# canyonsubc Documentation

***Release 1.0***

**Susan Allen, Karina Ramos Musalem, Robert Irwin**

October 14, 2016

Contents

Contents:

CHAPTER 1

# Literature Review, Flow Separation over a Sill

## 1.1 PF Cummins (2000) "Stratified flow over topography: time-dependent comparisons between model solutions and observations"

Using the Princeton Ocean Model (POM), the author was able to simulate flow over an idealized model of the Knight Inlet. The results obtained using this model were then compared to data collected over the Inlet. The model does a poor job of resolving the flow separation in lee of the sill; in particular the model flow demonstrates an apparent overturning which is not evident in the data. This overturning results in a "high drag state" – defined by Cummins to be the state of strong spatial acceleration of flow that is associated with a pressure drop across the sill. Essentially, this state causes an opposing force on the fluid due to the pressure gradient. High drag state is analogous to downslope windstorms in atmospheric dynamics.

High drag state is well known and is observed in the data, however, the overturning causes the drag state to be reached too early in the numerical simulations. This discrepency is likely due to the hypothesis put forward by Cummins; a poor representation of the boundary layer occurring in the lee of the sill.

*(a) The numerical model used and parameters used*

The author argues that the data obtained during the "Knight Inlet Experiment" suggests that the flow is largely 2D in the region during the ebb tide. By assuming 2D flow, the model simplifies significantly and resolution may be increased significantly.

The model uses a free surface, a nonlinear equation of state, suppresses rotational effects, Smagorinsky eddy viscosity, quadratic drag on the bottom boundary (using a drag coefficient derived from von Karman's law-of-the-wall). Additionally, a level 2.5 turbulence closure submodel is used to dissipate high wavenumber energy (Mellor and Yamada, 1982).

The author used 101 sigma levels for the cases presented and stated that increasing the vertical resolution to 201 sigma levels did not change the solutions significantly. Horizontal grid resolution around the sill was varied between 5m and 30m, with presented results of 10m horizontal resolution.

The tidal forcing is given as an influx condition on the left boundary (see figure 2 of paper). Maximum velocity acheived is ~62cm/s after 3.5h. The author focusses on the ebb-tide and states that consideration of the response to flood tide is beyond the scope of interest.

*(b) Results*

The stratification is analytical and three layer, matching well with observations. Internal normal mode wave solutions for this stratification were calculated, and the phase speeds were reported for the first and second vertical internal modes.

## 1.2 KG Lamb (2004), "On boundary-layer separation and internal wave generation at the Knight Inlet sill"

CHAPTER 2

# Approximating Flow Separation over a Half Cylinder

## 2.1 Potential Flow

Consider a half-cylinder at the bottom boundary of an idealized inviscid fluid with a background flow field moving at a constant speed $U_0$ from left-to-right. The governing equations are:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x}$$
$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z}$$
$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0$$

The form of topography is given by:

$$z = h(x) = \begin{cases} 0, & x < -R_0, \\ \sqrt{R_0^2 - x^2}, & -R_0 < x < R_0 \\ 0, & x > R_0 \end{cases}$$

The boundary conditions that will be imposed for the flow away from the cylinder are: far field conditions and no-normal flow conditions at the boundary.

$$\lim_{x,z\to\infty} u = U_0,$$
$$\vec{u}\cdot\hat{n} = 0, \quad z = h(x).$$

Away from the half-cylinder the flow is irrotational, $\vec{\nabla}\times\vec{u} = \vec{0}$, allowing a potential function to be defined:

$$u = \frac{\partial\phi}{\partial x}, \quad w = \frac{\partial\phi}{\partial z}.$$

The incompressibility condition reduces to:

$$\nabla^2\phi = 0.$$

This is Laplace's equation. Noting the axial symmetry, the math may be simplified by converting to polar coordinates. The Laplacian in polar coordinates may be represented by:

$$\frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} = 0.$$

This equation may be solved using a separation of variables decomposition, $\phi(r,\theta) = R(r)T(\theta)$. Substituting this into the previous equation yields two ordinary differential equations:

$$r^2R''(r) + rR'(r) - \mu^2R(r) = 0,$$
$$T''(\theta) + \mu^2T(\theta) = 0.$$

Solutions must be considered for both $\mu = 0$ and $\mu \neq 0$.

### 2.1.1 Case $\mu = 0$

Solving the ODEs yields:

$$R(r) = A_1 \ln r + A_2,$$
$$T(\theta) = B_1\theta + B_2.$$

The no-normal flow conditions in polar coordinates are:

$$\frac{\partial\phi}{\partial\theta} = u_\theta = 0 = T'(\theta), \qquad \theta = 0, \pi$$
$$\frac{\partial\phi}{\partial r} = u_r = 0 = R'(R_0), \qquad 0 < \theta < \pi, \quad r = R_0$$

This forces $T(\theta) = B_2$ and $R(r) = A_2$. This cannot satisfy the far-field condition and so the $\mu = 0$ case is degenerate.

### 2.1.2 Case $\mu \neq 0$

The general solution to the ODE for $R(r)$ is found by substituting an arbitrary polynomial of the form $R(r) = r^\gamma$:

$$\left[\gamma^2 - \mu^2\right] r^\gamma = 0.$$

If $r \neq 0$ (which is always true on this domain), then the solution for $R(r)$ becomes:

$$R(r) = A_1r^\mu + A_2r^{-\mu}$$

The solution for $T(\theta)$ becomes:

$$T(\theta) = B_1 \sin(\mu\theta) + B_2 \cos(\mu\theta).$$

Applying the polar coordinate boundary conditions to the previous solutions yield:

$$B_1 = 0, \qquad A_1R_0^{2\mu} = A_2.$$
$$\Rightarrow \qquad \phi(r,\theta) = C\left[r^\mu + R_0^{2\mu}r^{-\mu}\right]\cos\mu\theta.$$

Applying the far-field condition in polar coordinates:

$$\lim_{x,z\to\infty}\frac{\partial\phi}{\partial x} = U_0,$$
$$\lim_{r\to\infty}\left\{\cos\theta\frac{\partial\phi}{\partial r} - \frac{\sin\theta}{r}\frac{\partial\phi}{\partial\theta}\right\} = U_0,$$
$$\lim_{r\to\infty}\left\{\cos\theta\cos\mu\theta\left[r^{\mu-1} - R_0^{2\mu}r^{-\mu-1}\right] + \sin\mu\theta\sin\theta\left[r^{\mu-1} + R_0^{2\mu}r^{-\mu-1}\right]\right\} = \frac{U_0}{\mu C}.$$

If this is to be bounded as $r \to \infty$, then $\mu \in [-1, 1]/\{0\}$. It must also be independent of $r, \theta$ (because the right hand side is a constant). This is only satisfied when $\mu = \pm 1$. The final solution for $\phi$ is the same for either choice of $\mu$:

$$\phi(r,\theta) = U_0\left[r + R_0^2r^{-1}\right]\cos\theta,$$
$$\phi(x,z) = U_0x\left[1 + \frac{R_0^2}{x^2+z^2}\right].$$

This will be used as the outer flow field solution when considering the boundary layer solution for flow over cylinder.

## 2.2 Pressure at the Boundary

Using the solution for velocity potential, the pressure field at the boundary can be defined using Bernoulli's equation for steady state pressure:

$$p + \frac{\rho}{2}\vec{\nabla}\phi \cdot \vec{\nabla}\phi = C.$$

The undetermined coefficient $C$ may be solved by using a predetermined pressure reference. For this example, that reference will be the left stagnation point pressure (at $x = -R_0, z = 0$). Here $\vec{\nabla}\phi = \vec{0}$, and so $C = p_{sp}$ (using $p_{sp}$ to denote pressure at the stagnation point). The full equation for pressure becomes:

$$p = p_{sp} + \rho\frac{U_0^2}{2}\left[\left(R_0^2 - x^2\right)^2 + z^4 + 2x^2z^2 + 2R_0^2z^2\right]\left(x^2 + z^2\right)^{-2}.$$

Solving for pressure along the boundary yields:

$$p_{bdy}(x) = p_{sp} + 2\rho U_0^2\left(1 - \frac{x^2}{R_0^2}\right),$$
$$p_{bdy}(\theta) = p_{sp} + 2\rho U_0^2 \sin^2\theta.$$

## 2.3 Prandtl's Boundary Layer Equations

The equations that balance viscosity, pressure and advection near the boundary in a steady state are Prandtl's boundary layer equations, given by:

$$u^*\frac{\partial u^*}{\partial x^*} + w^*\frac{\partial u^*}{\partial z^*} = -\frac{1}{\rho}\frac{dp}{dx^*} + \nu\frac{\partial^2 u^*}{\partial z^{*2}},$$
$$\frac{\partial u^*}{\partial x^*} + \frac{\partial w^*}{\partial z^*} = 0.$$

The components are starred here to denote that the coordinate system differs from the standard Cartesian system. The $x^*$ coordinate denotes the along boundary coordinate, and the $z^*$ coordinate denotes the normal coordinate. In terms of the half-cylinder problem:

$$x^* = (\pi - \theta)R_0,$$
$$z^* = r - R_0.$$

Substituting in the equation for pressure at the boundary:

$$u^*\frac{\partial u^*}{\partial x^*} + w^*\frac{\partial u^*}{\partial z^*} = \frac{4U_0^2}{R_0}\sin\frac{x^*}{R_0}\cos\frac{x^*}{R_0} + \nu\frac{\partial^2 u^*}{\partial z^{*2}} = U_{bdy}(x^*)\frac{dU_{bdy}}{dx^*} + \nu\frac{\partial^2 u^*}{\partial z^{*2}},$$
$$\frac{\partial u^*}{\partial x^*} + \frac{\partial w^*}{\partial z^*} = 0.$$

The separation point is the point along the boundary that satisfies:

$$\left.\frac{\partial u^*}{\partial z^*}\right|_{z^*=0} = 0$$

The far-field condition satisfies the velocity potential at the boundary. The far-field potential in terms of the boundary variable $x^*$ is:

$$\phi_{ff}(R_0, \theta) = 2U_0R_0\cos\theta,$$
$$\phi_{ff}(x^*) = -2U_0R_0\cos\frac{x^*}{R_0}$$

Explicitly stating the far-field condition:

$$\lim_{z^*\to\infty} u^* = U_{bdy}(x^*) = \frac{d\phi_{ff}}{dx^*} = 2U_0 \sin\left(\frac{x^*}{R_0}\right).$$

Following "Boundary Layer Theory", Schlichting, 1979, the boundary layer equations may be treated using a Blausius series. In order to approach this problem, rewrite in terms of the streamfunction, $\psi$:

$$u^* = \frac{\partial\psi}{\partial z^*}, \quad w^* = -\frac{\partial\psi}{\partial x^*},$$
$$\frac{\partial\psi}{\partial z^*}\frac{\partial^2\psi}{\partial x^*\partial z^*} - \frac{\partial\psi}{\partial x^*}\frac{\partial^2\psi}{\partial z^{*2}} - \nu\frac{\partial^3\psi}{\partial z^{*3}} = \frac{4U_0^2}{R_0}\sin\frac{x^*}{R_0}\cos\frac{x^*}{R_0}$$

The solution must satisfy the no-slip condition, and the far field condition:

$$\frac{\partial\psi}{\partial x^*} = \frac{\partial\psi}{\partial z^*} = 0, \qquad \text{at } z^* = 0,$$
$$\lim_{z^*\to\infty}\psi = U_{bdy}(x^*)z^*$$

To solve this equation, a series solution must be obtained. A general series for $\psi$ is:

$$\psi(x^*, z^*) = \sum_{n=0}^{\infty} x^{*n} f_n(z^*)$$

Noting the far-field condition is an odd function in $x^*$:

$$\lim_{z^*\to\infty}\frac{\partial\psi}{\partial z^*} = U_{bdy}(x^*) = u_1x^* + u_3x^{*3} + u_5x^{*5} + \ldots$$

And so the streamfunction must also be an odd function of $x^*$. Using the odd expansion of the streamfunction and substituting into the streamfunction momentum equation:

$$(\text{DE}_1)x + (\text{DE}_3)x^3 + (\text{DE}_5)x^5 + \mathcal{O}(x^7) = 0$$

Where each of the quantities in brackets are ordinary differential equations in $f_i(z^*)$. To write them explicitly:

$$\text{DE1}: \quad (f_1')^2 - f_1(f_1'') - \nu f_1''' - 4\frac{U_0^2}{R_0^2} = 0$$
$$\text{DE3}: \quad 4f_1'f_3' - f_1f_3'' - 3f_3f_1'' - \nu f_3'''' + \frac{8}{3}\frac{U_0^2}{R_0^4} = 0$$
$$\text{DE5}: \quad 6f_1'f_5' + 3(f_3')^2 - f_1f_5'' - 5f_5f_1'' - 3f_3f_3'' - \nu f_5'''' - \frac{8}{15}\frac{U_0^2}{R_0^6} = 0$$

Where primed quantities denote the derivative with respect to $z^*$. The initial differential equation is the most difficult due to the nonlinearity. It must be treated asymptotically and an approximate solution must be acquired before solving the simpler differential equations (DE3, DE5, etc.).

It should be noted here that these equations may be used to derive a natural scale for the boundary layer thickness. In the middle of the boundary layer, each term of Prandtl's equations are balanced, and so must be the terms in DE1. If $\delta$ represents this natural scale for boundary layer thickness, use a scaled vertical component, $\eta = z^*/\delta$, in order, the terms are scaled as:

$$\frac{F^2}{\delta^2}, \frac{F^2}{\delta^2}, \frac{\nu F}{\delta^3}, 4\frac{U_0^2}{R_0^2}$$

where $F$ is the natural scaling for $f_1$. This leads to the termwise balance:

$$F \sim \nu/\delta, \quad R_0^2\nu^2/4U_0^2 \sim \delta^4$$
$$\Rightarrow \quad \delta \sim \sqrt{\frac{R_0\nu}{2U_0}}$$
$$\Rightarrow \quad F \sim \sqrt{\frac{2U_0\nu}{R_0}}$$

For numerical simulations, values of

$$R_0 = 2.5cm, U_0 = 1cm \cdot s^{-1}, \nu = 10^{-6}m^2 \cdot s^{-1}$$

are used, yielding a boundary thickness of $\delta \sim 1.12mm$.

The boundary and far-field conditions are important to determine for $f_n$. Explicitly:

$$\begin{aligned} \text{BCs}: &\quad f_n(0) = 0, \quad f_n'(0) = 0 \\ \text{FFCs}: &\quad \lim_{z^* \to \infty} f_n'(z^*) = \frac{2U_0}{n! R_0^n}, \quad n = 2j+1 \quad \text{for } j = 0 \ldots \infty \end{aligned}$$

### 2.3.1 Solving $f_1$

DE1 must be treated approximately. Because this analysis is concerned with separation processes, series solutions in $z^*$ will be sufficiently accurate. The series solution to DE1 reads:

$$f_1(z^*) \approx \frac{A_1}{2} z^{*2} - \frac{2}{3} \frac{U_0^2}{R_0^2 \nu} z^{*3} \quad \text{for} \quad z^* \ll \delta$$

In order to solve for $A_1$, this will have to be matched to the outer solution. The outer solution is obtained by considering the far field condition. Second-order and higher derivatives of $f_n$ vanish as $z^* \to \infty$ and so high order derivatives are ignored. This leaves a solution of the form:

$$f_1(z^*) \approx \frac{2U_0}{R_0} z^* + B_1, \quad \text{for} \quad z^* \gg \delta$$

The inner and outer solutions are matched as $z^* \to \delta$. By making the matched solution continuous and smooth, the values of $A_1$ and $B_1$ may be solved:

$$\begin{aligned} A_1 &= 3\sqrt{\frac{2}{\nu}} \frac{U_0^{3/2}}{R_0^{3/2}}, \\ B_1 &= -\frac{5}{12}\sqrt{\frac{2}{\nu}} \frac{\nu U_0^{1/2}}{R_0^{1/2}} \end{aligned}$$

CHAPTER 3

# Running MITgcm on multiple processors

The main source of information is the model's manual compilation section: http://mitgcm.org/public/r2_manual/latest/online_documents/node94.html . You can get the code from MITgcm.org via CVS.

Compile the code:

As described on the documentation, MITgcm uses the "make" program to compile the code. MITgcm provides a script genmake2 that creates a makefile used by make. This file allows the model to pre-process source files, specify the compiler and figure out dependencies. Afterwards, you need to build the dependencies and compile the code.

Again, there is a detailed explanation of how to build the code in the documentation above.

## 3.1 Specific hints and instructions for mpi runs

The two machines where we have run MITgcm in multiple processors are Westgrid's Bugaboo and Salish. The optfiles for each machine are under the optfiles repo. They have been adapted from other example optfiles on the MITgcm website.

Tips and hints for Westgrid's Bugaboo

- optfile: bugaboo_mpi.opt.

Tips and hints for Salish

- optfile: salish_mpi.opt
- To avoid broken pipe errors, execute mitgcmuv in the background (use & at the end of the mpi run command) and exit the session. If you want to monitor the process, start a new session.

CHAPTER 4

# Calculate numerical diffusivity

This module has functions to calculate the numerical diffusivity experienced by a tracer, associated to a specific configuration of the MITgcm. In particular, it was developed to calculate the equivalent diffusivity $kappa$, defined (here) as $kappa = kappa_{pres}+kappa_{num}$, where $kappa_{pres}$ is the prescibed or explicit tracer diffusivity one imposes on the model and $$k_{num}$$ is the additional diffusivity due to numerical truncation errors. Note that there are two $$kappa_{pres}$$ and therefore two $kappa$, one for the horizontal dimensions and one for the vertical one.

These calculations try to reproduce the method used by [1] Abernathy et al. 2010, [2] Hill et al. 2011, and [3] Leibensperger and Plumb, 2013 to determine the numerical diffusivity MITgcm Southern Ocean configurations [1,2] and a baroclinic flow simulation simulation [3].

The idea is that from the evolution equation for the variance of the tracer concentration in the model output

begin{equation} frac{1}{2}frac{partial{overline{q^{2}}}}{partial{t}}=-kappa_{h} overline{|nabla_h q|^2}-kappa_{v} overline{(frac{partial{q}}{partial {z}})^{2}} end{equation}

one can fit by a least squares regression, suitable values of $kappa_h$ and $kappa_v$ that satisfy the equation.

## 4.1 Calculate the volume of the domain (function: CalcDomVolume)

The volume of a tracer cell (remember we have an Arakawa C grid, so this changes depending on which kind of cell we are thinking about) is given by

$V(i,j,k)=depth times area = (hfacC(i,j,k)times dRf(k)) times rA(i,j) = (hfacC(i,j,k)times dRf(k)) times dXg(i,j) times dYg(i,j)$,

where hfacC is the fraction of the cell that is open (not occupied with land). So, the total volume of the domain is

$sumlimits_{i=1}^{nx}{sumlimits_{j=1}^{ny}{sumlimits_{k=1}^{nz}{(hfacC(i,j,k)times dRf(k)) times rA(i,j)}}}$

## 4.2 1st Term: The volume-weighted average of the squared concentration (function: CalcVariance, CalcTimeDer)

The first term in the variance evolution equation is $frac{1}{2}frac{partial{overline{q^{2}}}}{partial{t}}$. Note that we care about the time derivative of the variance, so that the mean concentration that usually appears in the definition of variance will not play a role here, since it is constant in time (we are not putting in or letting out any tracer).

We are going to calculate $overline{q^2}$, the volume-weighted average of the squared concentration, and then the time derivative of that using a centered difference scheme.

## 4.3 2nd Term: The volume-weighted average of the squared horizontal gradient (function: CalcAvgHorGrad)

The second term in the variance evolution equation is $-kappa_{h} overline{**|\nabla_h q|^2}$. Next, we calculate the square of the horizontal gradient $**|nabla_h q|^2=(frac{partial{q}}{partial{x}})^2+(frac{partial{q}}{partial{y}})^2$.

Spatial derivatives are approximated using a centered-difference scheme.

## 4.4 3rd Term: The volume-weighted average of the squared vertical derivative (function: CalcAvgVerGrad)

The third term in the variance evolution equation is $-kappa_{v} overline{(frac{partial{q}}{partial{z}})^2}$. Next, we calculate the square of the vertical gradient $(frac{partial{q}}{partial{z}})^2$.

The vertical derivative is approximated using a centered-difference scheme.

CHAPTER 5

# Building and Running MITgcm

## 5.1 Working on `orcinus`

This section describes the steps to set up and run the MITgcm code for the UBC EOAS Canyons group configurations on the orcinus.westgrid.ca HPC cluster.

When working on the Westgrid clusters the **`module`** command must be used to load extra software components. The required modules vary from cluster to cluster. On `orcinus` load the `python` module with:

```
$ module load python
```

to make Python 2.7, the python-netCDF4 package, and Mercurial available to you.

### 5.1.1 Create a Workspace and Get the Repos

```
$ mkdir -p $HOME/canyons
$ cd $HOME/canyons
```

Use the CVS version control tool to do a checkout of the latest MITgcm source code. Use the password `cvsanon` when the **`cvs login`** command prompts you for a password:

```
$ export CVSROOT=':pserver:cvsanon@mitgcm.org:/u/gcmpack'
$ cvs login
$ cvs co -P MITgcm
```

Use the Mercurial version control tool to clone the CanyonsUBC optfiles repo from Bitbucket:

```
$ hg clone ssh://hg@bitbucket.org/canyonsubc/optfiles
```

### 5.1.2 Building the Code

The MITgcm docs describe several ways of building the code. Here, we will do the build in a directory outside of the `MITgcm` and `optfiles` directory trees.

**Note:** For the purposes of developing the build instructions for `orcinus` the `MITgcm/verification/rotating_tank/` configuration is used, but the steps below should be adaptable to your research configuration(s).

Create a configuration build directory:

```
$ cd $HOME/canyons
$ mkdir -p rotating_tank/build
$ cd rotating_tank/build
```

Build the code:

```
    $ $HOME/canyons/MITgcm/tools/genmake2 \
        -rootdir=$HOME/canyons/MITgcm \
        -mods=$HOME/canyons/MITgcm/verification/rotating_tank/code
        -of=$HOME/canyons/optfiles/orcinus_mpi.opt \
        -mpi
    $ module load intel
    $ module load intel/14.0/netcdf_hdf5
    $ make depend
    $ make
```

The **`module load`** commands bring the Intel OpenMPI Fortran compiler, and its netcdf and hdf5 libraries into your environment for the **`make`** steps. Those modules are also required to run the code, so you need to include those **`module load`** commands in your PBC script. However, due to some weirdness in the `orcinus` modules setup, they *must not* be loaded when you run **`MITgcm/tools/genmake2`**. So, if you need to run **`genmake2`** again, make sure that you first do:

```
$ module unload intel
$ module unload intel/14.0/netcdf_hdf5
```

and then re-load the modules before running **`make`**.

CHAPTER 6

# Indices and tables

- genindex
- modindex
- search