Salish Sea MEOPAR Documentation

Salish Sea MEOPAR Project Contributors

Apr 30, 2019
CONTENTS

1 About the Project 3

2 Contents 5
  2.1 Organization of Mercurial Repositories ................................. 5
  2.2 Working Environment .................................................. 13
  2.3 Salish Sea NEMO ......................................................... 33
  2.4 Salish Sea Model Results Server ........................................ 86
  2.5 NEMO 3.6 Model Evaluation: 201702 .................................... 90
  2.6 Tidal evaluation .......................................................... 113
  2.7 Tidal currents evaluation ................................................ 134
  2.8 Rivers ................................................................. 164
  2.9 Stability ................................................................. 175
  2.10 SMELT Documentation .................................................. 179
  2.11 SMELT Sensitivity Analysis .............................................. 206
  2.12 Storm Surges .......................................................... 208
  2.13 Turbulence and Viscosity ............................................... 218
  2.14 Particle Tracking ....................................................... 223
  2.15 Things We Learned About NEMO ....................................... 237
  2.16 Citations ............................................................... 276
  2.17 Salish Sea MEOPAR Project Contributors .............................. 278

3 License 281

Bibliography 283
This is a collection of documentation about the Salish Sea MEOPAR project. There is a companion collection of project tools documentation.
CHAPTER ONE

ABOUT THE PROJECT

The Salish Sea is home to a large population of Canadians living in coastal communities at risk to ocean related hazards. There is an ongoing need to assess the impact of these hazards on human and marine environments through a multidisciplinary approach involving Canadian oceanographers, biologists, and social scientists. The Marine Environmental Observation Prediction and Response network (MEOPAR) provides a platform to accelerate this type of research.

The Salish Sea MEOPAR project team is developing a three-dimensional ocean model for the Strait of Georgia and Salish Sea. Using the NEMO modelling architecture the Salish Sea model will be used to evaluate storm surge risk in coastal communities. A long term goal is to include a coupled biogeochemical modelling component.
2.1 Organization of Mercurial Repositories

2.1.1 General

- The central location for all repos is the SalishSea-MEOPAR team account on Bitbucket
- People are free to push changes to code and docs that they own, but they should fork repos and create pull requests to provide changes to stuff owned by others so that changes are reviewed
- All repos have issue trackers enabled on Bitbucket
- Bitbucket wikis are disabled on all repos to force documentation into repo docs directories
- Docs use Sphinx
- Docs from public repos will be rendered as HTML on readthedocs.org. readthedocs also provides PDF rendering.
- There is no provision for rendering docs from private repos.

2.1.2 Repositories

docs

- public
- Creative Commons Attribution license, copyright project contributors and UBC
- project level documentation, reports, etc.
- Bitbucket: https://bitbucket.org/salishsea/docs/

private-docs

- private to SalishSea-MEOPAR team members
- meeting notes, plans, work-in-progress reports and papers, presentations, etc.
- anything that might go in docs repo but are not ready for release, or which cannot be released for some reason
- ideally at least some of this repo will eventually be moved to the public docs repo
tools

- public
- Apache v2.0 license, copyright project contributors and UBC
- scripts and docs for preparing, running, and post-processing NEMO runs, and any other support task that we write software for
- documentation is rendered at https://salishsea-meopar-tools.readthedocs.io/en/latest/
- Bitbucket: https://bitbucket.org/salishsea/tools/

private-tools

- private to SalishSea-MEOPAR team members
- scripts and docs that might go in tools repo but are not ready for release, or which cannot be released for some reason
- ideally this repo will eventually empty out as its contents are moved to the public tools repo
- Bitbucket: https://bitbucket.org/salishsea/private-tools/

SalishSeaCmd

- public
- Apache v2.0 license, copyright project contributors and UBC
- The Salish Sea NEMO command processor, salishsea, a command line tool for doing various operations associated with the Salish Sea NEMO model. It is a domain-specific command processor tool for the Salish Sea model that uses plug-ins from the NEMO-Cmd package.
- documentation is rendered at https://salishseacmd.readthedocs.io/en/latest/
- Bitbucket: https://bitbucket.org/salishsea/salishseacmd/

NEMO-Cmd

- public
- Apache v2.0 license, copyright project contributors and UBC
- The NEMO command processor, nemo, a command line tool for doing various operations associated with running the NEMO ocean model.
- documentation is rendered at https://nemo-cmd.readthedocs.io/en/latest/
- Bitbucket: https://bitbucket.org/salishsea/nemo-cmd/

SalishSeaNowcast

- public
- Apache v2.0 license, copyright project contributors and UBC
• The SalishSeaNowcast package is a collection of Python modules associated with running the Salish Sea NEMO model in a daily nowcast/forecast mode. The runs use as-recent-as-available (typically previous day) forcing data for the western boundary sea surface height and the Fraser River flow, and atmospheric forcing from the four-times-daily produced forecast results from the Environment Canada High Resolution Deterministic Prediction System (HRDPS) operational GEM 2.5km resolution model.

• documentation is rendered at https://salishsea-nowcast.readthedocs.io/en/latest/
• Bitbucket: https://bitbucket.org/salishsea/salishseanowcast/

analysis

• Read-only legacy repo
• public
• Apache v2.0 license, copyright project contributors and UBC
• analyses of the results of the Salish Sea MEOPAR NEMO model; most of the files are Jupyter Notebooks
• Bitbucket: https://bitbucket.org/salishsea/analysis/

Note: This was the project’s original analysis repo. It got so big that it was in danger of hitting the Bitbucket 2Gb limit at which changesets would no longer be accepted. In April 2016 it was split into a collection of personal analysis repos for each team member, as well as some other application-specific analysis repos. They are all public with the same license and copyright as the original analysis repo. They can be found in the SalishSea-MEOPAR Analysis project collection of repos on Bitbucket.

NEMO-3.6-code

• private (because NEMO project requires sign-in to access code)
• CeCILL license, copyright Centre National de la Recherche Scientifique CNRS
• NEMO-3.6 code that we run
• a merge of the http://forge.ipsl.jussieu.fr/nemo/svn/branches/2015/nemo_v3_6_STABLE and our local code
• maintenance of the repo to handle the merges is described in NEMO-3.6 Code Repo Maintenance
• Bitbucket: https://bitbucket.org/salishsea/nemo-3.6-code/

XIOS-2 (XML I/O Server)

• private (because the canonical source for this code is http://forge.ipsl.jussieu.fr/ioserver/wiki)
• CeCILL_V2 license, copyright Centre National de la Recherche Scientifique CNRS
• XIOS-2.0 code that we run with NEMO-3.6
• a checkout of http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk

2.1. Organization of Mercurial Repositories
- maintenance of the repo to handle the merging in changes from upstream is described in *XIOS-2 Code Repo Maintenance*

  - Bitbucket: [https://bitbucket.org/salishsea/xios-2/](https://bitbucket.org/salishsea/xios-2/)

**XIOS-ARCH**

- public
  - Apache v2.0 license, copyright project contributors and UBC
  - XIOS build configuration arch files for use with NEMO-3.6
  - Bitbucket: [https://bitbucket.org/salishsea/xios-arch](https://bitbucket.org/salishsea/xios-arch)
  - documentation: for *salish*, for *westgrid machines*

**grid**

- public
  - Apache v2.0 license, copyright project contributors and UBC
  - Coordinates, bathymetry, mesh mask, land processor elimination, and atmospheric forcing weights files for the Salish Sea NEMO model. Also included is the file that imposed the “no snow” atmospheric forcing condition. Files for AGRIF sub-grids are in appropriately named sub-directories of the *subgrids/* directory.

  Many of these files originated in the (private) *NEMO-forcing* repository. The commit message history of files from that repository is included in their initial commit messages.

  - Bitbucket: [https://bitbucket.org/salishsea/grid](https://bitbucket.org/salishsea/grid)

**rivers-climatology**

- public
  - Apache v2.0 license, copyright project contributors and UBC
  - Rivers runoff climatology files for the Salish Sea NEMO model. Included are discharge, temperature, nutrients & biological tracers, and depths over which river quantities are introduced into model grid. Files for AGRIF sub-grids are in appropriately named sub-directories of the *subgrids/* directory.

  Many of these files originated in the (private) *NEMO-forcing* repository. The commit message history of files from that repository is included in their initial commit messages.

  - Bitbucket: [https://bitbucket.org/salishsea/rivers-climatology](https://bitbucket.org/salishsea/rivers-climatology)

**tides**

- public
  - Apache v2.0 license, copyright project contributors and UBC
  - Tide boundary condition files for the Salish Sea NEMO model.

  Many of these files originated in the (private) *NEMO-forcing* repository. The commit message history of files from that repository is included in their initial commit messages.
• Bitbucket: https://bitbucket.org/salishsea/tides

**tracers**

• public

• Apache v2.0 license, copyright project contributors and UBC

• Tracer boundary condition climatology files and domain initialization files for the Salish Sea NEMO model. Included are temperature, salinity, nutrients, and biological tracers.

  Many of these files originated in the (private) *NEMO-forcing* repository. The commit message history of files from that repository is included in their initial commit messages.

• Bitbucket: https://bitbucket.org/salishsea/tracers

**NEMO-forcing**

• **Read-only** legacy repo

• private to SalishSea-MEOPAR team members

• private because the files are from project initialization tarballs received from J-P Paquin on 2013-10-02

• domain-specific set-up, initial conditions, forcing, etc. files used to run NEMO for the Salish Sea:
  – coordinates
  – bathymetry
  – initial temperature and salinity
  – restart files from spin-up runs
  – tidal forcing
  – open boundary condition forcing
  – etc.

• Bitbucket: https://bitbucket.org/salishsea/nemo-forcing/


**Note:** This was the project’s original repo of initial conditions, forcing, etc. files. It got so big that it was in danger of hitting the Bitbucket 2Gb limit at which changesets would no longer be accepted. It also contained multiple copies large binary files which are no longer used. Together, these factors make this repo very time consuming to clone and update. In August 2017 the repo was split into 4 smaller repos:

  • grid
  • rivers-climatology
  • tides
  • tracers

They are all public, licensed under Apache v2.0 license, and copyright by the project contributors and UBC. They can be found in the SalishSea-MEOPAR NEMO Model Runs collection of repos on Bitbucket.
SS-run-sets

- public
- Apache v2.0 license, copyright project contributors and UBC
- a collection of namelists and run description files for various sets of NEMO runs
- Bitbucket: https://bitbucket.org/salishsea/ss-run-sets/

salishsea-site

- public
- Apache v2.0 license, copyright project contributors and UBC
- code and documentation for the dynamic web app based on the Pyramid framework that serves the salishsea.eos.ubc.ca domain
- Bitbucket: https://bitbucket.org/salishsea/salishsea-site/
- documentation: https://salishsea-site.readthedocs.io/en/latest/

results

- public
- all rights reserved, copyright project contributors and UBC
- a collection of model results and analysis produced by the Salish Sea MEOPAR project
- Bitbucket: https://bitbucket.org/salishsea/results/

Storm-Surge

- public
- Apache v2.0 license, copyright project contributors and UBC, except the manuscript files which are copyright Taylor and Francis
- Salish Sea MEOPAR storm surge paper
- Bitbucket: https://bitbucket.org/salishsea/storm-surge/

Barotropic-Tides

- private until paper is published
- development of the Salish Sea NEMO barotropic tides paper
- Bitbucket: https://bitbucket.org/salishsea/barotropic-tides
internal-tides

- private until paper is published
- development of the Salish Sea NEMO interal tides paper
- Bitbucket: https://bitbucket.org/salishsea/internal-tides

mixing-paper

- private until paper is published
- development of the Salish Sea NEMO mixing paper
- Bitbucket: https://bitbucket.org/salishsea/mixing-paper

plume-paper

- private until paper is published
- development of the Salish Sea NEMO plume paper
- Bitbucket: https://bitbucket.org/salishsea/plume-paper

XIOS (XML I/O Server)

- private (because the canonical source for this code is http://forge.ipsl.jussieu.fr/ioserver/wiki)
- CeCILL_V2 license, copyright Centre National de la Recherche Scientifique CNRS
- XIOS-1.0 code that we run with NEMO-3.6
- a checkout of the http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/branches/xios-1.0 branch
- maintenance of the repo to handle the merging in changes from upstream is described in NEMO-MirrorMaintenance
- Bitbucket: https://bitbucket.org/salishsea/xios/

Note: This is the XIOS-1.0 code. It is archival. The Salish Sea model now uses XIOS-2 (XML I/O Server), and that is the recommended code for any new projects.

NEMO-code

- private (because NEMO project requires sign-in to access code)
- CeCILL license, copyright Centre National de la Recherche Scientifique CNRS
- NEMO-3.4 code that we run
- a merge of the http://forge.ipsl.jussieu.fr/nemo/svn/branches/2012/dev_v3_4_STABLE_2012 branch and our local code

2.1. Organization of Mercurial Repositories
maintenance of the repo to handle the merges is described in NEMO-MirrorMaintenance

- **Bitbucket**: https://bitbucket.org/salishsea/nemo-code/

**Note:** This is the NEMO-3.4 code. It is archival. The Salish Sea model is now based on **NEMO-3.6-code**, and that is the recommended code for any new projects.

---

**SoG-obs**

- private
- a collection of observations made in the Salish Sea
- This repo is a vestige of an experiment that didn’t work out because storing large binary files under Mercurial version control is not a good idea.
- **Bitbucket**: https://bitbucket.org/salishsea/sog-obs/

**NEMO_EastCoast**

- public
- a collection of files for pre-processing, running, and post-processing of numerical simulations with NEMO for MEOPAR project with primary focus on the Scotia Shelf deployment of the model
- **Bitbucket**: https://bitbucket.org/salishsea/nemo_eastcoast/

**NEMO-3.1**

- private (because NEMO project required sign-in to access code)
- CeCILL license, copyright Centre National de la Recherche Scientifique CNRS
- NEMO-3.1 reference repo
- a Mercurial repo of SVN checkouts of modipsl trunk, the NEMO-3.1 tag, and supporting repos that are believed to be the basis on which the 2-Oct-2013 CONCEPTS-110 CODE tarball was built
- **Bitbucket**: https://bitbucket.org/salishsea/nemo-3.1/

**CONCEPTS-110**

- private (because NEMO project required sign-in to access code)
- CeCILL license, copyright Centre National de la Recherche Scientifique CNRS
- CONCEPTS-110 reference repo
- a Mercurial repo of the CODE.tar tarball received from J-P Paquin on 2-Oct-2013
2.2 Working Environment

Instructions and recommendations for setting up your computational working environment for the Salish Sea MEOPAR project, and information about the major tools that we use.

Team members at UBC will typically set up their first working environment on a Waterhole workstation running Linux. If you are running the Salish Sea NEMO model you will eventually need to set up working environments on several machines.

The Waterhole workstations are a collection of Linux workstations owned co-operatively by Susan Allen, Rich Pawlowicz, Stephanie Waterman, and Roger Pieters. They are maintained by EOAS Comp Staff, primarily Charles Krzysik. They are mostly located in ESB 3024/3026 but some are located in users’ offices. The Waterhole machines are nominally identicaly configured so that you should be able to log-in on all of them and find the same software available for use. They are set up so that your user configuration is identical on all machines. All machines have a /ocean/ partition which stores storage that is large (several terabytes), redundant (RAID), and automatically backed up. That storage space is on the ocean.eos.ubc.ca storage server (also co-operatively owned and maintained by the same group). Each user has their own directory on /ocean/ named with their EOAS user id; e.g. /ocean/sallen/. You should use your /ocean/ directory as your primary file storage area.

Other machines that you may need working environments on later include:

- salish.eos.ubc.ca: Our group compute server that runs Linux and has several disk mounts in common with Waterhole workstations. salish is primarily used for short development runs of the Salish Sea NEMO model. salish has several terabytes of storage in its /data/ filesystem. If your Waterhole workstation does not have access to /data/ you should open a ticket via the Helpdesk link on https://helpdesk.eoas.ubc.ca/ to request that EOAS Comp Staff add a salish /data/ mount on the workstation you are using. That will enable you to read/write files on the salish /data/ filesystem without have to sign on to salish or copy the files from one machine to another.

- One or more of the Westgrid or ComputeCanada HPC clusters such as orcinus.westgrid.ca, cedar.computecanada.ca, or graham.computecanada.ca that run Linux. Those machines are used for longer research runs of the model.

skookum.eos.ubc.ca is our group results storage and web server. The Salish Sea Nowcast system results are stored on the /results/ file system on skookum, and the Strait of Georgia WaveWatch3 model results and Vancouver Harbour/Fraser River FVCOM model results are stored on the /opp/ file system on skokum. If your Waterhole workstation does not have access to /results/ and/or /opp/ you should open a ticket via the Helpdesk link on https://helpdesk.eoas.ubc.ca/ to request that EOAS Comp Staff add skookum /results/ and/or /opp/ mount(s) on the workstation you are using. That will enable you to read files from the skookum /results/ and/or /opp/ filesystem(s) without have to sign on to skookum or copy the files from one machine to another.

2.2.1 bash Configuration

All of the Salish Sea MEOPAR documentation assumes that you are using the bash shell. You can check which shell you are using with:

```
echo $SHELL
```

If the result is not something like /bin/bash you may be able to start bash with:
bash

You may want to request that your IT support change your default shell to bash to avoid having to start bash every time that you open a new shell window/tab. For the Waterhome workstations and salish you can open a ticket with the request on the EOAS Compostaff Helpdesk system. On Westgrid, send an email request to support@westgrid.ca.

If you are new to bash or the Linux command line the copy of “The Linux Command Line” by William E. Shotts, Jr. in the Waterhole will help you, as will this Unix Shell Quick Reference page from Software Carpentry.

.bashrc Snippets

bash executes the commands in $HOME/.bashrc every time a new shell window/tab is opened. If you do not have a $HOME/.bashrc file you can create it using your favourite editor.

The rest of this section described various snippets of bash code that you may want to include in your $HOME/.bashrc file.

To shorten your prompt so that it shows just the name of the machine that you are on and the directory that you are currently in instead of the whole path to that directory use:

PS1="\h:\W$ "

To force programs and commands that want to display output page by page to use less as their pager use:

export PAGER=less

To force less to allow control sequences that change the colour of output to work use:

export LESS=-R

If you are not a fan of the vi editor you can set the EDITOR and VISUAL environment variables to the command for your favourite editor and export them. For emacs use:

export EDITOR=emacs
export VISUAL=emacs

The $HOME/bin/ directory is the conventional place keep your own scripts so it is a good idea to add that directory to the end of your PATH:

export PATH=$PATH:$HOME/bin

The $HOME/.local/bin/ directory is where Python scripts installed via the --user option are stored so it should be near the beginning of your PATH:

export PATH=$HOME/.local/bin:$PATH

If you are using the Anaconda Python distribution you should add its bin/ directory to the beginning of your PATH:

export PATH=$HOME/anaconda3/bin:$PATH

Aliases allow you to run commands with different names or with particular option flags set.

To make the ls command use different colours for regular files, executable files, directories, symbolic links, etc., and post-fix characters to indicate those file types use:

alias ls="ls --color=auto -F"
To make `la` include hidden files in file listings use:

```
alias la="ls -a"
```

To make `ll` display long file listings that include permissions, owner and group, and last modification date/time, and also include hidden files use:

```
alias ll="ls -al"
```

Aliases are cumulative, so if the above three aliases are all defined in the order shown, `ls`, `la`, and `ll` will all produce coloured, post-fixed file listings.

To always be prompted to confirm file removals use:

```
alias rm="rm -i"
```

Some things cannot be easily accomplished with aliases and so `bash` also provides a way of writing functions. One good use for `bash` functions is creating commands that change directories for you to particular locations without having to type long paths. For example:

```
go_results() {
  cd /ocean/$USER/MEOPAR/SalishSea/results;
}
```

creates the `go_results` command that will `cd` from wherever you are to the directory where your Salish Sea NEMO model run results are stored.

**.bash_profile**

To ensure that `.bashrc` is executed when you login via ssh create a file `$HOME/.bash_profile` with the following lines:

```
if [ -f ~/.bashrc ]; then
  . ~/.bashrc;
fi
```

**Loading Modules on HPC Clusters**

When working on ComputeCanada or Westgrid clusters the `module` command must be used to load several software components required to compile, run, and work with the results of NEMO. The required modules vary from machine to machine:

- **On** `cedar` and `graham` the `module load` commands you should include in your `$HOME/.bashrc` are:

  ```
  module load python/3.7.0
  module load netcdf-fortran-mpi/4.4.4
  ```

- **On** `orcinus` the `module load` commands you should include in your `$HOME/.bashrc` are:

  ```
  module load python
  module load intel
  module load intel/14.0/netcdf-4.3.3.1_mpi
  module load intel/14.0/netcdf-fortran-4.4.0_mpi
  module load intel/14.0/hdf5-1.8.15p1_mpi
  module load intel/14.0/nco-4.5.2
  ```

2.2. Working Environment
You can inspect the collection of modules that are loaded with the `module list` command; for example, on `cedar`:

```
module list
Currently Loaded Modules:
  1) nixpkgs/16.09 (S) 4) ifort/.2016.4.258 (H) 7) openmpi/2.1.1 (m) 10) →
       → hdf5-mpi/1.8.18   (io)
  2) icc/.2016.4.258 (H) 5) intel/2016.4 (t) 8) StdEnv/2016.4 (S) 11) →
       → netcdf-mpi/4.4.1.1 (io)
  3) gcccore/.5.4.0  (H)  6) imkl/11.3.4.258 (math) 9) python/3.7.0 (t) 12) →
       → netcdf-fortran-mpi/4.4.4 (io)
```

Where:
- S: Module is Sticky, requires `--force` to unload or purge
- m: MPI implementations / Implémentations MPI
- math: Mathematical libraries / Bibliothèques mathématiques
- io: Input/output software / Logiciel d'écriture/lecture
- t: Tools for development / Outils de développement
- H: Hidden Module

### 2.2.2 Version Control with Mercurial

Please see the Version Control with Mercurial section of the UBC EOAS MOAD Group Documentation.

### 2.2.3 Anaconda Python Distribution

The Anaconda Python distribution is the easiest way to install Python and a collection of scientific packages and other tools (Sphinx, Jupyter Notebook, NumPy, matplotlib, to name a few) that we use in the Salish Sea MEOPAR project.

Choose the Python 3 download from the installation instructions for your operating system, follow the rest of those instructions, and accept the option at the end of the installation to make Anaconda your default Python.

If you are installing Anaconda Python on a Waterhole machine, choose the Python 3 Linux 64-bit download. Thanks to the shared storage and user configurations across all of the Waterhole machines and salish each user only needs to do the installation once on a Waterhole machine for the packages to be available on all of those machines. There is no need to install Anaconda on the Westgrid machines.

Start a new shell session and confirm that `$HOME/anaconda3/bin/` is on your path.

There is are a few packages that we use extensively that is not included in the base Anaconda installation:

- `basemap`
- `nbsphinx`
- `netcdf4-python`
- `sphinx_rtd_theme`
- `xarray`

To install them, do:

```
conda install basemap hdf4 netcdf4 xarray
```

and follow the prompts. Then do:

```
pip install nbsphinx
```
2.2.4 Salish Sea Repos and Packages

Once you have your Mercurial Configuration set up and the Anaconda Python Distribution installed and activated you need to set up a working area, clone some Mercurial repositories, and install some Salish Sea MEOPAR project tools.

In the working environment on your Waterhole workstation you should create a work space on the /ocean/ partition:

```
mkdir -p /ocean/$USER/MEOPAR
```

On your own laptop, well, the choice of where you put files is up to you...

In the work space you need to clone a number of Salish Sea MEOPAR project repos from Bitbucket. The repos that you need depends whether the environment is for running the model, doing analysis, etc.; the Quick Start Guide provides guidance for the repos to install in model run environments. Here, we’ll assume that you are setting up your Waterhole machine working environment where you will be doing analysis, documentation, etc.

```
cd /ocean/$USER/MEOPAR/
hg clone ssh://hg@bitbucket.org/salishsea/docs
hg clone ssh://hg@bitbucket.org/salishsea/tools
hg clone ssh://hg@bitbucket.org/salishsea/nemo-cmd NEMO-Cmd
hg clone ssh://hg@bitbucket.org/salishsea/salishseacmd SalishSeaCmd
```

Next, install the SalishSeaTools Package and the Salish Sea NEMO Command Processor:

```
pip install --user --editable tools/SalishSeaTools
pip install --user --editable NEMO-Cmd
pip install --user --editable SalishSeaCmd
```

The links above contain information about the contents of those packages.

Next, you need a repo to store you analysis notebooks and other bits of code, text, etc. in. We used to all share a single analysis repo, but it got too big. So, we broke it up so that each team member has their own analysis repo, and there are a few special purpuse analysis repos. You can find them all at https://bitbucket.org/account/user/salishsea/projects/SSM_ANALYSIS. The original analysis repo is now read-only.

If you are joining the team as a researcher there should be an analysis repo already set up for you. Its name will be something like analysis-james, but with your name. Go ahead and clone that repo too:

```
cd /ocean/$USER/MEOPAR/
hg clone ssh://hg@bitbucket.org/salishsea/analysis-james
```

If you are joining the team for a sprint, please clone the analysis-sprints repo:

```
cd /ocean/$USER/MEOPAR/
hg clone ssh://hg@bitbucket.org/salishsea/analysis-sprints
```

and create a directory in it named after yourself to work in.

2.2.5 Python, netCDF, and Model Results Visualization

Intro to Python

Most of the analysis and tools in the Salish Sea MEOPAR project are written in Python, though Matlab makes occasional guest appearances.

This slide deck from a physics course at Cornell University provides a good, fairly detailed, introduction to Python for people who already know at least one programming language. Of course, no two groups make exactly the same
choices within a language and the few differences to our choices are detailed below. Also, don’t get too bogged down in the details of object-oriented and functional programming (especially slides 18 through 22) as we don’t use those aspects much.

A few differences you will see compared to our Python code:

- The Cornell course uses an older syntax for string interpolation:

  ```python
  print 'value of %s = %s' % (name, val)
  ```

  In our notebooks and code you are more likely to see that spelled like:

  ```python
  print('value of {n} = {v}'.format(n=name, v=val))
  ```

  or perhaps:

  ```python
  print('value of {0} = {1}'.format(name, val))
  ```

- The `scipy.array` syntax discussed on slides 25 through 28 is a synonym for `numpy.ndarray` and you will see it used in our code as:

  ```python
  import numpy as np
  a = np.array([[1,2,3], [4,5,6], [7,8,9]])
  ...
  p = np.arange(0.,1.,0.1)
  etc.
  ```

- The `pylab` namespace mentioned on slide 31 is a Matlab-like interface to the `Matplotlib` library. In our code we try to use the `pyplot` object-oriented interface, so you will see things like:

  ```python
  import matplotlib.pyplot as plt
  import numpy as np
  xvals = np.linspace(-10., 10., 100)
  yvals = xvals**3
  fig, (ax1, ax2, ax3) = plt.subplots(1, 3)
  ax1.plot(xvals, yvals)
  ax2.plot(xvals, yvals, 'r.'
  ax3.hist(yvals)
  ```

**Jupyter Notebook, netCDF, and Model Results**

We have an ongoing project to develop a collection of Jupyter Notebooks that provide discussion, examples, and best practices for plotting various kinds of model results from netCDF files. There are code examples in the notebooks and also examples of the use of functions from the SalishSeaTools Package.

If you are new to the Salish Sea project, or to Jupyter Notebook, netCDF, and Matplotlib you should read the notebooks in the following order:

- Exploring netCDF Files.ipynb
- Plotting Bathymetry Colour Meshes.ipynb
- Plotting Tracers on Horizontal Planes.ipynb
- Plotting Velocity Fields on Horizontal Planes.ipynb
- Plotting Velocities and Tracers on Vertical Planes.ipynb
ERDDAP and xarray

From late-2013 until early-2016 we used the netCDF4-python library to open locally stored files. The notebooks above describe that way of working. In early-2016 we set up an ERDDAP server to provide public access to our model results. The netCDF4-python library can open datasets from ERDDAP URLs just as easily as it can open them from local files. So, here is a reworking of the Exploring netCDF Files.ipynb notebook using ERDDAP:

- Exploring netCDF Datasets from ERDDAP.ipynb

One reason that you might want to use ERDDAP to access our model results is if you don’t have access to our results files stored on the UBC EOAS Ocean cluster. Our ERDDAP server is public.

Another reason to use ERDDAP is that it provides access to the daily model results as continuous data streams, hiding the fact that they are stored in per-day files. ERDDAP makes it much easier to work with a dataset that spans multiple days because it removes the task of opening each day's file(s) and splicing the variable values into arrays. You can just ask for a slice of the dataset in time and space and ERDDAP takes care of the slicing and splicing (provided that the resulting dataset is less than 2Gb in size).

Another new development is the xarray package. Quoting from the introduction to its documentation:

  xarray ... is an open source project and Python package that aims to bring the labeled data power of pandas to the physical sciences, by providing N-dimensional variants of the core pandas data structures.

  Our goal is to provide a pandas-like and pandas-compatible toolkit for analytics on multi-dimensional arrays, rather than the tabular data for which pandas excels. Our approach adopts the Common Data Model for self-describing scientific data in widespread use in the Earth sciences: xarray.Dataset is an in-memory representation of a netCDF file.

Here is a reworking of the Exploring netCDF Files.ipynb notebook using xarray:

- Exploring netCDF Datasets Using xarray.ipynb

xarray uses the netCDF4-python library so it is capable of accessing netCDF datasets from either local files or from ERDDAP servers. The xarray.Dataset object hides many of the low level details of the netcdf4.Dataset objects to provide a more Pythonic interface to the dataset that is heavily inspired by pandas. Like pandas variables, xarray variables have a plot() method that makes quick visualization of datasets very easy.

xarray provides sophisticated handling of the time coordinate of datasets. In combination with ERDDAP that feature makes accessing arbitrary length time slices from the daily Salish Sea Nowcast system results collection very easy.

In summary, you can think of ERDDAP as a higher level abstraction for storage of our model results, and xarray as a higher level abstraction for working with the results as Python objects. The ERDDAP abstraction hides some of the discrete daily runs storage details, and the xarray abstraction hides some of the netCDF4 file structure details.

Here is a notebook that demonstrates some of the features of xarray combined with accessing model results from our ERDDAP server:

- Exploring a Nowcast Time Series from ERDDAP.ipynb

2.2.6 Jupyter Notebook READMEs on Bitbucket

Bitbucket.org has a feature whereby a README.md file containing text and Markdown markup present in any directory is rendered to HTML below the list of files in that directory. See https://bitbucket.org/salishsea/analysis-ben/src/tip/notebooks/ as an example.
We can use that feature in directories that contain Jupyter Notebook files to provide links to our notebooks rendered to HTML by the Jupyter Notebook Viewer service. Doing so makes the notebooks easily visible to anyone without having to run Jupyter Notebook. It is also an easy way to generate notebook viewer links to paste into the Google Drive “whiteboard” documents for weekly group meetings.

You could hand edit the README.md file, but that’s tedious and error prone, so it is an obvious candidate for code automation. Here is a prototype make_readme.py module that provides that automation:

```python
"""Salish Sea NEMO Jupyter Notebook collection README generator
""
import datetime
import glob
import json
import os
import re

NBVIEWER = 'https://nbviewer.jupyter.org/urls'
REPO = 'bitbucket.org/salishsea/tools/raw/tip'
REPO_DIR = 'SalishSeaNowcast/notebooks/figures/publish'
TITLE_PATTERN = re.compile('#{1,6} ?

def main():
    url = os.path.join(NBVIEWER, REPO, REPO_DIR)
    readme = ""

    The Jupyter Notebooks in this directory are for development and testing of the results figures generation modules of the Salish Sea model nowcast system.
    The links below are to static renderings of the notebooks via [nbviewer.jupyter.org](https://nbviewer.jupyter.org/).
    Descriptions under the links below are from the first cell of the notebooks (if that cell contains Markdown or raw text).

    ""
    for fn in glob.glob('*.ipynb'):
        readme += '* ##[{fn}](%s/{fn})

        license = ""
        notebook_description(fn)"
```

(continues on next page)
and The University of British Columbia.

They are licensed under the Apache License, Version 2.0.
https://www.apache.org/licenses/LICENSE-2.0
Please see the LICENSE file for details of the license.
{}

```python
with open('README.md', 'wt') as f:
    f.writelines(readme)
    f.writelines(license)
```

```python
def notebook_description(fn):
    description = ''
    with open(fn, 'rt') as notebook:
        contents = json.load(notebook)
        try:
            first_cell = contents['worksheets'][0]['cells'][0]
        except KeyError:
            first_cell = contents['cells'][0]

            first_cell_type = first_cell['cell_type']
            if first_cell_type not in 'markdown raw'.split():
                return description
            desc_lines = first_cell['source']
            for line in desc_lines:
                suffix = ''
                if TITLE_PATTERN.match(line):
                    line = TITLE_PATTERN.sub('**', line)
                    suffix = '**'
                    if line.endswith('

'):
                        description += '{line}{suffix}
'.format(line=line[:-1], suffix=suffix)
                    else:
                        description += '{line}{suffix} '.format(line=line, suffix=suffix)
                return description
        if __name__ == '__main__':
            main()
```

Here’s how to set up and use this script:

1. Put the code above into a file called `make_readme.py` in a directory that contains Jupyter Notebook files.

2. Edit line 26 to point to the repo that your directory is in. If you are setting this up for a directory in your local clone of the `analysis-fred/` repository you should change line 26 from:

   ```python
   REPO = 'bitbucket.org/salishsea/tools/raw/tip'
   ```

   to:

   ```python
   REPO = 'bitbucket.org/salishsea/analysis-fred/raw/tip'
   ```

3. Edit line 27 to point to the directory containing this `make_readme.py` script and the notebooks that it will create links to. If the directory is `analysis-fred/notebooks/` you should change line 27 from:
REPO_DIR = 'SalishSeaNowcast/notebooks/figures/publish'

to:

REPO_DIR = 'notebooks'

4. Edit lines 34-35 to describe what your notebooks are about. You can put as much text as you want there. It is the beginning of the text that will appear between the list of files on the Bitbucket page and the list of links to the Notebook Viewer renderings of your notebooks.

5. Save the make_readme.py file. You won’t need to edit it again unless you want to change the preamble text starting at line 34.

6. Run the make_readme.py script to create your README.md file:

   $ python make_readme.py

7. Use Mercurial to add, commit, and push to Bitbucket your new notebook(s), the make_readme.py script, and the README.md file:

   $ hg add make_readme.py README.md MyNotebook.ipynb
   $ hg ci -m "Add new notebook, make_readme script and README file."
   $ hg push

8. Use your browser to navigate to the repo and directory on Bitbucket.org and you should see the rendered README.md and a link to the Notebook Viewer for your notebook(s).

9. Each time you create a new notebook in the directory, run python make_readme.py to update the README.md file and commit it along with your new notebook.

   The make_readme.py script reads the first cell of each notebook in the directory and, if that cell contains text, adds it to the README.md file. That lets you include a title and brief description of your notebooks along with the links on the Bitbucket page. If you change the contents of that 1st cell in an existing notebook you need to run python make_readme.py, commit the README.md changes, and push them to Bitbucket in order to update the page there.

2.2.7 Create a WestGrid Account

WestGrid is a computing resource shared by western Canadian universities. We use WestGrid for computationally intensive simulations. The large number of processors available often reduces simulation run-time considerably.

In order to use WestGrid you need an account. To create an account follow the steps on this page:

https://www.westgrid.ca/support/accounts/registering_ccdb

Note: When prompted to select an institution, choose WestGrid: University of British Columbia.

If you are creating an account as a sponsored user ask your supervisor for their CCRi code.

2.2.8 Documentation with Sphinx

We use Sphinx for most documentation in the Salish Sea MEOPAR project. Sphinx provides:

- direct rendering to HTML for online publication
- easy inclusion of LaTeX math syntax with in-browser rendering via MathJax
Salish Sea MEOPAR Documentation

- easy inclusion of figures, graphs, and images
- deep linkability
- optional PDF rendering

LaTeX should be used for manuscripts of publications, for which PDFs must be rendered, uploaded, and linked into other documentation to make them available online.

All documentation is under Version Control with Mercurial and stored in either the docs repo, or in the docs directory of another appropriate project repo (see Organization of Mercurial Repositories). Most notably, the tools repo includes a large public documentation tree. When changes that have been committed to the docs and tools repos are pushed to Bitbucket a signal is sent to readthedocs.org to automatically rebuild and render the docs at https://salishsea-meopar-docs.readthedocs.io/en/latest/ and https://salishsea-meopar-tools.readthedocs.io/en/latest/, respectively.

Sphinx uses reStructuredText (reST), a simple, unobtrusive markup language. The Sphinx documentation provides a brief introduction to reST concepts and syntax. Sphinx extends reST with a collection of directives and interpreted text roles for cross-referencing, tables of contents, code examples, and specially formatted paragraphs like notes, alerts, warnings, etc.

Installing Sphinx

Sphinx and the packages that it depends on are included in the Anaconda Python Distribution that you should already have installed.

Building and Previewing Documentation

As you are writing and editing Sphinx documentation you can build the HTML rendered docs locally and preview them in your browser to ensure that there are no reST syntax errors and that the docs look the way you want them to.

In the top level docs/ directory (e.g. docs/ in the docs repo, or tools/docs/ in the tools repo) use the command:

```
make html
```

to build the docs. You will be notified of any syntax or consistency errors. The HTML pages produced by the make html command are stored in the _build/html/ subdirectory and you can use your browser to open the index.html file in that directory to preview them. You can keep a browser tab open to the rendered docs and refresh after each build to see updates.

Note: The top level docs/ directory contains (at minimum) the files conf.py, Makefile, and index.rst, and the directory _static/. After the docs have been built it will also contain the _build/ directory.

The result of running make html should look something like:

```
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.1.3
loading pickled environment... done
building [html]: targets for 9 source files that are out of date
updating environment: 0 added, 0 changed, 0 removed
looking for now-outdated files... none found
preparing documents... done
writing output... [100%] sphinx_docs
writing additional files... search
copying static files... done
```

(continues on next page)

2.2. Working Environment 23
Writing Style

Consider using semantic line breaks in your Sphinx files.

Links and Cross-references

External Links

The preferred way to including external links is via markup like:

```
This is a paragraph that contains `a link`_.

.. _a link: http://example.com/
```

If the link text should be the web address, you don’t need special markup at all, the parser finds links and mail addresses in ordinary text.

Internal Links

To support cross-referencing to arbitrary locations in any document, the standard reST labels are used. For this to work label names must be unique throughout the entire documentation. There are two ways in which you can refer to labels:

- If you place a label directly before a section title, you can reference to it with :ref:`label-name`. Example:

  ```
  .. _my-reference-label:
  
  Section to cross-reference
  --------------------------
  
  This is the text of the section.
  
  It refers to the section itself, see :ref:`my-reference-label`.
  ```

  The :ref: role would then generate a link to the section, with the link title being “Section to cross-reference”. This works just as well when section and reference are in different source files.

  Automatic labels also work with figures: given:

  ```
  .. _my-figure:
  
  .. figure:: whatever
  
  Figure caption
  ```
a reference :ref:`my-figure` would insert a reference to the figure with link text “Figure caption”.

The same works for tables that are given an explicit caption using the `table` directive.

- Labels that aren’t placed before a section title can still be referenced to, but you must give the link an explicit title, using this syntax: :ref:`Link title <label-name>`.
- Inter-Sphinx links are enabled between the `tools` docs and the `docs` repo; i.e. whenever Sphinx encounters a cross-reference that has no matching target in the `tools` docs, it looks for targets in the `docs`, and vice-versa.

Using `ref` is advised over standard reStructuredText links to sections (like `Section title`) because it works across files, when section headings are changed, and for all builders that support cross-references.

### Links to Rendered Jupyter Notebooks

To link to a rendered representation of an Jupyter Notebook that has been pushed to a Bitbucket repo use markup like:

```
.. _SalishSeaBathy.ipynb: https://nbviewer.jupyter.org/urls/bitbucket.org/salishsea/_tools/raw/tip/bathymetry/SalishSeaBathy.ipynb
```

### Forcing Line Breaks

In most cases you should just let Sphinx take care of inserting line breaks in the rendered docs; it will almost always do the right thing by putting breaks between paragraphs, between list items, around block quotations and code examples, etc.

Occasionally though you may need to force line breaks. The most common case for this is to add line breaks within table cells so as to avoid excessive sideways scrolling of the rendered table. You can force a line break in the HTML that Sphinx renders by defining a substitution that will insert a break tag (<br>). Here’s an example of doing that and using the substitution in a table cell:

```
|br| raw:: html

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
<th>New Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>27-Oct-2014</td>
<td>1st :file:<code>nowcast/</code> run results</td>
<td>N/A</td>
</tr>
<tr>
<td>20-Nov-2014</td>
<td>1st :file:<code>forecast/</code> run results</td>
<td>N/A</td>
</tr>
<tr>
<td>26-Nov-2014</td>
<td>Changed to tidal forcing tuned for better</td>
<td>br</td>
</tr>
<tr>
<td></td>
<td>accuracy at Point Atkinson</td>
<td></td>
</tr>
</tbody>
</table>
```

**Note:** The `|br|` substitution needs to be defined once (but only once) per file.
2.2.9 ssh Configuration

This section assumes that you have completed the `ssh-keygen` step of the Bitbucket ssh Set-up instructions referenced in the *Version Control with Mercurial* section.

If you set up working environments to run the Salish Sea NEMO model on salish or on a Westgrid cluster, or need to access model results files from a Westgrid cluster, you should set up `ssh-agent` forwarding to minimize the need to repeatedly type your `ssh` key pass phrase and to minimize the number of machines on which your private key is stored.

ssh keys work by having a public key and a private key pair with the public key on remote machines and the private key on the local machine you log into first. Your private key is usually protected by a long passphrase that you only have to enter once per login session on the local machine. After that an ssh-agent program on the local machine uses the private key to exchange encrypted authentication information with the remote machines.

The public key needs to be stored on every machine that you want to ssh into. Because your home directory is shared across all of the Waterhole machines (and salish, and skookum) you only need to put your public key on any one of those machines. On WestGrid and ComputeCanada you need to put your public key on each machine that you use (cedar, graham, orcinus, etc.). The sections below include instructions for how to store your public key on various machines.

**ssh-agent Forwarding for salish**

To set up agent forwarding for salish create a `$HOME/.ssh/config` file on your Waterhole machine containing the following (or append the following if `$HOME/.ssh/config` already exists):

```
Host salish
  Hostname salish.eos.ubc.ca
  User userid
  ForwardAgent yes
```

where `userid` is your EOAS user id.

The first two lines establish `salish` as a short alias for `salish.eos.ubc.ca` so that you can just type `ssh salish`.

The third line sets the user id to use on the remote system, which is convenient if it differs from your EOAS user id.

The last line enables agent forwarding so that authentication requests received on the remote system are passed back to your Waterhole machine for handling. That means that connections to Bitbucket (for instance) in your session on `salish` will be authenticated by your Waterhole machine. So, after you type your `ssh` key pass phrase in to your Waterhole machine once, you should not have to type it again until you log off and log in again.

The other thing that is required for agent forwarding to work is that your `ssh` public key be stored in the `$HOME/.ssh/authorized_keys` file on the remote system. Thanks to shared storage between the Waterhole machines and salish that is really easy to do:

```bash
cd $HOME/.ssh
cat id_rsa.pub >> authorized_keys
```

**ssh-agent Forwarding for Westgrid and ComputeCanada Clusters**

To set up agent forwarding for a Westgrid or ComputeCanada machine append the appropriate block below to the `$HOME/.ssh/config` file on your Waterhole machine:
where userid is your Westgrid or ComputeCanada user id.

Note: If you do not have a Westgrid or ComputeCanada account follow the instructions here to make one: Create a WestGrid Account.

Install your ssh public key on the remote machine; cedar, for example:

```
ssh-copy-id -i $HOME/.ssh/id_rsa cedar
```

You will be prompted for your Westgrid or ComputeCanada password. After the key has been installed you should be able to use ssh, scp, and sftp to connect to the remote machine without having to type your password. Likewise, Mercurial commands on the remove machine should not require you to type your ssh key pass phrase.

### 2.2.10 emacs Configuration

If you use emacs as your editor, you may want to amend your configuration with some or all of the snippets below to add syntax highlighting for the various file types commonly used in the Salish Sea MEOPAR project, and to improve your emacs editing experience in other ways.

The snippets go in your $HOME/.emacs file and some of them assume that you also have a $HOME/elisp/ directory to contain downloaded mode files, etc. If you don’t already have one or both of those, go ahead and create them with:

```
touch $HOME/.emacs
mkdir $HOME/elisp
```

**.emacs Snippets**

You can use as many or as few of these as you wish by adding them to your $HOME/.emacs file. You will need to restart emacs for the changes to take effect.

Enable syntax highlighting whenever possible:

```lisp
;; enable syntax highlighting
(global-font-lock-mode 1)
```

Disable display of the splash screen when emacs starts:
Salish Sea MEOPAR Documentation

;;; don't display the splash screen
(setq inhibit-startup-message t)

Display line and column numbers in the mode line:

;;; show line and column numbers in mode line
(line-number-mode 1)
(column-number-mode 1)

Force the use of spaces instead of tab characters:

;;; force emacs to always use spaces instead of tab characters
(setq-default indent-tabs-mode nil)

Set the default tab width to 4 spaces. (Many modes override this setting.):

;;; set default tab width to 4 spaces
(setq default-tab-width 4)
(setq tab-width 4)

Show trailing whitespace characters in red. Trailing whitespace (i.e. spaces, tabs, etc. at the ends of lines or empty lines at the ends of files) is a terrible thing that should be eliminated with extreme prejudice. This setting makes it stick out like a sore thumb. The M-x delete-trailing-whitespace command deletes all trailing whitespace characters from a buffer. Please commit whitespace deletions separately from other file modifications to make reviewing and merging easier.

;;; default to showing trailing whitespace
(setq-default show-trailing-whitespace t)

Automatically use f90-mode for files whose names end with .F90, .h90, or that start with namelist. These patterns are applied in addition to the file name patterns that normally trigger f90-mode.

;;; use f90-mode for .F90, .h90, and namelist files
(add-to-list 'auto-mode-alist '(*\.\.F90\" . f90-mode))
(add-to-list 'auto-mode-alist '(*\.\.h90\" . f90-mode))
(add-to-list 'auto-mode-alist '(*namelist* . f90-mode))

Additional Editing Modes

These instructions are for installing an configuring editing modes that are not part of the *emacs* distribution. Modes provide syntax highlighting, command shortcuts, and other features to help you edit files of various types.

To use 3rd party modes that are in your $HOME/elisp/ directory you need to add the following to your $HOME/.emacs file:

;;; add my personal elisp repository to the load-path
(add-to-list 'load-path "~/elisp")

The general procedure to install a 3rd party mode is:

- Download the EmacsLisp file (file extension .el) into your $HOME/elisp/ directory
- Add the appropriate configuration statements listed below to your $HOME/.emacs file
- Restart *emacs* for the changes to take effect

You can view the help for a mode when it is active with the *emacs* command C-h m.
Salish Sea MEOPAR Documentation

YAML Mode
This is useful for working on Salish Sea NEMO run description files.
Download yaml-mode.el from https://raw.github.com/yoshiki/yaml-mode/master/yaml-mode.el into your
$HOME/elisp/ directory:
cd $HOME/elisp/
wget https://raw.github.com/yoshiki/yaml-mode/master/yaml-mode.el

Configure emacs to use yaml-mode automatically whenever you visit a file with the extension .yaml or .yml:
;; YAML mode
;; https://raw.github.com/yoshiki/yaml-mode/master/yaml-mode.el
(require 'yaml-mode)
(add-to-list 'auto-mode-alist '("\\.yaml$" . yaml-mode))
(add-to-list 'auto-mode-alist '("\\.yml$" . yaml-mode))

ReStructuredText Mode
This is useful for working on the Salish Sea MEOPAR project documentation files.
Download rst.el from http://docutils.sourceforge.net/tools/editors/emacs/rst.el into your $HOME/elisp/ directory:
cd $HOME/elisp/
wget http://docutils.sourceforge.net/tools/editors/emacs/rst.el

Configure emacs to use rst-mode automatically whenever you visit a file with the extension .rst:
;; add reStructuredText mode
;; http://docutils.sourceforge.net/tools/editors/emacs/rst.el
(require 'rst)
(setq auto-mode-alist
(append '(("\\.rst$" . rst-mode)) auto-mode-alist))

Python Mode
emacs includes a Python editing mode, but these instructions are for installing a more comprehensive python-mode
that is maintained by the Python community.
Download python-mode.el from https://gitlab.com/python-mode-devs/python-mode/raw/master/python-mode.el
and move it into your $HOME/elisp/ directory.
Configure emacs to use python-mode automatically whenever you visit a file with the extension .py:
;; add better Python mode
;; https://gitlab.com/python-mode-devs/python-mode/raw/master/python-mode.el
(setq auto-mode-alist
(cons '("\\.py$" . python-mode)
auto-mode-alist))
(setq interpeter-mode-alist
(cons '("python" . python-mode)
interpreter-mode-alist))
(autoload 'python-mode "python-mode" "Python editing mode." t)

2.2. Working Environment

29


Python Source Code Checking via Flake8

These settings enable on-the-fly static analysis of Python code using the flake8 tool. flake8 uses a collection of Python packages to check Python code for syntax error and adherence to the PEP8 coding style guide. The snippet below configures emacs flymake mode to run flake8 in the background whenever you pause in your typing for a few seconds, and whenever a Python file is loaded or saved. Lines that have messages from flake8 associated with them are colourfully highlighted and the messages should appear in the minibuffer region at the bottom of the screen when you hover your cursor over the highlighting. The flymake-cursor mode displays messages from flake8 for the line that the cursor is on in the minibuffer - useful for ssh sessions when mouse integration with emacs is not available.

Install flake8 into your local Python environment:

```
$ pip install --user flake8
```

and, if you have not already done so, add $HOME/.local/bin to your PATH (see .bashrc Snippets). The additions to your PATH should be such that $HOME/.local/bin comes after $HOME/anaconda/bin.

Download flymake-cursor.el from http://www.emacswiki.org/emacs/download/flymake-cursor.el into your $HOME/elisp/ directory:

```
cd $HOME/elisp/
wget http://www.emacswiki.org/emacs/download/flymake-cursor.el
```

Configure emacs to run flake8 via flymake on Python files, and to enable flymake-cursor:

```
;;; connect flymake for Python buffers to Flake8
(defun flymake-flake8-init ()
  ;; temp-file (flymake-init-create-temp-buffer-copy
  ;;   'flymake-create-temp-inplace))
  ;; local-file (file-relative-name
  ;;   temp-file
  ;;   (file-name-directory buffer-file-name)))
  (list "flake8" (list local-file)))
(add-to-list 'flymake-allowed-file-name-masks
  "\\.py\\" flymake-flake8-init))
(add-hook 'find-file-hook 'flymake-find-file-hook)
;;; display flymake messages for cursor line in minibuffer
(require 'flymake-cursor)
```

### Code Folding for Python

Allows you to collapse and expand (aka fold) function definitions in Python code by hooking into emacs outline-mode.

Defaults to collapsing all definitions when a Python file is loaded. See the comments at the top of python-outline.el for how to change that default. C-q toggles folding of the function the cursor is in. C-c C-a toggles folding of all functions.

Copy python-outline.el from /home/dlatorne/elisp/ into your $HOME/elisp/ directory:

```
cd $HOME/elisp/
cp /home/dlatorne/elisp/python-outline.el
```

Configure emacs to enable python-outline mode:
Mercurial Mode

Allows you to work with Mercurial from within **emacs**.

Download mercurial.el from [http://hg.intevation.org/mercurial/file/tip/contrib/mercurial.el](http://hg.intevation.org/mercurial/file/tip/contrib/mercurial.el) into your $HOME/elisp/ directory:

```bash
cd $HOME/elisp/
wget http://hg.intevation.org/mercurial/file/tip/contrib/mercurial.el
```

Configure **emacs** to always load Mercurial mode:

```lisp
;; always load mercurial support
;; http://hg.intevation.org/mercurial/file/tip/contrib/mercurial.el
(load-file "~/elisp/mercurial.elc")
```

### 2.2.11 Building a Python 3 Conda Environment

**Note:** If you followed the Anaconda Python Distribution instructions to install a Python 3 environment, you can skip this section. It is about creating a Python 3 environment when your default environment is Python 2.

In the first stages of our move to Python 3, some important tools we use will still be in Python 2. To facilitate the change, you may find it useful to have Python 3 in a separate environment. If you are doing EOSC 511 in 2015 or later you will need a Python 3 environment. This section explains how to set one up assuming you already have conda and anaconda installed.

At the command line create your new environment, here called “mypython3” but you can call it what you wish:

```
conda create -n mypython3 python=3
```

To activate the environment:

```
source activate mypython3
```

To get all the goodies (e.g. Jupyter Notebook, NumPy, matplotlib) you can install Anaconda, which will auto-magically use Python 3.

```
conda install anaconda
```

and then our additional install, netcdf4-python

```
conda install netcdf4
```

Now you have set-up your Python 3 environment. To start it in a new terminal

```
source activate mypython3
```

When you are using that environment your prompt will change and will include (mypython3). To return to using your “base” environment you can de-activate the conda environment with:
2.2.12 Porting Code to Python 3

While there are many differences between Python 2 and Python 3 few of them impact most of the code that we write in the Salish Sea project. This section describes the types of changes that had to be made in order to convert (also known as “port”) the SalishSeaTools Package to Python 3.

If you encounter other changes that you need to make to port our code to Python 3, please feel free to add them below. If you are interested in the details of the differences between Python 2 and Python 3 they can be found in the What's New in Python documentation.

Part of the move to Python 3 was a reorganization of the standard library. That means that some import need to be changed when code is ported from Python 2 to Python 3. Specific instances of that (like the StringIO module) are described below. The description of all of the standard library changes is contained in PEP 3108.

Porting the SalishSeaTools Package

This section describes the types of changes that had to be made to port the SalishSeaTools Package (including the nowcast codebase) from Python 2.7 to Python 3.5 in October 2015.

Mixed TABs and Spaces for Indentation

While mixing TABs and spaces for indentation in a Python module was never a good idea, it causes a TabError exception to be raised when such a module is imported in Python 3.

All Python code should use spaces for indentation and the indentation levels should be 4 spaces.

Change print Statements to print() Functions

print was a statement in Python 2. It is a function in Python 3. So, code like:

```
print 'red is clockwise'
```

has to be changed to:

```
print('red is clockwise')
```

Change cStringIO.StringIO Imports to io.StringIO

In Python 2 StringIO class in the standard library has two implementations, one in Python, and a faster one in C. The former was imported like:

```
from StringIO import StringIO
```

and the latter like:

```
from cStringIO import StringIO
```
In Python 3 the `StringIO` class has been moved to the `io` module and the interpreter takes care of first trying to import the faster C version or falling back to the Python version if necessary. So, those imports need to be changes to:

```python
from io import StringIO
```

The mock library that was developed as a separate, stand-alone library for Python 2 is included in the standard library in Python 3. So, instead from it like:

```python
from mock import (Mock, patch,
)
```

the Python 3 import looks like:

```python
from unittest.mock import (Mock, patch,
)
```

Also, because `mock` is now part of the standard library, it no longer needs to be installed separately or included in `setup.py` or environment descriptions files (`requirements.txt`, `requirements.pip`, `environment.yaml`, etc.).

It is also possible to set up a working environment for most things other than running the NEMO model on your own laptop, if you wish. The instructions and recommendations above are most applicable to doing that on a laptop that runs OS/X or Linux. Our best advice for Windows is to install puTTY and use it to connect remotely (via `ssh`) to your Linux environments (though additions to these docs by experienced Windows users who set up a working environment similar to the one described above are most welcome).

## 2.3 Salish Sea NEMO

This section documents the NEMO code and supporting files used to model the Salish Sea.

### 2.3.1 Quick Start Guide

The sections below describe very briefly the steps to set up and run the Salish Sea NEMO code on several compute platforms.

**Working on salish: NEMO v3.6**

This section describes very briefly the steps to set up and run the NEMO version 3.6 code on out group’s development machine, salish. To set up NEMO version 3.4, go [here](#). Details of what this all means and why the steps below are what they are can be found in subsequent sections.
Create a Workspace

You can call your workspace directory whatever you want but for consistency across systems we’re going to call it MEOPAR.

salish has a large local storage partition mounted at /data/ which is where we will put the code and run configuration file repos:

```
mkdir -p /data/$USER/MEOPAR
```

Create directories for the temporary directories that are created to run the model, and to receive results files from runs. We’ll call them SalishSea/ and SalishSea/results/:

```
mkdir -p /data/$USER/MEOPAR/SalishSea/results
```

The ocean.eos.ubc.ca storage storage server space that you use on your Waterhole workstation is mounted at /ocean/ on salish so you should be able to see your Salish Sea Repos and Packages at /ocean/$USER/MEOPAR/.

You may want to open an EOAS help desk ticket requesting that the salish/data/ partition be mounted on your Waterhole workstation so that you can easily view and copy files when you are not logged into salish without having to use scp or sftp.

Clone the Repos

Assuming that you are using SSH key authentication on Bitbucket (see Version Control with Mercurial), clone the NEMO-3.6-code, XIOS-2 (XML I/O Server), NEMO-forcing, and SS-run-sets repos into your workspace on /data/:

```
cd /data/$USER/MEOPAR/
hg clone ssh://hg@bitbucket.org/salishsea/nemo-3.6-code NEMO-3.6-code
hg clone ssh://hg@bitbucket.org/salishsea/xios-2 XIOS-2
hg clone ssh://hg@bitbucket.org/salishsea/xios-arch XIOS-ARCH
hg clone ssh://hg@bitbucket.org/salishsea/ss-run-sets SS-run-sets
hg clone ssh://hg@bitbucket.org/salishsea/grid
hg clone ssh://hg@bitbucket.org/salishsea/rivers-climatology
hg clone ssh://hg@bitbucket.org/salishsea/tides
hg clone ssh://hg@bitbucket.org/salishsea/tracers
```

Compile XIOS-2

Please see the XIOS-2 section of the UBC EOAS MOAD Group Documentation.

Compile NEMO-3.6

Compile the SalishSea NEMO configuration and link it to XIOS-2 using the salish architecture definitions, distributing the compilation over 8 cores. The NEMO ARCH files use the XIOS_HOME environment variable to find the XIOS-2 library you built above. XIOS_HOME must be an absolute path to your XIOS-2 clone directory. You can set XIOS_HOME on the command-line before the makenemo and maketools commands as shown below, or you can set and export the value of XIOS_HOME in your $HOME/.bashrc file.

```
cd NEMO-3.6-code/NEMOGCM/CONFIG
XILOS_HOME=/data/$USER/MEOPAR/XIOS-2/ ./makenemo -n SalishSea -m GCC_SALISH -j8
```
The resulting executable is located in NEMO-3.6-code/NEMOGCM/CONFIG/SalishSea/BLD/bin/.

Compile and link the `rebuild_nemo` tool:

```bash
cd NEMO-3.6-code/NEMOGCM/TOOLS
XIOS_HOME=/data/$USER/MEOPAR/XIOS-2/ ./maketools -m GCC_SALISH -n REBUILD_NEMO
```

See `REBUILD_NEMO` for more information about it.

**Prepare and Execute Runs**

The SS-run-sets `SalishSea Directory` contains a subdirectory called `v201702/` that contains a version controlled sample run description file and namelist segment files. Create your own directory in that repo where you can copy, edit, and version control those files to define the run that you want to execute.

The run description file is described in the Run Description File Structure section of the project tools documentation. The namelists are described in the NEMO-3.6 Book.

Use `salishsea run` Sub-command to prepare, execute, and gather the results for a run:

```bash
salishsea run SalishSea.yaml /data/$USER/MEOPAR/SalishSea/results/my_excellent_results
```

`salishsea run` returns the path and name of the temporary run directory, and the job identifier assigned by the queue manager, something like:

```bash
salishsea_cmd.prepare INFO: Created run directory ../../SalishSea/38e87e0c-472d-11e3-˓
→ 9c8e-0025909a8461
salishsea_cmd.run INFO: 57.master
```

`salishsea run` has a number of command-line option flags that are useful for controlling details of how runs are executed, and for debugging your YAML files and the symlinks in the temporary run directory. Please see `salishsea help run` or the SalishSeaCmd package docs.

You can use the `qstat` command to monitor the execution status of your job.

A convenient command to monitor the memory use of a run and its time step progress is:

```bash
watch -n 5 "(free -m; cat time.step)"
```

When the job completes the results should have been gathered in the directory you specified in the `salishsea run` command and the temporary run directory should have been deleted.

You should receive and email something like:

```plaintext
Date: Thu, 2 Feb 2017 15:51:05 -0800
From: <admin@salish.eos.ubc.ca>
To: <dlatornell@eoas.ubc.ca>
Subject: PBS JOB 3926.master

PBS Job Id: 3926.master
Job Name: test-cmd
Exec host: master/0
Begun execution
```

when your run starts execution (usually immediately).

When the run finishes you should receive an email something like:
You may also receive a email when the run finishes that talks about:

Unable to copy file /var/spool/torque/spool/...

Please see the Getting stdout and stderr into Your Results Directory section for instructions on how to resolve that issue.

Look at the Results

A number of notebooks that look at NetCDF files are available in tools/analysis_tools/. To start these, go to the top level directory of the analysis repo on your local machine (not on salish) and type:

```
jupyter notebook
```

At this point lots of information will appear in your terminal. This terminal session is now running a server and cannot be used for anything else until you are finished with the notebooks. At that point you need to CTRL-C to get out.

At the same time a window should open in your browser. If it doesn’t, look at your terminal, find the ip address, something like:

```
The Jupyter Notebook is running at: http://127.0.0.1:8888/
```

and put that UTL into your browser. From this initial window you can open the notebooks in analysis_tools directory and look around. The links to the various files will probably not work. Change them to point to your file space. You will probably want to build your own notebook but these notebooks give you lots of examples to copy from.

Profiling with the GNU Profiler

The GNU profiler allows you to find out which parts of the code are taking the longest to run.

1. Compile the code with the -pg flag.

This requires adding -pg to the two lines in your arch file that start with %FCFLAGS and %LDFLAGS (as in the following excerpt from NEMO-3.6-code/NEMOGCM/ARCH/UBC_EOAS/arch-GCC_SALISH_ocean_gprof.fcm):

```
%XIOS_HOME /ocean/$USER/MEOPAR/XIOS-2
```

(continues on next page)
Using the modified arch file, compile your NEMO configuration, e.g.:

```bash
XIOS_HOME=/data/$USER/MEOPAR/XIOS-2/ ./makenemo -n SalishSea -m GCC_SALISH_ocean_gprof
```

2. Submit the model run from your prepared run directory with the **no-submit** option

```bash
salishsea run --no-submit example.yaml /path/to/results
```

and then, from the temporary run directory, run

```bash
mpirun -n 7 ./nemo.exe : -n 1 ./xios_server.exe > stdout 2>stderr &
```

A file called gmon.out will be created in your run directory.

3. In the temporary run directory, see the **gprof** command with the executable name and gmon.out as input to create a readable summary of the timing output. Redirect the output to a text file to save it; you can then view this file using **less**.

```bash
gprof nemo.exe gmon.out > gprof_out.txt
gprof_out.txt
```

For more information, see [https://sourceware.org/binutils/docs/gprof/Call-Graph.html#Call-Graph](https://sourceware.org/binutils/docs/gprof/Call-Graph.html#Call-Graph)

## Getting stdout and stderr into Your Results Directory

If you receive email messages like:

```plaintext
Date: Thu, 2 Feb 2017 15:53:55 -0800
From: <adm@salish.eos.ubc.ca>
To: <dlatournell@eoas.ubc.ca>
Subject: PBS JOB 3926.master

PBS Job Id: 3926.master
Job Name: test-cmd
Exec host: master/0
```

(continues on next page)
An error has occurred processing your job, see below.
Post job file processing error; job 3926.master on host master/0

Unable to copy file /var/spool/torque/spool/3926.master.OU to dlatorne@salish.eos.ubc.ca:/data/dlatorne/MEOPAR/test-cmd/test-fspath2/stdout

*** error from copy
Permission denied (publickey,password).
lost connection
*** end error output
Output retained on that host in: /var/spool/torque/undelivered/3926.master.OU

Unable to copy file /var/spool/torque/spool/3926.master.ER to dlatorne@salish.eos.ubc.ca:/data/dlatorne/MEOPAR/test-cmd/test-fspath2/stderr

*** error from copy
Permission denied (publickey,password).
lost connection
*** end error output
Output retained on that host in: /var/spool/torque/undelivered/3926.master.ER

when your runs on salish finish, the system is telling you that it can’t copy the master.OU (stdout) and master.ER (stderr) files from your run to your results directory. You can manually retrieve them from the paths given in the email.

To resolve the copy error and get the files to be renamed to stdout and stderr in your results directory you need to set up an ssh key pair without a passphrase, configure ssh to be able to use them, and make the key pair trusted on salish. The steps to do that are:

1. Create a passphrase-less ssh key pair:

```bash
cd $HOME/.ssh
ssh-keygen -C"salish-torque" -f $HOME/.ssh/salish_torque_id_rsa
```

Just hit `Enter` twice when you are prompted to enter and confirm a passphrase:

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

2. Configure ssh to use the key pair with the user and hostname that the system uses to copy files from the torque spool to your results directory by adding a block like the following to your $HOME/.ssh/config file:

```bash
Host salish.eos.ubc.ca
  Hostname salish.eos.ubc.ca
  User dlatorne
  IdentityFile ~/.ssh/salish_torque_id_rsa
```

replacing dlatorne with your user id.

3. Make the key pair trusted on salish by appending the public key to your $HOME/.ssh/authorized_keys file:

```bash
cat $HOME/.ssh/salish_torque_id_rsa.pub >> $HOME/.ssh/authorized_keys
```
Working on westgrid: NEMO v3.6

This section describes the steps to set up and run the Salish Sea NEMO version 3.6 code on the WestGrid machine orcinus.westgrid.ca.

This guide assumes that your Working Environment is set up and that you are familiar with Working on salish: NEMO v3.6.

Modules setup

Similar to when working on ComputeCanada clusters the module load command must be used to load extra software components on orcinus.

You can manually load the modules each time you log in, or you can add the lines to your .bashrc file so that they are automatically loaded upon login.

The modules are:

```
module load python
module load intel/14.0.2
module load intel/14.0/ncdf-4.3.3.1_mpi
module load intel/14.0/ncdf-fortran-4.4.0_mpi
module load intel/14.0/hdf5-1.8.15p1_mpi
module load intel/14.0/nco-4.5.2
```

PATH Additions

Lastly, you need to modify your search path such that your shell can find python scripts installed with pip --user. Change the $PATH line in the modify search path section of .bash_profile to include $HOME/.local/bin and $HOME/bin in your search path:

```
$PATH=$HOME/.local/bin:$HOME/bin
```

Create a Workspace and Clone the Repositories

```
mkdir -p $HOME/MEOPAR/SalishSea/results
```

Clone the repos needed to run the model:

```
cd $HOME/MEOPAR
hg clone ssh://hg@bitbucket.org/salishsea/nemo-3.6-code NEMO-3.6-code
hg clone ssh://hg@bitbucket.org/salishsea/xios-2 XIOS-2
hg clone ssh://hg@bitbucket.org/salishsea/xios-arch XIOS-ARCH
hg clone ssh://hg@bitbucket.org/salishsea/ss-run-sets SS-run-sets
hg clone ssh://hg@bitbucket.org/salishsea/grid
hg clone ssh://hg@bitbucket.org/salishsea/rivers-climatology
hg clone ssh://hg@bitbucket.org/salishsea/tides
hg clone ssh://hg@bitbucket.org/salishsea/tracers
hg clone ssh://hg@bitbucket.org/salishsea/nemo-cmd NEMO-Cmd
hg clone ssh://hg@bitbucket.org/salishsea/salishseacmd SalishSeaCmd
```

There is no need to clone the docs or analysis repos at WestGrid.
Install Tools and Command Processor Packages

Install the SalishSeaTools Package and Salish Sea NEMO Command Processor Python packages:

```bash
mkdir -p $HOME/.local
cd $HOME/MEOPAR/
pip install --user --editable NEMO-Cmd
pip install --user --editable SalishSeaCmd
```

Compile XIOS-2

Please see the XIOS-2 section of the UBC EOAS MOAD Group Documentation.

Compile NEMO-3.6

Compile the SalishSea NEMO configuration and link it to XIOS-2, and compile the `rebuild_nemo` tool. The NEMO ARCH files use the `XIOS_HOME` environment variable to find the XIOS-2 library you built above. `XIOS_HOME` must be an absolute path to your XIOS-2 clone directory. You can set `XIOS_HOME` on the command-line before the `makenemo` and `maketools` commands as shown below, or you can set and export the value of `XIOS_HOME` in your `$HOME/.bashrc` file.

```bash
cd $HOME/MEOPAR/NEMO-3.6-code/NEMOGCM/CONFIG
XIOS_HOME=$HOME/MEOPAR/XIOS-2/ ./makenemo -n SalishSea -m X64_ORCINUS -j 8
cd $HOME/MEOPAR/NEMO-3.6-code/NEMOGCM/TOOLS
XIOS_HOME=$HOME/MEOPAR/XIOS-2/ ./maketools -n REBUILD_NEMO -m X64_ORCINUS
```

To build a configuration other than SalishSea, replace `SalishSea` with the name of the configuration to be built, e.g. SMELT:

```bash
cd $HOME/MEOPAR/NEMO-3.6-code/NEMOGCM/CONFIG
XIOS_HOME=$HOME/MEOPAR/XIOS-2/ ./makenemo -n SMELT -m X64_ORCINUS -j 8
```

Prepare and Execute Runs

The SS-run-sets/v201702/ directory in the SalishSea Directory repo contains version controlled sample run description files and namelist segment files. In your own directory in that repo copy, edit, and version control those files to define the runs that you want to execute.

The run description file is described in the Run Description File Structure section of the project tools documentation. The namelists are described in the NEMO-3.6 Book.

Use `salishsea` run Sub-command to prepare, execute, and gather the results for a run:

```bash
salishsea run SalishSea.yaml $HOME/MEOPAR/SalishSea/results/my_excellent_results
```

`salishsea run` returns the path and name of the temporary run directory, and the job identifier assigned by the queue manager, something like:

```bash
salishsea_cmd.prepare INFO: Created run directory /home/dlatorne/MEOPAR/SalishSea/a90d391c-0e1e-11e4-aa4e-6431504ad786
salishsea_cmd.run INFO: 3544250.orca2.ibb
```
Salish Sea MEOPAR Documentation

salishsea run has a number of command-line option flags that are useful for controlling details of how runs are executed, and for debugging your YAML files and the symlinks in the temporary run directory. Please see salishsea help run or the SalishSeaCmd package docs.

You can use the job identifier with qstat, showstart, and checkjob to monitor the execution status of your job. When the job completes the results should have been gathered in the directory you specified in the salishsea run command and the temporary run directory should have been deleted.

To view and analyze the run results copy them to your EOAS /data/$USER/results/ workspace with scp, sftp or rsync.

Working on computecanada: NEMO v3.6

This section describes the steps to set up and run the Salish Sea NEMO version 3.6 code on the ComputeCanada machines graham.computecanada.ca and cedar.computecanada.ca.

This guide assumes that your Working Environment is set up and that you are familiar with Working on salish : NEMO v3.6.

Modules setup

When working on the ComputeCanada clusters, the module load command must be used to load extra software components.

You can manually load the modules each time you log in, or you can add the lines to your $HOME/.bashrc file so that they are automatically loaded upon login.

At present, beluga, cedar, and graham are configured similarly. The modules needed are:

```bash
module load netcdf-fortran-mpi/4.4.4
module load perl/5.22.4
module load python/3.7.0
```

Create a Workspace and Clone the Repositories

cedar and graham provide several different types of file storage. We use project space for our working environments because it is large, high performance, and backed up. Scratch space is even larger, also high performance, but not backed up, so we use that as the space to execute NEMO runs in, but generally move the run results to project space.

Both systems provide environment variables that are more convenient that remembering full paths to access your project and scratch spaces:

- Your project space is at $PROJECT/$USER/
- Your scratch space is at $SCRATCH/
- Daily atmospheric, river, and west boundary forcing files are in the $PROJECT/SalishSea/forcing/ tree

Create a MEOPAR/ directory tree in your project space:

```bash
mkdir -p $PROJECT/$USER/MEOPAR/SalishSea/results
```

Clone the repos needed to run the model:
Install the Command Processor Packages

Install the NEMO Command Processor and Salish Sea NEMO Command Processor Python packages:

```
    cd $PROJECT/$USER/MEOPAR/
    python3.7 -m pip install --user --editable NEMO-Cmd
    python3.7 -m pip install --user --editable SalishSeaCmd
```

Compile XIOS-2

Please see the XIOS-2 section of the UBC EOAS MOAD Group Documentation.

Compile NEMO-3.6

Compile the SalishSeaCast NEMO configuration and link it to XIOS-2, and compile the `rebuild_nemo` tool. The NEMO ARCH files use the `XIOS_HOME` environment variable to find the XIOS-2 library you built above. `XIOS_HOME` must be an absolute path to your XIOS-2 clone directory. You can set `XIOS_HOME` on the command-line before the `makenemo` and `maketools` commands as shown below, or you can set and export the value of `XIOS_HOME` in your `$HOME/.bashrc` file.

**beluga**

```
    cd $PROJECT/$USER/MEOPAR/NEMO-3.6-code/NEMOGCM/CONFIG
    XIOS_HOME=$PROJECT/$USER/MEOPAR/XIOS-2/ ./makenemo -n SalishSeaCast -m X64_BELUGA -j 8
    cd $PROJECT/$USER/MEOPAR/NEMO-3.6-code/NEMOGCM/TOOLS
    XIOS_HOME=$PROJECT/$USER/MEOPAR/XIOS-2/ ./maketools -n REBUILD_NEMO -m X64_BELUGA
```

**cedar**

```
    cd $PROJECT/$USER/MEOPAR/NEMO-3.6-code/NEMOGCM/CONFIG
    XIOS_HOME=$PROJECT/$USER/MEOPAR/XIOS-2/ ./makenemo -n SalishSeaCast -m X64_CEDAR -j 8
    cd $PROJECT/$USER/MEOPAR/NEMO-3.6-code/NEMOGCM/TOOLS
    XIOS_HOME=$PROJECT/$USER/MEOPAR/XIOS-2/ ./maketools -n REBUILD_NEMO -m X64_CEDAR
```
To build a configuration other than SalishSeaCast, replace SalishSeaCast with the name of the configuration to be built, e.g. SMELT:

```
cd $PROJECT/$USER/MEOPAR/NEMO-3.6-code/NEMOGCM/CONFIG
XIOS_HOME=$PROJECT/$USER/MEOPAR/XIOS-2/ ./makenemo -n SMELT -m X64_CEDAR -j 8
```

### Prepare and Execute Runs

The SS-run-sets/v201702/ directory in the SalishSea Directory repo contains version controlled sample run description files and namelist segment files. In your own directory in that repo copy, edit, and version control those files to define the runs that you want to execute.

The run description file is described in the Run Description File Structure section of the project tools documentation. The namelists are described in the NEMO-3.6 Book.

Please see the sections below for details of using forcing climatology and shared daily forcing files in your runs, and examples of run description and namelist file sections.

Use `salishsea run` Sub-command to prepare, execute, and gather the results for a run:

```
salishsea run SalishSea.yaml $PROJECT/$USER/MEOPAR/SalishSea/results/my_excellent_˓
˓→results
```

`salishsea run` returns the path and name of the temporary run directory, and the job identifier assigned by the queue manager, something like:

```
salishsea_cmd.run INFO: Created run directory /scratch/dlatorne/20mar17nowcast16x34_ ˓→2017-10-06T101548.694389-0700
salishsea_cmd.run INFO: Submitted batch job 1578481
```

`salishsea run` has a number of command-line option flags that are useful for controlling details of how runs are executed, and for debugging your YAML files and the symlinks in the temporary run directory. Please see `salishsea help run` or the SalishSeaCmd package docs.

You can use the batch job number with `squeue --job` and `sacct --job` to monitor the execution status of your job.

When the job completes the results should have been gathered in the directory you specified in the `salishsea run` command and the temporary run directory should have been deleted.

To view and analyze the run results copy them to your EOAS /data/$USER/results/ workspace with `scp`, `sftp` or `rsync`.

### Forcing Climatology and Daily Files

Model runs use a mixture of climatologies and daily forcing from other operational models or observations:

- Atmospheric forcing is almost always from the Environment and Climate Change Canada (ECCC) High Resolution Deterministic Prediction System (HRDPS) model hourly forecasts.
• Tides are, by definition, climatological.
• Most of the river run-offs are climatological, but daily average discharge and turbidity for the Fraser River may also be used.
• Tracers at the northern boundary in Johnstone Strait are climatological. At the western boundary at the mouth of the Juan de Fuca Strait we have hourly tracer fields from the University of Washington LiveOcean model since 4-Feb-2017 as well as climatologies.

Details of what this all means and why the steps in the sections linked from this page are what they are can be found in the following sections:

• NEMO-code
• Set-up, Initial Conditions, Forcing, etc. Files
• Salish Sea Run Sets Files

Running NEMO-3.4

The sections below briefly describe how to set up and run the NEMO v3.4 version of the Salish Sea NEMO code on various platforms.

Working on salish

This section describes very briefly the steps to set up and run the Salish Sea NEMO code. Details of what this all means and why the steps below are what they are can be found in subsequent sections.

Create a Workspace

You can call your workspace directory whatever you want but for consistency across systems we’re going to call it MEOPAR.

salish has a large local storage partition mounted at /data/ which is where we will put the code and run configuration file repos:

```
mkdir -p /data/$USER/MEOPAR
```

Create directories for the temporary directories that are created to run the model, and to receive results files from runs. We’ll call them SalishSea/ and SalishSea/results/:

```
mkdir -p /data/$USER/MEOPAR/SalishSea/results
```

The ocean.eos.ubc.ca storage server space that you use on your Waterhole workstation is mounted at /ocean/ on salish so you should be able to see your Salish Sea Repos and Packages at /ocean/$USER/MEOPAR/.

You may want to open an EOAS help desk ticket requesting that the salish /data/ partition be mounted on your Waterhole workstation so that you can easily view and copy files when you are not logged into salish without having to use scp or sftp.

Clone the Repos

Assuming that you are using SSH key authentication on Bitbucket (see Version Control with Mercurial), clone the NEMO-code, NEMO-forcing, and SS-run-sets repos into your workspace on /data/:
Compile the Code

Compile and link the full domain Salish Sea NEMO configuration and the IOM output server with the salish architecture definitions with the compilation distributed over 8 cores.

```bash
cd NEMO-code/NEMOGCM/CONFIG
./makenemo -n SalishSea -m salish -j8
```

The resulting executables are located in NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/bin/.

Compile and link the `rebuild_nemo` tool:

```bash
cd NEMO-code/NEMOGCM/TOOLS
./maketools -m salish -n REBUILD_NEMO
```

See REBUILD_NEMO for more information about it.

Prepare and Execute Runs

The SS-run-sets/ SalishSea/ Directory contains version controlled sample run description files and namelist segment files. Create your own directory in that repo where you can copy, edit, and version control those files to define the run that you want to execute.

The run description file is described in the Run Description File Structure section of the project tools documentation. The namelists are described in the NEMO-3.4 Book.

Use `salishsea run` Sub-command to prepare, execute, and gather the results for a run:

```bash
salishsea run --nemo3.4 SalishSea.yaml iodef.xml /data/$USER/MEOPAR/SalishSea/results/ →my_excellent_results
```

`salishsea run` returns the path and name of the temporary run directory, and the job identifier assigned by the queue manager, something like:

```
salishsea_cmd.prepare INFO: Created run directory ../../SalishSea/38e87e0c-472d-11e3- →9c8e-0025909a8461
salishsea_cmd.run INFO: 57.master
```

`salishsea run` has a number of command-line option flags that are useful for controlling details of how runs are executed, and for debugging your YAML files and the symlinks in the temporary run directory. Please see `salishsea help run` or the SalishSeaCmd package docs.

You can use the `qstat` command to monitor the execution status of your job.

A convenient command to monitor the memory use of a run and its time step progress is:

```
watch -n 5 "(free -m; cat time.step)"
```

When the job completes the results should have been gathered in the directory you specified in the `salishsea run` command and the temporary run directory should have been deleted.
Look at the Results

A number of notebooks that look at NetCDF files are available in analysis/analysis_tools/. To start these, go to the top level directory of the analysis repo on your local machine (not on salish) and type:

```
jupyter notebook
```

At this points alot of information will appear in your terminal. This terminal session is now running a server and cannot be used for anything else until you are finished with the notebooks. At that point you need to CTRL-C to get out.

At the same time a window should open in your browser. If it doesn’t, look at your terminal, find the ip address, something like:

```
The Jupyter Notebook is running at: http://127.0.0.1:8888/
```

and put that number in your browser. From this initial window you can open the notebooks in analysis_tools directory and look around. The links to the various files will probably not work. Change them to point to your file space. You will probably want to build your own notebook but these notebooks give you lots of examples to copy from.

**Working on orcinus: NEMO v3.4**

This section describes very briefly the steps to set up and run the Salish Sea NEMO version 3.4 code on the orcinus.westgrid.ca HPC cluster. This guide assumes that your Working Environment is set up, and that you are familiar with Working on salish : NEMO v3.6.

**Set-up SSH key-forwarding**

See orcinus instructions at the bottom of *ssh Configuration*.

**.bash_profile and .bashrc**

orcinus uses 2 files for bash settings: .bash_profile and .bashrc. In both files the location at which to add your personal customizations is indicated in comments in the default version of the files. Environment variables go in .bash_profile, for example:

```
#!/bin/bash

# Include your own tailored environment below
#
# Pager setup
export PAGER=less
export LESS=-r

# Make emacs the default full-screen editor (to ward off vi)
export EDITOR=emacs
export VISUAL=emacs
```

Shell variables, module loading (see below), and aliases go in .bashrc:
Please see bash Configuration for explanations of the above settings.

When working on Westgrid clusters the module command must be used to load extra software components. The required modules vary from cluster to cluster. On orcinus only the python module should be loaded when you log in:

```
module load python
```

Doing so makes Python, Mercurial, and the netCDF4 library available to you.

You can manually load the python module each time you log in, or you can add it to your .bashrc file (as shown above) so that it is automatically loaded when you ssh into orcinus.

Change the lpath line in the modify search path section of .bash_profile to include $HOME/.local/bin and $HOME/bin in your search path:

```
lpath=$HOME/.local/bin:$HOME/bin
```

### Create a Workspace and Clone the Repos

```
mkdir -p $HOME/MEOPAR/SalishSea/results
```

Clone the repos needed to run the model:

```
cd $HOME/MEOPAR
hg clone ssh://hg@bitbucket.org/salishsea/nemo-code NEMO-code
hg clone ssh://hg@bitbucket.org/salishsea/nemo-forcing NEMO-forcing
hg clone ssh://hg@bitbucket.org/salishsea/ss-run-sets SS-run-sets
hg clone ssh://hg@bitbucket.org/salishsea/nemo-cmd NEMO-Cmd
hg clone ssh://hg@bitbucket.org/salishsea/salishseacmd SalishSeaCmd
```

There is no need to clone the docs or analysis repos on orcinus.

### Install Tools and Command Processor Packages

Install the SalishSeaTools Package and Salish Sea NEMO Command Processor Python packages:
Compile the Code

Compile the Salish Sea NEMO configuration, and the rebuild_nemo tool:

```
cd $HOME/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea
source orcinus_build.sh
cd $HOME/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO
source orcinus_build.sh
```

Both invocations of the orcinus_build.sh script will produce lots of output that mentions build failures, but they should end with success messages and show the newly created executables. For CONFIG/SalishSea/orcinus_build.sh the output should end something like:

```
mpif90 -o nemo.o -I/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/inc -c -fpp -r8 -O3 -assume byterecl -heap-arrays -I/global/software/lib64/intel/ncsa-
tools/include /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/WORK/nemo.f90
fcm_internal load: F nemo nemo.o nemo.exe
mpif90 -o server.exe /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/obj/
  -server.o -L/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/lib -l__fcm_ 
  __server -shared-intel -lnetcdf -lnetcdff -lhdf5 -lhdf5_hl -lz -lsz
mpif90 -o nemo.exe /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/obj/
  -nemo.o -L/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/lib -l__fcm__ 
  -nemo -shared-intel -lnetcdff -lnetcdff_hl -lhdf5 -lhdf5_hl -lz -lsz
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_cpu_feature_indicator_x'
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_cpu_features_init_x'
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_ssse3_memmmove'
fcm_internal load failed (256)
make: *** [server.exe] Error 1
make: *** Waiting for unfinished jobs....
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_cpu_feature_indicator_x'
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_cpu_features_init_x'
/glerv/software/lib64/intel/ncsa-tools/lib/libnetcdf.so: undefined reference to `__ 
  -intel_ssse3_memmmove'
fcm_internal load failed (256)
make: *** [nemo.exe] Error 1
make -f /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea/BLD/Makefile -j 8
  --all failed (2) at /global/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/EXTERNAL/fcm/bin/..
  /lib/Fcm/Build.pm line 597
  >Make: 56 seconds
  >TOTAL: 86 seconds
/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG
/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/CONFIG/SalishSea
I/O server build succeeded at Mon Aug 18 12:34:23 PDT 2014
-rwrxr-x--- 1 dlatorne dlatorne 9935884 Aug 18 12:34 /home/dlatorne/MEOPAR/NEMO-code/ 
  _NEMOGCM/CONFIG/SalishSea/BLD/bin/server.exe*
```
NEMO executable build succeeded at Mon Aug 18 12:34:25 PDT 2014
-rwxr-x--- 1 dlatorne dlatorne 16102827 Aug 18 12:34 /home/dlatorne/MEOPAR/NEMO-code/
 ˓→NEMOGCM/CONFIG/SalishSea/BLD/bin/nemo.exe
I/O server executable symlinked in SalishSeaEXP00/ at Mon Aug 18 12:34:25 PDT 2014
lrwxrwxrwx 1 dlatorne dlatorne 75 Aug 18 12:34 /home/dlatorne/MEOPAR/NEMO-code/
 ˓→NEMOGCM/CONFIG/SalishSea/EXP00/server.exe -> /home/dlatorne/MEOPAR/NEMO-code/
 ˓→NEMOGCM/CONFIG/SalishSea/BLD/bin/server.exe
NEMO executable symlinked in SalishSea/EXP00/ as opa at Mon Aug 18 12:34:25 PDT 2014
lrwxrwxrwx 1 dlatorne dlatorne 73 Aug 18 12:34 /home/dlatorne/MEOPAR/NEMO-code/
 ˓→NEMOGCM/CONFIG/SalishSea/EXP00/opa -> /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/
 ˓→CONFIG/SalishSea/BLD/bin/nemo.exe

Similarly, the output of TOOLS/REBUILD_NEMO/orcinus_build.sh should end like:

```
mpif90 -o rebuild_nemo.o -I/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/
 ˓→BLD/inc -c -fpp -r8 -O3 -assume byterecl -heap-arrays -I/global/software/lib64/
 ˓→-intelfc/ncsa-tools/include -I/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/
 ˓→src/rebuild_nemo.f90
touch /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/flags/LD.flags
 ˓→flags
touch /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/flags/LD__nemo__
 ˓→rebuild_nemo.flags
touch /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/flags/LDFLAGS.
 ˓→flags
touch /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/flags/LDFLAGS__
 ˓→nemo.flags
touch /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/flags/LDFLAGS__
 ˓→nemo__rebuild_nemo.flags
fcm_internal load:F nemo rebuild_nemo.o rebuild_nemo.exe
Use of uninitialized value in split at /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/
 ˓→EXTERNAL/fcm/bin/fcm_internal line 377.
mpif90 -o rebuild_nemo.exe /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/
 ˓→BLD/lib -shared-intel -inetcdf -inetcdff -lhdf5 -lhdf5_hl -lz -lsz
/global/software/lib64/intelfc/ncsa-tools/lib/libnetcdf.so: undefined reference to `__
 ˓→intel_cpu_feature_indicator_x'
/global/software/lib64/intelfc/ncsa-tools/lib/libnetcdf.so: undefined reference to `__
 ˓→intel_cpu_features_init_x'
/global/software/lib64/intelfc/ncsa-tools/lib/libnetcdf.so: undefined reference to `__
 ˓→intel_ssse3_memmove'
fcf_internal load failed (256)
makes: *** [rebuild_nemo.exe] Error 1
make -f /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/Makefile -j 1,
 ˓→all failed (2) at /global/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/EXTERNAL/fcm/bin/..
 ˓→-l/lib/Fcm/Build.pm line 597
=>Make: 4 seconds
=>TOTAL: 4 seconds
/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS
ls: cannot access /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO/BLD/bin/
 ˓→*.exe: No such file or directory
/home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/REBUILD_NEMO
rebuild_nemo build succeeded at Mon Aug 18 12:56:08 PDT 2014
-rwxr-x--- 1 dlatorne dlatorne 108611 Aug 18 12:56 /home/dlatorne/MEOPAR/NEMO-code/
 ˓→NEMOGCM/TOOLS/REBUILD_NEMO/BLD/bin/rebuild_nemo.exe
rebuild_nemo executable symlinked in /home/dlatorne/MEOPAR/NEMO-code/NEMOGCM/TOOLS/
 ˓→REBUILD_NEMO/ at Mon Aug 18 12:56:08 PDT 2014
```
Prepare and Execute Runs

The SS-run-sets/ SalishSea/ Directory contains version controlled sample run description files and namelist segment files. In your own directory in that repo copy, edit, and version control those files to define the runs that you want to execute.

The run description file is described in the Run Description File Structure section of the project tools documentation. The namelists are described in the NEMO-3.4 Book.

Use salishsea run Sub-command to prepare, execute, and gather the results for a run:

```
salishsea run--nemo3.4 SalishSea.yaml iodef.xml $HOME/MEOPAR/SalishSea/results/my_ -->excellent_results
```

salishsea run returns the path and name of the temporary run directory, and the job identifier assigned by the queue manager, something like:

```
salishsea_cmd.prepare INFO: Created run directory /home/dlatorne/MEOPAR/SalishSea/ -->a90d391c-0e1e-11e4-aa4e-6431504adba6
salishsea_cmd.run INFO: 3544250.orca2.ibb
```

salishsea run has a number of command-line option flags that are useful for controlling details of how runs are executed, and for debugging your YAML files and the symlinks in the temporary run directory. Please see salishsea help run or the SalishSeaCmd package docs.

You can use the job identifier with qstat, showstart, and checkjob to monitor the execution status of your job. When the job completes the results should have been gathered in the directory you specified in the salishsea run command and the temporary run directory should have been deleted.

To view and analyze the run results copy them to your EOAS /data/$USER/results/ workspace with scp or sftp.

2.3.2 Set-up, Initial Conditions, Forcing, etc. Files

The collection of domain-specific set-up, initial conditions, open boundary conditions, forcing, etc. files used to run NEMO for the Salish Sea are maintained in 4 repositories:

- grid
- rivers-climatology
- tides
- tracers

They can be found in the SalishSea-MEOPAR NEMO Model Runs collection of repos on Bitbucket.

If you have set up SSH key authentication on Bitbucket, you can clone the grid, rivers-climatology, tides, and tracers repos with:
hg clone ssh://hg@bitbucket.org/salishsea/grid
hg clone ssh://hg@bitbucket.org/salishsea/rivers-climatology
hg clone ssh://hg@bitbucket.org/salishsea/tides
hg clone ssh://hg@bitbucket.org/salishsea/tracers

For password authentication use:

hg clone https://<you>@bitbucket.org/salishsea/grid
hg clone https://<you>@bitbucket.org/salishsea/rivers-climatology
hg clone https://<you>@bitbucket.org/salishsea/tides
hg clone https://<you>@bitbucket.org/salishsea/tracers

where <you> is your Bitbucket user id.

**Atmospheric Forcing**

NEMO includes a CORE bulk formulae interface that allows the CGRF Dataset to be used as atmospheric forcing (aka surface boundary conditions; SBC). The namelist that controls that is nambc_core. That NEMO interface requires that all of the files (including the Interpolation Weights) be in a single directory and have a prescribed name pattern. To satisfy that requirement the following file management strategy has been established:

- The Salish Sea MEOPAR repository of CGRF product files and their NEMO CORE bulk interface representation reside in /ocean/dlatorne/MEOPAR/CGRF/

- Within that directory is 1 sub-directory:
  - NEMO-atmos/ is a collection of CGRF files using the names required by the NEMO CORE bulk interface. The NEMO-atmos/ directory also contains symbolic links to the Interpolation Weights file and the No-Snow Constraint file in the grid repo.

- The files in the /ocean/dlatorne/MEOPAR/CGRF/ directory are managed by the salishsea get_cgrf tool.

- The namelist that directs NEMO to use the files in NEMO-atmos/ looks like:

```plaintext
!-----------------------------------------------------------------------
! nambc_core   nambc_core CORE bulk formulae
!-----------------------------------------------------------------------
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!
weights
!”
```

(continues on next page)
cn_dir = 'NEMO-atmos/' ! root directory for the location of the bulk files
ln_2m = .true. ! air temperature and humidity referenced at 2m

The file name roots and the variable names come from the CGRF files.

- The directory given in as value associated with the atmospheric key in the forcing section of the run description file is symlinked as NEMO-atmos in the run directory to complete the chain from the namelist to the CGRF products files:

```
forcing:
  atmospheric: /ocean/dlatorne/MEOPAR/CGRF/NEMO-atmos/
```

No-Snow Constraint

The NEMO CORE bulk interface requires a snow file but it has been decided to impose a constraint on the Salish Sea model whereby all precipitation falls in liquid phase. That constraint is provided by the no_snow.nc file in the grid repo. That file ensures that the solid phase precipitation is zero at all times and locations. It is used as annual climatology forcing in the namsbc_core name list.

Interpolation Weights

JP Paquin created a weights file that allows NEMO’s Interpolation On the Fly (IOF) feature to be used to take atmospheric forcing values from the CGRF Dataset. It is in the 4_weights_ATMOS/ directory of the 2-Oct-2013 WC3_PREP tarball and has been added to the grid/ of the NEMO-forcing repo. Those weight values were subsequently used to create a netCDF4 weights file with zlib compression enabled and CF-1.6 conventions compliant attributes using the I_ForcingFiles/Atmos/netCDF4weights-CGRF.ipynb notebook.

Creating New Weights Files

The NEMO_Preparation/4_weights_ATMOS/get_weight_nemo program in the NEMO_EastCoast repo can be used in conjunction with a bathymetry file and atmospheric forcing file(s) to create a weights file that allows NEMO’s Interpolation On the Fly (IOF) feature to use the atmospheric forcing values. An example of the use of get_weight_nemo to create a weights file for datasets from the operational West deployment of Environment Canada’s High Resolution Deterministic Prediction System (HRDPS) is presented here:

Clone the NEMO_EastCoast repo on salish and edit the NEMO_Preparation/4_weights_ATMOS/make.sh file to comment out the default build commands and enable the salish ones:

```
#!/ On salish (UBC)
LIBNETCDF=/usr
mpif90 -c grid.f90 -I$(LIBNETCDF)/include -L$(LIBNETCDF)/lib -lnetcdf
mpif90 -c map.F90 -I$(LIBNETCDF)/include -L$(LIBNETCDF)/lib -lnetcdf
mpif90 -c get_weight_nemo.F90 -I$(LIBNETCDF)/include -L$(LIBNETCDF)/lib -lnetcdf
mpif90 -o get_weight_nemo get_weight_nemo.o map.o grid.o -I$(LIBNETCDF)/include -L$(LIBNETCDF)/lib -lnetcdf -lnetcdff

Build get_weight_nemo:
```
get_weight_nemo creates a file of weighting factors that allow atmospheric forcing variable values on one grid to be interpolated onto the model grid (as defined in the bathymetry dataset). To do that it requires:

1. a bathymetry dataset, the name of which is hard-coded to bathy_meter.nc
2. a namelist file, the name of which is hard-coded to namelist, and an example of which is contained in the NEMO_Preparation/4_weights_ATMOS/ directory
3. one or more atmospheric forcing dataset file(s), the name of which is defined in the namelist

The output of get_weight_nemo is a weights file, the name of which is hard-coded to met_gem_weight.nc.

We’ll run get_weight_nemo in NEMO-forcing/grid/, so start by copying the sample namelist file to there, changing to that directory, and symlinking it as namelist:

```
$ cp namelist ../../../NEMO-forcing/grid/namelist.get_weight_nemo.gem2.5-ops
$ cd ../../../NEMO-forcing/grid/
$ ln -s namelist.get_weight_nemo.gem2.5-ops namelist
```

Symlink the bathymetry dataset as bathy_meter.nc:

```
$ ln -s bathy_meter_SalishSea2.nc bathy_meter.nc
```

The only values that get_weight_nemo actually uses from the atmospheric forcing dataset file is the grid point locations, but the namelist file is more complicated. We can reduce the complexity by using a single atmospheric forcing dataset file as a climatology, so we symlink one as atmos.nc:

```
$ ln -s /ocean/sallen/allen/research/Meopar/Operational/oper_allvar_ss_y2014m09d23.nc atmos.nc
```

Next, edit the namelist file to point to that symlink:

```
!-----------------------------------------------------------------------
&namsbc_core
! namsbc_core CORE bulk formulae
!-----------------------------------------------------------------------
! file name ! variable ! clim ! 'yearly'/'monthly'
! name ! (T/F) 

sn_wndi = 'atmos', 'u_wind' , .true., 'yearly'
sn_wndj = 'atmos', 'v_wind' , .true., 'yearly'
sn_qsr  = 'atmos', 'solar'  , .true., 'yearly'
sn qlw  = 'atmos', 'therm_rad', .true., 'yearly'
sn_tair = 'atmos', 'tair'   , .true., 'yearly'
sn humi = 'atmos', 'qair'   , .true., 'yearly'
sn prec = 'atmos', 'precip' , .true., 'yearly'
sn snow = 'atmos', 'snow'   , .true., 'yearly'
 cn_dir = './' ! root directory for the location of the bulk files
```

The important things here are:

- The file name must match the name of the atmospheric forcing dataset file symlink, without the .nc extension.
- The climatology field (clim (T/F)) must be set to .true. for all variables.
- The value of cn_dir must be './'.

Finally, run get_weight_nemo:
The output should be something like:

```plaintext
sbc_blk_core : flux formulation for ocean surface boundary condition
~~~~~~~~~~
nsbc_core Namelist
  list of files
    root filename: ./atmos variable name: u_wind climatology: T data
    type: yearly
    root filename: ./atmos variable name: v_wind climatology: T data
    type: yearly
    root filename: ./atmos variable name: qair climatology: T data
    type: yearly
    root filename: ./atmos variable name: solar climatology: T data
    type: yearly
    root filename: ./atmos variable name: therm_rad climatology: T data
    type: yearly
    root filename: ./atmos variable name: tair climatology: T data
    type: yearly
    root filename: ./atmos variable name: precip climatology: T data
    type: yearly
    root filename: ./atmos variable name: snow climatology: T data
    type: yearly
reading : ./atmos.nc
atmospheric forcing netcdf grid dimensions: nx= 256 , ny= 266
  get_atmo_grid ~~ found X axis varid: 3
  get_atmo_grid ~~ found Y axis varid: 2
grid_type 2
xmin/xmax/origin 0.230833E+03 0.240530E+03 0.230833E+03
writing variable : src01
  8065 8065 8321 8321
status put 0
writing variable : wgt01
writing variable : src02
  8321 8321 8065 8065
status put 0
writing variable : wgt02
writing variable : src03
  8064 8064 8320 8320
status put 0
writing variable : wgt03
writing variable : src04
  8320 8320 8064 8064
status put 0
writing variable : wgt04
```

and a `met_gem_weight.nc` file should be created.

Use the `I_ForcingFiles/Atmos/ImproveWeightsFile.ipynb` notebook to transform `met_gem_weight` into a netCDF4 file called `weights-2.5kmGEM-ops.nc` with well-structured metadata (see netCDF4 Files Creation and Conventions).

### CGRF Dataset

The Canadian Meteorological Centre’s (CMC) Global Deterministic Prediction System (GDPS) Reforecasts (CGRF) dataset is a relatively high-resolution forcing dataset for ocean models [Smith_etal2013]. The dataset is hosted on an
rsync server at goapp.ocean.dal.ca. User id and password credentials are required to access it.

At the command line you can explore the dataset with commands like:

```
rsync <userid>@goapp.ocean.dal.ca::canadian_GDPS_reforecasts_v1/2002/2002091500
Password: 
```
```
dr-xr-xr-x 4096 2012/06/14 06:59:22 2002091500
```

and

```
rsync <userid>@goapp.ocean.dal.ca::canadian_GDPS_reforecasts_v1/2002/2002091500/
Password: 
```
```
dr-xr-xr-x 4096 2012/06/14 06:59:22 .
-r-r-xr-x 8844469 2011/06/06 07:46:01 2002091500_precip.nc.gz
-r-r-xr-x 27045976 2011/01/14 21:37:09 2002091500_q2.nc.gz
-r-r-xr-x 20960161 2011/01/14 21:37:26 2002091500 qlw.nc.gz
-r-r-xr-x 10451631 2011/01/14 21:37:34 2002091500_qsw.nc.gz
-r-r-xr-x 11655341 2011/01/14 21:37:37 2002091500 slp.nc.gz
-r-r-xr-x 27080056 2011/01/14 21:37:15 2002091500_t2.nc.gz
-r-r-xr-x 37703920 2011/01/14 21:37:04 2002091500_u10.nc.gz
-r-r-xr-x 37641390 2011/01/14 21:37:05 2002091500_v10.nc.gz
```

Note that the trailing slash causes the contents of a directory to be accessed while its absence refers to the directory itself.

To make a local copy of files use the -rltv options and provide a destination directory (which will be created if it doesn’t already exist):

```
rsync -rltv <userid>@goapp.ocean.dal.ca::canadian_GDPS_reforecasts_v1/2002/2002091500/ → 2002-09-15/
Password: 
```
```
receiving incremental file list
./
2002091500_precip.nc.gz
2002091500_q2.nc.gz
2002091500 qlw.nc.gz
2002091500_qsw.nc.gz
2002091500 slp.nc.gz
2002091500_t2.nc.gz
2002091500_u10.nc.gz
2002091500_v10.nc.gz
```
```
sent 234 bytes received 181405678 bytes 6596578.62 bytes/sec
total size is 181382944 speedup is 1.00
```

The local files are created with 555 permissions. Make them user and group writable so that they can be decompressed, and non-executable with:

```
chmod 664 2002-09-15/*
```

The salishsea get_cgrf tool automates this process.

Pressure Correction

The CGRF atmospheric model uses a terrain following vertical coordinate system which means that the lowest grid cells are not at sea level in mountainous regions such as those surrounding the Salish Sea. As such, we have developed an algorithm to adjust CGRF pressure files to sea level. First, the altitude of each grid cell is computed since this is
not given in the CGRF output. Given the of an air parcel, we can approximate its height $z_1$ above sea level using the following formula [Holton1992]:

$$p_s = p_1 \left( \frac{\gamma z_1}{T_1} + 1 \right)^{\frac{g}{\gamma R}}$$

where $g$ is the acceleration due to gravity, $R$ is the ideal gas constant, and $\gamma$ is the temperature lapse rate of the atmosphere (0.0098 degrees/m).

To arrive at this formula we have made a few assumptions:

1. The atmosphere is in hydrostatic equilibrium: $\frac{dp}{dz} = -\rho g$
2. The atmosphere is an ideal gas: $p = \rho RT$
3. The temperature of the atmosphere decreases with height at a constant rate: $\frac{dT}{dz} = -\gamma$

The altitude of each grid cell is stored in a file `altitude_CGRF.nc` in the `tools/I_ForcingFiles/Atmos` repository.

This sea level pressure calculation is performed in `nc_tools.generate_pressure_file`, which is used in `salishsea get_cgrf` to correct pressure files on download. Corrected pressure files are named `slp_corr_y0000m00d00.nc`. See the tools docs for details on `nc_tools.generate_pressure_file` method.

**Note:** `salishsea get_cgrf` requires a link to `altitude_CGRF.nc` in `/NEMO-atmos/`.

### Initializing T+S with a restart file

This section describes how to initialize the salinity and temperature fields with data from a restart file. This is useful for taking the stratification from a spin-up or nowcast run.

First, in the `SalishSea.yaml` file provide an initial conditions forcing link that points to the path of the directory that contains the restart file. For example, the restart file `SalishSea_00069120_restart.nc` is in `/ocean/dlatorne/MEOPAR/SalishSea/results/spin-up/18oct25oct/`, so the forcing section of the `.yaml` files should contain:

```
forcing:
  ...
  initial_strat:
    link to: /ocean/dlatorne/MEOPAR/SalishSea/results/spin-up/18oct25oct/
    ...
```

Next, modify the `&namtsd` section of `namelist.domain` so that NEMO reads in the restart file. For example:

```plaintext
&namtsd
  ! data : Temperature & Salinity
  !-----------------------------------------------------------------------
  ! " file name ! freq (hr) ! variable ! "
  ! time ! clim ! period ! weights ! rotation !
  ! " ! (mo) ! name ! "
  ! interp ! (T/F) ! filename ! pairing !
  sn_tem = 'SalishSea_00069120_restart.nc', -12, 'tb', .
  ->false, .true., 'yearly', '', 
  sn_sal = 'SalishSea_00069120_restart.nc', -12, 'sb', .
  ->false, .true., 'yearly', '', 
  cn_dir = 'initial_strat/' ! directory containing initial condition
  ->files
```

(continues on next page)
The file name must be modified to the restart file’s name. Additionally, since restart files have a different naming convention for the field variables, ensure that 'tb' and 'sb' are used for temperature and salinity in the variable name section.

**Considerations**

- Bathymetry should be consistent between the restart file and the run you are initializing.
- There may be issues of continuity between the restart T+S data and the boundary conditions at Juan de Fuca, depending on the date chosen for a restart.

**Open Boundary Conditions**

NEMO separates open boundary conditions into four parts.

- Tides: see tide description elsewhere
- T & S: Temperature and Salinity fields. For these we are using the built-in FRS system. This is a multi-grid point forcing toward the specified boundary conditions with the forcing equal to one at the boundary and decreasing as a tanh from there.
- Baroclinic velocities: We are using the time-splitting scheme so barotropic and baroclinic velocities are set separately.
- Barotropic velocities and SSH

Currently open boundaries are only used on the western boundary. It would be good to include T&S boundary conditions on the northern boundary. Many updates for 2 boundaries occurred between stable and current on 3-4. We should update before implementing.

**Temperature and Salinity**

We are using the FRS scheme with a 10-grid point rim. Initial values come from a quick fit to Thomson et al. These use identical values on the 10 grid points and uniform cross-strait values. Are new set of values are from Masson and Fine (2013), with different values on the 10 grid points, and with a seasonal cycle, changing weekly.

**Baroclinic Velocities**

We have implemented zero-gradient boundary conditions. These appeared to work well provided the temperature and salinity changes were started weakly. However, when attempting a 40-day run, these conditions blew at about day 9.5 with an error at i=392, k=12. We smoothed the near boundary topography identical in j (outward from the boundary for 6 grid points) and smoothed it. Now we are running well beyond 50 days.

Velocities set at #2 grid points because #1 points are not used by BDY (set by OBC)
Barotropic Velocities & SSH

Built in Flather boundary conditions take the tides (ssh and velocity) and add them to barotropic forcing (ssh and velocity). These conditions are implemented only on the barotropic velocities directly but affect the ssh indirectly. We are setting ssh based on Tofino measurements. We are no forcing barotropic velocities but allowing any “waves” set inside the Strait to propagate out (which is what Flather conditions do).

Velocities set at #2 grid points.

netCDF4 Files Creation and Conventions

The Salish Sea MEOPAR project uses netCDF4 files as input for the NEMO model and for other purposes, where appropriate. The netCDF4 Files Creation and Conventions section in the project tools documentation documents the recommended way of creating netCDF4 files with compression of variables, limitation of variables to appropriate precision, and appropriate metadata attributes for the variables and the dataset as a whole. The recommendations are based on the NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.6, 5 December, 2011. Use of the netCDF4-python library (included in Anaconda Python Distribution) is assumed.

The nc_tools Module in the SalishSeaTools Package is a library of Python functions for exploring and managing the attributes of netCDF files.

2.3.3 Salish Sea Run Sets Files

The collection of NEMO namelist, salishsea command processor run description, and NEMO output server control files used to run NEMO for the Salish Sea are maintained in the SS-run-sets repo.

We choose to call these run-set files but they are also referred to as “experiments” in the NEMO community. That term is difficult for the laboratory modelers on the Salish Sea MEOPAR team to apply to computational model runs.

The salishsea command processor run description files are YAML files that are used by our Salish Sea NEMO Command Processor tool for managing NEMO runs and results. The Run Description File Structure section describes the file syntax.

The namelist files contain collections of related namelists. They are concatenated to form a complete namelist for a NEMO run by the salishsea prepare Sub-command.

The output server control files is like those found in the EXP00/ directories of the reference configurations in the NEMO-code repo.

Getting the Repo

If you use SSH key authentication on Bitbucket you may clone the SS-run-sets repo with:

```
hg clone ssh://hg@bitbucket.org/salishsea/ss-run-sets SS-run-sets
```

For password authentication use:

```
hg clone https://<you>@bitbucket.org/salishsea/ss-run-sets SS-run-sets
```

where <you> is your Bitbucket user id.
**Repo Contents**

**SalishSea/ Directory**

The `SalishSea` directory contains the run-set files for the initial full domain runs:

- `SalishSea.yaml`: Sample run description file for use with the Salish Sea NEMO Command Processor
- `namelist.bottom`: Sample bottom boundary conditions namelists
- `namelist.compute.1core`: Sample compute parameters namelists for a single core run (useful for debugging on salish)
- `namelist.compute.4x4`: Sample compute parameters namelists for 4x4 MPI domain decomposition (the recommended decomposition for running on salish)
- `namelist.compute.6x14`: Sample compute parameters namelists for 6x14 MPI decomposition (the recommended decomposition for running on jasper)
- `namelist.domain`: Sample domain configuration namelists
- `namelist.dynamics`: Sample dynamics parameter namelists
- `namelist.lateral`: Sample lateral boundary conditions and forcing namelists
- `namelist.surface`: Sample surface boundary conditions namelists
- `namelist.time`: Sample model time parameters namelists
- `namelist.tracers`: Sample tracer quantities configuration namelist
- `iodef.xml`: Sample IOM output server definitions
- `xmlio_server.def`: IOM output server control settings

**JPP/ Directory**

The `JPP/` directory contains the run-set files for the initial sub-domain test case also known as `WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY`:

- `JPP.yaml`
- `namelist`

### 2.3.4 Land Processor Elimination

NEMO-3.6 includes land processor elimination and it lets you reduce your computational cost when some of your processors are assigned tiles that are entirely land. The amount of land (and computational cost) eliminated depends on the MPI decomposition and the model domain. The feature is activated by specifying a MPI decomposition (eg, `jnpi`, `jpnj` = 8, 18) and then specifying the number of water processors (`jpnij` = 88). When NEMO notices that `jpnij` is less than the product of `jnpi` and `jpnj` (here 144) it switches on the land elimination feature. However, you need to know the number of water processors before running NEMO so you can request that number of processors on the compute system (eg, westgrid).

**Forward problem**

Determining the number of water processors, given a domain and a decomposition, is the forward problem. NEMO includes a tool under TOOLS/MPP_PREP that computes the number of water processors for all possible decompositions (up to a maximum number of processors). The calculation is unique to each bathymetry file, and we store the
calculation results in a .csv files in the grid. The salishsea run and salishsea prepare commands from the Salish Sea NEMO Command Processor can use the .csv file as a lookup table in order to request the correct number of water processors for the domain decomposition that you set.

Reverse problem

Determining the decomposition to use, given a domain and target number of processors, is the reverse problem. It turns out that there are often multiple decompositions that yield the same number of water processors.

A good decomposition has a high fraction of land eliminated and a tile aspect ratio near unity. We select the preferred decompositions by filtering for small aspect ratio (ar <= 1.15) and then for most land eliminated (smallest r = ratio of water processors to total). The tables linked below list the preferred decompositions for bathymetries “downbyone2” and “201702”. The scripts here produce the tables.

Preferred MPI LPE Decompositions

Preferred MPI LPE Decompositions for bathymetry downonegrid2

This table shows the preferred MPI decompositions that optimize Land Processor Elimination for the SalishSea and SMELT configuration running in NEMO-3.6 with the bathy_downonegrid2.nc bathymetry.

<table>
<thead>
<tr>
<th>MPI breakdown</th>
<th>Water</th>
<th>Land</th>
<th>r</th>
<th>Tile size</th>
<th>Tile aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x7 = 21</td>
<td>20</td>
<td>1</td>
<td>0.952</td>
<td>134x130 = 17420</td>
<td>1.031</td>
</tr>
<tr>
<td>4x8 = 32</td>
<td>27</td>
<td>5</td>
<td>0.844</td>
<td>101x114 = 11514</td>
<td>1.129</td>
</tr>
<tr>
<td>4x9 = 36</td>
<td>29</td>
<td>7</td>
<td>0.806</td>
<td>101x102 = 10302</td>
<td>1.010</td>
</tr>
<tr>
<td>4x10 = 40</td>
<td>33</td>
<td>7</td>
<td>0.825</td>
<td>101x92 = 9292</td>
<td>1.098</td>
</tr>
<tr>
<td>5x10 = 50</td>
<td>38</td>
<td>12</td>
<td>0.760</td>
<td>82x92 = 7544</td>
<td>1.122</td>
</tr>
<tr>
<td>5x11 = 55</td>
<td>42</td>
<td>13</td>
<td>0.764</td>
<td>82x84 = 6888</td>
<td>1.024</td>
</tr>
<tr>
<td>5x12 = 60</td>
<td>44</td>
<td>16</td>
<td>0.733</td>
<td>82x77 = 6314</td>
<td>1.065</td>
</tr>
<tr>
<td>6x12 = 72</td>
<td>51</td>
<td>21</td>
<td>0.708</td>
<td>68x77 = 5236</td>
<td>1.132</td>
</tr>
<tr>
<td>6x13 = 78</td>
<td>54</td>
<td>24</td>
<td>0.692</td>
<td>68x71 = 4828</td>
<td>1.044</td>
</tr>
<tr>
<td>6x15 = 90</td>
<td>59</td>
<td>31</td>
<td>0.656</td>
<td>68x62 = 4216</td>
<td>1.097</td>
</tr>
<tr>
<td>7x15 = 105</td>
<td>64</td>
<td>41</td>
<td>0.610</td>
<td>59x62 = 3658</td>
<td>1.051</td>
</tr>
<tr>
<td>7x16 = 112</td>
<td>72</td>
<td>40</td>
<td>0.643</td>
<td>59x58 = 3422</td>
<td>1.017</td>
</tr>
<tr>
<td>7x17 = 119</td>
<td>74</td>
<td>45</td>
<td>0.622</td>
<td>59x55 = 3245</td>
<td>1.073</td>
</tr>
<tr>
<td>7x18 = 126</td>
<td>76</td>
<td>50</td>
<td>0.603</td>
<td>59x52 = 3068</td>
<td>1.135</td>
</tr>
<tr>
<td>8x16 = 128</td>
<td>83</td>
<td>45</td>
<td>0.648</td>
<td>52x58 = 3016</td>
<td>1.115</td>
</tr>
<tr>
<td>8x18 = 144</td>
<td>88</td>
<td>56</td>
<td>0.611</td>
<td>52x52 = 2704</td>
<td>1.000</td>
</tr>
<tr>
<td>8x20 = 160</td>
<td>96</td>
<td>64</td>
<td>0.600</td>
<td>52x47 = 2444</td>
<td>1.106</td>
</tr>
<tr>
<td>9x18 = 162</td>
<td>100</td>
<td>62</td>
<td>0.617</td>
<td>46x52 = 2392</td>
<td>1.130</td>
</tr>
<tr>
<td>9x19 = 171</td>
<td>104</td>
<td>67</td>
<td>0.608</td>
<td>46x50 = 2300</td>
<td>1.087</td>
</tr>
<tr>
<td>9x20 = 180</td>
<td>109</td>
<td>71</td>
<td>0.606</td>
<td>46x47 = 2162</td>
<td>1.022</td>
</tr>
<tr>
<td>9x21 = 189</td>
<td>113</td>
<td>76</td>
<td>0.598</td>
<td>46x45 = 2070</td>
<td>1.022</td>
</tr>
<tr>
<td>9x22 = 198</td>
<td>114</td>
<td>84</td>
<td>0.576</td>
<td>46x43 = 1978</td>
<td>1.070</td>
</tr>
<tr>
<td>10x20 = 200</td>
<td>116</td>
<td>84</td>
<td>0.580</td>
<td>42x47 = 1974</td>
<td>1.119</td>
</tr>
<tr>
<td>9x23 = 207</td>
<td>123</td>
<td>84</td>
<td>0.594</td>
<td>46x41 = 1886</td>
<td>1.122</td>
</tr>
<tr>
<td>10x21 = 210</td>
<td>124</td>
<td>86</td>
<td>0.590</td>
<td>42x45 = 1890</td>
<td>1.071</td>
</tr>
<tr>
<td>10x22 = 220</td>
<td>126</td>
<td>94</td>
<td>0.573</td>
<td>42x43 = 1806</td>
<td>1.024</td>
</tr>
<tr>
<td>11x22 = 242</td>
<td>129</td>
<td>113</td>
<td>0.533</td>
<td>38x43 = 1634</td>
<td>1.132</td>
</tr>
<tr>
<td>10x23 = 230</td>
<td>132</td>
<td>98</td>
<td>0.574</td>
<td>42x41 = 1722</td>
<td>1.024</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>MPI breakdown</th>
<th>Water</th>
<th>Land</th>
<th>r</th>
<th>Tile size</th>
<th>Tile aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>11x23 = 253</td>
<td>134</td>
<td>119</td>
<td>0.530</td>
<td>38x41 = 1558</td>
<td>1.079</td>
</tr>
<tr>
<td>11x24 = 264</td>
<td>137</td>
<td>127</td>
<td>0.519</td>
<td>38x40 = 1520</td>
<td>1.053</td>
</tr>
<tr>
<td>10x25 = 250</td>
<td>142</td>
<td>108</td>
<td>0.568</td>
<td>42x38 = 1596</td>
<td>1.105</td>
</tr>
<tr>
<td>11x25 = 275</td>
<td>145</td>
<td>130</td>
<td>0.527</td>
<td>38x38 = 1444</td>
<td>1.000</td>
</tr>
<tr>
<td>10x26 = 260</td>
<td>146</td>
<td>114</td>
<td>0.562</td>
<td>42x37 = 1554</td>
<td>1.135</td>
</tr>
<tr>
<td>11x26 = 286</td>
<td>147</td>
<td>139</td>
<td>0.514</td>
<td>38x37 = 1406</td>
<td>1.027</td>
</tr>
<tr>
<td>11x27 = 297</td>
<td>156</td>
<td>141</td>
<td>0.525</td>
<td>38x36 = 1368</td>
<td>1.056</td>
</tr>
<tr>
<td>12x25 = 300</td>
<td>157</td>
<td>143</td>
<td>0.523</td>
<td>35x38 = 1330</td>
<td>1.086</td>
</tr>
<tr>
<td>11x28 = 308</td>
<td>159</td>
<td>149</td>
<td>0.516</td>
<td>38x34 = 1292</td>
<td>1.118</td>
</tr>
<tr>
<td>12x26 = 312</td>
<td>166</td>
<td>146</td>
<td>0.532</td>
<td>35x37 = 1295</td>
<td>1.057</td>
</tr>
<tr>
<td>12x27 = 324</td>
<td>170</td>
<td>154</td>
<td>0.525</td>
<td>35x36 = 1260</td>
<td>1.029</td>
</tr>
<tr>
<td>13x26 = 338</td>
<td>175</td>
<td>163</td>
<td>0.518</td>
<td>33x37 = 1221</td>
<td>1.121</td>
</tr>
<tr>
<td>12x28 = 336</td>
<td>177</td>
<td>159</td>
<td>0.527</td>
<td>35x34 = 1190</td>
<td>1.029</td>
</tr>
<tr>
<td>13x27 = 351</td>
<td>181</td>
<td>170</td>
<td>0.516</td>
<td>33x36 = 1188</td>
<td>1.091</td>
</tr>
<tr>
<td>12x30 = 360</td>
<td>183</td>
<td>177</td>
<td>0.508</td>
<td>35x32 = 1120</td>
<td>1.094</td>
</tr>
<tr>
<td>13x28 = 364</td>
<td>187</td>
<td>177</td>
<td>0.514</td>
<td>33x34 = 1122</td>
<td>1.030</td>
</tr>
<tr>
<td>12x31 = 372</td>
<td>191</td>
<td>181</td>
<td>0.513</td>
<td>35x31 = 1085</td>
<td>1.129</td>
</tr>
<tr>
<td>13x29 = 377</td>
<td>193</td>
<td>184</td>
<td>0.512</td>
<td>33x33 = 1089</td>
<td>1.000</td>
</tr>
<tr>
<td>14x29 = 406</td>
<td>195</td>
<td>211</td>
<td>0.480</td>
<td>31x33 = 1023</td>
<td>1.065</td>
</tr>
<tr>
<td>14x28 = 392</td>
<td>196</td>
<td>196</td>
<td>0.500</td>
<td>31x34 = 1054</td>
<td>1.097</td>
</tr>
<tr>
<td>14x30 = 420</td>
<td>199</td>
<td>221</td>
<td>0.474</td>
<td>31x32 = 992</td>
<td>1.032</td>
</tr>
<tr>
<td>13x31 = 403</td>
<td>205</td>
<td>198</td>
<td>0.509</td>
<td>33x31 = 1023</td>
<td>1.065</td>
</tr>
<tr>
<td>14x31 = 434</td>
<td>208</td>
<td>226</td>
<td>0.479</td>
<td>31x31 = 961</td>
<td>1.000</td>
</tr>
<tr>
<td>14x32 = 448</td>
<td>211</td>
<td>237</td>
<td>0.471</td>
<td>31x30 = 930</td>
<td>1.033</td>
</tr>
<tr>
<td>15x29 = 435</td>
<td>214</td>
<td>221</td>
<td>0.492</td>
<td>29x33 = 957</td>
<td>1.138</td>
</tr>
<tr>
<td>13x33 = 429</td>
<td>217</td>
<td>212</td>
<td>0.506</td>
<td>33x30 = 990</td>
<td>1.100</td>
</tr>
<tr>
<td>15x30 = 450</td>
<td>220</td>
<td>230</td>
<td>0.489</td>
<td>29x32 = 928</td>
<td>1.103</td>
</tr>
<tr>
<td>14x33 = 462</td>
<td>221</td>
<td>241</td>
<td>0.478</td>
<td>31x30 = 930</td>
<td>1.033</td>
</tr>
<tr>
<td>14x34 = 476</td>
<td>225</td>
<td>251</td>
<td>0.473</td>
<td>31x29 = 899</td>
<td>1.069</td>
</tr>
<tr>
<td>15x31 = 465</td>
<td>227</td>
<td>238</td>
<td>0.488</td>
<td>29x31 = 899</td>
<td>1.069</td>
</tr>
<tr>
<td>14x35 = 490</td>
<td>233</td>
<td>257</td>
<td>0.476</td>
<td>31x28 = 868</td>
<td>1.107</td>
</tr>
<tr>
<td>16x31 = 496</td>
<td>235</td>
<td>261</td>
<td>0.474</td>
<td>27x31 = 837</td>
<td>1.148</td>
</tr>
<tr>
<td>14x36 = 504</td>
<td>236</td>
<td>268</td>
<td>0.468</td>
<td>31x27 = 837</td>
<td>1.148</td>
</tr>
<tr>
<td>15x33 = 495</td>
<td>240</td>
<td>255</td>
<td>0.485</td>
<td>29x30 = 870</td>
<td>1.034</td>
</tr>
<tr>
<td>14x37 = 518</td>
<td>242</td>
<td>276</td>
<td>0.467</td>
<td>31x27 = 837</td>
<td>1.148</td>
</tr>
<tr>
<td>15x34 = 510</td>
<td>244</td>
<td>266</td>
<td>0.478</td>
<td>29x29 = 841</td>
<td>1.000</td>
</tr>
<tr>
<td>16x32 = 512</td>
<td>245</td>
<td>267</td>
<td>0.479</td>
<td>27x30 = 810</td>
<td>1.111</td>
</tr>
<tr>
<td>16x33 = 528</td>
<td>254</td>
<td>274</td>
<td>0.481</td>
<td>27x30 = 810</td>
<td>1.111</td>
</tr>
<tr>
<td>16x34 = 544</td>
<td>255</td>
<td>289</td>
<td>0.469</td>
<td>27x29 = 783</td>
<td>1.074</td>
</tr>
<tr>
<td>15x36 = 540</td>
<td>256</td>
<td>284</td>
<td>0.474</td>
<td>29x27 = 783</td>
<td>1.074</td>
</tr>
</tbody>
</table>

Preferred MPI LPE Decompositions for bathymetry 201702

This table shows the preferred MPI decompositions that optimize Land Processor Elimination for the SalishSea and SMELT configuration running in NEMO-3.6 with the bathymetry_201702.nc bathymetry.

<table>
<thead>
<tr>
<th>MPI breakdown</th>
<th>Water</th>
<th>Land</th>
<th>r</th>
<th>Tile size</th>
<th>Tile aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x7 = 21</td>
<td>20</td>
<td>1</td>
<td>0.952</td>
<td>134x130 = 17420</td>
<td>1.031</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>MPI breakdown</th>
<th>Water</th>
<th>Land</th>
<th>r</th>
<th>Tile size</th>
<th>Tile aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x8 = 32</td>
<td>28</td>
<td>4</td>
<td>0.875</td>
<td>101x114 = 11514</td>
<td>1.129</td>
</tr>
<tr>
<td>4x9 = 36</td>
<td>30</td>
<td>6</td>
<td>0.833</td>
<td>101x102 = 10302</td>
<td>1.010</td>
</tr>
<tr>
<td>4x10 = 40</td>
<td>33</td>
<td>7</td>
<td>0.825</td>
<td>101x92 = 9292</td>
<td>1.098</td>
</tr>
<tr>
<td>5x10 = 50</td>
<td>40</td>
<td>10</td>
<td>0.800</td>
<td>82x92 = 7544</td>
<td>1.122</td>
</tr>
<tr>
<td>5x11 = 55</td>
<td>45</td>
<td>10</td>
<td>0.818</td>
<td>82x84 = 6888</td>
<td>1.024</td>
</tr>
<tr>
<td>5x12 = 60</td>
<td>48</td>
<td>12</td>
<td>0.800</td>
<td>82x77 = 6314</td>
<td>1.065</td>
</tr>
<tr>
<td>6x12 = 72</td>
<td>55</td>
<td>17</td>
<td>0.764</td>
<td>68x77 = 5236</td>
<td>1.132</td>
</tr>
<tr>
<td>6x13 = 78</td>
<td>58</td>
<td>20</td>
<td>0.744</td>
<td>68x71 = 4828</td>
<td>1.044</td>
</tr>
<tr>
<td>6x14 = 84</td>
<td>62</td>
<td>22</td>
<td>0.738</td>
<td>68x66 = 4488</td>
<td>1.030</td>
</tr>
<tr>
<td>6x15 = 90</td>
<td>65</td>
<td>25</td>
<td>0.722</td>
<td>68x62 = 4216</td>
<td>1.097</td>
</tr>
<tr>
<td>7x14 = 98</td>
<td>68</td>
<td>30</td>
<td>0.694</td>
<td>59x66 = 3894</td>
<td>1.119</td>
</tr>
<tr>
<td>7x15 = 105</td>
<td>71</td>
<td>34</td>
<td>0.676</td>
<td>59x62 = 3658</td>
<td>1.051</td>
</tr>
<tr>
<td>7x16 = 112</td>
<td>79</td>
<td>33</td>
<td>0.705</td>
<td>59x58 = 3245</td>
<td>1.073</td>
</tr>
<tr>
<td>7x17 = 119</td>
<td>80</td>
<td>39</td>
<td>0.672</td>
<td>59x52 = 2968</td>
<td>1.135</td>
</tr>
<tr>
<td>7x18 = 126</td>
<td>85</td>
<td>41</td>
<td>0.675</td>
<td>59x50 = 2600</td>
<td>1.087</td>
</tr>
<tr>
<td>8x16 = 128</td>
<td>89</td>
<td>39</td>
<td>0.695</td>
<td>52x58 = 3016</td>
<td>1.115</td>
</tr>
<tr>
<td>8x17 = 136</td>
<td>93</td>
<td>43</td>
<td>0.684</td>
<td>52x55 = 2860</td>
<td>1.058</td>
</tr>
<tr>
<td>8x18 = 144</td>
<td>95</td>
<td>49</td>
<td>0.660</td>
<td>52x52 = 2704</td>
<td>1.000</td>
</tr>
<tr>
<td>8x20 = 160</td>
<td>104</td>
<td>56</td>
<td>0.650</td>
<td>52x47 = 2444</td>
<td>1.106</td>
</tr>
<tr>
<td>9x18 = 162</td>
<td>108</td>
<td>54</td>
<td>0.667</td>
<td>46x52 = 2392</td>
<td>1.130</td>
</tr>
<tr>
<td>9x19 = 171</td>
<td>111</td>
<td>60</td>
<td>0.649</td>
<td>46x50 = 2300</td>
<td>1.087</td>
</tr>
<tr>
<td>9x20 = 180</td>
<td>115</td>
<td>65</td>
<td>0.639</td>
<td>46x47 = 2162</td>
<td>1.022</td>
</tr>
<tr>
<td>9x21 = 189</td>
<td>122</td>
<td>67</td>
<td>0.646</td>
<td>46x45 = 2070</td>
<td>1.022</td>
</tr>
<tr>
<td>10x20 = 200</td>
<td>123</td>
<td>77</td>
<td>0.615</td>
<td>42x47 = 1974</td>
<td>1.119</td>
</tr>
<tr>
<td>9x22 = 198</td>
<td>126</td>
<td>72</td>
<td>0.636</td>
<td>46x43 = 1978</td>
<td>1.070</td>
</tr>
<tr>
<td>9x23 = 207</td>
<td>131</td>
<td>76</td>
<td>0.633</td>
<td>46x41 = 1886</td>
<td>1.122</td>
</tr>
<tr>
<td>10x21 = 210</td>
<td>132</td>
<td>78</td>
<td>0.629</td>
<td>42x45 = 1890</td>
<td>1.071</td>
</tr>
<tr>
<td>10x22 = 220</td>
<td>136</td>
<td>84</td>
<td>0.618</td>
<td>42x43 = 1806</td>
<td>1.024</td>
</tr>
<tr>
<td>10x23 = 230</td>
<td>140</td>
<td>90</td>
<td>0.609</td>
<td>42x41 = 1722</td>
<td>1.024</td>
</tr>
<tr>
<td>11x22 = 242</td>
<td>141</td>
<td>101</td>
<td>0.583</td>
<td>38x43 = 1634</td>
<td>1.132</td>
</tr>
<tr>
<td>11x23 = 253</td>
<td>143</td>
<td>110</td>
<td>0.565</td>
<td>38x41 = 1558</td>
<td>1.079</td>
</tr>
<tr>
<td>10x24 = 240</td>
<td>144</td>
<td>96</td>
<td>0.600</td>
<td>42x40 = 1680</td>
<td>1.050</td>
</tr>
<tr>
<td>11x24 = 264</td>
<td>147</td>
<td>117</td>
<td>0.557</td>
<td>38x40 = 1520</td>
<td>1.053</td>
</tr>
<tr>
<td>10x25 = 250</td>
<td>152</td>
<td>98</td>
<td>0.608</td>
<td>42x38 = 1596</td>
<td>1.105</td>
</tr>
<tr>
<td>11x25 = 275</td>
<td>156</td>
<td>119</td>
<td>0.567</td>
<td>38x38 = 1444</td>
<td>1.000</td>
</tr>
<tr>
<td>10x26 = 260</td>
<td>158</td>
<td>102</td>
<td>0.608</td>
<td>42x37 = 1554</td>
<td>1.135</td>
</tr>
<tr>
<td>11x26 = 286</td>
<td>161</td>
<td>125</td>
<td>0.563</td>
<td>38x37 = 1406</td>
<td>1.027</td>
</tr>
<tr>
<td>11x27 = 297</td>
<td>167</td>
<td>130</td>
<td>0.562</td>
<td>38x36 = 1368</td>
<td>1.056</td>
</tr>
<tr>
<td>12x25 = 300</td>
<td>168</td>
<td>132</td>
<td>0.560</td>
<td>35x38 = 1330</td>
<td>1.086</td>
</tr>
<tr>
<td>11x28 = 308</td>
<td>172</td>
<td>136</td>
<td>0.558</td>
<td>38x34 = 1292</td>
<td>1.118</td>
</tr>
<tr>
<td>12x26 = 312</td>
<td>179</td>
<td>133</td>
<td>0.574</td>
<td>35x37 = 1295</td>
<td>1.057</td>
</tr>
<tr>
<td>12x27 = 324</td>
<td>183</td>
<td>141</td>
<td>0.565</td>
<td>35x36 = 1260</td>
<td>1.029</td>
</tr>
<tr>
<td>12x28 = 336</td>
<td>190</td>
<td>146</td>
<td>0.565</td>
<td>35x34 = 1190</td>
<td>1.029</td>
</tr>
<tr>
<td>13x26 = 338</td>
<td>191</td>
<td>147</td>
<td>0.565</td>
<td>33x37 = 1221</td>
<td>1.121</td>
</tr>
<tr>
<td>12x29 = 348</td>
<td>196</td>
<td>152</td>
<td>0.563</td>
<td>35x33 = 1155</td>
<td>1.061</td>
</tr>
<tr>
<td>12x30 = 360</td>
<td>197</td>
<td>163</td>
<td>0.547</td>
<td>35x32 = 1120</td>
<td>1.094</td>
</tr>
<tr>
<td>13x27 = 351</td>
<td>199</td>
<td>152</td>
<td>0.567</td>
<td>33x36 = 1188</td>
<td>1.091</td>
</tr>
<tr>
<td>13x28 = 364</td>
<td>203</td>
<td>161</td>
<td>0.558</td>
<td>33x34 = 1122</td>
<td>1.030</td>
</tr>
<tr>
<td>12x31 = 372</td>
<td>207</td>
<td>165</td>
<td>0.556</td>
<td>35x31 = 1085</td>
<td>1.129</td>
</tr>
</tbody>
</table>

Continued on next page
Table 2 – continued from previous page

<table>
<thead>
<tr>
<th>MPI breakdown</th>
<th>Water</th>
<th>Land</th>
<th>r</th>
<th>Tile size</th>
<th>Tile aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>13x30 = 390</td>
<td>214</td>
<td>176</td>
<td>0.549</td>
<td>33x32 = 1056</td>
<td>1.031</td>
</tr>
<tr>
<td>14x28 = 392</td>
<td>217</td>
<td>175</td>
<td>0.554</td>
<td>31x34 = 1054</td>
<td>1.097</td>
</tr>
<tr>
<td>14x29 = 406</td>
<td>219</td>
<td>187</td>
<td>0.539</td>
<td>31x33 = 1023</td>
<td>1.065</td>
</tr>
<tr>
<td>13x31 = 403</td>
<td>223</td>
<td>180</td>
<td>0.553</td>
<td>33x31 = 1023</td>
<td>1.065</td>
</tr>
<tr>
<td>14x30 = 420</td>
<td>226</td>
<td>194</td>
<td>0.538</td>
<td>31x32 = 992</td>
<td>1.032</td>
</tr>
<tr>
<td>14x31 = 434</td>
<td>231</td>
<td>203</td>
<td>0.532</td>
<td>31x31 = 961</td>
<td>1.000</td>
</tr>
<tr>
<td>15x29 = 435</td>
<td>235</td>
<td>200</td>
<td>0.540</td>
<td>29x33 = 957</td>
<td>1.138</td>
</tr>
<tr>
<td>13x33 = 429</td>
<td>236</td>
<td>193</td>
<td>0.550</td>
<td>33x30 = 990</td>
<td>1.100</td>
</tr>
<tr>
<td>14x32 = 448</td>
<td>237</td>
<td>211</td>
<td>0.529</td>
<td>31x30 = 930</td>
<td>1.033</td>
</tr>
<tr>
<td>15x30 = 450</td>
<td>238</td>
<td>212</td>
<td>0.529</td>
<td>29x32 = 928</td>
<td>1.103</td>
</tr>
<tr>
<td>13x34 = 442</td>
<td>239</td>
<td>203</td>
<td>0.541</td>
<td>33x29 = 957</td>
<td>1.138</td>
</tr>
<tr>
<td>15x31 = 465</td>
<td>247</td>
<td>218</td>
<td>0.531</td>
<td>29x31 = 899</td>
<td>1.069</td>
</tr>
<tr>
<td>14x34 = 476</td>
<td>250</td>
<td>226</td>
<td>0.525</td>
<td>31x29 = 899</td>
<td>1.069</td>
</tr>
<tr>
<td>15x32 = 480</td>
<td>254</td>
<td>226</td>
<td>0.529</td>
<td>29x30 = 870</td>
<td>1.034</td>
</tr>
</tbody>
</table>

2.3.5 Adding Passive Tracers to SalishSea

This page provides instructions on how a configuration including tracers was built for the Salish Sea model and how to use it and modify it. More information on adding passive tracers can also be found [here](#).

Compile SalishSea_TRC

The configuration including the tracers is SalishSea_TRC. The NEMO ARCH files use the `XIOS_HOME` environment variable to find the XIOS-2 library you built above. `XIOS_HOME` must be an absolute path to your XIOS-2 clone directory. You can set `XIOS_HOME` on the command-line before the `makenemo` and `maketools` commands as shown below, or you can set and export the value of `XIOS_HOME` in your `$HOME/.bashrc` file.

```bash
cd NEMO-3.6-code/NEMOGCM/CONFIG
XIOS_HOME=/data/$USER/MEOPAR/XIOS-2/ ./makenemo -n SalishSea_TRC -m GCC_SALISH -j8
```

New MY_SRC Files

There are three extra modified fortran files to be found in MY_SRC (most of the other files are symlinked to the SalishSea and SMELT configuration).

The file `par_my_trc.F90` specifies that six passive tracers are to be used and will have the last index of `jpmyt1, jpmyt2, jpmyt3, jpmyt4, jpmyt5, jpmyt6` respectively.

The file `trcini_my_trc.F90` has been barely changed from the original (in `NEMO-3.6-code/NEMOGCM/NEMO/TOP_SRC/MY_TRC`). It now writes out the number of tracers and sets the tracer concentration to 1 below depth level 10, 15, 19, 22, 24, and 25 corresponding to depths 10, 15, 20, 30, 45, and 60 m respectively. If you wished to initialize the tracers differently, that would go here.

The file `trcsms_my_trc.F90` defines the chemistry/biology to be done to the tracers. It now dictates that the tracers decay when they are above depths 10, 15, 20, 30, 45, and 60 m respectively. For example, if a tracer was at concentration 1 in the ‘decay’ zone, after one model day, their concentration would be $-0.3679$ (1/e). The decay time for each tracer could be modified by changing the factor used to multiply the `trb` array on lines 73, 78, 83, 88, 93, and 98 in the file.

An old version of this file is available at `nemo-code` in `NEMOGCM/CONFIG/SalishSea_TRC/MY_SRC`. Here we just specify the value of each tracer to be 1 in the grid cell we think contains the sewage outfall for each of Clover,
Macaulay, Iona, Nanaimo and Campbell River outfalls. Note that cells are defined by lats and lons. Specifying grid locations plays havoc when running mpi.

**New Namelist Files**

Two new namelist files are included: `namelist_top_ref` and `namelist_top_cfg`. The main `namelist_ref` and `namelist_cfg` should be copied or symlinked from an appropriate SalishSea namelist. `namelist_top_ref` defines the output names of the tracer. You could also add tracer damping using this namelist. Also change this file to use a restart file for the tracers.

**New IODEF.xml**

Writing out of the tracers goes into its own netcdf4 file with key prtc. The example given writes out 1h data.

**Grid/Input Files**

Use appropriate files from the SalishSea configuration.

### 2.3.6 Output Server Configuration

This section describes how to control the output files from NEMO. We will discuss how to output certain fields at a specific grid point. First, some overview on the NEMO output configuration file `iodef.xml` is provided.

**iodef.xml**

NEMO’s output configuration is controlled by a file called `iodef.xml`. It is found in the `SS-run-sets/SalishSea/` directory along with all of the namelists. This file allows the user to specify which fields should be outputted, how often, and at which grid points. It is written in an xml language and the details can be found in section 11.2 of the NEMO documentation.

Here is an example snippet from the `<file definition>` segment of the default `iodef.xml` file.

```
<group id="4h" output_freq="14400" output_level="10" enabled=".true." >
  <field id="4h_grid_T" name="auto">
    <field ref="ssh" name="sossheig"/>
    <field ref="toce" name="votemper"/>
    <field ref="soce" name="vosaline"/>
    <field ref="rain" name="rain_rate"/>
    <field ref="snowpre" name="snow_rate"/>
  </field>
  <field id="4h_grid_U" name="auto">
    <field ref="uoce" name="vozocrtx"/>
    <field ref="utau" name="u_wind_stress"/>
  </field>
  <field id="4h_grid_V" name="auto">
    <field ref="voce" name="vomecrty"/>
    <field ref="vtau" name="v_wind_stress"/>
  </field>
</group>
```

This part of the code defines a group of variables that will be saved every 4 hours. A few notes on the first line:
• output_freq="14400" specifies that this group of variables will be saved every 4 hours (14400 seconds).
• enabled=".true." tells NEMO that this group of variables should be saved. If this were set to enabled=".false." then NEMO would not save this group.

In the next set of lines, we define several files that will contain the output.

• The first file is tagged id="4h_grid_T" and contains the T fields listed.
• The second file is tagged id="4h_grid_U" and contains the U fields listed.
• The third file is tagged id="4h_grid_V" and contains the V fields listed.
• name="auto" means that the file name is automatically generated based on the id tags.

In order to learn how to output at a specific grid point we need to visit the <grid definition> segment of the iodef.xml file. Here is an example snippet:

```
<grid id="grid_T" description="grid T">
  <!-- Eq section -->
  <zoom id="EqT" ibegin="1" jbegin="0000" ni="0000" nj="1" />
  <!-- TAO -->
  <!-- 137e -->
  <zoom id="2n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  <zoom id="5n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  <zoom id="8n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  ...
</grid>
```

This part of the code defines a tag for the T grid and highlights the “zoom” feature. The zoom lines in the code create “zoom” tags that can be used when defining groups as outlined in the above section.

• id="2n137eT" defines the zoom_ref tag to be used later.
• ibegin="0000" states that the zoom tag should begin at grid point i=0.
• jbegin="0000" states that the zoom tag should begin at grid point j=0.
• ni="1" states that the zoom tag should span one grid point in the x direction.
• nj="1" states that the zoom tag should span one grid point in the y direction.

There are many predefined zoom tags in this section of the code. However, they all seem to begin at (i,j)=(0,0). In the next sections we will work on adding our own.

These zoom tags are defined on the T grid. You could also define zoom tags on the U or V grid by editing those sections of the grid definition.

**Outputting a specified grid point**

In order to output at a specified point we should first define a new zoom tag. We will modify the above section of code to include a new zoom tag.

```
<grid id="grid_T" description="grid T">
  <!-- Eq section -->
  <zoom id="EqT" ibegin="1" jbegin="0000" ni="0000" nj="1" />
  <!-- TAO -->
  <!-- 137e -->
  <zoom id="2n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  <zoom id="5n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  <zoom id="8n137eT" ibegin="0000" jbegin="0000" ni="1" nj="1" />
  ...
</grid>
```

(continues on next page)
We have added a zoom tag with id="PointAtkinson" at the grid point (329,469).

Next, we will go back to the <file definition> segment to define a new group of variables to be saved at this grid point.

```xml
<group id="1h_freq" output_freq="3600" output_level="10" enabled=".true." >
  <file id="1h_PointAtkinson" name="1h_PointAtkinson" enabled=".true." description="Point Atkinson 1h outputs">
    <group id="1h_PointAtkinson" zoom_ref="PointAtkinson" >
      <field ref="ssh" name="sossheig"/>
      <field ref="toce" name="votemper"/>
      <field ref="soce" name="vosaline"/>
      <field ref="rain" name="rain_rate"/>
      <field ref="snowpre" name="snow_rate"/>
      <field ref="uoce" name="vozocrtx"/>
      <field ref="utau" name="u_wind_stress"/>
      <field ref="voce" name="vomecrty"/>
      <field ref="vtau" name="v_wind_stress"/>
    </group>
  </file>
</group>
```

Here we have added a group of variables tagged as “1h_freq” that will be saved every hour.

- The second line defines a file for saving the Point Atkinson data. The file will be called 1h_PointAtkinson.nc.
- The third line defines a group of variables tagged as “1h_PointAtkinson”. These variables will be taken at the grid defined by zoom tag zoom_ref="PointAtkinson". Since this group of variables is within the “1h_freq” group they will also be saved every hour.
- The rest of the lines define the fields that will be saved.
- Other files and groups can be added to the “1h_freq” group as is done in iodef_freq.xml in SS-run-sets/SalishSea/.

### Storm Surge Outputs

The file iodef.xml has been set up to give one hour outputs at known storm surge locations. The storm surge locations are outlined in the *Storm Surges* doc. The “enabled” attributes for these files and group must be set to true in order to produce the new output files. After NEMO is run, the output should include four new files 1h_PointAtkinson.nc, 1h_Victoria.nc, 1h_PatriciaBay.nc, and 1h_CampbellRiver.nc.

### Other Notes

Users can also change how the output is calculated (instantaneous vs. average fields) with the “operation” attribute. This feature is outlined in the NEMO documentation section 11.2.
2.3.7 Simplified Model Domain - 2D

This page provides instructions on how to build a simplified two-dimensional version of the Salish Sea model. The two-dimensional model is based on a simplified bathymetry profile along the domain's thalweg. A simplified model allows for quick sensitivity tests of mixing parametrizations and resolution.

New NEMO Configuration

The first step is to create a new NEMO configuration. Here we will define parameters such as number of grid points and resolution. These instructions explain how to create a new configuration called SalishSea2D.

1. In `NEMO-code/NEMOGCM/CONFIG`, create directory for the new configuration.

```bash
cd NEMO-code/NEMOGCM/CONFIG
mkdir SalishSea2D
```

2. Create a `cpp_SalishSea2D.fcm` file and place it in the configuration directory. Add a key for the new configuration and any other keys you require. For example,

```bash
cd SalishSea2D
ls cpp_SalishSea2D.fcm
bld::tool::fppkeys key_bdy key_vectopt_loop key_dynspg_ts key_vv1
key_distant key_mpp_mpi key_netcdf4 key_nosignedzero key_traldf_c2d key_dynldf_c3d
key_tide key_zdfgls key_iomput key_salishsea2D
```

3. Add configuration source files to `SalishSea2D/MY_SRC`. You need at least two modified source files: `par_oce.F90` and `par_SalishSea2D.h90`. In `par_oce.F90`, add a case for dealing with the new configuration key:

```fortran
#elif defined key_salishsea2D
!!---------------------------------------------------------------------
!! 'key_salishsea2D': Strait of Georgia: 2D
!!---------------------------------------------------------------------
# include "par_SalishSea2D.h90"
```

4. Specify choices in domain size, grid and resolution by editing `par_SalishSea2D.h90`. For example, these choices will set up a simplified grid 1100 by 10 by 40 grid points with 500 m resolution. See the NEMO documentation for details on parameter choices.

```fortran
jpidta = 1100, &  ! first horizontal dimension > or = to jpi
jpjdta = 10, &  ! second > or = to jpj
jpkdta = 40, &  ! number of levels > or = to jpk

jphgr_msh = 2 &  ! type of horizontal mesh
   ! = 2 f-plane with regular grid-spacing
ppgphi0 = 0.0_wp, &  ! latitude for the Coriolis or Beta parameter (jphgr_msh = 2 or 3)
ppe1_m = 500, &  ! zonal grid-spacing (meters)
ppe2_m = 500 &  ! meridional grid-spacing (meters)
```

5. Add `SalishSea2D OPA_SRC` to the last line of `NEMO-code/NEMOGCM/CONFIG/cfg.txt`

6. Try compiling with this new configuration. For example, on salish
Bathymetry

A simplified 2D bathymetry was created by smoothing the bathymetry along the thalweg. See Generate_2D_bathy.ipynb

Initial Conditions

Initial conditions for temperature and salinity were taken from model 2003 spin-up. The velocities are initialized to zero values. Winter and summer stratifications were created using Generate_2D_T+S.ipynb. Note that a test run with basic namelists was used to create the 2D mesh_mask.nc. This file was used to ensure the initial temperature and salinity covered the full 2D domain.

Tides

Tidal elevations are based on the 3D model tides, averaged across the mouth of the Strait of Juan de Fuca. Tidal currents for U are also based on 3D tidal currents, averaged across the boundary. Tidal currents for V are ignored. Some adjustments to the tidal currents are made to enforce rapid velocities over the sill since the horizontal constriction is not present in the 2D domain. See Generate_2D_Tides.ipynb for generation of tide forcing files.

River

A simple representation of the Fraser River was added to replace mixed stratification. A constant flow rate was used. Details in Generate_rivers_forcing.ipynb.

Namelist

The namelists need to be modified to reflect the new forcing files and boundary conditions. See /data/nsoontie/MEOPAR/2Ddomain/namelists.

Changes in Resolution

To be added

2.3.8 Spin-up Runs

This section documents the workflows and results of a series of runs that were done to spin up the model to produce restart files that are representative of state of the Salish Sea over a 12 month period. The spin-up runs span the period from 12-Sep-2002 to 31-Dec-2003, with the last 12 months of that period providing the spun-up restart files. Those restart files can be used as initial conditions for research runs in other years.

The spin-up run results are stored in /ocean/dlatorne/MEOPAR/SalishSea/results/spin-up/. Directory names there indicate the date range of the run results that they contain; e.g. 1jan30jan/ contains the results of the 30 day 1-Jan-2003 to 30-Jan-2003 run. The restart files contained in the run directories have
the name pattern SalishSea_ddddddd_restart.nc, where dddddddd is the number of time steps after 00:00 on 16-Sep-2002 (left padded with zeros). The model time step used for spin-up was 50 s. So, SalishSea_00069120_restart.nc would contain a snapshot of the model state at 00:00 on 26-Oct-2002 (69120 time-steps / (86400 s/day / 50 s/time-step) = 40 days).

Spin-up Sections

16-Sep-2002 to 21-Sep-2002

The first 6 days of spin-up were run on salish 1 day at a time. The lateral turbulent viscosity (NEMO namdyn_ldf namelist variable rn_ahm_0_lap) was initially set to 80 m²/s and reduced by 5 m²/s each day to 55 m²/s on day 6. That was done to stabilize the model as the initial boundary condition values (especially deep salinity) propagated through the moderate, uniform initial stratification that was set for the entire domain.

Note: These runs have nn_date0 set to the day for which the calculations are being done, not the day on which nn_it000 is 1 (see Value and Use of nn_date0). So, the tidal forcing is inconsistent. This does not invalidate these runs as part of spin-up because they are early in the spin-up, and the objective of spin-up is temperature, salinity, and velocity fields, not tides.

22-Sep-2002

After the nn_date0 issue was discovered 22-Sep was rerun with the following changes:

1. Viscosity was set to 55 m²/s
2. nn_date0 was set to 20020916 so that tides were consistent with time in the spin-up. That means that there is a jump in the tidal forcing at the beginning of this run.


The next days of spin-up were run with the lateral turbulent viscosity (NEMO namdyn_ldf namelist variable rn_ahm_0_lap) set to as low as possible but high enough that the code did not blow up. The maximum value used was 55 m²/s. This was done to try to avoid too much Juan de Fuca deep water from reaching the bottom of the Strait of Georgia as happened in a 10d spin-up run started on 23-Sep. The fresh water in San Juan needs to be flushed to allow enough pre-mixing there. Harmonics for the M2 and K1 tides were calculated over the entire duration of this run with a resolution of 9 time steps (450 seconds). The results during this period also include sea surface height at selected locations for analysis of storm surges, and sea surface height and profiles of temperature, salinity, and u and v velocity components at 6 points along the thalweg and at a location in the Fraser River plume.

26-Oct-2002

Lack of stability in the previous runs and the continuous need to increase viscosity lead to an investigation of stability. It was determined that the viscosity was too high (currents in the SoG were smaller than observed), and the time step was too large for a vertical CFL condition. Reducing the time step (and reducing the viscosity) led to a stable run of 26-Oct and onwards.

2.3. Salish Sea NEMO
31-Oct-2002 –

Northern boundary was opened using tides (all 8 constituents) and seasonal T&S (based on Thomson and Huggett paper).

20-Dec-2002 –

Error in phase of northern boundary constituents was found and corrected (all 8 constituents).

10-Feb-2003 –

Error in the barotropic velocities at the western boundary was found and corrected. The length of the arrays needs to be carefully matched or ssh contaminates the velocities.

28-Sep-2003 –

Our northern cross-section is bigger than the cross-section where Thomson & Huggett measured their currents. Thus our flux in is too large. We corrected the K1 tides by the estimated amount 25% and the M2 tides by twice this to better agree with observations.

The Runs

<table>
<thead>
<tr>
<th>Dates</th>
<th>dt s</th>
<th>Viscosity m²/s</th>
<th>EVD m²/s</th>
<th>Status</th>
<th>Platform</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 23</td>
<td>50</td>
<td>50</td>
<td>100</td>
<td>blew up off Pender</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 23</td>
<td>50</td>
<td>55</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 24</td>
<td>50</td>
<td>50</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 25</td>
<td>50</td>
<td>45</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 26</td>
<td>50</td>
<td>40</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 27</td>
<td>50</td>
<td>35</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 28</td>
<td>50</td>
<td>30</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 29</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 30</td>
<td>50</td>
<td>20</td>
<td>100</td>
<td>blew up off Stuart</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 30</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>blew up off Pender</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Sep 30</td>
<td>50</td>
<td>30</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 1-2</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 1</td>
<td>50</td>
<td>30</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 2-3</td>
<td>50</td>
<td>30</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 4-5</td>
<td>50</td>
<td>30</td>
<td>100</td>
<td>blew up off Stuart</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 4-5</td>
<td>50</td>
<td>35</td>
<td>100</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 6-9</td>
<td>50</td>
<td>35</td>
<td>100</td>
<td>blew up BP Mouth</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Oct 6-7</td>
<td>50</td>
<td>40</td>
<td>100</td>
<td>complete</td>
<td>Jasper</td>
<td>first 2 days of 4</td>
</tr>
<tr>
<td>Oct 8-9</td>
<td>50</td>
<td>40</td>
<td>100</td>
<td>blew up NaN</td>
<td>Jasper</td>
<td>last 2 days of 4</td>
</tr>
<tr>
<td>Oct 8-9</td>
<td>50</td>
<td>45</td>
<td>100</td>
<td>complete</td>
<td>Jasper</td>
<td>first 2 days of 4</td>
</tr>
<tr>
<td>Oct 10-11</td>
<td>50</td>
<td>45</td>
<td>100</td>
<td>blew up off Pender</td>
<td>Jasper</td>
<td>last 2 days of 4</td>
</tr>
<tr>
<td>Oct 10-13</td>
<td>50</td>
<td>50</td>
<td>100</td>
<td>blew up off San J.</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Oct 10-13</td>
<td>50</td>
<td>50</td>
<td>150</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Oct 14-17</td>
<td>50</td>
<td>50</td>
<td>150</td>
<td>complete</td>
<td>Jasper</td>
<td>first 4 days of 10</td>
</tr>
<tr>
<td>Oct 18-23</td>
<td>50</td>
<td>50</td>
<td>150</td>
<td>blew up off Pender</td>
<td>Jasper</td>
<td>last 6 days of 10</td>
</tr>
<tr>
<td>Oct 18-25</td>
<td>50</td>
<td>50</td>
<td>200</td>
<td>complete</td>
<td>Jasper</td>
<td>first 8 days of 10</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 3 – continued from previous page

<table>
<thead>
<tr>
<th>Dates</th>
<th>dt s</th>
<th>Viscosity (m^2/s)</th>
<th>EVD (m^2/s)</th>
<th>Status</th>
<th>Platform</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 26-27</td>
<td>50</td>
<td>50</td>
<td>200</td>
<td>blew up</td>
<td>Jasper</td>
<td>last 2 days of 10</td>
</tr>
<tr>
<td>Oct 26-Nov4</td>
<td>50</td>
<td>55</td>
<td>200</td>
<td>blew up</td>
<td>Jasper</td>
<td>prevailed before prev.</td>
</tr>
<tr>
<td>Oct 26</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Salish</td>
<td></td>
</tr>
<tr>
<td>Oct 27-30</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Oct 31</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>opened North</td>
</tr>
<tr>
<td>Nov 1-4</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Nov 5-14</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Nov 15-19</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Nov 20-24</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>timed-out</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Nov 20-29</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Nov 30-Dec9</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Dec 10-19</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Dec 20-31</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>timed-out</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Dec 20-25</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>fixed North tides</td>
</tr>
<tr>
<td>Dec 26-31</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jan 1-5</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jan 6-10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jan 11-20</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>Jan 21-30</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>Jan 31-Feb9</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Feb 10-19</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>fixed West bt vels</td>
</tr>
<tr>
<td>Feb 20-Mar1</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Mar 2-11</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Mar 12-21</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Mar 22-31</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Apr 1-10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Apr 11-20</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Apr 21-30</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>May 1-10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>May 11-20</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>May 21-30</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>May 31-Jun9</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jun 10-19</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jun 20-29</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jun 30-Jul9</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jul 10-19</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jul 20-29</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Jul 30-Aug8</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>9-18 Aug</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>19-28 Aug</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Aug 29-Sep17</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>8-17 Sep</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>18-27 Sep</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>28 Sep-7 Oct</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>flux corr N tides</td>
</tr>
<tr>
<td>8-17 Oct</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>18-27 Oct</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td></td>
</tr>
<tr>
<td>Oct 28-Nov6</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>7-16 Nov</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>17-26 Nov</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
</tbody>
</table>

Continued on next page
Table 3 – continued from previous page

<table>
<thead>
<tr>
<th>Dates</th>
<th>dt s</th>
<th>Viscosity $m^2/s$</th>
<th>EVD $m^2/s$</th>
<th>Status</th>
<th>Platform</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov27-Dec6</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>7-16 Dec</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>complete</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>17-26 Dec</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>finished</td>
<td>Jasper</td>
<td>5d restart files</td>
</tr>
<tr>
<td>27-31 Dec</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>queued</td>
<td>Jasper</td>
<td>5d restart file</td>
</tr>
</tbody>
</table>

Spin-up Run Workflows

Run Preparation and Queuing

These are the steps to prepare and queue a spin-up run on jasper.westgrid.ca:

1. If the CGRF atmospheric forcing files for the period of the run are not already in place on jasper, prepare them. Files for the day before the run starts and the day after it finishes are required so that interpolation of forcing values in NEMO works, for example:

```bash
ssh jasper
cd MEOPAR/CGRF/
salishsea get_cgrf 2002-10-03 -d 10
```

You will be prompted for a userid and password for the goapp.ocean.dal.ca rsync server. Those credentials can also be supplied in the command via the --user and --password options.

2. Create a YAML run description file for the run in the SS-run-sets/SalishSea/spin-up/ directory. That can be done by copying and renaming a previous run file. The name pattern for run description files is SalishSea.ddmmmddmmm.yaml, where ddmmmddmmm is the day and month of the first and last days of the run; e.g. SalishSea.23sep2oct.yaml. The 2nd ddmmm is omitted for 1 day long runs.

The values that must be set correctly in every new spin-up run description file are:

- initial conditions in the forcing stanza, which must be set to the path and file name of the restart file to use as initial conditions for the run, typically the last restart file from the previous spin-up run
- the namelist.time file name in the namelists stanza (see below)

Other namelist file names may also be used to set special conditions for the run. In general, the namelists from SS-run-sets/SalishSea/ are used unless there are changes for a particular spin-up run. Special condition namelists are created and committed to version control in the spin-up/ directory.

A typical spin-up run description file looks like:

```yaml
# salishsea command processor run description for Salish Sea case
#
# Spin-up run
#
# Salish Sea full domain with:
# Smoothed JdF mouth bathymetry
# S4-1 uniform initial T and S, depth corrected
# Open, unstructured western boundary across Strait of Juan de Fuca
# Tidal forcing
# Masson model, depth corrected, T, S, U & V
# Monthly climatology river run-off forcing, all rivers
# Atmospheric forcing from CGRF dataset
# Atmospheric pressure as inverse sea surface height effect enabled

config_name: SalishSea
```

(continues on next page)
paths:
  - NEMO-code: ../../../NEMO-code/
  - forcing: ../../../NEMO-forcing/
  - runs directory: ../../../SalishSea/

grid:
  # If relative, paths are taken from forcing path above
  coordinates: coordinates_seagrid_SalishSea.nc
  bathymetry: bathy_meter_SalishSea2.nc

forcing:
  # If relative, paths are taken from forcing path above
  atmospheric: ../CGRF/NEMO-atmos/
  initial conditions: ../../../SalishSea/results/spin-up/22sep/SalishSea_00012096_
  →restart.nc
  open boundaries: open_boundaries/
  rivers: rivers/

namelists:
- namelist.time.23sep24sep
- ../namelist.domain
- ../namelist.surface
- ../namelist.lateral
- ../namelist.bottom
- ../namelist.tracers
- namelist.dynamics.nu55evd100  # 23sep24sep run only
- ../namelist.compute.6x14

3. Create a namelist.time file for the run in the SS-run-sets/SalishSea/spin-up/ directory. That can be done by copying and renaming a previous run file. The name pattern for run description files is namelist.time.ddmmmddmmm, where ddmmmdddmmm is the day and month of the first and last days of the run; e.g. namelist.time.23sep2oct. The 2nd ddmmm is omitted for 1 day long runs.

The values that must be set correctly in every new spin-up run namelist.time file are:

- **nn_it000**: the first time step for the run, typically 1 greater than the final time step of the previous run that is included in the name of the restart in the run description file
- **nn_itend**: the final time step for the run, nn_it000 + days * 8640 - 1, where days is the run duration in days
- **nn_date0**: the date when nn_it000 was 1; i.e. 20021026. Note that this convention changes in NEMO 3.6
- **nit000_han**: the first time step for tidal harmonic analysis, typically the same value as nn_it000
- **nitend_han**: the final time step for tidal harmonic analysis, typically the same value as nn_itend

Also ensure that ln_rstart is set to .true.

A typical namelist.time file looks like:

```fortran
!! Run timing control
!!
!! *Note*: The time step is set in the &namdom namelist in the namelist.domain
!! file.
!!
!! namrun ! Parameters of the run
```

(continues on next page)
4. Create any special condition namelist files and ensure that they are correctly included in the nameslists stanza of the run description file.

5. Choose or create an iodef.xml file for the run. The name pattern for iodef.xml files is iodef.nnt.xml, where nn is the frequency of output of the _grid_[TUV].nc files, and t is the output interval; e.g. iodef.1d.xml.

6. Create or update a TORQUE batch job file for the run. The name pattern for batch job files is SalishSea.nnd.pbs, where nn is the duration of the run in days; e.g. SalishSea.10d.pbs.

The values that must be set correctly for every job are:

- The `ddmmmdmnnmm` part in the following lines:
  - `#PBS -N: the job name`
  - `#PBS -o: the path and name for stdout from the job`
  - `#PBS -e: the path and name for stderr from the job`
  - `RESULTS_DIR=": the path and name of the results directory where the run results are to be gathered`
- The `walltime` limit; e.g.
Runs typically required about 80 minutes of compute time per model-day if they allocated to the fast (X5675) nodes on jasper. However, if a run is allocated to the slow (L5420) nodes on jasper it can take nearly 180 minutes of compute time per model-day. It appears to be more advantageous to request that only fast nodes be used rather than requesting sufficient run walltime to allow runs to complete on the slow nodes. The directive to request only fast nodes is:

Wall time values that have been found to be adequate are 3h for a 2d run, 8h for a 5d run, and 16h for a 10d run.

You should also set your email address in the `#PBS -M` line so that job start, end, and abort messages are emailed to you.

A typical TORQUE batch job file looks like:

```
#!/bin/bash

#PBS -N SpinUp26oct4nov
#PBS -S /bin/bash
#PBS -l procs=84
#PBS -l pmem=2gb
#PBS -l walltime=16:00:00
#PBS -l feature=X5675
#PBS -m bea
#PBS -M sallen@eos.ubc.ca
#PBS -o ../SpinUp26oct4nov/stdout
#PBS -e ../SpinUp26oct4nov/stderr

RESULTS_DIR=../SpinUp26oct4nov

cd $PBS_O_WORKDIR

echo working dir: $(pwd)

module load application/python/2.7.3
module load library/netcdf/4.1.3
module load library/szip/2.1

echo "Starting run at $(date)"
mkdir -p $RESULTS_DIR
mpirun ./nemo.exe

echo "Ended run at $(date)"

echo "Results gathering started at $(date)"
$PBS_O_HOME/.local/bin/salishsea gather --no-compress SalishSea*.yaml $RESULTS_DIR
chmod go+rx $RESULTS_DIR
chmod go+r $RESULTS_DIR/*

echo "Results gathering ended at $(date)"

echo "Scheduling cleanup of run directory"
echo rmdir $PBS_O_WORKDIR > /tmp/SpinUp26oct4nov_cleanup
at now + 1 minutes -f /tmp/SpinUp26oct4nov_cleanup 2>&1
```

7. Commit and push the run set file changes for each run prior to queuing the run so that there is a clear record of runs in the `SS-run-sets` repo. Don’t forget to add any files created for a run to the repo.
8. Prepare the run, copy the TORQUE batch job file to the run directory, go to the run directory, and submit the job to the scheduler:

```
salishsea prepare SalishSea.23sep24sep.yaml iodef.1d.xml
cp SalishSea.2d.pbs ../../../SalishSea/bb1357d6-8c6e-11e3-bdd0-0025902b0cdc
pushd ../../../SalishSea/bb1357d6-8c6e-11e3-bdd0-0025902b0cdc
qsub SalishSea.2d.pbs
```

2.3.9 NEMO and XIOS Code Repositories Maintenance

This section documents the maintenance processes for the NEMO and XIOS code repositories that are based on upstream Subversion repositories.

Researchers generally don’t need to be concerned with the docs below. They can follow the Quick Start Guide instructions for the system they are working on to clone the Mercurial code repositories they need from Bitbucket. The docs below are about how those repos are created and kept in sync with the upstream Subversion repos.

NEMO-3.6 Code Repo Maintenance

Set-up

The /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror repository is an svn checkout of http://forge.ipsl.jussieu.fr/nemo/svn/branches/2012/dev_v3_4_STABLE_2012 and also a read-only Mercurial repository. It was initialized with:

```
$ cd /ocean/sallen/hg_repos
$ svn co -r 5072 http://forge.ipsl.jussieu.fr/nemo/svn/branches/2015/nemo_v3_6_STABLE_˓
   ~NEMO-3.6-hg-mirror
$ hg init NEMO-3.6-hg-mirror
$ cd NEMO-3.6-hg-mirror
$ cat > .hgignore
  syntax: glob
  .svn
  ctrl-d
$ hg add
$ hg ci -m "Initialize NEMO-3.6 svn mirror at r5072 of ^/trunk."
```

svn v1.8.8 was used on skookum for the svn part of the initialization.

Doug maintains an NEMO-3.6-mirror-merge repo on his laptop. That repo is used to merge changes from the upstream Subversion repository that are brought in via the /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror repo, and from the Salish Sea team NEMO-3.6-code repo on Bitbucket. The NEMO-3.6-mirror-merge repo was created by cloning the /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror repo:

```
hg clone ssh://skookum.eos.ubc.ca//ocean/sallen/hg_repos/NEMO-3.6-code-hg-mirror NEMO-˓
   _3.6-code-mirror-merge
```

and setting the paths in its .hg/hgrc to:

```
[paths]
bb = ssh://hg@bitbucket.org/salishsea/nemo-3.6-code
default-push = ssh://hg@bitbucket.org/salishsea/nemo-3.6-code
mirror = ssh://skookum.eos.ubc.ca//ocean/sallen/hg_repos/NEMO-3.6-code-hg-mirror
```
Those paths mean that the repo for `hg pull` and `hg incoming` commands must be specified explicitly. The `bb` and mirror paths are provided to facilitate pulling from nemo-3.6-code on Bitbucket and `/ocean/sallen/hg_repos/NEMO-3.6-code-hg-mirror`, respectively. `hg push` and `hg outgoing` commands will act on the nemo-3.6-code repo on Bitbucket, unless otherwise specified.

The Salish Sea team NEMO-3.6-code repo on Bitbucket was created via the Bitbucket web interface and populated there by an `hg push` from Doug's NEMO-3.6-mirror-merge repo.

A working copy was then created by cloning the Salish Sea team NEMO-3.6-code repo on Bitbucket as NEMO-3.6-code. The workflow sections below explain how these 4 repo clones are used to pull changes from upstream and merge them with changes that Salish Sea team members push to Bitbucket.

![Fig. 1: NEMO-3.6 code repositories and workflow to update and merge SVN and local changes](image)

**Workflow to Pull Changes from NEMO-3.6 svn Repo**

The workflow to pull changes from the master NEMO svn repo and commit them to our NEMO-3.6-hg-mirror repo is somewhat automated by the Marlin – Salish Sea NEMO svn-hg Maintenance Tool.

1. Review the upstream changes in the source browser at [http://forge.ipsl.jussieu.fr/nemo/svn/branches/2012/dev_v3_4_STABLE_2012](http://forge.ipsl.jussieu.fr/nemo/svn/branches/2012/dev_v3_4_STABLE_2012) to select a range of changes to be pulled into our NEMO-3.6-hg-mirror repo.

   **Note**: Pay special attention to changes in the OPA_SRC/ and TOP_SRC/ trees that involve files that have been copied into NEMOGCM/CONFIG/SalishSea/MY_SRC/ or team members’ MY_SRC/ directories. Those files must be manually merged with their MY_SRC/ counterparts.

2. Working on `salish` in the `/ocean/sallen/hg_repos/NEMO-3.6-hg-mirror` repo with an activated virtualenv in which `marlin` is installed:

   ```
   $ ssh salish
   $ workon marlin
   (marlin)$ cd /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror
   ```

3. Use `marlin incoming` information about the next SVN revision that will be pulled from upstream and confirm that it is the expected revision:

   ```
   (marlin)$ marlin incoming
   r6482 2016-04-19 09:59:19 UTC
   #1687 - Add a consistency check for the setting of ln_useCT and nn_eos
   ```

   The `--limit` option can be used to see more incoming revisions; see `marlin help incoming` for details.

4. Use `marlin update` to update the working copy to the next upstream commit and commit the SVN update as a Mercurial changeset with the SVN commit message as the body of the Mercurial commit message and echo that message:

   ```
   (marlin)$ marlin update
   Update to svn r6482.
   #1687 - Add a consistency check for the setting of ln_useCT and nn_eos
   ```

   The `--to-rev` option can be used to apply a series of upstream updates, committing them to Mercurial one at a time; see `marlin help update` for details.
Workflow to Merge NEMO-3.6 svn Repo and Salish Sea Revisions

Merging changes from NEMO svn and the Salish Sea team NEMO-3.6-code repo on Bitbucket is done in a repo that is used for only that purpose. Doug does the merges on his laptop. The repo in which the merging is done was created by cloning the /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror repo as described in the Set-up section.

After the PullChangesFromNEMOsvn has been completed the workflow to merge those changes with Salish Sea MEOPAR project revisions is:

1. Pull and update recent changes from the Salish Sea team NEMO-3.6-code repo into NEMO-3.6-mirror-merge:
   ```bash
   cd NEMO-3.6-mirror-merge
   hg pull --update bb
   ```

2. Pull and update the changes from /ocean/sallen/hg_repos/NEMO-3.6-hg-mirror into NEMO-3.6-mirror-merge:
   ```bash
   hg pull mirror
   ```

3. Because the changesets pulled from the Salish Sea team NEMO-3.6-code repo are public a branch merge is necessary:
   ```bash
   hg merge
   hg commit -m"Merge svn updates."
   ```

4. Manually merge and commit changes that involve files that have been copied into NEMOGCM/CONFIG/SalishSea/MY_SRC/ or team members' MY_SRC/ directories. Those files are most likely to be in OPA_SRC/ and TOP_SRC/.

5. Push the result of the updates and merges to the Salish Sea team NEMO-3.6-code repo:
   ```bash
   hg push bb
   ```

   If other users have pushed changes to the Salish Sea team NEMO-3.6-code repo while merge conflicts were being handled `hg pull --rebase` can be used to bring in those changes and deal with any additional merge conflicts.

6. Notify team members of the upstream merge, especially if manual merges of MY_SRC/ files were required, so that they can manage merging changes into any untracked MY_SRC/ files they may have.

Local commits

We are maintaining several local commits to files that do not fit in MY_SRC/ directories. They are:

1. Import changeset 7751 from upstream NEMO: 900e1ce4066f2a64223b502167a9a15a6d7c7b2d
2. Make rebuild_nemo write compressed netCDF4 restart files: 28e2d29cd2a8288e475a16804837eee22ea342f2
3. Sort cfg.txt in makenemo, 9f0335b59c780d055b7466e5f8f4d41d4ab023aa
4. Escape ‘ls’ in Fmake_WORK.sh, 4979fc15317c1c899697c1b89d5a64d688917253

Some of these are fixed in NEMO upstream:

1. Escape ‘ls’ in Fmake_WORK.sh, Ticket 1895, Changeset 8527
XIOS-2 Code Repo Maintenance

Set-up

The `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` repository is an `svn` checkout of `http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk` at revision 1066 and also a read-only Mercurial repository. It was initialized with:

```
$ cd /ocean/sallen/hg_repos
$ svn co -r 1066 http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk XIOS-2-hg-mirror
$ hg init XIOS-2-hg-mirror
$ cd XIOS-2-hg-mirror
$ cat > .hgignore
  syntax: glob
  .svn
  ctrl-d
$ hg add
$ hg ci -m"Initialize XIOS-2 svn mirror at r1066 of ^/trunk."
```

`svn` v1.8.8 was used on skookum for the `svn` part of the initialization.

Doug maintains an XIOS-2-mirror-merge repo on his laptop. That repo is used to merge changes from the upstream Subversion repository that are brought in via the `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` repo, and from the Salish Sea team XIOS-2 repo on Bitbucket. The XIOS-2-mirror-merge repo was created by cloning the `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` repo:

```
hg clone ssh://skookum.eos.ubc.ca//ocean/sallen/hg_repos/XIOS-2-hg-mirror XIOS-2-mirror-merge
```

and setting the paths in its `.hg/hgrc` to:

```
[paths]
  bb = ssh://hg@bitbucket.org/salishsea/xios-2
  default-push = ssh://hg@bitbucket.org/salishsea/xios-2
  mirror = ssh://skookum.eos.ubc.ca//ocean/sallen/hg_repos/XIOS-2-hg-mirror
```

Those paths mean that the repo for `hg pull` and `hg incoming` commands must be specified explicitly. The `bb` and `mirror` paths are provided to facilitate pulling from xios-2 on Bitbucket and `/ocean/sallen/hg_repos/XIOS-2-hg-mirror`, respectively. `hg push` and `hg outgoing` commands will act on the xios-2 repo on Bitbucket, unless otherwise specified.

The Salish Sea team XIOS-2 repo on Bitbucket was created via the Bitbucket web interface and populated there by an `hg push` from Doug’s XIOS-2-mirror-merge repo.

A working copy was then created by cloning the Salish Sea team XIOS-2 repo on Bitbucket as XIOS-2.

Getting the Salish Sea team XIOS-2 repo on Bitbucket to a state in which team members can clone it and easily build XIOS is annoyingly fiddly. The difficulties largely stem from the fact that the `svn` checkout includes compressed tarballs in the `archive/` directory. On the first build, those tarballs are decompressed into `extern/` and `tools/FCM/`. Subsequent builds check to see if the tarballs have been decompressed, so the tarballs can’t be removed, only their compressed forms.

Here are the steps that were done in a working copy on `salish` cloned from the Salish Sea team XIOS-2 repo on Bitbucket:

- Tag XIOS-2r1066
- Add README.rst by copying and editing the one from the XIOS-1 repo
- Symlink the XIOS-ARCH/UBC-EOAS/arch-GCC_SALISH.* files into arch/
• Run `make_xios`

• `hg forget tools/archive/FCM.tar.gz tools/archive/blitz.tar.gz tools/archive/boost.tar.gz tools/archive/rapidxml.tar.gz`

• Add `tools/archive/*.tar.gz` to `.hgignore`

• `hg add tools/archive/*.tar` tools/FCM/ extern/blitz/ extern/boost/ extern/rapidxml/`

• `hg commit -m"Replace tools/archive compressed tarballs with their extracted contents."`

• `hg push`

• Clone the Salish Sea team XIOS-2 repo on Bitbucket on orcinus

• Confirm that XIOS-2 builds successfully

• Return to the working copy on salish

• Add the following lines to `.hgignore` (it is critical that this not be done earlier because it will prevent needed files from being pushed to Bitbucket):

```
.cache/
.void_file
Makefile
arch.*/
bin/
cfg/
config.fcm
done/
etc/libxios.cfg
extern/netcdf4
fcm_env.*/
flags/
inc/
lib/
obj/
ppsrc/
```

• `hg commit -m"Don't track build products files and directories."`

The workflow sections below explain how these 4 repo clones are used to pull changes from upstream and merge them with changes that Salish Sea team members push to Bitbucket.

Fig. 2: XIOS-2 code repositories and workflow to update and merge SVN and local changes

**Workflow to Pull Changes from XIOS-2 svn Repo**

The workflow to pull changes from the master XIOS svn repo and commit them to our XIOS-2-hg-mirror repo is somewhat automated by the Marlin – Salish Sea NEMO svn-hg Maintenance Tool.

1. Review the upstream changes in the source browser at http://forge.ipsl.jussieu.fr/ioserver/log/ to select a range of changes to be pulled into our XIOS-2-hg-mirror repo.

2. Working on salish in the `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` repo with an activated virtualenv in which `marlin` is installed:
$ ssh salish
$ workon marlin
(marlin)$ cd /ocean/sallen/hg_repos/XIOS-2-hg-mirror

3. Use `marlin incoming` information about the next SVN revision that will be pulled from upstream and confirm that it is the expected revision:

   (marlin)$ marlin incoming
   r1062 2017-02-23 17:32:17 UTC
   Bug fix in interpolation for cell boundaries generation.

   The `--limit` option can be used to see more incoming revisions; see `marlin help incoming` for details.

4. Use `marlin update` to update the working copy to the next upstream commit and commit the SVN update as a Mercurial changeset with the SVN commit message as the body of the Mercurial commit message and echo that message:

   (marlin)$ marlin update
   Update to svn r1062.
   Bug fix in interpolation for cell boundaries generation.

   The `--to-rev` option can be used to apply a series of upstream updates, committing them to Mercurial one at a time; see `marlin help update` for details.

**Workflow to Merge XIOS-2 svn Repo and Salish Sea Revisions**

Merging changes from NEMO `svn` and the Salish Sea team XIOS-2 `svn` repo on Bitbucket is done in a repo that is used for only that purpose. Doug does the merges on his laptop. The repo in which the merging is done was created by cloning the `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` repo as described in the `Set-up` section.

After the `PullChangesFromNEMOsvn` has been completed the workflow to merge those changes with Salish Sea MEOPAR project revisions is:

1. Pull and update recent changes from the Salish Sea team XIOS-2 repo on Bitbucket into XIOS-2-mirror-merge:

   cd XIOS-2-mirror-merge
   hg pull --update bb

2. Pull and update the changes from `/ocean/sallen/hg_repos/XIOS-2-hg-mirror` into XIOS-2-mirror-merge:

   hg pull mirror

3. Because the changesets pulled from the Salish Sea team XIOS-2 repo on Bitbucket are public a branch merge is necessary:

   hg merge
   hg commit -m"Merge svn updates."

4. Push the result of the updates and merges to the Salish Sea team XIOS-2 repo on Bitbucket:

   hg push bb
If other users have pushed changes to the Salish Sea team XIOS-2 repo on Bitbucket while merge conflicts were being handled `hg pull --rebase` can be used to bring in those changes and deal with any additional merge conflicts.

5. Notify team members of the upstream merge.

### 2.3.10 NEMO-3.1 and CONCEPTS-110 Reference Repos

These notes describe the NEMO-3.1 and CONCEPTS-110 reference repositories.

The construction of the NEMO-3.1 repo is described in **NEMO-3.1** below. The CONCEPTS-110 repo was constructed from the CODE.tar tarball received on 2-Oct-2013 from J-P Paquin.

Also included here is an *analysis of the differences* between the NEMO/ directory trees in the 2 repos that was done early in the Salish Sea MEOPAR project in Oct-2013.

#### NEMO-3.1

The following steps took the NEMO-3.1 repo from initialization to a checkout of https://forge.ipsl.jussieu.fr/nemo/svn/tags/nemo_v3_1 that could be built on jasper.westgrid.ca and used to successfully run the GYRE configuration:

- An `svn` checkout of the trunk of the modipsl framework was done from http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk. That yielded revision 2163. The repo state was tagged as modipsl-r2163.

**Note:** At this point the only directories in the modipsl tree that are populated at `doc/` and `util/`. The `bin/`, `config/`, `lib/`, `modeles/`, and `tmp/` directories are empty and therefore not included in the Mercurial repo.

- Added J-P Paquin’s build script and `util/AA_make.gdef` from 2-Oct-2013 CODE.tar tarball. The latter is named `AA_make.gdef_BIO`.
- Added definitions for `util/model` script to `util/mod.def` to get NEMO-3.1 based on those in that file in the 2-Oct-2013 CODE.tar tarball.
- Used the `modipsl/util/model` script to obtain the NEMO-3.1 code and other supporting files. The commands to do so were:

  ```bash
  cd modipsl/util
  ./model NEMO_31
  ```

  Added the resulting 1295 files (some of which are in `.svn/` directories).

- Added global makefile definitions for `jasper.westgrid.ca` and the BIO HPC cluster to `util/AA_make.gdef`. The source for the former was Paul Myers’ NEMO-3.1 installation on `jasper` in `/home/pmyers/NEMODRAK_3.1/DRAKKAR/modipsl/util/AA_make.gdef`. The latter came from `util/AA_make.gdef` in the 2-Oct-2013 CODE.tar tarball.

#### Build and Run NEMO-3.1

**Note:** These instructions are included for completeness. The Salish Sea MEOPAR project does not use pristine NEMO-3.1.
1. Clone the repository from Bitbucket and update it to the NEMO-3.1 tag state:

```
hg clone -u NEMO-3.1 ssh://hg@bitbucket.org/salishsea/nemo-code NEMO-code-3.1
cd NEMO-code-3.1
mkdir modipsl/bin modispl/lib modipsl/tmp
```

2. Set up a configuration. We’ll use GYRE as an example and assume that we are building and running on jasper:

```
cd modispl/util
../modeles/UTIL/fait_config GYRE
```

3. Edit `../config/GYRE/scripts/BB_make.ldef` to add the appropriate pre-processing prefix for the system you are working on (near the end of the file). For jasper that is:

```
#-Q- jasper prefix = -D
```

**Note:** If you are working on a system other than those that already have global makefile definitions in modipsl/util/AA_make.gdef you will need to add an appropriate block of definitions to that file.

4. Calculate compilation rules, options, and build dependencies so as to create modipsl/modelels/NEMO/WORK/AA_make (which is symlinked to modipsl/config/GYRE/scripts/BB_make):

```
cd ../modeles/NEMO
../UTIL/fait_AA_make
```

**Note:** `fait_AA_make` must be run from the modipsl/modelels/NEMO/ directory.

5. Remove any existing Makefiles and create new ones:

```
cd ../../util
./clr_make
./ins_make -t jasper
```

6. On jasper several modules must be loaded prior to compiling and linking:

```
module load compiler/intel/12.1
module load library/intelmpi/4.0.3.008
module load library/netcdf/4.1.3
module load library/szip/2.1
```

That only needs to be done once per login so you may wish to add those commands to your `~/.bashrc` file (See `bashrc Snippets`).

7. Compile and link the code:

```
cd ../config/GYRE
make clean
make
```

The results of a successful build are:

* a :file:`../../bin/opa` executable
* a :file:`../../lib/libioipsl.a` library
* a :file:`../../lib/oce/libopa.a` library
8. Run the model:

```
cd EXP00
./../../bin/opa
```

On jasper the above command is only appropriate for short test runs. Longer runs should be done using a TORQUE batch job script submitted via the `qsub` command.

**Diffs Analysis**

The image below shows the results of an analysis of the file-level differences between the NEMO/ directory trees in the NEMO-3.1 and CONCEPTS-110 repos.

- Green indicates directories in which all files are identical
- Red indicates directories in which 1 or more files are different
- Black shows the number of files in the directories and the difference in the number of files between NEMO-3.1 and CONCEPTS-110 when applicable
- The `OPA_SRC/TDE/` directory in purple is present only in the CONCEPTS-110 directory
C1D_SRC - no diffs 6 files

LIM_SRC_2 - 3.1 23 files -> CONCEPTS 25 files
LIM_SRC_3 - no diffs 30 files
NST_SRC - no diffs 10 files
OFF_SRC - no diffs 79 files
OPA_SRC - 3.1 232 files -> CONCEPTS 327 files Δ = 95
TOP_SRC - minor diffs in 1 file 116 files

OPA_SRC/ - 3.1 44 files -> CONCEPTS 71 files Δ = 27

BDY - 8 files
DIA - 3/10 files -> CONCEPTS 20 files Δ = 10
DOM - 11 files
DTA - 2 files
DYN - 3.1 24 files -> CONCEPTS 31 files Δ = 7
FLO - no diffs 6 files
IOM - 3.1 6 files -> CONCEPTS 7 files (8 files) Δ = 1
LDF - 3.1 17 files -> CONCEPTS 19 files, Δ = 2 Songer
OBC - " 16 " -> " 40 " Δ = 24
SBC - " 28 " -> " 32 " Δ = 12
SOL - 7 files
TRA - 3.1 25 files -> CONCEPTS 27 files Δ = 2
TRD - minor diffs in 1 file 9 files
ZDF - 3.1 12 files -> CONCEPTS 14 files Δ = 2
TDE - new dir 5 files

2.3. Salish Sea NEMO
The numbers of files in each directory were calculated with commands like:

```
find OPA_SRC/ZDF/*/ -prune ! -type d | wc -l
```

In the directories in which the number of files was the same checks for differences between files was done with commands like:

```
CONCEPTS=CONCEPTS-110/CONCEPTS110_WCSD_OW_NOBC_tide/modipsl/modeles/NEMO
cd NEMO-3.1/NEMO
for file in $(find OPA_SRC/ZDF/*/ -prune ! -type d);
do
diff $CONCEPTS/$file $file;
done
```

The information about the NEMO-3.1 and CONCEPTS-110 reference repos is included here because some code from CONCEPTS-110 is used in our model and CONCEPTS-110 is based on NEMO-3.1. Those repos are not used for model runs.

### 2.4 Salish Sea Model Results Server

#### 2.4.1 Model Results & Forcing Files Storage

The `/results/` file system on `skookum` houses the storage used for:

- the Nowcast Production Deployment
- the nowcast system run results
- the Salish Sea NEMO model spin-up run results
- results from the Environment Canada GEM 2.5km HRDPS operational model runs that are used to force the nowcast system runs, and results from the research deployment of that model that are periodically evaluated
- Fraser River run-off forcing files produced from data downloaded from the Environment Canada Wateroffice service that are used to force the nowcast system runs
- Neah Bay sea surface height forcing files produced from data downloaded from the NOAA water level observations and forecast service that are used to force the nowcast system runs

The `/results/` file system is organized as follows:

```
/results/
|-- forcing/
 | |-- atmospheric/
 | | `-- GEM2.5/
 | |   |-- Grib/
 | |     |   |-- 20140911/
 | |     |     |-- 00/
 | |     |     |-- 01/
 | |     `-- grib2
 | |       |-- ...  
 | | `-- grib2
 | `-- grib2
   |-- ...  
   | `-- 048/
   `-- ...  
```

(continues on next page)
The /results/lost+found/ directory is a filesystem maintenance directory used by Linux. Don’t worry about it.

### 2.4.2 Salish Sea Model Results

The Salish Sea NEMO Model results from “production” runs of the model are stored in the /results/SalishSea/ directory. The sub-directories there are:
• /results/SalishSea/forecast/ Results from the nowcast system daily forecast runs using NEMO-3.6 since 2016-10-15. Forecast for day + 1 based on restart file from nowcast run for day, same atmospheric and river run-off forcing, and updated western boundary sea surface height forcing.

Earliest daily results directory is /results/SalishSea/forecast/15oct16/. Most, but not all dates since then are available. forecast/ runs are secondary priority (below nowcast/) when the nowcast automation system has difficulties.

See Nowcast Model Results for details of the configuration and model parameter values changes over time.

• /results/SalishSea/forecast2/ Results from the nowcast system daily forecast2 runs using NEMO-3.6 since 2016-10-15. Forecast for day + 2 based on restart file from forecast run for day + 1, updated atmospheric, river run-off forcing, and western boundary sea surface height forcing.

Earliest daily results directory is /results/SalishSea/forecast2/15oct16/. Most, but not all dates since then are available. forecast2/ runs are lowest priority (below forecast/) when the nowcast automation system has difficulties.

See Nowcast Model Results for details of the configuration and model parameter values changes over time.

• /results/SalishSea/nowcast-blue/ Results from the nowcast system daily nowcast runs using NEMO-3.6 since 2016-10-15.

Earliest daily results directory is /results/SalishSea/nowcast/15oct16/.

See Nowcast Model Results for details of the configuration and model parameter values changes over time.

• /results/SalishSea/nowcast-green/ Results from the nowcast system daily nowcast green ocean runs.

Earliest daily results directory is /results/SalishSea/nowcast-green/05dec15/.

See Nowcast Model Results for details of the configuration and model parameter values changes over time.

2.4.3 Details of Configurations for Results

Nowcast Model Results

/results/SalishSea/nowcast-green/ holds the results from the daily nowcast green ocean runs.

From 12-Sep-2014 until 24-Aug-2017 these results were produced from a hindcast (201702). From the 25-Aug-2017 on, these results come from the daily nowcast, started on the 24-Aug-2017 hindcast result.

Note that hindcast runs were typically done 10 days at a time. The first day directory includes the configuration etc files. The last day includes restarts from day 5 and day 10.

These runs started 12-Sep-2014 using physical and biological initial conditions from the previous nowcast-green run at 11-Sep-2016.

• NEMO-3.6
• coordinates v201702 (expanded resolution for Fraser River)
• bathymetry v201702 (based on DFO, Cascadia and ABC, included datum and extended Fraser River)
• jetty with enhanced bottom friction over it
• R201702 rivers (daily Fraser, climatology else)
• river biology and temperature (all the same, based on Fraser)
• N36_AF tides
• TEOS-10 and reworked Juan de Fuca TEOS-10 boundary conditions
• SMELT biological model
• passive turbidity
• Hollingsworth + energy and enstrophy conserving
• 5-step substepping vertical advection
• Orlanski with sponge open boundaries on baroclinic T&S
• decreased horizontal diffusivity (1 m2/s) and viscosity (1.1 m2/s)
• horizontal diffusivity (so can increase values for sponge near boundaries)
• decreased bottom friction (7 cm bottom roughness)

Model Parameter Changes Over Time

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
<th>New Value</th>
<th>Change-set</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-Sep-2014</td>
<td>1st run results</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>28-Dec-2014</td>
<td>Updated Fraser turbidity tracer code and reset</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>25-Sep-2015</td>
<td>Increased vertical time stepping nn_traadv_tvd_zts</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>05-Feb-2017</td>
<td>Switched to LiveOcean boundary conditions at JdF</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>25-Aug-2017</td>
<td>Started running as nowcast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07-Oct-2017</td>
<td>Updated bathymetry to correct smoothing in Puget Sound, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-Dec-2017</td>
<td>New LiveOcean boundary condition processor with longer western boundary and NO3 from LiveOcean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18-Apr-2018</td>
<td>Reduced bSi and PON sinking rates</td>
<td>5.6e-5 m/2</td>
<td>e9a5bc834f46</td>
</tr>
<tr>
<td>25-Apr-2018</td>
<td>Change boundary PON and DON climatologies to be based on model-derived profiles</td>
<td></td>
<td>f253dfb8277b</td>
</tr>
<tr>
<td>28-Apr-2018</td>
<td>Changed phytoplankton growth temperature dependence parameter values</td>
<td></td>
<td>f253dfb8277b</td>
</tr>
<tr>
<td>19-Jun-2018</td>
<td>Apply Neah Bay ssh to north boundary as well as west boundary</td>
<td>6b89d1af1ae</td>
<td>c1939be496ba</td>
</tr>
<tr>
<td>16-Aug-2018</td>
<td>Reflect part of the PON/DON from the bottom boundary</td>
<td>d1017236ff0</td>
<td>c1939be496ba</td>
</tr>
<tr>
<td>16-Aug-2018</td>
<td>Change Orlanksi boundary conditions to caculate along boundary wave on the boundary not one grid cell in.</td>
<td>7c6a30e8bb0</td>
<td>c1939be496ba</td>
</tr>
</tbody>
</table>

2.5 NEMO 3.6 Model Evaluation: 201702

Evaluating NEMO3.6 v201702
2.5.1 Data Used

Canadian Hydrographic Service (CHS) Realtime Water Level

CHS maintains a number of water level observation sites. Water level data are downloaded and compared to water levels at the nearest model grid point. Comparison in this document is for Sandy Cove from Aug 2017 - May 2018.

Citizen Science

The Salish Sea Marine Survival Project, with the support of the Pacific Salmon Foundation, have volunteer citizen scientists to take CTD casts and water samples at various stations in the Strait of Georgia. Data collected include temperature and salinity for all stations and nitrate, silicon, and chlorophyll for select stations. These trips take place between February and October and was done for 2015, 2016, and 2017. The stations are grouped by region, which include Lund, Powell River, Irvine, Baynes Sound, Nanaimo, Campbell River, Steveston, Victoria, Cowichan Bay, and Ladysmith. The 2015 nutrient dataset and 2015, 2016, and 2017 CTD datasets were processed by Rich and additional edits were made by Colin of PSF and Elise. Datasets for 2016 and 2017 nutrients and 2015, 2016, and 2017 chlorophyll were also made available to us. Edits of these datasets were made by Colin of PSF and Elise.

Drifters

During various trips to the Salish Sea, drifters are deployed and left to drift. They are all tracked with GPS devices and sometimes are picked up by bystanders and returned. As of present, there have been 21 deployments in the Salish Sea and the Juan de Fuca, each releasing up to ten drifters.

Fraser Plume CTD casts

Three boat trips, lead by Elise, were made to take CTD casts and collect water samples of the Fraser River Plume on April 10, 2017, May 31, 2017, and November 1, 2017, thus spanning the different seasons in a year. Data collected include pressure, temperature, salinity, beam transmission, PAR (photosynthetically active radiation), fluorescence, oxygen, and turbidity. Processing of the water sample was done at (ask Elise) Using loadDataFRP from SalishSeaTools, a choice of the raw data, slightly cleaned up data, and data interpolated to the Salish Sea model depths can be loaded.

IOS research cruises

The Institute of Ocean Sciences make research cruises every year in the Strait of Georgia. Available to us, we have nutrient data from the start of 201702 hindcast to until the end of 2016.

NANOOS San Juans Dataset

Downloaded from Nanoos, we have temperature and salinity data from the San Juans islands from fall 2014 - fall 2017.

ONC ferries

Since 2012, BC Ferry has been with ONC instruments to collect data. This data was recently made available on the salishsea erdapp server. Data collected include temperature, salinity, chlorophyll, turbidity and other variables along with their standard deviation and sample count. This particular ferry travelled from Tsawwassen to Duke Point (and back), crossing the Fraser River plume region. The data is collected every 1 minute.
ONC nodes

Refer to tidal current comparison.

Sentry Shoal

Temperature, salinity, and nitrate data at Sentry Shoal was made available to us by Stephanie King and Katie Pocock. The data was processed by Katie Pocock. The dataset covers most of 2015, 2016, and 2017.

Water quality buoy

A water quality buoy, maintained by Environment Canada and the BC Ministry of Environment, is located in the Fraser River. The raw data was processed by Rich and covers January 2009 to November 2017. Data collected include temperature, salinity, conductivity, pH, dissolved oxygen, stream velocity, and some atmospheric fields such as wind speed, wind direction, air temperature, humidity, and pressure.

2.5.2 Temperature and Salinity

Citizen Science

2015 - 2017 observations

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Temperature</th>
<th>Salinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>0.0078</td>
<td>0.14</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.55</td>
<td>0.72</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.97</td>
<td>0.93</td>
</tr>
</tbody>
</table>

In addition to an observation - model plot that includes both 2015 and 2016, separate comparisons were also made for each region. The Victoria stations are shown below.

CitSci - full notebook
CitSci - single days notebook
Only samples with valid times, longitude, latitude, and salinity values were used in the comparison.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>1.66856979031</td>
</tr>
<tr>
<td>RMSE</td>
<td>4.92462599804</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.782013448965</td>
</tr>
</tbody>
</table>

Ferry salinity notebook
Fraser Plume ctd

Statistics, available in the notebook below, were calculated for each depth. An example depth profile from May 31, 2017 is shown below.

CTD casts notebook

IOS cruises

Surface refers to depths less than 15m, intermediate refers to depths between 15 and 30m, and deep refers to depths greater than 30m.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Temperature</th>
<th>Salinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>0.05512010187655747</td>
<td>-0.0410403550465262145</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.7015201515771843</td>
<td>1.066426061995313</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.9684729188294054</td>
<td>0.954343789715143</td>
</tr>
</tbody>
</table>

San Juan Islands

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Temperature</th>
<th>Salinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>-0.4300265998100876</td>
<td>0.13353483468716476</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.5960042238649761</td>
<td>0.41932263796326535</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.7320908904807624</td>
<td>0.8433595642923585</td>
</tr>
</tbody>
</table>

Sentry Shoal

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Temperature</th>
<th>Salinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>0.4049012974465036</td>
<td>-0.28715771259387424</td>
</tr>
<tr>
<td>RMSE</td>
<td>1.4453098826255502</td>
<td>1.1850808771660581</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.9424725482806958</td>
<td>0.8290720829792324</td>
</tr>
</tbody>
</table>

Time series notebook

VENUS nodes

Comparison to the observed salinity were made to the model’s (then called hindcast) salinity.

Water quality buoy

Comparison to the model temperature was made at the model’s surface and depth = 1.5m, as well as daily and hourly averaged values. Below, statistics and plots are for model surface, and the observation - model plot is using the daily average.
DFO Temperature

- surface
- intermediate
- deep
- Juan de Fuca surface
- Juan de Fuca intermediate
- Juan de Fuca deep
Water quality buoy notebook (daily averages)

Water quality buoy notebook (hourly interpolated averages)

### 2.5.3 Nutrients

#### Citizen Science

Nitrate and silica observation were compared with the model.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Nitrate</th>
<th>Silica</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>0.8947724863656923</td>
<td>-13.375982797018917</td>
</tr>
<tr>
<td>RMSE</td>
<td>7.026915453406394</td>
<td>19.691922315063145</td>
</tr>
<tr>
<td>Willmott Score</td>
<td>0.8232355848913526</td>
<td>0.6499662706883669</td>
</tr>
</tbody>
</table>

CitSci Nutrients 2015

CitSci Nutrients 2016

CitSci Nutrients 2017

#### IOS cruises

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Nitrate</th>
<th>Silica</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>-2.35359531664837</td>
<td>-12.516292687941863</td>
</tr>
<tr>
<td>RMSE</td>
<td>5.36571883921976</td>
<td>15.864570645189147</td>
</tr>
<tr>
<td>Willmott Score</td>
<td>0.912549833123462</td>
<td>0.7493683577201395</td>
</tr>
</tbody>
</table>

IOS notebook
2.5. NEMO 3.6 Model Evaluation: 201702
Sentry Shoal

Observations were made at 1m.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nitrate bias</td>
<td>-1.8818928507550012</td>
</tr>
<tr>
<td>Nitrate RMSE</td>
<td>5.801877196234966</td>
</tr>
<tr>
<td>Willmott Score</td>
<td>0.8469078330784996</td>
</tr>
</tbody>
</table>

SS notebook

2.5.4 Chlorophyll

Citizen Science

2015-2017 chlorophyll measurements from the Citizen Science program was compared with the model.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>0.864778575809408</td>
</tr>
<tr>
<td>RMSE</td>
<td>4.599369232848489</td>
</tr>
<tr>
<td>Willmott Score</td>
<td>0.5048541500786355</td>
</tr>
</tbody>
</table>

CitSci Chl 2015

CitSci Chl 2016

CitSci Chl 2017

Ferry

In addition to not using chlorophyll data where one or more of longitude, latitude, chlorophyll, or time is masked, ferry chlorophyll values > 25 µg/L were excluded.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>2.18282683221</td>
</tr>
<tr>
<td>RMSE</td>
<td>4.86143879112</td>
</tr>
<tr>
<td>Willmott Score</td>
<td>0.498952254572</td>
</tr>
</tbody>
</table>

Ferry Chl notebook
Chlorophyll Concentration ($\mu g/L$), depth = 5m
Time series notebook

**IOS cruises**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>1.1295675984707625</td>
</tr>
<tr>
<td>RMSE</td>
<td>3.1130318776174235</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.6498509292361905</td>
</tr>
</tbody>
</table>

Chlorophyll Concentration (μg/L)

- Depth < 15m
- 15m < Depth < 30m

Model Results vs. IOS cruises
2.5.5 Tides

Sea level height for Sandy Cove was extracted from the CHS water level observations and compared to the model results using a notebook.

Raw Sea level from the Model is well-correlated with the observations and when the unmodelled constituents are added, the correlation improves. Even when removing the tides, the sea level is very well predicted.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Model (Raw)</th>
<th>Model (Corrected)</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.11</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>Willmott Skill Score</td>
<td>0.997</td>
<td>0.998</td>
<td>0.947</td>
</tr>
</tbody>
</table>

One note, the maximum amplitude of the tides seems small. See the first two figures.

2.5.6 Storm Surge

Sea level height for Sandy Cove was extracted from the CHS water level observations and compared to the model results using a notebook.

Raw Sea level from the Model is well-correlated with the observations and when the unmodelled constituents are added, the correlation improves. Even when removing the tides, the sea level is very well predicted.

2.5.7 Drifters

Using Ariane, particles are set up to be released in the same time and location as the drifters and then tracked through their journey. The particles are restricted to the first depth level, similar to how the drifters are always at the surface.
Sea Surface Height at Sandy Cove,
Aug 20, 2017 - May 8, 2018
2.6 Tidal evaluation

2.6.1 Literature review (tides)

Evaluating the performance of the NEMO model of the Strait of Georgia will begin with an evaluation of its skill at reproducing the tides in the domain.

(a) how are models evaluated in terms of tidal data?

(b) how are tidal forcings applied at boundaries?

Foreman et al. (1995)

Three dimensional, barotropic model of eastern Juan de Fuca Strait and southern Strait of Georgia, does not include stratification, wetting/drying or estuarine flow

(a) how is model evaluated?

• For tidal heights, amplitudes and phases of the eight major constituents at each tidal observation site were calculated by harmonic analysis (Foreman 1977)

• For tidal currents, ellipse parameters from a similar analysis were used for comparisons

• Used tidal data at 38 sites, subsampled from a list of 90 tidal stations (taken from Parker 1977) to only include stations with more than 120 days of records

• Used current meter data from 10 sites

• Model values at each observation location were interpolated between model node values

• Differences calculated as distances in the complex plane:

\[ D = \left[ (A_0 \cos g_0 - A_m \cos g_m)^2 + (A_0 \sin g_0 - A_m \sin g_m)^2 \right]^{1/2} \]

where \( A_0, A_m, g_0 \) and \( g_m \) are observed and modelled amplitudes and phases

• M2 differences were between 0.8cm (Pedder Bay) and 8.7cm (Sidney), with average around 3cm

• K1 differences were between 0.4cm (Pedder Bay) and 5.4cm (Sidney), with average around 2.5cm

• M2 is too early at Sooke, possibly because of large frictional effects of mudflats, which are not considered in calculation

• M2 is too small and too late at Clover Point and Sidney possibly because the interaction of estuarine flow with tides was not taken into effect

• M2 inaccuracies at Point Grey may be due to proximity of Fraser River mudflats and a long jetty that is not in the model domain

• M2 inaccuracies at Ladysmith, Maple Bay and Patricia Bay may be due to the Cowichan River, whose discharge could interfere with tidal propagation in narrow channels around these sites

• Also calculated root-mean-square differences between all measured and modelled amplitudes and phase differences

  – rms amplitude differences all within 2.0cm

  – rms phase differences all within 6.3 degrees

• Plotted out co-amplitude and co-phase lines for major constituents to compare to Crean et al (1988)
• Ellipses were compared (qualitatively?) between measured currents and modelled currents throughout the water column, but could not account for baroclinic effects such as internal tides which were probably affecting speed variations with depth

• Observed M2 constituent was not constant throughout the year at Victoria, possibly due to the exclusion of estuarine flow

(b) how are tidal forcings applied at boundaries?

• $e^{i\omega t}$ time dependence assumed for each tidal constituent with frequency $\omega$

• 8 tidal constituents used at boundaries (M2, S2, N2, K2, K1, O1, P1 and Q1) plus a residual tide and compound tides and overtides as a result of nonlinear interactions between these constituents

• Zero flow normal to the coast at boundaries

• Specified elevations on open sea boundaries mostly taken from observations
  – cotidal charts (Parker 1977, Crean et al. 1988) were sometimes used to deduce trends for interpolation and extrapolation along boundaries

• Specific boundaries (see Figure 2 of Foreman et al. (1995) for map):
  – Admiralty Inlet boundary: forced with Port Townsend and Admiralty Head tidal harmonics
  – Juan de Fuca boundary: forced with Sheringham Point and Seiku tidal harmonics
  – Northern boundaries: forced with Ivines Landing, Northwest Bay, Squitty Bay, False Bay and Skerry Bay tidal harmonics

Foreman et al. (2000)

Model of the north east Pacific Ocean, including Alaskan and BC shelf. Resolution ranges from 80km off shore to about 100m along the coast. The grid in eastern Juan de Fuca Strait and southern Strait of Georgia is identical to Foreman et al. (1995). Since there are only two crossings in the Strait of Georgia, only part of the Strait is included in the model.

(a) how is model evaluated?

• Compared model $M_2$ and $K_1$ harmonics against those calculated from harmonic analysis of 5.3 years of altimeter observations (Topex/Poseidon), at locations of crossover of the two satellite paths within the model domain

• Model accuracy was determined by calculating RMS differences between modelled and observed tidal harmonics at the crossover locations:

$$D_{rms} = (C_{rms}^2 + S_{rms}^2)^{1/2}$$

$$C_{rms} = \left[ \sum_{1}^{N} (A_T \cos G_T - A_m \cos G_m)^2 / N \right]^{1/2}$$

$$S_{rms} = \left[ \sum_{1}^{N} (A_T \sin G_T - A_m \sin G_m)^2 / N \right]^{1/2}$$

where N is the number of crossover sites, $A_T$, $G_T$, $A_m$ and $G_m$ are the altimeter and modelled amplitudes and phases respectively

• $M_2$ differences showed that modelled amplitude and phase lags were generally too small, possibly due to the existence of an amphidromic ridge near the south and west boundaries

• $K_1$ amplitude and phase lags looked better
other semi-diurnal and diurnal constituents had similar (but scaled down) inaccuracies

assimilated

(b) how are tidal forcings applied at boundaries?

- along southern and western open boundaries, initial elevation amplitudes and phases for major constituents (M2, S2, N2, K2, K1, O1, P1 and Q1) were calculated from the TPXO.3 world tidal model
- Puget Sound boundary (Admiralty Inlet) forced with identical amplitudes and phases to Foreman et al. (1995)
- Strait of Georgia boundary (Northern boundaries) forced with identical amplitudes and phases to Foreman et al. (1995)
- Queen Charlotte Strait forced with identical amplitudes and phases to Foreman et al. (1993)
- All coastal boundaries were free slip


POM model of southern Strait of Georgia and Juan de Fuca Strait

(a) how is model evaluated?

- Compared qualitatively to Foreman et al. (1995) e.g. the model reproduces the degenerate M2 amphidrome
- Calculated root-mean-square differences between all measured and modelled amplitudes and phase differences
- Average relative and absolute rms differences (D) between observed and calculated amplitudes and phases at 44 tide gauge sites, calculated by:

\[ D = \left[ \frac{1}{2} (A_m^2 + A_o^2) - A_m A_o \cos(\phi_m - \phi_o) \right]^{1/2} \]

where \( A_m \) and \( A_o \) are sea level amplitude of model and observations and \( \phi_m \) and \( \phi_o \) phases
- Absolute error of 1.7cm - 5.5cm, relative error (=D/Ao) of 2.2% - 13.7%

(b) how are tidal forcings applied at boundaries?

- Model is forced at two open boundaries with four tidal constituents (K1, O1, M2 and S2) through a 'forced gravity wave radiation condition on the normal component of the depth-integrated velocity (Flather 1987)'
- These four constituents account for about 70% of tidal stream velocity

Sutherland et al. (2011)

ROMS model of Salish Sea and Puget Sound

Group’s website: http://faculty.washington.edu/pmacc/MoSSea/index.html

(a) how is model evaluated?

- Calibrated their model with tidal data from NOAA tide gauges (none in BC, all in OR and WA)
- Amplitude, phase and spring-neap variability of tidal signals
- Calculated Skill Score (SS) and
- Calculated correlation coefficient (R2), which is the variance between two variables:

\[ R = \frac{1}{\sigma_m \sigma_o} \frac{1}{N} \sum_{i=1}^{N} (m_i - \bar{m})(o_i - \bar{o}) \]
where $m_i$ is the model variable at time or location $i$, $o_i$ is the observed variable at time or location $i$, $N$ is the number of observations, $\sigma_m$ and $\sigma_o$ are the standard deviations of model and observed variables and overbar indicates an average

- Also compared ratios of modeled to observed amplitudes of M2, S2 and K1 constituents using t_tide (Pawlowicz et al 2002)
- The model had better skill at diurnal frequencies because at semi-diurnal frequencies, modeled amplitude was too low
- In regions where tidal observations were not available, comparisons were made to an empirical tidal model developed for Puget Sound (Lavelle et al. 1988)
- Weather induced pressure anomalies are not represented in the model, so large difference occurred during one winter event

(b) how are tidal forcings applied at boundaries?

- Open boundaries forced with eight constituents (M2, S2, K1, O1, N2, P1, K2 and Q1) derived from the 1/4 degree TPXO7.1 inverse global tidal model (Egbert and Erofeeva 2002)
- Open boundaries were offshore i.e. straight boundary along 127 degrees W and the northern boundary in the Strait of Georgia was closed

References


2.6.2 Data acquisition (predicted tides and measured water level)

Tidal forcing is required on the boundaries of the NEMO model. Also, to evaluate the performance of the NEMO model in reproducing the tides, modelled water level output must be compared to measured water level data.

DFO modelled tidal predictions

WebTide is an online tidal prediction tool developed by DFO. The predictions for the North East Pacific region are based on the model of Foreman et al. (2000). Tidal currents and elevations can be predicted from the model.

WebTide was used for the Juan de Fuca boundary conditions in the preliminary runs of the NEMO model. WebTide doesn’t quite include our Johnstone Strait boundary, which is around -126.5 deg, 50.5 deg. Webtide cuts out around -126.8 deg, 50.6 deg… so it’s close!

Tidal constituents (amplitude and phase) can be output from Webtide for surface elevation and tidal currents.

For tidal currents, these constituents are relative to compass north, south etc. However, firstly our grid is rotated 29 degrees and secondly, NEMO takes Z1 and Z2, where
\[ Z1 = A \cos \phi \]
\[ Z2 = A \sin \phi \]

So to go from Webtide constituents to grid constituents, with a rotation of theta (= 29 degrees), for \( U \)

\[ Z1 = u_a \cos \theta \cos \phi_u - v_a \sin \theta \sin \phi_v \]
\[ Z2 = u_a \cos \theta \sin \phi_u + v_a \sin \theta \cos \phi_v \]

And for \( V \)

\[ Z1 = -u_a \sin \theta \cos \phi_u - v_a \cos \theta \sin \phi_v \]
\[ Z2 = -u_a \sin \theta \sin \phi_u + v_a \cos \theta \cos \phi_v \]

where

\[ u_a = \text{Webtide } u \text{ amplitude} \]
\[ v_a = \text{Webtide } v \text{ amplitude} \]
\[ \phi_u = \text{Webtide } u \text{ phase} \]
\[ \phi_v = \text{Webtide } v \text{ phase} \]
\[ \theta = \text{Grid rotation(29°)} \]

**Thomson & Huggett (1980) - Johnstone Strait**

Thomson & Huggett (1980) measured tidal elevations at 4 stations in Johnstone Strait 1976 and 1977. Tidal elevation harmonics for O1, K1, M2 and S2 are reported in Table 2 and stations are shown in Figure 4.

They also measured tidal currents at 11 stations (at various depths) in Johnstone Strait and report tidal current harmonics in Table 1.

Note that phase in these tables is reported as phase lag relative to 120 degrees W longitude. This means the measurements were probably in Pacific Standard Time (PST) and that’s how the lags were calculated.

Now, 120 degrees W (Pacific Standard Time) is 8 hours different to GMT, so to convert M2 phase from PST to GMT:

\[ \text{GMTphase} = \text{PSTphase} - (-8\text{hours}) \times (1/12.42\text{hours}) \times 360\text{degrees} \]

(For details on this conversion, see Manual for Tide Heights Analysis and Prediction by M.G.G. Foreman, Pacific Marine Science Report 77-10, IOS, 1977 (Revised 2004), Section 2.3.1 Astronomical argument and Greenwich phase lag, available from here or also here)

This paper also have some info on temperature and salinity in Johnstone Strait that may be useful.

**Thomson (1976) and Thomson (1977) - Johnstone Strait**

Four months of current observations in Johnstone Strait
TPXO7.1 modelled tidal predictions

TPXO7.1 is an online tidal prediction tool funded by NASA. The predictions are based on the model by Egbert and Erofeeva (2002) at Oregon State University. The model is a 1/4 degree x 1/4 degree global inverse tide model. Predictions can be calculated using a Matlab based GUI called TMD (the Tide Model Driver).

TPXO7.1 was used for the boundary conditions of the model by Sutherland et al. (2011) . . . but resolution is coarse in Juan de Fuca Strait and the Strait of Georgia is an inland sea . . . Sutherland et al. (2011) used the tidal predictions for a straight boundary along 127 degrees W. The TPXO7.1 predictions are probably only useful when boundaries are offshore from the coast.

DFO measured data

Measured water level data can be downloaded in .csv format from Canada’s Department of Fisheries and Oceans (DFO) website.

- 10 years of hourly water level measured data can be downloaded at a time
- 1 month of highest resolution (e.g. 1 minute water level at Point Atkinson) can be downloaded at a time

Permanent data stations

Permanent DFO stations in Strait of Georgia and on Vancouver Island (station number, sampling interval, starting year):
- Point Atkinson (7795, 1min, 1914)
- Vancouver (7735, 1min, 1909)
- Patricia Bay (7277, 1min, 1966)
- Victoria Harbour (7120, 1min, 1909)
- Bamfield (8545, 1min, 1969)
- Tofino (8615, 1min, 1909)
- Winter Harbour (8735, 1min, 1989)
- Port Hardy (8408, 1min, 1964)
- Campbell River (8074, 1min, 1965)
- New Westminster (7654, 1min, 1969)

Temporary data stations

Temporary DFO stations in Strait of Georgia and on Vancouver Island (station number, sampling interval, starting year, end year, days of data, consecutive record?):

Vancouver:
- station name (num, ??min, year1, year2, numdays)
- Point Grey (7635, 60min, 1977, 1978, 552)
- North Arm (7634, 60min, 1969, 1969, 175)
- Sand Heads (Stn Harry) (7604, 60min, 1968, 1969, 283)
- Sand Heads (7594, 60min, 1969, 1969, 188)
• Roberts Bank (7592, 60min, 1981, 1982, 203)
• Tsawwassen (7590, 60min, 1967, 1978, 4002)
• Steveston (7607, 60min, 1969, 1997, 10440)
• Canoe Pass (7603, 60min, 1991, 1993, 423, no)
• White Rock (7577, 60min, 1972, 1972, 158, yes)
• Sandy Cove (7786, 1min, 2009, 2017, 2931, yes)
• Cascadia Terminals (7743, 1min, 2001, 2002, 84, yes)
• Stanovan (7747, 60 & 1min, 1963, 2002, 304, no)
• Port Moody (7755, 60min, 1962, 1965, 672, no)
• Sea Island (7625, 60min, 1969, 1969, 146, yes)
• N. Arm, Fraser (7640, 60min, 1959, 1959, 274, yes)
• Port Mann (7657, N/A, N/A, N/A, N/A)
• Pitt River (7660, 15min, 1959, 1959, 273, yes)
• Pitt Lake (7666, N/A, N/A, N/A, N/A)
• Whonnock (7676, N/A, N/A, N/A, N/A)
• Mission City (7680, N/A, N/A, N/A, N/A)

Gulf Islands:

• Montague Harbour (7420, 60min, 1964, 1964, 29, yes)
• Ganges (7407, 60min, 1915, 1915, 30, yes)
• Whaler Bay (7532, 60min, 1964, 1975, 1665, no)
• Georgina Point (7525, 60min, 1959, 1959, 37, yes)
• Village Bay (7414, 60min, 1964, 1964, 29, yes)
• Samuel Island (7370, 60min, 1961, 1961, 31, yes)
• Samuel Island (north shore) (7515, 60min, 1961, 1961, 31, yes)
• Hope Bay (7360, 60min, 1961, 1961, 32, yes)
• Bedwell Harbour (7350, 1min, 2001, 2002, 119, no)
• Narvaez Bay (7345, 60min, 1965, 1965, 40, yes)
• Tumbo Channel (7510, 60min, 1967, 1976, 3267, yes)
• Fulford Harbour (7330, 60min, 1952, 1992, 14493, yes)
• Maple Bay (7315, 60min, 1969, 1970, 367, yes)
• Crofton (7450, 60min, 1960, 1971, 485, no)
• Chemainus (7455, 60min, 1961, 1961, 34, yes)
• Ladysmith (7460, 60min, 1954, 1955, 402, yes)
• Preedy Harbour (7471, 60min, 1961, 1961, 35, yes)
• North Galliano (7435, 60min, 1972, 1972, 55, yes)
• Dionisio Point (7535, 60min, 1963, 1968, 50, no)
• **Valdes Island** (7542, 60min, 1963, 1963, 15, yes)

**Nanaimo region:**

- **Nanaimo Harbour** (7917, 3min, 1997, 2003, 2259, yes)
- **Hammond Bay** (7924, ??min, 1959, 1959, numdays)*
- **Winchelsea Is.** (7935, 60min, 1967, 1978, 4159, yes)
- **NanOOSE Harbour** (7930, 60min, 1986, 1993, 2764, yes)
- **Northwest Bay** (7938, 60min, 1967, 1968, 411, yes)
- **Boat Harbour** (7480, 60min, 1972, 1972, 50, yes)
- **Northumberland Channel** (7915, ??min, 1949, 1949, numdays)*
- **Pylades Channel** (7442, 3min, 1999, 1999, 1, yes)
- **Silva Bay** (7550, 60min, 1967, 2002, 873, no)
- **Cowichan Bay** (7310, 60min, 1961, 1961, 39, yes)
- **Piers Harbour** (7272, 60min, 1980, 1980, 212, yes)
- **Swartz Bay** (7270, 60min, 1963, 1963, 29, yes)
- **Tsehum Harbour** (7262, 60min, 1983, 1983, 62, yes)

**Victoria region:**

- **Sidney** (7260, 60min, 1953, 2000, 936, no)
- **Finlayson Arm** (7284, 60min, 1966, 1966, 102, yes)
- **Finnerty Cove** (7140, 60min, 1967, 1975, 2829, yes)
- **Oak Bay** (7130, 1min, 2001, 2004, 1017, no)
- **Clover Point** (7115, 60min, 1967, 1967, 235, yes)
- **Selkirk Water** (7121, 1min, 2011, 2011, 145, yes)
- **Esquimalt Lagoon** (7107, 60min, 1972, 1984, 2569, no)
- **Esquimalt Harbour** (7109, 60min, 1981, 2010, 1313, no)
- **Esquimalt Government Hbr** (7110, 60min, 1972, 1978, 1604, no)
- **Portage Inlet** (7125, ??min, year1, year2, numdays)
- **Gorge at Craigflower** (7124, ??min, year1, year2, numdays)
- **Gorge at Tillicum** (7123, ??min, year1, year2, numdays)
- **Gorge at Aaron Point** (7122, ??min, year1, year2, numdays)
- **Selkirk Water** (7121, ??min, year1, year2, numdays)

**Southern Vancouver Island**

- **Pedder Bay** (7080, 60min, 1967, 1969, 601, yes)
- **Becher Bay** (7030, 60min, 1976, 1976, 56, yes)
- **Twin Cove** (7022, 1min, 2011, 2011, 50, yes)
- **Sooke Basin** (7024, 60min, 1977, 1983, 1077, no)
- **Sooke** (7020, 60min, 1972, 1985, 4416, yes)
Washington:

- **Ferndale, Wash.** (7564, 60min, 1967, 1970, 1087, yes)
- **Patos Island** (7505, 60min, 1967, 1969, 422, no)
- **Reservation Bay** (7196, 60min, 1971, 1971, 302, yes)
- **Cornet Bay** (7193, 60min, 1971, 1971, 121, yes)
- **Port Townsend** (7160, 60min, 1971, 1971, 121, yes)
- **Meadowdale** (7182, 60min, 1971, 1971, 121, yes)
- **Seattle** (7180, 60min, 1970, 1970, 365, yes)
- **Port Angeles** (7060, 60min, 1973, 1973, 151, yes)
- **Crescent Bay** (7050, 60min, 1964, 1964, 30, yes)
- **Sekiu (Clallam Bay)** (7037, 60min, 1973, 1974, 370, yes)

West coast Vancouver Island (southern)

- **Port Renfrew** (8525, ??min, year1, year2, numdays)
- **Mutine Point** (8556, ??min, year1, year2, numdays)
- **Pocahontas Pt** (8560, ??min, year1, year2, numdays)
- **Chesnuknuw Cr** (8562, ??min, year1, year2, numdays)
- **Sproat Narrows** (8564, ??min, year1, year2, numdays)
- **Franklin River** (8565, ??min, year1, year2, numdays)
- **Stamp Narrows** (8570, ??min, year1, year2, numdays)
- **Port Alberni** (8575, ??min, year1, year2, numdays)
- **Brooksby Point** (8558, ??min, year1, year2, numdays)
- **Kildonan** (8557, ??min, year1, year2, numdays)
- **Head of Uchucklesit** (8559, ??min, year1, year2, numdays)
- **Effingham** (8585, ??min, year1, year2, numdays)
- **Walsh Island** (8586, ??min, year1, year2, numdays)
- **Ucluelet** (8595, ??min, year1, year2, numdays)
- **Kennedy Cover** (8623, ??min, year1, year2, numdays)
- **Warn Bay** (8626, ??min, year1, year2, numdays)
- **Cypress Bay** (8630, ??min, year1, year2, numdays)
- **Herbert Inlet** (8632, ??min, year1, year2, numdays)
- **Sulphur Passage** (8634, ??min, year1, year2, numdays)
- **Riley Cove** (8637, ??min, year1, year2, numdays)

West coast Vancouver Island (northern)

- **Gold River** (8650, ??min, year1, year2, numdays)
- **Saavedra Islands** (8645, ??min, year1, year2, numdays)
- **Esperanza** (8665, ??min, year1, year2, numdays)

2.6. Tidal evaluation
• Tahsis (8658, ??min, year1, year2, numdays)
• Zeballos (8670, ??min, year1, year2, numdays)
• Kyuquot (8710, ??min, year1, year2, numdays)
• Port Alice (8750, ??min, year1, year2, numdays)
• Bergh Cove (8754, ??min, year1, year2, numdays)
• Kwokwesta Creek (8755, ??min, year1, year2, numdays)
• Makwazniht I (8756, ??min, year1, year2, numdays)
• Coal Harbour (8765, ??min, year1, year2, numdays)
• Hunt Islet (8736, ??min, year1, year2, numdays)
• Cape Scott (8790, ??min, year1, year2, numdays)

Howe Sound
• Squamish Inner (7811, ??min, 1961, 1962, 395)
• Squamish (7810, ??min, 1961, 1961, 30)
• Latona Beach (7805, ??min, 1973, 1973, 60)
• Gibsons (7820, ??min, 1967, 1973, numdays)

Sunshine Coast
• Roberts Creek (7824, ??min, 1968, 1969, numdays)
• Porpoise Bay (7852, ??min, 1951, 1952, numdays)
• Halfmoon Bay (7830, ??min, 1967, 1969, numdays)
• Storm Bay (7847, ??min, 1963, 1963, numdays)
• Irvines Landing (7836, ??min, 1967, 1974, numdays)
• Egmont (7842, ??min, year1, year2, numdays)
• Saltery Bay (7868, ??min, 1967, 1969, numdays)
• Powell River (7880, ??min, 1965, 1973, numdays)
• Okeover Inlet (8006, ??min, year1, year2, numdays)
• Lund (7885, ??min, year1, year2, numdays)
• Prideaux Haven (8008, ??min, year1, year2, numdays)

Mid Strait of Georgia
• Squitty Bay (7980, ??min, year1, year2, numdays)
• Skerry Bay (7985, ??min, year1, year2, numdays)
• False Bay (7982, ??min, year1, year2, numdays)
• Welcome Bay (7990, ??min, year1, year2, numdays)
• Blubber Bay (7975, ??min, year1, year2, numdays)
• Hornby Island (7953, ??min, year1, year2, numdays)
• Denman Island (7955, ??min, year1, year2, numdays)
• Comox (7965, ??min, year1, year2, numdays)
Salish Sea MEOPAR Documentation

- Little River (7993, ??min, year1, year2, numdays)

Northern Strait of Georgia
- Mitlenatch (7895, ??min, year1, year2, numdays)
- Twin Island (7892, ??min, year1, year2, numdays)
- Surge Narrows (8045, ??min, year1, year2, numdays)
- Florence Cove (8055, ??min, year1, year2, numdays)
- Octopus Island (8050, ??min, year1, year2, numdays)
- Owen Bay (8120, ??min, year1, year2, numdays)
- Okis Island (8124, ??min, year1, year2, numdays)
- Brown Bay (8110, ??min, year1, year2, numdays)
- Seymour Narrows (8105, ??min, year1, year2, numdays)
- Chatham Pt (8180, ??min, year1, year2, numdays)
- Hardinge Is (8127, ??min, year1, year2, numdays)
- Big Bay, Stuart Island (8060, ??min, year1, year2, numdays)
- Blind Channel (8155, ??min, year1, year2, numdays)
- Kelsey Bay (8215, 60min, 1988, 1993, 1825)
- Yorke Island (8233, 60min, 1943, 1944, 365)
- Warren Islands (8254, 60min, 1981, 1981, 35)
- Cedar Island (8325, ??min, year1, year2, numdays)
- Alert Bay (8280, ??min, year1, year2, numdays)
- Sullivan Bay (8364, ??min, year1, year2, numdays)
- Stuart Narrows (8379, ??min, year1, year2, numdays)
- Drury Inlet (8381, ??min, year1, year2, numdays)
- Jennis Bay (8384, ??min, year1, year2, numdays)
- Frederick Sd (8458, ??min, year1, year2, numdays)
- Alison Sound (8488, ??min, year1, year2, numdays)
- Nugent Sound (8464, ??min, year1, year2, numdays)
- Charlotte Bay (8443, ??min, year1, year2, numdays)
- Mereworth Sound (8476, ??min, year1, year2, numdays)
- Egg Island (8805, ??min, year1, year2, numdays)
- Wadhams (8840, ??min, year1, year2, numdays)
- = not digitised

2.6. Tidal evaluation
Data format

Data can be downloaded online until the end of the previous month.

Downloaded files have 8 header lines with station name, number, lat, long, datum, time zone, type of data and column headers. Data is then in the form:

```
YYYY/MM/DD HH:MM,SLEV,
```

For example:

```
Station_Name,New Westminster, BC
Station_Number,7654
Latitude_Decimal_Degrees,49.2
Longitude_Decimal_Degrees,122.91
Datum,CD
Time_Zone,PST
SLEV=Observed Water Level
Obs_date,SLEV(metres)
2013/09/01 00:00,2.21,
2013/09/01 01:00,2.3,
2013/09/01 02:00,2.37,
2013/09/01 03:00,2.3,
```

NOAA measured data

Harmonic constituents and up to 31 days of measured water level data at NOAA tide gauges in the US can be downloaded here.

(remember that tidal heights are in feet!)

Tulare Beach, Port Susan 9448043 May 20 2013 - present
9447963 Tulalip Bay, WA - present
Tulalip Bay 9447773 Apr 8 1935 - Nov 13 1974
Toke Point 9440910 Feb 20 1922 - present
Westport 9441102 Jun 7 1982 - Jul 23 1982
La Push 9442396 Dec 19 1924 - Dec 18 1969
Neah Bay 9443090 Jul 23 1934 - Dec 18 1969
Port Angeles 9444090 Aug 30 1975 - present
Port Townsend 9444900 Dec 31 1971 - present
Tacoma 9446484 Jul 22 1996 - present
Seattle 9447130 Jan 1 1899 - present
Shilshole Bay Gps Buoy 9447214 Jan 1 2009 - present
Cherry Point 9449424 Nov 7 1971 - present
Friday Harbor 9449880 Jan 25 1932 - present
Cherry Point cp0101 (currents)
2.6.3 Tidal comparisons

Harmonics comparisons

Water level

- All 8 components (M2, K1, O1, S2, P1, N2, Q1 and K2) computed by NEMO can be compared to harmonics from water level (calculated by Mike Foreman, Institute of Ocean Sciences and for M2 and K1 shown in his papers Foreman et al (1995, 2000)).
- Jupyter Notebook: Analysis8Components.ipynb
- This notebook focuses on a set of 31 stations that form a line from Juan de Fuca up through the Strait of Georgia into Discovery Passage and the North End of the Model.

- Tidal harmonics are calculated using a least squares fit to the last 37.5 days of a 40-day model run without wind. Scatter plots comparing the measured and modelled harmonics are made, example:
• The notebook calculates differences (as described by Foreman et al (1995)) for M2 and K1 and saves them to a text file and plots them.

• This image clearly shows that other than at Seymour Narrows (station 26) the model K1 tide is within 5 cm of the measured tide. However, the M2 tide is poor not only in the Discovery Passage area (stations 24-28) but also in the Strait of Juan de Fuca. The latter is a choice. The model was tuned to match in Strait of Georgia (SoG). Due to over-prediction of the amphidrome near Victoria the model cannot reproduce both the M2 tide in Juan de Fuca and in SoG at the same time. This error is the only significant error with the tides. We think it is due to over mixing in the Gulf/San Juan Island region and are working on that.

In the above figure, green circles represent errors less than 5 cm, yellow circles represent errors between 5 and 10 cm and red circles mark errors greater than 10 cm.
2.6. Tidal evaluation
M2 and K1

- M2 and K1 tidal ellipses calculated from current measurements at the ONC nodes have been compared to the model output. The agreement is generally good, but the model currents are somewhat lower.
- M2 and K1 tidal ellipses calculated from CODAR current measurements have also been compared to the model output. The agreement is generally good, but the model currents are somewhat higher!
- Drifter measurements to be made in Sep 2014 will also be used to compare to the modelled tidal ellipses.

2.6.4 Tidal Sensitivity

This document outlines the tidal sensitivity to various parameters such as bottom friction and lateral viscosity. The following details are presented: set up of simulations, analysis techniques, and tidal response. The $M_2$ and $K_1$ tidal constituents are the main focus.

The modelled tidal response is compared to observations from Foreman et al. (1995), Foreman et al. (2004), and Foreman et al. (2012).

Procedure

Several 5-day long simulations forced with $M_2$ and $K_1$ tidal constituents have been completed. A sinusoidal curve fitting technique is applied to the time series of the modelled sea surface height to estimate the $M_2$ and $K_1$ amplitude and phase. The fitting curve is defined as follows:

$$f(t) = A_{M_2} \cos(\omega_{M_2} t - \phi_{M_2}) + A_{K_1} \cos(\omega_{K_1} t - \phi_{K_1})$$

where $A_{M_2}$, $A_{K_1}$, $\phi_{M_2}$, and $\phi_{K_1}$ are the $M_2$ and $K_1$ amplitudes and phases determined by the fitting procedure. The $M_2$ and $K_1$ frequencies $\omega_{M_2}$, $\omega_{K_1}$ are known.

Results

The modelled amplitude and phase will be compared to observations at several points within the domain: Port Renfrew, Point Atkinson, and Yorke Island. This way we can evaluate model performance over different regions. Other locations are available but will not be summarized here.

The one area that the model performs badly is the Gulf/San Juan Islands. The phase change through the Islands is too large. Probably related, around Victoria, there is an $M_2$ amphidrome; the amphidrome is too deep in the model. So one critical area is the difference in phase across these Islands, here measured by considering the full change in phase between Port Renfrew and Port Atkinson. In the observations, this value is 150.1 degrees.

We have calculated the phase difference $\Delta \phi = \phi_{mod} - \phi_{obs}$ and the amplitude ratio $R = \frac{A_{mod}}{A_{obs}}$. The results are summarized in the tables below.
### M\(_2\) Results

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Description</th>
<th>Bottom Friction</th>
<th>Viscosity</th>
<th>Mean Error (m/deg)</th>
<th>RMS Error (m/deg)</th>
<th>Diff. in Phase betw. PR &amp; PA</th>
<th>Port Renfrew</th>
<th>Point Atkinson</th>
<th>Yorke Island</th>
</tr>
</thead>
<tbody>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.100/9</td>
<td>0.13/10</td>
<td>5.158.6</td>
<td>1.01</td>
<td>0.91</td>
<td>10.4</td>
<td>1.11</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.051/10</td>
<td>0.11/12</td>
<td>1.257.3</td>
<td>1.16</td>
<td>0.94</td>
<td>13.4</td>
<td>1.15</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.090/9</td>
<td>0.13/11</td>
<td>1.161.0</td>
<td>0.99</td>
<td>1.60</td>
<td>1.09</td>
<td>1.19</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.098/9</td>
<td>0.13/10</td>
<td>3.158.5</td>
<td>1.00</td>
<td>0.94</td>
<td>10.2</td>
<td>1.16</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.083/11</td>
<td>0.14/17</td>
<td>1.644.4</td>
<td>0.96</td>
<td>1.54</td>
<td>1.19</td>
<td>1.10</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.100/9</td>
<td>0.13/10</td>
<td>5.158.7</td>
<td>1.01</td>
<td>0.94</td>
<td>10.5</td>
<td>1.15</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.005</td>
<td>0.082/12</td>
<td>0.14/17</td>
<td>2.164.6</td>
<td>0.96</td>
<td>1.55</td>
<td>1.19</td>
<td>1.10</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.001</td>
<td>0.071/6</td>
<td>0.13/13</td>
<td>1.166.0</td>
<td>0.96</td>
<td>1.01</td>
<td>0.66</td>
<td>1.08</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.001</td>
<td>0.094/8</td>
<td>0.13/11</td>
<td>1.370.1</td>
<td>0.96</td>
<td>1.07</td>
<td>0.04</td>
<td>1.03</td>
</tr>
<tr>
<td>tide_M2</td>
<td>tide_M2</td>
<td>0.001</td>
<td>0.098/9</td>
<td>0.08/18</td>
<td>1.923.1</td>
<td>0.99</td>
<td>0.92</td>
<td>2.14</td>
<td></td>
</tr>
</tbody>
</table>

### Summary

- Decreasing bottom friction increases M\(_2\) amplitude at Point Atkinson but decreases it at Port Renfrew. There is a good match at Point Atkinson when bottom friction is 1 × 10\(^{-3}\).
- Decreasing bottom friction decreases the M\(_2\) phase difference at all three locations, with the largest response at Port Renfrew and Yorke Island.
- Decreasing bottom friction increases the phase difference between Point Atkinson and Port Renfrew.
- Decreasing the viscosity has little effect at all three stations but does slightly decrease the phase difference between Point Atkinson and Port Renfrew.
- Increasing the flux decreased the phase difference between Point Atkinson and Port Renfrew but increased the amplitude drop.
• Decreasing the $K_1$ phase has little effect on the $M_2$ amplitude and phase.

• Point Atkinson $M_2$ phases are very consistent over all of the iterations, except the last where the $M_2$ phase has changed in forcing. The phases at Port Renfrew and Yorke Island are more sensitive to changes in bottom friction and viscosity.

• Increasing the slip, increases the phase difference between Point Atkinson and Port Renfrew. Decreasing the slip, decreases the phase difference but increases the amplitude drop.
### $K_1$ Results

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Description</th>
<th>Bottom Friction $R$</th>
<th>Viscosity $\Delta \phi$</th>
<th>Mean Error (m/deg)</th>
<th>RMS Error (m/deg)</th>
<th>$R$</th>
<th>$\Delta \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>tide_flux_M2K1</td>
<td>original tides at west, flux corrected at north</td>
<td>0.005</td>
<td>20</td>
<td>0.060/7.14</td>
<td>0.066/7.5</td>
<td>1.07</td>
<td>3.51</td>
</tr>
<tr>
<td></td>
<td>8.46</td>
<td>1.14</td>
<td>-5.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_flux_west</td>
<td>$M_2$ and $K_1$ flux increased 25% at west</td>
<td>0.005</td>
<td>20</td>
<td>0.126/5.75</td>
<td>0.131/6.3</td>
<td>1.19</td>
<td>0.151</td>
</tr>
<tr>
<td></td>
<td>7.11</td>
<td>1.16</td>
<td>-5.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_bottom</td>
<td>decrease bottom friction</td>
<td>0.003</td>
<td>20</td>
<td>0.079/6.7</td>
<td>0.085/7.2</td>
<td>1.08</td>
<td>6.31</td>
</tr>
<tr>
<td></td>
<td>7.10</td>
<td>1.14</td>
<td>-4.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_nu15</td>
<td>decrease lateral viscosity</td>
<td>0.005</td>
<td>15</td>
<td>0.059/7.0</td>
<td>0.066/7.4</td>
<td>1.07</td>
<td>3.48</td>
</tr>
<tr>
<td></td>
<td>8.24</td>
<td>1.14</td>
<td>-5.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_bottom1e-3</td>
<td>deceased bottom friction</td>
<td>0.001</td>
<td>20</td>
<td>0.110/6.4</td>
<td>0.119/7.7</td>
<td>1.10</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>5.27</td>
<td>1.14</td>
<td>-2.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_K1phase2</td>
<td>$K_1$ phase decreased 5 deg</td>
<td>0.005</td>
<td>20</td>
<td>0.063/3.5</td>
<td>0.069/4.2</td>
<td>1.07</td>
<td>-1.36</td>
</tr>
<tr>
<td></td>
<td>3.52</td>
<td>1.16</td>
<td>-5.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tide_K1amp</td>
<td>above plus $K_1$ amplitude decreased 15% and b.f.</td>
<td>0.001</td>
<td>20</td>
<td>0.030/2.7</td>
<td>0.044/4.5</td>
<td>0.934</td>
<td>5.81</td>
</tr>
</tbody>
</table>

#### 2.6. Tidal evaluation
Summary

- Decreasing the bottom friction increases the $K_1$ amplitude at Port Renfrew and Point Atkinson. There is no effect at Yorke Island.
- Decreasing the bottom friction increases the $K_1$ phase difference at Port Renfrew and Yorke Island, with the largest response at Port Renfrew. The Point Atkinson phase difference has decreased.
- Decreasing the viscosity has little effect on the $K_1$ amplitude and phase at these three locations.
- Increasing the flux greatly increased the amplitude errors.

Implications

From the M2 results, we decided that the bottom friction should not be reduced and we fixed it at 0.005, that the smaller viscosity was better so we fixed that at 15 $m^2s^{-1}$. We increased the Flux by 25% but left the slip at 0.5.

Both amplitude drop and phase increase between Port Renfrew and Point Atkinson are important, but the large phase change causes the largest total errors. So choices were made to reduce the phase increase at the expense of increasing the amplitude drop.

References


2.6.5 Tide Tuning

This document outlines our efforts to best tune the tides. Emphasis was on tuning the tides in the Strait of Georgia. Specifically, in the final tuning, the western tides were tuned to produce the best match with data at Point Atkinson, Gibsons Landing, Winchelsea and Halfmoon Bay and the northern tides were tuned to produce the best match with data at Kelsey Bay and Yorke Island.

The modelled tidal response is compared to observations from Foreman et al. (1995), Foreman et al. (2004), and Foreman et al. (2012).

Step One: New Base Run: M2 & K1 only

Based on the tidal variations study, we started with a run that included a shift downward by 5 degrees at the west for both the M2 and K1 tides and an increase in amplitude of 15% for K1. Run time was 5 days, bottom friction was 0.005, viscosity was 15 $m^2s^{-1}$. (Called CBase)

Step Two: Vary the Velocity Phase for M2

A series of runs (Cd10, Cd20, Cd30, Cu10, Cu20, Cu30) were done which varied the phase of the tidal velocity field at the western boundary compared the sea level. Increasing the phase by 30 degrees, decreased the phase difference between Point Atkinson and Port Renfrew by 2 degrees.
However, this run (Cu30) had low amplitude and late phase everywhere. Three runs were done (RC_amp, RC_pha, RC_both), that corrected the amplitude, phase and both, respectively. Correcting the amplitude and phase, returned the phase difference to its original value.

Thus, varying the velocity phase relative to the sea surface height phase was abandoned. (Two more runs FTN and FTK1 were done, tuning the north boundary and the west K1).

**Step Three: Correct North Flux**

The cross-section of the northern boundary is larger than the cross-section where the north measurements were made by Thomson. This required a correction (reduction) in the northern flux. Double the correction was used for M2 to better match with observations.

- Jupyter Notebook: Analysis8Components.ipynb

This run was called CBase2.

**Step Four: Correct west K1 and north M2**

The western K1 amplitude and phase were tuned and the northern M2 was tuned (run FTK1b)

**Step Five: Western M2 Flux**

From tide variations, we noticed that increasing the velocity flux for the M2 tide at the west, decreased the phase error. In Flux_RC we also tuned the phase and amplitude for the western M2 tide.

**Step Six: Add O1**

A series of runs including the next largest constituent (O1) were run. After the first one listed below, all runs were done for 10 days because at 5 days the K1 and O1 could not be reliably separated.

- Flux_RC_wO1 : same as Flux_RC but including O1 tide
- Chase_RC_wO1 : same as FTK1b (no M2 Flux correction) but with adjusted western M2 phase and amplitude and including O1 tide
- Flux_RC2_wO1 : like Flux_RC_wO1 but tuning M2 phase and amplitude, west and north
- Flux_RC3_wO1 : like Flux_RC2_wO1 but further tuning of west M2, west and north K1/O1

Note that for diurnal constituents O1, P1 and Q1 the ratio of their amplitude to K1 and difference of their phase to K1 was tuned. Similarly for S2, N2 and K2 relative to M2. This was on the suggestion of David Greenberg and worked very well.

**Step Seven: Add S2**

- Flux_RC3_wO1S2 : like Flux_RC3_wO1 but include S2
- RC4_wO1S2: further fine tuning of M2, K1, O1 and S2 phase and amplitude west and north
Step Eight: Add more

- RC4_w01S2P1N2 : add another two
- RC4_w01S2P1N2Q1K2 : all eight

10 days was not enough to separate constituents so further runs were done with 20 days.

From RC5-RC7, continued fine tuning.

Then moved to 30 days for RC8 and 40 days for R9 and further.

From RC9 on, tuning used Newton-Ralphson method assuming that the western amplitude only affected the amplitude in SoG, the western phase determines the phase in SoG and the northern amplitude and phase determined the Johnstone Strait amplitude and phase, respectively. There is indeed little correlation from the west to the north. However, phase does affect amplitude and vice versa.

Stopping

For the longer runs, error estimates were made by sub-sampling the full time-series multiple times. This allowed an estimate of the error in determining the harmonics. Once the difference between the mean value and the observations was less than twice the standard deviation of the model results, the tuning was consider finished.

At RC13 it was determined that the western tides were tuned.

At corr15 it was determined that the northern tides were tuned.

Full run parameters are at:

- spreadsheet: TideTuningRunParameters.ods

Newton-Ralphson Process is at:

- spreadsheet: diagnostics.ods

And final results can be seen in:

- Jupyter Notebook: Analysis8Components

2.7 Tidal currents evaluation

2.7.1 Literature review (tidal currents)

(a) Why use tidal ellipses?

(b) What have researchers previously done in the Salish Sea and the Strait of Georgia?

Overview of tidal ellipses

(a) *Why use tidal ellipses?*

Tidal currents vary a lot within a fjord-like estuarine system. The Salish Sea is not an exception of this. The hydrodynamics of the channel can drastically change the tidal currents depending on the gradient of these variations. We can observe larger current speeds in the many narrow straits of the Georgia Strait, as well as near the entrance, where it connects to the Pacific Ocean. Tidal currents also vary vertically, generally due to bottom friction, but also baroclinic effects. (Parker, 2007)

- Tidal current constituent ellipses describe the flow as it rotates at a single location for a single constituent’s cycle.
• Tidal ellipses can communicate a lot about a depth profile of the flow at particular locations with only a couple of parameters. At present, we have focused our analysis on the M2 and K1 constituents because they are the largest components in our domain. The M2 component is much stronger than the K1. (Thomson 1981)

• Tidal ellipses enable us to see the speed and direction of a flow due to a chosen tidal constituent at every hour of the day as a depth profile using only orthogonal velocities at that point over a long period of time. (Parker, 2007)

• Comparisons of tidal ellipses from the model output with observations facilitates the understanding of which physical processes are properly and poorly represented and may even inform techniques to improve the poorly represented processes.

(b) What have researchers previously done in the Salish Sea and the Strait of Georgia?

**National Oceanic and Atmospheric Administration (2007)**

• Suggest using harmonic constituent ellipses to combine the orthogonal component time series (u and v current vectors)

• The orthogonal components can be chosen to be north/south or along direction of maximum flood and perpendicular to it.

• This book uses the derivation of Doodson and Warburg (1941, p.180-1):

\[
N(t) = W_N \cos(\eta t - \kappa_N) \\
E(t) = W_E \cos(\eta t - \kappa_E)
\]

where \(N(t)\), and \(E(t)\), are the north and east components of the tidal constituents, \(W_N\) and \(W_E\) are the amplitudes of the north and east components, \(\eta\) is the frequency and \(\kappa_N\) and \(\kappa_E\) are the phase lags in the respective direction.

• This can be generalized for any two orthogonal components \(MJ(t)\) and \(MN(t)\), major and minor components:

\[
MJ(t) = N(t) \cos(\theta) + E(t) \sin(\theta) \\
MN(t) = E(t) \cos(\theta) - N(t) \sin(\theta)
\]

where \(\theta\), is the major axis direction clockwise from the north.

**Foreman et al. (2004)**

• Currents calculated with the TIDE3D finite element model. This model did not include vertical variations in currents by using a large vertical viscosity. This basically outputs constant currents through the water column.

• Presents along channel amplitude and phase.

• Although stratification is known to be important at the Haro Strait and Johnstone Strait Central sites, baroclinic effects are not captured by this model.

• They only report amplitude and phase.

• The model uses data assimilation of the elevation observations to improve predictions. These model values are an improvement to the original model presented.
  
  – The semi-major axes are smaller at the more southern sites and larger at the northern sites
  
  – Improvements at southern sites but mostly no changes at the northern sites
– Decreased current speeds by 10 cm/s in most of Juan de Fuca Strait, Gulf and San Juan Islands and the southern Strait of Georgia

– Increased currents in northern Georgia and in Johnstone Strait

• We can transform these into tidal ellipses using Xu (2000) technique described below to facilitate the comparison of our Salish Sea model output to the Foreman (2004) model.

Zhigang Xu (2000)

Short paper describing a technique for ellipse conversion explained in much detail.

• Uses complex tidal currents to convert between tidal current amplitude and phase lag parameters and tidal current ellipse parameters and vice versa.

\[
\begin{align*}
w &= u + iv \\
u &= a_u \cos(\omega t - \phi_u) \\
v &= a_v \cos(\omega t - \phi_v)
\end{align*}
\]

where \( w \) is the complex tidal current, \( \omega \), is the frequency of the chosen tidal constituent, \( \phi_u \) and \( \phi_v \) are the phase lag for the u- and v- components and \( a_u \) and \( a_v \) are the amplitudes for the u- and v- components.

• Tracing out \( w \) on a complex plane gives an ellipse, from this ellipse we can calculate many parameters that provide information about the flow.

\[
\begin{align*}
w &= W_p e^{i(\omega t + \theta_p)} + W_m e^{-i(\omega t - \theta_m)}
\end{align*}
\]

• From these equations we can extract all the ellipse parameters

• This is the method we used to calculate the tidal ellipses from the Salish Sea model outputs.


Johnstone Strait is part of the domain, it is important to understand the research that has previously been done regarding the area and the observations that are being compiled.

• Five current meters were deployed across the the Johnstone Strait, eastward of Newcastle Sill, in June 1973.

• More were deployed between 1976 and 1978.

• Table of 10 stations in the western basin of Johnstone Strait containing the along channel amplitude (semi-major axis) and the phase (from 120 deg W.) and 2 to 10 depths for each station. The moorings were out for 7 to 92 days, depending on the device type, location and start time.

• The semi-minor axis amplitude was not reported because it is very small in this region.

• M2 components are only fully resolved for time series that are longer than 27.6 days.

References


• Thomson, R.E., 1981. Oceanography of the British Columbia Coast. Canadian Special Publication of Fisheries and Aquatic Sciences 56, Department of Fisheries and Oceans, Ottawa, 291pp.


### 2.7.2 Tidal current sensitivity

• The Salish Sea MEOPAR model outputs daily averages and hourly averages of tracers and velocities at every grid point. It also outputs quarter-hourly results at the Ocean Networks Canada VENUS Central and East nodes. The image below indicates where the Central and East nodes are located.

• We wonder if it is beneficial to compile the quarter-hourly data for the velocities and whether or not it would produce a more accurate tidal ellipse.

• The quarter-hourly output has started on May 9th 2015, where as the hourly output has been produced with the current tidal configuration since November 26th 2014.

• The separation of harmonic tidal constituents is important when we want to look at a single constituent’s effect on the current. This is accomplished with a long time series. Therefore, we want to see the effects of the length of a time series on the tidal ellipses.

• The figure below is a comparison of the depth profile of tidal ellipses for one month of quarter-hourly output, one month of hourly output and 6 months hourly output.

• The table below is populated with the values that were used to create the figure above.
2.7. Tidal currents evaluation
Table 1: Sensitivity to Frequency and Length Comparison

<table>
<thead>
<tr>
<th>Depth (m)</th>
<th>Central Quarter-Hourly</th>
<th>Hourly</th>
<th>Hourly (6months)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major-Axis (m/s)</td>
<td>Minor-Axis (m/s)</td>
<td>Inclination (deg. ccw E)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.29</td>
<td>-0.13</td>
<td>129</td>
</tr>
<tr>
<td>2.5</td>
<td>0.26</td>
<td>-0.12</td>
<td>129</td>
</tr>
<tr>
<td>4.5</td>
<td>0.22</td>
<td>-0.10</td>
<td>129</td>
</tr>
<tr>
<td>6.5</td>
<td>0.20</td>
<td>-0.09</td>
<td>130</td>
</tr>
<tr>
<td>8.5</td>
<td>0.20</td>
<td>-0.08</td>
<td>131</td>
</tr>
<tr>
<td>10.5</td>
<td>0.19</td>
<td>-0.08</td>
<td>131</td>
</tr>
<tr>
<td>12.5</td>
<td>0.19</td>
<td>-0.07</td>
<td>132</td>
</tr>
<tr>
<td>14.6</td>
<td>0.18</td>
<td>-0.06</td>
<td>132</td>
</tr>
<tr>
<td>16.8</td>
<td>0.17</td>
<td>-0.05</td>
<td>131</td>
</tr>
<tr>
<td>19.5</td>
<td>0.16</td>
<td>-0.04</td>
<td>129</td>
</tr>
<tr>
<td>24.1</td>
<td>0.15</td>
<td>-0.02</td>
<td>125</td>
</tr>
<tr>
<td>34.7</td>
<td>0.14</td>
<td>-0.00</td>
<td>119</td>
</tr>
<tr>
<td>58.5</td>
<td>0.12</td>
<td>0.02</td>
<td>116</td>
</tr>
<tr>
<td>98.1</td>
<td>0.13</td>
<td>0.03</td>
<td>134</td>
</tr>
<tr>
<td>147.1</td>
<td>0.18</td>
<td>0.00</td>
<td>144</td>
</tr>
<tr>
<td>199.6</td>
<td>0.19</td>
<td>0.00</td>
<td>137</td>
</tr>
<tr>
<td>253.1</td>
<td>0.22</td>
<td>-0.00</td>
<td>120</td>
</tr>
<tr>
<td>306.8</td>
<td>0.15</td>
<td>0.05</td>
<td>109</td>
</tr>
</tbody>
</table>

**Frequency**

- One month of quarter-hourly values makes no significant difference on the tidal ellipse parameters at the ONC VENUS Central node. The changes are sparse and minimal.
- We will need to wait until we have a longer quarter-hourly data set to see if it is beneficial for tidal ellipses, or try other more sensitive locations.

**Length**

- The 6 months time series has two significant differences that lead us to believe that these ellipses are a better representation of the currents:
  - The smaller surface currents indicates that it averages out some of the surface currents that are due to winds or Fraser River outflow.
  - The smoother variations throughout the water column. It is not logical for the flow to fluctuate every couple of meters throughout the water column.
- A longer time series enables us to separate and resolve the individual harmonic constituents so that we can be certain that we are only looking at the M2 tidal constituent.
- We observe the benefits of a longer time series at the ONC VENUS East node as well.

**Seasons**
Table 2: The effect of seasonality on tidal ellipse parameters

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major-Axis (m/s)</td>
<td>Minor-Axis (m/s)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.38</td>
<td>-0.14</td>
</tr>
<tr>
<td>2.5</td>
<td>0.33</td>
<td>-0.11</td>
</tr>
<tr>
<td>4.5</td>
<td>0.29</td>
<td>-0.06</td>
</tr>
<tr>
<td>6.5</td>
<td>0.27</td>
<td>-0.04</td>
</tr>
<tr>
<td>8.5</td>
<td>0.27</td>
<td>-0.03</td>
</tr>
<tr>
<td>10.5</td>
<td>0.27</td>
<td>-0.03</td>
</tr>
<tr>
<td>12.5</td>
<td>0.28</td>
<td>-0.04</td>
</tr>
<tr>
<td>14.6</td>
<td>0.29</td>
<td>-0.04</td>
</tr>
<tr>
<td>16.8</td>
<td>0.30</td>
<td>-0.05</td>
</tr>
<tr>
<td>19.5</td>
<td>0.31</td>
<td>-0.05</td>
</tr>
<tr>
<td>24.1</td>
<td>0.32</td>
<td>-0.06</td>
</tr>
<tr>
<td>34.7</td>
<td>0.31</td>
<td>-0.05</td>
</tr>
<tr>
<td>58.5</td>
<td>0.31</td>
<td>-0.03</td>
</tr>
<tr>
<td>98.1</td>
<td>0.35</td>
<td>-0.05</td>
</tr>
<tr>
<td>147.1</td>
<td>0.25</td>
<td>0.09</td>
</tr>
</tbody>
</table>

- Larger, more circular currents at the surface in the spring
- Narrower and smaller currents at mid-depths in the spring
- The deep bottom currents are almost identical.
- The Central node has a similar but muted difference between the seasons.
- The bottom ~150m are mostly the same throughout winter and spring. We expect this because there isn’t much movement there.
- The semi-major and semi-minor axis are larger throughout the upper part of the water column in the spring.

2.7.3 Tidal current comparisons

- Comparisons of the model to observations and other models is very important in evaluating the model output.
- These comparisons give information about which physical processes are poorly represented and may even give hints towards how we can improve model performance.

Ocean Networks Canada VENUS nodes

- ONC VENUS has three seafloor nodes in the Salish Sea; Central, East and Delta. The nodes are equipped with many sensors.
- The comparisons below use the Acoustic Doppler Current Profiler (ADCP) at the Central and East nodes.

Nodes

- The ONC VENUS data has been processed by Dr. Rich Pawlowicz and Dr. Mark Halverson, these are the ADCP values in the table below and the opaque ellipses in the figure.
2.7. Tidal currents evaluation
• Note that the observational values are averaged over approximately 10-11 months where as the model output values were averaged over 8 months - Nov 26, 2014 to Jul 13, 2015.

Comparison of the M2 harmonic constituent

<table>
<thead>
<tr>
<th></th>
<th>Major-Axis (m/s)</th>
<th>Minor-Axis (m/s)</th>
<th>Inclination (deg ccw E)</th>
<th>Phase (deg GMT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M2 - Central - (35-290m)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>0.16</td>
<td>0.013</td>
<td>130</td>
<td>305</td>
</tr>
<tr>
<td>ADCP</td>
<td>0.18</td>
<td>0.006</td>
<td>109</td>
<td>311</td>
</tr>
<tr>
<td><strong>M2 - East - (20-160m)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>0.29</td>
<td>-0.017</td>
<td>137</td>
<td>324</td>
</tr>
<tr>
<td>ADCP</td>
<td>0.26</td>
<td>-0.005</td>
<td>125</td>
<td>320</td>
</tr>
</tbody>
</table>

• The figure shows that the depth averaged observational and model output values agree quite well.

• The model output’s semi-minor axis is larger than the observations, this informs us that the model flow is too circular.

• Also the Salish Sea model output is angled slightly more to the West, this discrepancy is within a reasonable error window.

• Scale = 50

• Here we are plotting individual parameters to be able to see what aspects are well represented with the model.

Comparison of the K1 harmonic constituent

<table>
<thead>
<tr>
<th></th>
<th>Major-Axis (m/s)</th>
<th>Minor-Axis (m/s)</th>
<th>Inclination (deg ccw E)</th>
<th>Phase (deg GMT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K1 - Central - (35-290m)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>0.08</td>
<td>0.005</td>
<td>135</td>
<td>184</td>
</tr>
<tr>
<td>ADCP</td>
<td>0.11</td>
<td>0.007</td>
<td>112</td>
<td>188</td>
</tr>
<tr>
<td><strong>K1 - East - (20-160m)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>0.19</td>
<td>-0.025</td>
<td>131</td>
<td>226</td>
</tr>
<tr>
<td>ADCP</td>
<td>0.17</td>
<td>-0.011</td>
<td>122</td>
<td>220</td>
</tr>
</tbody>
</table>

CODAR

• The Fraser River plume creates some very interesting surface currents where it meets the Georgia Strait. These currents can be studied using the two High Frequency coastal radars installed by ONC that measure surface current velocities. The system is called the Coastal Ocean Dynamics Applications Radar system (CODAR).

• The tidal ellipses at 0.5 m depth at both nodes are very large for the M2 constituent but there is quite a good agreement for the K1 values. These ellipses were calculated over a 5 months period, form November 26th 2014 to April 26th 2015, to avoid the freshet. The ellipses with the freshet, 8 month time series, are slighter wider and the inclination does not match the observations as well as the values without the freshet.
2.7. Tidal currents evaluation
2.7. Tidal currents evaluation
<table>
<thead>
<tr>
<th></th>
<th><strong>Major-Axis (m/s)</strong></th>
<th><strong>Minor-Axis (m/s)</strong></th>
<th><strong>Inclination (deg ccw E)</strong></th>
<th><strong>Phase (deg GMT)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M2 - Central - (0.5m)</strong></td>
<td>0.18</td>
<td>-0.057</td>
<td>139</td>
<td>317</td>
</tr>
<tr>
<td>Model</td>
<td>0.11</td>
<td>0.008</td>
<td>126</td>
<td>273</td>
</tr>
<tr>
<td>CODAR</td>
<td>0.11</td>
<td>0.008</td>
<td>126</td>
<td>273</td>
</tr>
<tr>
<td><strong>M2 - East - (0.5m)</strong></td>
<td>0.37</td>
<td>-0.151</td>
<td>129</td>
<td>299</td>
</tr>
<tr>
<td>Model</td>
<td>0.14</td>
<td>0.037</td>
<td>134</td>
<td>282</td>
</tr>
<tr>
<td>CODAR</td>
<td>0.14</td>
<td>0.037</td>
<td>134</td>
<td>282</td>
</tr>
<tr>
<td><strong>K1 - Central - (0.5m)</strong></td>
<td>0.06</td>
<td>0.003</td>
<td>130</td>
<td>203</td>
</tr>
<tr>
<td>Model</td>
<td>0.07</td>
<td>0.006</td>
<td>134</td>
<td>131</td>
</tr>
<tr>
<td>CODAR</td>
<td>0.07</td>
<td>0.006</td>
<td>134</td>
<td>131</td>
</tr>
<tr>
<td><strong>K1 - East - (0.5m)</strong></td>
<td>0.10</td>
<td>0.007</td>
<td>119</td>
<td>204</td>
</tr>
<tr>
<td>Model</td>
<td>0.10</td>
<td>0.007</td>
<td>119</td>
<td>204</td>
</tr>
<tr>
<td>CODAR</td>
<td>0.08</td>
<td>-0.009</td>
<td>129</td>
<td>187</td>
</tr>
</tbody>
</table>

**Full Surface Comparisons**

We have performed additional comparisons of the model phase over the full CODAR region. In the following plots, a harmonic analysis was performed at the model surface grid cells (0.5m) from Nov. 26, 2014 to Apr. 26, 2015.

- The figure below is the phase of the M2 tidal constituents from the model output at 0.5 m depth throughout the CODAR region compared to the observational values processed by Dr. Mark Halverson.
- We can see that the model and the observations are in the same range of values, however, the observations have steep gradients and variations in the center of the Strait of Georgia unlike the model. The accuracy of the CODAR values are still in the process of being determined therefore it is unwise to pull any conclusions from this.
- This figure below is the phase of the K1 tidal constituent.
- There is similar features as with M2 phase, the steep changes in gradient. The zone at the mouth of the Fraser that has a smaller phase than the rest of the area in the model seems like it may extend further out in the observations.

**Foreman Model (2004)**

In addition to observations, we have compared with a two dimensional, high-resolution barotropic model of the Puget Sound and around Vancouver Island (Foreman et al, 2004). We use this model and a comparison tool because it can help identify which aspects we are represented similarly or differently to the Foreman model and which of these are or aren’t comparable to the observations he has reported from current meter deployments.

**Along the thalweg**

- The thalweg is the set of grid points that follow along the deepest point of the Strait of Georgia from every cross section.
- When looking along the thalweg we can see that both models are reasonable through the Juan de Fuca Strait. The Salish Sea model is a bit closer to the observations. Through Haro Strait both models seem to be off by 0.3 m/s in the semi-major axis. Through the center of the Strait both models have very low velocities as expected. The northern part of the Strait is better resolved by the Foreman model. The Salish Sea model is more focused on getting the southern area correct and many rivers and passages are closed off in this model which causes this
Surface M2 tidal ellipses at VENUS nodes

0.2 m/s
Surface K1 tidal ellipses at VENUS nodes

2.7. Tidal currents evaluation
discrepancy. Foreman et al (2004) reports that their discrepancies in the northern Strait either due to either the grid resolution being too coarse and/or the observations being inaccurate.

2.7.4 Tidal currents tools

Tidal currents

Tidal currents are tightly knit with the hydrodynamics. When the currents are averaged over a long period of time what we have left is the tidal effect on the currents. We can average out the winds and surface currents anomalies. It is deeply dependant on bottom friction and mixing (Parker, 2007). Evaluating tidal currents enables one to see these effects at a chosen locations.

To calculate and evaluate tidal currents we need long (many months) currents time series for a particular location.

Observations

- Current data can be obtained or seen at Ocean Networks Canada (ONC) website.
- The ONC data is then processed by Dr. Rich Pawlowicz.
- Phase and amplitude information can be reported in papers, we use values reported in Foreman et al. (2004).

Tidal Ellipses

We perform a harmonic analysis on the time series in order to extract the tidal harmonic constants. From the speed and direction of the tidal current at one location over time we can construct harmonic constituents ellipse which are used to describe the motion of the water due to one tidal constituent (eg. M2).

Separating the Constituents

Similarly to tides, tidal currents are composed of many different harmonic constituents. When calculating and comparing tidal ellipses it is of one constituents at one location, it is important to have a long enough time series to be able to fully separate the constituents.

Freshet and Changing Shorelines

It is important to avoid using data for a tidal analysis during the Fraser River freshet. The outflow from the river is very larger from the end of may to mid June and this skews the tidal ellipses. The bathymetry and shoreline drastically effects the tidal currents. However, they can change over time. It is important to select data that would have a consistent shoreline and bathymetry across datasets that are being compared.

Python Tools for Tidal Analysis

Tidal Analysis and Ellipse Calculation

Some python functions have been written to facilitate calculating amplitude and phase from velocity vectors and to calculate the ellipse parameters from these values. These scripts are found in tools/SalishSeaTools/salishsea_tools/ellipse.py and tools/SalishSeaTools/salishsea_tools/tidetools.py

- tidetools.fittit - This script fits a time series to a tide curve and extracts the amplitude and phase of each tidal constituent in the tide curve.
This function finds tidal parameters from a tidal constituent across a specified area of the grid at a single depth, at a single point through the water column or a single depth averaged grid point. For currents, this function must perform twice, once for each tidal current vector in order to complete the analysis. This function should also work with water level analysis.

**Note:** This function can be used to analyze a time series of sea surface height, u velocity, or v velocity. In fact, it can be used for any scalar variable. But it does not handle important things like inference or nodal corrections. In those cases, it is much better to use \texttt{t\_tide} or apply inference and nodal corrections on your own.

The \texttt{nconst} input sets how many tidal constituents will be analysed. They come in pairs and in order of importance the domain. It returns a dictionary object containing the phase and amplitude for each component for the input array.

The fitting routine solves for a set of amplitude and phase parameters (one set for each tidal constituent) by finding the best match between the model’s time series and a tidal curve that is predetermined by the tidal constituents that it will be solving for. Below is an example of the equation for the \( u \) and \( v \) tidal curves for the M2 and K1 constituents:

\[
\begin{align*}
    u &= \text{mean} + A_{M2}\cos(\omega_{M2}t - \theta_{M2}) + A_{K1}\cos(\omega_{K1}t - \theta_{K1}) \\
    v &= \text{mean} + A_{M2}\cos(\omega_{M2}t - \theta_{M2}) + A_{K1}\cos(\omega_{K1}t - \theta_{K1})
\end{align*}
\]

where \( \omega_{M2} \) and \( \omega_{K1} \), \( \theta_{M2} \) and \( \theta_{K1} \), and \( A_{M2} \) and \( A_{K1} \) are the frequencies, phase lags and amplitudes for the M2 and K1 components. “Mean” is an unknown values that the fitting routine will solve for however it is not used in the tidal ellipse calculations.

- \texttt{ellipse.ellipse\_params} - This script converts from the amplitude and phase lag parameters to the tidal current ellipse parameters:

\[
\text{ellipse.ellipse\_params}(uamp, upha, vamp, vpha)
\]

This function calculates the tidal ellipse parameters based on the conversions shown in Xu, Z. (2000). It outputs the positively and negatively rotating amplitude and phase, as well as the major and minor axis and the axis tilt and phase.

- \texttt{ellipse.get\_params\_nowcast} - This script gives the tidal ellipse parameters for a given date range and location based on the hourly model output values:

\[
\text{ellipse.get\_params\_nowcast}(to, tf, i, j, path, nconst, depthrange='None', \text{depav=False, tidecorr=CorrTides})
\]

This function loads all the data between the start and the end date that contains hourly velocity netCDF4 files. Then it mask, unstaggers and rotates the velocities by component about the grid point described by the \( i \) and \( j \). Lastly it fits the velocities and calculates the tidal ellipse parameters for that date range using the \texttt{fittit} and \texttt{ellipse\_param} functions above.

After finding the amplitude and phase of the orthogonal vector by using \texttt{fittit} it does a nodal correction, determined by the start date of the nowcasts, Sept 10, 2014. These values values and other constituents tide corrections can be found in: /data/dlatorne/MEOPAR/SalishSea/nowcast/08jul15/ocean.output/. This function outputs a dictionary object containing the ellipse parameters for each tidal harmonic constituent.

- In this notebook: \texttt{UsingEllipse.py.ipynb} there are simple examples of the functions above.
MATLAB Scripts for Tidal Analysis

Some MATLAB tools have been written for analyzing barotropic and baroclinic tidal currents from NEMO u/v output. The scripts load NEMO data and then apply t_tide to perform a harmonic analysis. The advantage of using t_tide is that it can analyze many constituents and easily handles nodal corrections and inference. Several scripts for baroclinic, barotropic and surface currents analysis have been written. These scripts take care of masking, unstaggering, rotating and depth averaging as needed.

Note: The NEMO u and v output are expected to be contained in a single netCDF file. Remember that u and v are stored on slightly different lat and lon grids. The longitude and latitude grid stored in the netCDF file should correspond to the T-grid.

Note: Depending on the length of your time series and size of your subdomain, it may be very memory intensive to load your files.

These scripts and their dependencies are stored in analysis/Nancy/currents/t_tide_analysis.

- **area_surface_tides.m** - This script analyzes the full tidal current at the surface. It saves the ellipse parameters for each constituent in a file.

```matlab
area_surface_tides(filename, outfile, t0, ref_time, time_units)
```

- **area_deipv_tides.m** - This script analyzes the depth-averaged tidal currents.

```matlab
area_deipv_tides(filename, outfile, depthfile, t0, ref_time, time_units, use_mask)
```

- **area_baroclinic_tides.m** - This script analyzes the baroclinic tidal currents. The baroclinic tidal currents are defined as the full current subtract the depth-averaged current. This definition may be inaccurate in regions where boundary layer effects are important.

```matlab
area_baroclinic_tides(filename, outfile, depthfile, t0, ref_time, time_units, use_mask)
```
Output

This notebook gives an example of loading the output from these scripts in python. It makes use of functions in analysis/Nancy/currents/baroclinic.py.

References


2.7.5 Ocean Networks Canada ADCP Currents

Ocean Networks Canada (ONC) maintains several deep water nodes with ADCP current meters in the Strait of Georgia. This page describes the scripts used to process their data.

MATLAB Scripts for ONC Data

Loading and processing of the observational data from the ONC VENUS Central, East and Delta nodes is done in MATLAB scripts written by Dr. Rich Pawlowicz. The processing is done in three parts and is tailored for each deployment at each node.

Processing Scripts

- The first part is GET_DATA_fun.m This script will get the data that is directly output from the ADCP. It does this for the two days before the day indicated. It will put this data in a directory at pth/raw/ and organize it by year and month. This function calls a script written by Marlene Jeffries at Ocean Network Canada getSoGAdcpDataMay15_mod. This script contains many functions that are used to ultimately retrieve the raw data from the ONC website.
- The next step is to run GET_DEPL_fun.m goes through all the data in the raw directory gathered by GETDATA_fun and bins it into 30 minutes bins.
- Lastly, the bulk of the processing is done in LTIM_fun.m. This script filters out the tides, corrects the angles for the velocities to get major axis in the direction of the flood current.

Adjustments for Running Daily

- When running these scripts for a single day of data at a time to have daily comparisons a few modifications have to be done to keep the scripts running. First of all, GETDEPL_fun creates a new deployment file with the new updated raw data that was loaded by GETDATA_fun however LTIM_fun needs only one mat file per deployment in the directory where it looks. The compare_daily functions works helps seamlessly join the new update deployment file and the previous deployment file.
New deployment

Once or twice a year ONC performs maintenance on the nodes that involves bringing the ADCP package to the surface and redeploying the same or a different sensor to the bottom a few hours later. Each deployment has its own unique set of data and metadata including:

- date/time range (resolved to the hour)
- sensor serial number (also known as id)
- number of data bins
- data bin size
- data bin 1 distance
- vertical offset
- rotation angle

Values for those data and metadata are required in order for the automated daily download and processing of the ADCP data to be restarted following a new deployment. Those values have to be added to arrays in the GETDATA_fun.m, GETDEPL_fun.m, and LTIM_fun.m Matlab scripts (details below). Some of the values must be determined by graphical analysis of data from the new sensor deployment, and there must be enough data available for that analysis that the tidal signal can be averaged out. So, at least a week’s data must be collected after a new deployment before the graphical analysis can be done. Once the analysis has been completed the data since the beginning of the deployment must be re-processed with the newly obtained sensor data and metadata values.

As of October 2015, downloading that data from ONC is accomplished via a Matlab script (getSogAdcpData*.m) provided after each deployment by Marlene Jefferies of ONC.

The notes that follow were written during the process of restarting the daily download automation following the new deployments at the end of August 2015.

Obtain and Modify the getSogAdcpData.m Script from ONC

Obtain the new version of the getSogAdcpData.m Matlab script for the deployment from Marlene Jefferies at ONC, and store it in /ocean/dlatorne/MEOPAR/ONC_ADCP/ with the deployment date added to the file name; e.g. getSogAdcpData_2Sep2015.m.

Change the email and userId lines near line 180 to the email address and 5-digit ONC user id of the user that owns the automation cron job, for example:

```matlab
p.addParamValue('email','dlatornell@eos.ubc.ca', @ischar);
p.addParamValue('userId', ddddd, @isnumeric);
```

The LocationName comment blocks starting at about line 50 in the getSogAdcpData.m script provide the site name (e.g. VIP-13) and sensor serial number (e.g. 8497) for all of the deployments to date. Those values for the most recent deployments are required to update the GETDATA_fun.m and GETDEPL_fun.m scripts. Also required are the end date/time for the previous deployment, and the start date/time for the present deployment from the switch statement at about line 430.

Update the GETDATA_fun.m Script

Add the new deployment numbers, start date/times, and sensor serial numbers to the history arrays in the switch statement that starts at about line 35 in the copy of private-tools/ONC-ADCP/GETDATA_fun.m that is symlinked into /ocean/dlatorne/MEOPAR/ONC-ADCP/. The history arrays also require deployment end date/times; choose a value in the future beyond the expected duration of the deployment.
The updates made for the late-August 2015 deployments are highlighted below:

```
switch nodloc,
  case 'east',
    node='SOG-East-Node';
    history=[...
      01 2007 10 19 00 2008 09 25 00 8497;
      ... 2 2015 03 31 22 2015 08 27 16 8497;
    13 2015 08 27 22 2016 12 31 00 8497]; % or present...

  case 'central',
    node='SOG-Central-Node';
    history=[...
      01 2008 09 24 00 2009 09 27 00 8580;
      ... 12 2014 09 20 02 2015 08 30 15 8580;
    13 2015 08 31 02 2016 12 31 00 8580]; % or present...

  case 'ddl',
    node='SOG-Delta-Node';
    history=[...
      08 2013 10 23 17 2014 03 06 17 2940; % DDL 148m
      02 2014 03 08 21 2014 09 20 18 17955; % BBL-SG-02 at 142m
      03 2014 09 22 00 2015 08 28 15 17955; % BBL-SG-03 at 149m
      04 2015 08 30 19 2016 12 31 00 17955; % BBL-SG-04 at ???m
    depname={'DDL-','BBL-SG-','BBL-SG-','BBL-SG-'};
```

Finally, update the name used to call the `getSogAdcpData.m` script at the end of the `GETDATA_fun.m` script (about line 173). The update made for the late-August 2015 deployments was:

```
getSogAdcpData_2Sep2015('locationName',node,'siteName',deploy,'deviceName',instr,
   'dateStart',sdate,'dateEnd',edate,'maxTimeout',120,
   'outputdirectory',outdir);
```

**Download Raw Data from Deployment Date to Present**

Run the `GETDATA_fun.m` script to download the raw data for each node for the days from the deployment to present. That can be accomplished by running Matlab in command-line mode on salish:

```
$ cd /ocean/dlatorne/MEOPAR/ONC_ADCP/
$ matlab -nodesktop -nodisplay
... >> GETDATA_fun('08-Sep-2015', 'central', '/ocean/dlatorne/MEOPAR/ONC_ADCP/', 10)
```

The data download takes about 10 minutes per day requested, so it is advisable to start Matlab in 3 separate terminal sessions and run commands like the above for each of the nodes: central, east, and ddl. The `GETDATA_fun.m` script handles breaking the requested number of days into 7 day chunks (the maximum that the ONC hardware can handle). The `getSogAdcpData*.m` script downloads the data into the `path/mode/raw/` directory; i.e. `/ocean/dlatorne/MEOPAR/ONC_ADCP/central/raw/` in the example above.

The `get_VENUS_ADCP_raw.cron.sh` and `get_VENUS_ADCP_raw.m` scripts can be used to automate daily
downloading of the raw data during the period required to obtain enough data to complete the analysis required to get
the variable values to facilitate fully automated processing.

**Update the GETDEPL_fun.m Script**

The GETDEPL_fun.m script must be updated with the same deployment numbers, start date/times, and sensor serial
numbers that were added to the GETDATA_fun.m script. Also required are the number of bins, the bin size, and
the bin 1 distances values from one of the sensor metadata files that is generated for each download request. The
easiest way to obtain those values is to follow the link to the metadata HTML file that is included in the download
completion email message generated by the ONC system. Add those values to the history arrays in the switch
statement that starts at about line 22 in the copy of private-tools/ONC-ADCP/GETDEPL_fun.m that is sym-
linked into /ocean/dlatorne/MEOPAR/ONC-ADCP/. The history arrays also require vertical offset values
for the deployment; use the same value as for the previous deployment to start with, it will be tuned later.

The updates made for the late-August 2015 deployments are highlighted below:

```matlab
switch nodloc,
    case 'east',
        node='SOG-East-Node';
        history=[ ... % ID #bins binsize bin1distance (one that
        '`best' works) offset
        01 2007 10 19 00 2008 09 25 00 8497 85 2.614 0;
        ... 2 2015 03 31 22 2015 12 31 00 8497 85 2 6.39 +3;
        13 2015 08 27 22 2016 12 31 00 8497 85 2 6.40 +3];
    case 'central',
        node='SOG-Central-Node';
        history=[ ... 01 2008 09 24 00 2009 09 27 00 8580 38 8 12.47 0;
        ... 12 2014 10 01 00 2015 12 31 00 8580 60 5 9.24 0; % <- change in
        parameters here (same VIP)
        13 2015 08 31 02 2016 12 31 00 8580 60 5 9.25 0];
    case 'ddl',
        node='SOG-Delta-Node';
        history=[ ... 08 2013 10 23 16 2014 03 06 17 2940 75 2 4.3 0; % DDL 148m
        02 2014 03 08 21 2014 09 20 18 17955 115 1.33 3.96 0; % BBL-SG-02 at
        142m
        03 2014 09 22 00 2015 12 31 00 17955 115 1.33 3.96 0; % BBL-SG-03 at 149m
        04 2015 08 30 19 2016 12 31 00 17955 115 1.33 3.96 0]; % BBL-SG-04 at
        147m
        depname={'DDL-','BBL-SG-','BBL-SG-','BBL-SG-'};
```

Chapter 2. Contents
Update the **LTIM_fun.m** Script

The **LTIM_fun.m** script must be updated with the rotation angles of the ADCPs for the new deployments. Initially the angles are set to the same values as for the previous deployments so that the **LTIM_fun.m** script can be run.

The updates for the late-August 2015 deployments are highlighted below:

```matlab
switch nodloc,
    case 'central',
      rotang=[0; % VIP 01
                22; % 02
                ... % 58+180; % 12- v2
                58+180; % 13
                ]-61;
    ...
    case 'east',
      rotang=[57;
                0; % VIP 02 - can not start with 1!!
                50; % 03
                ... % 72; % east2
                72; % 13
                ]-106;
    ...
    case 'ddl',
      rotang=[-60; % DDL-08
                35; % BBL-SG-02
                -20; % BBL-SG-03
                -20; % BBL-SG-04
                ];
end
```

After at least 7 to 10 days of data have been downloaded and processed by the **GETDEPL_fun.m** script, the **LTIM_fun.m** script will be run interactively to produce plots that will allow the instrument rotation angles for the deployment to be determined. Once that has been done the values highlighted above will be updated.

**Changing users**

If you will be running the processing in a new directory for the first time there are a couple things to change in order to facilitate the transitions.

- 1. **In** `compare_daily.m` **change** the path to be where you want everything to be saved. Many extra files will appear in this directory every time you run the scripts.

- 2. **Make an account on** [http://www.oceannetworks.ca/information](http://www.oceannetworks.ca/information) **to get** `userId`. **In** `getSoGAdcpDataMay15_mod.m` **insert** your email and `userId` at lines 173 and 174 of the script. You will receive an email every time you load raw data from the website.

- 3. **In** `GET_DATA_fun` **change** the `firstdate` variable to be at least 3 days before the `lastdate`. This is because the filter length in **LTIM_fun** needs at least that much data for the processing.
Setup of the /ocean/ ONC ADCP Data Filespace

This section describes the setup of the storage filespace on /ocean/ containing the accumulated raw and processed ONC ADCP data. Those data are from the Strait of Georgia Central, East, and Delta Dynamics Laboratory (DDL) nodes. Also described in this section is the software automation that updates those data daily with the observations from the previous day.

The data and processing scripts are stored in /ocean/dlatorne/MEOPAR/ONC_ADCP/. The accumulated, processed data for the 3 nodes are in the files:
- /ocean/dlatorne/MEOPAR/ONC_ADCP/ADCPcentral
- /ocean/dlatorne/MEOPAR/ONC_ADCP/ADCPddl
- /ocean/dlatorne/MEOPAR/ONC_ADCP/ADCPeast

The raw data downloaded from ONC are in directory trees organized by year and month number; e.g. 2015/07/ in the directories:
- /ocean/dlatorne/MEOPAR/ONC_ADCP/central/raw/
- /ocean/dlatorne/MEOPAR/ONC_ADCP/ddl/raw/
- /ocean/dlatorne/MEOPAR/ONC_ADCP/east/raw/

The other files in the /ocean/dlatorne/MEOPAR/ONC_ADCP/ tree are the processing scripts, sensor deployment data files, etc. Many of those files are symlinked from version controlled files in the private-tools.

Preparing the /ocean/dlatorne/MEOPAR/ONC_ADCP/ Filespace

- Create the directory and make it group writable for the sallen group:
  ```bash
  $ mkdir /ocean/dlatorne/MEOPAR/ONC_ADCP
  $ cd /ocean/dlatorne/MEOPAR/ONC_ADCP
  $ chgrp sallen
  $ chmod g+w .
  ```

- Copy the accumulated-to-date processed data files into the filespace:
  ```bash
  $ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/ Nodes/ADCPcentral.mat ./
  $ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/ Nodes/ADCPddl.mat ./
  $ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/ Nodes/ADCPeast.mat ./
  ```

- Create directories for the raw data and per-node processing files, and make them group writable for the sallen group:
  ```bash
  $ mkdir central ddl east
  $ chgrp sallen central ddl east
  $ chmod g+w central ddl east
  ```

- Symlink the historic sensor deployment data files for each node into their respective directories:
  ```bash
  $ cd central
  $ for n in {01..10}; do
    ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/central/DEPL$n*.mat
  done
  ```

(continues on next page)
$ cd ../ddl
$ for n in {1..2}; do
> ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/ddl/DEPL0${n}*.mat
> done

$ cd ../east
$ for n in {01..12}; do
> ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/east/DEPL${n}*.mat
> done
$ cd ..

• Copy the current sensor deployment data files for each node into their respective directories:

$ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/˓
→Nodes/central/DEPL1VIP-12-11.mat central/
$ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/˓
→Nodes/ddl/DEPL03BBL-SG-03-03.mat ddl/
$ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/˓
→Nodes/east/DEPL13VIP-02-13.mat east/

• Copy the accumulated-to-date raw data file trees for each node into their respective directories and make the directories group writable for the sallen group:

$ mkdir central/raw
$ chgrp sallen central/raw
$ cp -r --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/ ˓
→TidalEllipseData/Nodes/central/raw/2015 central/raw/
$ find central/raw -type d | xargs chmod g+w

$ mkdir ddl/raw
$ chgrp sallen ddl/raw
$ cp -r --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/ ˓
→TidalEllipseData/Nodes/ddl/raw/2015 ddl/raw/
$ find ddl/raw -type d | xargs chmod g+w

$ mkdir east/raw
$ chgrp sallen east/raw
$ cp -r --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/ ˓
→TidalEllipseData/Nodes/east/raw/2015 east/raw/
$ find east/raw -type d | xargs chmod g+w

• Create symlinks to the version-controlled processing scripts:

$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/compare_daily.m
$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/GETDATA_fun.m
$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/GETDEPL_fun.m
$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/LTIM_fun.m
$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/get_VENUS_ADCP.m
$ ln -s /data/dlatorne/MEOPAR/private-tools/ONC_ADCP/get_VENUS_ADCP.cron.sh

• Copy the ONC-provided data download script into the filespace:

$ cp --preserve=timestamps /ocean/mdunn/MEOPAR/analysis/Muriel/TidalEllipseData/ ˓
→Nodes/getSogAdcpDataMay15_mod.m ./

TODO: That script should be symlinked from a version controlled copy

2.7. Tidal currents evaluation
• Create a **matlab** function in `get_VENUS_ADCP.m` to run `compare_daily.m` for each node:

```matlab
function get_VENUS_ADCP
    % Run the compare_daily.m script for each ONC VENUS node of interest
    % to download and process the ADCP data for the previous day
    % using Rich Pawlowicz's scripts.
    compare_daily(date, 'central', 2)
    compare_daily(date, 'ddl', 2)
    compare_daily(date, 'east', 2)
end
```

• Create a **bash** script called `get_VENUS_ADCP.cron.sh` for **cron** to execute to run `get_VENUS_ADCP.m`:

```bash
# Download and process VENUS nodes ADCP data for the previous day
# using matlab scripts written by Muriel Dunn and Rich Pawlowicz.
# usage: 0 14 * * * /ocean/dlatorne/MEOPAR/ONC_ADCP/get_VENUS_ADCP.cron.sh
cd /ocean/dlatorne/MEOPAR/ONC_ADCP
matlab -nodesktop -nodisplay -r get_VENUS_ADCP
```

• Make `get_VENUS_ADCP.cron.sh` owner and group executable:

```bash
$ chmod ug+x get_VENUS_ADCP.cron.sh
```

• Add a line to the **crontab** on `salish` to execute `get_VENUS_ADCP.cron.sh` daily:

```bash
OCEAN_MEOPAR=/ocean/dlatorne/MEOPAR
0 10 * * * $OCEAN_MEOPAR/ONC_ADCP/get_VENUS_ADCP.cron.sh
```

## 2.8 Rivers

### 2.8.1 River Input

**Sources**

River input provides a significant volume of freshwater to the Salish Sea and can influence stratification, circulation and primary productivity. We need to parametrize the rivers that flow into the Salish Sea throughout the domain.

Morrison et al. (2011) provides a method for estimating freshwater runoff in the Salish Sea region based on precipitation. We acquired the exact data from Morrison, which includes the runoff volumes for each watershed for each year from 1970 to 2012, as well as monthly averages. These data are saved in netcdf files in the `rivers-climatology`.

Figure 1 of Morrison et al. (2011) shows the watershed boundaries but for more precise detail, some of these boundaries coincide with watershed boundaries shown on this ArcGIS map.

Next, we split the freshwater runoff from each watershed between the rivers in that watershed. To do this, we needed accurate maps of the rivers in the region. The following sources were used:

- For BC, maps of rivers can be found on the Atlas of Canada - Toporama site.
- In Washington, maps of watersheds are available at this site.
For the Fraser River, the split between the arms is given in Thomson, 1981, available here.

From the maps, the percentage of the watershed that each river drains was estimated.

Based on the values in Morrison et al. (2011), the approximate percentage of freshwater input from each watershed in our domain is given below:

- Fraser 44%
- Skagit 12%
- East Vancouver Is (North and South) 12%
- Howe 7%
- Bute 7%
- Puget 6%
- Juan de Fuca 5%
- Jervis 4%
- Toba 3%

Through the method described above, we have parameterised a total of 150 rivers in our domain.

**Watersheds**

**Fraser**

- Source used: BCCF map and WA map and Thomson, 1981 to split Fraser flow among its arms
- Includes three arms of the Fraser River (v. important to Salish Sea!) and some little US rivers south of the Fraser, shown in WRIA1
- Assume that the Fraser River itself occupies 98% of this watershed, and that WRIA1 occupies 2% of this watershed
- WRIA1
  - 20% of WRIA1 flows into the Fraser anyway
  - WRIA1 = Flux*(0.02*0.80)
  - Fraser = Flux - WRIA1
  - Dakota Creek occupies 6% of WRIA1, enters at 48.9684184, -122.7371679, i = 362, j = 357
  - Terrel Creek occupies 4% of WRIA1, enters at 48.9063480, -122.7649440, i = 351, j = 345
  - Nooksack River occupies 75% of WRIA1, enters at 48.7896486, -122.6667206, i = 321, j = 347
  - Squallum River occupies 5% of WRIA1, enters at 48.7602333, -122.5129287, i = 305, j = 345
  - Lake river thingo occupies 6% of WRIA1, i = 302, j = 367, enters at 48.7520645, -122.4882228
  - Chuckanut occupies 4% of WRIA1, enters at 48.7243762, -122.5068995, i = 298, j = 361
- Fraser River
  - Main South Arm cells based on map are i=414,415,416 and j=334
  - Assumed that 75% goes into the main South Arm (Thomson, 1981)
  - Southern South Arm 5% (aka Canoe Pass, although that is not open), cells i= 409,410 j=315 . This position was corrected in November 2015.
– Main Arm 5%, cells i=434,435, j=318
– North Arm 15%, cells i=440, j=323,324

Skagit

• Source used: WA map
• Includes subwatersheds WRIA3, WRIA4, WRIA5 and WRIA7 from WA map
• WRIA4 represents 33% of Skagit Watershed
  – Baker River and lake drains 8% of WRIA4 but does not enter ocean (24.7)… but Morrison spreadsheet is for coastal runoff therefore assume Skagit River drains 100% of WRIA4 and then enters WRIA3
• WRIA3 represents 17% of Skagit Watershed
  – Skagit River drains 75% of WRIA3
    * assume 50% enters at 48.297470, -122.390614 (i = 207, j = 326)
    * assume 50% enters at 48.370163, -122.497387 (i = 229, j = 319)
  – Samish River drains 20% of WRIA3 (48.569395, -122.472496) (i = 265, j = 348)
  – Joe Leary Slough drains 5% of WRIA 3 (48.520811, -122.484426) (i = 257, j = 339)
• WRIA5 represents 17% of Skagit Watershed
  – Stillaguamish River drains 100% of WRIA5
    * assume 70% enters at 48.196188, -122.371902 (i = 186, j = 316)
    * assume 10% enters at 48.222042, -122.392502 (i = 192, j = 315)
    * assume 20% enters at 48.254513, -122.40263 (i = 200, j = 318)
• WRIA7 represents 33% of Skagit Watershed
  – Snohomish River drains 96% of WRIA7, Quilceda Creek drains 1% of WRIA7, Allen Creek drains 1% of WRIA7, all enter at 48.028853, -122.212429 (i = 143, j = 318)
  – Tulalip Creek drains 1% of WRIA7 (48.064343, -122.284926) (i = 154, j = 311)
  – Mission Creek drains 1% of WRIA7 (48.056656, -122.274742) (i = 152, j = 312)

EVI_N

• Source used: Toporama map at 1:100,000 (CA). Areas (other than Oyster, Campbell, Sayward) were estimated from print outs of the graphs. Large rivers areas are given in Environment Canada (EC) database. Note that not all of Morrison’s EVI_N drains into our model.

Percent drained from each river was proportional to its drainage area versus the total drainage area given by Morrison. Below, rivers marked with ‘+’ means I made up the name.

• Oyster River is at 705,122: area 363 km2 (according to EC) - corrected in November 2015 to move it off land
• Quinism River flows into Campbell River. #08HD003 is Campbell River at the town, drainage area 1470 km2 (EC) Latitude: 50.0353065 Longitude: -125.2629857, Lat/Lon of mouth -125.2601, 50.0510 for the mouth gives two points 123 749 and 750
• Snowden Creek 139 km2 CA Latitude: 50.1125819 Longitude: -125.3723186 point 117 770
• Menzies Creek 31 km² CA Latitude: 50.1383412 Longitude: -125.3908277 point 117 773
• Creek 1+ 23 km² CA Latitude: 50.1952619 Longitude: -125.3825383 point 123 786
• Creek 2+ 16 km² CA Latitude: 50.2333137 Longitude: -125.3975887 point 126 795
• Creek 3 23 km² CA Latitude: 50.2435598 Longitude: -125.3991498 point 127 798
• Elk Creek+ 23 km² CA Latitude: 50.2819399 Longitude: -125.4402655 point 127 807
• Slab Creek+ 12 km² CA Latitude: 50.3063334 Longitude: -125.4381633 point 129 813
• Pye Creek 109 km² CA Latitude: 50.336607 Longitude: -125.5188295 point 121 826
• Bear Point Creek+ 12 km² CA Latitude: 50.3628639 Longitude: -125.6340551 point 107 839
• Amor de Cosmos Creek 229 km² CA Latitude: 50.3582412 Longitude: -125.6876354 point 96 834
• Humpback+ 10 km² CA Latitude: 50.3561671 Longitude: -125.7174369 point 93 844
• Palmer+ 14 km² CA Latitude: 50.3603414 Longitude: -125.7371761 point 92 845
• Hkusam+ 14 km² CA Latitude: 50.3624995 Longitude: -125.7693388 point 87 848
• Camp Point South+ 14 km² CA Latitude: 50.3815933 Longitude: -125.8429452 point 77, 858
• Camp Point North+ 14 km² CA Latitude: 50.3807285 Longitude: -125.8540664 point 78, 858 so done together with Camp Point South
• Salmon River drainage area 1210 km² (EC), #08HD006 is Salmon River near Sayward, Latitude: 50.3930713 Longitude: -125.9514349 points 64 866-867
• Sayward+ 14 km² CA Latitude: 50.388379 Longitude: -125.9592292 point 64, 866
• Kelsey+ 10 km² CA Latitude: 50.4122688 Longitude: -125.9864134 point 62 872
• double rivers Communication+ 7 km² CA Latitude: 50.4253357 Longitude: -126.0181504 & Latitude: 50.4292907 Longitude: -126.0299097 points 59, 877 and 58, 879 put both in at 59, 878
• unmarked+ 7 km² CA Latitude: 50.4427467 Longitude: -126.0688793 point 54 884
• Newcastle+ 34 km² CA Latitude: 50.4526841 Longitude: -126.1194916 point 47 890
• Windy+ 10 km² CA Latitude: 50.4552649 Longitude: -126.1585149 point 42 893

**Howe**

• Source used: BCCF information to determine amount coming from Burrard Inlet. This is a region we could return to using the Toporama maps.
• Squamish River is 90% of watershed (i = 532 and j=385, j=386)
• Burrard Inlet is 10% of watershed (i=457-459 and j=343)

**Bute**

• Source used: Numerous sources on major rivers in the region including wikipedia entries, tourist agencies etc. This is a region we could return to using the Toporama maps.
• Assume Homathko is 58% of watershed (i=897,j=294)
• Assume Southgate is 35% of watershed (i=885,j=296-297)
• Assume Orford is 7% of watershed (i=831, j=249)
Puget

- Source used: WA map
- Includes subwatersheds WRIA17, WRIA16, WRIA15, WRIA14, WRIA08, WRIA09, WRIA10, WRIA12 and WRIA11 from WA map
- WRIA17 10% of Puget Sound Watershed
  - Johnson 5% of WRIA17, 48.061231, -123.039665, i = 207, j = 202
  - Jimmycomelately 5% of WRIA17, 48.028911, -123.004131, i = 199, j = 202
  - Salmon and Snow 25% of WRIA17, 47.997331, -122.873926, i = 182, j = 219
  - Chimacum 20% of WRIA17, 48.048939, -122.769771, i = 185, j = 240
  - Thorndike 5% of WRIA17, 47.808831, -122.739944, i = 137, j = 215
  - Torboo 5% of WRIA17, 47.843407, -122.812986, i = 149, j = 208
  - Little Quilcene/Big Quilcene 35% of WRIA17, 47.813846, -122.854614, i = 146, j = 199
- WRIA16 10% of Puget Sound Watershed
  - Dosewalips 20% of WRIA16, 47.681628, -122.893496, i = 124, j = 177
  - Duckabush 14% of WRIA16, 47.645094, -122.92973, i = 119, j = 167
  - Fulton 2% of WRIA16, 47.616376, -122.973876, i = 116, j = 156
  - Waketick 2% of WRIA16, 47.557241, -123.023751, i = 108, j = 141
  - Hamma Hamma 14% of WRIA16, 47.548001, -123.038936, i = 107, j = 139
  - Jorsted 2% of WRIA16, 47.527069, -123.049386, i = 104, j = 135
  - Eagle 2% of WRIA16, 47.484004, -123.076165, i = 98, j = 127
  - Lilliwaup 2% of WRIA16, 47.462407, -123.113351, i = 95, j = 118
  - Finch 2% of WRIA16, 47.406308, -123.138102, i = 87, j = 108
  - Skokomish 40% of WRIA16, 47.345802, -123.121719, i = 75, j = 103
- WRIA15 15% of Puget Sound Watershed
  - Rendsland 2.5% of WRIA15, 47.385624, -123.114982, i = 81, j = 107
  - Tahuya 20% of WRIA15, 47.36842, -123.052325, i = 72, j = 114
  - Mission 5% of WRIA15, 47.428697, -122.873712, i = 73, j = 149
  - Union 10% of WRIA15, 47.437899, -122.854443, i = 74, j = 153
  - Coulter 5% of WRIA15, 47.400179, -122.821827, i = 64, j = 153
  - Minter 5% of WRIA15, 47.358072, -122.690935, i = 46, j = 168
  - Butley 5% of WRIA15, 47.380568, -122.633307, i = 47, j = 178
  - Olalla 5% of WRIA15, 47.42125, -122.54071, i = 48, j = 197
  - Blackjack 5% of WRIA15, 47.545278, -122.627292
  - Clear 5% of WRIA15, 47.64735, -122.686901
  - Barker 2.5% of WRIA15, 47.636998, -122.674971
  - Big Valley 10% of WRIA15, 47.736812, -122.653127
– Assume 50% of Blackjack+Clear+Barker+BigValley enters Puget Sound at i = 68, j = 210
– Assume 50% of Blackjack+Clear+Barker+BigValley enters Puget Sound at 47.724083,-122.551725 i = 108, j = 232
– Big Bear 5% of WRIA15, 47.657482,-122.785542, i = 112, j = 189
– Swaback 2.5% of WRIA15, 47.638589,-122.835217, i = 112, j = 182
– Stavis 2.5% of WRIA15, 47.632595,-122.868519, i = 113, j = 174
– Anderson 5% of WRIA15, 47.567261,-122.97143, i = 107, j = 150
– Dewatta 5% of WRIA15, 47.452208,-123.058977, i = 94, j = 122
– WRIA14 5% of Puget Sound Watershed
  – Sherwood 15% of WRIA14, 47.378098,-122.828994, i = 60, j = 149
  – Deer 10% of WRIA14, 47.250193,-123.026683
  – Johns 10% of WRIA14, 47.243843,-123.043656
  – Goldborough 15% of WRIA14, 47.210765,-123.089018
  – Mill 15% of WRIA14, 47.19779,-122.99336
  – Skookum 10% of WRIA14, 47.136374,-123.075929
  – Kennedy 10% of WRIA14, 47.057873,-123.006234
  – Schneider 5% of WRIA14, 47.057932,-122.998338
  – Perry 10% of WRIA14,
  – 50% of Deer+Johns+Goldborough+Mill+Skookum+Kennedy+Schneider enter Puget Sound at 47.289476,-122.894711, i = 47, j = 130
  – 50% of Deer+Johns+Goldborough+Mill+Skookum+Kennedy+Schneider +100% of Perry enter Puget Sound at 47.166609,-122.861266, i = 20, j = 120
– WRIA13 3% of Puget Sound Watershed
  – McClane 10% of WRIA13
  – Deschutes 70% of WRIA13
  – Woodward 10% of WRIA13
  – Woodland 10% of WRIA13
  – Assume McClane+Deschutes+Woodward+Woodland enter Puget Sound at 47.182713,-122.83659, i = 22, j = 121
– WRIA12 2% of Puget Sound Watershed
  – Chambers 100% of WRIA12 47.187438,-122.584419, i = 6, j = 162
– WRIA11 15% of Puget Sound Watershed
  – Nisqually 99.5% of WRIA11 47.099227,-122.701149
  – McAllister 0.5% of WRIA11 47.098233,-122.723994
  – Assume Nisqually+McAllister enter Puget Sound at i = 0, j = 137
– WRIA10 20% of Puget Sound Watershed
  – Puyallup 99.5% of WRIA10 47.269678,-122.428036, i = 10, j = 195

2.8. Rivers
– Hylebas 0.5% of WRIA10 47.284935,-122.410011, i = 13, j = 199

• WRIA9 10% of Puget Sound Watershed
  – Duwamish 100% of WRIA9, 50% 47.586831,-122.361259, (i = 68, j = 243) 50% 47.592099,-122.344866 (i = 68, j = 246)

• WRIA8 10% of Puget Sound Watershed
  – Cedar/Sammamish 100% of WRIA8, 47.672894,-122.409207, i = 88, j = 246

JdF

• Source used: BCCF map and Toporama map

• The Juan de Fuca watershed in Morrison et al (2011) includes the north side of Juan de Fuca Strait from Victoria to Port Renfrew (inclusive) and the south side of Juan de Fuca Strait from Cape Flattery to Port Townsend.

• Assume that 50% of the area of the JdF watershed defined by Morrison et al (2011) is on north side of JdF (Canada side):
  – From bccf map, assume 33% of Canada side is part of San Juan River/Harris Creek watershed
    * San Juan River (in the steelhead map) (includes Harris Creek from the steelhead map) 48.560449,-124.404595 (i = 402, j = 56)
  – Assume that 14% of Canada side is in the Gordon River Watershed 48.575897,-124.415281 (i = 403, j = 56)
  – Assume that 20% of Canada side is in Muir/Loss/Tugwell/Jordan
    * Loss Creek (5% of Canada side) 48.480062,-124.27331 (i = 375, j = 71)
    * River Jordan (5% of Canada side) 48.421255,-124.056244 (i = 348, j = 96)
    * Muir Creek (5% of Canada side) 48.378744,-123.867352 (i = 326, j = 119)
    * Tugwell Creek (5% of Canada side) 48.375024,-123.853737 (i = 325, j = 120)
  – Assume that 33% of Canada side is in Sooke River Watershed 48.383846,-123.700011 (i = 308, j = 137)

• Assume that 50% of the area of the watershed defined by Morrison et al (2011) is on south side of JdF (US side)
  – Assume that 60% of US side of JdF is occupied by watershed WRIA 18, two main rivers Elwha and Dungeness
    * Elwha River 48.14616,-123.567095 (50% of watershed WRIA 18) 48.148193,-123.565807 (i = 261, j = 134)
    * Tumwater Creek (1% of watershed WRIA 18) 48.124708,-123.445626 (i = 248, j = 151)
    * Valley Creek (1% of watershed WRIA 18) 48.122445,-123.437018 (i = 247, j = 152)
    * Ennis Creek (2% of watershed WRIA 18) 48.117202,-123.405132 (i = 244, j = 156)
    * Morse Creek (7% of watershed WRIA 18) 48.117861,-123.354084 (i = 240, j = 164)
    * Bagley Creek (2% of watershed WRIA 18) 48.114344,-123.340791 (i = 239, j = 165)
    * Siebert Creek (2% of watershed WRIA 18) 48.120669,-123.289497 (i = 235, j = 174)
    * McDonald Creek (3% of watershed WRIA 18) 48.12561,-123.220167 (i = 233, j = 183)
    * Matriotti Creek (2% of watershed WRIA 18) Dungeness River (30% of watershed) are at same grid point 48.154520, -123.130217 (i = 231, j = 201)
– Assume that 40% of US side of JdF is occupied by watershed WRIA 19 Lyre-Hoko
  * Coville Creek (5% of watershed WRIA 19) 48.138342,-123.611684 (i = 263, j = 128)
  * Salt Creek (5% of watershed WRIA 19) 48.16328,-123.70481 (i = 275, j = 116)
  * Field Creek (5% of watershed WRIA 19) 48.154406,-123.8150554 (i = 281, j = 100)
  * Lyre River (20% of watershed WRIA 19) at 48.160675, -123.828499 (i = 283, j = 98)
  * East Twin River/West Twin River (5% of watershed WRIA 19) 48.165957,-123.949835 (i = 293, j = 81)
  * Deep Creek (5% of watershed WRIA 19) 48.175316,-124.026289 (i = 299, j = 72)
  * Pysht River (10% of watershed WRIA 19) 48.204541,-124.095984 (i = 310, j = 65)
  * Clallom River (10% of watershed WRIA 19) 48.254713,-124.267824 (i = 333, j = 45)
  * Hoko River (20% of watershed WRIA 19) 48.287419, -124.362191 (i = 345, j = 35)
  * Sekiu River (10% of watershed WRIA 19) 48.288676,-124.394159 (i = 348, j = 31)
  * Sail River (5% of watershed WRIA 19) 48.360327,-124.556508 (i = 373, j = 17)

EVI_S

• Source used: BCCF map and fluxes. Could return here using the Toporama maps.
• Total flux (according to Morrison is 329.5 m3/s. Adding fluxes and areas to estimate fluxes as given below gave 292.6 so values were multiplied up to make the difference
• Koksilah flows into Cowichan (9.77 m3/s)
• Cowichan (55 m3/s + Koksilah) gives 22% of watershed (i=383, j=201,202)
• Chemanius 19.2 m3/s and by area, north and south of Chemanius another 1/2. Mouths are split: o Chemanius1 6.5% of watershed (i=414, j=211) o Chemanius2, 6.5% of watershed of watershed (i=417, j=212)
• Nanaimo 39.7 m3/s. Mouths are split: o Nanaimo1, 9.4% of watershed of watershed (i=478, j=208, 209) o Nanaimo2, 4.6% of watershed of watershed (i=477, j=210)
• NorNanaimo, North of Naniamo and area of 1/2 Little Qualicum gives 2% of watershed (i=491-493, j=213)
• Goldstream, 2.2 m3/s gives 8% of watershed (i=334, j=185)
• Nanoose, area of 1/2 Little Qualicum gives 2% of watershed (i=518, j=185)
• Englishman, 14 m3/s gives 5% of watershed (i=541, j=175)
• FrenchCreek, area of 1/2 of Qualicum gives 1% of watershed (i=551, j=168)
• LittleQualicum, 11.8 m3/s plus 1/2 of Qualicum in nearby area gives 5% of watershed (i=563, j=150)
• Qualicum, 7.3 m3/s gives 2% of watershed (i=578, j=137)
• SouthDenman, about the size of Tsable + Qualicum gives 5% of watershed (i=602, j=120)
• Tsable, 7.99 m3/s but double for surrounding region 3% of watershed (i=616-617, j=120)
• Trent, 3 m3/s gives 1% of watershed (i=648, j=121)
• Puntledge, 42 m3/s gives 14% of watershed (i=656, j=119-120)
• BlackCreek, 1.8 m3/s plus area of Qualicum gives 3% of watershed (i=701, j=123)
Jervis

• Source used: Toporama map,
• See this site

• As there were no gauged rivers in the Jervis Inlet watershed, Trites (1955) estimated the freshwater discharge using the area of the watershed (~1400 km²) and local precipitation data. The estimated mean annual discharge of 180 m³ s⁻¹ is considerably smaller than the discharge in most of the longer BC inlets. Unlike many of the BC inlets where the main river enters at the head, there are many small rivers and streams distributed along the shores of Jervis Inlet. The runoff cycle for Jervis Inlet more closely follows the local precipitation cycle as the area of snow fields which store winter precipitation is relatively small (Macdonald and Murray 1973).

• Pickard (1961) (http://www.nrcresearchpress.com/doi/pdf/10.1139/f61-062): The chief difference between these inlets is that Jervis has less than 40% as much river runoff as Bute, and only one-quarter of this comes in at the head whereas in Bute three-quarters of the total enters at the head. The flushing effect of the large runoff into the head of Bute is expected to be greater on the whole of the inlet length than that of the smaller runoff distributed along the length of Jervis.

• Flow out of Powell Lake taken from Sanderson et al. (1986)

Jervis Inlet only area = 1400km² (Trites 1955) ==> 25% of Jervis watershed (5785km²)

• Assume Skwawaka/Hunaechin/Lausmann/Slane/Smanit/ account for 30% of Jervis only watershed

• Assume Loquilts accounts for 4% of Jervis only, enters at 50.204868,-123.77326 (ish) i = 650, j = 318

• Assume Potato Creek accounts for 4% of Jervis only, enters at 50.154741,-123.837075

• Assume Deserted River accounts for 10% of Jervis only, enters at 50.0922,-123.745022

• Assume Stakawus Creek accounts for 4% of Jervis only, enters at 50.074273,-123.776457

• Crabapple Creek accounts for 4% of Jervis only, enters at 50.1207422, -123.8436382

• Osgood Creek accounts for 4% of Jervis only, enters at 50.0371886, -123.8964722

• Skwawka/Hunaechin/Lausmann/Slane/Smanit/Loquilts/Potato/Deserted/Crabapple/Stakawus/Osgood all enter domain at the same point 50.0894746,-123.7828011, i = 650, j = 309 - moved off land in November 2015

• Glacial Creek accounts for 5% of Jervis only, enters at 50.0062107, -123.9070838, i = 649, j = 310 - moved off land in November 2015

• Seshal Creek accounts for 5% of Jervis only, enters at 50.0246890, -123.9260495, i = 651, j = 307 - moved off land in November 2015

• Brittain River/Treth Creek accounts for 10% of Jervis only, enters at 49.9958119. -124.0119219, i = 650, j = 301

• Assume Vancouver River/High Creek accounts for 10% of Jervis only and enter at 49.9219882, -123.8696986, i = 626, j = 311

• Assume Perketts Creek accounts for 5% of Jervis only and enters at 49.8799903, -123.8681308, i = 619, j = 307

• Assume Treat Creek accounts for 5% of Jervis only and enters at 49.8423159, -123.8742022, i = 612, j = 301

• Sechelt is about 66% of Jervis Inlet, based on values in Table II of Pickard (1961) (110m³/s / 180m³/s) ==> 17% of Jervis watershed

• Sechelt Inlet isn’t in the domain, assume the input enters at 49.770844,-123.955708, i = 604, j = 280

• Outflow from Powell Lake is 3e9m³/year (Sanderson et al 1986) ==> 32% of Jervis watershed, enters at 49.874421,-124.565288, i = 666, j = 202
• From Section 4 of this report (http://www.powellriverrd.bc.ca/wp-content/uploads/2011/09/Community-Profile.pdf),
  Lois Lake drains 45,000ha = 450km² ==> 8% of Jervis watershed... make it 10% to account for little rivers nearby, enters at 49.771481, -124.332197, i = 629, j = 224

• From Section 4 of this report (http://www.powellriverrd.bc.ca/wp-content/uploads/2011/09/Community-Profile.pdf),
  Haslam Lake drains 13,140ha = 131km² ==> 2% of Jervis watershed, enters at 49.77356, -124.367173, i = 632, j = 219
  estimate Chapman Creek drains about 2% of the catchment, enters at 49.4381655, -123.7229658, i = 522, j = 273
  estimate Lapan Creek drains about 2% of the catchment, enters at 49.8368204, -123.9942065, i = 619, j = 282
  estimate Nelson Island represents 2% of the catchment and this drains from West Lake, into the domain at 49.7350557, -124.0575565, i = 599, j = 257
  estimate Wakefield Creek represents 2% of the catchment, into the domain at 49.4673394, -123.8048516, i = 533, j = 263
  estimate Halfmoon Creek represents 2% of the catchment, into the domain at 49.5103863, -123.9119698, i = 549, j = 253
  estimate Myers/Kleindale/Anderson represent 4% of catchment, into the domain at 49.6340820, -123.9952235, i = 571, j = 248

Toba

• Source used: Toporama Maps
  Toba River 50% at 50.492 124.365
  Theodosia River 12% at 50.080 124.66
  Quatam River 9% at 50.380 124.942
  Brem River 9% at 50.435 124.663
  Tahumming River 8% at 50.493 124.387
  Racine Creek (with neighbours) 4% at 50.399 124.555
  Homfray Creek (wn) 3% at 50.293 124.635
  Forbes Creek wn 3% at 50.242 124.591
  Chusan Creek wn 2% at 50.473 124.381

Temperature

Temperature records are available for a number of the Rivers but at this point we are only using the record from Hope (originally from Water Office data, as compiled by Allen and Wolfe, 2013).

References


• Sanderson et al (1986)

• Thomson, 1982: Physical Oceanography of the BC Coast.

2.8.2 River Implementation

Creating input files for NEMO

The grid point of the location of each river mouth was found. A large dictionary of the grid points and flow mapping is the rivertools.py script. The Jupyter Notebook Add Rivers Month and Constant.ipynb creates a NetCDF files con-
taining the river flow at the respective grid cell for each river throughout the domain. Where the river mouth was not included in the domain, the river was added to the closest grid point to the river mouth.

In some cases (e.g. the end of Jervis inlet, Puget Sound) numerous rivers were not included in the domain, so the sum of all the omitted rivers’ flow was added to the closest grid point.

Everywhere the depth of freshwater is set to 3 m so that the freshwater is distributed over the top 3 m at the specified grid cell.

Everywhere the temperature of the freshwater is set to the temperature of the Fraser at Hope.

Two files are created rivers_month_201702.nc and rivers_cnst_201702.nc. The first contains the monthly climatology, the second the yearly average.

**Nowcast Files**

For the nowcast runs, the Fraser River is split into the amount that enters at Hope and amount downstream of that. The amount at Hope is taken from the published measurements (Environment Canada, WaterOffice via data mart) and distributed appropriately to the Fraser River mouths. All other freshwater is set to climatology.

### 2.9 Stability

This page outlines our investigation into code stability.

#### 2.9.1 Stability: Introduction

As of January 2014, the code was running, but during the spin-up runs and other tests it became obvious that the code was only marginally stable. We increased the vertical mixing coefficient under static-instability, which helped greatly, but we still needed to run with larger horizontal viscosity than we felt was appropriate. Code is stable with horizontal viscosity = 100 m2/s but is not stable with 50 m2/s. We felt 20 m2/s, like they are using on the East Coast, would be better.

So the two choices appeared to be: a) increase the viscosity or b) smooth the bathymetry.

As the OSM a talk by Lemarie suggested that small scale codes are often stability limited by the vertical advection. So I wished to evaluate:

1) our velocities, are they as large as we expect and
2) is the code subject to a vertical CFL condition and if so, are we meeting it.

#### 2.9.2 Stability: Evaluation of Velocities

Are the velocities in the model similar to those in the field. For field velocities, we will use ADCP data from VENUS at the two Strait of Georgia Stations: East and Central. East is at 49o2.531N, 123o19.055’W which is close to python grid points [416,282]. Central is at 49o2.4083N, 123o25.5287W which is close to python grid points [424,267]

To do this, we ran the model for Oct 18, 2002 and output average 1-hr velocities throughout the day. Of course, VENUS was not running in 2002. We found a similar day (slightly higher amplitudes at Point Atkinson) in Oct 15, 2013. Note that there is a shift of 0.05 days between the two for best match. These tidal predictions are downloaded from: Integrated Science Data Management.

Model velocities at these two points at various depths are given in the following two plots.

Actual ACDP velocities (for Oct 15, 2013) are stronger by 20-40%.
In NEMO advection is through the leap-frog scheme. (NEMO book section 3.1 and 6.2) Thus it should be subject to a vertical CFL condition.

Vertical velocities in the model are high. Using a one day run for Oct 18, 2002 (near neap tides), with one-hour averaged output, vertical velocities as high as 18 cm/s were found. Plot below is for level 28 (160 m) at 21 hours in the Boundary Pass region.

One can calculate the time for the flow to cross one grid cell vertically using the maximum velocity at each depth over the whole horizontal domain and all 24 hours.

Note that the above is log plot. The minimum value is 21.5 s. Given that the CFL condition is 1/2 of this for a leapfrog scheme with staggered grids, that would imply a CFL time step of 10.75 s or less!
Testing

The code was run with baroclinic time step of 10s, and the barotropic time step unchanged at 2s. Started Oct 26, with density fields from spin up. Ran 1 day on Salish and a further 4 days on Jasper with horizontal viscosity of 20 m2/s and vertical edd at 20 m2/s. The code was stable and the output looks reasonable. Velocities and mixing increased, particularly in the island. Current widths decreases (deep current into SoG for example).

Code is now slow! The four days on Jasper took over 4 hours. The day on Salish took 8 hours.

Tried with a baroclinic time step of 16s, and the barotropic time step unchanged at 2s. Repeated Oct 26. Noise seen in surface height as in the large dt cases. So need to be smaller.

2.10 SMELT Documentation

This section documents the development of Salish Model Ecosystem - Lower Trophic (SMELT), an ecosystem model for the Salish Sea. SMELT was derived from the 1-D SOG model and implemented in 3-D, coupled to NEMO, using PISCES as a template.
SMELT is implemented as a configuration of NEMO version 3.6 (revision 6770 as of March 8, 2017). The base NEMO 3.6 code and associated documentation can be accessed through the NEMO official site: http://www.nemo-ocean.eu/

Files that are modified from the base code are stored within our SMELT configuration directory in the MY_SRC subdirectory. These files are not currently publicly available.

### 2.10.1 Transition from SOG/PISCES to SMELT

The biological model dynamics are adapted from the 1-d SOG model with some modifications for the 3-d domain, as described in the following sections.

**Differences in Mesozooplankton Closure implementation**

It was necessary to adapt the implementation of the mesozooplankton closure term to three dimensions. In SMELT, the seasonally-varying mesozooplankton population is distributed in three dimensions so that its abundance is proportional to its food source.

**Parameter definitions**

Parameter names have been adapted as described in the tables linked below.

**Namelist: nampisprod**

<table>
<thead>
<tr>
<th>SOG Name</th>
<th>SMELT Name</th>
<th>Description</th>
<th>SOG Value</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate_micro%R</td>
<td>zz_rate_R_diat</td>
<td>micro-phyto maximum growth rate</td>
<td>0.600E-04</td>
<td>1/s</td>
<td>Hitchcock 1980: 1.4 d-1 for T. nordsenkelii at 10 degrees</td>
</tr>
<tr>
<td>rate_nano%R</td>
<td>zz_rate_R_myri</td>
<td>M. rubrum maximum growth rate</td>
<td>0.250E-04</td>
<td>1/s</td>
<td>Yih 2004: 0.5 d-1 for Mesodinium rubrum (at 15 degrees) and Q10<strong>2 tuned up and at Q10</strong>1</td>
</tr>
<tr>
<td>rate_pico%R</td>
<td>zz_rate_R_nano</td>
<td>nanoflagellate maximum growth rate</td>
<td>0.250E-04</td>
<td>1/s</td>
<td></td>
</tr>
<tr>
<td>rate_micro%max temp</td>
<td>zz_rate_maxtemp</td>
<td>high temperature limit</td>
<td>35</td>
<td>deg C</td>
<td>Durbin 1974: T. nordsenkelii cannot grow at 18 degrees; general diatoms increase this</td>
</tr>
<tr>
<td>rate_nano%max temp</td>
<td>zz_rate_maxtemp</td>
<td>high temperature limit</td>
<td>35</td>
<td>deg C</td>
<td>Mesodinium, Cloern 1977</td>
</tr>
<tr>
<td>SOG Name</td>
<td>SMELT Name</td>
<td>Description</td>
<td>SOG Value</td>
<td>Units</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>------------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>rate_pico%maxtemp</td>
<td>zz_rate_maxtemp</td>
<td>high temperature limit</td>
<td>35</td>
<td>deg C</td>
<td>equal growth at 10 and 15 so range down 8 (which Q10 should be about const)</td>
</tr>
<tr>
<td>rate_micro%temprange</td>
<td>zz_rate_temprange</td>
<td>growth limit range below max temperature</td>
<td>5</td>
<td>deg C</td>
<td>starts turning over earlier</td>
</tr>
<tr>
<td>rate_nano%temprange</td>
<td>zz_rate_temprange</td>
<td>growth limit range below max temperature</td>
<td>5</td>
<td>deg C</td>
<td></td>
</tr>
<tr>
<td>rate_pico%temprange</td>
<td>zz_rate_temprange</td>
<td>growth limit range below max temperature</td>
<td>5</td>
<td>deg C</td>
<td></td>
</tr>
<tr>
<td>rate_micro%lopt</td>
<td>zz_rate_lopt_diat</td>
<td>micro-phyto growth optimal light level</td>
<td>42</td>
<td>W/m^2</td>
<td>Durbin 1974: 3.3 lg/s for T. nordskelii at 10 degrees</td>
</tr>
<tr>
<td>rate_nano%lopt</td>
<td>zz_rate_lopt_myri</td>
<td>M. rubrum growth optimal light level</td>
<td>37</td>
<td>W/m^2</td>
<td>Yih plus Cloern 1977 Half-sat very similar</td>
</tr>
<tr>
<td>rate_pico%lopt</td>
<td>zz_rate_lopt_myri</td>
<td>pico-phyto growth optimal light level</td>
<td>10</td>
<td>W/m^2</td>
<td></td>
</tr>
<tr>
<td>rate_micro%gamma</td>
<td>zz_rate_gamma_diat</td>
<td>growth light limitation loss parameter</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%gamma</td>
<td>zz_rate_gamma_myri</td>
<td>growth light limitation loss parameter</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%gamma</td>
<td>zz_rate_gamma_nano</td>
<td>growth light limitation loss parameter</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%K_Si</td>
<td>zz_rate_K_Si_diat</td>
<td>Si half saturation constant</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%K_Si</td>
<td>zz_rate_K_Si_myri</td>
<td>Si half saturation constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%K_Si</td>
<td>zz_rate_K_Si_nano</td>
<td>Si half saturation constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%gamma_o</td>
<td>zz_rate_gamma_o_diat</td>
<td>exponential NH inhibition of NO3 uptake</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%gamma_o</td>
<td>zz_rate_gamma_o_myri</td>
<td>exponential NH inhibition of NO3 uptake</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 4 – continued from previous page

<table>
<thead>
<tr>
<th>SOG Name</th>
<th>SMELT Name</th>
<th>Description</th>
<th>SOG Value</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate_pico%gamma</td>
<td>zz_rate_gamma_diat</td>
<td>exponential NH inhibition of NO3 uptake</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%kapu</td>
<td>zz_rate_kapa_diat</td>
<td>preference for NO3 over NH factor</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%kapu</td>
<td>zz_rate_kapa_myri</td>
<td>preference for NO3 over NH factor</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%kapu</td>
<td>zz_rate_kapa_nano</td>
<td>preference for NO3 over NH factor</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%k</td>
<td>zz_rate_k_diat</td>
<td>NO3 half-saturation constant</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%k</td>
<td>zz_rate_k_myri</td>
<td>NO3 half-saturation constant</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%k</td>
<td>zz_rate_k_nano</td>
<td>NO3 half-saturation constant</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%N_x</td>
<td>zz_rate_N_x_diat</td>
<td>N inhibition exponent</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%N_x</td>
<td>zz_rate_N_x_myri</td>
<td>N inhibition exponent</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%N_x</td>
<td>zz_rate_N_x_nano</td>
<td>N inhibition exponent</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%N_o</td>
<td>zz_rate_N_o_diat</td>
<td>overall half-saturation constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%N_o</td>
<td>zz_rate_N_o_myri</td>
<td>overall half-saturation constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%N_o</td>
<td>zz_rate_N_o_nano</td>
<td>overall half-saturation constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_micro%Si_ratio</td>
<td>zz_rate_Si_ratio_diat</td>
<td>micro-phyto silicon/nitrogen ratio</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_nano%Si_ratio</td>
<td>zz_rate_Si_ratio_myri</td>
<td>silicon/nitrogen ratio</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rate_pico%Si_ratio</td>
<td>zz_rate_Si_ratio_nano</td>
<td>silicon/nitrogen ratio</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.10.2 Open Boundary Conditions for Tracers

Initially, several adaptations to the NEMO code were necessary in order to use BDY open boundary conditions on tracers. These were carried out using the NEMO development branch, dev_r5144_CMCC5_BDY_for_TOP, as a template. The ticket associated with that branch has since been closed.
**Code Changes**

Currently, our code differs from the distribution in the following ways. The output of diff operations between our configuration version and the standard version of each file is shown.

**MY_SRC$ diff bdy_oce.F90 ../../../NEMO/OPA_SRC/BDY/bdy_oce.F90**

```fortran
73,78d72
< #if defined key_top
< CHARACTER(LEN=20) :: cn_obc !: type of boundary condition to
< apply
< REAL(wp) :: rn_fac  !: multiplicative scaling factor
< REAL(wp), POINTER, DIMENSION(:, :) :: trc  !: now field of the tracer
< LOGICAL :: dmp  !: obc damping term
< #endif
111d104
< REAL(wp), DIMENSION(jp_bdy) :: rn_max_sponge  !: Maximum viscosity,
< for sponge (m2/s)
```

**MY_SRC$ diff trc.F90 ../../../NEMO/TOP_SRC/trc.F90**

```fortran
16,18d15
< #if defined key_bdy
< USE bdy_oce, only: nb_bdy, OBC_DATA
< #endif
96,100d92
< LOGICAL :: llobc  !: read in a file or not
< LOGICAL :: llsbc  !: read in a file or not
< LOGICAL :: llcbc  !: read in a file or not
< !#if defined key_bdy
201,209d192
< CHARACTER(len=20), PUBLIC, ALLOCATABLE, SAVE, DIMENSION(:) :: cn_trc_dflt
< CHARACTER(len=20), PUBLIC, ALLOCATABLE, SAVE, DIMENSION(:) :: cn_trc
< INTEGER, PUBLIC, ALLOCATABLE, SAVE, DIMENSION(:) :: nn_trcdmp_bdy
< ! External data structure of BDY for TOP. Available elements: cn_obc, ll_trc,
< trcnow, dmp
< !#endif
232,238c215,216
< & ln_trc_ini(jptra) , ln_trc_wri(jptra) , qsr_mean(jpi,jpj)
< & ln_trc_sbc(jptra) , ln_trc_cbc(jptra) , ln_trc_obc(jptra)
< & cn_trc_dflt(nb_bdy) , cn_trc(nb_bdy) , nn_trcdmp_bdy(nb_
< bdy)
< & trcdta_bdy(jptra,nb_bdy)
< !#endif
< STAT = trc_alloc )
< ---
< > ln_trc_ini(jptra) , ln_trc_wri(jptra) , qsr_mean(jpi,jpj)
< ... STAT = trc_alloc )
```

(continues on next page)
We also use a modified trcbc.F90 file and additional file trcbdy.F90 (March 8, 2017 versions).
MODULE trcbc
!!!======================================================================
!! *** MODULE trcdta ***
!! TOP : module for passive tracer boundary conditions
!!======================================================================
!!----------------------------------------------------------------------
#if defined key_top
!!----------------------------------------------------------------------
!! 'key_top' TOP model
!!----------------------------------------------------------------------
!! trc_dta : read and time interpolated passive tracer data
!!----------------------------------------------------------------------
USE par_trc ! passive tracers parameters
USE oce_trc ! shared variables between ocean and passive tracers
USE trc ! passive tracers common variables
USE iom ! I/O manager
USE lib_mpp ! MPP library
USE fldread ! read input fields
#endif
USE bdy_oce, only: nb_bdy , idx_bdy, ln_coords_file, rn_time_dmp, rn_time_dmp_out
#ifndef defined key_bdy
USE bdy_oce, only: nb_bdy , idx_bdy, ln_coords_file, rn_time_dmp, rn_time_dmp_out
#endif
IMPLICIT NONE
PRIVATE
PUBLIC trc_bc_init ! called in trcini.F90
PUBLIC trc_bc_read ! called in trcstp.F90 or within

INTEGER , SAVE, PUBLIC :: nb_trcobc ! number of tracers with open BC
INTEGER , SAVE, PUBLIC :: nb_trcsbc ! number of tracers with surface BC
INTEGER , SAVE, PUBLIC :: nb_trccbc ! number of tracers with coastal BC
INTEGER , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: n_trc_indobc ! index of tracer with OBC data
INTEGER , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: n_trc_indsbcb ! index of tracer with SBC data
INTEGER , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: n_trc_indcbc ! index of tracer with CBC data
!!INTEGER , SAVE, PUBLIC :: ntra_obc ! MAX( 1, nb_trcxxx ) to avoid compilation error with bounds checking
!!INTEGER , SAVE, PUBLIC :: ntra_sbc ! MAX( 1, nb_trcxxx ) to avoid compilation error with bounds checking
!!INTEGER , SAVE, PUBLIC :: ntra_cbc ! MAX( 1, nb_trcxxx ) to avoid compilation error with bounds checking
REAL(wp) , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: rf_trofac ! multiplicative factor for OBC tracer values
TYPE(FLD), SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:), TARGET :: sf_trcobc ! structure of data input OBC (file informations, fields read)
REAL(wp) , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: rf_trsfac ! multiplicative factor for SBC tracer values
TYPE(FLD), SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: sf_trcsbc ! structure of data input SBC (file informations, fields read)
REAL(wp) , SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: rf_trcfac ! multiplicative factor for CBC tracer values

(continues on next page)
TYPE(FLD), SAVE, PUBLIC, ALLOCATABLE, DIMENSION(:) :: sf_trccbc ! structure of data input CBC (file informations, fields read)
TYPE(MAP_POINTER), ALLOCATABLE, DIMENSION(:) :: nbmap_ptr ! array of pointers to nbmap

!! * Substitutions
#include "domzgr_substitute.h90"
!!----------------------------------------------------------------------
!! $Id: trcbc.F90 5215 2015-04-15 16:11:56Z nicolasmartin $
!! Software governed by the CeCILL licence (NEMOGCM/NEMO_CeCILL.txt)
!!----------------------------------------------------------------------
CONTAINS

SUBROUTINE trc_bc_init(ntrc)
!!----------------------------------------------------------------------
!! *** ROUTINE trc_bc_init ***
!!
!! ** Purpose : initialisation of passive tracer BC data
!! ** Method : - Read namtsd namelist
!! - allocates passive tracer BC data structure
!!----------------------------------------------------------------------
!
INTEGER,INTENT(IN) :: ntrc ! number of tracers
INTEGER :: jl, jn, ib, ibd, ii, ij, ik ! dummy loop indices
INTEGER :: ierr0, ierr1, ierr2, ierr3 ! temporary integers
INTEGER :: ios ! Local integer output
INTEGER :: nblen, igrd ! support arrays for BDY
CHARACTER(len=100) :: clndta, clntrc
CHARACTER(len=100) :: cn_dir_sbc, cn_dir_cbc, cn_dir_obc

TYPE(FLD_N), ALLOCATABLE, DIMENSION(:) :: slf_i ! local array of namelist informations on the fields to read

TYPE(FLD_N), DIMENSION(jpmaxtrc) :: sn_trccbc ! coastal
REAL(wp) , DIMENSION(jpmaxtrc) :: rn_trofac ! multiplicative factor for
REAL(wp) , DIMENSION(jpmaxtrc) :: rn_trsfac ! multiplicative factor for
REAL(wp) , DIMENSION(jpmaxtrc) :: rn_trcfac ! multiplicative factor for
NAMELIST/namtrc_bc/ cn_dir_sbc, cn_dir_cbc, sn_trccbc, rn_trofac, sn_trccbc, rn_

#if defined key_bdy
NAMELIST/namtrc_bdy/ cn_trc_dflt, cn_trc, nn_trcdmp_bdy
NAMELIST/nambdy_bc/ cn_dir_obc, sn_trccbc, rn_trofac
#endif

IF( nn_timing == 1 ) CALL timing_start('trc_bc_init')
! IF( lwp ) THEN
  WRITE(numout,* )'trc_bc_init : Tracers Boundary Conditions (BC)'
  ! ENDIF
WRITE(numout,*) '~~~~~~~~~~~ '

!WRITE(numout,*) "In trc_bc_init at A: nstop=", nstop
ENDIF
!
! Initialisation and local array allocation
ierr0 = 0 ; ierr1 = 0 ; ierr2 = 0 ; ierr3 = 0
ALLOCATE( slf_i(ntrc), STAT=ierr0 )
IF( ierr0 > 0 ) THEN
   CALL ctl_stop( 'trc_bc_init: unable to allocate local slf_i' ) ; RETURN
ENDIF
!
! Compute the number of tracers to be initialised with open, surface and
→boundary data
ALLOCATE( n_trc_indobc(ntrc), STAT=ierr0 )
IF( ierr0 > 0 ) THEN
   CALL ctl_stop( 'trc_bc_init: unable to allocate n_trc_indobc' ) ; RETURN
ENDIF
nb_trcobc = 0
n_trc_indobc(:) = 0
!
ALLOCATE( n_trc_indsbc(ntrc), STAT=ierr0 )
IF( ierr0 > 0 ) THEN
   CALL ctl_stop( 'trc_bc_init: unable to allocate n_trc_indsbc' ) ; RETURN
ENDIF
nb_trcsbc = 0
n_trc_indsbc(:) = 0
!
ALLOCATE( n_trc_indcbc(ntrc), STAT=ierr0 )
IF( ierr0 > 0 ) THEN
   CALL ctl_stop( 'trc_bc_init: unable to allocate n_trc_indcbc' ) ; RETURN
ENDIF
nb_trccbc = 0
n_trc_indcbc(:) = 0
!
!DO jn = 1, ntrc
!   IF( ln_trc_obc(jn) ) THEN
!      nb_trcobc = nb_trcobc + 1
!      n_trc_indobc(jn) = nb_trcobc
!   ENDIF
!   IF( ln_trc_sbc(jn) ) THEN
!      nb_trcsbc = nb_trcsbc + 1
!      n_trc_indsbc(jn) = nb_trcsbc
!   ENDIF
!   IF( ln_trc_cbc(jn) ) THEN
!      nb_trccbc = nb_trccbc + 1
!      n_trc_indcbc(jn) = nb_trccbc
!   ENDIF
!ENDDO
!
!ntra_obc = MAX( 1, nb_trcobc ) ! To avoid compilation error with bounds
→checking
!IF( lwp ) WRITE(numout,*) ' ',
!IF( lwp ) WRITE(numout,*) ' Number of passive tracers to be initialized with
→open boundary data ':, nb_trcobc
!IF( lwp ) WRITE(numout,*) ' ',
!ntra_sbc = MAX( 1, nb_trcsbc ) ! To avoid compilation error with bounds
→checking

(continues on next page)
!IF( lwp ) WRITE(numout,*) ' ' 
!IF( lwp ) WRITE(numout,*) ' Number of passive tracers to be initialized with surface boundary data :', nb_trcsbc 
!IF( lwp ) WRITE(numout,*) ' ' 
!ntra_cbc = MAX( 1, nb_trccbc ) ! To avoid compilation error with bounds checking 
!IF( lwp ) WRITE(numout,*) ' ' 
!IF( lwp ) WRITE(numout,*) ' Number of passive tracers to be initialized with coastal boundary data :', nb_trccbc 
!IF( lwp ) WRITE(numout,*) ' ' 

! Read Boundary Conditions Namelists 
REWIND( numnat_ref ) ! Namelist namtrc_bc in reference namelist : 
   Passive tracer data structure 
      READ ( numnat_ref, namtrc_bc, IOSTAT = ios, ERR = 901) 
901 IF( ios /= 0 ) CALL ctl_nam ( ios , 'namtrc_bc in reference namelist', lwp ) 
   Rewind( numnat_cfg ) ! Namelist namtrc_bc in configuration namelist 
      Passive tracer data structure 
      READ ( numnat_cfg, namtrc_bc, IOSTAT = ios, ERR = 902 ) 
902 IF( ios /= 0 ) CALL ctl_nam ( ios , 'namtrc_bc in configuration namelist', lwp ) 
   IF(lwm) WRITE ( numont, namtrc_bc ) 
   #if defined key_bdy 
      Rewind( numnat_ref ) ! Namelist namtrc_bc in reference namelist : 
         Passive tracer data structure 
         Read ( numnat_ref, namtrc_bdy, IOSTAT = ios, ERR = 903) 
903 IF( ios /= 0 ) CALL ctl_nam ( ios , 'namtrc_bdy in reference namelist', lwp ) 
      Rewind( numnat_cfg ) ! Namelist namtrc_bc in configuration namelist 
         Passive tracer data structure 
         Read ( numnat_cfg, namtrc_bdy, IOSTAT = ios, ERR = 904 ) 
904 IF( ios /= 0 ) CALL ctl_nam ( ios , 'namtrc_bdy in configuration namelist', lwp ) 
   IF(lwm) WRITE ( numont, namtrc_bdy ) 
! set up preliminary informations for BDY structure 
Rewind( numnat_ref ) 
Rewind( numnat_cfg ) 
DO ib = 1, nb_bdy 
DO jn = 1, ntrc 
   ! Set type of obc in BDY data structure (around here we may plug user override of obc type from nml) 
   WRITE(numout,*) "ln_trc_obc(jn)="", ln_trc_obc(jn) 
   WRITE(numout,*) "jn="", jn, " ib="", ib, " cn_trc(ib)=", cn_trc(ib) 
   WRITE(numout,*) "jn="", jn, " ib="", ib, " cn_trc_dflt(ib)=", cn_trc_dflt(ib) 
   IF ( ln_trc_obc(jn)) THEN 
      trcdta_bdy(jn,ib)%cn_obc = TRIM( cn_trc(ib) ) 
   ELSE 
      trcdta_bdy(jn,ib)%cn_obc = TRIM( cn_trc_dflt(ib) ) 
   ENDIF 
   ! set damping use in BDY data structure 
   trcdta_bdy(jn,ib)%dmp = .false. 
   IF(nn_trcdmp_bdy(ib) .EQ. 1 .AND. ln_trc_obc(jn) ) trcdta_bdy(jn,ib)%dmp = .true. 
   IF(nn_trcdmp_bdy(ib) .EQ. 2 ) trcdta_bdy(jn,ib)%dmp = .true. 
   IF(trcdta_bdy(jn,ib)%cn_obc == 'frs' .AND. nn_trcdmp_bdy(ib) .NE. 0 ) 
   &
& CALL ctl_stop( 'Use FRS OR relaxation' )

IF (nn_trcdmp_bdy(ib) .LT. 0 .OR. nn_trcdmp_bdy(ib) .GT. 2)
  THEN
  WRITE(numout,*), "nn_trcdmp_bdy=", nn_trcdmp_bdy(ib), " ib=", ib
  CALL ctl_stop( 'Not a valid option for nn_trcdmp_bdy. Allowed: 0,1,2.' )
  ENDIF
ENDIF
ENDDO
ENDDO

#ifelse

! Force all tracers OBC to false if bdy not used
ln_trc_obc = .false.
#endif

! compose BC data indexes
DO jn = 1, ntrc
  IF( ln_trc_obc(jn) ) THEN
    nb_trcobc = nb_trcobc + 1 ; n_trc_indobc(jn) = nb_trcobc
  ENDIF
  IF( ln_trc_sbc(jn) ) THEN
    nb_trcsbc = nb_trcsbc + 1 ; n_trc_indsbc(jn) = nb_trcsbc
  ENDIF
  IF( ln_trc_cbc(jn) ) THEN
    nb_trccbc = nb_trccbc + 1 ; n_trc_indcbc(jn) = nb_trccbc
  ENDIF
ENDDO

! print some information for each
! IF( lwp ) THEN
!   DO jn = 1, ntrc
!     IF( ln_trc_obc(jn) ) THEN
!       clndta = TRIM( sn_trcobc(jn)%clvar )
!       IF( lwp ) WRITE(numout,*), 'Preparing to read OBC data file for passive tracer number :', jn, ' name : ', clndta, 
!       & ' multiplicative factor : ', rn_trofac(jn)
!     ENDIF
!     IF( ln_trc_sbc(jn) ) THEN
!       clndta = TRIM( sn_trcsbc(jn)%clvar )
!       IF( lwp ) WRITE(numout,*), 'Preparing to read SBC data file for passive tracer number :', jn, ' name : ', clndta, 
!       & ' multiplicative factor : ', rn_trsfac(jn)
!     ENDIF
!     IF( ln_trc_cbc(jn) ) THEN
!       clndta = TRIM( sn_trccbc(jn)%clvar )
!       IF( lwp ) WRITE(numout,*), 'Preparing to read CBC data file for passive tracer number :', jn, ' name : ', clndta, 
!       & ' multiplicative factor : ', rn_trcfac(jn)
!     ENDIF
!   END DO
! ENDIF
!
! The following code is written this way to reduce memory usage and repeated for each boundary data
! MAV: note that this is just a placeholder and the dimensions must be changed according to what will be done with BDY. A new structure will probably need to be included
!

(continues on next page)
! OPEN Lateral boundary conditions
! Print summary of Boundary Conditions
IF( lwp ) THEN
WRITE(numout,*') ' Total tracers to be initialized with SURFACE BCs_'
 WRITE(numout,*) ' data:', nb_trcsbc
 IF ( nb_trcsbc > 0 ) THEN
 WRITE(numout,*) ' #trc NAME Boundary Mult.Fact.'
 DO jn = 1, ntrc
 IF ( ln_trc_sbc(jn) ) WRITE(numout,9001) jn, TRIM( sn_trcsbc(jn)%clvar_%clvar_), 'SBC', rn_trsfac(jn)
 ENDDO
 ENDIF
 WRITE(numout,'(2a)') ' SURFACE BC data repository : ', TRIM(cn_dir_sbc)
 WRITE(numout,*') ' Total tracers to be initialized with COASTAL BCs_'
 WRITE(numout,*) ' data:', nb_trccbc
 IF ( nb_trccbc > 0 ) THEN
 WRITE(numout,*) ' #trc NAME Boundary Mult.Fact.'
 DO jn = 1, ntrc
 IF ( ln_trc_cbc(jn) ) WRITE(numout, 9001) jn, TRIM( sn_trccbc(jn)_%clvar_), 'CBC', rn_trcfac(jn)
 ENDDO
 ENDIF
 WRITE(numout,'(2a)') ' COASTAL BC data repository : ', TRIM(cn_dir_cbc)
 WRITE(numout,*') ' Total tracers to be initialized with OPEN BCs_'
 WRITE(numout,*) ' data:', nb_trcobc
 #if defined key_bdy
 IF ( nb_trcobc > 0 ) THEN
 WRITE(numout,*') ' #trc NAME Boundary Mult.Fact. OBC_'
 WRITE(numout,*) ' Settings'
 DO jn = 1, ntrc
 IF ( ln_trc_obc(jn) ) WRITE(numout, 9001) jn, TRIM( sn_trcobc(jn)_%clvar_), 'OBC', rn_trofac(jn), (trcdta_bdy(jn,ib)%cn_obc,ib=1,nb_bdy)
 IF ( .NOT. ln_trc_obc(jn) ) WRITE(numout, 9002) jn, 'Set data to IC_'
 ENDDO
 ENDIF
 WRITE(numout,*') ' Boundary ',ib,' - NO damping of tracers'
 IF (nn_trcdmp_bdy(ib) .EQ. 0) WRITE(numout,9003) ' Boundary ',ib,' -
 WRITE(numout,*) ' damping ONLY for tracers with external data provided'
 IF (nn_trcdmp_bdy(ib) .EQ. 1) WRITE(numout,9003) ' Boundary ',ib,' -
 WRITE(numout,*) ' damping of ALL tracers'
 IF (nn_trcdmp_bdy(ib) .GT. 0) THEN
 WRITE(numout,9003) ' USE damping parameters from nambdy for
 WRITE(numout,9003) ' - Inflow damping time scale : ', rn_time_dmp(ib),' days'
 WRITE(numout,9003) ' - Outflow damping time scale : ',
 ENDIF
 ENDDO

(continues on next page)
ENDIF
#endif

WRITE(numout,'(2a)') ' OPEN BC data repository : ', TRIM(cn_dir_obc)

9001 FORMAT(2x,i5, 3x, a15, 3x, a5, 6x, e11.3, 4x, 10a13)
9002 FORMAT(2x,i5, 3x, a41, 3x, 10a13)
9003 FORMAT(a, i5, a)

#if defined key_bdy
  ! OPEN Lateral boundary conditions
  IF( nb_trcobc > 0 ) THEN
    ALLOCATE ( sf_trcobc(nb_trcobc*nb_bdy), rf_trofac(nb_trcobc*nb_bdy), nbmap_ptr(nb_trcobc*nb_bdy), STAT=ierr1 )
    IF( ierr1 > 0 ) THEN
      CALL ctl_stop( 'trc_bc_init: unable to allocate sf_trcobc structure' ) ; RETURN
    ENDIF
    igrd = 1 ! Everything is at T-points here
    DO ib = 1, nb_bdy
      READ ( numnat_ref, nambdy_bc, IOSTAT = ios, ERR = 905)
      IF( ios /= 0 ) CALL ctl_nam ( ios , 'nambdy_bc in reference namelist', lwp )
      READ ( numnat_cfg, nambdy_bc, IOSTAT = ios, ERR = 906 )
      IF( ios /= 0 ) CALL ctl_nam ( ios , 'nambdy_bc in configuration namelist', lwp )
      IF(lwm) WRITE ( numont, nambdy_bc )
      nblen = idx_bdy(ib)%nblen(igrd)
      DO jn = 1, ntrc
        IF ( ln_trc_obc(jn)) THEN
          ! Initialise from external data
          jl = n_trc_indobc(jn)
          sli(jl) = sn_trcobc(jn)
          rf_trofac(jl+(ib-1)*nb_trcobc) = rn_trofac(jn)
          IF( sn_trcobc(jn)%ln_tint ) ALLOCATE( sf_trcobc(jl+(ib-1)*nb_trcobc)%fdta(nblen,1,jpk,2) , STAT=ierr3 )
          IF( ierr2 + ierr3 > 0 ) THEN
            CALL ctl_stop( 'trc_bc_init : unable to allocate passive tracer OBC data arrays' ) ; RETURN
          ENDIF
          trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
          trcdta_bdy(jn,ib)%rn_fac = rf_trofac(jl+(ib-1)*nb_trcobc)
        ELSE
          ! Initialise obc arrays from initial conditions
          nblen = idx_bdy(ib)%nblen(igrd)
          DO ibd = 1, nblen
            DO ik = 1, jpkm1
              ii = idx_bdy(ib)%nbi(ibd,igrd)
              trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
            ENDIF
          ENDIF
          trcdta_bdy(jn,ib)%rn_fac = rf_trofac(jl+(ib-1)*nb_trcobc)
        ENDIF
      ENDIF
    ENDIF
  ELSE
    ! Initialise obc arrays from initial conditions
    ALLOCATE ( trcdta_bdy(jn,ib)%trc(nblen,jpk) )
    DO ibd = 1, nblen
      DO ik = 1, jpkm1
        ii = idx_bdy(ib)%nbi(ibd,igrd)
        trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
      ENDIF
    ENDIF
  ENDIF
ENDIF

DO ib = 1, nb_bdy
  IF ( ln_trc_obc(jn)) THEN
    ! Initialise from external data
    jl = n_trc_indobc(jn)
    sli(jl) = sn_trcobc(jn)
    rf_trofac(jl+(ib-1)*nb_trcobc) = rn_trofac(jn)
    IF( sn_trcobc(jn)%ln_tint ) ALLOCATE( sf_trcobc(jl+(ib-1)*nb_trcobc)%fdta(nblen,1,jpk,2) , STAT=ierr3 )
    IF( ierr2 + ierr3 > 0 ) THEN
      CALL ctl_stop( 'trc_bc_init : unable to allocate passive tracer OBC data arrays' ) ; RETURN
    ENDIF
    trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
    trcdta_bdy(jn,ib)%rn_fac = rf_trofac(jl+(ib-1)*nb_trcobc)
  ELSE
    ! Initialise obc arrays from initial conditions
    nblen = idx_bdy(ib)%nblen(igrd)
    DO jn = 1, ntrc
      IF ( ln_trc_obc(jn)) THEN
        ! Initialise from external data
        jl = n_trc_indobc(jn)
        sli(jl) = sn_trcobc(jn)
        rf_trofac(jl+(ib-1)*nb_trcobc) = rn_trofac(jn)
        IF( sn_trcobc(jn)%ln_tint ) ALLOCATE( sf_trcobc(jl+(ib-1)*nb_trcobc)%fdta(nblen,1,jpk,2) , STAT=ierr3 )
        IF( ierr2 + ierr3 > 0 ) THEN
          CALL ctl_stop( 'trc_bc_init : unable to allocate passive tracer OBC data arrays' ) ; RETURN
        ENDIF
        trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
      ELSE
        ! Initialise obc arrays from initial conditions
        nblen = idx_bdy(ib)%nblen(igrd)
        DO ibd = 1, nblen
          DO ik = 1, jpkm1
            ii = idx_bdy(ib)%nbi(ibd,igrd)
            trcdta_bdy(jn,ib)%trc => sf_trcobc(jl+(ib-1)*nb_trcobc)%fnow(:,1,:)
          ENDIF
        ENDIF
        trcdta_bdy(jn,ib)%rn_fac = rf_trofac(jl+(ib-1)*nb_trcobc)
      ENDIF
    ENDIF
  ENDIF
ENDIF

2.10. SMELT Documentation
ij = idx_bdy(ib)%nbj(ibd,igrd)
trcdta_bdy(jn,ib)%trc(ibd,ik) = trn(ii,ij,ik,jn) * tmask(ii,
    ij,ik)
END DO
END DO
trcdta_bdy(jn,ib)%rn_fac = 1._wp
ENDIF
ENDDO
CALL fld_fill( sf_trcobc((1+(ib-1)*nb_trcobc):(nb_trcobc+(ib-1)*nb_-
    trcobc)), slf_i, cn_dir_obc, &
    'trc_bc_init', 'Passive tracer OBC data', 'namtrc_bc' )
ENDDO
ENDIF
#ifdef
!IF( nb_trcobc > 0 ) THEN ! allocate only if the number of tracer to-
    initialise is greater than zero
! ALLOCATE( sf_trcobc(nb_trcobc), rf_trofac(nb_trcobc), STAT=ierr1 )
! IF( ierr1 > 0 ) THEN
! CALL ctl_stop( 'trc_bc_init: unable to allocate sf_trcobc structure' )
! RETURN
! ENDIF
! DO jn = 1, ntrc
! IF( ln_trc_obc(jn) ) THEN ! update passive tracers arrays with
! jl = n_trc_indobc(jn)
! slf_i(jl) = sn_trcobc(jn)
! rf_trofac(jl) = rn_trofac(jn)
! ALLOCATE( sf_trcobc(jl)%fnow(jpi,jpj,
    jpk) , STAT=ierr2 )
! IF( sn_trcobc(jn)%ln_tint ) ALLOCATE( sf_trcobc(jl)%fdta(jpi,jpj,jpk,
    2) , STAT=ierr3 )
! IF( ierr2 + ierr3 > 0 ) THEN
! CALL ctl_stop( 'trc_bc_init : unable to allocate passive tracer OBC-
    data arrays' ) ; RETURN
! ENDIF
! ENDIF
! ! ENDDO
! ! ! fill sf_trcdta with slf_i and control print
! CALL fld_fill( sf_trcobc, slf_i, cn_dir, 'trc_bc_init', 'Passive tracer OBC-
    data', 'namtrc_bc' )
! ! !ENDIF
! ! ! SURFACE Boundary conditions
! IF( nb_trcsbc > 0 ) THEN ! allocate only if the number of tracer to-
    initialise is greater than zero
ALLOCATE( sf_trcsbc(nb_trcsbc), rf_trsfac(nb_trcsbc), STAT=ierr1 )
IF( ierr1 > 0 ) THEN
CALL ctl_stop( 'trc_bc_init: unable to allocate sf_trcsbc structure' ) ; RETURN
ENDIF
DO jn = 1, ntrc
IF( ln_trc_sbc(jn) ) THEN ! update passive tracers arrays with input-
    data read from file
    jl = n_trc_indobc(jn)
    slf_i(jl) = sn_trcsbc(jn)
    rf_trsfac(jl) = rn_trsfac(jn)
    ALLOCATE( sf_trcsbc(jl)%fnow(jpi,jpj,
        jpk) , STAT=ierr2 )
    ! IF( sn_trcobc(jn)%ln_tint ) ALLOCATE( sf_trcobc(jl)%fdta(jpi,jpj,jpk,
    2) , STAT=ierr3 )
    ! IF( ierr2 + ierr3 > 0 ) THEN
    ! CALL ctl_stop( 'trc_bc_init : unable to allocate passive tracer OBC-
    data arrays' ) ; RETURN
    ! ENDIF
    ! ENDIF
    ! ! ENDDO
    ! ! fill sf_trcdta with slf_i and control print
    ! CALL fld_fill( sf_trcsbc, slf_i, cn_dir, 'trc_bc_init', 'Passive tracer OBC-
    data', 'namtrc_bc' )
    ! ! !ENDIF
    ! ! ! SURFACE Boundary conditions
    IF( nb_trcsbc > 0 ) THEN ! allocate only if the number of tracer to-
        initialise is greater than zero
        ALLOCATE( sf_trcsbc(nb_trcsbc), rf_trsfac(nb_trcsbc), STAT=ierr1 )
        IF( ierr1 > 0 ) THEN
            CALL ctl_stop( 'trc_bc_init: unable to allocate sf_trcsbc structure' ) ; RETURN
        ENDIF
        DO jn = 1, ntrc
            IF( ln_trc_sbc(jn) ) THEN ! update passive tracers arrays with input-
                data read from file
jl = n_trc_indscbc(jn)
slf_i(jl) = sn_trcsbc(jn)
rf_trsfac(jl) = rn_trsfac(jn)
ALLOCATE ( sf_trcsbc(jl)%fnow(jpi,jpj,1) , STAT=ierr2 )
IF ( sn_trcsbc(jn)%ln_tint ) ALLOCATE ( sf_trcsbc(jl)%fdta(jpi,jpj,1,2) , STAT=ierr3 )
IF ( ierr2 + ierr3 > 0 ) THEN
  CALL ctl_stop ('trc_bc_init : unable to allocate passive tracer SBC data arrays' ) ; RETURN
ENDIF
ENDIF
!
ENDDO
!
! fill sf_trcsbc with slf_i and control print
CALL fld_fill ( sf_trcsbc, slf_i, cn_dir_sbc, 'trc_bc_init', 'Passive tracer SBC data', 'namtrc_bc' )
!
ENDIF
!
! COASTAL Boundary conditions
!
IF ( nb_trccbc > 0 ) THEN ! allocate only if the number of tracer to initialise is greater than zero
  ALLOCATE ( sf_trccbc(nb_trccbc), rf_trcfac(nb_trccbc), STAT=ierr1 )
  IF ( ierr1 > 0 ) THEN
    CALL ctl_stop ('trc_bc_ini: unable to allocate sf_trccbc structure' ) ; RETURN
  ENDIF
ENDIF
!
DO jn = 1, ntrc
  IF ( ln_trc_cbc(jn) ) THEN ! update passive tracers arrays with input data read from file
    jl = n_trc_indcbbc(jn)
slf_i(jl) = sn_trccbc(jn)
rf_trcfac(jl) = rn_trcfac(jn)
ALLOCATE ( sf_trccbc(jl)%fnow(jpi,jpj,1) , STAT=ierr2 )
IF ( sn_trccbc(jn)%ln_tint ) ALLOCATE ( sf_trccbc(jl)%fdta(jpi,jpj,1,2) , STAT=ierr3 )
IF ( ierr2 + ierr3 > 0 ) THEN
  CALL ctl_stop ('trc_bc_ini : unable to allocate passive tracer CBC data arrays' ) ; RETURN
ENDIF
ENDIF
!
ENDDO
!
! fill sf_trccbc with slf_i and control print
CALL fld_fill ( sf_trccbc, slf_i, cn_dir_cbc, 'trc_bc_init', 'Passive tracer CBC data', 'namtrc_bc' )
!
ENDIF
!
DEALLOCATE ( slf_i ) ! deallocate local field structure
!
WRITE(numout,*) "In trc_bc_init at B: nstop=", nstop
IF ( nn_timing == 1 ) CALL timing_stop('trc_bc_init')
END SUBROUTINE trc_bc_init

SUBROUTINE trc_bc_read(kt, jit)
!!------------------------------------------------------------------------------------------------------
!! ** ROUTINE trc_bc_init **
!! ** Purpose : Read passive tracer Boundary Conditions data**
!! ** Method : Read BC inputs and update data structures using fldread**
!! *------------------------------------------------------------------------------------------------------

! NEMO
USE fldread

!! * Arguments
INTEGER, INTENT( in ) :: kt ! ocean time-step index
INTEGER, INTENT( in ), OPTIONAL :: jit ! subcycle time-step index (for
 timesplitting option)
INTEGER :: ib

!---------------------------------------------------------------------
!
IF( nn_timing == 1 ) CALL timing_start('trc_bc_read')

IF( kt == nit000 ) THEN
IF(lwp) WRITE(numout,*)
IF(lwp) WRITE(numout,*) 'trc_bc_read : Surface boundary conditions for
passive tracers.'
IF(lwp) WRITE(numout,*) '~~~~~~~ '
ENDIF
!
IF ( PRESENT(jit) ) THEN
!---------------------------------------------------------------------
!
# if defined key_bdy

! OPEN boundary conditions (use time_offset=+1 as they are applied at the
 end of the step)
IF( nb_trcobc > 0 ) THEN
   if (lwp) write(numout,'(a,i5,a,i10)') ' reading OBC data for ', nb_
 trcobc , ' variable(s) at step ', kt
   DO ib = 1, nb_bdy
      CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trcobc((1+(ib-1)*nb_trcobc):(nb_
 trcobc+(ib-1)*nb_trcobc)), &
 &             map=nbmap_ptr((1+(ib-1)*nb_trcobc):(nb_trcobc+(ib-1)*nb_
 trcobc)), kit=jit, kt_offset=+1)
   END DO
END IF
# endif

! SURFACE boundary conditions
IF ( nb_trcsbc > 0 ) THEN
   if (lwp) write(numout,'(a,i5,a,i10)') ' reading SBC data for ', nb_
 trcsbc , ' variable(s) at step ', kt
   CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trcsbc, kit=jit)
END IF

! COASTAL boundary conditions

(continues on next page)
IF( nb_trccbc > 0 ) THEN
  if (lwp) write(numout,'(a,i5,a,i10)') ' reading CBC data for ', nb_trccbc, variable(s) at step ', kt
  CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trccbc, kit=jit)
ENDIF

ELSE
#if defined key_bdy
  ! OPEN boundary conditions (use time_offset=+1 as they are applied at the end of the step)
  IF( nb_trcobc > 0 ) THEN
    if (lwp) write(numout,'(a,i5,a,i10)') ' reading OBC data for ', nb_trcobc, variable(s) at step ', kt
    DO ib = 1, nb_bdy
       CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trcobc((1+(ib-1)*nb_trcobc):(nb_trcobc+(ib-1)*nb_trcobc)), map=nbmap_ptr((1+(ib-1)*nb_trcobc):(nb_trcobc+(ib-1)*nb_trcobc)), kt_offset=+1)
    END DO
  ENDIF
#endif
!
SURFACE boundary conditions
IF( nb_trcsbc > 0 ) THEN
  if (lwp) write(numout,'(a,i5,a,i10)') ' reading SBC data for ', nb_trcsbc, variable(s) at step ', kt
  CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trcsbc)
ENDIF
!
! COASTAL boundary conditions
IF( nb_trccbc > 0 ) THEN
  if (lwp) write(numout,'(a,i5,a,i10)') ' reading CBC data for ', nb_trccbc, variable(s) at step ', kt
  CALL fld_read(kt=kt, kn_fsbc=1, sd=sf_trccbc)
ENDIF
!
ENDIF

END SUBROUTINE trc_bc_read
#else
!!----------------------------------------------------------------------
!! Dummy module NO 3D passive tracer data
!!----------------------------------------------------------------------
CONTAINS
SUBROUTINE trc_bc_read( kt ) ! Empty routine
  WRITE(*,*) 'trc_bc_read: You should not have seen this print! error?', kt
END SUBROUTINE trc_bc_read
#Endif

!!----------------------------------------------------------------------
END MODULE trcbc
**trcdby.F90**

```
MODULE trcdby
!!!======================================================================
!! *** MODULE bdytrc ***
!! Ocean tracers: Apply boundary conditions for tracers in TOP component
!!!======================================================================
!! History : 1.0 ! 2005-01 (J. Chanut, A. Sellar) Original code
!! 3.0 ! 2008-04 (NEMO team) add in the reference version
!! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY
!!merge
!! 3.5 ! 2012 (S. Mocavero, I. Epicoco) Optimization of BDY
!!communications
!! 3.6 ! 2015 (T. Lovato) Adapt BDY for tracers in TOP component
!!----------------------------------------------------------------------
#if defined key_bdy && key_top
!!!----------------------------------------------------------------------
!! 'key_bdy' Unstructured Open Boundary Conditions
!!----------------------------------------------------------------------
!! trc_bdy : Apply open boundary conditions to T and S
!! trc_bdy_frs : Apply Flow Relaxation Scheme
!!----------------------------------------------------------------------
USE timing ! Timing
USE oce_trc ! ocean dynamics and tracers variables
USE par_trc
USE trc ! ocean space and time domain variables
USE bdylib ! for orlanski library routines
USE lbclnk ! ocean lateral boundary conditions (or mpp link)
USE in_out_manager ! I/O manager
USE bdy_oce, only: idx_bdy, OBC_INDEX, BDYTMASK ! ocean open boundary
!!conditions
USE bdy_par, only: lk_bdy
IMPLICIT NONE
PRIVATE
PUBLIC trc_bdy ! routine called in trcnxt.F90
PUBLIC trc_bdy_dmp ! routine called in trcstp.F90
!!----------------------------------------------------------------------
!! NEMO/OPA 3.6 , NEMO Consortium (2015)
!! $Id$
!! Software governed by the CeCILL licence (NEMOGCM/NEMO_CeCILL.txt)
!!----------------------------------------------------------------------
CONTAINS

SUBROUTINE trc_bdy( kt )
!!!----------------------------------------------------------------------
!! *** SUBROUTINE trc_bdy ***
!! ** Purpose : - Apply open boundary conditions for tracers in TOP component
!! and scale the tracer data
!!----------------------------------------------------------------------
INTEGER, INTENT( in ) :: kt ! Main time step counter
!! INTEGER :: ib_bdy, jn ! Loop indeces
```
IF( nn_timing == 1 ) CALL timing_start('trc_bdy')
!
WRITE(numout,*) "In trc_bdy at A: nstop=", nstop
DO jn = 1, jptra
  DO ib_bdy=1, nb_bdy
    SELECT CASE( trcdta_bdy(jn,ib_bdy)%cn_obc )
    CASE('none')
      CYCLE
    CASE('frs')
      CALL bdy_trc_frs( jn, idx_bdy(ib_bdy), trcdta_bdy(jn,ib_bdy), kt )
    CASE('specified')
      CALL bdy_trc_spe( jn, idx_bdy(ib_bdy), trcdta_bdy(jn,ib_bdy), kt )
    CASE('neumann')
      CALL bdy_trc_nmn( jn, idx_bdy(ib_bdy), trcdta_bdy(jn,ib_bdy), kt )
    CASE('orlanski')
      CALL bdy_trc_orlanski( jn, idx_bdy(ib_bdy), trcdta_bdy(jn,ib_bdy), ll_
                        -->npo=.false. )
    CASE('orlanski_npo')
      CALL bdy_trc_orlanski( jn, idx_bdy(ib_bdy), trcdta_bdy(jn,ib_bdy), ll_
                        -->npo=.true. )
    CASE DEFAULT
      WRITE(numout,*) "jn=", jn, "ib_bdy=", ib_bdy, "trcdta_bdy(jn,ib_bdy)
                        -->%cn_obc="", trcdta_bdy(jn,ib_bdy)%cn_obc
      CALL ctl_stop( 'trc_bdy : unrecognised option for open boundaries for
                        -->T and S' )
    END SELECT
    !WRITE(numout,*) "In trc_bdy at B: nstop=", nstop
    ! Boundary points should be updated
    CALL lbc_bdy_lnk( tra(:,,:,:), 'T', 1., ib_bdy )
    !WRITE(numout,*) "In trc_bdy at C: nstop=", nstop
  ENDDO
ENDDO
IF( nn_timing == 1 ) CALL timing_stop('trc_bdy')
END SUBROUTINE trc_bdy

SUBROUTINE bdy_trc_frs( jn, idx, dta, kt )
!!----------------------------------------------------------------------
!! *** SUBROUTINE bdy_trc_frs ***
!! ** Purpose : Apply the Flow Relaxation Scheme for tracers at open boundaries.
!!----------------------------------------------------------------------
INTEGER, INTENT(in) :: kt
INTEGER, INTENT(in) :: jn ! Tracer index
TYPE(OBC_INDEX), INTENT(in) :: idx ! OBC indices
TYPE(OBC_DATA), INTENT(in) :: dta ! OBC external data
!!
REAL(wp) :: zwgt ! boundary weight
INTEGER :: ib, ik, igrd ! dummy loop indices
INTEGER :: ii, ij ! 2D addresses
!!----------------------------------------------------------------------
!
IF( nn_timing == 1 ) CALL timing_start('bdy_trc_frs')
!
igrd = 1 ! Everything is at T-points here
DO ib = 1, idx%nblen(igrd)
  DO ik = 1, jqkm1
    ii = idx%nbi(ib,igrd)
    ij = idx%nbj(ib,igrd)
    zwgt = idx%nbw(ib,igrd)
    tra(ii,ij,ik,jn) = ( tra(ii,ij,ik,jn) + zwgt * ( ( dta%trc(ib,ik) * dta
      \rightarrow%rn_fac) &
      & - tra(ii,ij,ik,jn) ) ) * tmask(ii,ij,ik)
  END DO
END DO
!
IF( kt .eq. nit000 ) CLOSE( unit = 102 )
!
IF( nn_timing == 1 ) CALL timing_stop('bdy_trc_frs')
!
END SUBROUTINE bdy_trc_frs

SUBROUTINE bdy_trc_spe( jn, idx, dta, kt )
!!----------------------------------------------------------------------
!! *** SUBROUTINE bdy_trc_frs ***
!!
!! ** Purpose : Apply a specified value for tracers at open boundaries.
!!
!!----------------------------------------------------------------------
INTEGER, INTENT(in) :: kt
INTEGER, INTENT(in) :: jn ! Tracer index
TYPE(OBC_INDEX), INTENT(in) :: idx ! OBC indices
TYPE(OBC_DATA), INTENT(in) :: dta ! OBC external data
!!
REAL(wp) :: zwgt ! boundary weight
INTEGER :: ib, ik, igrd ! dummy loop indices
INTEGER :: ii, ij ! 2D addresses
!!----------------------------------------------------------------------
!
IF( nn_timing == 1 ) CALL timing_start('bdy_trc_spe')
!
igrd = 1 ! Everything is at T-points here
DO ib = 1, idx%nblenrim(igrd)
  ii = idx%nbi(ib,igrd)
  ij = idx%nbj(ib,igrd)
  DO ik = 1, jqkm1
    tra(ii,ij,ik,jn) = dta%trc(ib,ik) * dta%rn_fac * tmask(ii,ij,ik)
  END DO
END DO
!
IF( kt .eq. nit000 ) CLOSE( unit = 102 )
!
IF( nn_timing == 1 ) CALL timing_stop('bdy_trc_spe')
!
END SUBROUTINE bdy_trc_spe

SUBROUTINE bdy_trc_nmn( jn, idx, dta, kt )
!!----------------------------------------------------------------------
!!----------------------------------------------------------------------

(continues on next page)
!! *** SUBROUTINE bdy_trc_nmn ***
!! ** Purpose: Duplicate the value for tracers at open boundaries. **
!!------------------------------------------------------------------------------------------------------------------
INTEGER, INTENT(in) :: kt
INTEGER, INTENT(in) :: jn ! Tracer index
TYPE(OBC_INDEX), INTENT(in) :: idx ! OBC indices
TYPE(OBC_DATA), INTENT(in) :: dta ! OBC external data
!! REAL(wp) :: zwgt ! boundary weight
INTEGER :: ib, ik, igrd ! dummy loop indices
INTEGER :: ii, ij, zcoef, zcoef1, zcoef2, ip, jp ! 2D addresses
!!----------------------------------------------------------------------
!
IF( nn_timing == 1 ) CALL timing_start('bdy_trc_nmn')
!
!WRITE(numout,*) "Called bdy_trc_nmn"
igrd = 1 ! Everything is at T-points here
DO ib = 1, idx%nblenrim(igrd)
   ii = idx%nbi(ib,igrd)
   ij = idx%nbj(ib,igrd)
   DO ik = 1, jpkm1
      ! search the sense of the gradient
      zcoef1 = bdytmask(ii-1,ij ) + bdytmask(ii+1,ij )
      zcoef2 =! bdytmask(ii ,ij-1) + bdytmask(ii ,ij+1)
      IF ( zcoef1+zcoef2 == 0) THEN
         ! corner
         zcoef = tmask(ii-1,ij,ik) + tmask(ii+1,ij,ik) + tmask(ii,ij-1,ik) + tmask(ii,ij+1,ik)
      tra(ii,ij,ik,jn) = tra(ii-1,ij ,ik,jn) * tmask(ii-1,ij ,ik) + &
         & tra(ii+1,ij ,ik,jn) * tmask(ii+1,ij ,ik) + &
         & tra(ii ,ij-1,ik,jn) * tmask(ii ,ij-1,ik) + &
         & tra(ii ,ij+1,ik,jn) * tmask(ii ,ij+1,ik)
      tra(ii,ij,ik,jn) = ( tra(ii,ij,ik,jn) / MAX( 1, zcoef ) ) * tmask(ii,ij,
      !-------------------------------------------------------------
      ELSE
         ip = bdytmask(ii+1,ij ) - bdytmask(ii-1,ij )
         jp = bdytmask(ii ,ij+1) - bdytmask(ii ,ij-1)
      tra(ii,ij,ik,jn) = tra(ii+ip,ij+jp,ik,jn) * tmask(ii+ip,ij+jp,ik)
      ENDIF
   END DO
END DO
!
IF( kt .eq. nit000 ) CLOSE( unit = 102 )
!
IF( nn_timing == 1 ) CALL timing_stop('bdy_trc_nmn')
!
END SUBROUTINE bdy_trc_nmn

SUBROUTINE bdy_trc_orlanski( jn, idx, dta, ll_npo )
!!----------------------------------------------------------------------
!! *** SUBROUTINE bdy_trc_orlanski ***
!! ! Apply Orlanski radiation to tracers of TOP component.
!! ! Wrapper routine for bdy_orlanski_3d
(continues on next page)
!! References: Marchesiello, McWilliams and Shchepetkin, Ocean Modelling vol. 3 (2001)
!!----------------------------------------------------------------------
INTEGER, INTENT(in) :: jn ! Tracer index
TYPE(OBC_INDEX), INTENT(in) :: idx ! OBC indices
TYPE(OBC_DATA), INTENT(in) :: dta ! OBC external data
LOGICAL, INTENT(in) :: ll_npo ! switch for NPO version

INTEGER :: igrd ! grid index
!!----------------------------------------------------------------------
IF( nn_timing == 1 ) CALL timing_start('bdy_trc_orlanski')
!
igrd = 1 ! Orlanski bc on tracers;
!
CALL bdy_orlanski_3d( idx, igrd, trb(:,:,:,jn), tra(:,:,:,jn), (dta%trc * dta ...
-->%rn_fac), ll_npo )
!
IF( nn_timing == 1 ) CALL timing_stop('bdy_trc_orlanski')
!
END SUBROUTINE bdy_trc_orlanski

SUBROUTINE trc_bdy_dmp( kt )
!!----------------------------------------------------------------------
!! *** SUBROUTINE trc_bdy_dmp ***
!! ** Purpose : Apply damping for tracers at open boundaries.
!! It currently applies the damping to all tracers!!!
!!----------------------------------------------------------------------
INTEGER, INTENT(in) :: kt
!!
INTEGER :: jn ! Tracer index
REAL(wp) :: zwgt ! boundary weight
REAL(wp) :: zta, zsa, ztime
INTEGER :: ib, ik, igrd ! dummy loop indices
INTEGER :: ii, ij ! 2D addresses
INTEGER :: ib_bdy ! Loop index
!!----------------------------------------------------------------------
IF( nn_timing == 1 ) CALL timing_start('trc_bdy_dmp')
!
DO jn = 1, jptra
  DO ib_bdy=1, nb_bdy
    IF ( trcdta_bdy(jn, ib_bdy)%dmp ) THEN
      igrd = 1 ! Everything is at T-points here
      DO ib = 1, idx_bdy(ib_bdy)%nblen(igrd)
        ii = idx_bdy(ib_bdy)%nbi(ib,igrd)
        ij = idx_bdy(ib_bdy)%nbj(ib,igrd)
        zwgt = idx_bdy(ib_bdy)%nbd(ib,igrd)
        DO ik = 1, jpkml
          zta = zwgt * ( trcdta_bdy(jn, ib_bdy)%trc(ib,ik) - trb(ii,ij,ik, ...
-->jn) ) * tmask(ii,ij,ik)
          tra(ii,ij,ik,jn) = tra(ii,ij,ik,jn) + zta
        END DO
      END DO
    END IF
  END DO
END DO
!
END SUBROUTINE trc_bdy_dmp

END DO
END DO
ENDIF
ENDDO

IF( nn_timing == 1 ) CALL timing_stop('trc_bdy_dmp')

END SUBROUTINE trc_bdy_dmp

#else
!!----------------------------------------------------------------------
!! Dummy module NO Unstruct Open Boundary Conditions
!!----------------------------------------------------------------------
CONTAINS
SUBROUTINE trc_bdy(kt) ! Empty routine
WRITE(*,*) 'trc_bdy: You should not have seen this print! error?', kt
END SUBROUTINE trc_bdy

SUBROUTINE trc_bdy_dmp(kt) ! Empty routine
WRITE(*,*) 'trc_bdy_dmp: You should not have seen this print! error?', kt
END SUBROUTINE trc_bdy_dmp

#Endif

!!======================================================================
END MODULE trcbdy

Namelist contents

We have two open boundaries, one northern and one western.

Boundary-related sections from our namelist_top_cfg are reproduced below.

namtrc (note additional columns)

<table>
<thead>
<tr>
<th>name</th>
<th>title of the field</th>
<th>units</th>
<th>initial</th>
<th>data</th>
<th>initial data</th>
<th>obc</th>
<th>sbc</th>
<th>cbc</th>
<th>save</th>
</tr>
</thead>
<tbody>
<tr>
<td>sn_tracer(1) = 'NO3 ' , 'Nitrates Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_tracer(2) = 'NH4 ' , 'Ammonium Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_tracer(3) = 'Si ' , 'Silicate Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_tracer(4) = 'DIAT ' , 'Diatoms Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_tracer(5) = 'PHY ' , 'Nanophytoplankton Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_tracer(6) = 'MYRI ' , 'Mesozooplankton Concentration ', 'mol-C/L' , .true. , .true. , .false. , .false. , .true.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(continues on next page)
sn_tracer(7) = 'MICZ ', 'Microzooplankton Concentration ', 'mol-C/L', 
   .true. , .true. , .false. , .false. , .true.
sn_tracer(8) = 'DON ', 'Dissolved organic Concentration ', 'mol-C/L', 
   .true. , .true. , .false. , .false. , .true.
sn_tracer(9) = 'PON ', 'Small organic carbon Concentration ', 'mol-C/L', 
   .true. , .true. , .false. , .false. , .true.
sn_tracer(10) = 'bSi ', 'biogenic Silicate Concentration ', 'mol-C/L', 
   .true. , .true. , .false. , .false. , .true.
sn_tracer(11) = 'TRA ', 'River Tracer ', 'mol-C/L', 
   .false. , .true. , .false. , .false. , .true.
ln_trcdta = .true.
ln_trcdmp = .false.

nambdy_bc (first boundary):

!------------------------------------------------------------------------
! nambdy_bc ! data for BDY boundary conditions: 1st boundary
!------------------------------------------------------------------------
&nambdy_bc
!
! ! file name ! frequency (hours) ! variable ! time interp. !
→clim ! 'yearly'/ ! weights ! rotation ! land/sea mask !

→(T/F ) ! 'monthly' ! filename ! pairing ! filename !

→ sn_trcobc(1) = 'bioOBC_full.nc', 168, 'NO3', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(2) = 'bioOBC_full.nc', 168, 'NH4', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(3) = 'bioOBC_full.nc', 168, 'Si', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(4) = 'bioOBC_full.nc', 168, 'DIA', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(5) = 'bioOBC_full.nc', 168, 'CRY', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(6) = 'bioOBC_full.nc', 168, 'MYRI', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(7) = 'bioOBC_full.nc', 168, 'MICZ', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(8) = 'bioOBC_full.nc', 168, 'DON', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(9) = 'bioOBC_full.nc', 168, 'PON', .true., .
   .true. , 'yearly', '', '', '',

→ sn_trcobc(10) = 'bioOBC_full.nc', 168, 'bSi', .true., .
   .true. , 'yearly', '', '', '',

   .true. , 'yearly', '', '', '',

→ cn_dir_obc = './open_boundaries/west/' ! root directory for the
→ location of OPEN data files

rn_trofac(1) = 1.0 ! - - - -
rn_trofac(2) = 1.0 ! - - - -
rn_trofac(3) = 1.0 ! multiplicative factor
rn_trofac(4) = 1.0 ! - - - -
rn_trofac(5) = 1.0 ! - - - -
rn_trofac(6) = 1.0 ! - - - -
rn_trofac(7) = 1.0 ! - - - -
rn_trofac(8) = 1.0 ! - - - -
rn_trofac(9) = 1.0 ! - - - -
rn_trofac(10) = 1.0 ! - - - -
rn_trofac(11) = 1.0 ! - - - -

nambdy_bc (second boundary):

```
!-------------------------------------------------------------
! nambdy_bc        data for BDY boundary conditions: 2nd boundary
!-------------------------------------------------------------
&nambdy_bc

! file name    frequency (hours) variable time interp. ...
  ! (if <0 months) ! name ! (logical) ! ...
  -clim ! 'yearly'/ ! weights ! rotation ! land/sea mask !

sn_trcobc(1) = 'bioOBC_full_north.nc', 4380, 'NO3', .true., .true., 'yearly', '', '', ''
sn_trcobc(2) = 'bioOBC_full_north.nc', 4380, 'NH4', .true., .true., 'yearly', '', '', ''
sn_trcobc(3) = 'bioOBC_full_north.nc', 4380, 'Si', .true., .true., 'yearly', '', '', ''
sn_trcobc(4) = 'bioOBC_full_north.nc', 4380, 'DIA', .true., .true., 'yearly', '', '', ''
sn_trcobc(5) = 'bioOBC_full_north.nc', 4380, 'CRY', .true., .true., 'yearly', '', '', ''
sn_trcobc(6) = 'bioOBC_full_north.nc', 4380, 'MYRI', .true., .true., 'yearly', '', '', ''
sn_trcobc(7) = 'bioOBC_full_north.nc', 4380, 'MICZ', .true., .true., 'yearly', '', '', ''
sn_trcobc(8) = 'bioOBC_full_north.nc', 4380, 'DON', .true., .true., 'yearly', '', '', ''
sn_trcobc(9) = 'bioOBC_full_north.nc', 4380, 'PON', .true., .true., 'yearly', '', '', ''
sn_trcobc(10) = 'bioOBC_full_north.nc', 4380, 'bSi', .true., .true., 'yearly', '', '', ''
sn_trcobc(11) = 'bioOBC_full_north.nc', 4380, 'O2', .true., .true., 'yearly', '', '', ''
   cn_dir_obc = './open_boundaries/north/' ! root directory for the ...

Corresponding boundary options in namelist_cfg:

```
$nambdy ! unstructured open boundaries ("key_bdy")
```

(continues on next page)
nb_bdy = 2 ! number of open boundary sets According to merge namelist, only 1
ln coords file = .false., .false. ! =T : read bdy coordinates from file
  cn_dyn2d = 'flather', 'flather' ! = 0, bdy data are equal to the initial state
  nn_dyn2d_dta = 3, 2 ! = 1, bdy data are read in 'bdydata .nc files
  nn_from files ! = 2, use tidal harmonic forcing data
  nn_dyn3d_dta = 3, 2 ! = 3, use external data AND tidal harmonic forcing
  cn_dyn3d = 'orlanski', 'orlanski' ! = 0, bdy data are equal to the initial state
  nn_dyn3d_dta = 0, 0 ! = 1, bdy data are read in 'bdydata .nc files
  cn_tra = 'frs', 'frs' ! = 0, bdy data are equal to the initial state
  nn_tra_dta = 1, 1 ! = 1, bdy data are read in 'bdydata .nc files
  ln_tra_dmp = .false., .false. ! open boundary conditions for tracers
  ln_dyn3d_dmp = .false., .false. ! open boundary condition for baroclinic velocities
  nn_rimwidth = 10, 10 ! Damping time scale in days (nudging on orlanski inflow)
  nn_time_dmp = 1., 1. ! Outflow damping time scale (nudging on orlanski outflow)

!-----------------------------------------------------------------------
&nambdy_index ! open boundaries - definition ("key_bdy")

ctypebdy = 'W'
  nbdyind = 2 ! i-index for segment
  nbdybeg = 385 ! j-index for segment beginning
  nbdyend = 471 ! j-index for segment end
&end

&nambdy_dta ! open boundaries - external data ("key_bdy")

bn_ssh = 'ssh/ssh', 1, 'sossheig', .true., .false., 'daily', '', ''
bn_u2d = 'ssh/ssh', 1, 'vobtcrtx', .true., .false., 'daily', '', ''
bn_v2d = 'ssh/ssh', 1, 'vobtcrty', .true., .false., 'daily', '', ''
bn_u3d = '', 24, 'vozocrtx', .true., .false., 'daily', '', ''
bn_v3d = '', 24, 'vomecrty', .true., .false., 'daily', '', ''
bn_tem = 'SalishSea_west_TEOS10', 168, 'votemper', .true., .false., 'yearly', '', ''
bn_sal = 'SalishSea_west_TEOS10', 168, 'vosaline', .true., .false., 'yearly', '', ''

(continues on next page)
Boundary condition file formats

Our western and northern boundary forcing files for passive tracers contain variables of shape (52, 40, 1, 870) and (2, 40, 1, 300), respectively. The first dimension is time; the second is depth; the third is always 1; and the fourth is the length of the boundary multiplied by the rimwidth, both of which are defined in namelist_cfg. The boundary data is repeated nn_rimwidth times in the along-boundary direction.

Sample files:
2.11 SMELT Sensitivity Analysis

2.11.1 SMELT Sensitivity Analysis: Introduction

SMELT (Salish Model Ecosystem- Lower Trophic) is a one dimensional biological model (SOG) coupled to a physical model of the Salish Sea (NEMO). This model tracks the interactions and movement of several types of plankton and nutrients.

In the model parameters are used to describe things like plankton growth functions, sinking rates, mortality rates, light requirements and nutrient requirements. There are over one hundred biological parameters with values that have been chosen based on related research.

A sensitivity analysis was performed on this model to better understand the impact of individual parameter choices on model results. These pages describes the methodology used and current findings.

The analysis was performed by modifying one parameter at a time and comparing the results. Parameter notebooks and metric functions were used to qualitatively and quantitatively examine the impact of these changes.

2.11.2 SMELT Sensitivity Analysis: Batch Simulation Runs

The first task was to create a dataset of simulation results for the different parameter changes. It was decided to test each parameter at 10%, 50%, 90%, 110%, 200% and 1000% of its default value. There are 6 changes per parameter and over 100 parameters in total, so this is about 600 simulation runs. The SMELT model has a large physical domain (398x898x40) and so it is fairly slow to run. Due to the number of runs required it was not feasible to use the full model. Instead we created a smaller domain (5x5x40) with simplified physics that was much faster to run (3 months simulation time in about 30 minutes on one processor). Ideally trends observed in the smaller model will also be applicable to the full model.

With a minimum of several hundred simulation runs it made sense to automate this process. A python script was created to modify the parameters one at a time and run the model for each change. The code can be found here.

This script was run on the local computer ‘Salish’. It would start up simulation runs such that the total CPU usage did not exceed a preset threshold. This was intended to be less obtrusive to other users of the same machine. Shown below is a point form overview of how the script works.

- create a list of parameter change sets. Each item specifies which parameters should be different from the default namelist
- for each parameter change set
  - wait until there are enough processors available
  - create a biological namelist file with the default values and the change applied
  - create a run identifier- typically the name of the changed parameter and its new value
  - call salishsea_cmd.api.run_in_subprocess(), specifying the name of the result directory as the run identifier and using the modified namelist

The various datasets created using this method are currently stored in /data/jpetrie/MEOPAR/SalishSea/results/ on the local filesystem. All individual result directories are stored with the long name name of the parameter and the value used for that parameter, eg. “nampismezo_zz_rate_mesozaoo_alpha_0.5”. Detailed biological parameter information for the run can be found in the file namelist_pisces_cfg stored in the run directory. The combined dataset contains...
results for every parameter that doesn’t contain the string “zz_frac_waste” in its name and does not have a default value of 0.

The same modifications were made with February, April, and June initial conditions. The April and June result directories are stored in all_params_AprIC_june_22 and all_params_JunIC_june_23 respectively. The February results are stored differently: they are sorted by section and the file names do not explicitly say that they are using February initial conditions. The following list details the directories the February runs are split into: nampiszoo_june_14/, nampisopt_june_14/, nampisms_june_14/, nampissink_june_17/, nampisprod_june_16/, nampismort_june_17/, nampisrem_june_17, nampismezo_june_20/. They have a format of [section name]_[date computed].

If the model is changed and the results need to be regenerated follow the steps below:

- Compile the 5x5 model if you haven’t already
- Open the python script
- Create a modified version of the reference yaml file to match your directory organisation
- Change the reference_yaml variable in the script to point to your new yaml file
- Change the results_dir variable to wherever you want the output
- If only certain parameters are needed, modify the first for loop to only include these in the patch list
- ssh onto Salish (optional)
- Run the script (using “python batch_5x5.py”)

### 2.11.3 SMELT Sensitivity Analysis: Parameter Notebooks

Parameter notebooks are used to take an in depth look at how that parameter affects model results. They compare how tracer values evolve over time differently based on the value of the parameter.

Three main types of plots are used in these notebooks:

- Heatmap of tracer concentration against depth and time for a particular parameter value
- Heatmap of the difference between tracer concentrations for two different runs against depth and time
- Facetplot showing tracer concentration vs. time at certain depths, with a coloured line for each of the six parameter values

This is an example notebook for the parameter nampisrem_zz_remin_d_pon. This parameter controls the rate at which particulate organic nitrogen becomes dissolved organic nitrogen.

A notebook was automatically created for every parameter using this python script. Some of these notebooks have been uploaded to Bitbucket, and all of them can be found on the local filesystem at /ocean/jpetrie/MEOPAR/analysis-james/notebooks/.

### 2.11.4 SMELT Sensitivity Analysis: Metrics

Metrics are used to quantitatively compare all of the parameters. This provides a useful overview of how the parameters affect certain parts of the simulation.

An example of a metric would be peak primary producer biomass. This is defined as the maximum value of a three day running average of total primary producer biomass in the model domain. For every model result this metric can be calculated.

Then we can see how the value of this metric changes for different parameter modifications. This metric can be thought of as a function with inputs being the parameter values and a single valued output. Under this paradigm we can define partial derivatives.
This notebook is used to compare the normalised partial derivatives of every parameter for 3 different starting months. The partial derivatives are normalised so that the impact of parameters is related to their percent change, not their magnitude change. This allows for better comparison of parameters that are several orders of magnitude apart.

The notebook linked above uses a list of result directories and a list of metric functions, it applies each metric function to every result within those directories. Then a dataset is created with columns for metric name, modified parameter name, percent change in parameter, start month, and metric value. This dataset is used to create two types of plots.

- The first type of plot is a facet plot with an individual axes for every unique metric and season combination. It shows the metric value on the y axis and the percent change in parameter on the x axis. The 15 parameters with the most impact on these metrics are drawn as differently coloured lines on these plots. This type of plot is useful to examine the trend of the metric value as these parameters change.

- The next type of plot is a horizontal bar plot. As before there is an individual axes for every metric/season combination. These bar plots show the scaled partial derivative of that metric with respect to every parameter (with the specified start month). The bars are coloured by parameter section to more easily group together similar parameters. The bar plots give a quick overview of what parameters change each metric the most. They are a useful place to start when trying to tune the model.

All of the metric functions used in the linked notebook are found in this python file. They all take `grid_t` as the only input, where grid_t is an xarray object containing the tracer data found in *ptrc_t.nc files.

Using a new metric is fairly easy: define a function that takes grid_t as input and add this function to metric_func_list. Next run the entire notebook (this may take a while because the notebook has to load 1800 xarray datasets and calculate each metric for every dataset).

### 2.11.5 SMELT Sensitivity Analysis: Applications

The results of the sensitivity analysis can be used to tune the full model to better match observational data.

An example use case is a discrepancy found between measured and predicted nitrate concentrations at 20m depth at several locations in the Salish Sea. This was found by comparing Nowcast Green results from 2016 with nitrate measurements at various locations from 2003, 2004, and 2005. The notebook used to compare these datasets can be found here. These are notebooks comparing temperature and salinity, chlorophyl, and diatom concentrations at the same locations. The temperature and salinity notebook indicates that the model is capturing mixing effects properly at these locations because the difference between surface and 20m concentrations is similar.

To determine what parameters are most useful to tune a ‘Nitrate at 20m’ metric is created. This metric is defined as the integral of Nitrate between 15 and 25 metres depth for all times of the model run. The value of this metric is then calculated for every one of the existing 5x5 model results in this notebook. The notebook shows that there are only a few parameters that have a substantial impact on the nitrate concentration at a depth of 20m, and only when the simulation is started in the winter. The reason the parameters don’t impact nitrate in later months is that by that time it has already been depleted by the spring bloom.

### 2.12 Storm Surges

This page outlines some of our review on past storm surges, as well as details on how to set up a storm surge simulation and tools for analysis.

#### 2.12.1 Storm surge overview

We need to test our model to see how well it can represent storm surge events in the Salish Sea. We have wind forcing available from 2002-2010, so storms need to be in this period.
High water levels can be due to a combination of high seasonal tide, strong winds, low atmospheric pressure and sea level height anomalies due to ENSO events.

What classifies as a storm?

How long is a storm?

- More than 6 hours => the algorithm below finds 65 ‘storms’ from 2002-2012
- More than 12 hours => the algorithm below finds 25 ‘storms’ from 2002-2012
- Decision: more than 12 hours is a more appropriate number to test

What is the water level elevation during a storm?

- Decision: sea surface anomaly greater than 40cm

Does the storm affect the whole domain or does it not matter?

- For now, look at the DFO water level sites that are in the domain and have water level data for the period of interest (2002 - 2010). These sites are:
  - Point Atkinson
    * lat/lon: 49.34,-123.25
    * grid co-ordinates: i=468, j=328
  - Victoria
    * lat/lon: 48.42,-123.37
    * grid co-ordinates: i=298, j=195
  - Patricia Bay
    * lat/lon: 48.65,-123.45
    * grid co-ordinates: i=351, j=213
  - Campbell River
    * lat/lon: 50.04,-125.25
    * grid co-ordinates: i=747, j=124

Finding storms in the record

Compare predicted tide with measured water level using t_tide (Pawlowicz et al, 2002) and the following MATLAB scripts:

```
get_tidal_anomaly.m
find_storm_events.m
```

Usage:

```
[pred,wlev,anomaly,tim] = get_tidal_anomaly(csvfilename)
[startind,endind,lengthstorm] = find_storm_events(anomaly,tim,anomthres,stormlength)
```

where:

- csvfilename - name of csv file of hourly measured water level at Point Atkinson
- pred - predicted tides from t_tide (m CD)
• wlev - measured water level at Point Atkinson (m CD)
• anomaly - difference between prediction and water level (m)
• tim - time vector for pred, wlev and anomaly (MATLAB date format)
• startind - indice in tim for start of each storm (-)
• endind - indice in tim for end of each storm (-)
• lengthstorm - length of each storm (hours)
• anomthres - water level elevation defined as a storm (m)
• stormlength - minimum length defined as a storm (hrs)

Outputs a text file called 'storms.txt' that contains a list of the start dates of storms and the length of each storm in hours. e.g.:

```matlab
[pred,wlev,anomaly,tim] = get_tidal_anomaly('wlev_ts_for_storms.csv');
[startind,endind,lengthstorm] = find_storm_events(anomaly,tim,0.4,12);
```
gives for Point Atkinson:

<table>
<thead>
<tr>
<th>Start date</th>
<th>Duration (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-Jan-2003 04:00:00</td>
<td>13</td>
</tr>
<tr>
<td>13-Mar-2003 04:00:00</td>
<td>27</td>
</tr>
<tr>
<td>14-Mar-2003 23:00:00</td>
<td>16</td>
</tr>
<tr>
<td>05-Dec-2003 20:00:00</td>
<td>13</td>
</tr>
<tr>
<td>22-Dec-2005 04:00:00</td>
<td>14</td>
</tr>
<tr>
<td>31-Dec-2005 00:00:00</td>
<td>17</td>
</tr>
<tr>
<td>31-Jan-2006 18:00:00</td>
<td>13</td>
</tr>
<tr>
<td>04-Feb-2006 01:00:00</td>
<td>18</td>
</tr>
<tr>
<td>15-Nov-2006 09:00:00</td>
<td>15</td>
</tr>
<tr>
<td>14-Dec-2006 18:00:00</td>
<td>15</td>
</tr>
<tr>
<td>12-Nov-2007 04:00:00</td>
<td>14</td>
</tr>
<tr>
<td>03-Dec-2007 02:00:00</td>
<td>35</td>
</tr>
<tr>
<td>04-Jan-2008 12:00:00</td>
<td>27</td>
</tr>
<tr>
<td>11-Jan-2010 20:00:00</td>
<td>13</td>
</tr>
<tr>
<td>18-Jan-2010 02:00:00</td>
<td>16</td>
</tr>
<tr>
<td>19-Jan-2010 02:00:00</td>
<td>45</td>
</tr>
<tr>
<td>21-Jan-2010 02:00:00</td>
<td>43</td>
</tr>
<tr>
<td>29-Mar-2010 01:00:00</td>
<td>16</td>
</tr>
<tr>
<td>23-Oct-2010 23:00:00</td>
<td>19</td>
</tr>
<tr>
<td>22-Nov-2011 23:00:00</td>
<td>21</td>
</tr>
<tr>
<td>20-Jan-2012 20:00:00</td>
<td>14</td>
</tr>
<tr>
<td>12-Mar-2012 07:00:00</td>
<td>13</td>
</tr>
<tr>
<td>29-Mar-2012 02:00:00</td>
<td>26</td>
</tr>
<tr>
<td>30-Nov-2012 21:00:00</td>
<td>15</td>
</tr>
<tr>
<td>01-Dec-2012 16:00:00</td>
<td>22</td>
</tr>
</tbody>
</table>

**Water level at Point Atkinson**

Example of hourly water level at Point Atkinson during the storm on 4 February 2004:

Outputting water level data at each hour (or perhaps every 30 minutes?) from the NEMO model should be therefore be appropriate for assessing storm surge performance.
Literature search for big storms

- Jan 02, 2003, Victoria
- Dec 24, 2003, Vancouver
- Nov 2006, Vancouver
- Dec 15, 2006, Vancouver
- Dec 25, 2008, Vancouver
- Nov 24, 2011, Vancouver (not within range)
- Jan 05, 2012, Vancouver (not within range)
- Dec 17, 2012, Vancouver (not within range)
- Sep 30, 2013, Vancouver (not within range)

Prediction for 2013/2014 from Storm Surge Almanac:

*Flooding risk is greatest during the seasonal perigean spring tides, which correspond to times of extreme high tidal levels during the winter months. This year the highest tides for the Lower Mainland are expected in the first weeks of December, January and February; for Victoria the highest tides are expected in the first and last weeks of December, and the final week of January.*

Also, from Abeysirigunawardena et al (2011), extremes generally occur from October to March.

Existing storm surge models

The Government of British Columbia launched the BC Storm Surge Forecast System in 2011. This model is driven by the Pacific Ocean Model (which is driven by 7 day weather forecasts from NOAA) and a 6 day forecast from Environment Canada. Forecast bulletins are available for Point Atkinson, Victoria and Campbell River. There is no hindcasting available in the model.
2.12.2 Storm surge simulation set up

Once we have identified an appropriate date for a storm surge simulation hindcast (see Storm surge overview for examples), we need to generate forcing files. We are forcing the storm surge system in two ways: sea surface height anomaly at the Juan de Fuca boundary and atmospheric winds and pressure at the sea surface. The usual tidal forcing is also applied. The initial conditions should also be considered carefully.

Initial conditions and start date

Initial conditions on temperature and salinity are generated by model spin-up. The initial conditions on the velocities and sea surface height are zero. We use restart files from model spin-up runs to initialize the temperature and salinity. The procedure is described here: Initializing T+S with a restart file.

It is a good idea to choose a restart file close to the start date of your simulation.

Since the velocities and sea surface height are initially zero, the model needs time for spin up. Storm surge simulations should begin at least one day prior to the storm event of interest. Two days would be safer. Analysis of the spin up period can be found here.

Sea surface height forcing

We force the sea surface height anomaly at the Juan de Fuca boundary based on tidal observations from Tofino. The tidal component of the observed water level height at Tofino has been removed. The water level height observations are taken from the DFO and the tidal predictions are generated with t_tide. Details are found in this notebook: SSH_Tofino.ipynb.

This notebook can be edited to generate forcing files for other time frames.

To use these forcing files in NEMO, we must edit namelist.lateral. The title of the forcing files should have the format ssh_y2006m01.nc. Each file contains one month of the hourly sea surface height anomaly. First, tell NEMO that you are using external data on the barotropic fields:

```
&nambdy
  ! unstructured open boundaries
  !
  ! key_bdy"

 nb_bdy = 1 ! number of open boundary sets
 ln_coords_file = .false. ! =T : read bdy coordinates from file
 cn_coords_file = '' ! bdy coordinates files
 ln_mask_file = .false. ! =T : read mask from file
 cn_mask_file = '' ! name of mask file (if ln_mask_file=.TRUE.)
 nn_dyn2d = 2 ! boundary conditions for barotropic fields
 nn_dyn2d_dta = 3 ! = 0, bdy data are equal to the initial state
 ! = 1, bdy data are read in 'bdydata .nc' files
 ! = 2, use tidal harmonic forcing data from files
 ! = 3, use external data AND tidal harmonic forcing
```

References

Ensure that nn Dyn2d = 2 and nn Dyn2d_dta = 3.

Next, insert the forcing files into the nambdy_dta section. It should look like:

```plaintext
&nambdy_dta  ! open boundaries - external data (*key_bdy*)
!---------------------------------------------------------------
! file name ! freq (hr) ! variable ! time ! clim ! period ! weights ! rotation !
! ! ! ! ! ! ! ! ! (0 <= mo) ! name ! interp ! (T/F)
bn_ssh = 'ssh/ssh', 1, 'sossheig', .true., .false.,
bn_u2d = 'ssh/ssh', 1, 'vobtcrtx', .true., .false.,
bn_v2d = 'ssh/ssh', 1, 'vobtcrty', .true., .false.,
bn_u3d = '', 24, 'vozocrtx', .true., .false.,
bn_v3d = '', 24, 'vomecrty', .true., .false.,
bn_tem = 'SalishSea2_Masson_DC', 168, 'votemper', .true., .true.,
bn_sal = 'SalishSea2_Masson_DC', 168, 'vosaline', .true., .true.,
cn_dir = 'open_boundaries/west/'
ln_full_vel = .false.
&end
```

**Atmospheric Forcing**

Atmospheric forcing is an important component of storm surges. Currently, we are using model output for winds, pressure and a variety of other fields from CGRF (see Atmospheric Forcing). Ensure that the atmospheric forcing files for the simulation dates of interest have been generated.

**Namelists**

Example namelists and run set files for storm surge simulations are located in the SS-run-sets repo. The iodef.xml tells NEMO to output hourly data for several storm surge locations of interest.

**2.12.3 Tools**

We have developed a variety of tools used for analysis and creation of storm surge simulations.

**The Surge**

We are often interested in the behaviour of the surge component of the water level, that is, the anomaly after the tides have been removed. Further, we could like to compare the modelled surge to the observed surge. To determine the modelled surge, we perform two simulations: one with all of our forcing conditions, including the tides, atmospheric conditions, rivers, and sea surface height at the open boundaries and another with only the tidal forcing and rivers. We define the modelled surge as the difference between the sea surface height of these two simulations.

To calculate the observed surge, we need water level observations and tidal predictions.
Observations

Water level observations for Canadian tide gauges are easily obtained from the Fisheries and Oceans Canada (DFO) website. We also use observations from NOAA tide gauges.

Tidal predictions

Tidal predictions are generated using a MATLAB package called t_tide. The general procedure is as follows:

1. Perform a harmonic analysis on a year-long time series using t_tide.
2. Use the tidal constituents produced by the harmonic analysis to generate a tidal prediction. Typically, 67 constituents are analyzed with t_tide for a year-long time series.

However, there are some subtleties that need to be considered before we generate the tidal predictions for use in residual calculations.

Note: In order to use these tools, you need to download t_tide and add the t_tide directory to your MATLAB path.

Filtering

In a year with many storm surges, the harmonic analysis may result in over predicted constituents due to the large surges. As such, before the harmonic analysis is performed, we filter the time series by applying a Doodson tide filter (Parker, 2007) and we remove periods with large non-tidal energy. The harmonic analysis is then applied to the filtered time series.

Long Period Constituents

In a harmonic analysis, long period constituents (Sa, Ssa, etc) in this region are often contaminated by non-tidal energy due to seasonal meteorological events. We would like to represent this seasonal variability in our residual forcing. As such, we do not include long period constituents in our tidal predictions.

Constituents with low signal to noise ratio

Constituents with a low signal to noise ratio (less than 2) are not used in tidal predictions.

Model Correction

Because our model is forced with only 8 tidal constituents, our water level predictions are missing some tidal energy. We have attempted to correct for the missing energy in the following way.

1. Generate a tidal prediction with constituents from the harmonic analysis (excluding long period constituents, constituents with low signal to noise ratio, and shallow water constituents).
2. Generate a tidal prediction with only the eight constituents used to force the model.
3. Calculate the difference between these two predictions and add to the model sea surface height as a correction.

The shallow water constituents are excluded from the tidal prediction because they will be generated by the model and we should avoid counting them twice.
MATLAB Scripts

Several MATLAB scripts have been designed to calculate the tidal predictions as described above. These scripts are found in the analysis-storm-surges/tide_analysis_scripts/ repository and are described below.

*generate_tidal_predictions.m*

This script does most of the work. The end result is a series of tidal predictions.

```matlab
generate_tidal_predictions(filename, location, starts, ends, type, ...
\rightarrow exclude_long, cut_off, ssh_units, time_zone)
```

- Uses water level observations or harmonic constituents stored in `filename` to calculate tidal predictions over a time period defined by date strings `starts` and `ends`. Water level observations can either be from the DFO website or the NOAA website, as specified by the `type` argument. Or a file with harmonic constituents from CHS can be used, in which case type is set to `CHS`. Also, a file with NOAA constituents can be used, in which case type is `NOAA_const`.

- If a harmonic analysis is necessary, the calculated harmonics are saved in `location_harmonics_date1_date2_filter.csv` where `location` is one of the arguments of `generate_tidal_predictions.m`. `date1` and `date2` are string representations of the start and end date of the observation time series.

- The tidal predictions are stored in a file called `location_tidal_prediction_starts_ends.csv` where `starts` and `ends` are arguments of `generate_tidal_predictions.m`. This file contains three types of tidal predictions:
  - `pred_all` - predictions with all constituents except shallow water and ones with low signal to noise
  - `pred_8` - predictions with only eight constituents
  - `pred_noshallow` - like `pred_all` but with no shallow water constituents.

- `exclude_long` is a flag that specifies whether or not long period constituents should be excluded from the tidal predictions. `exclude_long = 1` means exclude long period constituents like Sa, Ssa, etc from the tidal prediction. `exclude_long = 0` means include long period constituents in tidal predictions. Note that if `exclude_long = 0` then a lot of the variability between `pred_all` and `pred_8` because `pred_all` uses long period constituents but `pred_8` does not.

- `cut_off` is the amplitude at which non-tidal energy is removed from the harmonic analysis. Time periods for which the filtered time series is greater than `cut_off` are removed from the water level time series and then the harmonic analysis is performed. A reasonable value is 0.3. If filtering is not desired then set `cut_off` very high (>1).

- `ssh_units` is the units of the water level information in the harmonics or time series file (eg. ‘m’ or ‘feet’)

- `time_zone` is the time zone of the time information stored in the harmonics or time series files. For example, if `time_zone` is ‘PST’ then the phase in the harmonics files is relative to PST time or the time in the time series file is relative to PST.

**Note:** If a harmonic analysis is necessary, then the water level time series should be less than one year but long enough to separate the important constituents. Typically, one year is a reasonable length.

*calculate_harmonics.m and calculate_harmonics_NOAA.m* These files perform the harmonics analysis for DFO and NOAA data respectively.

*filter_tides.m and filter_tides_NOAA.m* These files do the filtering work.
read_CHS_harmonics.m  Read the CHS tidal harmonics from a file.
read_NOAA_harmonics.m  Read the NOAA tidal harmonics from a file.

**Note:** The NOAA observations csv files should have the station’s latitude in the second row, second column of the file.

**Note:** See /ocean/nsoontie/MEOPAR/tides/NOAA_tidal_constituents/NeahBay_harmonics.csv for an example of how the NOAA harmonics files should be formatted.

**Warning:** Some of the constituents published on the NOAA website are not recognized by t_tide. We have attempted to match these constituents to once valid in ttide but this approach can lead to errors in the nodal corrections. It is best to use a time-series when producing tidal predictions for a NOAA station.

### Storm surge forcing files

Several notebooks have been developed for generating the anomaly forcing files used in simulation hindcasts.

- SSH_Tofino.ipynb
- SSH_PortHardy.ipynb

### Analysis

Some analysis functions are stored in a module salishsea_tools/stormtools. Examples include functions that calculate the observed residual, modelled residual, error statistics, and so on.

**Note:** A different module was used for analysis in the AO storm surge paper. It is in a private repository storm-surge/stormtools_revisions.py. The functions are almost identical as stormtools.py but with a few minor changes.

### References


### 2.12.4 Nowcast Stations

The Salish Sea Nowcast system provides water level predictions and storm surge warnings for several locations in the Salish Sea. Initially, we used Point Atkinson, Victoria and Campbell River but we found a need to provide predictions for other key locations in the domain. This section explains how to add a new station to the nowcast production system.
Adding a new station

In order to add a new station, we first need some key information beyond the trivial latitude/longitude of the desired station. This information includes:

1. Historical extreme water level and mean sea level.
2. A year long time series of water levels or a set of tidal constituents for generating tidal predictions.
3. Latitude and longitude of desired station.

Points 1 and 2 require that a tide gauge was in operation at this location at some point in history. The reason we need this information is to calculate appropriate thresholds for high water based on historical values. We also calculate a correction to the model water level using tidal predictions.

To search for this information, you can visit the DFO and NOAA websites.

Steps for adding a new station to nowcast

Adding new tidal stations requires access to the Salish Sea tools repository. Most of the work will be on editing figures.py in the SalishSeaNowcast package. You will also need access to scripts in the analysis-storm-surges repository and the nowcast iodef.xml file in SS-run-sets.

1. Add extreme water level, mean sea level, latitude and longitude for this location in the PLACES dictionary object of places.py.
2. Add location name to TIDAL_SITES list in figures.py.
3. Use the MATLAB scripts in analysis-storm_surges/tide_analysis_scripts/generate_tidal_predictions.m to generate Jan 1, 2015 to Jan 1, 2020 tidal predictions for your location.
   - The input file can either be a year-long water level time series from NOAA/DFO or a constituent file from DFO or NOAA.
   - If using a water level time series, be sure that the time zone is PST and the latitude is added to the second row, second column of the csv file.
   - Use exclude_long=1,cut_off=0.3.
   - Copy the tidal predictions output file to tools/SalishSeaNowcast/tidal_predictions/. Add this file to the repository, commit and push.
   - See Tools for a description of how to use the MATLAB scripts.
   - You will need ttide to run these scripts.

Warning: Some of the constituents published on the NOAA website are not recognized by ttide. So, it is best to use a time-series to produce tidal predictions at a NOAA site.

4. Look up the grid point indices of your station and add a 15 minute ssh output to SS-run-sets/SalishSea/nowcast/iodef.xml
5. Test ssh plotting calls with your new station in OutTemplate.ipynb
   - Testing won’t be possible until we have a 15 minute output file for this station. You can test the process by linking one of the existing 15 minute files into a new file with the appropriate name. The plots will be gibberish but you can test the procedure.
6. Add appropriate calls to plotting functions and saving/displaying figures in tools/SalishSeaNowcast/nowcast/make_plots.py and tools/SalishSeaNowcast/nowcast/make_site_page.py.
   - Testing the workers won’t be possible until we have a 15 minute output file for this station. You can test the process by linking one of the existing 15 minute files into a new file with the appropriate name. The plots will be gibberish but you can test the procedure.

2.13 Turbulence and Viscosity

Configuring NEMO to model the Salish Sea requires careful consideration of the appropriate turbulence models and parameterizations. Vertical mixing is an important physical mechanism within the Strait of Georgia. A discussion on the lateral and vertical dynamics associated with mixing is provided here, as well as justifications for the choice of several model parameters and parameterizations.

2.13.1 Literature Review

This section documents modelling choices from other studies in this region or similar regions. It includes an outline of choices in grid type, resolution, bottom friction, and turbulence closure.
<table>
<thead>
<tr>
<th>Study</th>
<th>Region</th>
<th>Model</th>
<th>Horizontal Grid</th>
<th>Vertical Grid</th>
<th>Horizontal Mixing</th>
<th>Vertical Mixing</th>
<th>Bottom Friction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sutherland et al (2011)</td>
<td>Salish Sea and Puget Sound</td>
<td>ROMS</td>
<td>min 280 m res</td>
<td>20 terrain-following</td>
<td>constant diff 2 m2/s</td>
<td>k-eps with Canuto A functions</td>
<td>quadratic 3e-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>smoothing dh/h&lt;0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Masson and Cummins (2004)</td>
<td>Salish Sea</td>
<td>POM</td>
<td>2km res</td>
<td>31 sigma levels</td>
<td>Smagorinsky</td>
<td>Smagorinsky sensitivity to HORCON</td>
<td>Mellor Yamada 2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HORCON</td>
<td>HORCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.03 or 0.06</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>coefficient 0.2</td>
<td>background diff/visc 1e-6 m2/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stacey et al (1995)</td>
<td>Knight Inlet</td>
<td>2D</td>
<td></td>
<td></td>
<td>constant diff/visc</td>
<td>Mellor Yamada 2.5</td>
<td>quadratic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>chosen for stability</td>
<td>adjustment for internal waves</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.13. Turbulence and Viscosity

- **Foreman et al (2006)**
  - Broughton Archipelago
  - ELCIRC
  - **z-coordinate**
  - GLS
  - quadratic 0.003
### 2.13.2 References


### 2.13.3 Lateral Dynamics

The lateral dynamics are controlled in the `namelist.dynamics`. We have decided to experiment with the lateral eddy viscosity `rn_ahm_0_lap`, the biharmonic operator `ln_dynldf_bilap`, and the lateral boundary conditions for momentum set in the `namelist.lateral`.

#### Lateral Eddy Viscosity

The lateral eddy viscosity controls how dissipative the simulation is. Since we are forcing tides at the Juan de Fuca boundary we would like the viscosity to be as small as possible in order to prevent tidal phase lags within the Strait of Georgia. We have had some success at reducing the lateral viscosity as outlined in the table below.

<table>
<thead>
<tr>
<th>Simulation</th>
<th><code>rn_ahm_0_lap</code></th>
<th><code>ln_apr_dyn</code></th>
<th><code>rn_avevd</code></th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu = 100$</td>
<td>100</td>
<td>true or false</td>
<td>10</td>
<td>stable</td>
</tr>
<tr>
<td>$\nu = 60$</td>
<td>60</td>
<td>false</td>
<td>10</td>
<td>stable</td>
</tr>
<tr>
<td>$\nu = 60$</td>
<td>60</td>
<td>true</td>
<td>10</td>
<td>unstable near islands</td>
</tr>
<tr>
<td>$\nu = 60$</td>
<td>60</td>
<td>true</td>
<td>100</td>
<td>stable</td>
</tr>
<tr>
<td>$\nu = 50$</td>
<td>50</td>
<td>false</td>
<td>10</td>
<td>unstable</td>
</tr>
<tr>
<td>$\nu = 40$</td>
<td>40</td>
<td>true</td>
<td>60</td>
<td>stable</td>
</tr>
<tr>
<td>$\nu = 40$</td>
<td>40</td>
<td>true</td>
<td>50</td>
<td>unstable near islands</td>
</tr>
</tbody>
</table>

These simulations were run on `salish` with model time seven days. The boolean `ln_apr_dyn` in `namelist.surface` controls forcing at the free surface due to atmospheric pressure. We are having difficulty with stability when this is set to true and the viscosity is lower than 100. Unfortunately, this type of forcing is likely important when modelling storm surges.

The parameter `rn_avevd` in `namelist.dynamics` controls the vertical eddy viscosity only in areas where/when the stratification is statically unstable. When the model is presented with an unstable stratification it locally increases the amount of vertical diffusion which effectively mixes the unstable region (see Vertical Mixing). Through our analysis, we have observed that the stability of the model at lower viscosities depends on how it treats vertical mixing. Thus, this parameter is likely important for achieving stability at low lateral viscosity. This must be investigated further.
We are still having trouble with stability during our spin up runs. See Spin-up Runs.

**Biharmonic Operator**

The biharmonic operator dissipates energy selectively at smaller scales. It is a fourth order diffusive operator in the momentum equations and is chosen by modifying \texttt{ln\_dyl_{df\_bilap}} in \texttt{namelist.dynamics}. It can be used in conjunction with the second order laplacian operator. The AMM configuration employs the bilaplacian with \texttt{rn\_ahm\_0\_blp=-1e10} and \( \nu = 60 \) for the laplacian operator. Note that AMM also uses free slip lateral boundary conditions and s-coordinates.

The biharmonic operator can be used in conjunction with the second order laplacian operator. Under the current resolution, typical values for the operator coefficient should be around \texttt{rn\_ahm\_0\_blp=-2000}. Decreasing the magnitude of this parameter has some stabilizing effect with little change in the maximum currents. However, the simulations with this operator in use still display overturning and poor behaviour in the vertical salinity profiles. A summary of simulations is given below.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>\texttt{rn_ahm_0_lap}</th>
<th>\texttt{rn_ahm_0_blp}</th>
<th>\texttt{rn_avevd}</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>apr60_nu50</td>
<td>50</td>
<td>none</td>
<td>60</td>
<td>unstable near islands</td>
</tr>
<tr>
<td>apr60_nu50_bi2000</td>
<td>50</td>
<td>-2000</td>
<td>60</td>
<td>unstable near islands</td>
</tr>
<tr>
<td>apr60_nu50_bi4000</td>
<td>50</td>
<td>-4000</td>
<td>60</td>
<td>unstable near islands</td>
</tr>
<tr>
<td>apr60_nu50_bi1000</td>
<td>50</td>
<td>-1000</td>
<td>60</td>
<td>stable, poorly behaved salinity</td>
</tr>
</tbody>
</table>

**Lateral Boundary Conditions**

Currently we are using partial slip boundary conditions with \texttt{rn\_shlat =0.5} in \texttt{namelist.lateral}. No slip conditions are applied when \texttt{rn\_shlat =2} and free slip when \texttt{rn\_shlat =0}.

At \( \nu = 50 \), we have seen some stabilizing features as we take the lateral boundary towards no slip. It seems that the no slip conditions change the location of the maximum velocities in the island regions, which can have an affect on the mixing. Our concern with no slip stems from resolving the boundary layer. We fear that using no slip BCs will leave the boundary layer unresolved, especially at lower viscosity.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>\texttt{rn_ahm_0_lap}</th>
<th>\texttt{rn_shlat}</th>
<th>\texttt{rn_avevd}</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>partial25/apr100_nu50</td>
<td>50</td>
<td>0.25</td>
<td>100</td>
<td>unstable at Stuart Island</td>
</tr>
<tr>
<td>apr60_nu50</td>
<td>50</td>
<td>0.5</td>
<td>60</td>
<td>unstable near islands</td>
</tr>
<tr>
<td>partial1/apr60_nu50</td>
<td>50</td>
<td>1</td>
<td>60</td>
<td>stable</td>
</tr>
<tr>
<td>noslip/apr60_nu50</td>
<td>50</td>
<td>2</td>
<td>60</td>
<td>stable</td>
</tr>
</tbody>
</table>

**2.13.4 Vertical Mixing**

The vertical mixing and diffusion are controlled in the \texttt{namelist.dynamics} and in \texttt{cpp} keys for vertical turbulence selection. We would like to determine the most appropriate combination of turbulence parameterization and lower boundary condition for modelling the Salish Sea. NEMO has several choices for turbulence closure which will be discussed here.

**Vertical Turbulence**

NEMO parameterizes vertical turbulence using eddy coefficients. There are several different methods for calculating the eddy coefficients:

1. Constant coefficients
2. Richardson number based coefficients
3. Turbulent kinetic energy closure
4. Generalized length scale

The most relevant options for our purposes are the 3 and 4. Currently we are using the generalized length scale with a $k - \epsilon$ closure.

**Turbulent kinetic energy**

The turbulent kinetic energy closure schemes involve a prognostic equation for the turbulent kinetic energy. The eddy coefficients are proportional to the turbulent kinetic energy and a turbulent length scale. The details are explained in the NEMO documentation.

This method includes options for dealing with strongly stratified regions and unstable stratifications which are attractive features for this project due to the strong mixing in the island regions. At this point, we have not attempted any simulations with this method of turbulence closure.

To do:
- Examine vertical mixing with this turbulence closure scheme.

**Generalized Length Scale**

The generalized length scale (GLS) turbulence option also involves a prognostic equation for the turbulent kinetic energy. Additionally, it uses a prognostic equation to determine the turbulent length scale. This method allows for easy comparison between several well-known turbulence closure schemes such as $k - \epsilon$ and Mellor-Yamada.

The behaviour of each of the closure methods in GLS has been examined by Warren et al (2005). Their results indicate that the $k - \epsilon$ option is appropriate for our scenario. This option performed well in a simulation of a wind-driven mixed layer and also in a simulation of estuarine circulation. This is a good first place to start in determining an appropriate mixing parameterization for our setup.

**Convection**

The NEMO model is hydrostatic and thus cannot directly handle unstable density stratifications that arise due to convection or overturning. There are several ways to parameterize this process which include enhanced vertical diffusion and a non-penetrative convective algorithm. Tests with the non-penetrative convective algorithm have not been successful.

**Enhanced Vertical Diffusion**

The enhanced vertical diffusion option increases the amount of vertical mixing locally whenever an unstable density stratification is encountered. It accomplishes this by changing the vertical eddy coefficient to a namelist set value $\text{rn}_\text{avevd}$ in namelist.dynamics.

Note that the turbulent kinetic energy closure schemes inherently produce higher eddy coefficients whenever the squared buoyancy frequency is negative. As such, we have observed a warning in ocean.output when using enhanced vertical diffusion in conjunction with a turbulent kinetic energy closure. However, several simulations with no enhanced vertical diffusion have been unsuccessful so we need a reliable way of dealing with unstable stratifications and will continue to use the enhanced vertical diffusion.

As noted in the discussion on lateral viscosity (Lateral Dynamics), the model often experiences instability near the islands where vertical mixing is expected. Several experiments with a higher enhanced vertical diffusion parameter
\texttt{rn\_avevd} have remained stable, even at low lateral viscosity. There will be a balance between an appropriate amount of vertical diffusion and a low enough lateral viscosity.

The simulations performed to date are outlined in \textit{Lateral Dynamics}.

**Bottom Friction**

Bottom friction is parameterized through a bottom boundary layer with either linear or nonlinear flux terms. We are using the nonlinear setting. The NEMO documentation recommends using an implicit calculation of the bottom boundary condition when using the split-explicit time stepping.

There is some flexibility in setting the coefficients on the flux term, \texttt{rn\_bfri2} for the nonlinear setting. This is controlled in \texttt{namelist.bottom}. Also, the NEMO documentation suggests using a low or zero value for \texttt{rn\_bfeb2} when tides are treated explicitly.

Reducing the parameter \texttt{rn\_bfri2} from $5 \times 10^{-3}$ to $4 \times 10^{-3}$ caused instability in the Puget Sound region. Setting \texttt{rn\_bfeb2} = 0 made no difference.

**References**


**2.14 Particle Tracking**

This page outlines the tools we are using for offline particle tracking with NEMO model output.

**2.14.1 Overview**

**What is particle tracking?**

Description and applications

**Particle tracking tools**

List of tools like Ariane, tracmass, etc.

**References**

**2.14.2 Ariane**

Please see the Ariane section of the UBC EOAS MOAD Group Documentation.

**2.14.3 Tracers with Ariane**

In addition to a particle’s trajectory (longitude and latitude) and depth, Ariane can also help us analyze tracers along the particle’s trajectory.

- Temperature
Salish Sea MEOPAR Documentation

- Salinity
- Density

We will be making changes in `namelist`.

**Namelist: Add Sections**

Ariane requires both salinity and temperature data as input. It also requires density data or an indication that density should instead be calculated using salinity and temperature.

Our model produces files with filenames that follow this format: `SalishSea_t_yyyymmdd_yyyymmdd_grid_T.nc`. These files contain salinity and temperature. Therefore, Ariane will be calculating density.

Add the following sections to `namelist`. Remember to modify the directory and filename of your input files.

**Temperature**

```
&TEMPERAT
  c_dir_te = '/ocean/nsoontie/MEOPAR/SalishSea/results/storm-surges/final/dec2006/all_forcing/30min/',
  c_prefix_te = 'SalishSea_30m_20061214_20061215_grid_T.nc',
  ind0_te = -1,
  indn_te = -1,
  maxsize_te = -1,
  c_suffix_te = 'NONE',
  nc_var_te = 'votemper',
  nc_att_mask_te = 'NONE',
/
```

**Salinity**

```
&SALINITY
  c_dir_sa = '/ocean/nsoontie/MEOPAR/SalishSea/results/storm-surges/final/dec2006/all_forcing/30min/',
  c_prefix_sa = 'SalishSea_30m_20061214_20061215_grid_T.nc',
  ind0_sa = -1,
  indn_sa = -1,
  maxsize_sa = -1,
  c_suffix_sa = 'NONE',
  nc_var_sa = 'vosaline',
  nc_att_mask_sa = 'NONE',
/
```

`votemper` and `vosaline` are the variable names for temperature and salinity in our input file.

**Namelist: Add Parameters**

Add the following parameters to the `ARIANE` section of the namelist:

- `key_computesigma`: False if density is read into Ariane; True if Ariane calculates density.
- `zsigma`: Reference depth for calculation of density.

And change:
- **key_alltracers**: True if tracers are required.

**Ariane**

```fortran
ARIAINE
  key_alltracers = .TRUE.,
  key_sequential = .FALSE.,
  key_ascii_outputs = .TRUE.,
  mode = 'qualitative',
  forback = 'forward',
  bin = 'nobin',
  init_final = 'init',
  nmax = 5,
  tunit = 3600.,
  ntfic = 1,
  tcyc = 0.,
  key_computesigma = .TRUE.,
  zsigma = 1.,
/
```

**Results**

The variables names for the tracers are:

- **traj_temp**
- **traj_salt**
- **traj_dens**

**Plots**

The results produced for the example above:

![Plots](image)

**Notebooks**

- Ariane_Tracers.ipynb
2.14.4 Sequential Mode in Ariane

Until now, we have entered only one input file into Ariane. Use Ariane’s sequential mode to enter multiple files.

Input Files

The NetCDF files used as input must have the following format: `prefix_number_suffix`

If the file names do not follow this format, create symbolic links that do. Create this link by using the command `ln -s [target file directory] [symbolic link name]`

For example, you may consider:

- `prefix = SalishSea_`
- `number = 01, 02, etc`
- `suffix = _grid_T.nc, _grid_U.nc, _grid_V.nc`

Note: `number` must contain a constant digit number and its value must increase by one in chronological order. For example, file `SalishSea_01_grid_T.nc` contains tracers for November 1st and `SalishSea_02_grid_T.nc` contains tracers for November 2nd.

Namelist: Modify Sections

First, let’s take a closer look at the parameters in the namelist sections. The parameter names are `c_dir_X`, `c_prefix_X`, `ind0_X`, `indn_X`, `maxsize_X`, and `c_suffix_X` where `X` is `zo`, `me`, `te`, `sa` for the ZONALCRT, MERIDCRT, TEMPERAT, and SALINITY sections, respectively.

Input File Directory

`c_dir_X` is the directory with the symbolic links for the input files, the namelist, and the initial positions text file.

New Input Filename

Previously, we have been entering the full filename, `SalishSea_t_yyyymmdd_yyyymmdd_grid_T.nc`, into `c_prefix_X`.

Now that we have formatted the filenames as `prefix_number_suffix`, `c_prefix_me` takes on the value of the `prefix` and `c_suffix_me` takes the value of `suffix`.

`ind0_X` is the `number` for the earliest input file and `indn_X` is the latest.

`maxsize_X` is the number of digits in `number`.

For example, the ZONALCRT section would look like the following for input files `SalishSea_01_grid_U.nc` and `SalishSea_02_grid_U.nc`:

```
&ZONALCRT
  c_dir_zo = '/ocean/imachuca/MEOPAR/Ariane/results/drifter_compare/sequential/',
  c_prefix_zo = 'SalishSea_',
  ind0_zo = 01,
  indn_zo = 02,
  maxsize_zo = 2,
  c_suffix_zo = '_grid_U.nc',
```

(continues on next page)
Sequential Parameter

Under the ARIANE section in namelist, change key_sequential to TRUE.

Namelist: Add Section

Add a SEQUENTIAL section in namelist. This section has one parameter, maxcycles. We recommend the value of this parameter to be 1 since this tells Ariane to stop generating trajectory points once it has run out of input data.

Sequential

```plaintext
$SEQUENTIAL
maxcycles =1,
/
```

Results

The results produced for the example above:

Notebooks

• Ariane_Sequential.ipynb

2.14.5 Frequency Sensitivity Studies

The model produces datasets containing information about the velocity field for a region. Ariane uses this information to produce particle trajectories. We wanted to know at what frequency would the model output need to be to produce the most reliable particle trajectories.

For the frequency sensitivity studies, we used model outputs with 30 minute, 1 hour, and 4 hour frequencies. This data was used in Ariane to generate particle trajectories with points at 30 minute intervals. We did this for particles starting their trajectories at the Fraser River and at various points along the thalweg.

Color scheme:

• 30 minute: green
• 1 hour: blue
• 4 hour: pink
On the Surface

At the Fraser River, we found that the particle trajectory generated using data at a 4 hour frequency does not capture subtleties in particle motion as do the trajectories derived from data at 30 minute and 1 hour frequencies.

The trajectory that used 1 hour frequency data very closely resembles the trajectory that used 30 minute data.

**Conclusion:** We can use 1 hour or 30 minute NEMO output data when particle trajectories start at the Fraser River.

At Depth

Figures 1, 2, 3, and 4 show particle trajectories with initial positions along the thalweg, but the initial positions in figures 3 and 4 are located in regions with stronger mixing. Figures A, B, C, and D show the change in depth of the particles as they progressed in their trajectories.

**Conclusion:** In regions of moderate mixing (Figures A and B), 30 minute data would be preferable; we can use 1 hour data with some caution. In regions of heavy mixing (Figures C and D), we should exercise caution in analyzing trajectories and depths since results vary greatly depending on frequencies.

**Notebooks**

- Ariane_TimeRes.ipynb
2.14.6 Quantitative Mode in Ariane

Ariane can be used to make estimates of transport through cross-sections by releasing a large number of particles and calculating how many particles pass through each section. Next, we will go through how to set up a quantitative experiment in Ariane.

Namelists

The namelist for quantitative mode is very similar to qualitative mode. Here is an example of a quantitative namelist.

```
&ARIANE
  key_alltracers =.FALSE.,
  key_sequential =.FALSE.,
  key_ascii_outputs =.TRUE.,
  mode = 'quantitative',
  forback = 'forward',
  bin = 'nobin',
  init_final = 'init',
  nmax = 30000,
  tunit = 3600.,
  ntfic = 1,
  key_computesigma = .FALSE.,
  zsigma = 100.,
/

&OPAPARAM
```

(continues on next page)
imt = 398,
jmt = 898,
kmt = 40,
lmt = 24,
key_periodic = .FALSE.,
key_jfold = .FALSE.,
key_computew = .FALSE.,
key_partialsteps = .TRUE.,
/

&QUANTITATIVE
key_eco = .TRUE.,
key_reducmem = .TRUE.,
key_unitm3 = .TRUE.,
key_nointerpstats = .FALSE.,
max_transport = 1.e4,
lmin = 1,
lmax = 6,
/

&ZONALCRT
  c_dir_zo = '/results/SalishSea/nowcast/01jan16/',
c_prefix_zo = 'SalishSea_lh_20160101_20160101_grid_U.nc',
  ind0_zo = -1,
  indn_zo = -1,
  maxsize_zo = -1,
c_suffix_zo = 'NONE',
  nc_var_zo = 'vozocrtx',
  nc_var_eivu = 'NONE',
  nc_att_mask_zo = 'NONE',
/

&MERIDCRT
  c_dir_me = '/results/SalishSea/nowcast/01jan16/',
c_prefix_me = 'SalishSea_lh_20160101_20160101_grid_V.nc',
  ind0_me = -1,
  indn_me = -1,
  maxsize_me = -1,
c_suffix_me = 'NONE',
  nc_var_me = 'vomecrty',
  nc_var_eivv = 'NONE',
  nc_att_mask_me = 'NONE',
/

&VERTICRT
  c_dir_ve = '/results/SalishSea/nowcast/01jan16/',
c_prefix_ve = 'SalishSea_lh_20160101_20160101_grid_W.nc',
  ind0_ve = -1,
  indn_ve = -1,
  maxsize_ve = -1,
c_suffix_ve = 'NONE',
  nc_var_ve = 'vovecrtz',
  nc_var_eivw = 'NONE',
  nc_att_mask_ve = 'NONE',
/

(continues on next page)
&TEMPERAT  
  c_dir_te = '/results/SalishSea/nowcast/01jan16/',
  c_prefix_te = 'SalishSea_lh_20160101_20160101_grid_T.nc',
  ind0_te = -1,
  indn_te = -1,
  maxsize_te = -1,
  c_suffix_te = 'NONE',
  nc_var_te = 'votemper',
  nc_att_mask_te = 'NONE',
/

&SALINITY  
  c_dir_sa = '/results/SalishSea/nowcast/01jan16',
  c_prefix_sa = 'SalishSea_lh_20160101_20160101_grid_T.nc',
  ind0_sa = -1,
  indn_sa = -1,
  maxsize_sa = -1,
  c_suffix_sa = 'NONE',
  nc_var_sa = 'vosaline',
  nc_att_mask_sa = 'NONE',
/

&MESH  
  dir_mesh = '/data/nsoontie/MEOPAR/NEMO-forcing/grid/',
  fn_mesh = 'mesh_mask_SalishSea2.nc',
  nc_var_xx_tt = 'glamt',
  nc_var_xx_uu = 'glamu',
  nc_var_yy_tt = 'gphit',
  nc_var_yy_vv = 'gphiv',
  nc_var_zz_ww = 'gdepw',
  nc_var_e2u = 'e2u',
  nc_var_e1v = 'e1v',
  nc_var_e1t = 'e1t',
  nc_var_e2t = 'e2t',
  nc_var_e3t = 'e3t',
  nc_var_tmask = 'tmask',
  nc_mask_val = 0.,
/

Key namelist parameters

There are some key differences between the namelists in quantitative and qualitative mode. Pay special attention to the following options:

- **nmax**: The maximum number of particles. This parameter is typically much higher in quantitative mode.
- **key_eco**: Setting to .TRUE. reduces CPU time.
- **key_reducMem**: Setting to .TRUE. reduces memory by only reading model data over selected region.
- **key_unitm3**: Setting to .TRUE. prints transport calculation in m^3/s instead of Sverdrups.
- **max_transport**: Maximum transport (in m^3/s) that should not be exceeded by the transport associated with each initial particle. A lower values means more initial particles and higher accuracy. Example values are 1e9 for one particle in one model cell and 1e4 for typical experiments.
- **lmin**: First time step to generate particles.
- \texttt{lmax}: Last time step to generate particles.
- \texttt{key\_alltracers}: .\texttt{TRUE.} to print tracer information in diagnostics.
- \texttt{key\_computesigma}: .\texttt{TRUE.} to compute density from temperature and salinity.
- \texttt{zsigma}: reference level for sigma computation

**Defining Sections**

You must define a closed region in your domain for transport calculations. Ariane calculates the mass transport between an initial section in your region and the other sections. Ariane provides a couple of useful tools for defining the sections.

- \texttt{mkseg0}: This program reads your land-ocean mask and writes it as a text file. Run this program in the same directory as your namelist. You may need to add the ariane executables to your path.

```
mkseg0
```

- \texttt{segrid}: After you run \texttt{mkseg0}, you should see a new file called \texttt{segrid}. Edit this file with

```
nedit segrid
```

- If you turn off text wrapping, you might see something like this:

Land points are \# and ocean points are -. 

- Add numbered sections to this file. Be sure your sections are over ocean points and not land points. Ariane will initialize particles along the section labelled as 1 and will calculate transport through all other sections. Your sections must make up a closed volume. Place the @ symbol somewhere within your closed subdomain. Your final edit might look something like this.

```
Run \texttt{mkseg}
```

2.14. Particle Tracking 233
Copy the section definitions into a file called `sections.txt`. The section definitions can be found from the output of `mkseg`. `sections.txt` should look something like this:

```
1 250 313 -409 -409 1 40 "1section"
2 264 312 386 386 1 40 "2section"
3 1 398 1 898 0 0 "Surface"
```

You can rename "1section" and "2section" to something more intuitive if you desire. You should also add a "Surface" section as above.

- Run `ariane`. Remember to check that you have added the `ariane` executable to your path.

The output on the screen should indicate that `ariane` completed successfully. You should also see a new file called `stats.txt`. This file contains statistics about the initial and final particles through each section and the transport calculations. It might look something like this:

```
total transport (in m3/s): 230033.88767527405 ( x lmt = 5520813. → 3042065771 )
max_transport (in m3/s) : 1000000000.0000000
# particles : 20380
initial state # 20380
stats. for: x y z a
min: -123.457 48.946 0.500 0.000
max: -123.134 49.063 226.275 0.000
mean: -123.347 48.986 74.893 0.000
std. dev.: 0.062 0.022 61.722 0.000
```

(continues on next page)
meanders 166079.1572 0
1section .0000 1
2section 63952.7799 2
Surface .0000 3
total 230033.8877
except mnds 63954.7305
lost 1.9506

final state meanders # 11094
stats. ini: x y z a
min: -123.457 48.946 0.500 0.006
max: -123.134 49.063 226.275 16.858
mean: -123.343 48.987 91.665 1.010
std. dev.: 0.055 0.020 61.438 1.010
stats. fin: x y z a
min: -123.458 48.945 0.019 0.006
max: -123.132 49.064 238.621 16.858
mean: -123.329 48.992 91.483 0.606
std. dev.: 0.052 0.019 62.670 1.010

final state 2section # 9285
stats. ini: x y z a
min: -123.457 48.946 0.500 0.191
max: -123.134 49.063 226.275 16.074
mean: -123.357 48.982 31.337 1.715
std. dev.: 0.075 0.028 35.675 1.639
stats. fin: x y z a
min: -123.317 48.883 0.058 0.191
max: -123.079 48.970 151.722 16.074
mean: -123.192 48.929 25.504 1.715
std. dev.: 0.068 0.025 25.477 1.639

- lost are the particles not intercepted by any section.
- meanders are the particles that go back out the source section.

### Time considerations

Particles initialized later in the simulation do not have as much time to cross one of the sections so it could be beneficial to impose a maximum age of each particle. This can be achieved by modifying `mod_criter1.f90` in `src/ariane` as follows:

```
!----------------------------------------!
!- ADD AT THE END OF EACH LINE "!!ctr1" -!
!----------------------------------------!
!criter1=.FALSE. !! ctr1
!
!------------!
!- Examples -!
!------------!
!
criter1=(abs(hl-fl).ge. lmt-lmax) !! ctr1
```

- `lmt` and `lmax` should be substituted by the values you set in the namelist.
- You must remake and install ariane when making a change to any of the fortran files.
• In stats.txt, you will now see the particles intercepted by this time criterion.

```
meanders  135298.2260  0
1section  .0000  1
2section  13650.4035  2
Surface    .0000  3
Criter1    81085.2582  4
          total  230033.8877
          except mnds 94735.6616
          lost -.0000
```

---

**Defining and tracking water masses**

You can also impose a density and/or salinity and/or temperature criteria on the initial particles in order to track different water masses. You can achieve this by editing `mod_criter0.f90`.

```fortran
!criter0=.TRUE. !!ctr0
!
!------------!
!- Examples -!
!------------!

!- Examples -!

```fortran
!- Examples -!

`criter0=(zinter(ss,hi,hj,hk,hl).le.29_rprec) !!crt0`

• Once again, you must remake and install ariane.

• You’ll also need to make sure that `key_alltracers` and `key_computesigma` are .TRUE. and `zsigma` are defined in your namelist.

• Now particles will be initialized with salinity less than 29.

• There are other examples of useful criteria in `mod_criter0.f90`.

• Once again, the output of `stats.txt` will be different. Here is an example of part of `stats.txt`:

```
total transport (in m3/s): 76419.9825 ( x lmt = 1834079.
<5790278877 )
max_transport (in m3/s) : 1000000000.0000000
# particles : 16133

initial state # 16133
stats. for: x y z a t s r
min: -123.457 48.946 0.500 0.000 4.693 16.243 13.336
max: -123.134 49.063 45.041 0.000 9.960 29.000 22.816
mean: -123.333 48.991 15.077 0.000 8.526 27.842 22.038
std. dev.: 0.075 0.027 10.570 0.000 0.973 1.458 1.040
meanders  26404.6357  0
1section  1057.5257  1
2section 12998.1853  2
Surface    .0000  3
Criter1    35959.6357  4
          total  76419.9825
          except mnds 50015.3468
          lost .0000
```

• From the initial state statistics, you can see that the particles satisfy the salinity criteria. This might not be true of the final particles.
Ariane will also produce netCDF files `ariane_positions_quantitative.nc` and `ariane_statistics_quantitative.nc` that can be used to plot the particle trajectories and statistics.

## 2.15 Things We Learned About NEMO

This section documents things that we learned about various version of NEMO and its configurations in the course of development of the *Salish Sea NEMO* model. It also includes analyses and discussion of decisions that lead to *Salish Sea NEMO* being the way that it is.

### 2.15.1 Notes on the Oct-2013 CODE.tar Version

These are notes on building and running the version of NEMO configured for the Salish Sea as received in the `CODE.tar` archive prepared by J-P Paquin and downloaded at UBC on 2-Oct-2013.

The “Notes on NEMO/OPA Usage” document in the `DOCUMENTATION.tar` archive provided guidance for the creation of the notes below.

**Todo:** Add links to “Notes on NEMO/OPA Usage” .doc and PDF

### What's in CODE.tar

The `CODE.tar` tarball unpacks to a directory called `CONCEPTS110_WCSD_OW_NOBC_tide` which apparently indicates:

- **CONCEPTS110**: Canadian Operational Network of Coupled Environmental PredicTion Systems model at the version 110 level
- **WCSD**: MEOPAR West Coast Sub-Domain configuration
- **OW**: Open West boundary
- **NOBC**: ?? maybe refers to reading OBC file data instead of initial conditions ??
- **tide**: focus of the model runs is to calculate tides in the domain

`CONCEPTS110_WCSD_OW_NOBC_tide` contains `jcompile.sh`, as bash script that runs the build tool chain described below, and the `modipsl/` directory.

`modipsl` is the framework that was used to obtain and build the NEMO code prior to NEMO up to v3.2. `modipsl` is available via anonymous `svn` checkout from `http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/`. However, it is unclear what revision/tag of `modipsl` the tarball contains.

The documented (in “Notes on NEMO/OPA Usage” and in the NEMO Quick Start Guide) means of obtaining the NEMO code is to run:

```
cd modipsl/util
model NEMO
```

which automates the process of doing checkouts of code from a collection of `cvs` and `svn` repositories.

Examination of the `modipsl/util/log` file that the `modipsl/util/model` script creates and the the `modipsl/util/mod.def` definitions file for the script indicate that `model` was run on 18-Mar-2010 and that the `nemo_v3.1` tag was used to obtain the NEMO code.
Build Notes

Note: The build process described here was attempted on several platforms with the following results:

- MacBook running OS/X 10.8.5 with Xcode installed: fait_AA_make reported sed: RE error: illegal byte sequence numerous times,
- salish: fait_AA_make failed because ksh was not installed
- jasper: build was successful

The build NEMO 3.1 for a new configuration the following steps are required:

- modeles/NEMO/OPA_SRC/par_oce.F90 must be edited to set the number of processors to be used, and include configuration parameters via a .h90 file. The edits are done in C-preprocessor (CPP) blocks. J-P Paquin did this, using the CPP key key_wc3 to specify an 8x8 processor layout and parameters in par_WC3.h90. The edits are flagged with comments that start with !JPP.

- Based on modeles/NEMO/OPA_SRC/par_ORCA_R2.h90, create the configuration parameters header file included via the new CPP key in modeles/NEMO/OPA_SRC/par_oce.F90. The file created by J-P Paquin is par_WC3.h90.

- Edit modeles/UTIL/fait_config to define the NEMO source files that are required for the build. The configuration name, WC3 in the present case, must be added to the LIST shell variable, a newline-separated list of configuration names. A set -A statement must also be added to the collection that follows the LIST. The configuration is referenced in the set -A statement as DIR_configname; i.e. DIR_WC3 for the present case. The set -A statement lists the source code directories that are to be included in the build. Those edits are present in fait_config in the CODE.tar tarball.

- Run:

```
cd modeles/UTIL
./fait_config WC3
```

That results in the creation of the modeles/NEMO/WORK/ directory in which the source files from the directories given in the set -A DRI_WC3 ... statement in fait_config are symlinked so as to appear to be all in the same directory. The modipsl/config/WC3/ directory is also created. Its scripts/ directory contains the BB_make and BB_make.ldef files.

- Edit BB_make.ldef to activate/deactivate CPP keys for the configuration. The keys are listed in a single, space-separated line as the value of the P_P variable. modipsl/config/WC3/scripts/BB_make.ldef in CODE.tar contains three P_P lists identified with #-- JPP comments. The uncommented one (and so presumably most recently used) is:

```
#-- JPP 20130717 Run compilation keys for TEST1
P_P = key_wc3 key_dtatem key_dtasal key_flx_core key_vvl key_zrefsurf key_zdfk
    key_traldf_c2d key_dynldf_c3d key_mpp_mpi key_ldfslp key_dynspg_ts2 key_dtatem_month key_dtasal_month key_obc_mer key_tide key_diaharm
```

Also ensure that there is a prefix for preprocessing line for the build/run target, for example:

```
#-Q- jasper prefix = -D
```

- Ensure that there is an appropriate set of definitions in modipsl/util/AA_make.gdef for the build/run target, for example:
• Run:

```
cd modeles/NEMO
./UTIL/fait_AA_make
```

to calculate compilation rules, options, and build dependencies so as to create NEMO/WORK/AA_make (which is symlinked to modipsl/config/WC3/scripts/BB_make)

**Note:** fait_AA_make *must* be run from the modeles/NEMO/ directory.

• Run:

```
cd modipsl/util
./clr_make
./ins_make -t target
```

to remove existing Makefiles and create new ones. The target argument to ins_make specifies a compiler or host name defined in modipsl/util/AA_make.gdef and modipsl/config/WC3/scripts/BB_make.ldef.

• Run:

```
cd modipsl/config/WC3
make clean
make
```

to compile and link the code.

---

2.15. Things We Learned About NEMO 239
The results of a successful build are:

- a `../../bin/opa` executable
- a `../../lib/libioipsl.a` library
- a `../../lib/oce/libopa.a` library

Problems

With the CPP keys above in `BB_make.ldef` the `make` command on `jasper` completes with these messages:

```plaintext
dynadv_ppm.F90(76): warning #6843: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [PHTRA_ADV]
   SUBROUTINE adv_ppm_hor ( kt, pun, pvn, tra, traa, phtra_adv, z2, sort )
   ____________________________________________________________
./ldfdyn_c3d.h90(148): remark #8291: Recommended relationship between field width 'W' and the number of fractional digits 'D' in this edit descriptor is 'W>=D+7'.
   IF(lwp) WRITE(numout,'(34x,E7.2,8x,i3)') zcoef(jk) * ahm0, jk
   ____________________________________________________________
dynzdf_imp.F90(20): remark #6536: All symbols from this module are already visible due to another USE; the ONLY clause will have no effect. Rename clauses, if any, will be honored. [OCE]
   USE oce ! ocean dynamics and tracers
   _______
```

The library is up-to-date

```plaintext
mpifort -o ../../../bin/opa model.o ../../../lib/oce/libopa.a ../../../lib/libioipsl.a -L/lustre/jasper/software/netcdf/netcdf-4.1.3/lib -lnetcdf -lnetcdff -lhdff5_hl -lhdff5 -lz -lsz /lustre/jasper/software/intel/1_ics_2012.0.032/composer_xe_2011_sp1.10.319/compiler/ -lib/intel64/libimf.so: warning: warning: feupdateenv is not implemented and will always fail
OPA model is OK
```

Run Notes

The `WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY.tar` tarball contain the namelist and scripts to setup and run on the BIO HPC cluster:

- `linkfile.sh` links the initial conditions, forcing, etc. files into the run directory with the file names that NEMO expects
- `namelist` is the NEMO namelist for the run
- `submit_64.sh` is the file containing PBS directives and shell commands that is submitted to the TORQUE resource manager via `qsub`

The meaning of `WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY` (from J-P’s README.txt) is:

<table>
<thead>
<tr>
<th>WCSD</th>
<th>West Coast Sub Domain (398x345)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2</td>
<td>Run with only M2 tides from WebTide</td>
</tr>
<tr>
<td>OW_ON</td>
<td>Open West &amp; OpenNorth boundaries</td>
</tr>
<tr>
<td>file</td>
<td>reading OBC file (not initial conditions)</td>
</tr>
<tr>
<td>DAMP</td>
<td>increased horizontal eddy viscosity</td>
</tr>
<tr>
<td>ANALY</td>
<td>Analytical forcing (namsbc_ana) - no atm-ocean fluxes or atmospheric forcing</td>
</tr>
</tbody>
</table>
The WCSD_PREP.tar tarball contains the initial conditions, forcing, etc. files for the WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY case.

With those two tarballs unpacked beside each other the dirPREP variable in linkfile.sh need to be set to:

```
:dirPREP=../WCSD_PREP
```

and linkfile.sh run in WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY to prepare for the run.

submit_64.sh is tailored to the BIO HPC cluster. To run on jasper, the following script was used:

```
#!/bin/bash

#PBS -N WCSD_RUN_tide_M2_OW_ON_file_DAMP_ANALY
#PBS -S /bin/bash
#PBS -l procs=64
#PBS -l pmem=2gb
#PBS -l walltime=1:00:00
#PBS -m bea
#PBS -M dlatornell@eos.ubc.ca
#PBS -o OPA.output
#PBS -e OPA.output.error

cd $PBS_O_WORKDIR
echo working dir: $(pwd)

module load compiler/intel/12.1
module load library/intelmpi/4.0.3.008
module load library/netcdf/4.1.3
module load library/szip/2.1

mpiexec ./opa
```

If that script is stored as jasper.pbs, a run is submitted with the command:

```
qsub jasper.pbs
```

As an initial test, the run duration was set to 720 time steps via the &namrun.nitend namelist item. The run completed in just over 2 minutes. A subsequent 4320 time step run took about 17 minutes.

**Post-Processing**

The results of the runs described above are groups of 64 netCDF files (one for each processor) for each of the calculated quantities:

- U, V, W, and T
- output.init: initial time step output ??
- restart and open boundary condition restart
- 2D slice timeseries results
- tidal harmonics diagnostic results

---

2.15. Things We Learned About NEMO 241
NOCSCOMBINE

Google lead to the NOCSCOMBINE tool at ftp://ftp.soc.soton.ac.uk/omfftp/NE
MO/NOCSCOMBINE.tar. Building it on jasper required creation of a new makefile with NHOME and LIBS variable set to:

| NCHOME = /lustre/jasper/software/netcdf/netcdf-4.1.3 |
| LIBS = -L$(NCHOME)/lib -I$(NCHOME)/include -lnetcdf -lnetcdff -lhdf5_hl -lhdf5 -lz -lzs |

Commands like:

cd WCSD_RUN_tide_M2_OW_ON_FILE_DAMP_ANA/
  .../NOCSCOMBINE/nocscombine -f WC3_CU60_20020102_20020104_grid_U_0000.nc

result in the 64 pre-processor files of u velocity results being combined into a single WC3_CU60_20020102_20020104_grid_U.nc file. The process takes over 10 minutes per quantity for U, V, and T for the 72 hour run, and nearly 30 minutes for W.

2.15.2 Notes on NEMO v3.1

These are notes on building and running NEMO v3.1 from https://forge.ipsl.jussieu.fr/nemo/browser/tags/nemo_v3_1. The Installing NEMO using modipsl instructions and the modipsl documentation provided some guidance for the creation of the notes below.

Getting the Code

• Create a new working directory and do an svn trunk checkout of the modipsl framework:

  mkdir nemo31
  cd nemo31
  svn checkout http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl

• Edit modipsl/util/mod.def to add a NEMO_31 section containing:

  #H- NEMO_31 NEMO
  #H- NEMO_31 OPA
  #H- NEMO_31 LIM
  #H- NEMO_31 TOP
  #H- NEMO_31 IOIPSIL/src - svn - tag v2_1_4
  #H- NEMO_31 NEMO sources and configurations - svn - tag nemo_v3_1
  #M- NEMO_31 nemo_st@locean-ipsl.upmc.fr
  #C- NEMO_31 IOIPSIL/tags/v2_1_9/src HEAD 8 IOIPSIL/src modeles
  #C- NEMO_31 XMLF90 HEAD 12 XMLF90
  #C- NEMO_31 XMLIO_SERVER/trunk 51 12 XMLIO_SERVER
  #C- NEMO_31 tags/nemo_v3_1/EXTERNAL/XMLF90 HEAD 7 XMLF90/external
  #C- NEMO_31 tags/nemo_v3_1/EXTERNAL/XMLIO_SERVER HEAD 7 XMLIO_SERVER/external
  #C- NEMO_31 tags/libIGCM_v1_4 HEAD 10 libIGCM
  #C- NEMO_31 tags/nemo_v3_1/AGRIF HEAD 7 .

(continues on next page)
• Install NEMO v3.1 and its dependencies:

```
cd modipsl/util
./model NEMO_31
```

Your nemos-ocean.eu credentials are required for most of the checkouts that model does and you will be asked for your password for each one unless you have `svn` password caching configured.

### Build Notes

The build process described here was done on jasper.westgrid.ca.

• Load the following modules to configure the jasper environment:

```
module load compiler/intel/12.1
module load library/intelmpi/4.0.3.008
module load library/netcdf/4.1.3
module load library/szip/2.1
module load application/ncview/2.1.1
```

• For the initial test the GYRE configuration was used. Still working in the modipsl/util/ directory:

```
../modeles/UTIL/fait_config GYRE
```

• Edit `AA_make.gdef` to add a jasper section containing:

```
#-Q- jasper #- Global definitions for jasper.westgrid.ca using Linux Compiler
 \  \ Intel v8
 #=Q- jasper LIB_MPI = MPI2
 #=Q- jasper LIB_MPI_BIS =
 #=Q- jasper M_K = make
 #=Q- jasper P_C = cpp
 #=Q- jasper P_O = -P -C -traditional $(P_P)
```

(continues on next page)
Edit `../config/GYRE/BB_make.ldef` to add a jasper prefix for preprocessing:

```bash
#-Q- jasper prefix = -D
```

Run:

```bash
cd ../modeles/NEMO
..UTIL/fait_AA_make
```

to calculate compilation rules, options, and build dependencies so as to create `NEMO/WORK/AA_make` (which is symlinked to `modipsl/config/GYRE/scripts/BB_make`)

Note: `fait_AA_make` must be run from the `modeles/NEMO/` directory.

Remove any existing Makefiles and create new ones:

```bash
cd ../../util
./clr_make
./ins_make -t jasper
```

Compile and link the code:

```bash
cd ../config/GYRE
make clean
make
```

The results of a successful build are:

- a `../../bin/opa` executable
- a `../../lib/libioipsl.a` library
- a `../../lib/oce/libopa.a` library
2.15.3 Notes on Downloading/Running NEMO 3.4 on Ocean Cluster including Salish

Note, this set-up runs on one core on one processor.

Getting the Code

- Goto http://www.nemo-ocean.eu/
- Logon.
- Goto Using NEMO, then User Guides, the NEMO Quick Start Guide
- locally switch to BASH as your shell (e.g. type bash)
- check perl is installed (type perl and cntrl C when it sits on the new line wang)
- check svn is installed (type svn, it will suggest help for you)
- check you have a fortran compiler (on our systems gfortran)
- netcdf (not sure yet how you check this)
- then type

```
svn --username "sallen@eos.ubc.ca" co -r 3819 http://forge.ipsl.jussieu.fr/nemo/
→svn/branches/2012/dev_v3_4_STABLE_2012
```

- EXCEPT change my username to your username on the NEMO system.
- NOTE no < before svn unlike what they have on the website
- You will be prompted for your password
- and now you have the code

Making a Project

- Nemo uses an ARCH (architecture) file to determine compiler, maker, netcdf library location.

```
cd dev_v3_4_STABLE_2012/NEMOGCM/ARCH
```

- NEMO ships with a gfortran_linux file. This file needs some edits to work on ocean. Changes in bold.
- New ARCH file: arch-ocean.fcm containing:

```
# generic gfortran compiler options for linux
# NCDF_INC  netcdf include file
# NCDF_LIB  netcdf library
# FC        Fortran compiler command
# FCFLAGS   Fortran compiler flags
# FFLAGS    Fortran 77 compiler flags
# LD        linker
# LDFLAGS   linker flags, e.g. -L<lib dir> if you have libraries in a
# FFPFLAGS  pre-processing flags
# AR        assembler
# ARFLAGS   assembler flags
# MK        make
# USER_INC  additional include files for the compiler, e.g. -I<include dir>
```

(continues on next page)
# USER_LIB additional libraries to pass to the linker, e.g. -l<library>
%NCDF_INC  ***I/usr/include**
%NCDF_LIB  ***L/usr/lib -lnetcdfff**
%FC         gfortran
%FCFLAGS    -fdefault-real-8 -O3 -funroll-all-loops -fcray-pointer
%FFLAGS     %FCFLAGS
%LD         gfortran
%LDFLAGS    
%FPFPFLAGS  -P -C -traditional
%AR         ar
%ARFLAGS    -rs
%MK         **make**
%USER_INC   %NCDF_INC
%USER_LIB   %NCDF_LIB

- then change directory and make a project, e.g. then for a new GYRE configuration using your new arch file ocean

- Note we need to add the key_nosignedzero for our fortran 90 compiler

- Add the netcdf4 key to use netcdf4 capabilities

    cd ..;/CONFIG
    ./makenemo -m ocean -r GYRE -n MY_GYRE add_key "key_nosignedzero key_netcdf4"

- If you are compiling version 3.6 on salish use the following command instead:

    cd ..;/CONFIG
    ./makenemo -m GCC_SALISH -r GYRE -n MY_GYRE -j8

- If the following error comes up:

    /* Copyright (C) 1991-2012 Free Software Foundation, Inc.
    Error: Invalid character in name at (l)

then modify the arch-ocean.fcm file line:

    %FPFPFLAGS   -P -C -traditional

to:

    %FPFPFLAGS   -P -traditional

**Running the Code**

    cd MY_GYRE/EXP00
    nice ./opa

### 2.15.4 Notes on Downloading/Running NEMO 3.4 on Jasper

Set-up for runs on one processor or multiprocessors included.
Getting your Jasper Shell Ready

- make sure your shell is bash (echo $SHELL), if its not, write to Westgrid support and get it changed.
- follow the instructions in Loading Modules on HPC Clusters to manually load the necessary software component modules or edit your jasper $HOME/.bashrc to make them load automatically when you ssh into jasper.

Getting the Code

see Getting the Code in Notes on Downloading/Running NEMO 3.4 on Ocean Cluster including Salish

Making a Project

- NEMO uses an ARCH (architecture) file to determine compiler, maker, netcdf library location.

```bash
cd dev_v3_4_STABLE_2012/NEMOGCM/ARCH
```

- NEMO ships with a number of different arch files. After changes to Jasper’s operating system and updated arch file is needed:

```bash
# makefile definitions for mpif90 on jasper.westgrid.ca; based on ifort file from P. Myers group
#
# NCDF_INC netcdf include file
# NCDF_LIB netcdf library
# FC Fortran compiler command
# FCFLAGS Fortran compiler flags
# LD linker
# LDFLAGS linker flags, e.g. -L<lib dir> if you have libraries in a
# FPPFLAGS pre-processing flags
# AR assembler
# ARFLAGS assembler flags
# MK make
# USER_INC additional include files for the compiler, e.g. -I<include dir>
# USER_LIB additional libraries to pass to the linker, e.g. -l<library>

%NCDF_HOME /global/software/netcdf/netcdf-4.1.3
%HDF5_HOME /global/software/hdf5/hdf5-1.8.9
%NCDF_INC -I%NCDF_HOME/include
%NCDF_LIB -L%NCDF_HOME/lib -lnetcdff -lnetcdf -L%HDF5_HOME/lib -lhdf5 -lhdf5
%CPP cpp
%FC mpif90
%FCFLAGS -c -fpp -r8 -O3 -assume byterecl -convert big_endian -heap-arrays
%LD mpif90
%FFLAGS %FCFLAGS
%LDFLAGS -lstdc++
%FPPFLAGS -P -C -traditional
%AR ar
%ARFLAGS -r
%MK make
%USER_INC %NCDF_INC
%USER_LIB %NCDF_LIB
```

2.15. Things We Learned About NEMO
• then change directory and make a project, e.g. then for a new GYRE configuration using your new arch file
  ocean
• Add the netcdf4 key to use netcdf4 capabilities
• Exactly the same (except GYRE -> AMM12) for AMM12

```
cd ../CONFIG
./makenemo -m mpif90_jasper -r GYRE -n MY_GYRE add_key "key_netcdf4"
```

Running the Code: GYRE

• Go to your version (where you want the results to end up)

```
cd MY_GYRE/EXP00
```

• Created a .pbs run file. For a simple run of GYRE that could be

• PBS file: GYRE.pbs containing:

```
# Script for running simple GYRE configuration

#PBS -l procs=1
#PBS -l pmem=500mb
#PBS -l walltime=00:05:00

module load compiler/intel/12.1
module load library/intelmpi/4.0.3.008
module load library/netcdf/4.1.3
module load library/szip/2.1

module list

echo "Current working directory is `pwd`"

cd dev_v3_4_STABLE_2012/NEMOGCM/CONFIG/MY_GYRE/EXP00

echo "Current working directory is `pwd`"

echo "Starting run at: `date`"

./opa

echo "Program opa finished with exit code $? at: `date`"
```

• and run

```
qsub GYRE.pbs
```

Running the CODE: AMM12: 32 Processors

• Need to get the AMM12 forcing and initialization files, untar and unzip

```
curl -L0 http://dodsp.idris.fr/reee512/NEMO/amm12_inputs_v3_4.tar

cd dev_v3_4_STABLE_2012/NEMOGCM/CONFIG/MY_AMM12/EXP00/
tar xvf ~/amm12_inputs_v3_4.tar
gunzip *.gz

cd ~/amm12_input_v3_4.tar
```

• To make AMM12 run on multiple processors, edit the namelist file, changing the following lines
• Need a .pbs file for multiple core run PBS file: AMM_multi.pbs containing:

```bash
# Script for running multiple processor AMM12 configuration

#PBS -l procs=32
#PBS -l pmem=500mb
#PBS -l walltime=00:15:00

module load compiler/intel/12.1
module load library/intelmpi/4.0.3.008
module load library/netcdf/4.1.3
module load library/szip/2.1

module list

echo "Current working directory is `pwd`"

cd dev_v3_4_STABLE_2012/NEMOGCM/CONFIG/MY_AMM12/EXP00

echo "Current working directory is `pwd`"

mpiexec ./opa

• and run

qsub AMM12_multi.pbs
```

2.15.5 Notes on NEMO v3.4 Compiler Options


Note that FCFLAGS are fortran 90 flags and FFLAGS are fortran 77 flags.

• -c Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.

• -C == -check: Checks for certain conditions at run time. - with argument none Prevents all run-time checking.

• -fpp Runs the Fortran preprocessor on source files before compilation.

• -r8 == real_size 64 Makes default real and complex variables 8 bytes long. REAL declarations are treated as DOUBLE PRECISION (REAL(KIND=8)) and COMPLEX declarations are treated as DOUBLE COMPLEX (COMPLEX(KIND=8)).

• -O Specifies the code optimization for applications. - O3 is aggressive optimization. See second webpage above for the details.... long!

• -assume Tells the compiler to make certain assumptions.
  – [no]byterecl Determines whether units for the OPEN statement RECL specifier (record length) value in unformatted files are in bytes or longwords (four-byte units).

• -convert Specifies the format of unformatted files containing numeric data.
– **big_endian** Specifies that the format will be big endian for integer data and big endian IEEE floating-point for real and complex data.

• **-heap-arrays**: This option puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.
  
  – If heap-arrays is specified and size is omitted, all automatic and temporary arrays are put on the heap. If 10 is specified for size, all automatic and temporary arrays larger than 10 KB are put on the heap.

• **-extend-source**: Specifies the length of the statement field in a fixed-form source file. If you specify extend_source without size, the statement field ends at column 132.

• **-traceback**: Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time.

• **-openmp**: Enables the parallelizer to generate multi-threaded code based on the OpenMP* directives.

### 2.15.6 Understanding the AMM12 Namelist

The file is in sections:

```plaintext
!!!-----------------------------------------------------------------------
!!! NEMO/OPA : 1 - run manager  (namrun)
!!! namelists  2 - Domain     (namzgr, namzgr_sco, namdom, namtsd)
!!!               3 - Surface boundary (namsbc, namsbc_sco, namsbc_flx, namsbc_clio)
!!!               -> namsbc_core
!!!               4 - lateral boundary (namlbc, namcla, namobc, namagrif, nambdy)
!!!               -> nambdy_tide)
!!!               5 - bottom boundary (nambfr, nambbc, nambbl)
!!!               6 - Tracer     (nameos, namtra_adv, namtra_ldf, namtra_dmp)
!!!               7 - dynamics   (namdyn_adv, namdyn_vor, namdyn_sp, namdyn_sp)
!!!               -> namdyn_ldf)
!!!               8 - Vertical physics (namzdf, namzdef_ric, namzdf_tke, namzdf_kpp)
!!!               -> namzdf_ddm, namzdf_tmx)
!!!               9 - diagnostics (namnc4, namtrd, namspr, namflo, namptr, namhsb)
!!!               10 - miscellaneous (namsol, nammpp, nammpp_dyndist, namctl)
!!!               11 - Obs & Assim (nambobs, nam_asminc)  
!!!-----------------------------------------------------------------------

The first part is the Run Management (note MPI directives are further down, under misc)

• **nn_intend**: Number of time steps

• **nn_date0**: Date to start by

• **nn_write**: frequency of write in the NetCDF output files

• **nn_stock**: restart file frequency

and restart controls.

```
The next part is setting up the grid and bathymetry

- vertical coordinate, switch between z and s-coordinates
- \( rn_{\text{hmin}} \): set minimum ocean depth
- \( rn_{\text{c3*}} \): to do with partial steps
- \( rn_{\text{rdt}} \): time step
- \( rn_{\text{baro}} \): barotropic time steps
- \( rn_{\text{rd}} \): tracer time steps
- initialization of TS with input data, and damping back to that data.

```fortran
&namrun ! parameters of the run
!-----------------------------------------------------------------------
nn_no = 0 ! job number (no more used...)
cn_exp = "AMM12" ! experience name
nn_it000 = 1 ! first time step
nn_itend = 576 ! last time step (std 1 day = 576)
nn_date0 = 20070101 ! date at nit_0000 (format yyyymmdd) used if ln_rstart=F
  ! or (ln_rstart=T and nn_rstctl=0 or 1)
nn_leapy = 1 ! Leap year calendar (1) or not (0)
ln_rrestart = .false. ! start from rest (F) or from a restart file (T)
nn_rstctl = 0 ! restart control => activated only if ln_rrestart = T
  ! = 0 nn_date0 read in namelist ; nn_it000 : read in
  ! = 1 nn_date0 read in namelist ; nn_it000 : check
  ! = 2 nn_date0 read in restart ; nn_it000 : check
  ! referenced to 1
  ! referenced to nit000)
nn_write = 12 ! frequency of write in the output file (modulo nit000)
ln_dimgnnn = .false. ! DIMG file format: 1 file for all processors (F) or by processor (T)
ln_mskland = .false. ! mask land points in NetCDF outputs (costly; + ~15%)
nn_clobber = .false. ! clobber (overwrite) an existing file
nn_chunksz = 0 ! chunksize (bytes) for NetCDF file (works only with iom_{nf90} routines)

```

2.15. Things We Learned About NEMO
Part 3 is the surface boundary conditions

- **ln_flx true** sets fluxes from files, in namsbc_flx set file names and characteristics
- **ln_rnf** sets runoffs, in namsbc_rnf set files and configure river inflow
left in the penetrative light formulation. Suspect it is turned off here, but docs not clear.

```plaintext
!!======================================================================
!! *** Surface Boundary Condition namelists ***
!!======================================================================
!! namsbc  surface boundary condition
!! namsbc_ana  analytical formulation
!! namsbc_flx  flux formulation
!! namsbc_clio  CLIO bulk formulae formulation
!! namsbc_core  CORE bulk formulae formulation
!! namsbc_mfs  MFS bulk formulae formulation
!! namsbc_cpl  CouPLed formulation (**key_coupled**)
!! namtra_qsr  penetrative solar radiation
!! namsbc_rnf  river runoffs
!! namsbc_apr  Atmospheric Pressure
!! namsbc_ssr  sea surface restoring term (for T and/or S)
!! namsbc_alb  albedo parameters
!!======================================================================
!
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;namsbc</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

| nn_fsbc   | 1                     | frequency of surface boundary condition computation |
| ln_ana    | .false.               | analytical formulation (T => fill namsbc_ana) |
| ln_flx    | .true.                | flux formulation (T => fill namsbc_flx) |
| ln_blk_clio| .false.              | CLIO bulk formulation (T => fill namsbc_clio) |
| ln_blk_core| .false.             | CORE bulk formulation (T => fill namsbc_core) |
| ln_blk_mfs| .false.              | MFS bulk formulation (T => fill namsbc_mfs) |
| ln_cpl    | .false.              | Coupled formulation (T => fill namsbc_cpl) |
| ln_apr_dyn| .false.              | Patm gradient added in ocean & ice Eqs. (T => fill namsbc_apr) |
| nn_ice    | 0                     | =0 no ice boundary condition , |
|           |                       | =1 use observed ice-cover , |
|           |                       | =2 ice-model used (**key_lim3**)
| ln_dm2dc  | .false.              | daily mean to diurnal cycle on short wave |
| ln_rnf    | .true.                | runoffs (T => fill namsbc_rnf) |
| ln_ssr    | .false.              | Sea Surface Restoring on T and/or S (T => fill namsbc_ssr) |
| nn_fwb    | 0                     | FreshWater Budget: =0 unchecked |
|           |                       | =1 global mean of e-p-r set to zero at each time |
|           |                       | =2 annual global mean of e-p-r set to zero |
|           |                       | =3 global emp set to zero and spread out over |
|           |                       | =erp area |
| ln_cdgw   | .false.              | Neutral drag coefficient read from wave model (T => fill namsbc_wave) |
|-----------------------------------------------------------------------|

(continues on next page)

2.15. Things We Learned About NEMO

Salish Sea MEOPAR Documentation

253
&namsbc_flx  ! surface boundary condition : flux formulation
!
! file name ! frequency (hours) ! variable ! time interp. !
! (if <0 months) ! name ! (logical) ! (T/F)
! ! ! ! !
clim 'yearly' ! weights ! rotation!
! ! ! !

-> sn_utau = 'amm12_utau' , 1 , 'utau' , .false. , .false. , 'daily' , '

-> sn_vtau = 'amm12_vtau' , 1 , 'vtau' , .false. , .false. , 'daily' , '

-> sn_qtot = 'amm12_flx' , 3 , 'sonsfldo' , .true. , .false. , 'daily' , '

-> sn_qsr = 'amm12_flx' , 3 , 'soshfldo' , .true. , .false. , 'daily' , '

-> sn_emp = 'amm12_flx' , 3 , 'sowafldo' , .true. , .false. , 'daily' , '

-> cn_dir = './fluxes/' ! root directory for the location of the

! flux files
!

&namsbc_clio  ! namsbc_clio CLIO bulk formulae
!

NOT USED
!

&namsbc_core  ! namsbc_core CORE bulk formulae
!

NOT USED
!

&namsbc_mfs  ! namsbc_mfs MFS bulk formulae
!

NOT USED
!

&namsbc_cpl  ! coupled ocean/atmosphere model

("key_coupled")
!

NOT USED
!

&namtra_qsr  ! penetrative solar radiation
!
! file name ! frequency (hours) ! variable ! time interp. !
! (if <0 months) ! name ! (logical) ! (T/F)
! ! ! !
clim 'yearly' ! weights ! rotation!
! ! ! !

-> sn_chl = 'chlorophyll' , -1 , 'CHLA' , .true. , .true. , 'yearly' , '

-> cn_dir = './' ! root directory for the location of the runoff files

ln_traqsr = .false. ! Light penetration (T) or not (F)
ln_qsr_rgb = .true. ! RGB (Red-Green-Blue) light penetration
ln_qsr_2bd = .false. ! 2 bands light penetration
ln_qsr_bio = .false. ! bio-model light penetration

(continues on next page)
Section Four, Boundary Conditions

- Free slip along coasts $\text{rn}\_\text{shlat} = 0$
- No cross land advection through thin peninsulas $\text{nn}\_\text{cla} = 0$
- Open Boundaries: things to change: time relaxation for the different open boundaries
• Check AGRIF conditions: sponge layer in particular
• Unstructured open boundaries, may be able to greatly simplify or even remove
• nambdy_dta open boundary files
• nambdy_tide tide files

```fortran
!!======================================================================
!! *** Lateral boundary condition ***
!!======================================================================
!! namIBC  lateral momentum boundary condition
!! namCLA  cross land advection
!! namOBc  open boundaries parameters ("key_obc")
!! namAGrif agrif nested grid (read by child model only) ("key_agrif")
!! namBDY  Unstructured open boundaries ("key_bdy")
!! namTIDe Tidal forcing at open boundaries ("key_bdy_tides")
!!======================================================================
!
!-----------------------------------------------------------------------
& namIBC
! lateral momentum boundary condition
!-----------------------------------------------------------------------
rn_shlat = 0 ! shlat = 0 ! 0 < shlat < 2 ! shlat = 2 ! 2 < shlat
  < free slip ! partial slip ! no slip ! strong slip
ln_vorlat = .false. ! consistency of vorticity boundary condition with analytical eqs.

!-----------------------------------------------------------------------
& namCLA
! cross land advection
!-----------------------------------------------------------------------
nn_cla = 0 ! advection between 2 ocean pts separates by land

!-----------------------------------------------------------------------
& namOBc
! open boundaries parameters ("key_obc")
!-----------------------------------------------------------------------
ln_obc_clim = .false. ! climatological obc data files (T) or not (F)
ln_vol_cst = .true. ! impose the total volume conservation (T) or not (F)
ln_obc_fla = .false. ! Flather open boundary condition
rn_obcdta = 1 ! = 0 the obc data are equal to the initial state
  = 1 the obc data are read in 'obc.dta' files
cn_obcdta = 'annual' ! set to annual if obc datafile hold 1 year of data
  ! set to monthly if obc datafile hold 1 month of data
rn_dpein = 1. ! damping time scale for inflow at east open boundary
rn_dpwin = 1. !  --  !  --  !  west  !  --
rn_dpnin = 1. !  --  !  --  !  north  !  --
rn_dpsin = 1. !  --  !  --  !  south  !  --
rn_dpeob = 3000. ! time relaxation (days) for the east open boundary
rn_dpwob = 15. !  --  !  --  !  west  !  --
rn_dpnob = 3000. !  --  !  --  !  north  !  --
rn_dpsob = 15. !  --  !  --  !  south  !  --
rn_volemp = 1. ! = 0 the total volume change with the surface flux (E-P-R)
```

(continues on next page)
! = 1 the total volume remains constant

!-----------------------------------------------------------------------
&namagrif
! AGRIF zoom (*key_
→agrif")
!-----------------------------------------------------------------------
nn_cln_update = 3 ! baroclinic update frequency
ln_spc_dyn = .true. ! use 0 as special value for dynamics
rn_sponge_tra = 2880. ! coefficient for tracer sponge layer [m2/s]
rn_sponge_dyn = 2880. ! coefficient for dynamics sponge layer [m2/s]

!-----------------------------------------------------------------------
&nam_tide
! tide parameters (#ifdef key_tide)
!-----------------------------------------------------------------------
ln_tide_pot = .true. ! use tidal potential forcing
nb_harmo = 11 ! number of constituents used
ciname(1) = 'M2' ! name of constituent
ciname(2) = 'S2'
ciname(3) = 'N2'
ciname(4) = 'K1'
ciname(5) = 'O1'
ciname(6) = 'Q1'
ciname(7) = 'M4'
ciname(8) = 'K2'
ciname(9) = 'P1'
ciname(10) = 'Mf'
ciname(11) = 'Mm'

!-----------------------------------------------------------------------
&nambdy
! unstructured open boundaries (*key_bdy
→")
!-----------------------------------------------------------------------
nb_bdy = 1 ! number of open boundary sets
ln_coords_file = .true. ! =T : read bdy coordinates from file
cn_coords_file = 'coordinates.bdy.nc' ! bdy coordinates files
ln_mask_file = .false. ! =T : read mask from file
cn_mask_file = '' ! name of mask file (if ln_mask_file=.
→TRUE.)
nn_dyn2d = 2 ! boundary conditions for barotropic
→fields
nn_dyn2d_dta = 2 ! = 0, bdy data are equal to the
→initial state
→.nc' files
! = 1, bdy data are read in 'bdydata
→from files
! = 2, use tidal harmonic forcing data
→harmonic forcing
nn_dyn3d = 0 ! boundary conditions for baroclinic
→velocities
nn_dyn3d_dta = 0 ! = 0, bdy data are equal to the
→initial state
nn_tra = 1 ! boundary conditions for T and S
nn_tra_dta = 0 ! = 0, bdy data are equal to the
→initial state

(continues on next page)
nn_rimwidth = 10  ! width of the relaxation zone
ln_vol = .false.  ! total volume correction (see nn_
  volctl parameter)
nn_volctl = 1    ! = 0, the total water flux across
  open boundaries is zero
/
!-----------------------------------------------------------------------
&nambdy_dta    ! open boundaries - external data ("key_bdy")
!-----------------------------------------------------------------------
!   ! file name ! frequency (hours) ! variable ! time interpol.
!   ! clim ! 'yearly'/ ! weights ! rotation !
!   ! ! ! (if <0 months) ! name ! (logical)
!
->! (T/F) ! 'monthly'! filename ! pairing !
  bn_ssh = 'amm12_bdyT_u2d', 24 , 'sossheig', .true.
  , .false. , '', '', '',
  bn_u2d = 'amm12_bdyU_u2d', 24 , 'vobtcrtx', .true.
  , .false. , '', '', '',
  bn_v2d = 'amm12_bdyV_u2d', 24 , 'vobtcrty', .true.
  , .false. , '', '', '',
  bn_u3d = 'amm12_bdyU_u3d', 24 , 'vococrtx', .true.
  , .false. , '', '', '',
  bn_v3d = 'amm12_bdyV_u3d', 24 , 'vomecrty', .true.
  , .false. , '', '', '',
  bn_tem = 'amm12_bdyT_tra', 24 , 'votemper', .true.
  , .false. , '', '', '',
  bn_sal = 'amm12_bdyT_tra', 24 , 'vosaline', .true.
  , .false. , '', '', '',
  cn_dir = 'bdydta/
ln_full_vel = .false.
/
!-----------------------------------------------------------------------
&nambdy_tide    ! tidal forcing at open boundaries
!-----------------------------------------------------------------------
!  forcing files
filtide = 'bdydta/amm12_bdytide_'  ! file name root of tidal
  tide_cpt(1) = 'Q1'  ! names of tidal components used
  tide_cpt(2) = 'O1'  ! names of tidal components used
  tide_cpt(3) = 'P1'  ! names of tidal components used
  tide_cpt(4) = 'S1'  ! names of tidal components used
  tide_cpt(5) = 'K1'  ! names of tidal components used
  tide_cpt(6) = 'N2'  ! names of tidal components used
  tide_cpt(7) = 'M2'  ! names of tidal components used
  tide_cpt(8) = 'N2'  ! names of tidal components used
  tide_cpt(9) = 'N2'  ! names of tidal components used
  tide_cpt(10) = 'M2'  ! names of tidal components used
  tide_cpt(11) = 'L2'  ! names of tidal components used
  tide_cpt(12) = 'T2'  ! names of tidal components used
  tide_cpt(13) = 'S2'  ! names of tidal components used
  tide_cpt(14) = 'K2'  ! names of tidal components used
  tide_cpt(15) = 'M4'  ! names of tidal components used
  tide_speed(1) = 13.398661  ! phase speeds of tidal components (deg/hour)
  tide_speed(2) = 13.943036  ! phase speeds of tidal components (deg/hour)
  tide_speed(3) = 14.958932  ! phase speeds of tidal components (deg/hour)
  tide_speed(4) = 15.000001  ! phase speeds of tidal components (deg/hour)
  tide_speed(5) = 15.041069  ! phase speeds of tidal components (deg/hour)
  tide_speed(6) = 27.895355  ! phase speeds of tidal components (deg/hour)
  tide_speed(7) = 27.968210  ! phase speeds of tidal components (deg/hour)
tide_speed(8) = 28.439730 ! phase speeds of tidal components (deg/hour)
tide_speed(9) = 28.512585 ! phase speeds of tidal components (deg/hour)
tide_speed(10) = 28.984106 ! phase speeds of tidal components (deg/hour)
tide_speed(11) = 29.528479 ! phase speeds of tidal components (deg/hour)
tide_speed(12) = 30.000002 ! phase speeds of tidal components (deg/hour)
tide_speed(13) = 57.968212 ! phase speeds of tidal components (deg/hour)
l_n_tide_date = .true. ! adjust tidal harmonics for start date of run

Section 5: bottom boundaries

• probably don’t have to change anything here on first cut. May need to do later to get tides correct

&nambfr
! bottom friction

nn_bfr = 2 ! type of bottom friction : = 0 : free slip, = 1 : linear friction

rn_bfr1 = 4.e-4 ! bottom drag coefficient (linear case)
rn_bfr2 = 2.5e-3 ! bottom drag coefficient (non linear case)

ln_bfr2d = .false. ! horizontal variation of the bottom friction coefficient (read a 2D mask file)

rn_bfrien = 50. ! local multiplying factor of bfr (ln_bfr2d=1)

ln_bfrimp = .true. ! implicit bottom friction (requires ln_zdfexp = .false. if true)

&nambbc
! bottom temperature boundary condition

ln_trabbc = .false. ! Apply a geothermal heating at the ocean bottom

nn_geoflx = 2 ! geothermal heat flux: = 0 : no flux

rn_geoflx_cst = 86.4e-3 ! Constant value of geothermal heat flux [W/m2]

&nambbl
! bottom boundary layer scheme

nn_bbl_ldf = 0 ! diffusive bbl (=1) or not (=0)
nn_bbl_adv = 0 ! advective bbl (=1/2) or not (=0)

rn_ahtbbl = 1000. ! lateral mixing coefficient in the bbl [m2/s]

rn_gambbl = 10. ! advective bbl coefficient [s]
Section 6 : Tracers

- probably nothing to change here (No TEOS!)

```plaintext
!!----------------------------------------------------------------------------
!! Tracer (T & S ) namelists
!!----------------------------------------------------------------------------
!! nameos  equation of state
!! namtra_adv  advection scheme
!! namtra_ldf  lateral diffusion scheme
!! namtra_dmp  T & S newtonian damping
!!----------------------------------------------------------------------------
!
!-----------------------------------------------------------------------
&nameos
! ocean physical parameters
!-----------------------------------------------------------------------
nn_eos = 0  ! type of equation of state and Brunt-Vaisala
    frequency
        = 0, UNESCO (formulation of Jackett and
            McDougall (1994) and of McDougall (1987) )
        = 1, linear: rho(T) = rau0 * ( 1.028 - ralpha
            * T )
        = 2, linear: rho(T,S) = rau0 * ( rbeta * S -
            ralpha * T )
    rn_alpha = 2.0e-4  ! thermal expansion coefficient (nn_eos= 1 or 2)
    rn_beta = 7.7e-4  ! saline expansion coefficient (nn_eos= 2)
/
!-----------------------------------------------------------------------
&namtra_adv
! advection scheme for tracer
!-----------------------------------------------------------------------
ln-traadv_cen2 = .false.  ! 2nd order centered scheme
ln-traadv_tvd = .true.   ! TVD scheme
ln-traadv_muscl = .false.  ! MUSCL scheme
ln-traadv_musc12 = .false.  ! MUSCL2 scheme + cen2 at boundaries
ln-traadv_ubs = .false.  ! UBS scheme
ln-traadv_qck = .false.  ! QUICKEST scheme
/
!-----------------------------------------------------------------------
&namtra_ldf
! lateral diffusion scheme for tracers
!-----------------------------------------------------------------------
  ! Operator type:
ln-traldf_lap = .true.  ! laplacian operator
ln-traldf_bilap = .false.  ! bilaplacian operator
  ! Direction of action:
ln-traldf_level = .false.  ! iso-level
ln-traldf_hor = .true.   ! horizontal (geopotential)  (needs "key_ldfslp
when ln_sco=T")
ln-traldf_iso = .false.  ! iso-neutral    (needs "key_ldfslp
"")
  ! Griffies parameters  (all need "key_ldfslp
"")
ln-traldf_grif = .false.  ! use griffies triads
ln-traldf_gdia = .false.  ! output griffies eddy velocities
```

(continues on next page)
ln_triad_iso = .false.  ! pure lateral mixing in ML
ln_botmix_grif = .false.  ! lateral mixing on bottom
! ! Coefficients
! Eddy-induced (GM) advection always used with Griffies; otherwise needs "key_traldf_eiv"
! Value rn_aeiv_0 is ignored unless = 0 with Held-Larichev spatially varying
! (key_traldf_c2d & key_traldf_eiv & keyorca_r2, _r1 or _r05)

rn_aeiv_0 = 0.  ! eddy induced velocity coefficient [m2/s]
rn_sht_0 = 50.  ! horizontal eddy diffusivity for tracers [m2/s]
rn_shtb_0 = 0.  ! background eddy diffusivity for ldf_iso [m2/s]
! (normally=0; not used with Griffies)

nl_hpg = .false.  ! add a damping termn (T) or not (F)

-----------------------------------------------------------------------
&namtra_dmp  ! tracer: T & S newtonian damping
-----------------------------------------------------------------------
ln_tradmp = .false.  ! add a damping termn (T) or not (F)

7. Dynamics

• note: cpp keys matter here too
• AMM12 has split-explicit free surface key_dynspg_ts
• key_ldfslp for s-coordinates, won’t need in z
• key_zdfgls for GLS vertical mixing
• Hydrostatic pressure depends on z coordinate ln_hpg
• Horz Eddy viscosity set here rn_ahm_0_lap = 60.0 m2/s
• Vert Eddy viscosity/diffusivity rn_avt0, rn_avm0 = 0.1e-6 m2/s

-----------------------------------------------------------------------
&namdyn_adv  ! formulation of the momentum advection
-----------------------------------------------------------------------
ln_dynadv_vec = .true.  ! vector form (T) or flux form (F)
ln_dynadv_cen2 = .false.  ! flux form - 2nd order centered scheme
ln_dynadv_ubs = .false.  ! flux form - 3rd order UBS scheme

-----------------------------------------------------------------------
&namdyn_vor  ! option of physics/algorithm (not control by CPP keys)
-----------------------------------------------------------------------

ln_dynvor_ene = .false.  ! energy conserving scheme
ln_dynvor_ens = .false.  ! enstrophy conserving scheme
ln_dynvor_mix = .false.  ! mixed scheme
ln_dynvor_een = .true.  ! energy & enstrophy scheme
/
!-----------------------------------------------------------------------
&namdyn_hpg  ! Hydrostatic pressure gradient option
!=======================================================================
ln_hpg_zco = .false.  ! z-coordinate - full steps
ln_hpg_zps = .false.  ! z-coordinate - partial steps (interpolation)
ln_hpg_sco = .true.  ! s-coordinate (standard jacobian formulation)
ln_hpg_djc = .false.  ! s-coordinate (Density Jacobian with Cubic polynomial)
ln_hpg_prj = .false.  ! s-coordinate (Pressure Jacobian scheme)
ln_dynhpg_imp = .false.  ! time stepping: semi-implicit time scheme (T)
                   !    centered time scheme (F)
/
!-----------------------------------------------------------------------
!&namdyn_spg  ! surface pressure gradient  (CPP key only)
!=======================================================================
! ! explicit free surface        ("key_
!  -->dynspg_exp")
! ! filtered free surface        ("key_
!  -->dynspg_flt")
! ! split-explicit free surface  ("key_
!  -->dynspg_ts")
/
!-----------------------------------------------------------------------
&namdyn_ldf  ! lateral diffusion on momentum
!=======================================================================
! Type of the operator :
ln_dynldf_lap = .true.  ! laplacian operator
ln_dynldf_bilap = .true.  ! bilaplacian operator
   ! Direction of action :
ln_dynldf_level = .false.  ! iso-level
ln_dynldf_hor = .true.  ! horizontal (geopotential) (require
  "key_ldfslp" in s-coord.)
ln_dynldf_iso = .false.  ! iso-neutral (require
    "key_ldfslp")
   ! Coefficient
rn_ahm_0_lap = 60.0  ! horizontal laplacian eddy viscosity [m2/s]
rn_ahmb_0 = 0.0  ! background eddy viscosity for ldf_iso [m2/s]
rn_ahm_0_bilp = -1.0e+10  ! horizontal bilaplacian eddy viscosity [m4/s]
/
!!======================================================================
!! Tracers & Dynamics vertical physics namelists
!!=======================================================================
!! namzdf  vertical physics
!! namzdf_ric  richardson number dependent vertical mixing  ("key_
  -->zdfric")
!! namzdf_tke  TKE dependent vertical mixing  ("key_
  -->zdftke")
!! namzdf_kpp  KPP dependent vertical mixing  ("key_
  -->zdfkpp")
!! namzdf_ddm  double diffusive mixing parameterization  ("key_
  -->zdfddm")
!! namzdf_tmix tidal mixing parameterization  
!! (continued from previous page)
!!======================================================================
!-----------------------------------------------------------------------
&namzdf
! vertical physics
!-----------------------------------------------------------------------
rn_avm0 = 0.1e-6 ! vertical eddy viscosity [m2/s]  
  (background Kz if not "key_zdfcst")
rn_avt0 = 0.1e-6 ! vertical eddy diffusivity [m2/s]  
  (background Kz if not "key_zdfcst")
nn_avb = 0 ! profile for background avt & avm (=1) or not (=0)
nn_havtb = 0 ! horizontal shape for avtb (=1) or not (=0)
ln_zdfevd = .false. ! enhanced vertical diffusion (evd) (T) or not (F)
nn_evdm = 1 ! evd apply on tracer (=0) or on tracer and momentum  
  (=1)
rn_avevd = 100. ! evd mixing coefficient [m2/s]
ln_zdfnpc = .false. ! Non-Penetrative Convective algorithm (T) or not (F)
nn_npc = 1 ! frequency of application of npc
nn_npcp = 365 ! npc control print frequency
ln_zdfexp = .false. ! time-stepping: split-explicit (T) or implicit (F)  
  (time stepping)
nn_zdfexp = 3 ! number of sub-timestep for ln_zdfexp=T

!-----------------------------------------------------------------------
&namzdf_ric
! richardson number dependent vertical diffusion  
! ("key_zdfric")
! NOT USED
/

!-----------------------------------------------------------------------
&namzdf_tke
! turbulent eddy kinetic dependent vertical diffusion  
! ("key_zdftke")
! NOT USED
/

!-----------------------------------------------------------------------
&namzdf_kpp
! K-Profile Parameterization dependent vertical mixing  
! ("key_zdfkpp", and optionally:
! "kppcustom" or "key_kpplktb")
! NOT USED
/

!-----------------------------------------------------------------------
&namzdf_gls
! GLS vertical diffusion  
! ("key_zdfgls")
!-----------------------------------------------------------------------
rn_emin = 1.e-6 ! minimum value of e [m2/s2]
rn_epsmin = 1.e-12 ! minimum value of eps [m2/s3]
ln_length_lim = .true. ! limit on the dissipation rate under stable  
  stratification (Galperin et al., 1988)
rn_clim_galp = 0.53 ! galperin limit
ln_crban = .true. ! Use Craig & Banner (1994) surface wave mixing  
  parametrisation
ln_sigpsi = .true. ! Activate or not Burchard 2001 mods on psi schmidt  
  number in the wb case
rn_crban = 100. ! Craig and Banner 1994 constant for wb tke flux

(continues on next page)
rn_charn = 100000.  ! Charnock constant for wb induced roughness length
nn_tkebc_surf = 1  ! surface tke condition (0/1/2=Dir/Neum/Dir Mellor-
                    →Blumberg)
nn_tkebc_bot = 1  ! bottom tke condition (0/1=Dir/Neum)
nn_psi_bc_surf = 1 ! surface psi condition (0/1/2=Dir/Neum/Dir Mellor-
                     →Blumberg)
nn_psi_bc_bot = 1 ! bottom psi condition (0/1=Dir/Neum)
nn_stab_func = 2  ! stability function (0=Galp, 1= KC94, 2=CanutoA,
                   →3=CanutoB)
nn_clos = 1      ! predefined closure type (0=MY82, 1=k-eps, 2=k-w,  
                   →3=Gen)

!-----------------------------------------------------------------------
&namzdf_ddm
! double diffusive mixing parameterization ("key_
    →zdffddm")
!-----------------------------------------------------------------------
NOT USED
/
!-----------------------------------------------------------------------
&namzdf_tmx
! tidal mixing parameterization ("key_
   →zdftmx")
!-----------------------------------------------------------------------
NOT USED
/

9. Diagnostics (see below (switched order in this namelist)
10. Misc.
   • mpi settings for blocks are here, jpnj, jpnj, jpnj

[Continued on next page]
9. Diagnostics

- NetCDF chunking and compressions set here nn_nchunks
- Float parameters would be set here too
- Harmonic analysis of tidal constituents set here!

```fortran
!-----------------------------------------------------------------------
! cn_mpi_send = 'I' ! mpi send/recieve type = 'S', 'B', or 'I' for
! standard send, ! buffer blocking send or immediate non-blocking
! sends, resp. ! size in bytes of exported buffer ('B' case), 0 no
! nn_buffer = 0 ! activate code to avoid mpi_allgather use at the
! exportation ! northfold
! ln_nogather = .false. ! jpnij number of local domains (set automatically
! ! automatically if < 1) ! automatically if < 1)
! jpnj = 0 ! jpnij number of processors following j (set
! ! automatically if < 1) ! automatically if < 1)
! jpni = 0 ! jpni number of processors following i (set
! ! automatically if < 1)
! jpnij = 0
! !-----------------------------------------------------------------------
! &namctl ! Control prints & Benchmark
!-----------------------------------------------------------------------
! ln_ctl = .false. ! trends control print (expensive!)
! nn_print = 0 ! level of print (0 no extra print)
! nn_ictls = 0 ! start i indice of control sum (use to compare mono
! ! versus ! over a
! !subdomain) ! end i indice of control sum multi
! nn_jctls = 0 ! start j indice of control
! !processor runs ! over a
! nn_jctle = 0 ! end j indice of control
! nn_isplt = 1 ! number of processors in i-direction
! nn_jsplt = 1 ! number of processors in j-direction
! nn_bench = 0 ! Bench mode (1/0): CAUTION use zero except for bench
! ! (no physical validity of the results)
! nn_timing = 1 ! timing by routine activated (=1) creates timing.
! ! output file, or not (=0)
! /

9. Diagnostics

- NetCDF chunking and compressions set here nn_nchunks
- Float parameters would be set here too
- Harmonic analysis of tidal constituents set here!

```
&namnc4 ! netcdf4 chunking and compression settings  (**key_netcdf4**)
  !-----------------------------------------------------------------------
  nn_nchunks_i= 4 ! number of chunks in i-dimension
  nn_nchunks_j= 4 ! number of chunks in j-dimension
  nn_nchunks_k= 31 ! number of chunks in k-dimension
  ! setting nn_nchunks_k = jpk will give a chunk size
  of 1 in the vertical which is optimal for postprocessing which works
  exclusively with horizontal slabs
  ln_nc4zip = .true. ! (T) use netcdf4 chunking and compression
  ! (F) ignore chunking information and produce netcdf3-
  compatible files
  !-----------------------------------------------------------------------
&namtrd ! diagnostics on dynamics and/or tracer trends  (**key_trddyn** and/or **key_trdtra**)
  ! or mixed-layer trends or barotropic vorticity  (**key_trdml** or **key_trdvor**)
  !-----------------------------------------------------------------------
  NOT USED
  !-----------------------------------------------------------------------
&namgap ! level mean model-data gap  (**key_diagap**)
  !-----------------------------------------------------------------------
  NOT USED
  !-----------------------------------------------------------------------
&namflo ! float parameters  (**key_float**)
  !-----------------------------------------------------------------------
  NOT USED
  !-----------------------------------------------------------------------
&namptr ! Poleward Transport Diagnostic
  !-----------------------------------------------------------------------
  ln_diaptr = .false. ! Poleward heat and salt transport (T) or not (F)
  ln_diazml = .false. ! Add zonal means and meridional stream functions
  ln_subbas = .false. ! Atlantic/Pacific/Indian basins computation (T) or
  ! not (orca configuration only, need input basins mask)
  ! file named "subbasins.nc"
  ln_ptrcomp = .false. ! Add decomposition : overturning
  nn_fptr = 1 ! Frequency of ptr computation [time step]
  nn_fwri = 15 ! Frequency of ptr outputs [time step]
  !-----------------------------------------------------------------------
&namhsb ! Heat and salt budgets
  !-----------------------------------------------------------------------
  ln_diahsb = .false. ! check the heat and salt budgets (T) or not (F)
  !-----------------------------------------------------------------------
&nam_diaharm ! Harmonic analysis of tidal constituents (**key_diaharm**)
  !-----------------------------------------------------------------------
  nit000_han = 1 ! First time step used for harmonic analysis
  nitend_han = 75 ! Last time step used for harmonic analysis
(continues on next page)
nstep_han = 15 ! Time step frequency for harmonic analysis
tname(1) = 'M2' ! Name of tidal constituents
tname(2) = 'K1'

/  
!

nn_dct = 15 ! time step frequency for transports computing
nn_dctwri = 15 ! time step frequency for transports writing
nn_secdebug = 112 ! 0 : no section to debug
! -1 : debug all section
! 0 < n : debug section number n

11. Assimilation and Observation

• no changes here

2.15. Things We Learned About NEMO
2.15.7 CPP Keys

CPP keys used in C Pre-Processor directives in the source code that cause sections of code to be included/excluded from NEMO at the compile stage.

**Keys Common to AMM12 and CONCEPTS 110**

- **key_mpp_mpi** Use multiprocessors
- **key_vvl** Use variable volume coordinates. This means a nonlinear free surface.
- **key_ldfspl** Computes direction of lateral mixing.

**Keys in AMM12 that we need/want**

- **key_bdy** Use the unstructured open boundary conditions. We need these ones because they include tides. This replaces **key_obc_mer** the mercator open boundary conditions in CONCEPTS 110
- **key_vectopt_loop** Enables vector optimization. Good idea!
- **key_netcdf_4** Gives us chunking etc.

**Keys in CONCEPTS 110 that we need/want**

- **key_diaharm** calculate amplitude and phase of tidal components
- **key_tide** Required in addition to **key_diaharm** to enable calculation of amplitude and phase of tidal components. Also causes code for tidal potential to be included in the model and we are excluding that effect, so it is important to set **in_tide_pot** to **false**. in the nam_tide namelist.

**Decisions**

- AMM12 uses **key_dyngsp_c** Time splitting free surface. CONCEPTS 110 uses **key_dyngsp_c2**. We my need to return here.
- Turbulence scheme. CONCEPTS uses **key_zdftke**. AMM12 uses **key_zdfgls**. Start with tke and consider later.
- AMM12 uses **key_diaharm** which gives instantaneous fields rather than averages.
- CONCEPTS uses **key_diatem/key_datal** Read climatology initial temperature and salinity fields. Might be the best way to put in initial profile.
- **key_flx_core** Use short/long wave radiation forcing. Need to return to this issue later.
- Start with **key_trldf_c2d** horizontal eddy mixing of tracers from CONCEPTS
- Start with **key_dynldf_c3d** 3-dimensional mixing of momentum from CONCEPTS
CONCEPTS 110 that we don’t want

- **key-tide** Use tide potential. Not likely important in such a small domain.
- **key_zrefsurf** Use surface parameter set by Mercator
- **key_dtatem_month/key_dtasal_month** Read one time frame initial and temperature and salinity fields. Use with **key_dtatem/key_dtasal**.

### 2.15.8 Moving an AMM12 Configuration to JPP Configuration: Part One

**par_oce**

Note: Bottom Line: you need par_JPP.h90 and the new par_oce.F90 from the SALISH_amm/MY_SRC in your MY_SRC

The grid shape for a run is in a par_***.h90. So we need to create a par_JPP.h90 with the correct grid. These are just the changes

```fortran
cp_cfg = "WCSD" !: name of the configuration
jp_cfg = 120 , & !: resolution of the configuration (degrees) -- 1/
  \rightarrow 120
jpidta = 398, & !: first horizontal dimension > or = to jpi
jpjdta = 345, & !: second > or = to jpj
jpkdta = 40, & !: number of levels > or = to jpk
jpisl = 150, & !: number of islands
jpnisl = 3000 !: maximum number of points per island

ppkth = 25.00_wp , & !: (non dimensional): gives the approximate
  \rightarrow layer
ppacr = 3.0000000000_wp !: (non dimensional): stretching factor
  \rightarrow gdepw(jpk)
ppdzmin = 1._wp , & !: (meters) vertical thickness of the top
  \rightarrow gdepw(jpk)
pphmax = 500._wp !: (meters) Maximum depth of the ocean
```

Notes:
- **jp_cfg** has no effect as we are reading the coordinate files
- Islands copied from JPP
- **pp’s** define vertical grid

And this needs to be referenced in par_oce.F90. So add:

```fortran
#elif defined key_jpp
!!---------------------------------------------------------------------
!! 'key_jpp': Strait of Georgia Subdomain: JPP
!!---------------------------------------------------------------------
#include "par_JPP.h90"
```

2.15. Things We Learned About NEMO
Recompile

In your CONFIG directory (assuming you are using SALISH_amm as your config name)

```
./makenemo -n SALISH_amm add_key "key_jpp" del_key "key_amm_12km"
```

Initialization Files

Clone the nemo-forcing repository from here https://bitbucket.org/salishsea/nemo-forcing

**Note:** Bottom Line: you need files: SubDom_bathy_meter_NOBCchancomp.nc linked to bathy_meter.nc and SubDom_coordinates_seagrid_WestCoast.nc linked to coordinates.nc

Namelist File

Changes from AMM12 namelist file

- experience name `cn_exp`
- last time step `nn_itend`
- start date `nn_date0`
- frequency of restart file `nn_stock`
- frequency of write in output file `nn_write` *probably superceded*
- DIMG file format `ln_dimn` *probably superceded*
- vertical coordinate `ln_zps ln_sco`
- minimum depth of ocean `rn_hmin`
- partial step thickness `rn_e3ps_min rn_e3zps_rat`
- time step for the dynamics `rn_rdt`
- zero surface forcing `ln_ana ln_flx rn_utau0` *to be changed later!!*
- zero run off `ln_rnf` *to be changed later!!*
- light penetration `ln_traqsr`
- amount of slip on lateral boundaries `rn_shlat`
- climatological obs files `ln_obc_clim`
- total volume conservation `ln_vol_cst`
- use initial state for open boundary conditions `nn_obcdta` *to be changed later!!*
- time scales for open boundary relaxation `rn_dpeXob`
- turn tidal potential off `ln_tide_pot`
- number of tidal harmonics (for potential) & names `nb_harmo clname()`
- number of open boundary sets `nb_bdy` *to be changed next!*
- name of tide files, component names and speed `filtid tide_cpt() tide_speed()`
- bottom drag coefficient `rn_bfri2`
• bottom turbulent kinetic energy background \texttt{rn\_bfeb2}
• geothermal heat flux \texttt{nn\_geoflx}
• lateral mixing coefficient in the bbl \texttt{rn\_ahtbbl}
• horizontal eddy diffusivity for tracers \texttt{rn\_aht\_0}
• treatment of hydrostatic pressure \texttt{ln\_phg\_sco \ ln\_phg\_prj}
• bilaplacian operator \texttt{ln\_dynldf\_bilap}
• horizontal laplacian eddy viscosity \texttt{rn\_ahm\_0\_lap}
• vertical eddy viscosity and diffusivity \texttt{rn\_avm0 \ rn\_avt0}
• enhanced vertical diffusion and coefficient \texttt{ln\_zdfevd \ rn\_avevd}
• surface tke input coefficient \texttt{rn\_ebb}
• surface mixing length scale, Langmuir parameterization \texttt{ln\_mx10, \ ln\_lc}
• penetration of tke below mixed layer and its exponential decrease \texttt{nn\_etau \ nn\_htau}
• maximum iterations for the SOR solver and its coefficient \texttt{nn\_nmax \ nn\_sor}
• number of processors in various directions \texttt{jpni \ jpnj \ jpnij}
• control sums \texttt{nn\_ictlX \ nn\_jetlX}
• time step frequency for dynamics and tracer trends \texttt{nn\_trd}
• time steps used for harmonic analysis \texttt{nit000\_han \ nitend\_han \ nstep\_han}
• names for harmonic analysis \texttt{tname()}

\textbf{Note:} Bottom Line: you need to clone the namelist from https://bitbucket.org/salishsea/ss-run-sets see JPP. This run is based on hg changeset: b501af941889

\textbf{Run}

From inside your EXP00 directory, where you have linked to your forcing files you can run

```
mpiexec -n 16 ./opa
```

\textbf{2.15.9 Moving an AMM12 Configuration to JPP Configuration : Part Two: Add the Tides}

At the end of part one, we had working code with nothing but noise. Now we need to add the tides.

\textbf{Tide Forcing Files}

CONCEPTS 110 uses tide files that give the tides over the whole domain as amplitude and phase. Flow amplitude is in vertically integrated flux.

NEMO 3.4 uses tide files that give the tides only on the open boundary as cosine and sine components. Flow amplitudes are velocities.

In \textbf{Prepare Forcing Files} I calculate the latter from the former.

\textbf{2.15. Things We Learned About NEMO}
This produces three files: JPP_bdytide_M2_grid_X.nc where X = T, U or V. These files are available in the nemo-forcing repository in the bdydta folder.

**cpp Keys**

We have

```cpp
key_bdy key_vectopt_loop key_dynspq_ts key_ldfslp key_zdfglsl key_vvl key_diainstant
(key_mpp_mpi key_netcdf4 key_nosignedzero key_jpp
```

We need to

```cpp
add_key "key_diaharm key_tide key_zdftke key_traldf_c2d key_dynldf_c3d" del_key "key_zdfglsl"
```

So we rebuild (on Salish):

```bash
./makenemo -m salish -r SALISH_amm -n SALISH_JPP -j 8 add_key "key_diaharm key_tide key_zdftke key_traldf_c2d key_dynldf_c3d" del_key "key_zdfglsl"
```

**namelist**

Change:

```plaintext
nb_bdy = 1 ! number of open boundary sets
ln_coords_file = .false. ! =T: read bdy coordinates from file
cn_coords_file = '' ! bdy coordinates files

!-----------------------------------------------------------------------
&nambdy_index ! open boundaries - definition ("key_bdy")
!-----------------------------------------------------------------------

nbdysege = 0
nbdysegw = -1
nbdysegn = -1
nbdysegs = 0

filtide = 'bdydta/JPP_bdytide_' ! file name root of tidal forcing files
```

**Making it Work**

The above changes gave a code that ran for 295 time steps. It blew up with a large horizontal velocity error located at i=217, j=97 which is on the ridge north of the Chatham Island. Much work ensued but the final working version had the following changes. All this testing was done on Salish using 16 processors (4x4)

**Turbulence Scheme**

Turbulence scheme was changed back to GLS from TKE. This requires a change in key from key_zdftke to key_zdfglsl.
Horizontal Laplacian Viscosity

This was increased from 20.5 m²/s to 200 m²/s. Variable is called rn_ahm_0_lap. Increasing bottom friction (rn_bfeb2 and rn_bfri2) seems to increase the instability.

Topography Smoothing

Topography was hand smoothed in Chatham Islands region, Porlier Pass and in a deep basin south of Victoria. Details available in SalishSeaSubdomainBathy.ipynb.

2.15.10 AMM12 Boundary Coordinates

The AMM12-BDY-analysis.ipynb Jupyter Notebook is an analysis of the differences in results obtained from runs of the AMM12 configuration with the unstructured boundaries (BDY) set via the 2 available mechanisms in NEMO-3.4:

- using a coordinates.bdy.nc file (stock configuration)
- using a nambdy_index namelist

That analysis lead to our decision to use boundary coordinate calculations driven by the nambdy_index namelist for Salish Sea NEMO.

2.15.11 Notes on NEMO 3.6 Bugs and Quirks We Have Found

The issues described here are clearly bugs in some cases, and in other cases perhaps just things about NEMO that take time to understand. In either case, we hope that this page this will save someone some time!

Writes out to ocean.output that its 3.4

Top of the ocean.output file announces version number and it still says 3.4. This can be changed in nemogcm.F90 line 343.

Internally Calculated Vertical Grid Spacing

If you want to have the code calculate the vertical grid spacing, you need to set

| pp_sur   | 999999. |
| pp_a0    | 999999. |
| pp_a1    | 999999. |

as 999999. is the value of pp_to_be_computed. However, if the code writes out that it is calculating the vertical grid spacing if you set these values to 0 (but does not do anything). This can be fixed in domzgr.F90 by changing the 0._wp’s in this if statement to pp_to_be_computed.

Can’t reduce the number of Tidal Constituents

The base namelist_ref has 11 tidal constituents. The code sets the number of tidal constituents to the larger of that in namelist_ref or namelist_cfg. So with the base namelist it is not possible to run less than 11 tidal constituents. To fix this, remove all but one tidal constituent from namelist_ref.
Tidal Harmonics not Written Out Correctly

Tidal harmonics for bdy are now taken from tide.h90 in SBC. However, there is no call to tide_harmo before the tides are written out to ocean.output. To fix this, in bdytides.F90

- remove the bang (!) in front of USE tide_mod
- add

```fortran
call tide_harmo(omega_tide, v0tide, utide, ftide, ntide, nb_harmo)
```

before the tides are written.

- correct the units in the write statement, they are not deg/hr

```fortran
WRITE(numout,*),' Tidal cpt name - Phase speed (/s)'
```

Straight Boundary Segments from Namelist Crash

The allocated size of the arrays are not set correctly if you use straight boundary segments of less than a whole side and set them in the namelist. The variable jbdttau does not get set and this causes a memory error when the boundary data is read. To fix this add

```fortran
jbdttau = jpbdtas
```

in bdyini.F90, right after jpbdtas is calculated on line 482.

2.15.12 Notes on NEMO 3.4 Bugs and Quirks We Have Found

The issues described here are clearly bugs in some cases, and in other cases perhaps just things about NEMO that take time to understand. In either case, we hope that this page this will save someone some time!

Monthly (and Yearly) Averaging

Subroutine fldread.F90, NEMO v3_4_STABLE: Registered as trac issue #1201 and corrected fldread.F90 available there.

For monthly averaging, ztmp is initialized as a fractional part of a month, but inside the IF statement, it is added to in seconds.

I believe a similar error occurs in the yearly averaging.

Values go NaN

If you have unstable conditions and the values go NaN before your zonal velocities get greater than 20 m/s, the code will not stop. It will continue on calculating. In ocean.output Umax will be written out as 0 and minimum salinity will be written out as 100.

I have added NaN checking to our version of stpctl.F90 because if the code has gone unstable we don't want to waste the time waiting for completion. It probably slows the code down, so I don't suspect everyone will want to add it.

Email sallen@eos.ubc.ca if you want this code.
Isolated Ocean Grid Points Suppression Report

The list of isolated ocean grid points that is suppressed by OPA_SRC/DOM/domzgr.F90:zgr_bat_ctl() (i.e. single grid point bumps or holes in the bathymetry that are changed to match the level of adjacent grid points) is only output to ocean.output when the code is run on a single processor. Otherwise, it appears that only the isolated points (if any) on the MPI sub-domain on processor 0 are shown in ocean.output; for the Salish Sea domain that is none.

Values for Barotropic Boundary Conditions must be Interpolated

If you are using Tides + External Information on boundary values (nn_dyn2d_dta = 3), you must turn interpolation on for the barotropic files read. If you do not, the code will add the tides onto the previous value (of tides + external). If you do use interpolation, the code will recalculate the external value and then add the tides. So all is good.

Value and Use of nn_date0

The comment for the nn_date0 variable in the namrun in the AMM12 namelist (and probably other configurations too) from the SVN repo is very inaccurate:

```
! date at nit_0000 (format yyyymmdd) used if ln_rstart=F or (ln_rstart=T and nn_rstctl=0 or 1)
```

In models where tidal forcing is used via key_bdy and the nambdy_tide namelist, the value of nn_date0 is used to adjust the tidal forcing to the timeframe of the run (see Correcting tide for date: in the ocean. output file). That is the case regardless of the values of ln_rstart and nn_rstctl. So, the value of nn_date0 must be set to the day on which nn_it000 = 1 for the run even if the initial conditions are being supplied from a restart file.

We have revised our namelists to say:

```
! date at nit_0000 = 1 (format yyyymmdd)
! used to adjust tides to run date (regardless of restart control)
```

Also note that NEMO will accept and use some nonsensical values for nn_date0. For example, nn_date0 = 200209 (note that the day digits have been truncated) will result in tidal forcing being adjusted to a (biblical?) date of 9-Feb-20!

Salinity Extrema

Similar to others that have found extreme low temperatures in ORCA-LIM (see Trac Ticket 1180) we are getting extreme salinity values at depth, strongest near the Victoria Sill.

Reminder that we are running dev_v3.4_STABLE_2012 with vvl, partial z cells, tvd advection.

The advection scheme is not taking into account the partial cells, as shown in Fig 5.5 in the NEMO 3.4 book, but the Laplacian diffusion is. If we turn off the partial cell correction in Laplacian diffusion we improve things by about a factor of 10, but do not eliminate the problem.

For the deep Strait of Georgia it will be important to have the partial cell correction in the diffusion. Our plan is to keep that turned on and just to keep an eye on those extreme salinities as we go through a year of spin-up.
Possible Memory Leak in `timing.F90`

Several multi-day duration spin-up runs were terminated by the resource manager on `jasper` after 2000+ time steps because they substantially exceeded the per-processor memory requested for the jobs. Increasing the per-processor memory allowed the job to run longer but always resulted in termination for the same reason. By process of elimination it was found that the issue arose when the value of the `nn_timing` variable in the `&namctl` namelist was set to 1 but that runs were successful, with normal memory usage, when the value was set to 0.

A quick inspection of the `timing.F90` code reveals that there is memory allocation and pointer arithmetic in some of its subroutines, so a bug that is leaking memory is plausible. No deeper investigation was undertaken.

It is recommended that profiling be disabled by setting `nn_timing = 0` for all but short development runs lasting no longer than a few hundred time steps.

The problem has not been observed on `salish`, probably due to its less strict resource management. However, long runs with profiling enabled can be expected to fail when they exhaust machine memory on `salish`.

2.16 Citations

2.16.1 Model Configuration, Evaluation, and Storm Surge Hindcasting

The Salish Sea NEMO model configuration and its ability to calculate tides and sea surface height was evaluated by hindcasting storm surge events that occurred between 2002 and 2011 in:

to surge amplitude within the Strait of Georgia. Comparisons between simulations and high-resolution and low-resolution atmospheric forcing further emphasize that remote forcing is the dominant factor in surge amplitudes in this region. In addition, local wind patterns caused a slight increase in surge amplitude on the mainland side of the Strait of Georgia compared with Vancouver Island coastal areas during a major wind storm on 15 December 2006. Generally, surge amplitudes are found to be greater within the Strait of Georgia than in Juan de Fuca Strait.

doi = "10.1080/07055900.2015.1108899",

2.16.2 Carbon Chemistry and Aragonite Saturation State

The seasonal variability of aragonite saturation and pH in the surface Strait of Georgia and their drivers were determined using a 1-D coupled biochemical-physical model in:


@article {Moore-Maley-etal-2016,
  author = "Moore-Maley, Ben L. and Allen, Susan E. and Ianson, Debby",
  title = "Locally driven interannual variability of near-surface pH and A in the Strait of Georgia",
  journal = "Journal of Geophysical Research: Oceans",
  year = "2016",
  volume = "121",
  number = "3",
  pages = "1600--1625",
  issn = "2169-9291",
  url = "https://dx.doi.org/10.1002/2015JC011118",
  keywords = "Biogeochemical cycles, processes, and modeling, Carbon cycling, Estuarine processes, Marginal and semi-enclosed seas, Ecosystems, structure, dynamics, and modeling, acidification, estuarine, ecosystem, modeling, shellfish, rivers",
  abstract = "<p>Declines in mean ocean pH and aragonite saturation state (A) driven by anthropogenic CO2 emissions have raised concerns regarding the trends of pH and A in estuaries. Low pH and A can be harmful to a variety of marine organisms, especially those with calcium carbonate shells, and so may threaten the productive ecosystems and commercial fisheries found in many estuarine environments. The Strait of Georgia is a large, temperate, productive estuarine system with numerous wild and aquaculture shellfish and finfish populations. We determine the seasonality and variability of near-surface pH and A in the Strait using a one-dimensional, biophysical, mixing layer model. We further evaluate the sensitivity of these quantities to local wind, freshwater, and cloud forcing by running the model over a wide range of scenarios using 12 years of observations. Near-surface pH and A demonstrate strong seasonal cycles characterized by low pH, aragonite-undersaturated waters in winter and high pH, aragonite-supersaturated waters in summer. The aragonite saturation horizon generally lies at 20 m depth except in winter and during strong Fraser River freshets when it shoals to the surface. Periods of strong interannual variability in pH and aragonite saturation horizon depth arise in spring and summer. We determine that at different times of year, each of wind speed, freshwater flux, and cloud fraction are the dominant drivers of this variability. These results establish the mechanisms behind the emerging observations of highly variable near-surface carbonate chemistry in the

(continues on next page)
2.16.3 Turbulence and Advective Mixing

The sensitivity of the deep water renewal into the Strait of Georgia and of fresh water pulses into Juan de Fuca Strait to modelling choices affecting both turbulence and advection has been determined in:


This study investigates the sensitivity of a high resolution regional ocean model to several choices in mixing and advection. The oceanographic process examined is a deep water renewal event in the Juan de Fuca Strait–Strait of Georgia sill-basin estuarine system located on the west coast of North America. Previous observational work has shown that the timing of the renewal events is linked to the spring/neap tidal cycle, and in turn, is sensitive to the amount of vertical mixing induced by tidal currents interacting with sills and complicated bathymetry. It is found that the model’s representation of deep water renewal is relatively insensitive to several mixing choices, including the vertical turbulence closure and direction of lateral mixing. No significant difference in deep or intermediate salinity was found between cases that used kk versus kk closures and isoneutral versus horizontal lateral mixing. Modifications that had a stronger effect included those that involved advection such as modifying the salinity of the open boundary conditions which supply the source waters for the renewal event. The strongest impact came from the removal of the Hollingsworth instability, a kinetic energy sink in the energy-entrstrophy discretization of the momentum equations. A marked improvement to the salinity of the deep water renewal suggests that the removal of the Hollingsworth instability will correct a fresh drift in the deep and intermediate waters in an operational version of this model.

2.17 Salish Sea MEOPAR Project Contributors

The Salish Sea MEOPAR project is lead by Susan Allen in the Department of Earth, Ocean, and Atmospheric Sciences at the University of British Columbia. It is funded by the Marine Environment Observation Prediction And Response
The following people have contributed code, documentation, etc. to the Salish Sea MEOPAR project repositories hosted on Bitbucket:

- Susan Allen <sallen@eos.ubc.ca>
- Doug Latornell <dlatornell@eoas.ubc.ca>
- Nancy Soontiens <nsoontiens@eos.ubc.ca>
- Kate Le Souef <klesouef@eos.ubc.ca>
- Idalia Machuca <imachuca@eos.ubc.ca>
- Elise Olson <eolson@eos.ubc.ca>
- Michael Dunphy <mdunphy@eoas.ubc.ca>
- Muriel Dunn <mdunn@eos.ubc.ca>
- Tereza Jarníková <tjarniko@eos.ubc.ca>
- Jie Liu <jliu@eos.ubc.ca>
- Ben Moore-Maley <bmoorema@eos.ubc.ca>
- James Petrie <jpetrie@eos.ubc.ca>
- Giorgio Sgarbi <gsarbi@eos.ubc.ca>
- Vicky Do <vdo@eos.ubc.ca>
- Karina Ramos Musalem <kramosmusalem@eos.ubc.ca>
- Saurav Sahu <ssahu@eoas.ubc.ca>
- Birgit Rogalla <brogalla@eos.ubc.ca>
- Yingkai (Kyle) Sha <yingkai@eos.ubc.ca>
- Xiaoxin (Cindy) Yu <xiaoxiny@eos.ubc.ca>
- Melanie Grenier <mgrenier@eos.ubc.ca>
- Mark Halverson <mhalvers@eos.ubc.ca>
- Alex Huang
- Alyssia Law
- Dan Robb <drobb@eos.ubc.ca>
- Fatemeh Chegini <fatemeh.chegini@dal.ca>
- J-P Paquin <jpaquin.jeanphilippe@gmail.com>
- Youyu Lu <youyu.lu@dfo-mpo.gc.ca>
- Keith Thompson <keith.thompson@dal.ca>
- Vasily Korabel <v.korabel@dal.ca>
- Pramod Thupaki <pramod.thupaki@dfo-mpo.gc.ca>
BIBLIOGRAPHY


INDEX

E
EDITOR, 14
environment variable
   EDITOR, 14
   PATH, 14, 30, 39
   VISUAL, 14
   XIOS_HOME, 34, 40, 42, 63

P
PATH, 14, 30, 39

V
VISUAL, 14

X
XIOS_HOME, 34, 40, 42, 63