PROJECT PROPOSAL: TOPOGRAPHY IN LARGE-EDDY SIMULATION

Victoria Whitley
Advisor: Jacob Wenegrat
As the only computationally feasible technique for modeling ocean flows at scales that resolve both turbulence and large-scale dynamics, how can we include topography into large-eddy simulation, through either immerse boundary method or adding finite volume topography, to assist the state of the art solver, Oceananigans.jl, in answering the many research questions involving flow along and over topography?

Ramadhan, et al. (2020)
BACKGROUND: TURBULENCE

- Characteristics
  - Irregular and unsteady
  - Fluctuating in time and space

- Examples
  - Flow over a gold ball
  - Steam above a coffee cup
  - Mixed layers of the atmosphere or ocean

- Scales
  - Large eddies on scale with ocean mixed layer depth, ~100m, energy often added from boundary fluxes
  - Molecular level, where energy is dissipated by viscosity

BACKGROUND: LARGE-EDDY SIMULATION (LES)

- Resolves largest eddies
- Models smaller scales (inertial sub-range) based on physically motivated parametrizations
- Computationally feasible and accurate at desired resolution for turbulence
- Still constrained to ~10km x 10 km ocean domains
- Hard to deal with complex geometries in flow problems

Ramadhan, et al. (2020)
BACKGROUND: COMPARING SOLVERS

REYNOLDS-AVERAGED NAVIER-STOKES (RANS)
LES
DIRECT NUMERICAL SIMULATION (DNS)

Rodriguez. (2019)
• Climate Modeling Alliance (CliMA) to build climate model from scratch
• Oceananigans.jl is part of the model, to solve incompressible fluid problems for ocean
  • “fast and friendly”
• Updated parametrizations and numerical methods
• Supports LES and DNS for CPUs and GPUs
• Written in Julia

Ramadhan, et al. (2020)
MOTIVATION:
IMPORTANCE OF TOPOGRAPHY IN CLIMATE MODELS

- Topography can extract energy from geostrophic flows (10-100 km), creating sub-mesoscale turbulence (0.1 – 10 km)
- Complex bottom boundary layer can cause flow to separate from wall and mix with interior
- Undiscovered differences between sea surface layer and bottom boundary layer (BBL)
- Work around LES resolution limits with moving topography

Gula, (2016),
McWilliams, (2016),
METHODOLOGY: IMMERSED BOUNDARY METHOD (IBM)

• Instead of conforming the mesh to the fluid domain, $\Omega_f$, a cartesian grid is generated without regard to the solid body, $\Omega_b$

• Incorporate the BCs by modifying the equations near the boundary

• Allows discretization of complex domains without coordinate transformations or complicated discretization operators

Mittal and Iaccarino, (2005)
\[ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \frac{1}{Re} \nabla^2 \mathbf{u} = 0 \]

\[ \nabla \cdot \mathbf{u} = 0 \quad \subset \Omega_f \]

With the boundary,

\[ \mathbf{u} = \mathbf{u}_\Gamma \quad \text{on} \ \Gamma_b \]

Mittal and Iaccarino, (2005)
METHODOLOGY:
IMPOSING BOUNDARY CONDITIONS

\[ \mathcal{L}(U) = 0 \quad \in \Omega_f \]
\[ U = U_\Gamma \quad \in \Gamma_b \]

where \( U = (u, p) \) and \( \mathcal{L} \) is the Navier-Stokes equations operator.

Mittal and Iaccarino, (2005)
METHODOLOGY: CONTINUOUS FORCING

Modify continuous equations by adding in a forcing term $f_b = (f_m, f_p)$ to both momentum and pressure equations

$$\mathcal{L}(U) = f_b \in (\Omega_b \cup \Omega_f)$$

Then discretized on a Cartesian grid on the whole domain without regard to the original boundary

$$[L]\{U\} = \{f_b\}$$

Mittal and Iaccarino, (2005)
METHODOLOGY:
CONTINUOUS FORCING

IB represented by massless elastic fibers with Lagrangian points moving at local fluid velocity

\[ \frac{\partial X_k}{\partial t}(s, t) = u(X_k(s, t)) \]

Force \( f(x, t) \) is then defined

\[ f(x, t) = \sum_k F_k(t) \delta(|x - X_k|) \]

Where \( F_k \) is the stress of the fibers characterized by Hooke’s law

Note: \( \delta \) is usually implemented as a distribution function

Mittal and Iaccarino, (2005)
METHODOLOGY: DISCRETE FORCING

Discretize original system onto Cartesian grid without regard to boundary

\[[L]\{U\} = 0\]

Adjust the discretization near the boundary through a modified discretized system

\[[L']\{U\} = \{r\} \quad \text{or} \quad [L]\{U\} = \{f_b'\}\]

Where \( f_b' = \{r\} + [L]\{U\} - [L']\{U\} \)

Mittal and Iaccarino, (2005)
METHODOLOGY: DISCRETE FORCING

Discretize original system onto Cartesian grid without regard to boundary as a time stepping scheme

$$\frac{u_{i}^{n+1} - u_{i}^{n}}{\Delta t} = RHS_{i}$$

Solve without forcing for $u^{*(n+1)}$ (does not satisfy IBC)

Compute $f^{n+1}$ by adding it into the equation

$$\frac{u_{i}^{*(n+1)} - u_{i}^{n}}{\Delta t} = RHS_{i} + f^{n+1}$$

Recompute the true solution $u^{(n+1)}$ with the new forcing term

$$[L]\{U^{(n+1)}\} = f^{n+1}$$ (satisfies IBC)

Update velocity and pressure and continue for next time step

Mittal and Iaccarino, (2005)
<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Continuous</strong></td>
<td>• Easy to implement on elastic or moving boundaries</td>
<td>• Stability constraints for rigid bodies</td>
</tr>
<tr>
<td></td>
<td>• Independent of mesh</td>
<td>• Smoothing functions make sharp IB difficult</td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td>• No extra stability constraints</td>
<td>• Forcing procedure less practical</td>
</tr>
<tr>
<td></td>
<td>• Can create sharp IB</td>
<td>• Difficult to include boundary motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dependent on discretization</td>
</tr>
</tbody>
</table>

*Mittal and Iaccarino, (2005)*
PROJECT GOALS

1. Finding best way to implement topography into LES
   • IBM vs Finite Volume Topography
   • Discrete vs Continuous Forcing
   • Dealing with different types of boundary conditions

2. Implement topography in O.jl
   • Compatibility with pressure solver
   • Keeping added computational overhead low
   • Following their standards of validation

3. Implement logarithmic wall model for modeling boundary stresses rather than direct no slip condition

4. (Possibly) Consider BBL shedding off topography as an application to code
IMPLEMENTATION: THE LANGUAGE

- Programming Language: Julia
  - Aims to combine efficiency and ease of use
  - Already has many CFD tools
  - Good for GPU compatibility and parallel computing
  - Can call Python, C, and Fortran libraries
  - “Walks like Python, runs like C.”
  - Free and open source!

IMPLEMENTATION: JULIA IN COMPARISON

IMPLEMENTATION: TYPICAL EQUATIONS

- Spatially-filtered, incompressible Boussinesq equations, with tracer equations
  \[ \partial_t u + (u \cdot \nabla)u + (f - \nabla \times u^S) \times u = -\nabla \phi + b \hat{z} - \nabla \cdot \tau - \partial_t u^S + F_u \]

  \[ \partial_t \theta + u \cdot \nabla \theta = -\nabla \cdot q + F_\theta \]

  \[ \nabla \cdot u = 0 \]

- Linear buoyancy relationship
  \[ b = -\alpha g (\theta_0 - \theta) \]

Souza, (2020)
IMPLEMENTATION: SPACE DISCRETIZATION

• Finite volume method

\[ c_{i,j,k} = \frac{1}{V_{i,j,k}} \int c(x) \, dV_{i,j,k} \]

• Staggered Arakawa C-grid, interpolated to perform operations

• Centered second-order differences for advection and diffusion terms

Souza, (2020)
Eymard and Gallouët, (2010)
IMPLEMENTATION:
TEMPORAL DISCRETIZATION

• Time integral of momentum equation with pressure decomposition

\[ u^{n+1} - u^n = \int_{t_n}^{t_{n+1}} \left[ -\nabla\phi_{non} - \nabla_h\phi_{hyd} - (\mathbf{u} \cdot \nabla)\mathbf{u} - \mathbf{f} \times \mathbf{u} + \nabla \cdot \tau + F_u \right] dt \]

\[ u^{n+1} - u^n \approx -\Delta t \nabla\phi_{non}^{n+1} + \int_{t_n}^{t_{n+1}} G_u \, dt \]

• 2\textsuperscript{nd} order explicit Adams-Bashforth

\[ \int_{t_n}^{t_{n+1}} G \, dt \approx \Delta t \left[ \left( \frac{3}{2} + X \right) G^n - \left( \frac{1}{2} + X \right) G^{n-1} \right] \]

Souza, (2020)
Yinnian and Li, (2009)
IMPLEMENTATION: PRESSURE SOLVER

• Pressure projection method with Poisson pressure equation
  
  \[ \phi(x, t) = \phi_{\text{hyd}}(x, t) + \phi_{\text{non}}(x, t) \]

  \[ \nabla^2 \phi_{\text{non}} = \frac{\nabla \cdot u^n}{\Delta t} + \nabla \cdot G_u = F \]

• Ensures incompressibility of \( u \) while decoupling \( u \) and \( p \)

• FFT eigenfunction expansion of 2\textsuperscript{nd} order operator

  \[ \tilde{\phi}_{\text{non}}(i, j, k) = - \frac{\hat{F}}{\lambda_i^x + \lambda_j^y + \lambda_k^z} \]

  Souza, (2020)
  Ulrich and Sweet, (1988)
IMPLEMENTATION: HARDWARE

- Deepthought 2 HPC cluster
  - Primary HPC cluster maintained by the Division of IT at UMD
  - Being upgraded to RedHat Linux version 8 (RHEL8)
  - Includes 40 GPU nodes
IMPLEMENTATION: VALIDATION METHODS

1. Accuracy
   - Classic channel flow, but create wall with IB
   - Griffith, B. and Luo, X. (2017) and Kallemov, B., et al. (2016) “benchmark problems” with rigid, elastic, and contracting structures: (un)steady flow past stationary cylinder(s) or sphere(s), steady-state flow through a nozzle

2. Computational Cost
   - Compare to current O.jl solver with regular BC implementation

3. Oceananigans.jl Validation Methods
   - Convergence tests including 2D integration tests with non-trivial pressure fields, advection, diffusion
   - Lid-driven cavity test:
   - Stratified Couette flow
MILESTONES

- **October**: Familiarize with Oceananigans and determine IB method
- **November**: Implement topography for simpler case of Dirichlet BC
- **December**: Determine how to implement topography for Neumann BC
- **January**: Solidify and implement topography for various cases in O.jl
- **February**: Test accuracy for various boundary conditions and fix bugs
- **March**: Add wall model and reduce computational costs using O.jl methods
- **April**: Consider scientific question of bottom boundary layer dynamics
- **May**: Get code and method approved for inclusion in next O.jl release
FINAL PRODUCT

• Fast and easy method of implementing topography in Oceananigan.jl’s LES code

• Code approved and added to next Oceananigans.jl release

• A tool to use in future research simulating the separation of ocean boundary layers over rough topography
**SOURCES**


SOURCES


