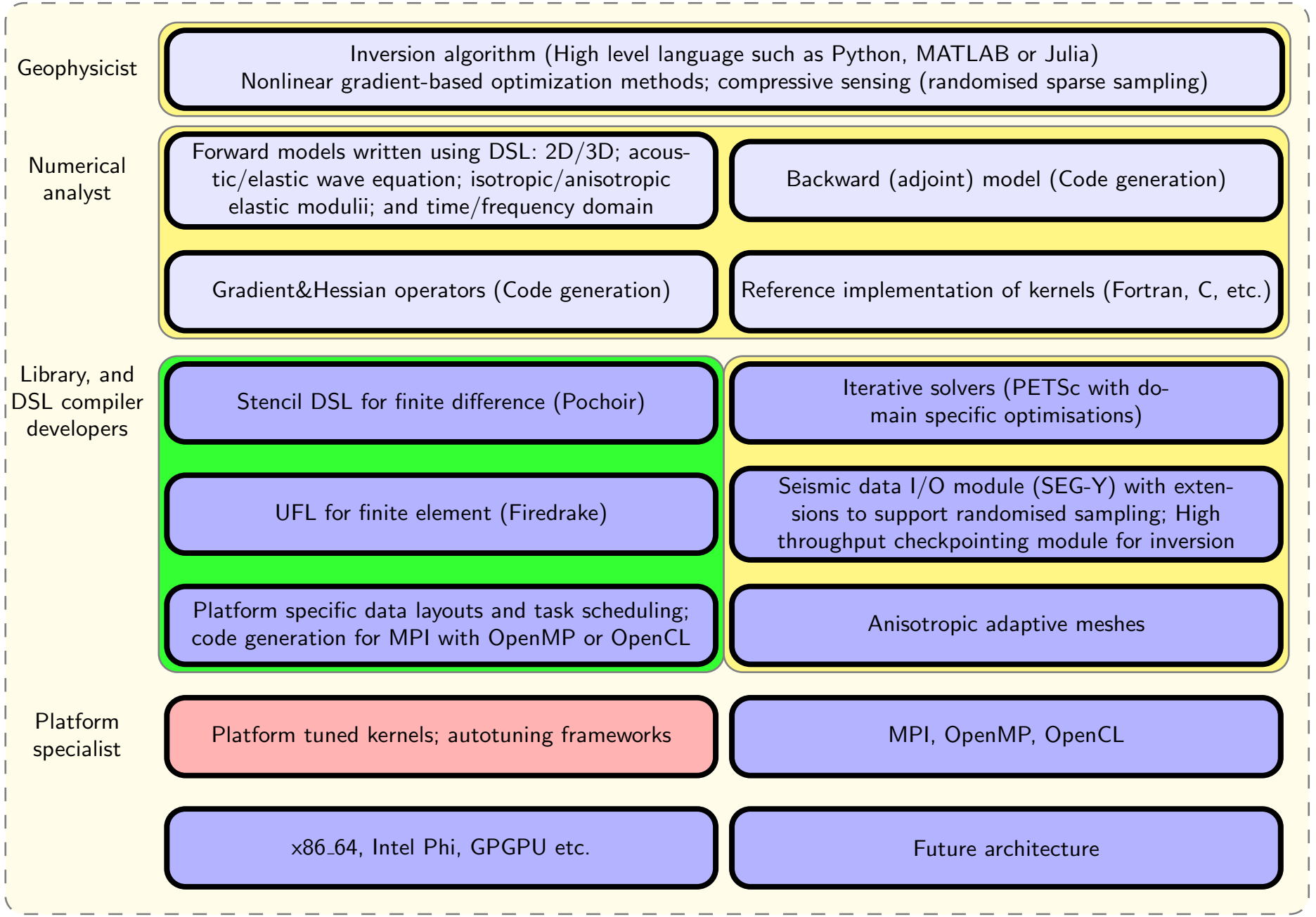
- Elastic wave model provides a significantly better representation of the wave field than the standard acoustic models.
  - Models both p-waves, s-waves and Rayleigh waves.
- It is also **much more expensive** to compute:
  - More terms in the equation to compute.
  - S-waves travel at about half the speed of p-waves, and there have half the wavelength, therefore:
    - Grid resolution needs to be doubled (factor of 8 increase in memory for 3D).
    - Time step needs to be halved – therefore must execute twice the number of time steps.

Opportunity & challenge II
Advanced numerical methods

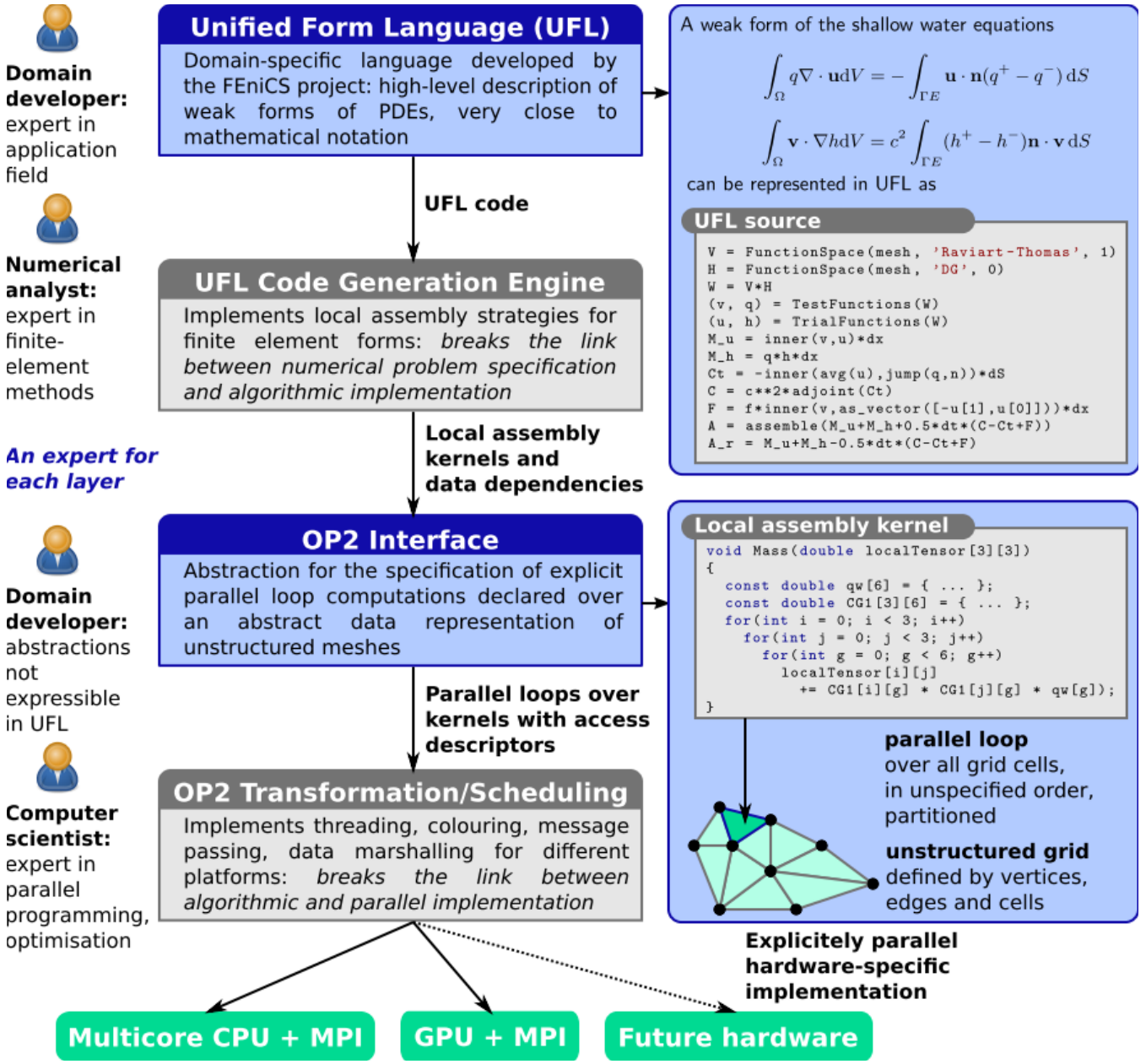
- Regular grids with finite difference is the modus operandi of oil and gas industry.
- However:
  - Unstructured grids are more efficient at representing complex geological features.
  - High order methods can achieve the same accuracy as finite difference methods using:
    - Coarser resolution (fewer grid points).
    - Larger time step.
    - Less memory.
    - Shorter time to solution.
    - Great data locality, opportunities for vectorisation etc.
- Best example from global seismic: SeisSol – Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG), 2014 Gorden Bell finalist.
- Implementation complexity!
  - A lot more software is required to manage unstructured grids.
  - Impacts entire inversion software stack.
  - High order methods are many times more involved finite difference methods.

Opportunity & challenge III
Architecture and code modernisation

- Computing performance continues to track Moore's Law – but you have to work harder to make use of it.
- Many-core era software must exploit parallelism at every level to achieve good computational efficiency, e.g.:
  - Various parallel programming models (MPI, OpenMP/pthreads, OpenCL, OpenMP, Cilk, etc.)
  - Deep memory hierarchy, data locality.
  - Vectorisation (AVX).
  - Heterogeneous computing – Intel Xeon, Xeon Phi, etc.
- Parallel programming has always been considered challenging – and now it has become over more demanding:
  - Greater need for specialists in parallel programming for HPC.
  - Increasingly difficult for domain specialists to implement high performance software although they are the algorithm specialists.
  - Traditional numerical algorithms may need to be discarded in favor of methods better suited to computer architectures.
- Domain Specific Languages (DSL's) offer a route to bridge the divide between domain specialists (often the application developers) and parallel programming specialists (in this case compiler writers).


Geophysicist
Inversion algorithm (High level language such as Python, MATLAB or Julia)
Nonlinear gradient-based optimization methods; compressive sensing (randomised sparse sampling)
Numerical analyst
Forward models written using DSL: 2D/3D; acoustic/elastic wave equation; isotropic/anisotropic elastic modulii; and time/frequency domain
Backward (adjoint) model (Code generation)
Gradient&Hessian operators (Code generation)
Reference implementation of kernels (Fortran, C, etc.)
Library, and DSL compiler developers
Stencil DSL for finite difference (Pochoir)
Iterative solvers (PETSc with domain specific optimisations)
UFL for finite element (Firedrake)
Seismic data I/O module (SEG-Y) with extensions to support randomised sampling; High throughput checkpointing module for inversion
Platform specific data layouts and task scheduling; code generation for MPI with OpenMP or OpenCL
Anisotropic adaptive meshes
Platform specialist
Platform tuned kernels; autotuning frameworks
MPI, OpenMP, OpenCL
x86_64, Intel Phi, GPGPU etc.
Future architecture

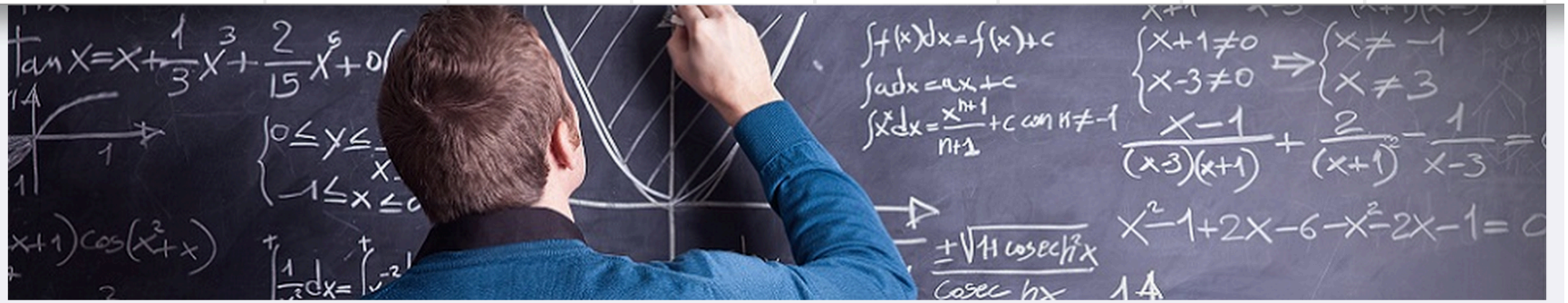
UFL/Firedrake



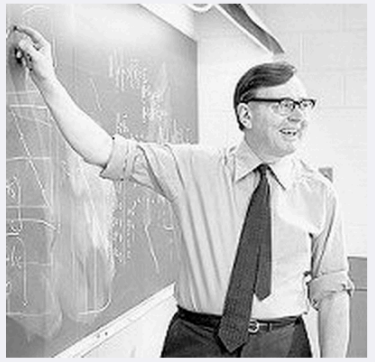
Abstraction facilitates automatic differentiation of models

nag®

Numerical Services | HPC and Supercomputing | Numerical Software | **Collaboration** | Industries | Support | About



THE WILKINSON PRIZE FOR NUMERICAL SOFTWARE 2015



The Wilkinson Prize was established to honour the outstanding contributions of Dr James Hardy Wilkinson to the field of numerical software. It is awarded every four years at the International Congress on Industrial and Applied Mathematics by Argonne National Laboratory, the National Physical Laboratory, and the Numerical Algorithms Group. The recipients are authors of an outstanding piece of numerical software, judged on:

- the clarity of the software implementation and documentation;
- the importance of the application(s) addressed by the software;
- the portability, reliability, efficiency and usability of the software implementation;
- the clarity and depth of analysis of the algorithms and the software in the submission;
- the quality of the test software.

The 2015 prize is awarded to P.E. Farrell (University of Oxford), S.W. Funke (Simula Research Laboratory), D.A. Ham (Imperial College London), and M.E. Rognes (Simula Research laboratory) for the development of **dolfin-adjoint**, a package which automatically derives and solves adjoint and tangent linear equations from high-level mathematical specifications of finite element discretisations of partial differential equations. The prize will be presented at ICIAM 2015 and will consist of $3000 plus a commemorative plaque for each winner.

Example using UFL/Firedrake

- Velocity-stress formulation of elastic wave equation, with isotropic stress:

$$\rho \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \mathbb{T} \tag{1}$$

$$\frac{\partial \mathbb{T}}{\partial t} = \lambda \left(\nabla \cdot \mathbf{u} \right) \mathbb{I} + \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}} \right) \tag{2}$$

- Weak form of equations written in the high-level Unified Form Language (Logg et al., 2012), e.g. (1):

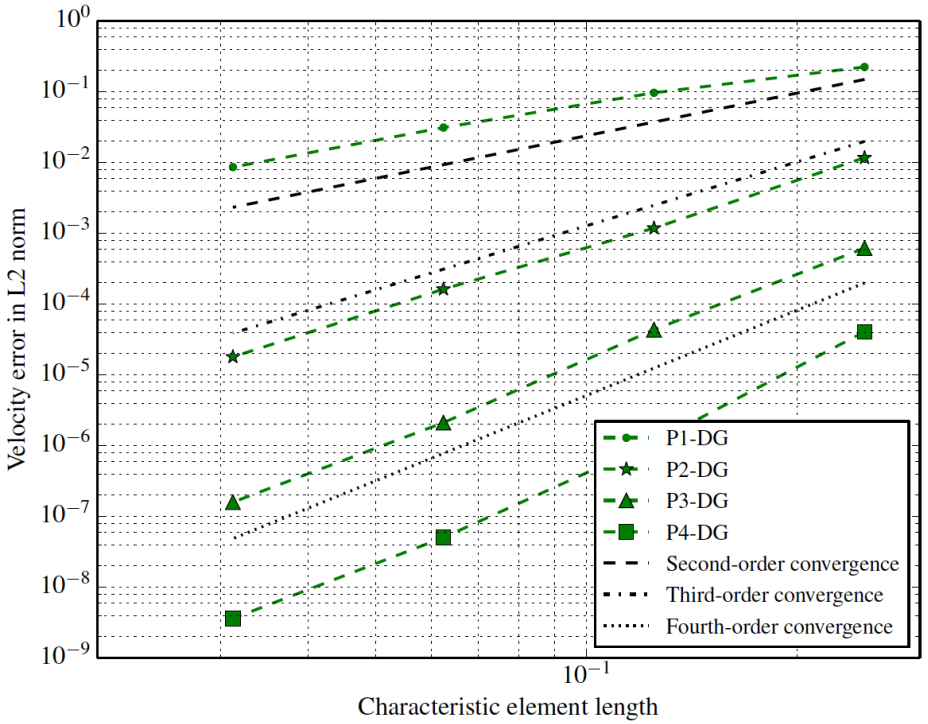
```
F_u = density*inner(w, (u - u0)/dt)*dx + inner(w, div(s0))*dx
```

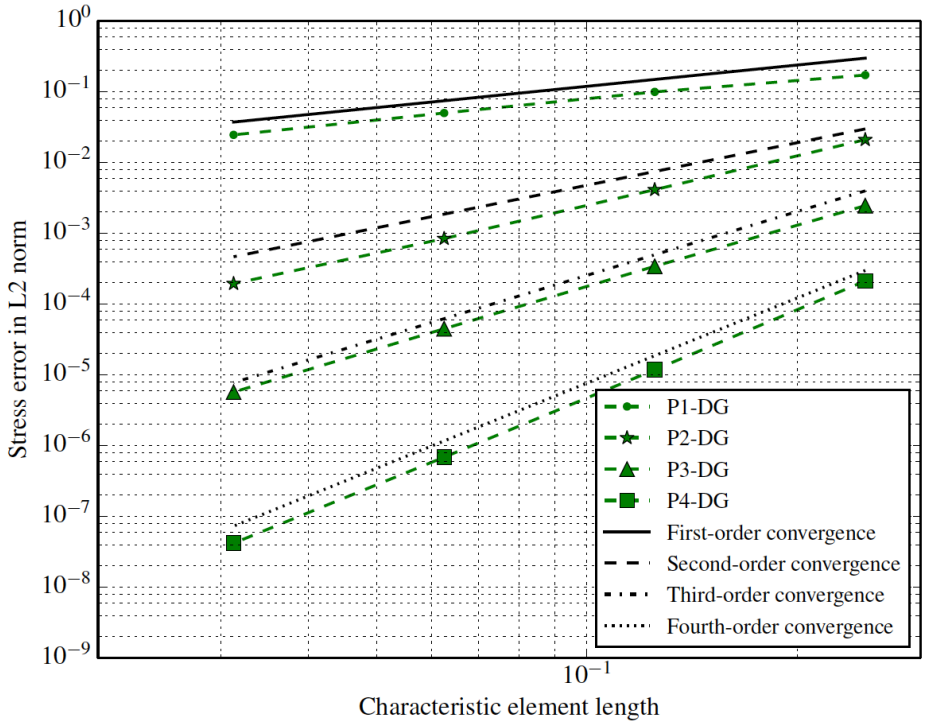
- Change order of DG spatial discretisation with a 'flick-of-a-switch', e.g. P1-DG to P4-DG:

```
U = TensorFunctionSpace(mesh, "DG", 1) →
U = TensorFunctionSpace(mesh, "DG", 4)
```

- UFL compiled down to low-level, optimised C code, and targetted to specific hardware architecture.

- Simulation of an eigenmode to verify model correctness.
- Numerical solution compared with analytical solution at $T = 5$ s.
- For degree d polynomial basis functions ($d = 1,2,3,4$) on increasing mesh resolutions, we observed $O(d+1)$ convergence for velocity, $O(d)$ for stress.
- ...Except for $d = 4$ where the overall convergence was limited to 4th order because of 4th order leapfrog scheme.





Alternative approach - Stencil languages

- Good for applications such as finite difference, image processing (see conference proceedings for HiStencils).
- Automate the entire workflow with smart abstractions
  - Python/SymPy (high level, symbolic math).
  - Code generation to target stencil language, e.g. Pochoir.
  - Code generation from stencil language to native source code.
- High level retains **expressiveness – key to innovation**.
- Extreme optimization techniques performed by the stencil compiler.
- Separation of concerns enables effective collaboration.

Conclusions

- To bring about disruptive change in areas such as data inversion and design **optimization progress must be made on three front**s:
  - Better models.
  - Advanced numerical models.
  - Domain specific languages (DSL's) and code generation.
- A number of stencil compilers are already available for finite difference and performance results show that these already offer an attractive code modernization solution without impacting the rest of the software stack.
- Methods pioneered by SeisSol for global seismic inversion appears to offer the best numerical approach currently. A drawback with moving to these sophisticated methods is that they are challenging to implement – which raises issues for code optimization and sustainability.
- UFL and the Firedrake compiler facilitates a separation of concerns which allows application developers to develop these sophisticated schemes without any knowledge of the hardware, while compiler writers take these abstract forms and automatically generate native source code whose performance is competitive to what a human expert could achieve.

Acknowledgements

- Imperial College London:
  - Gerard Gorman (Earth Science and Engineering)
  - David Ham (Computing/Mathematics)
  - Christian Jacobs (Earth Science)
  - Paul Kelly (Computing)
  - Michael Lange (Earth Science/Computing)
  - Lawrence Mitchell (Computing/Mathematics)
  - Matthew Piggott (Earth Science and Engineering)
  - Tianjiao Sun (Computing)
- SENAI CIMATEC, Brazil
  - Renato Miceli (Computing)
  - Marcos de Aguiar (Computing)
  - Felippe Vieira Zacarias (Computing)

Links

- OPESCI – http://opesci.github.io (Intel PCC)
- Firedrake – http://www.firedrakeproject.org
- FEniCS – http://fenicsproject.org
- Dolfin-adjoint – http://www.dolfin-adjoint.org
- PRAgMaTIc –https://github.com/meshadaptation/pragmatic
- PETSc - http://www.mcs.anl.gov/petsc/

- PRISM – http://prism.ac.uk
- AMCG – http://amcg.ese.ic.ac.uk