Efficient Solution of Incompressible Flows with Moving Bodies

Bill Henshaw

Center for Applied Scientific Computing,
Lawrence Livermore National Laboratory, Livermore, CA, USA.

University of Delaware,
Newark, Delaware, November 6, 2012.

Acknowledgments.

Primary collaborator for this effort:

Kyle Chand (LLNL),

Other key collaborators:

Jeff Banks (LLNL), Don Schwendeman (RPI).

Supported by:

ASCR Department of Energy, Office of Science, ASCR Applied Math Program .

LDRD LLNL: Laboratory Directed Research and Development (LDRD) program .

NSF National Science Foundation .

Outline

1. Background: overlapping grids, Overture and CG.
2. Incompressible flows and rigid bodies (split-step schemes).
   1. high-order accurate factored scheme and boundary conditions.
   2. matrix-free multigrid.
   3. moving grid generation.

The Overture project is developing PDE solvers for a wide class of continuum mechanics applications.

Overture is a toolkit for solving PDE's on overlapping grids and includes CAD, grid generation, numerical approximations, AMR and graphics.

The CG (Composite Grid) suite of PDE solvers (cgcns, cgins, cgmx, cgsm, cgad, cgmp) provide algorithms for modeling gases, fluids, solids and E&M.

Overture and CG are available from www.llnl.gov/CASC/Overture.

We are looking at a variety of applications:

- wind turbines, building flows (cgins),
- explosives modeling (cgcns),
- fluid-structure interactions (e.g. blast effects) (cgmp+cgcns+cgsm),
- conjugate heat transfer (e.g. NIF holhraum) (cgmp+cgins+cgad),
- damage mitigation in NIF laser optics (cgmx).

Top 3 reasons for using overlapping grids.

1. Complex geometry and accuracy: You need to solve a PDE on a complex geometry and require accurate representations at boundaries (e.g. boundary layers).
2. Moving geometry : fast moving grid generation and high quality grids.
3. Efficiency: take advantage of fast and memory efficient algorithms for structured (and Cartesian) grids.
   - Example: 3D, 4th-order Maxwell : Cartesian grids are $25\times$ faster than curvilinear grids which themselves are $2-10??\times$ faster than unstructured grids.
   - Example: multigrid solvers for overlapping grids: can be an order of magnitude faster (e.g. $50\times$) and more memory efficient (e.g. $10\times$) than the best Krylov based solvers.

Top 3 reasons for using overlapping grids.

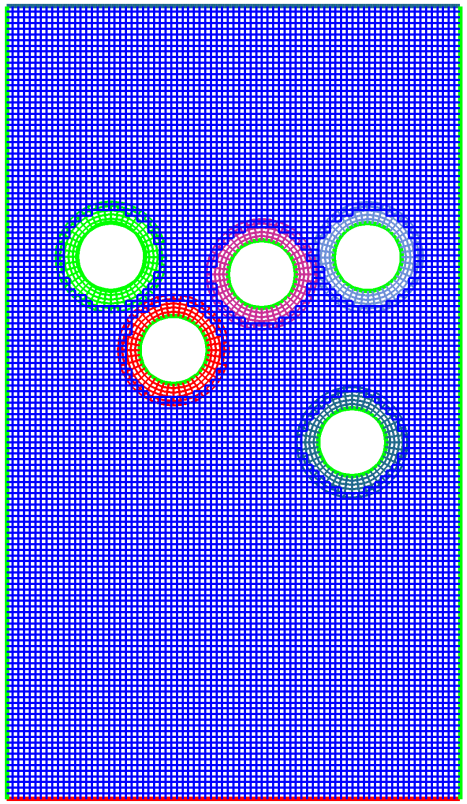
1. Complex geometry and accuracy: You need to solve a PDE on a complex geometry and require accurate representations at boundaries (e.g. boundary layers).
2. Moving geometry : fast moving grid generation and high quality grids.
3. Efficiency: take advantage of fast and memory efficient algorithms for structured (and Cartesian) grids.
   - Example: 3D, 4th-order Maxwell : Cartesian grids are $25\times$ faster than curvilinear grids which themselves are $2-10??\times$ faster than unstructured grids.
   - Example: multigrid solvers for overlapping grids: can be an order of magnitude faster (e.g. $50\times$) and more memory efficient (e.g. $10\times$) than the best Krylov based solvers.

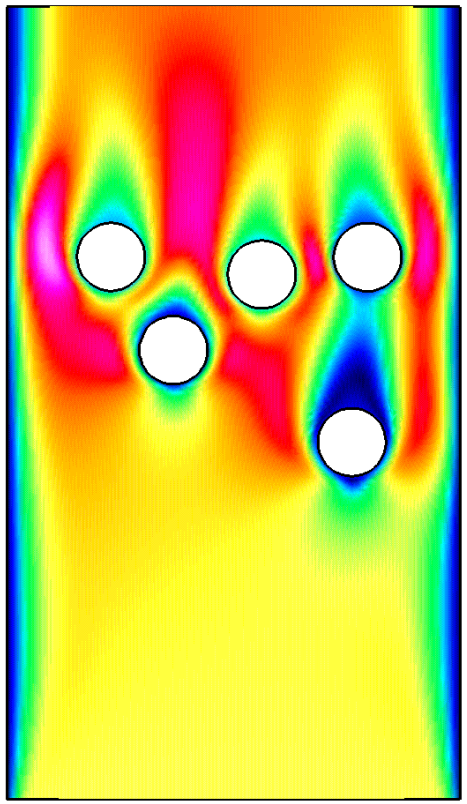
Top 3 reasons for using overlapping grids.

1. Complex geometry and accuracy: You need to solve a PDE on a complex geometry and require accurate representations at boundaries (e.g. boundary layers).
2. Moving geometry : fast moving grid generation and high quality grids.
3. Efficiency: take advantage of fast and memory efficient algorithms for structured (and Cartesian) grids.
   - Example: 3D, 4th-order Maxwell : Cartesian grids are $25\times$ faster than curvilinear grids which themselves are $2-10??\times$ faster than unstructured grids.
   - Example: multigrid solvers for overlapping grids: can be an order of magnitude faster (e.g. $50\times$) and more memory efficient (e.g. $10\times$) than the best Krylov based solvers.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

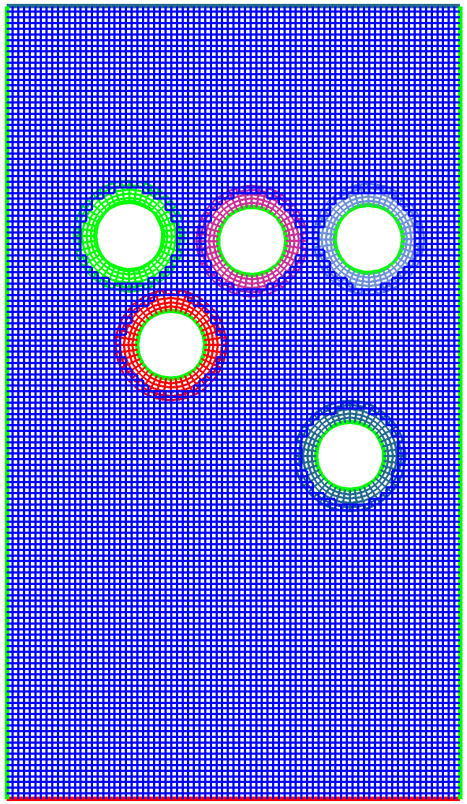


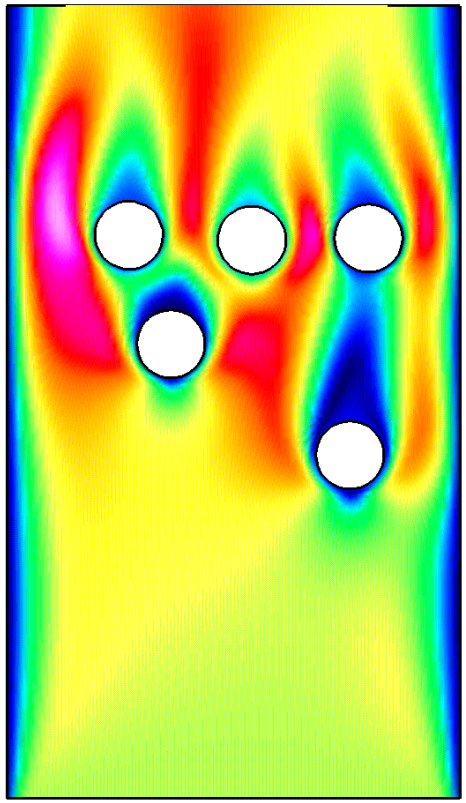


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

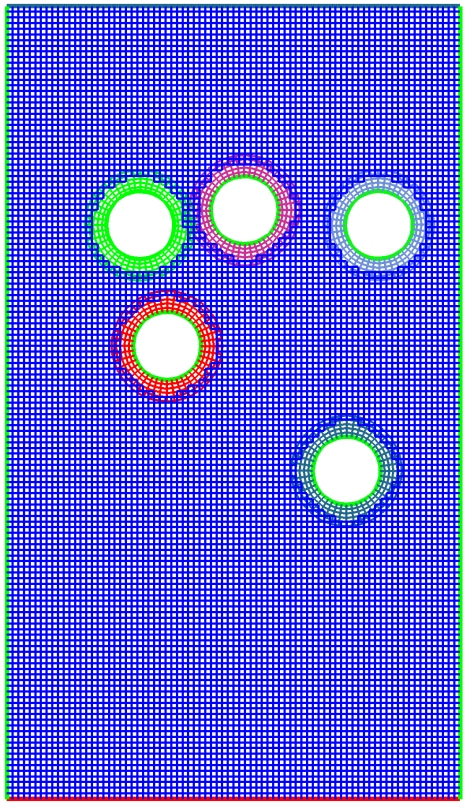


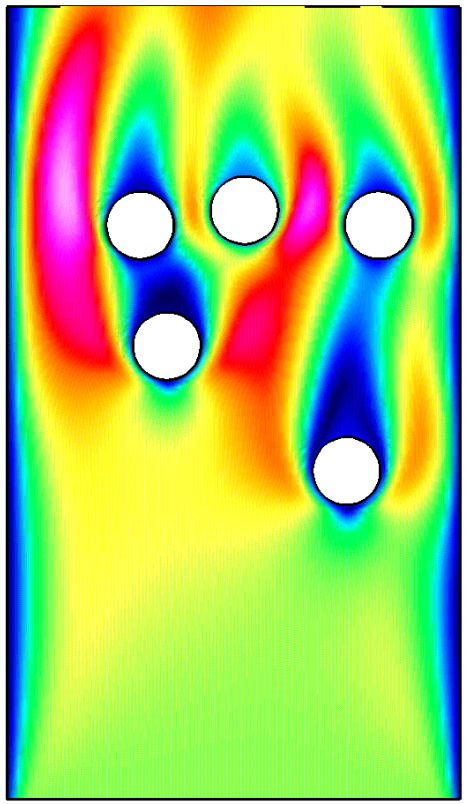


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

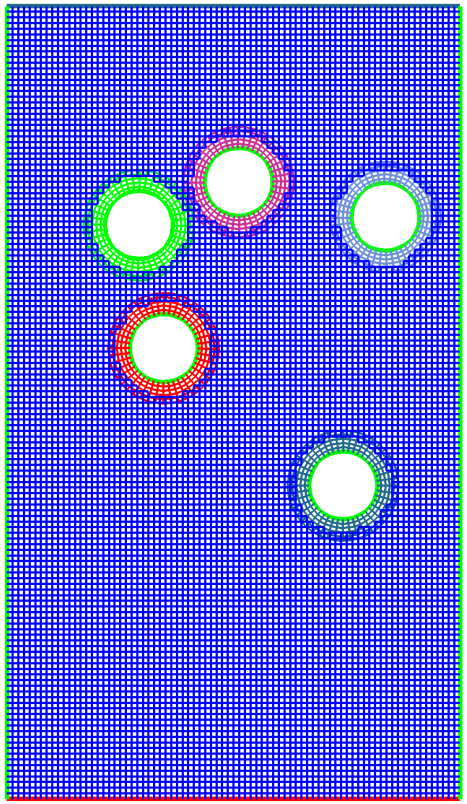


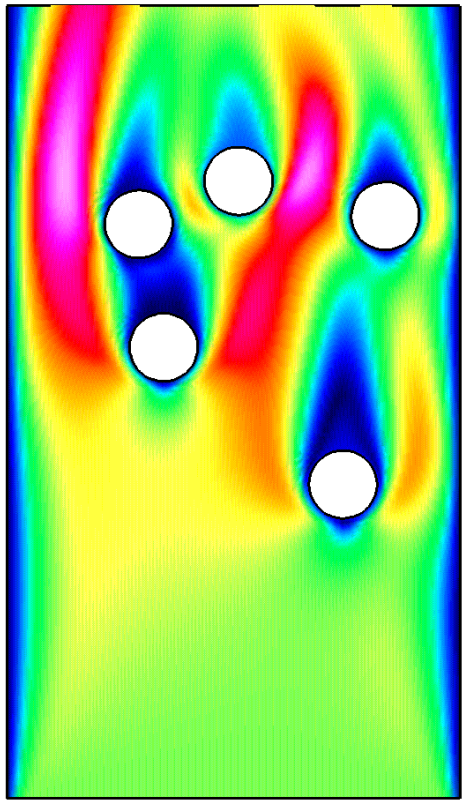


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

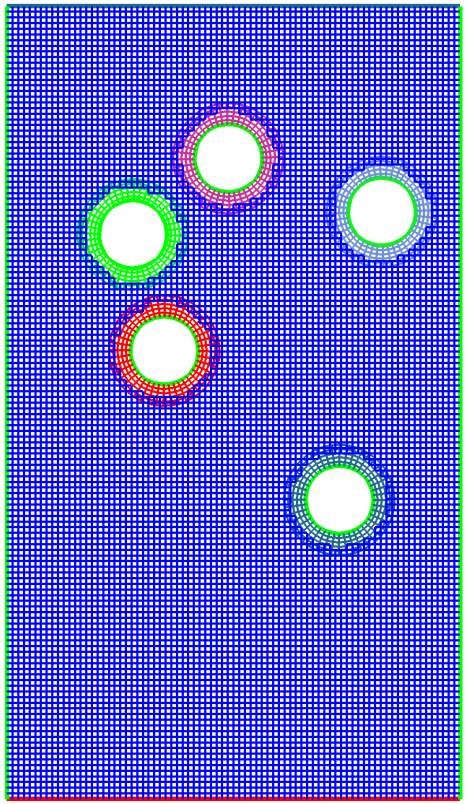


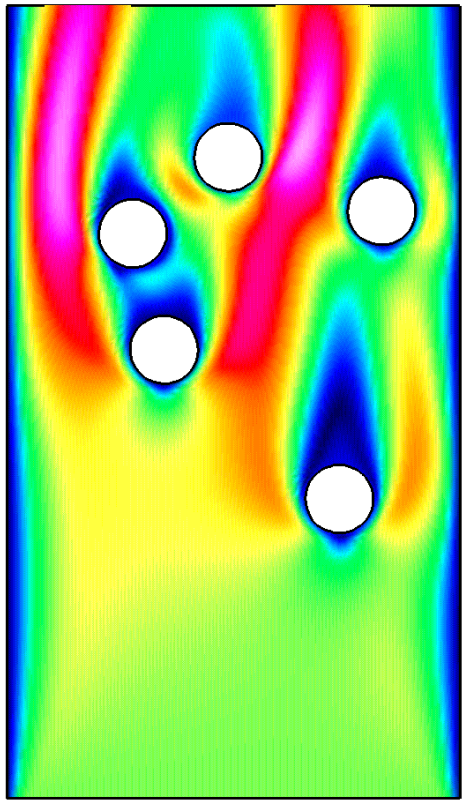


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

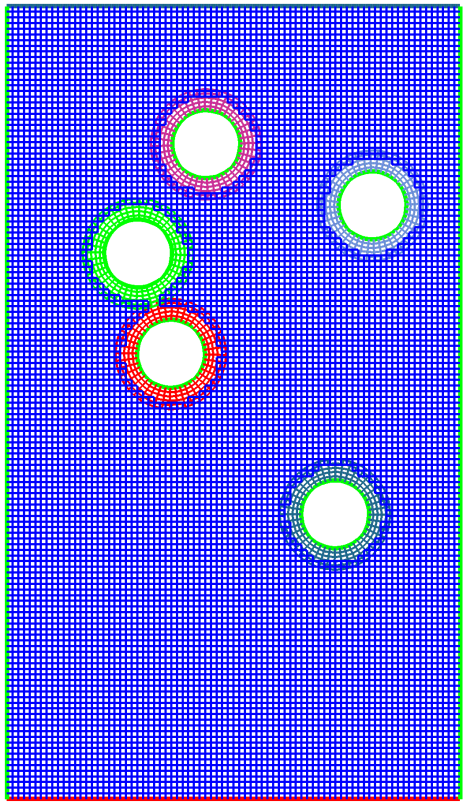


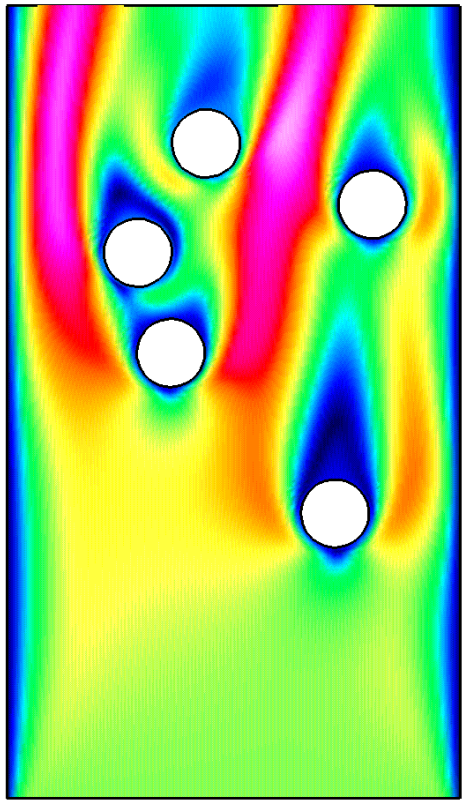


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

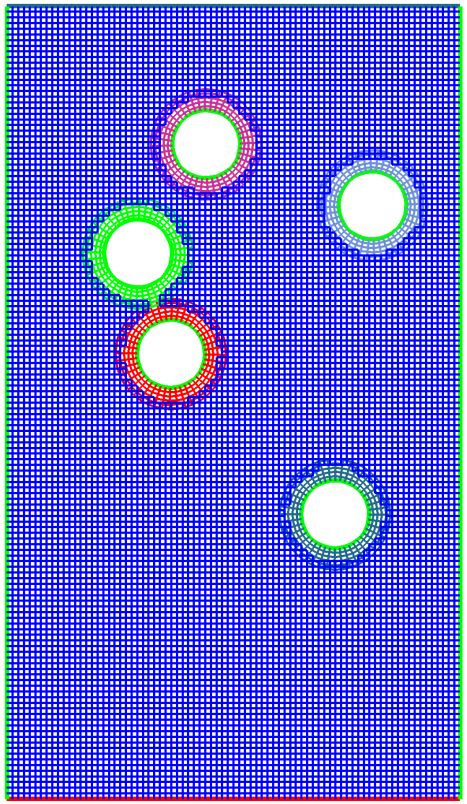


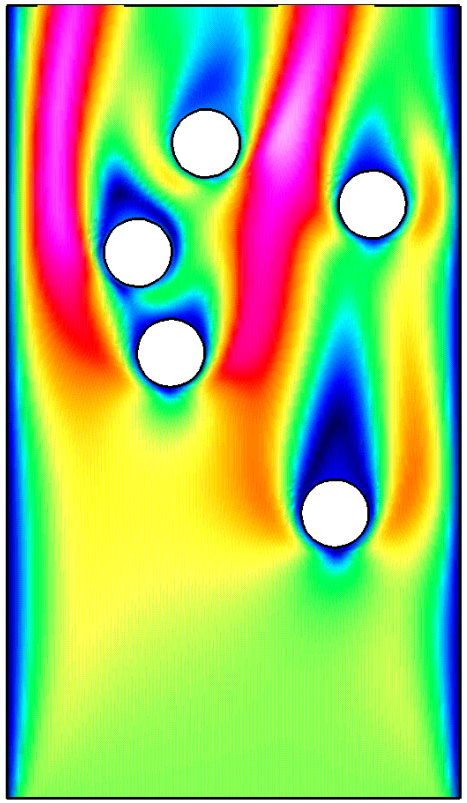


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.





- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Smooth grids for accuracy at boundaries.
- Efficient for high-order methods.

Asymptotic Performance Principle for overlapping grids

As grids are refined, total CPU/memory usage can approach that of a Cartesian grid.

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

Key Features of Overture

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities.
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

Different PDE solvers in the CG suite:

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

CG: Composite Grid PDE solvers built with Overture

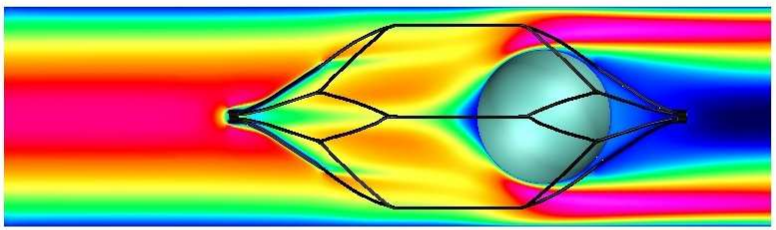
Different PDE solvers in the CG suite:

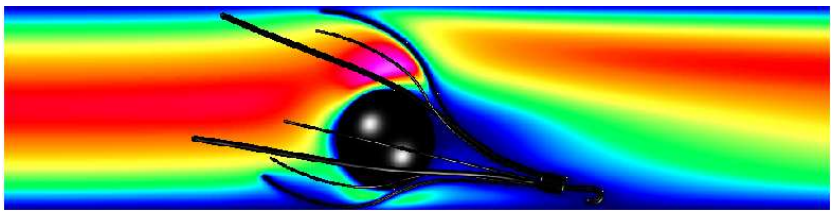
- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmx: time domain Maxwell's equations solver.
- cgsm: elastic wave equation (linear elasticity).
- cgmp: multi-physics solver (e.g. FSI, conjugate heat transfer).

Overture is used by research groups worldwide

Typical users are graduate students and University/Lab researchers.

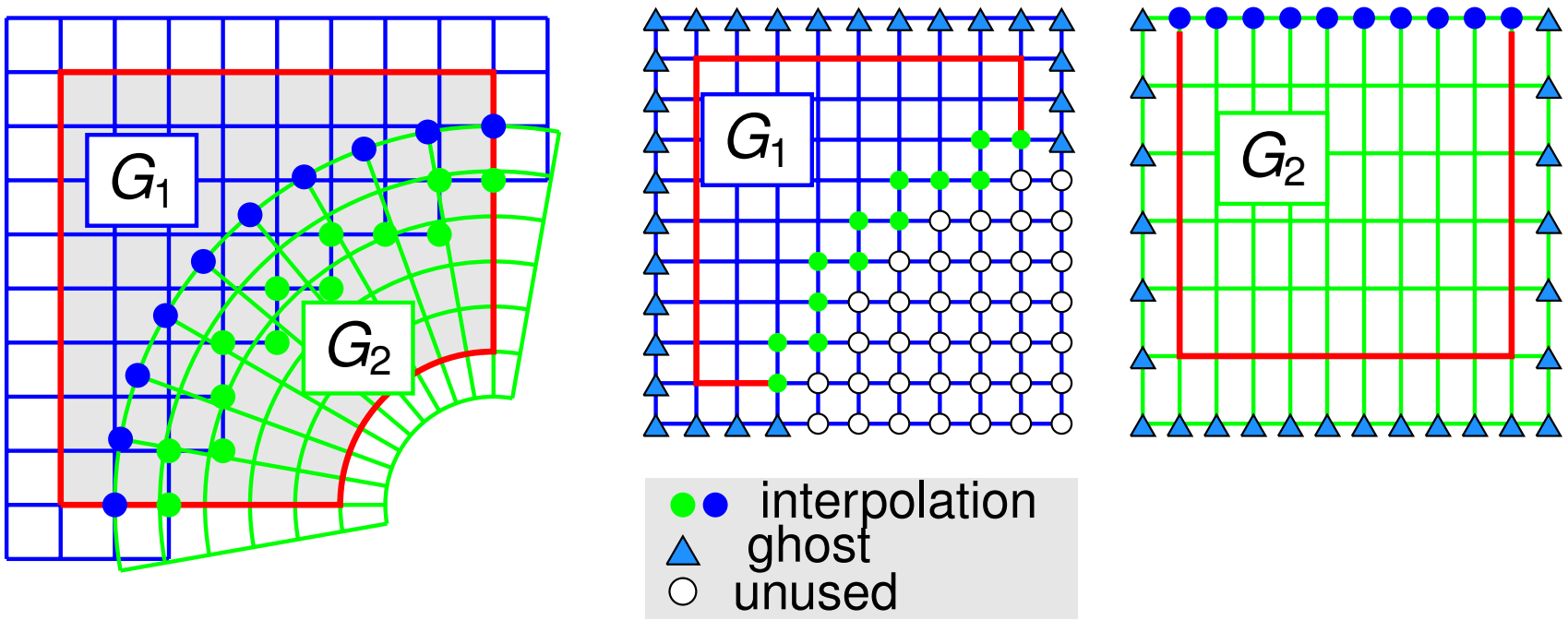
- Tear films and droplets (Dr. Kara Maki, IMA, and Prof. Richard Braun, U. Delaware).
- Blood flow and blood clot filters (Dr. Mike Singer).
- Flapping airfoils, mircro-air vehicles (Prof. Yongsheng Lian, U. of Louisville).
- Wave-energy devices (Dr. Robert Read, Prof. Harry Bingham, Technical U. of Denmark).
- Plasma physics (Dr. Jeff Banks, Dr. Richard Berger, LLNL).
- Flapping airfoils (Dr. Joel Guerrero, U. of Genoa).
- High-order accurate subsonic/transonic aero-acoustics (Dr. Philippe Lafon, CNRS, EDF).
- Elastic wave equation (Dr. Daniel Appelö, Caltech).
- Compressible flow/ice-formation (Graeme Leese, Prof. Nikos Nikiforakis, U. Cambridge).
- Relativistic hydrodynamics and Einstein field equations (Dr. Philip Blakely, U. Cambridge).
- Converging shock waves, shock focusing (Prof. Veronica Eliasson, USC).
- Pitching airfoils (Dr. D. Chandar, U. of Wyoming, Prof. M. Damodaran, NTU Singapore).
- Hypersonic flows for reentry vehicles, (Dr. Bjorn Sjögreen, LLNL, Dr. Helen Yee NASA).
- Acoustic lens design, shock lithotripsy (Prof. Andrew Szeri, UC Berkeley).
- High-order accurate, compact Hermite-Taylor schemes (Prof. Tom Hagstrom, SMU).
- High-order accurate aero-acoustics (Dr. Ramesh Balakrishnan, ANL).
- Incompressible flow in pumps (Dr. J.P. Potanza, Shell Oil).





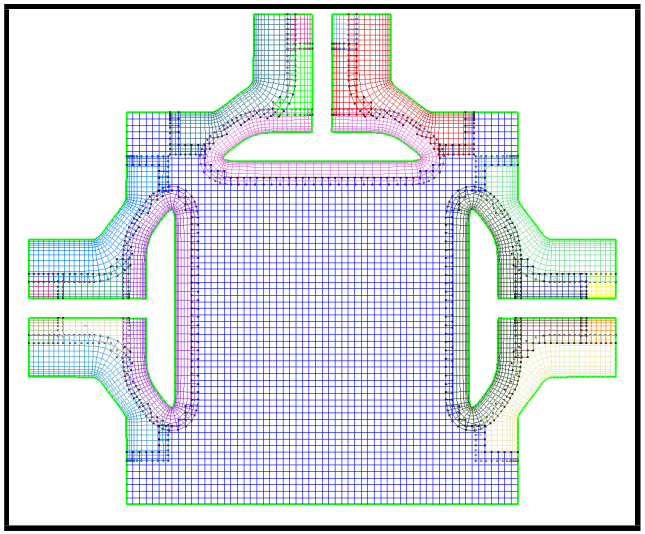
Blood flow past wire frame clot filters. M. Singer et.al.

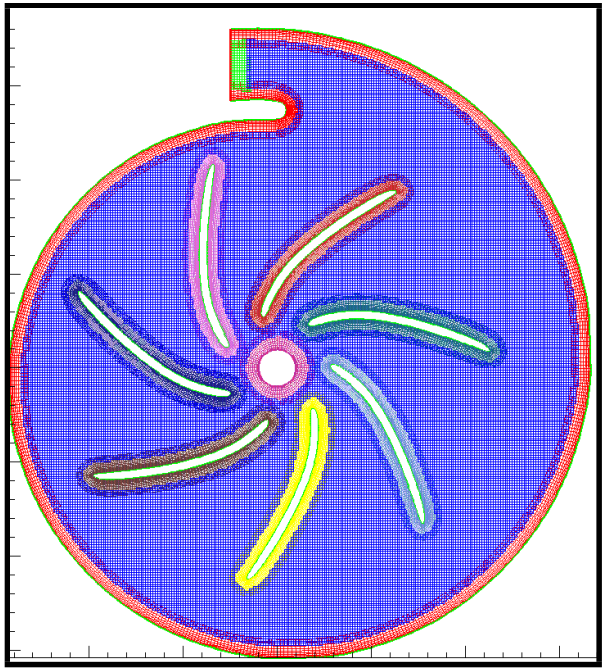
Components of an overlapping grid

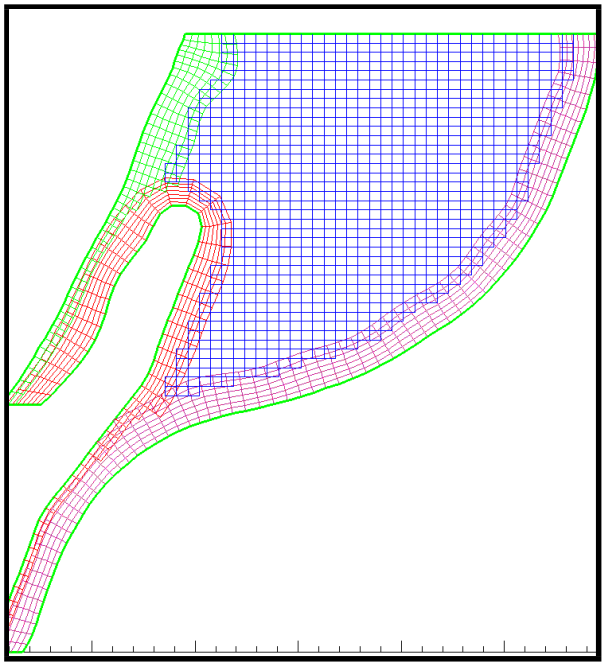


Left: an overlapping grid consisting of two structured curvilinear component grids, $\mathbf{x} = G_1(\mathbf{r})$ and $\mathbf{x} = G_2(\mathbf{r})$. Middle and right: component grids for the square and annular grids in the unit square parameter space $\mathbf{r}$. Grid points are classified as discretization points, interpolation points or unused points. Ghost points are used to apply boundary conditions.

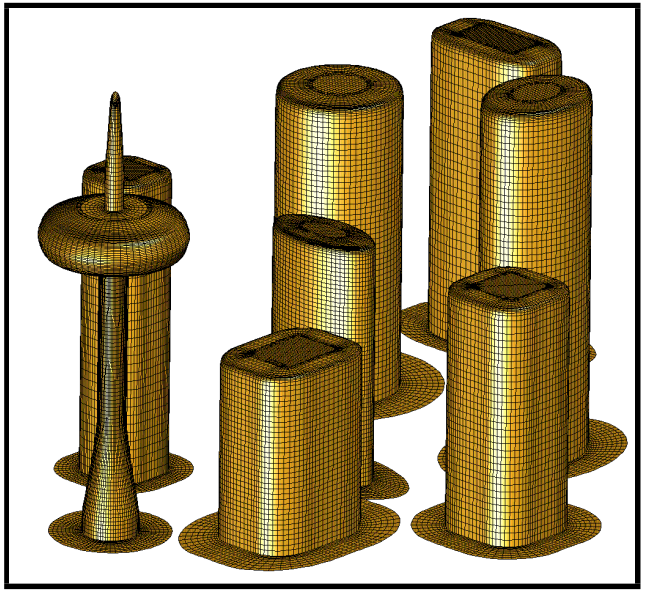
Sample 2D overlapping grids (Ogen).

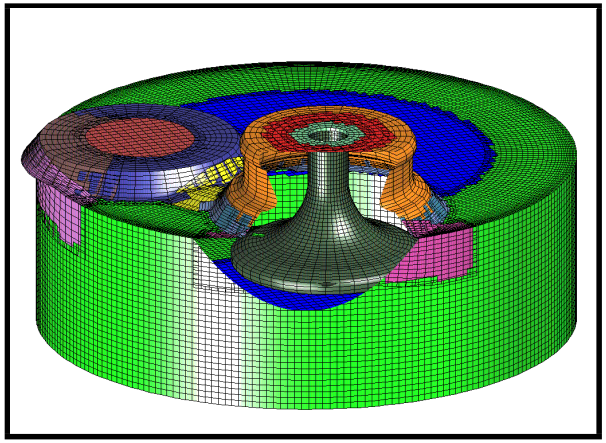


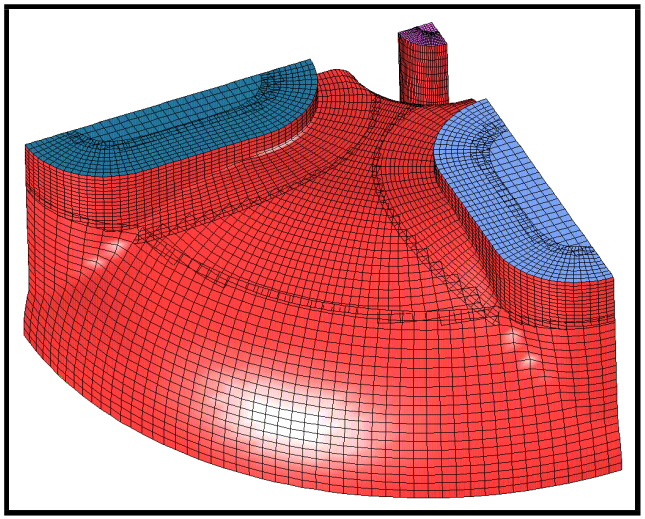


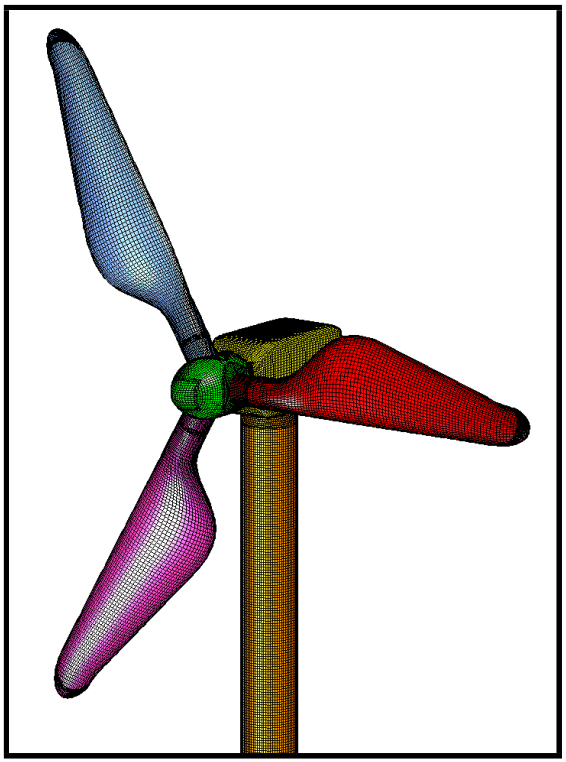


Sample 3D overlapping grids (Ogen).







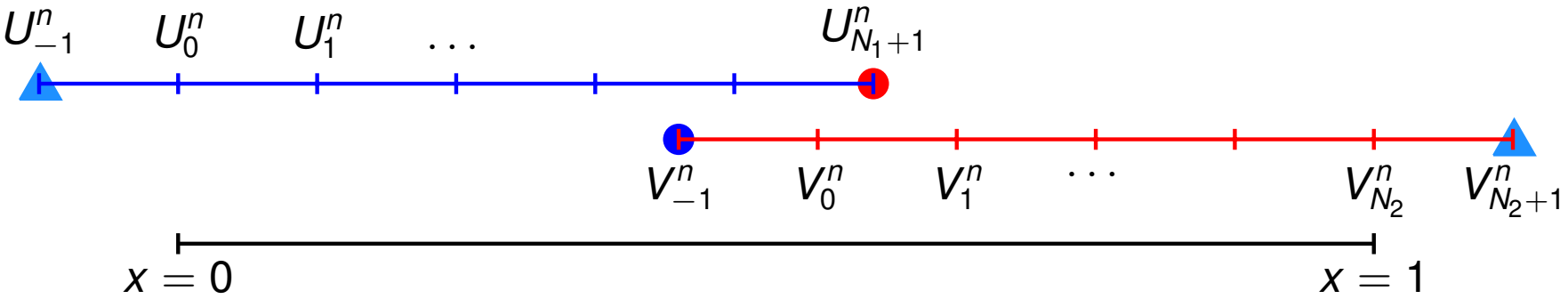


A one-dimensional overlapping grid example

To solve the advection-diffusion equation

$$
\begin{aligned}
u_t + a u_x &= \nu u_{xx}, && x \in (0,1) \\
u(0,t) = g_0(t), \quad u_x(1,t) &= g_1(t), && \text{(boundary conditions)} \\
u(x,0) &= u_0(x), && \text{(initial conditions)}
\end{aligned}
$$

introduce the grid functions $U_i^n \approx u(x_i^{(1)}, n\Delta t)$, $V_i^n \approx u(x_i^{(2)}, n\Delta t)$, and the overlapping grid:



How to advance the solution on an overlapping grid.

(1) interior equations, (2) boundary conditions, (3) interpolation points.

Given the solution at time t^n, compute the solution at time t^{n+1}:

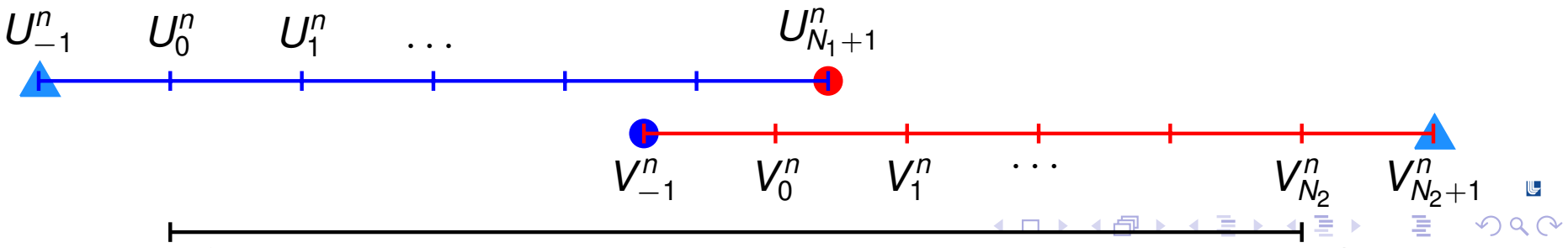
$$(U_i^{n+1} - U_i^n)/\Delta t = -a\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x} + \nu\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}, \qquad i = 1, 2, \ldots, N_1$$

$$(V_j^{n+1} - V_j^n)/\Delta t = -a\frac{V_{i+1}^n - V_{i-1}^n}{2\Delta x} + \nu\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta x^2}, \qquad j = 0, 2, \ldots, N_2$$

$$U_0^{n+1} = g(t^n), \quad D_0 V_{N_2}^{n+1} = g_1(t^{n+1}), \qquad \text{(boundary conditions)}$$

$$U_{N_1+1}^{n+1} = (1-\alpha)(1-\frac{\alpha}{2})\, V_{-1}^{n+1} + \alpha(2-\alpha)\, V_0^{n+1} + \frac{\alpha}{2}(\alpha-1)\, V_1^{n+1}, \quad \text{(interpolation)}$$

$$V_{-1}^{n+1} = (1-\beta)(1-\frac{\beta}{2})\, U_{N_1-1}^{n+1} + \beta(2-\beta)\, U_{N_1}^{n+1} + \frac{\beta}{2}(\beta-1)\, U_{N_1+1}^{n+1}, \quad \text{(interpolation)}$$



How to advance the solution on an overlapping grid.

(1) interior equations, (2) boundary conditions, (3) interpolation points.

Given the solution at time t^n, compute the solution at time t^{n+1}:

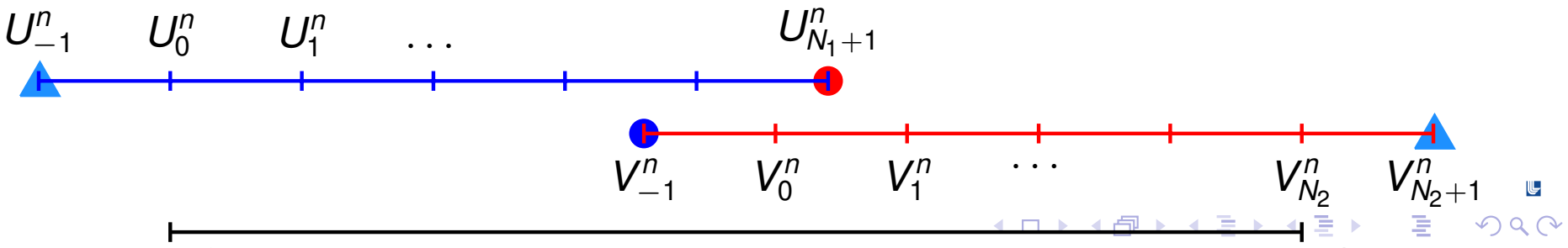
$$(U_i^{n+1} - U_i^n)/\Delta t = -a\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x} + \nu\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}, \qquad i = 1, 2, \ldots, N_1$$

$$(V_j^{n+1} - V_j^n)/\Delta t = -a\frac{V_{i+1}^n - V_{i-1}^n}{2\Delta x} + \nu\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta x^2}, \qquad j = 0, 2, \ldots, N_2$$

$$U_0^{n+1} = g(t^n), \quad D_0 V_{N_2}^{n+1} = g_1(t^{n+1}), \qquad \text{(boundary conditions)}$$

$$U_{N_1+1}^{n+1} = (1-\alpha)(1-\frac{\alpha}{2})\, V_{-1}^{n+1} + \alpha(2-\alpha)\, V_0^{n+1} + \frac{\alpha}{2}(\alpha-1)\, V_1^{n+1}, \quad \text{(interpolation)}$$

$$V_{-1}^{n+1} = (1-\beta)(1-\frac{\beta}{2})\, U_{N_1-1}^{n+1} + \beta(2-\beta)\, U_{N_1}^{n+1} + \frac{\beta}{2}(\beta-1)\, U_{N_1+1}^{n+1}, \quad \text{(interpolation)}$$



How to advance the solution on an overlapping grid.

(1) interior equations, (2) boundary conditions, (3) interpolation points.

Given the solution at time t^n, compute the solution at time t^{n+1}:

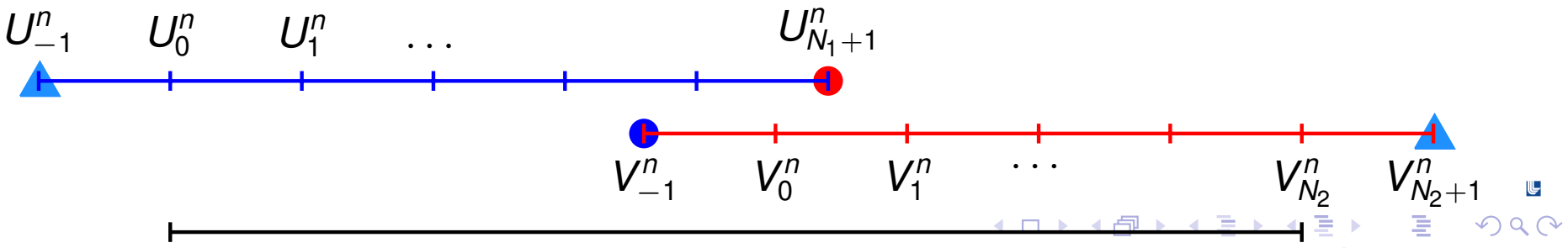
$$(U_i^{n+1} - U_i^n)/\Delta t = -a\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x} + \nu\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}, \qquad i = 1, 2, \ldots, N_1$$

$$(V_j^{n+1} - V_j^n)/\Delta t = -a\frac{V_{i+1}^n - V_{i-1}^n}{2\Delta x} + \nu\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta x^2}, \qquad j = 0, 2, \ldots, N_2$$

$$U_0^{n+1} = g(t^n), \quad D_0 V_{N_2}^{n+1} = g_1(t^{n+1}), \qquad \text{(boundary conditions)}$$

$$U_{N_1+1}^{n+1} = (1-\alpha)(1-\frac{\alpha}{2})\ V_{-1}^{n+1} + \alpha(2-\alpha)\ V_0^{n+1} + \frac{\alpha}{2}(\alpha-1)\ V_1^{n+1}, \quad \text{(interpolation)}$$

$$V_{-1}^{n+1} = (1-\beta)(1-\frac{\beta}{2})\ U_{N_1-1}^{n+1} + \beta(2-\beta)\ U_{N_1}^{n+1} + \frac{\beta}{2}(\beta-1)\ U_{N_1+1}^{n+1}, \quad \text{(interpolation)}$$



Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg; // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg); // create a grid function
u=1.;
CompositeGridOperators op(cg); // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg; // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg); // create a grid function
u=1.;
CompositeGridOperators op(cg); // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg; // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg); // create a grid function
u=1.;
CompositeGridOperators op(cg); // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg; // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg); // create a grid function
u=1.;
CompositeGridOperators op(cg); // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg; // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg); // create a grid function
u=1.;
CompositeGridOperators op(cg); // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
   u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
   t+=dt;
   u.interpolate();
   u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
   u.finishBoundaryConditions();
}
```

Incompressible flows and moving bodies.

Challenges:

- efficient solvers (esp. since grids change at every time step).
- high-order accuracy including the boundary.
- efficient grid generation.

Incompressible Navier-Stokes.

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad t > 0, \quad \mathbf{x} \in \Omega$$
$$\nabla\cdot\mathbf{u} = 0 \qquad t > 0, \quad \mathbf{x} \in \Omega$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \text{ (pressure BC)} \quad \mathbf{x} \in \partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,) + \mathbf{n}\cdot\mathbf{f}.$$

Use $-\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \text{ (pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,) + \mathbf{n}\cdot\mathbf{f}.$$

Use $-\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \text{ (pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,) + \mathbf{n}\cdot\mathbf{f}.$$

Use $-\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \text{ (pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,) + \mathbf{n}\cdot\mathbf{f}.$$

Use $-\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} = 0, \qquad & t > 0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} \;=\; 0, \quad \nabla\cdot\mathbf{u} = 0, \text{ (pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n \;=\; -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,) + \mathbf{n}\cdot\mathbf{f}.$$

Use $-\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

PDE based numerical boundary conditions (NBCs)

- High-order FD schemes with wide stencils require additional NBCs.
- One-sided difference approximations can be less stable and accurate.

NBC Principle

Derive NBCs from the PDE and physical boundary conditions.

Example (heat equation):

$$
\begin{aligned}
&u_t = \nu(u_{xx} + u_{yy}) + f(x,y,t), \quad 0 < x < 1, \quad -\infty < y < \infty, \\
&u(0,y,t) = g(y,t), \qquad \text{(Dirichlet boundary condition)}, \\
&u_x(1,y,t) = k(y,t). \qquad \text{(Neumann boundary condition)},
\end{aligned}
$$

Compatibility conditions are used as NBCs:

$$
\begin{aligned}
g_t(y,t) &= \nu(u_{xx}(0,y,t) + g_{yy}(0,y,t)) + f(0,y,t), \\
k_t(y,t) &= \nu(u_{xxx}(1,y,t) + k_{yy}(1,y,t)) + f_x(1,y,t).
\end{aligned}
$$

Normal mode theory shows why compatibility conditions can be applied to lower order accuracy.

PDE based numerical boundary conditions (NBCs)

- High-order FD schemes with wide stencils require additional NBCs.
- One-sided difference approximations can be less stable and accurate.

NBC Principle

Derive NBCs from the PDE and physical boundary conditions.

Example (heat equation):

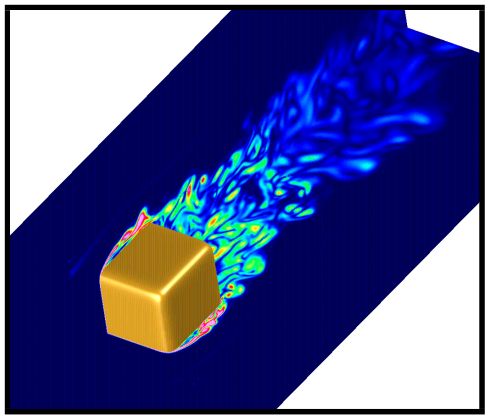
$$
\begin{aligned}
&u_t = \nu(u_{xx} + u_{yy}) + f(x,y,t), \quad 0 < x < 1, \quad -\infty < y < \infty, \\
&u(0,y,t) = g(y,t), \qquad \text{(Dirichlet boundary condition)}, \\
&u_x(1,y,t) = k(y,t). \qquad \text{(Neumann boundary condition)},
\end{aligned}
$$

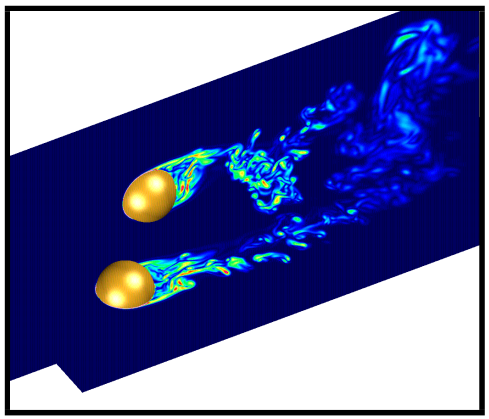
Compatibility conditions are used as NBCs:

$$
\begin{aligned}
g_t(y,t) &= \nu(u_{xx}(0,y,t) + g_{yy}(0,y,t)) + f(0,y,t), \\
k_t(y,t) &= \nu(u_{xxx}(1,y,t) + k_{yy}(1,y,t)) + f_x(1,y,t).
\end{aligned}
$$

Normal mode theory shows why compatibility conditions can be applied to lower order accuracy.

Cgins: incompressible Navier-Stokes solver.





- 2nd-order and 4th-order accurate (DNS and LES).
- accurate and stable treatment of boundaries.
- support for moving rigid-bodies.
- heat transfer and buoyancy (Boussinesq approx.).
- semi-implicit (time accurate), pseudo steady-state (efficient line solver), full implicit.
- SSLES nonlinear LES turbulence model.

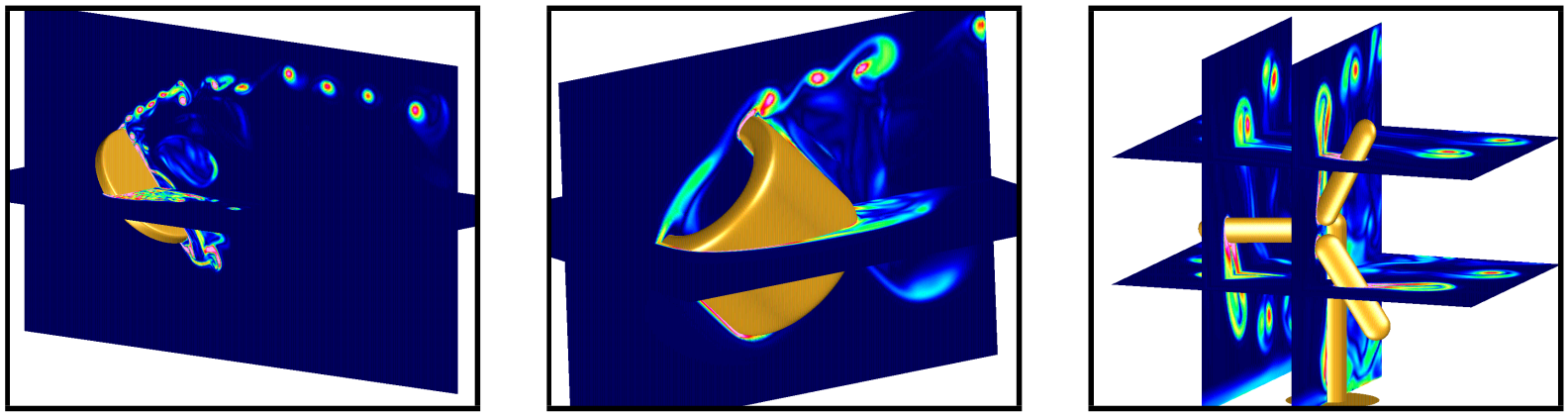
- WDH., *A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids*, J. Comput. Phys, **113**, no. 1, (1994) 13–25.
- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.
- WDH., K. K. Chand, *A Composite Grid Solver for Conjugate Heat Transfer in Fluid-Structure Systems*, J. Comput. Phys, 2009.

Cgins: New 4th-order AFS-MG parallel solver.

Approximate-factored/compact scheme and multigrid pressure solver

A parallel split-step solver is being developed based on:

1. Fourth-order accurate approximate-factored/compact time-stepping scheme for the momentum equations.
2. Fourth-order accurate multigrid solver for the pressure equation.
3. Fast overlapping grid generation for moving geometry.



Parallel moving grid computations.

- K.K. Chand and M.A. Singer, *Verification and validation of CgWind: a high-order accurate simulation tool for wind engineering*, 13th International Conference on Wind Engineering (ICWE13), 2011.
- K.K. Chand, WDH, K.A. Lundquist and M.A. Singer, *CgWind: A high-order accurate simulation tool for wind turbines and wind farms*, The Fifth International Symposium on Computational Wind Engineering (CWE2010), 2010.

Approximate factorization & compact discretizations

A key point is maintaining accuracy at boundaries.

- Approximate factorization (AF) schemes offer larger timesteps with second order accuracy in time:

$$(I + \frac{\Delta t}{2}(A+B))U^{n+1} = (I - \frac{\Delta t}{2}(A+B))U^{n}$$

becomes

$$(I + \frac{\Delta t}{2}A)(I + \frac{\Delta t}{2}B)U^{n+1} \quad = \quad (I - \frac{\Delta t}{2}A)(I - \frac{\Delta t}{2}B)U^{n}$$

- Compact schemes can be integrated into the AF solves
- Special "combined" compact schemes have been developed:
  $\longrightarrow$ reduce the number of factors

$$(a\partial_r + b\partial_{rr}^2)u \rightarrow P^{-1}(D_r a + D_{rr} b)u + \text{corrections}$$

  $\longrightarrow$ preserve accuracy at boundaries
  $\longrightarrow$ 4^{th} and 6^{th} order accuracy with a 5 point stencil
  $\longrightarrow$ special penta-diagonal solvers that handle wider boundary stencils

Smallest Scale LES (SSLES) model

- Our high-order accurate schemes all use central differences.
- Nonlinear dissipation is used to stablize the schemes.

It can be proved [HKR] that the minimum scale of the INS locally satisfies

$$\lambda_{\min} \propto \sqrt{\frac{\nu}{\|\nabla \mathbf{u}\|_{\text{loc}} + c}}.$$

Setting $h = \lambda_{\min}$ leads to the non-linear dissipation operators:

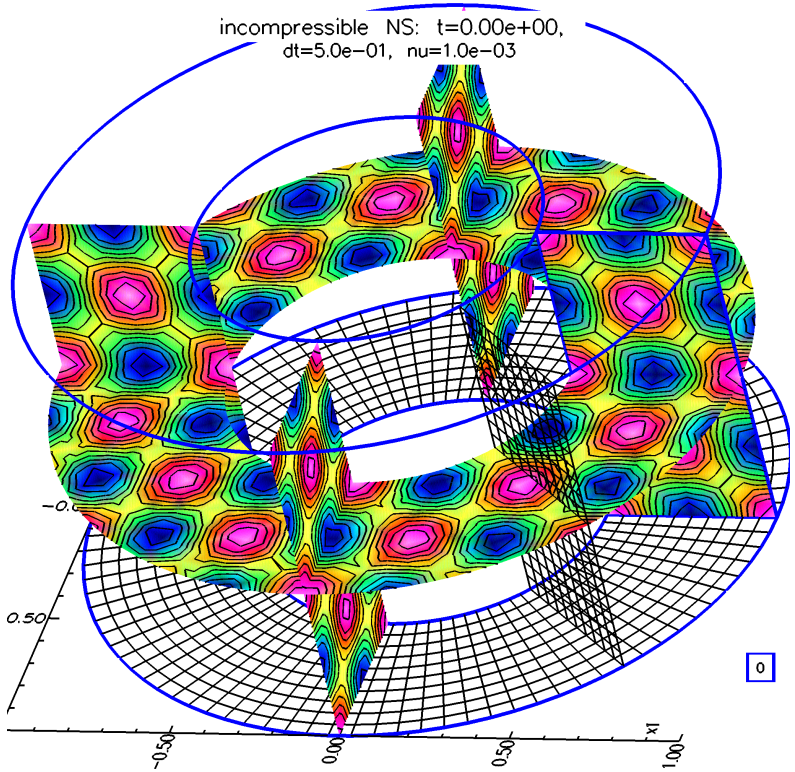
$$\mathcal{D}_2(\mathbf{u_i}) = (a_{21} + a_{22}\|\nabla_h \mathbf{u_i}\|)h^2 \Delta_h \mathbf{u_i},$$
$$\mathcal{D}_4(\mathbf{u_i}) = (a_{41} + a_{42}\|\nabla_h \mathbf{u_i}\|)h^4(-\Delta_h^2)\mathbf{u_i}.$$

1. These serve as simple LES models (Smagorinsky type).
2. These terms stabilize the scheme even if $\nu = 0$.

[HKR1] *On the smallest scale for the incompressible Navier-Stokes equations*, WDH, H.-O. Kreiss, L.G.M. Reyna, Theoret. Comput. Fluid Dynamics, 1989.
[HKR2] *Smallest scale estimates for the incompressible Navier-Stokes equations*, WDH, H.-O. Kreiss, L.G.M. Reyna, Arch. Rational Mech. Anal., 1990.

Code verification: an integral part of our development

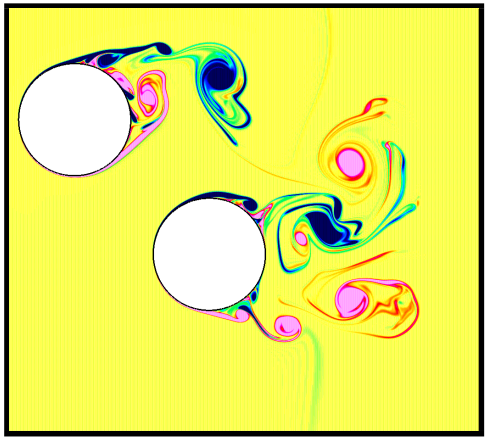
Twilight zone solutions (a.k.a. manufactured solutions) are extensively supported by the Overture framework.



h_{max}	$\|e_p\|_\infty$	$\|e_u\|_\infty$	$\|e_v\|_\infty$	$\|e_w\|_\infty$	$\|\nabla \cdot \mathbf{u}\|_\infty$
1.34e-01	1.91e-01	3.42e-01	2.14e-01	1.23e-01	2.80e+00
6.68e-02	1.20e-02 (15.9)	1.84e-02 (18.6)	1.01e-02 (21.2)	6.43e-03 (19.1)	2.71e-01 (10.3)
3.34e-02	1.02e-03 (11.8)	1.49e-03 (12.3)	7.50e-04 (13.5)	3.01e-04 (21.4)	3.10e-02 (8.7)
1.67e-02	7.90e-05 (12.9)	1.25e-04 (11.9)	6.55e-05 (11.5)	1.74e-05 (17.3)	4.00e-03 (7.8)
rate	3.7	3.8	3.9	4.3	3.1

AFS performance: flow past two cylinders.

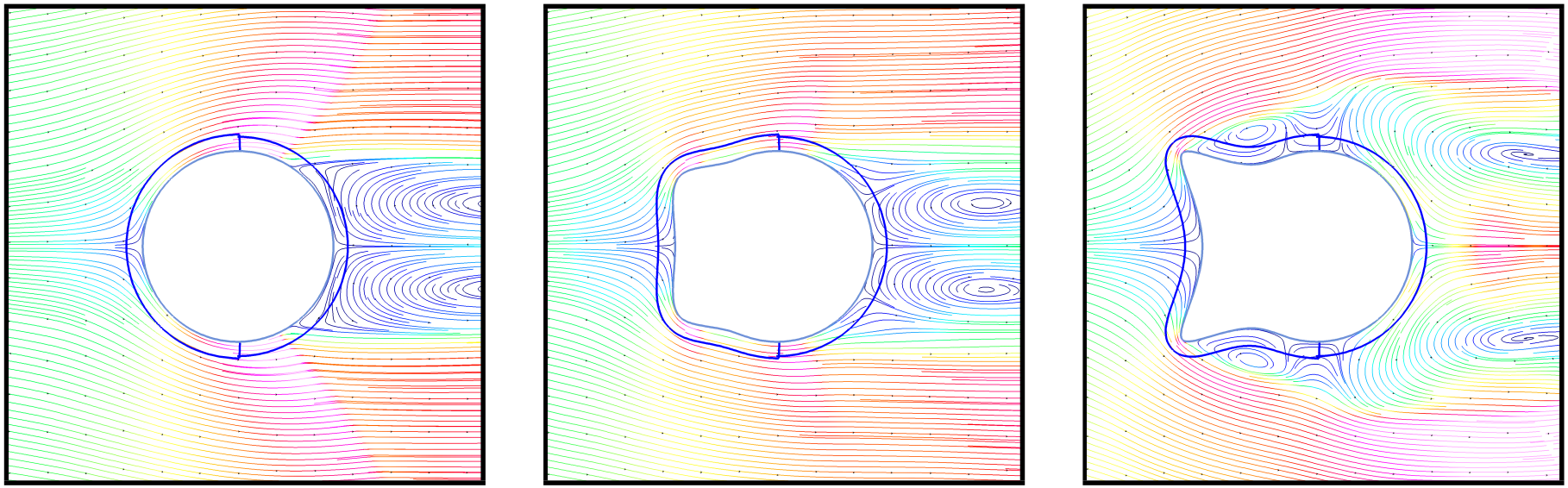
Grid tcilce64, $N_g = 22$M grid-points					
Method	P-solver	N_p	TPSM	TTS	RPG
PC24	BICGS(5)	16	840	16000	205
PC24	MG	16	15	210	67
AFS24	MG	16	34	54	32



1. Speed-up from baseline PC24-BICGS(5) to AFS24-MG is $16000/54 \approx 300$.
2. Memory is reduced by a factor of $205/32 \approx 6$.

N_p : number of processors.
N_g : number of grid-points.
TPS : CPU (s) time-per-step.
TPSM : CPU time (s) per-step, per-million-grid-points, per-processor = $N_p \times TPS/(N_g/10^6)$.
TTS : normalized time-to-solution = TPSM / CFL-number.
RPG : memory usage in real-per-grid-point.
PC24: predictor-corrector, 2nd-order time, 4th-order space.
AFS24: approximate-factored scheme, 2nd-order time, 4th-order space.
BICGS(5) : bi-Conjugate-Gradient-Stabilized with ILU(5) preconditionner.

Modeling deforming geometry with overlapping grids



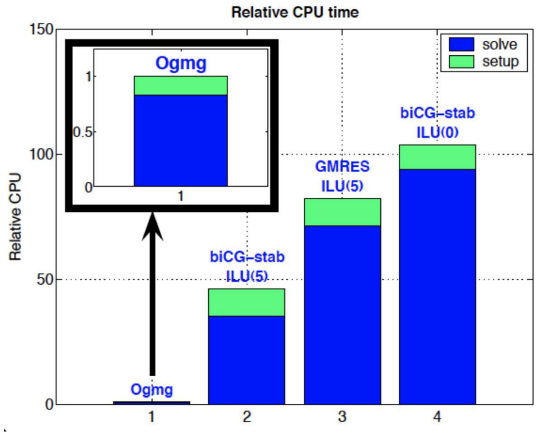
1. interface deforms over time.
2. local body fitted grids are regenerated at each time step with the hyperbolic grid generator.
3. equations are solved in the moving coordinate frame.

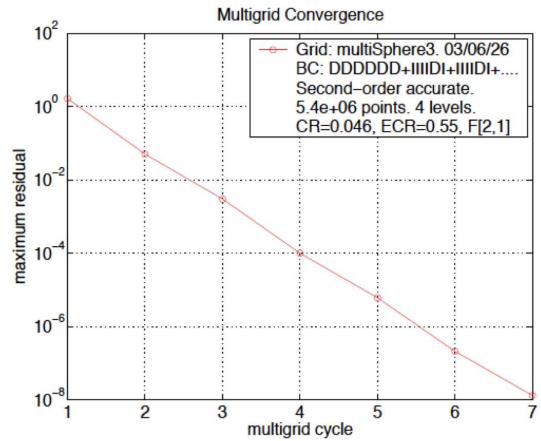
Fourth-order Accurate Parallel Multigrid Solver for the Pressure Equation.

The Ogmg overlapping grid multigrid solver has been extended to 4th-order accuracy and parallel.

Ogmg is many times faster than Krylov methods.

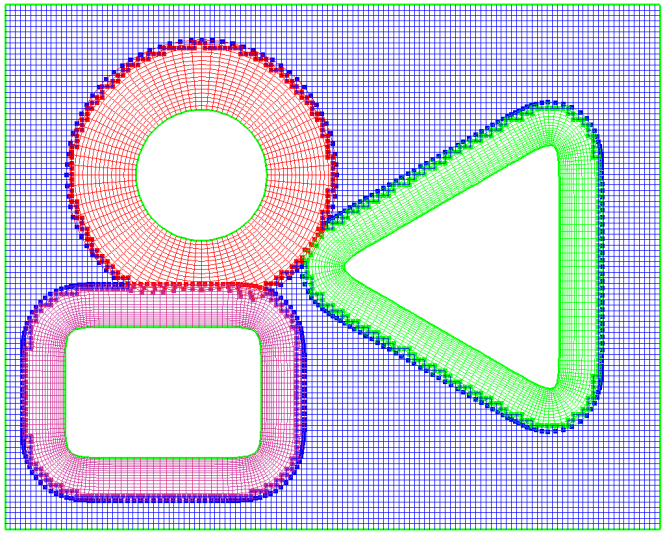
- matrix-free; optimized for Cartesian grids.
- automatic coarse grid generation.
- adaptive smoothing
  - variable sub-smooths per component grid.
  - interpolation-boundary smoothing (IBS).
- Galerkin coarse grid operators (operator averaging).
- *PDE-based* numerical boundary conditions for Dirichlet and Neumann problems.

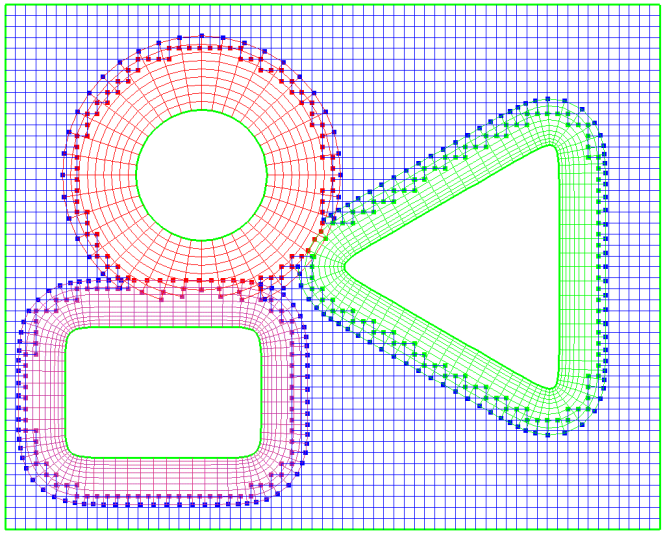


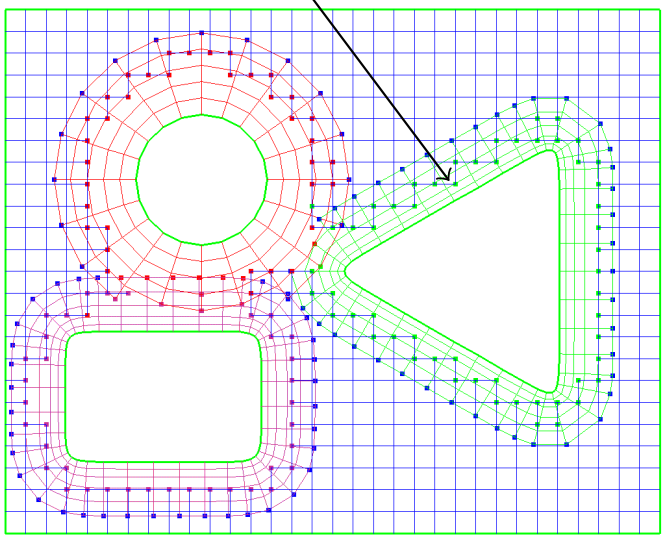


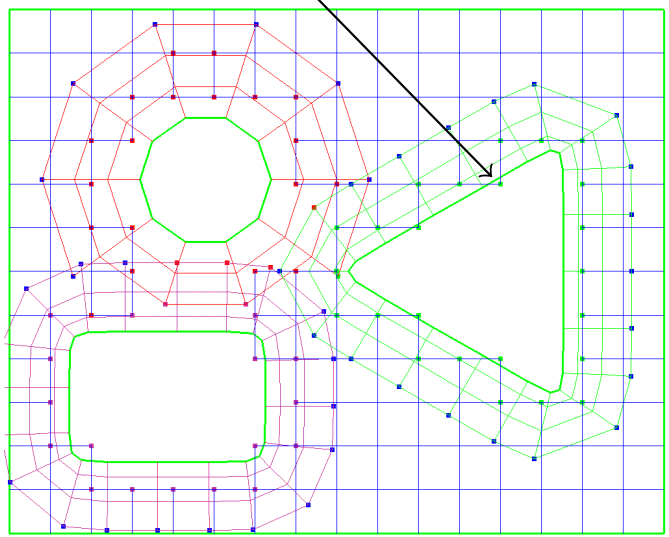
WDH., *On Multigrid For Overlapping Grids*, SIAM J. Sci. Comput. **26**, no. 5, (2005) 1547–1572.

Automatic coarse grid generation is a key feature.



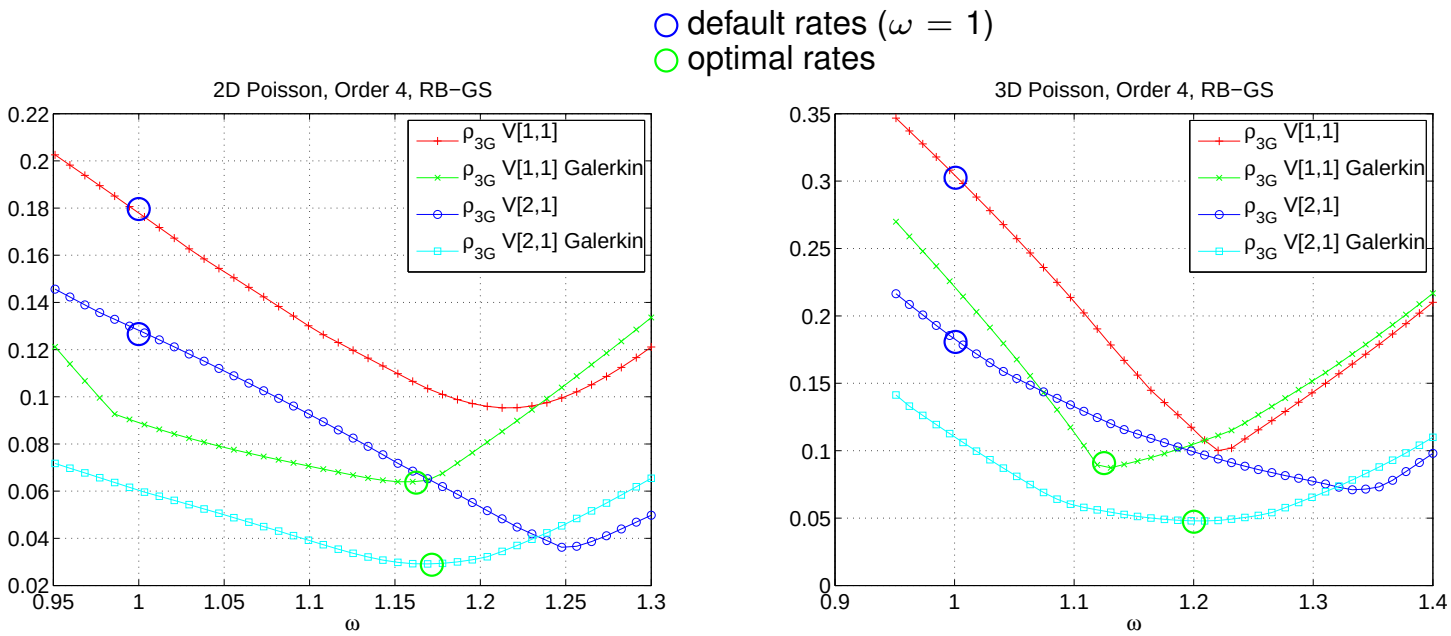






Local Fourier analysis significantly improves convergence rates.

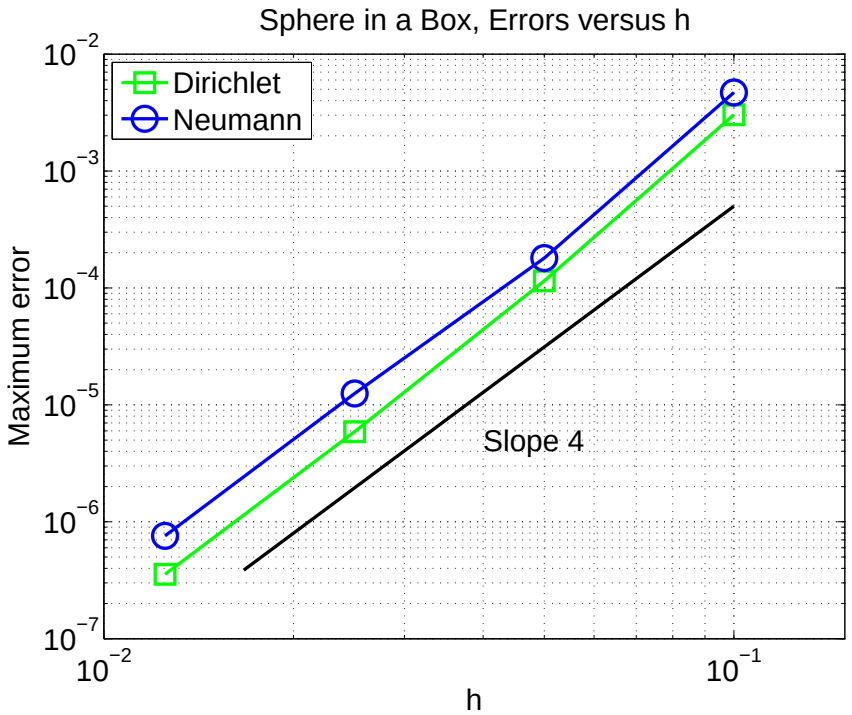
Over-relaxed Red-Black smoothers and Galerkin coarse grid operators.

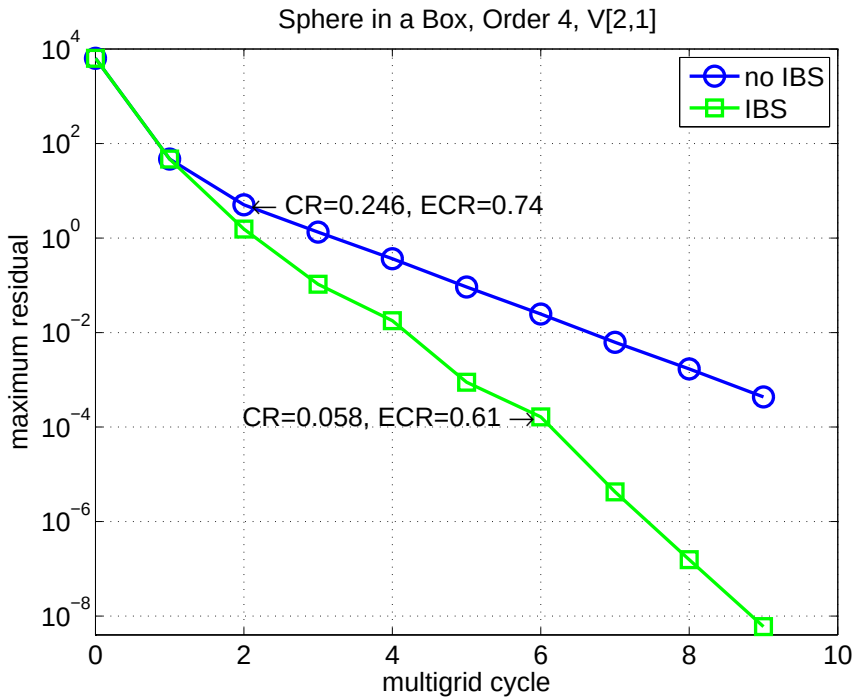


Three-grid multigrid convergence rates as a function ω.

ω : relaxation parameter in Red-Black Gauss-Seidel smoother.
ρ_{3G} : convergence rate per cycle for a 3 grid (i.e. 3 level) MG.
$V[m, n]$: MG V-cycle, m pre-smooths and n-post smooths.
Galerkin : Galerkin coarse grid operators.

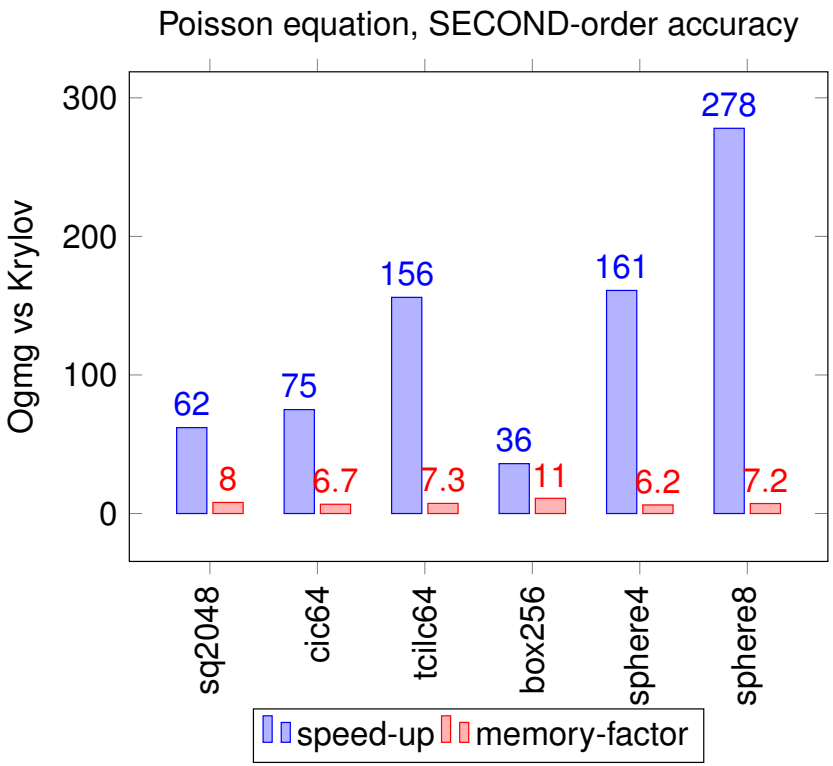
Accuracy and convergence of the new fourth-order accurate parallel version of Ogmg.

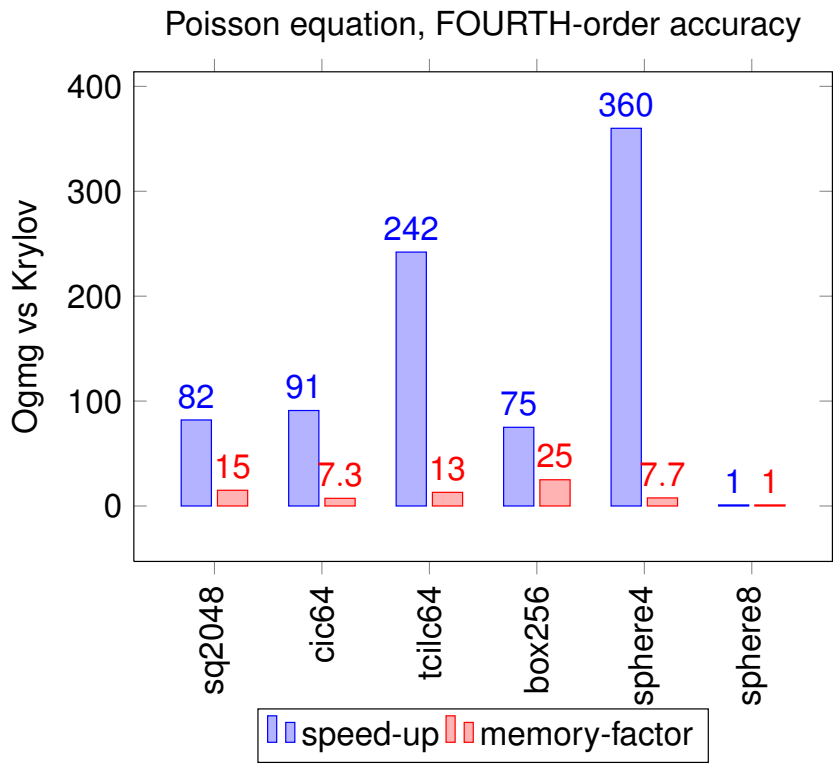




Multigrid is much faster than Krylov based methods.

And uses much less memory.





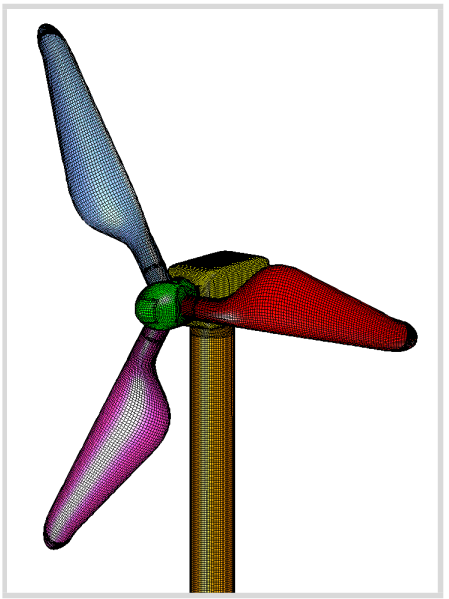
Performance of Ogmg for solving Poisson's equation to on various grids. Ogmg is compared to a Krylov solver (bi-CG stab, ILU(1)) from PETSc.

Rapid grid generation for moving geometry is critical.

Ogen is the overlapping grid generator in Overture.

Overlapping grid generation consists of two major steps:

1. construct component grids (Mappings).
2. grid connectivity: cut holes and determine interpolation information using Ogen (this is the step that requires most of the CPU time).



In recent work changes have been made to support

- the generation of large (billion point +) grids.
- parallel moving-grid flow simulations (Ogen is called at each time step).

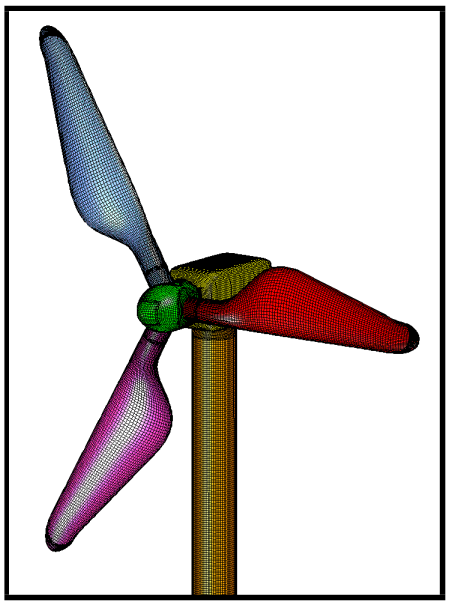
Grid	order of accuracy	grid points	processors (nodes$\times$ p/n)	cpu (s)
Sphere in a box	2	2.1 billion	16 (8×2)	136
Re-entry vehicle	4	215 million	128 (16×8)	1990

Rapid grid generation for moving geometry is critical.

Ogen is the overlapping grid generator in Overture.

Overlapping grid generation consists of two major steps:

1. construct component grids (Mappings).
2. grid connectivity: cut holes and determine interpolation information using Ogen (this is the step that requires most of the CPU time).



In recent work changes have been made to support

- the generation of large (billion point +) grids.
- parallel moving-grid flow simulations (Ogen is called at each time step).

Grid	order of accuracy	grid points	processors (nodes$\times$ p/n)	cpu (s)
Sphere in a box	2	2.1 billion	16 (8×2)	136
Re-entry vehicle	4	215 million	128 (16×8)	1990

Summary

We have been developing efficient algorithms for modeling

1. incompressible flows and rigid body motion,
2. fast solution of elliptic boundary value problems in complex geometry,

The approach is based on

1. overlapping grids for flexible representation of geometry,
2. high-order accurate algorithms,
3. approximate factored operators and compact schemes,
4. matrix free multigrid algorithms,
5. accurate treatment of boundary conditions,
6. fast parallel grid generation.