Using coordinate transformations in Nektar++ incompressible flow solver

Douglas Serson

Department of Aeronautics
Imperial College London

Nektar++ Workshop, June 2016

Outline

Motivation

Formulation

Implementation

Examples

Summary

Motivation

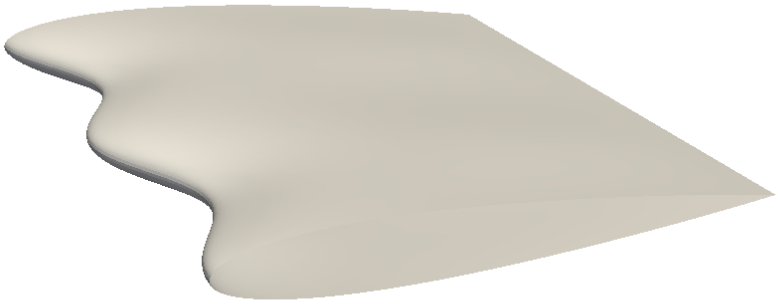
- Nektar++ already supports complex geometries, so why use coordinate transformations?

Motivation

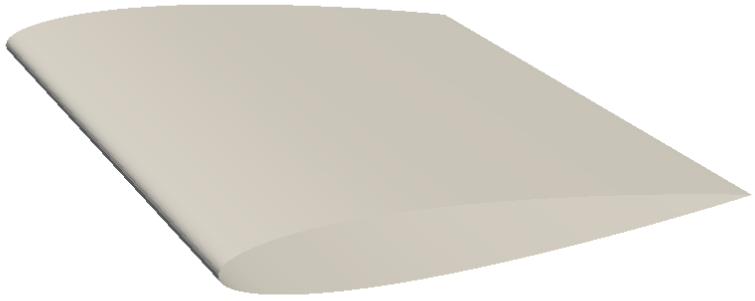
- Nektar++ already supports complex geometries, so why use coordinate transformations?
- By simplifying the geometry, we can reduce the computational cost
  - Quasi-3D approach
  - Moving bodies without deformable mesh

Motivation

Example:



Geometry we want to study



Transformed geometry

Motivation

- What is available in the literature?
  - Explicit approach for spectral/hp discretization (Newman, 1997; Darekar, 2001) $\Rightarrow$ restricted to constant Jacobian
  - Semi-implicit approach for pseudo-spectral method (Carlson, 1995) $\Rightarrow$ pressure boundary condition?

Motivation

- What is available in the literature?
  - Explicit approach for spectral/hp discretization (Newman, 1997; Darekar, 2001) $\Rightarrow$ restricted to constant Jacobian
  - Semi-implicit approach for pseudo-spectral method (Carlson, 1995) $\Rightarrow$ pressure boundary condition?
- These methods were generalized, leading to an explicit and a semi-implicit formulation, both of which
  - Support general transformations (including time-dependent)
  - Have consistent pressure boundary conditions

Formulation

Formulation - original

Typically, we solve the incompressible Navier-Stokes equations in a Cartesian coordinate system

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{N}(\mathbf{u}) - \nabla p + \nu \mathbf{L}(\mathbf{u})$$
$$\nabla \cdot \mathbf{u} = 0$$

where

$$\mathbf{N}(\mathbf{u}) = -(\mathbf{u} \cdot \nabla)\mathbf{u}$$

and

$$\nu \mathbf{L}(\mathbf{u}) = \nu \nabla^2 \mathbf{u}$$

Formulation - original

In the velocity-correction scheme, the pressure is solved by

$$\int_\Omega \nabla p^{n+1} \cdot \nabla \phi \, d\Omega = \int_\Omega \phi \nabla \cdot \left(-\frac{\hat{\mathbf{u}}}{\Delta t} \right) \, d\Omega + \int_\Gamma \phi \left[\frac{\hat{\mathbf{u}} - \gamma_0 \overline{\mathbf{u}}^{n+1}}{\Delta t} - \nu (\nabla \times \nabla \times \mathbf{u})^* \right] \cdot \mathbf{n} \, dS,$$

where

$$\hat{\mathbf{u}} = \mathbf{u}^+ + \Delta t \mathbf{N}^*,$$

* represents extrapolation in time and $^+$ represents backward differencing.

Formulation - original

The velocity is the solution of

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \hat{\mathbf{u}}}{\Delta t} = -\nabla p^{n+1} + \nu \mathbf{L}(\mathbf{u}^{n+1})$$

with appropriate boundary conditions.

Navier-Stokes equations in the transformed system

In the transformed coordinate system, the incompressible Navier-Stokes equations can be written as

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= \overline{\mathbf{N}}(\mathbf{u}) - \overline{\mathbf{G}}(p) + \nu \overline{\mathbf{L}}(\mathbf{u}), \\ D(\mathbf{u}) &= 0, \end{aligned}$$

where

$$\begin{aligned} \overline{\mathbf{N}}(\mathbf{u}) &= -u^j u^i_{,j} + V^j u^i_{,j} - u^j V^i_{,j} \\ \overline{\mathbf{G}}(p) &= g^{ij} p_{,j} \\ \nu \overline{\mathbf{L}}(\mathbf{u}) &= \nu g^{jk} u^i_{,jk} \\ D(\mathbf{u}) &= \frac{1}{J} \nabla \cdot (J u^i) \end{aligned}$$

are the terms obtained from tensor calculus.

Explicit formulation

For the explicit formulation, we rewrite the equations as

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{N}(\mathbf{u}) - \frac{\nabla p}{J} + \nu \mathbf{L}(\mathbf{u}) + \mathbf{A}(\mathbf{u}, p), \\ D(\mathbf{u}) &= 0, \end{aligned}$$

where the forcing term $\mathbf{A}$ can be obtained by

$$\mathbf{A}(\mathbf{u}, p) = \left[\overline{\mathbf{N}}(\mathbf{u}) - \mathbf{N}(\mathbf{u})\right] + \left[-\overline{\mathbf{G}}(p) + \frac{\nabla p}{J}\right] + \nu \left[\overline{\mathbf{L}}(\mathbf{u}) - \mathbf{L}(\mathbf{u})\right]$$

If $J \equiv 1$, we obtain the approach of Newman (1997), where we just have to add $\mathbf{A}$ to $\mathbf{N}$.

Explicit formulation

Following a derivation analogous to the one for Cartesian system, we obtain the pressure equation

$$\int_\Omega \nabla p^{n+1} \cdot \nabla \phi \, d\Omega =$$
$$\int_\Omega \phi \nabla \cdot \left[-\frac{J\hat{\mathbf{u}}}{\Delta t} + \nu \left(\nabla \left(\frac{\mathbf{u}}{J} \cdot \nabla J \right) \right)^* \right] + \nu \nabla J \cdot (\nabla \times \nabla \times \mathbf{u})^* \, d\Omega$$
$$+ \int_\Gamma \phi J \left[\frac{\hat{\mathbf{u}} - \gamma_0 \bar{\mathbf{u}}^{n+1}}{\Delta t} - \nu \left(\nabla \left(\frac{\mathbf{u}}{J} \cdot \nabla J \right) \right)^* - \nu (\nabla \times \nabla \times \mathbf{u})^* \right] \cdot \mathbf{n} \, dS$$

where

$$\hat{\mathbf{u}} = \mathbf{u}^+ + \Delta t (\mathbf{N}^* + \mathbf{A}^*)$$

Explicit formulation

The velocity is obtained by solving

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \hat{\mathbf{u}}}{\Delta t} = -\frac{\nabla p^{n+1}}{J} + \nu \mathbf{L}(\mathbf{u}^{n+1})$$

with appropriate boundary conditions.

Semi-implicit formulation

- The semi-implicit formulation consists in following the original procedure using the modified operators $\overline{\mathbf{N}}$, $\overline{\mathbf{G}}$, and $\overline{\mathbf{L}}$
- The modified Helmholtz equations are solved iteratively, since assembling the matrix would break the symmetries created by the transformation (in quasi-3D, the Fourier modes would get coupled)
- It is also necessary to modify the $(\nabla \times \nabla \times)$ operator.

Semi-implicit formulation

The pressure is solved by the iteration

$$\nabla p_{s+1}^{n+1} = \nabla p_s^{n+1} + J\left[\frac{\mathbf{u}^+ - \gamma_0 \bar{\mathbf{u}}^{n+1}}{\Delta t} - \nu \mathbf{Q}^* + \bar{\mathbf{N}}^* - \bar{\mathbf{G}}(p_s^{n+1})\right],$$

which leads to

$$\int_\Omega \nabla p_{s+1}^{n+1} \cdot \nabla \phi \, d\Omega =$$
$$\int_\Omega \phi \left[JD\left(\frac{-\hat{\mathbf{u}}}{\Delta t}\right) + JD(\bar{G}(p_s^{n+1})) - \nabla^2 p_s^{n+1} \right] d\Omega$$
$$+ \int_\Gamma \phi \left[J\left(\frac{\hat{\mathbf{u}} - \gamma_0 \bar{\mathbf{u}}^{n+1}}{\Delta t}\right) - \nu J \mathbf{Q}^* - J\bar{G}(p_s^{n+1}) + \nabla p_s^{n+1} \right] \cdot \mathbf{n} \, dS,$$

where s is the iteration counter and

$$\mathbf{Q} = \varepsilon^{imn} \varepsilon^{ljk} g_{nl} g_{kp} u^p_{,jm}$$

Semi-implicit formulation

The velocity is solved by

$$\frac{\gamma_0 \mathbf{u}_{s+1}^{n+1}}{\Delta t} - \nu \mathbf{L}(\mathbf{u}_{s+1}^{n+1}) = \frac{\hat{\mathbf{u}}}{\Delta t} - \bar{G}(p^{n+1}) + \nu \bar{\mathbf{L}}(\mathbf{u}_{s}^{n+1}) - \nu \mathbf{L}(\mathbf{u}_{s}^{n+1}),$$

where again s is the iteration counter.

A relaxation factor can be included in the iterative loops to improve the numerical stability.

Explicit vs. Semi-implicit

- Both are similar in terms of accuracy
- Explicit is faster
- Semi-implicit is more robust
- Both eventually become unstable as the transformation becomes more energetic

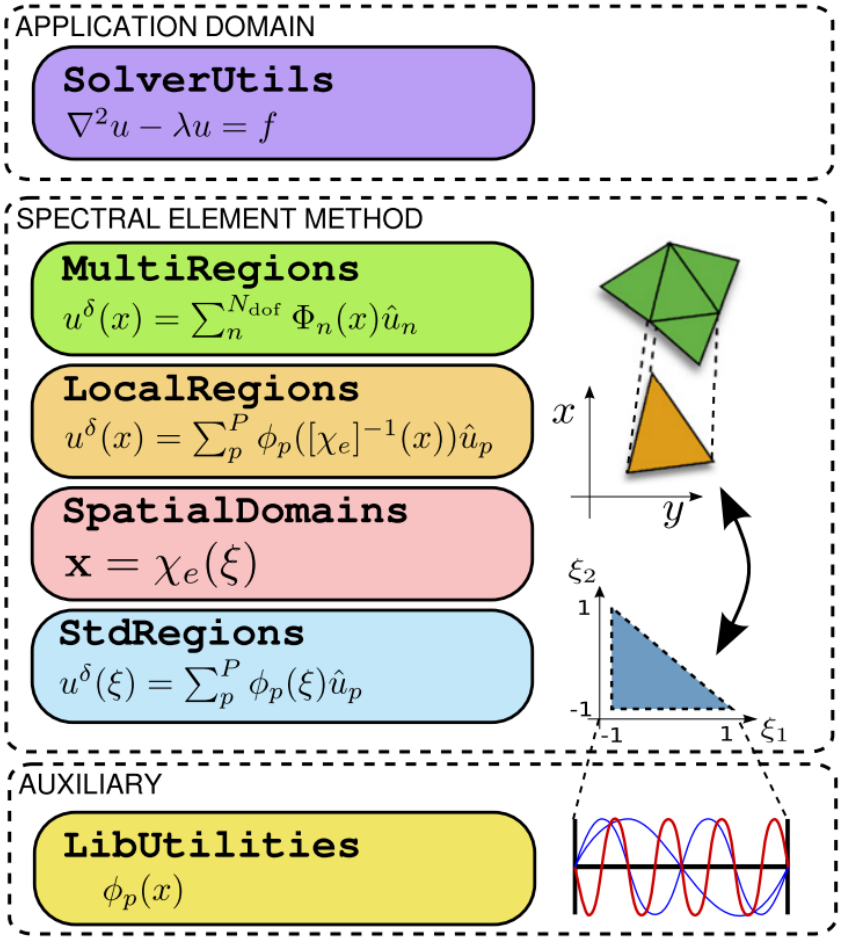
Implementation

Implementation

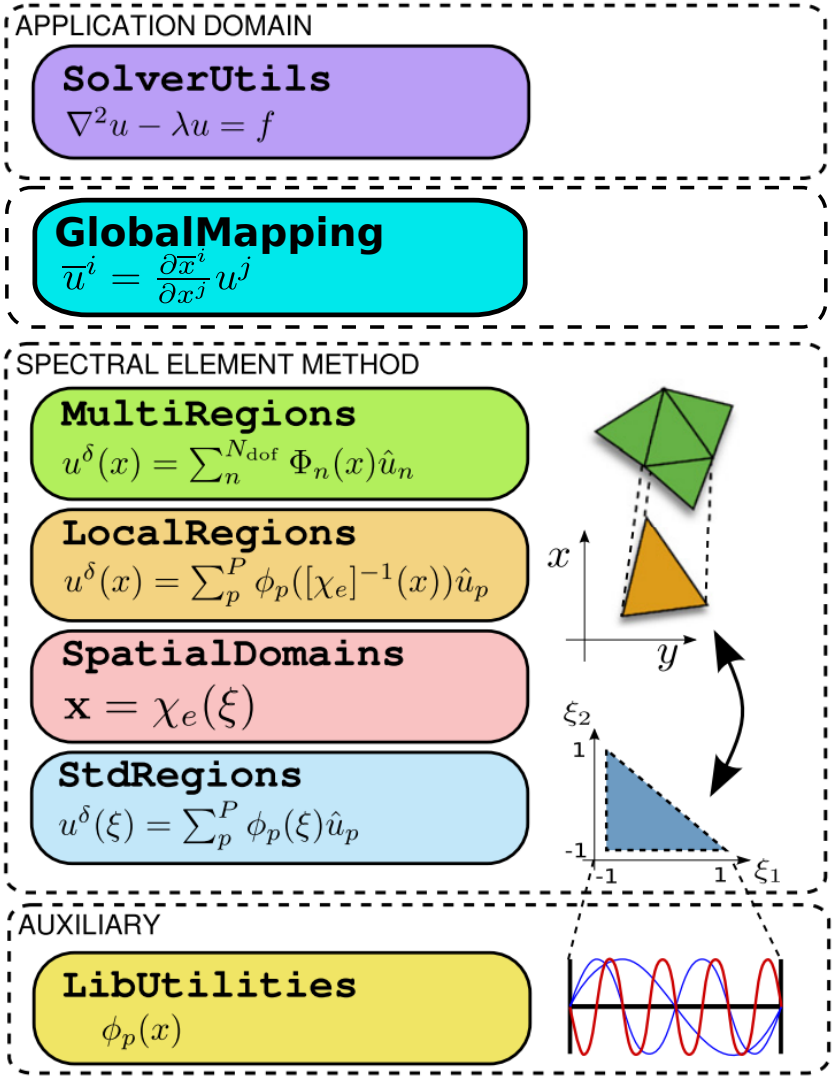
The implementation of the coordinate transformations consists of

- A new library encapsulating tensor calculus functionality
- Changes in the solver level
- Postprocessing changes

Libraries


APPLICATION DOMAIN
SolverUtils
$\nabla^2 u - \lambda u = f$
SPECTRAL ELEMENT METHOD
MultiRegions
$u^\delta(x) = \sum_n^{N_{\text{dof}}} \Phi_n(x)\hat{u}_n$
LocalRegions
$u^\delta(x) = \sum_p^P \phi_p([\chi_e]^{-1}(x))\hat{u}_p$
SpatialDomains
$\mathbf{x} = \chi_e(\xi)$
StdRegions
$u^\delta(\xi) = \sum_p^P \phi_p(\xi)\hat{u}_p$
x
y
ξ_2
1
-1
-1
1
ξ_1
AUXILIARY
LibUtilities
$\phi_p(x)$

Libraries


APPLICATION DOMAIN
SolverUtils
$\nabla^2 u - \lambda u = f$
GlobalMapping
$\overline{u}^i = \frac{\partial \overline{x}^i}{\partial x^j} u^j$
SPECTRAL ELEMENT METHOD
MultiRegions
$u^\delta(x) = \sum_n^{N_{\text{dof}}} \Phi_n(x) \hat{u}_n$
LocalRegions
$u^\delta(x) = \sum_p^P \phi_p([\chi_e]^{-1}(x)) \hat{u}_p$
SpatialDomains
$\mathbf{x} = \chi_e(\xi)$
StdRegions
$u^\delta(\xi) = \sum_p^P \phi_p(\xi) \hat{u}_p$
x
y
ξ_2
1
-1
-1
1
ξ_1
AUXILIARY
LibUtilities
$\phi_p(x)$

The GlobalMapping library

The GlobalMapping library is formed by

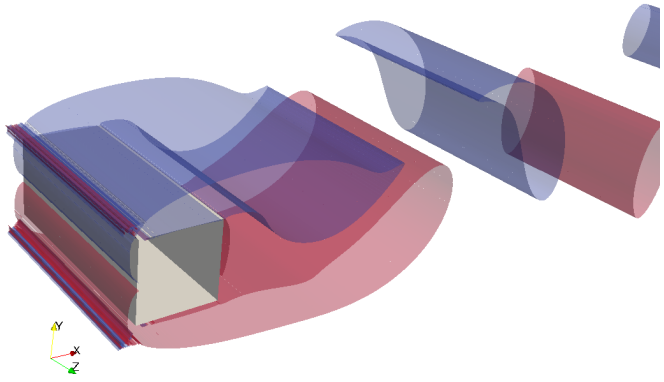
- A base class Mapping
  - Transformations between the two coordinate systems
  - Basic tensor calculus functions (e.g. Jacobian, raise index)
  - Differential operators
  - Auxiliary functions
- Implementation of particular mappings

Mapping types

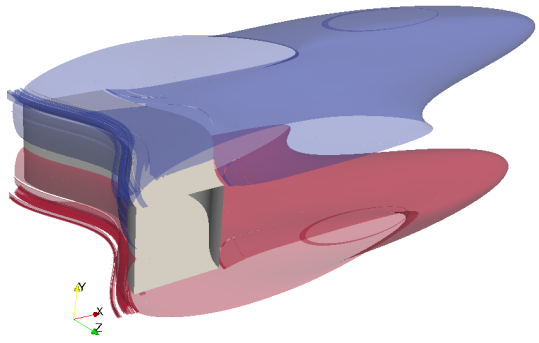
Mapping type	$\bar{x}$	$\bar{y}$	$\bar{z}$
Translation	$x + f(t)$	$y + g(t)$	$z + h(t)$
XofZ	$x + f(z, t)$	y	z
XofXZ	$f(x, z, t)$	y	z
XYofZ	$x + f(z, t)$	$y + g(z, t)$	z
XYofXY	$f(x, y, t)$	$g(x, y, t)$	z
General	$f(x, y, z, t)$	$g(x, y, z, t)$	$h(x, y, z, t)$

Examples

Flow around wavy square cylinder



Original geometry



Transformed geometry

Flow around wavy square cylinder

What changes are required in the session file?

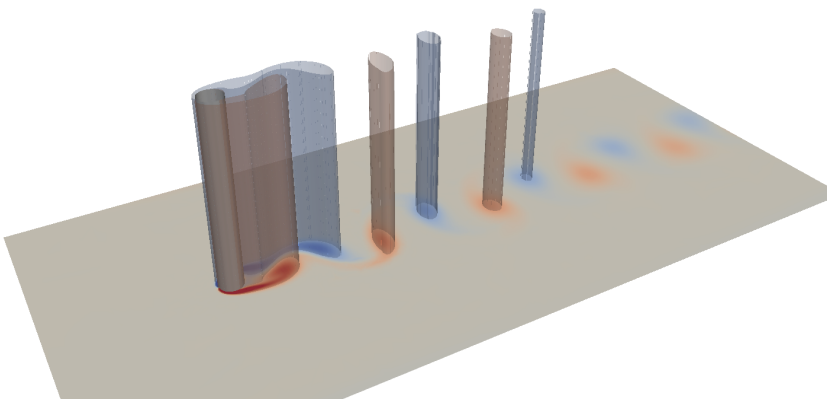
```
<CONDITIONS>
 <SOLVERINFO>
   <I PROPERTY="SolverType"
   VALUE="VelocityCorrectionScheme"/>
 </SOLVERINFO>




</CONDITIONS>
```

```
<CONDITIONS>
 <SOLVERINFO>
   <I PROPERTY="SolverType"
   VALUE="VCSMapping" />
 </SOLVERINFO>

 <FUNCTION NAME="MappingFcn">
  <E VAR="x"
     VALUE="x-0.15*cos(PI*z/1.5)"/>
  </FUNCTION>
</CONDITIONS>

<MAPPING TYPE="XofZ">
  <COORDS>MappingFcn</COORDS>
</MAPPING>
```

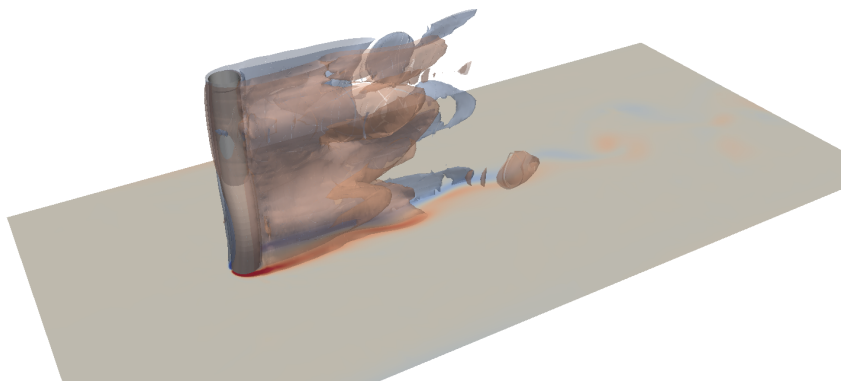
Flow around wavy square cylinder

- By default, the explicit formulation is used
- The simulation is around 70% more expensive than the original one
- This should still be advantageous compared to running a full 3D case
- The increase in cost is emphasized by the high efficiency of the implicit solves.

Flow around wavy flexible cylinder (forced vibration)



Original geometry



Transformed geometry

Flow around wavy flexible cylinder (forced vibration)

- Same changes to session file, but now the `MAPPING` section is

```
<MAPPING TYPE="General">
    <COORDS>MappingFcn</COORDS>
    <VEL>MappingVel</VEL>
    <TIMEDEPENDENT>True</TIMEDEPENDENT>
</MAPPING>
```

- For time dependent mappings, also need to define function describing coordinates velocity

```
<FUNCTION NAME="MappingVel">
  <E VAR="vx" VALUE="0.0" />
  <E VAR="vy" VALUE="omega*Av*cos(omega*t)*sin(2*PI*z/lambdaV)"/>
</FUNCTION>
```

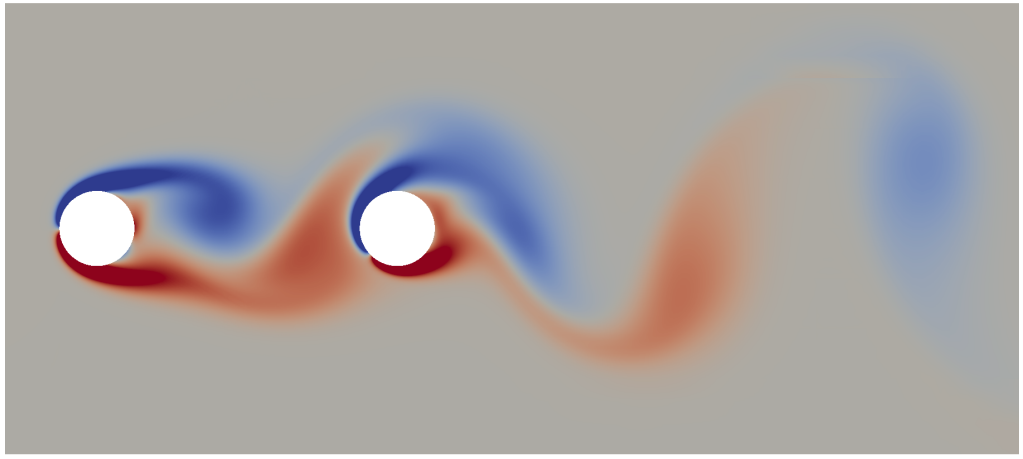
Flow around wavy flexible cylinder (forced vibration)

- The wall boundary condition for the moving body is set using

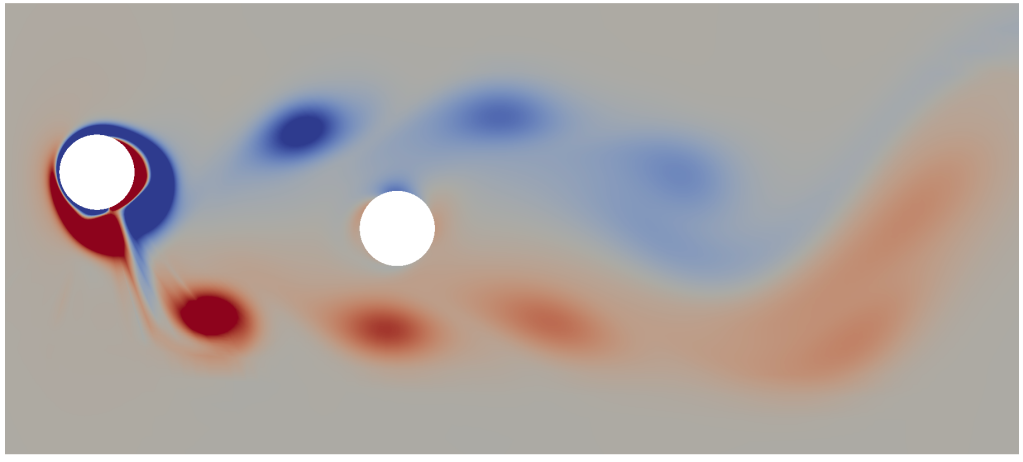
```
<REGION REF="0">
    <D VAR="u" USERDEFINEDTYPE="MovingBody" VALUE="0" />
    <D VAR="v" USERDEFINEDTYPE="MovingBody" VALUE="0" />
    <D VAR="w" VALUE="0" />
    <N VAR="p" USERDEFINEDTYPE="H" VALUE="0" />
</REGION>
```

- Extra parameters are required for using semi-implicit approach (see user guide)

Flow around two moving cylinders



Original geometry



Transformed geometry

Flow around two moving cylinders

- Need to calculate a coordinate transformation based on cylinder displacement (in this case solving a Laplace equation)
- Not possible with current version of master
- Potential for a future fluid-structure interaction module (possibly combined with re-meshing)

Summary

- Nektar++ incompressible solver now supports coordinate transformations
- An explicit and a semi-implicit approach are available
- These methods are
  - Flexible $\Rightarrow$ time-dependent, non-constant Jacobian
  - Easy to use
  - Efficient
  - Accurate
- Numerical stability can be a problem

Thank you!