PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex:
  multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist
  – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex: multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex: multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex: multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex: multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

PDE Software has changed from the 1990s to 2005

- Fewer scientists program
- If they program, they often prefer Matlab
- The problems get more complex:
  multi–physics/domain/scale/institutional/code
- More high-quality/mature software libraries exist
  – no need to reinvent the wheel
- Legacy codes are here to stay – could we integrate them in new systems as black boxes?
- Scientists want the numerics in problem solving environments

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Current views on PDE Software at Simula

- Software tools must be *programmable* at all levels
- Users can program with a Matlab-like syntax (Python)
- Problem solving environment = set of Python modules
- "Standard operations" are performed by "standard libraries": LAPACK, PETSc, Trilinos, ML, Hypre, Vtk, ...
- Solve PDE systems by connecting individual PDE solvers
- Potential users have 10 min attention span
  – requires strong focus on build/install and documentation

Example on connecting single-PDE components

Build turbulent flow solvers from PDE components

- Basic idea: solve PDE systems by operator splitting
- Can also split an implicit formulation by block preconditioning
- Navier-Stokes solver: Stokes solver, advection solver
- k-ϵ model: N-S solver, convection-diffusion-reaction solvers
- The same idea is applies to RANS/URANS, LES, DNS, elliptic relaxation models (and PDE systems in general)

Goal

Composing turbulent flow solvers is like composing matrix-based algorithms in Matlab

Example on connecting single-PDE components

Build turbulent flow solvers from PDE components

- Basic idea: solve PDE systems by operator splitting
- Can also split an implicit formulation by block preconditioning
- Navier-Stokes solver: Stokes solver, advection solver
- k-ϵ model: N-S solver, convection-diffusion-reaction solvers
- The same idea is applies to RANS/URANS, LES, DNS, elliptic relaxation models (and PDE systems in general)

Goal

Composing turbulent flow solvers is like composing matrix-based algorithms in Matlab

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python? (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python? (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

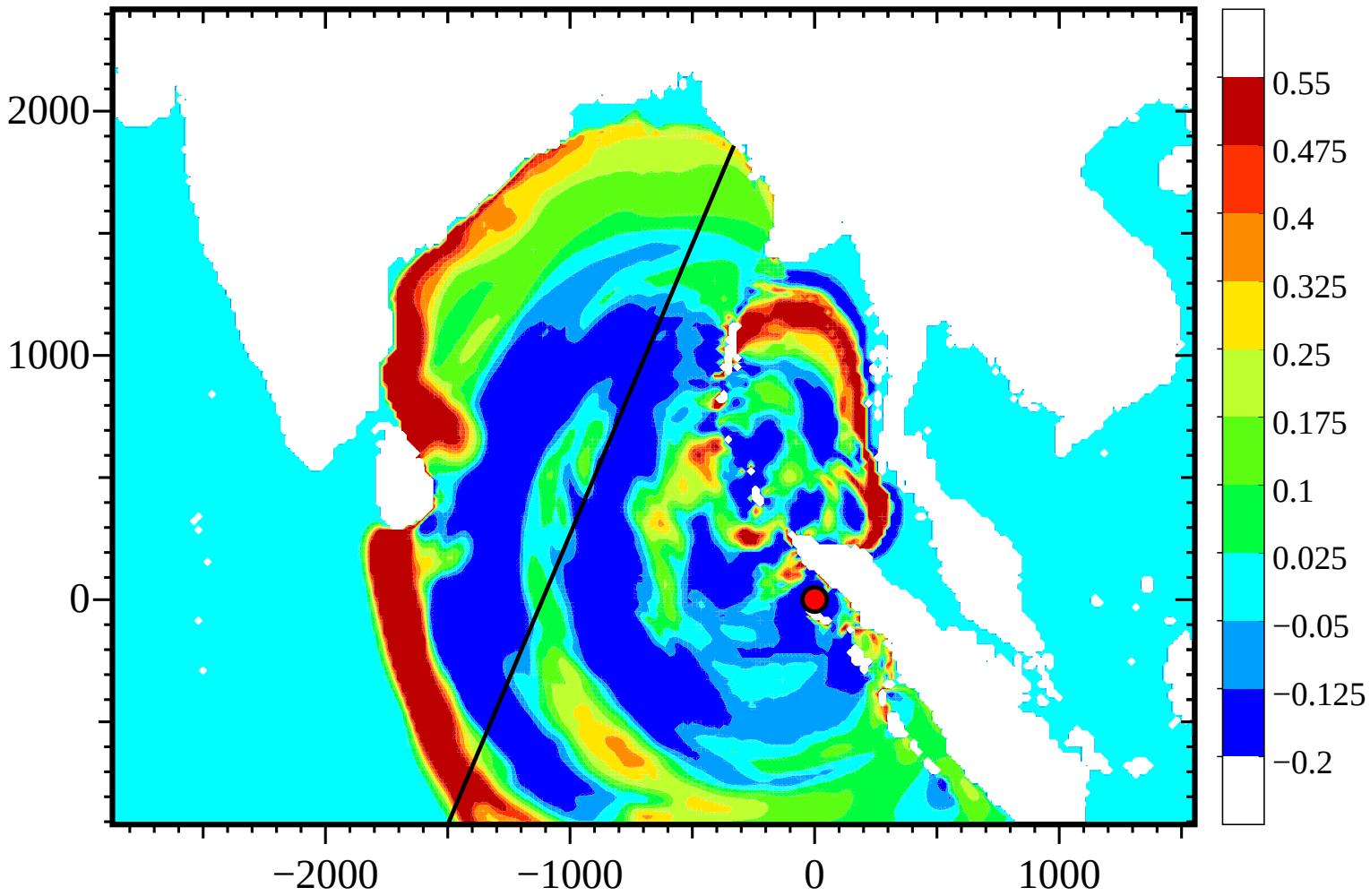
Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

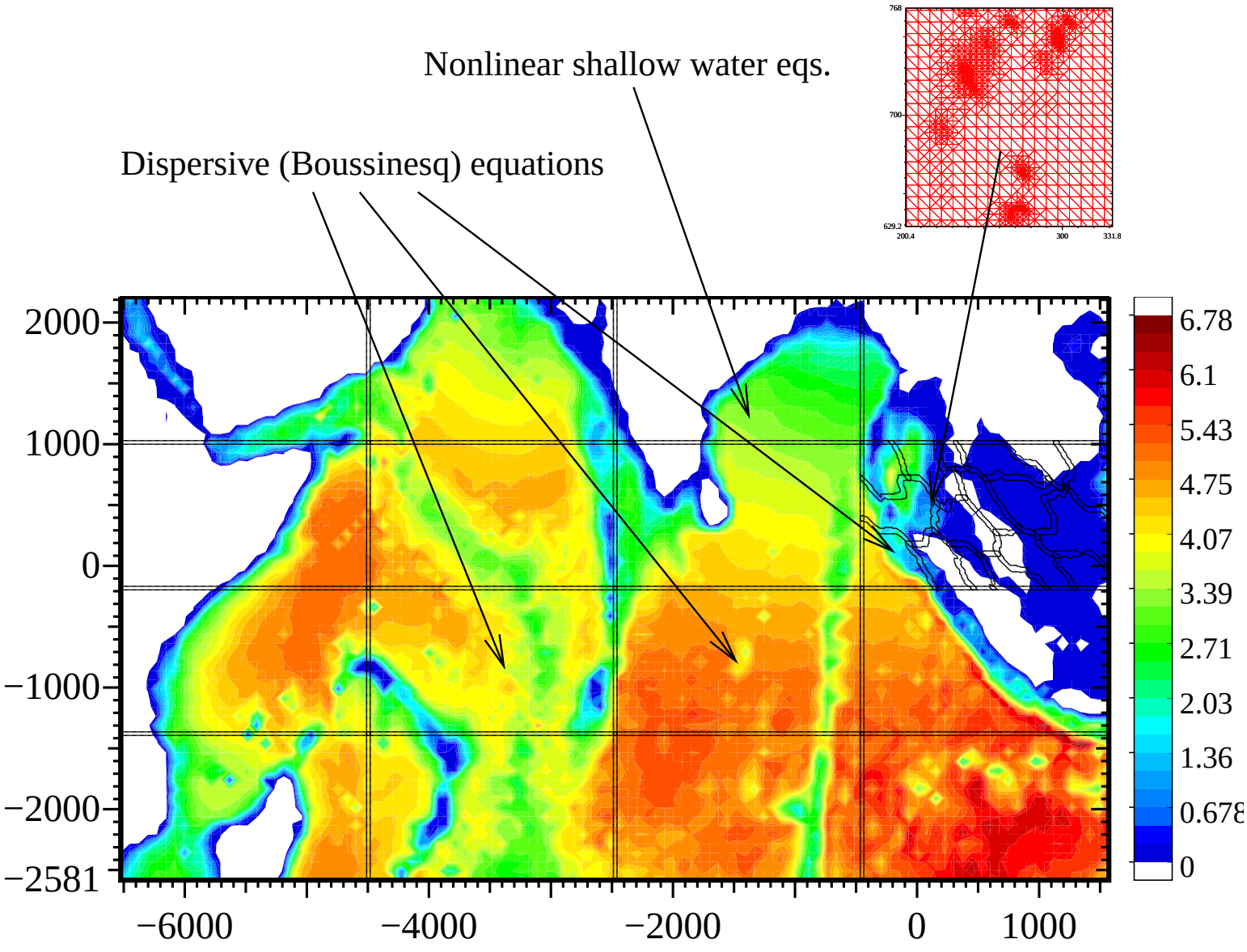
Current software initatives at Simula

- FAMMS: automatic PDE code verification
- Swiginac: symbolic math in Python
- PYSE: finite difference stencils in Python
- Symbolic "FIAT" (finite element basis functions)
- SciPy extensions for sparse matrices
- Anyplot: generic Matlab-like interface to curve plotting
- Plans for Anyviz (for scalar and vector fields over grids)
- How to write efficient PDE solvers in Python?
  (PETSc, ML, FEniCS, Diffpack)
- Parallelizing legacy codes via Python
- Multi-physics/domain/scale and parallelization via domain decomposition

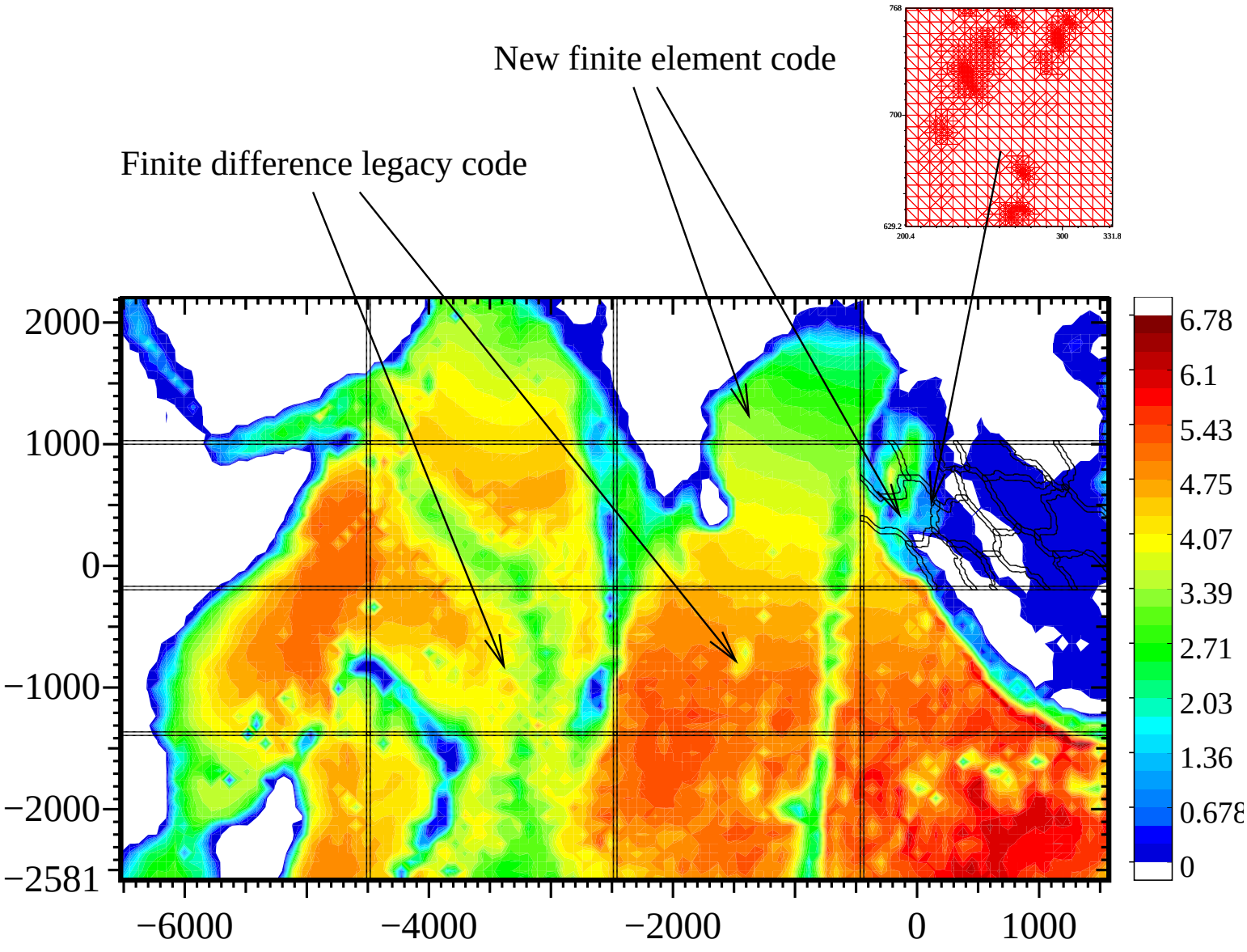
Example: tsunami simulation (the Dec 26, 2004 event)



Multi-physics/domain/code via domain decomposition



Multi-physics/domain/code via domain decomposition



Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field

Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field

Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field

Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field

Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field

Summary: Simula $\leftrightarrow$ FEniCS

- We see FEniCS as a good place to publish software
- The FEniCS modules (FIAT, FFC, Sieve, ...) looks very promising
- We are interested in the flow software and methods
- We have experience with writing generic PDE software (code, documentation, users, industry)
- We work on challenging applications
- We have experience with publishing, conferences and workshops in the "PDE software" field