MITgcm Documentation

Release 1.0

Alistair Adcroft, Jean-Michel Campin, Stephanie Dutkiewicz, Constantinos Evangelinos, David Ferreira, Mick Follows, Gael Forget, Baylor Fox-Kemper, Patrick Heimbach, Chris Hill, Ed Hill, Helen Hill, Oliver Jahn, Martin Losch, John Marshall, Guillaume Maze, Dimitris Menemenlis and Andrea Molod

Jun 08, 2017

CONTENTS:

CHAPTER

ONE

OVERVIEW

This document provides the reader with the information necessary to carry out numerical experiments using MITgcm. It gives a comprehensive description of the continuous equations on which the model is based, the numerical algorithms the model employs and a description of the associated program code. Along with the hydrodynamical kernel, physical and biogeochemical parameterizations of key atmospheric and oceanic processes are available. A number of examples illustrating the use of the model in both process and general circulation studies of the atmosphere and ocean are also presented.

1.1 Introduction

MITgcm has a number of novel aspects:

- it can be used to study both atmospheric and oceanic phenomena; one hydrodynamical kernel is used to drive forward both atmospheric and oceanic models - see Figure 1.1

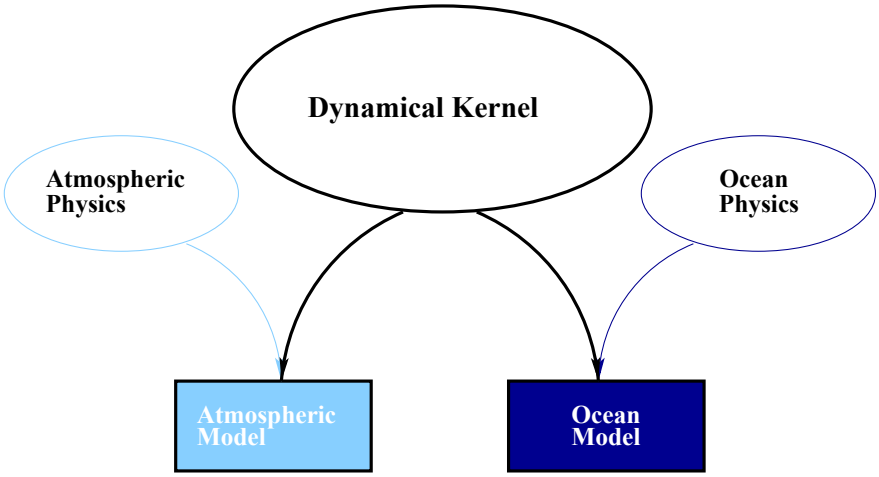

Figure 1.1: MITgcm has a single dynamical kernel that can drive forward either oceanic or atmospheric simulations.

- it has a non-hydrostatic capability and so can be used to study both small-scale and large scale processes - see Figure 1.2

- finite volume techniques are employed yielding an intuitive discretization and support for the treatment of irregular geometries using orthogonal curvilinear grids and shaved cells - see Figure 1.3

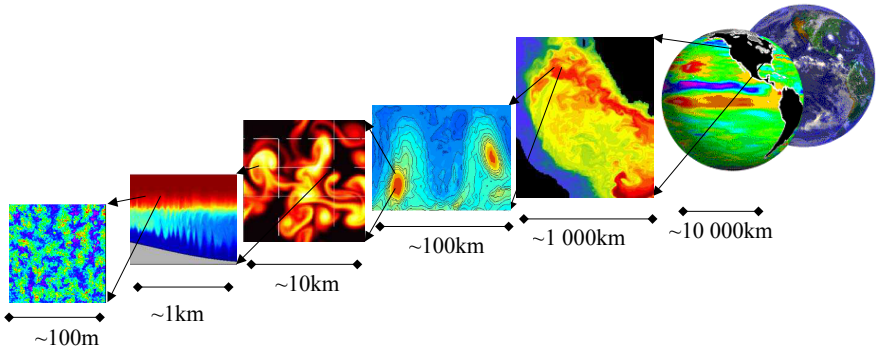

Figure 1.2: MITgcm has non-hydrostatic capabilities, allowing the model to address a wide range of phenomenon - from convection on the left, all the way through to global circulation patterns on the right.

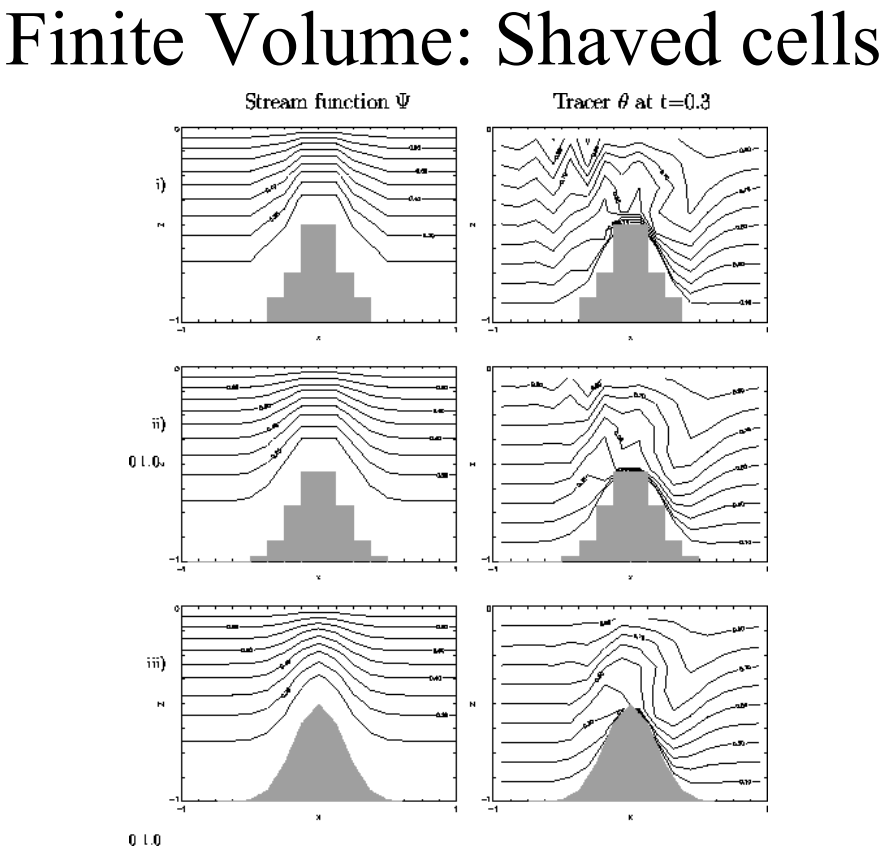

Figure 1.3: Finite volume techniques (bottom panel) are user, permitting a treatment of topography that rivals σ (terrain following) coordinates.

- tangent linear and adjoint counterparts are automatically maintained along with the forward model, permitting sensitivity and optimization studies.
- the model is developed to perform efficiently on a wide variety of computational platforms.

Key publications reporting on and charting the development of the model are *[HM95][MHPA97][MAH+97][AHM97][MJH98][AM99][CHM99][MGZ+99][AC04][ACHM04][MAC+04]* (an overview on the model formulation can also be found in *[AHC+04]*):

Hill, C. and J. Marshall, (1995) Application of a Parallel Navier-Stokes Model to Ocean Circulation in Parallel Computational Fluid Dynamics In Proceedings of Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers, 545-552. Elsevier Science B.V.: New York

Marshall, J., C. Hill, L. Perelman, and A. Adcroft, (1997) Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling J. Geophysical Res., 102(C3), 5733-5752.

Marshall, J., A. Adcroft, C. Hill, L. Perelman, and C. Heisey, (1997) A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers, J. Geophysical Res., 102(C3), 5753-5766.

Adcroft, A.J., Hill, C.N. and J. Marshall, (1997) Representation of topography by shaved cells in a height coordinate ocean model Mon Wea Rev, vol 125, 2293-2315

Marshall, J., Jones, H. and C. Hill, (1998) Efficient ocean modeling using non-hydrostatic algorithms Journal of Marine Systems, 18, 115-134

Adcroft, A., Hill C. and J. Marshall: (1999) A new treatment of the Coriolis terms in C-grid models at both high and low resolutions, Mon. Wea. Rev. Vol 127, pages 1928-1936

Hill, C, Adcroft,A., Jamous,D., and J. Marshall, (1999) A Strategy for Terascale Climate Modeling. In Proceedings of the Eighth ECMWF Workshop on the Use of Parallel Processors in Meteorology, pages 406-425 World Scientific Publishing Co: UK

Marotzke, J, Giering,R., Zhang, K.Q., Stammer,D., Hill,C., and T.Lee, (1999) Construction of the adjoint MIT ocean general circulation model and application to Atlantic heat transport variability J. Geophysical Res., 104(C12), 29,529-29,547.

We begin by briefly showing some of the results of the model in action to give a feel for the wide range of problems that can be addressed using it.

1.2 Illustrations of the model in action

MITgcm has been designed and used to model a wide range of phenomena, from convection on the scale of meters in the ocean to the global pattern of atmospheric winds - see figure ref{fig:all-scales}. To give a flavor of the kinds of problems the model has been used to study, we briefly describe some of them here. A more detailed description of the underlying formulation, numerical algorithm and implementation that lie behind these calculations is given later. Indeed many of the illustrative examples shown below can be easily reproduced: simply download the model (the minimum you need is a PC running Linux, together with a FORTRAN77 compiler) and follow the examples described in detail in the documentation.

subsection{Global atmosphere: 'Held-Suarez' benchmark} begin{rawhtml} <!– CMIREDIR:atmospheric_example: –> end{rawhtml}

A novel feature of MITgcm is its ability to simulate, using one basic algorithm, both atmospheric and oceanographic flows at both small and large scales.

Figure ref{fig:eddy_cs} shows an instantaneous plot of the 500mb temperature field obtained using the atmospheric isomorph of MITgcm run at $2.8^{circ }$ resolution on the cubed sphere. We see cold air over the pole (blue) and warm air along an equatorial band (red). Fully developed baroclinic eddies spawned in the northern hemisphere storm

track are evident. There are no mountains or land-sea contrast in this calculation, but you can easily put them in. The model is driven by relaxation to a radiative-convective equilibrium profile, following the description set out in Held and Suarez; 1994 designed to test atmospheric hydrodynamical cores - there are no mountains or land-sea contrast.

%% CNHbegin input{s_overview/text/cubic_eddies_figure} %% CNHend

As described in Adcroft (2001), a ‘cubed sphere’ is used to discretize the globe permitting a uniform griding and obviated the need to Fourier filter. The ‘vector-invariant’ form of MITgcm supports any orthogonal curvilinear grid, of which the cubed sphere is just one of many choices.

Figure ref{fig:hs_zave_u} shows the 5-year mean, zonally averaged zonal wind from a 20-level configuration of the model. It compares favorable with more conventional spatial discretization approaches. The two plots show the field calculated using the cube-sphere grid and the flow calculated using a regular, spherical polar latitude-longitude grid. Both grids are supported within the model.

%% CNHbegin input{s_overview/text/hs_zave_u_figure} %% CNHend

subsection{Ocean gyres} begin{rawhtml} <!– CMIREDIR:oceanic_example: –> end{rawhtml} begin{rawhtml} <!– CMIREDIR:ocean_gyres: –> end{rawhtml}

Baroclinic instability is a ubiquitous process in the ocean, as well as the atmosphere. Ocean eddies play an important role in modifying the hydrographic structure and current systems of the oceans. Coarse resolution models of the oceans cannot resolve the eddy field and yield rather broad, diffusive patterns of ocean currents. But if the resolution of our models is increased until the baroclinic instability process is resolved, numerical solutions of a different and much more realistic kind, can be obtained.

Figure ref{fig:ocean-gyres} shows the surface temperature and velocity field obtained from MITgcm run at $frac{1}{6}^{circ }$ horizontal resolution on a textit{lat-lon} grid in which the pole has been rotated by $90^{circ }$ on to the equator (to avoid the converging of meridian in northern latitudes). 21 vertical levels are used in the vertical with a ‘lopped cell’ representation of topography. The development and propagation of anomalously warm and cold eddies can be clearly seen in the Gulf Stream region. The transport of warm water northward by the mean flow of the Gulf Stream is also clearly visible.

%% CNHbegin input{s_overview/text/atl6_figure} %% CNHend

subsection{Global ocean circulation} begin{rawhtml} <!– CMIREDIR:global_ocean_circulation: –> end{rawhtml}

Figure ref{fig:large-scale-circ} (top) shows the pattern of ocean currents at the surface of a $4^{circ }$ global ocean model run with 15 vertical levels. Lopped cells are used to represent topography on a regular textit{lat-lon} grid extending from $70^{circ }N$ to $70^{circ }S$. The model is driven using monthly-mean winds with mixed boundary conditions on temperature and salinity at the surface. The transfer properties of ocean eddies, convection and mixing is parameterized in this model.

Figure ref{fig:large-scale-circ} (bottom) shows the meridional overturning circulation of the global ocean in Sverdrups.

%%CNHbegin input{s_overview/text/global_circ_figure} %%CNHend

1.2.1 Convection and mixing over topography

Dense plumes generated by localized cooling on the continental shelf of the ocean may be influenced by rotation when the deformation radius is smaller than the width of the cooling region. Rather than gravity plumes, the mechanism for moving dense fluid down the shelf is then through geostrophic eddies. The simulation shown in Figure 1.4 (blue is cold dense fluid, red is warmer, lighter fluid) employs the non-hydrostatic capability of MITgcm to trigger convection by surface cooling. The cold, dense water falls down the slope but is deflected along the slope by rotation. It is found that entrainment in the vertical plane is reduced when rotational control is strong, and replaced by lateral entrainment due to the baroclinic instability of the along-slope current.

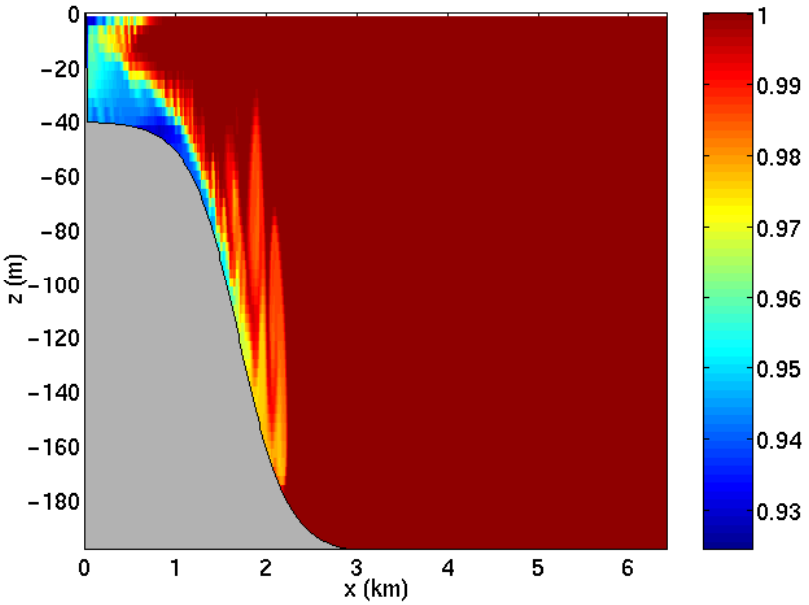

Figure 1.4: MITgcm run in a non-hydrostatic configuration to study convection over a slope.

1.2.2 Boundary forced internal waves

The unique ability of MITgcm to treat non-hydrostatic dynamics in the presence of complex geometry makes it an ideal tool to study internal wave dynamics and mixing in oceanic canyons and ridges driven by large amplitude barotropic tidal currents imposed through open boundary conditions.

Fig. ref{fig:boundary-forced-wave} shows the influence of cross-slope topographic variations on internal wave breaking - the cross-slope velocity is in color, the density contoured. The internal waves are excited by application of open boundary conditions on the left. They propagate to the sloping boundary (represented using MITgcm's finite volume spatial discretization) where they break under nonhydrostatic dynamics.

%%CNHbegin input{s_overview/text/boundary_forced_waves} %%CNHend

subsection{Parameter sensitivity using the adjoint of MITgcm} begin{rawhtml} <!– CMIREDIR:parameter_sensitivity: –> end{rawhtml}

Forward and tangent linear counterparts of MITgcm are supported using an 'automatic adjoint compiler'. These can be used in parameter sensitivity and data assimilation studies.

As one example of application of the MITgcm adjoint, Figure ref{fig:hf-sensitivity} maps the gradient $frac{partial J}{partial

mathcal{H}}$where J is the magnitude of the overturning

stream-function shown in figure ref{fig:large-scale-circ} at $60^{circ }N$ and $ mathcal{H}(lambda,varphi)$ is the mean, local air-sea heat flux over a 100 year period. We see that J is sensitive to heat fluxes over the Labrador Sea, one of the important sources of deep water for the thermohaline circulations. This calculation also yields sensitivities to all other model parameters.

%%CNHbegin input{s_overview/text/adj_hf_ocean_figure} %%CNHend

subsection{Global state estimation of the ocean} begin{rawhtml} <!– CMIREDIR:global_state_estimation: –> end{rawhtml}

An important application of MITgcm is in state estimation of the global ocean circulation. An appropriately defined

‘cost function’, which measures the departure of the model from observations (both remotely sensed and in-situ) over an interval of time, is minimized by adjusting ‘control parameters’ such as air-sea fluxes, the wind field, the initial conditions etc. Figure ref{fig:assimilated-globes} shows the large scale planetary circulation and a Hopf-Muller plot of Equatorial sea-surface height. Both are obtained from assimilation bringing the model in to consistency with altimetric and in-situ observations over the period 1992-1997.

%% CNHbegin input{s_overview/text/assim_figure} %% CNHend

subsection{Ocean biogeochemical cycles} begin{rawhtml} <!– CMIREDIR:ocean_biogeo_cycles: –> end{rawhtml}

MITgcm is being used to study global biogeochemical cycles in the ocean. For example one can study the effects of interannual changes in meteorological forcing and upper ocean circulation on the fluxes of carbon dioxide and oxygen between the ocean and atmosphere. Figure ref{fig:biogeo} shows the annual air-sea flux of oxygen and its relation to density outcrops in the southern oceans from a single year of a global, interannually varying simulation. The simulation is run at $1^{circ}times1^{circ}$ resolution telescoping to $frac{1}{3}^{circ}timesfrac{1}{3}^{circ}$ in the tropics (not shown).

%%CNHbegin input{s_overview/text/biogeo_figure} %%CNHend

subsection{Simulations of laboratory experiments} begin{rawhtml} <!– CMIREDIR:classroom_exp: –> end{rawhtml}

Figure ref{fig:lab-simulation} shows MITgcm being used to simulate a laboratory experiment inquiring into the dynamics of the Antarctic Circumpolar Current (ACC). An initially homogeneous tank of water ($1m$ in diameter) is driven from its free surface by a rotating heated disk. The combined action of mechanical and thermal forcing creates a lens of fluid which becomes baroclinically unstable. The stratification and depth of penetration of the lens is arrested by its instability in a process analogous to that which sets the stratification of the ACC.

%%CNHbegin input{s_overview/text/lab_figure} %%CNHend

section{Continuous equations in ‘r’ coordinates} begin{rawhtml} <!– CMIREDIR:z-p_isomorphism: –> end{rawhtml}

To render atmosphere and ocean models from one dynamical core we exploit ‘isomorphisms’ between equation sets that govern the evolution of the respective fluids - see figure ref{fig:isomorphic-equations}. One system of hydrodynamical equations is written down and encoded. The model variables have different interpretations depending on whether the atmosphere or ocean is being studied. Thus, for example, the vertical coordinate ‘r’ is interpreted as pressure, p, if we are modeling the atmosphere (right hand side of figure ref{fig:isomorphic-equations}) and height, z, if we are modeling the ocean (left hand side of figure ref{fig:isomorphic-equations}).

%%CNHbegin input{s_overview/text/zandpcoord_figure.tex} %%CNHend

The state of the fluid at any time is characterized by the distribution of velocity $vec{mathbf{v}}$, active tracers $theta $ and S, a ‘geopotential’ $phi $ and density $rho =rho (theta ,S,p)$ which may depend on $theta $, S, and p. The equations that govern the evolution of these fields, obtained by applying the laws of classical mechanics and thermodynamics to a Boussinesq, Navier-Stokes fluid are, written in terms of a generic vertical coordinate, r, so that the appropriate kinematic boundary conditions can be applied isomorphically see figure ref{fig:zandp-vert-coord}.

%%CNHbegin input{s_overview/text/vertcoord_figure.tex} %%CNHend

begin{equation} frac{Dvec{mathbf{v}_{h}}}{Dt}+left(2vec{Omega}times vec{mathbf{v}} right) _{h}+mathbf{nabla }_{h}phi =mathcal{F}_{vec{mathbf{v}_{h}}} text{ horizontal mtm} label{eq:horizontal_mtm} end{equation}

begin{equation} frac{Ddot{r}}{Dt}+widehat{k}cdot left(2vec{Omega}times vec{mathbf{ v}}right) +frac{partial phi }{partial r}+b=mathcal{F}_{dot{r}}text{ vertical mtm} label{eq:vertical_mtm} end{equation}

begin{equation} mathbf{nabla }_{h}cdot vec{mathbf{v}}_{h}+frac{partial dot{r}}{ partial r}=0text{ continuity} label{eq:continuity} end{equation}

begin{equation} b=b(theta ,S,r)text{ equation of state} label{eq:equation_of_state} end{equation}

begin{equation} frac{Dtheta }{Dt}=mathcal{Q}_{theta }text{ potential temperature} label{eq:potential_temperature} end{equation}

begin{equation} frac{DS}{Dt}=mathcal{Q}_{S}text{ humidity/salinity} label{eq:humidity_salt} end{equation}

Here:

begin{equation*} rtext{ is the vertical coordinate} end{equation*}

begin{equation*} frac{D}{Dt}=frac{partial }{partial t}+vec{mathbf{v}}cdot nabla text{ is the total derivative} end{equation*}

begin{equation*} mathbf{nabla }=mathbf{nabla }_{h}+widehat{k}frac{partial }{partial r} text{ is the ‘grad’ operator} end{equation*} with $mathbf{nabla }_{h}$ operating in the horizontal and $widehat{k} frac{partial }{partial r}$ operating in the vertical, where $widehat{k}$ is a unit vector in the vertical

begin{equation*} ttext{ is time} end{equation*}

begin{equation*} vec{mathbf{v}}=(u,v,dot{r})=(vec{mathbf{v}}_{h},dot{r})text{ is the velocity} end{equation*}

begin{equation*} phi text{ is the ‘pressure’/‘geopotential’} end{equation*}

begin{equation*} vec{Omega}text{ is the Earth’s rotation} end{equation*}

begin{equation*} btext{ is the ‘buoyancy’} end{equation*}

begin{equation*} theta text{ is potential temperature} end{equation*}

begin{equation*} Stext{ is specific humidity in the atmosphere; salinity in the ocean} end{equation*}

begin{equation*} mathcal{F}_{vec{mathbf{v}}}text{ are forcing and dissipation of }vec{ mathbf{v}} end{equation*}

begin{equation*} mathcal{Q}_{theta }mathcal{}text{are forcing and dissipation of }theta end{equation*}

begin{equation*} mathcal{Q}_{S}mathcal{}text{are forcing and dissipation of }S end{equation*}

The $mathcal{F}^{prime }s$ and $mathcal{Q}^{prime }s$ are provided by ‘physics’ and forcing packages for atmosphere and ocean. These are described in later chapters.

subsection{Kinematic Boundary conditions}

subsubsection{vertical}

at fixed and moving r surfaces we set (see figure ref{fig:zandp-vert-coord}):

begin{equation} dot{r}=0 text{at} r=R_{fixed}(x,y)text{ (ocean bottom, top of the atmosphere)} label{eq:fixedbc} end{equation}

begin{equation} dot{r}=frac{Dr}{Dt} text{at} r=R_{moving}text{ (ocean surface,bottom of the atmosphere)} label{eq:movingbc} end{equation}

Here

begin{equation*} R_{moving}=R_{o}+eta end{equation*} where $R_{o}(x,y)$ is the ‘$r-$value’ (height or pressure, depending on whether we are in the atmosphere or ocean) of the ‘moving surface’ in the resting fluid and $eta $ is the departure from $R_{o}(x,y)$ in the presence of motion.

subsubsection{horizontal}

begin{equation} vec{mathbf{v}}cdot vec{mathbf{n}}=0 label{eq:noflow} end{equation} where $vec{mathbf{n}}$ is the normal to a solid boundary.

subsection{Atmosphere}

In the atmosphere, (see figure ref{fig:zandp-vert-coord}), we interpret:

begin{equation} r=ptext{ is the pressure} label{eq:atmos-r} end{equation}

begin{equation} dot{r}=frac{Dp}{Dt}=omega text{ is the vertical velocity in }ptext{ coordinates} label{eq:atmos-omega} end{equation}

begin{equation} phi =g,ztext{ is the geopotential height} label{eq:atmos-phi} end{equation}

begin{equation} b=frac{partial Pi }{partial p}theta text{ is the buoyancy} label{eq:atmos-b} end{equation}

begin{equation} theta =T(frac{p_{c}}{p})^{kappa }text{ is potential temperature} label{eq:atmos-theta} end{equation}

begin{equation} S=q,text{ is the specific humidity} label{eq:atmos-s} end{equation} where

begin{equation*} Ttext{ is absolute temperature} end{equation*} begin{equation*} ptext{ is the pressure} end{equation*} begin{eqnarray*} &&ztext{ is the height of the pressure surface} \ &&gtext{ is the acceleration due to gravity} end{eqnarray*}

In the above the ideal gas law, $p=rho RT$, has been expressed in terms of the Exner function $Pi (p)$ given by (see Appendix Atmosphere) begin{equation} Pi (p)=c_{p}(frac{p}{p_{c}})^{kappa } label{eq:exner} end{equation} where p_{c} is a reference pressure and $kappa =R/c_{p}$ with R the gas constant and c_{p} the specific heat of air at constant pressure.

At the top of the atmosphere (which is 'fixed' in our r coordinate):

begin{equation*} R_{fixed}=p_{top}=0 end{equation*} In a resting atmosphere the elevation of the mountains at the bottom is given by begin{equation*} R_{moving}=R_{o}(x,y)=p_{o}(x,y) end{equation*} i.e. the (hydrostatic) pressure at the top of the mountains in a resting atmosphere.

The boundary conditions at top and bottom are given by:

begin{eqnarray} &&omega =0~text{at }r=R_{fixed} text{ (top of the atmosphere)} label{eq:fixed-bc-atmos} \ omega &=&frac{Dp_{s}}{Dt}text{; at }r=R_{moving}text{ (bottom of the atmosphere)} label{eq:moving-bc-atmos} end{eqnarray}

Then the (hydrostatic form of) equations (ref{eq:horizontal_mtm}-ref{eq:humidity_salt}) yields a consistent set of atmospheric equations which, for convenience, are written out in p coordinates in Appendix Atmosphere - see eqs(ref{eq:atmos-prime}).

subsection{Ocean}

In the ocean we interpret: begin{eqnarray} r &=&ztext{ is the height} label{eq:ocean-z} \ dot{r} &=&frac{Dz}{Dt}=wtext{ is the vertical velocity} label{eq:ocean-w} \ phi &=&frac{p}{rho _{c}}text{ is the pressure} label{eq:ocean-p} \ b(theta ,S,r) &=&frac{g}{rho _{c}}left(rho (theta ,S,r)-rho _{c}right) text{ is the buoyancy} label{eq:ocean-b} end{eqnarray} where $rho _{c}$ is a fixed reference density of water and g is the acceleration due to gravity.noindent

In the above

At the bottom of the ocean: $R_{fixed}(x,y)=-H(x,y)$.

The surface of the ocean is given by: $R_{moving}=eta $

The position of the resting free surface of the ocean is given by $ R_{o}=Z_{o}=0$.

Boundary conditions are:

begin{eqnarray} w &=&0~text{at }r=R_{fixed}text{ (ocean bottom)} label{eq:fixed-bc-ocean} \ w &=&frac{Deta }{Dt}text{ at }r=R_{moving}=eta text{ (ocean surface) label{eq:moving-bc-ocean}} end{eqnarray} where $eta $ is the elevation of the free surface.

Then equations (ref{eq:horizontal_mtm}-ref{eq:humidity_salt}) yield a consistent set of oceanic equations which, for convenience, are written out in z coordinates in Appendix Ocean - see eqs(ref{eq:ocean-mom}) to (ref{eq:ocean-salt}).

subsection{Hydrostatic, Quasi-hydrostatic, Quasi-nonhydrostatic and Non-hydrostatic forms} label{sec:all_hydrostatic_forms} begin{rawhtml} <!– CMIREDIR:non_hydrostatic: –> end{rawhtml}

Let us separate $phi $ in to surface, hydrostatic and non-hydrostatic terms:

begin{equation} phi (x,y,r)=phi _{s}(x,y)+phi _{hyd}(x,y,r)+phi _{nh}(x,y,r) label{eq:phi-split} end{equation} %and write eq(ref{eq:incompressible}) in the form: % ^- this eq is missing (jmc) ; replaced with: and write eq(ref{eq:horizontal_mtm}) in the form:

begin{equation} frac{partial vec{mathbf{v}_{h}}}{partial t}+mathbf{nabla }_{h}phi _{s}+mathbf{nabla }_{h}phi _{hyd}+epsilon _{nh}mathbf{nabla }_{h}phi _{nh}=vec{mathbf{G}}_{vec{v}_{h}} label{eq:mom-h} end{equation}

begin{equation} frac{partial phi _{hyd}}{partial r}=-b label{eq:hydrostatic} end{equation}

begin{equation} epsilon _{nh}frac{partial dot{r}}{partial t}+frac{partial phi _{nh}}{ partial r}=G_{dot{r}} label{eq:mom-w} end{equation} Here $epsilon _{nh}$ is a non-hydrostatic parameter.

The $left(vec{mathbf{G}}_{vec{v}},G_{dot{r}}right) $ in eq(ref {eq:mom-h}) and (ref{eq:mom-w}) represent advective, metric and Coriolis terms in the momentum equations. In spherical coordinates they take the form footnote{ In the hydrostatic primitive equations (textbf{HPE}) all underlined terms in (ref{eq:gu-spherical}), (ref{eq:gv-spherical}) and (ref {eq:gw-spherical}) are omitted; the singly-underlined terms are included in the quasi-hydrostatic model (textbf{QH}). The fully non-hydrostatic model (textbf{NH}) includes all terms.} - see Marshall et al 1997a for a full discussion:

begin{equation} left. begin{tabular}{l} $G_{u}=-vec{mathbf{v}}.nabla u$ \ $-left{ underline{frac{udot{r}}{{r}}}-frac{uvtan varphi}{{r}}right} $ \ $-left{ -2Omega vsin varphi+underline{2Omega dot{r}cos varphi}right} $ \ $+mathcal{F}_{u}$ end{tabular} right} left{ begin{tabular}{l} textit{advection} \ textit{metric} \ textit{Coriolis} \ textit{Forcing/Dissipation} end{tabular} right. qquad label{eq:gu-spherical} end{equation}

begin{equation} left. begin{tabular}{l} $G_{v}=-vec{mathbf{v}}.nabla v$ \ $-left{ underline{frac{vdot{r}}{{r}}}-frac{u^{2}tan varphi}{{r}}right} $ \ $-left{ -2Omega usin varphi right} $ \ $+mathcal{F}_{v}$ end{tabular} right} left{ begin{tabular}{l} textit{advection} \ textit{metric} \ textit{Coriolis} \ textit{Forcing/Dissipation} end{tabular} right. qquad label{eq:gv-spherical} end{equation} qquad qquad qquad qquad qquad

begin{equation} left. begin{tabular}{l} $G_{dot{r}}=-underline{underline{vec{mathbf{v}}.nabla dot{r}}}$ \ $+left{ underline{frac{u^{_{^{2}}}+v^{2}}{{r}}}right} $ \ ${+}underline{{2Omega ucos varphi}}$ \ $underline{underline{mathcal{F}_{dot{r}}}}$ end{tabular} right} left{ begin{tabular}{l} textit{advection} \ textit{metric} \ textit{Coriolis} \ textit{Forcing/Dissipation} end{tabular} right. label{eq:gw-spherical} end{equation} qquad qquad qquad qquad qquad

In the above ‘${r}$’ is the distance from the center of the earth and ‘$varphi$ ‘ is latitude.

Grad and div operators in spherical coordinates are defined in appendix OPERATORS.

%%CNHbegin input{s_overview/text/sphere_coord_figure.tex} %%CNHend

subsubsection{Shallow atmosphere approximation}

Most models are based on the ‘hydrostatic primitive equations’ (HPE’s) in which the vertical momentum equation is reduced to a statement of hydrostatic balance and the ‘traditional approximation’ is made in which the Coriolis force is treated approximately and the shallow atmosphere approximation is made. MITgcm need not make the ‘traditional approximation’. To be able to support consistent non-hydrostatic forms the shallow atmosphere approximation can be relaxed - when dividing through by $ r $ in, for example, (ref{eq:gu-spherical}), we do not replace r by a, the radius of the earth.

subsubsection{Hydrostatic and quasi-hydrostatic forms} label{sec:hydrostatic_and_quasi-hydrostatic_forms}

These are discussed at length in Marshall et al (1997a).

In the ‘hydrostatic primitive equations’ (textbf{HPE)} all the underlined terms in Eqs. (ref{eq:gu-spherical} $rightarrow $ref{eq:gw-spherical}) are neglected and ‘${r}$’ is replaced by ‘a’, the mean radius of the earth. Once the

pressure is found at one level - e.g. by inverting a 2-d Elliptic equation for $phi _{s}$ at $r=R_{moving}$ - the pressure can be computed at all other levels by integration of the hydrostatic relation, eq(ref{eq:hydrostatic}).

In the ‘quasi-hydrostatic’ equations (textbf{QH)} strict balance between gravity and vertical pressure gradients is not imposed. The $2Omega ucos varphi $ Coriolis term are not neglected and are balanced by a non-hydrostatic contribution to the pressure field: only the terms underlined twice in Eqs. (ref{eq:gu-speherical}$rightarrow $ref{eq:gw-spherical}) are set to zero and, simultaneously, the shallow atmosphere approximation is relaxed. In textbf{QH}textit{all} the metric terms are retained and the full variation of the radial position of a particle monitored. The textbf{QH}vertical momentum equation (ref{eq:mom-w}) becomes:

begin{equation*} frac{partial phi _{nh}}{partial r}=2Omega ucos varphi end{equation*} making a small correction to the hydrostatic pressure.

textbf{QH} has good energetic credentials - they are the same as for textbf{HPE}. Importantly, however, it has the same angular momentum principle as the full non-hydrostatic model (textbf{NH)} - see Marshall et.al., 1997a. As in textbf{HPE }only a 2-d elliptic problem need be solved.

subsubsection{Non-hydrostatic and quasi-nonhydrostatic forms}

MITgcm presently supports a full non-hydrostatic ocean isomorph, but only a quasi-non-hydrostatic atmospheric isomorph.

paragraph{Non-hydrostatic Ocean}

In the non-hydrostatic ocean model all terms in equations Eqs.(ref {eq:gu-speherical} $rightarrow $ref{eq:gw-spherical}) are retained. A three dimensional elliptic equation must be solved subject to Neumann boundary conditions (see below). It is important to note that use of the full textbf{NH} does not admit any new ‘fast’ waves in to the system - the incompressible condition eq(ref{eq:continuity}) has already filtered out acoustic modes. It does, however, ensure that the gravity waves are treated accurately with an exact dispersion relation. The textbf{NH} set has a complete angular momentum principle and consistent energetics - see White and Bromley, 1995; Marshall et.al.1997a.

paragraph{Quasi-nonhydrostatic Atmosphere}

In the non-hydrostatic version of our atmospheric model we approximate $dot{ r}$ in the vertical momentum eqs(ref{eq:mom-w}) and (ref{eq:gv-spherical}) (but only here) by:

begin{equation} dot{r}=frac{Dp}{Dt}=frac{1}{g}frac{Dphi }{Dt} label{eq:quasi-nh-w} end{equation} where p_{hy} is the hydrostatic pressure.

subsubsection{Summary of equation sets supported by model}

paragraph{Atmosphere}

Hydrostatic, and quasi-hydrostatic and quasi non-hydrostatic forms of the compressible non-Boussinesq equations in $p-$coordinates are supported.

subparagraph{Hydrostatic and quasi-hydrostatic}

The hydrostatic set is written out in $p-$coordinates in appendix Atmosphere - see eq(ref{eq:atmos-prime}).

subparagraph{Quasi-nonhydrostatic}

A quasi-nonhydrostatic form is also supported.

paragraph{Ocean}

subparagraph{Hydrostatic and quasi-hydrostatic}

Hydrostatic, and quasi-hydrostatic forms of the incompressible Boussinesq equations in $z-$coordinates are supported.

subparagraph{Non-hydrostatic}

Non-hydrostatic forms of the incompressible Boussinesq equations in $z-$ coordinates are supported - see eqs(ref{eq:ocean-mom}) to (ref {eq:ocean-salt}).

subsection{Solution strategy}

The method of solution employed in the textbf{HPE}, textbf{QH} and textbf{ NH} models is summarized in Figure ref{fig:solution-strategy}. Under all dynamics, a 2-d elliptic equation is first solved to find the surface pressure and the hydrostatic pressure at any level computed from the weight of fluid above. Under textbf{HPE} and textbf{QH} dynamics, the horizontal momentum equations are then stepped forward and $dot{r}$ found from continuity. Under textbf{NH} dynamics a 3-d elliptic equation must be solved for the non-hydrostatic pressure before stepping forward the horizontal momentum equations; $dot{r}$ is found by stepping forward the vertical momentum equation.

%%CNHbegin input{s_overview/text/solution_strategy_figure.tex} %%CNHend

There is no penalty in implementing textbf{QH} over textbf{HPE} except, of course, some complication that goes with the inclusion of $cos varphi $ Coriolis terms and the relaxation of the shallow atmosphere approximation. But this leads to negligible increase in computation. In textbf{NH}, in contrast, one additional elliptic equation - a three-dimensional one - must be inverted for p_{nh}. However the 'overhead' of the textbf{NH} model is essentially negligible in the hydrostatic limit (see detailed discussion in Marshall et al, 1997) resulting in a non-hydrostatic algorithm that, in the hydrostatic limit, is as computationally economic as the textbf{HPEs}.

subsection{Finding the pressure field} label{sec:finding_the_pressure_field}

Unlike the prognostic variables u, v, w, $theta $ and S, the pressure field must be obtained diagnostically. We proceed, as before, by dividing the total (pressure/geo) potential in to three parts, a surface part, $phi _{s}(x,y)$, a hydrostatic part $phi _{hyd}(x,y,r)$ and a non-hydrostatic part $phi _{nh}(x,y,r)$, as in (ref{eq:phi-split}), and writing the momentum equation as in (ref{eq:mom-h}).

subsubsection{Hydrostatic pressure}

Hydrostatic pressure is obtained by integrating (ref{eq:hydrostatic}) vertically from $r=R_{o}$ where $phi _{hyd}(r=R_{o})=0$, to yield:

begin{equation*} int_{r}^{R_{o}}frac{partial phi _{hyd}}{partial r}dr=left[phi _{hyd} right] _{r}^{R_{o}}=int_{r}^{R_{o}}-bdr end{equation*} and so

begin{equation} phi _{hyd}(x,y,r)=int_{r}^{R_{o}}bdr label{eq:hydro-phi} end{equation}

The model can be easily modified to accommodate a loading term (e.g atmospheric pressure pushing down on the ocean's surface) by setting:

begin{equation} phi _{hyd}(r=R_{o})=loading label{eq:loading} end{equation}

subsubsection{Surface pressure}

The surface pressure equation can be obtained by integrating continuity, (ref{eq:continuity}), vertically from $r=R_{fixed}$ to $r=R_{moving}$

begin{equation*} int_{R_{fixed}}^{R_{moving}}left(mathbf{nabla }_{h}cdot vec{mathbf{v} }_{h}+partial _{r}dot{r}right) dr=0 end{equation*}

Thus:

begin{equation*} frac{partial eta }{partial t}+vec{mathbf{v}}.nabla eta +int_{R_{fixed}}^{R_{moving}}mathbf{nabla }_{h}cdot vec{mathbf{v}} _{h}dr=0 end{equation*} where $eta =R_{moving}-R_{o}$ is the free-surface r-anomaly in units of $ r $. The above can be rearranged to yield, using Leibnitz's theorem:

begin{equation} frac{partial eta }{partial t}+mathbf{nabla }_{h}cdot int_{R_{fixed}}^{R_{moving}}vec{mathbf{v}}_{h}dr=text{sou label{eq:free-surface} end{equation} where we have incorporated a source term.

Whether $phi $ is pressure (ocean model, $p/rho _{c}$) or geopotential (atmospheric model), in (ref{eq:mom-h}), the horizontal gradient term can be written begin{equation} mathbf{nabla }_{h}phi _{s}=mathbf{nabla }_{h}left(b_{s}eta right) label{eq:phi-surf} end{equation} where b_{s} is the buoyancy at the surface.

In the hydrostatic limit ($epsilon _{nh}=0$), equations (ref{eq:mom-h}), (ref {eq:free-surface}) and (ref{eq:phi-surf}) can be solved by inverting a 2-d elliptic equation for $phi _{s}$ as described in Chapter 2. Both ‘free surface’ and ‘rigid lid’ approaches are available.

subsubsection{Non-hydrostatic pressure}

Taking the horizontal divergence of (ref{eq:mom-h}) and adding $frac{partial }{partial r}$ of (ref{eq:mom-w}), invoking the continuity equation (ref{eq:continuity}), we deduce that:

begin{equation} nabla _{3}^{2}phi _{nh}=nabla .vec{mathbf{G}}_{vec{v}}-left(mathbf{ nabla }_{h}^{2}phi _{s}+mathbf{nabla }^{2}phi _{hyd}right) =nabla . vec{mathbf{F}} label{eq:3d-invert} end{equation}

For a given rhs this 3-d elliptic equation must be inverted for $phi _{nh}$ subject to appropriate choice of boundary conditions. This method is usually called textit{The Pressure Method} [Harlow and Welch, 1965; Williams, 1969; Potter, 1976]. In the hydrostatic primitive equations case (textbf{HPE}), the 3-d problem does not need to be solved.

paragraph{Boundary Conditions}

We apply the condition of no normal flow through all solid boundaries - the coasts (in the ocean) and the bottom:

begin{equation} vec{mathbf{v}}.widehat{n}=0 label{nonormalflow} end{equation} where $widehat{n}$ is a vector of unit length normal to the boundary. The kinematic condition (ref{nonormalflow}) is also applied to the vertical velocity at $r=R_{moving}$. No-slip $left(v_{T}=0right) $or slip $ left(partial v_{T}/partial n=0right) $conditions are employed on the tangential component of velocity, v_{T}, at all solid boundaries, depending on the form chosen for the dissipative terms in the momentum equations - see below.

Eq.(ref{nonormalflow}) implies, making use of (ref{eq:mom-h}), that:

begin{equation} widehat{n}.nabla phi _{nh}=widehat{n}.vec{mathbf{F}} label{eq:inhom-neumann-nh} end{equation} where

begin{equation*} vec{mathbf{F}}=vec{mathbf{G}}_{vec{v}}-left(mathbf{nabla }_{h}phi _{s}+mathbf{nabla }phi _{hyd}right) end{equation*} presenting inhomogeneous Neumann boundary conditions to the Elliptic problem (ref{eq:3d-invert}). As shown, for example, by Williams (1969), one can exploit classical 3D potential theory and, by introducing an appropriately chosen $delta $-function sheet of ‘source-charge’, replace the inhomogeneous boundary condition on pressure by a homogeneous one. The source term rhs in (ref{eq:3d-invert}) is the divergence of the vector $ vec{mathbf{F}}.$ By simultaneously setting $ begin{array}{l} widehat{n}.vec{mathbf{F}} end{array} =0$and $widehat{n}.nabla phi _{nh}=0$on the boundary the following self-consistent but simpler homogenized Elliptic problem is obtained:

begin{equation*} nabla ^{2}phi _{nh}=nabla .widetilde{vec{mathbf{F}}}qquad end{equation*} where $widetilde{vec{mathbf{F}}}$ is a modified $vec{mathbf{F}}$ such that $widetilde{vec{mathbf{F}}}.widehat{n}=0$. As is implied by (ref {eq:inhom-neumann-nh}) the modified boundary condition becomes:

begin{equation} widehat{n}.nabla phi _{nh}=0 label{eq:hom-neumann-nh} end{equation}

If the flow is ‘close’ to hydrostatic balance then the 3-d inversion converges rapidly because $phi _{nh}$is then only a small correction to the hydrostatic pressure field (see the discussion in Marshall et al, a,b).

The solution $phi _{nh}$to (ref{eq:3d-invert}) and (ref{eq:inhom-neumann-nh}) does not vanish at $r=R_{moving}$, and so refines the pressure there.

subsection{Forcing/dissipation}

subsubsection{Forcing}

The forcing terms $mathcal{F}$ on the rhs of the equations are provided by ‘physics packages’ and forcing packages. These are described later on.

subsubsection{Dissipation}

paragraph{Momentum}

Many forms of momentum dissipation are available in the model. Laplacian and biharmonic frictions are commonly used:

begin{equation} D_{V}=A_{h}nabla _{h}^{2}v+A_{v}frac{partial ^{2}v}{partial z^{2}} +A_{4}nabla _{h}^{4}v label{eq:dissipation} end{equation} where A_{h} and $A_{v}$are (constant) horizontal and vertical viscosity coefficients and $A_{4}$is the horizontal coefficient for biharmonic friction. These coefficients are the same for all velocity components.

paragraph{Tracers}

The mixing terms for the temperature and salinity equations have a similar form to that of momentum except that the diffusion tensor can be non-diagonal and have varying coefficients. begin{equation} D_{T,S}=nabla .[underline{underline{K}}nabla (T,S)]+K_{4}nabla _{h}^{4}(T,S) label{eq:diffusion} end{equation} where $underline{underline{K}}$is the diffusion tensor and the K_{4} horizontal coefficient for biharmonic diffusion. In the simplest case where the subgrid-scale fluxes of heat and salt are parameterized with constant horizontal and vertical diffusion coefficients, $underline{underline{K}}$, reduces to a diagonal matrix with constant coefficients:

begin{equation} qquad qquad qquad qquad K=left(begin{array}{ccc} K_{h} & 0 & 0 \ 0 & K_{h} & 0 \ 0 & 0 & K_{v} end{array} right) qquad qquad qquad label{eq:diagonal-diffusion-tensor} end{equation} where $K_{h}$and $K_{v}$are the horizontal and vertical diffusion coefficients. These coefficients are the same for all tracers (temperature, salinity ...).

subsection{Vector invariant form}

For some purposes it is advantageous to write momentum advection in eq(ref {eq:horizontal_mtm}) and (ref{eq:vertical_mtm}) in the (so-called) ‘vector invariant’ form:

begin{equation} frac{Dvec{mathbf{v}}}{Dt}=frac{partial vec{mathbf{v}}}{partial t} +left(nabla times vec{mathbf{v}}right) times vec{mathbf{v}}+nabla left[frac{1}{2}(vec{mathbf{v}}cdot vec{mathbf{v}})right] label{eq:vi-identity} end{equation} This permits alternative numerical treatments of the non-linear terms based on their representation as a vorticity flux. Because gradients of coordinate vectors no longer appear on the rhs of (ref{eq:vi-identity}), explicit representation of the metric terms in (ref{eq:gu-speherical}), (ref {eq:gv-spherical}) and (ref{eq:gw-spherical}), can be avoided: information about the geometry is contained in the areas and lengths of the volumes used to discretize the model.

subsection{Adjoint}

Tangent linear and adjoint counterparts of the forward model are described in Chapter 5.

section{Appendix ATMOSPHERE}

subsection{Hydrostatic Primitive Equations for the Atmosphere in pressure coordinates}

label{sect-hpe-p}

The hydrostatic primitive equations (HPEs) in p-coordinates are: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+mathbf{nabla }_{p}phi &=&vec{mathbf{mathcal{F}}} label{eq:atmos-mom} \ frac{partial phi }{partial p}+alpha &=&0 label{eq-p-hydrostart} \ mathbf{nabla }_{p}cdot vec{mathbf{v}}_{h}+frac{partial omega }{ partial p} &=&0 label{eq:atmos-cont} \ palpha &=&RT label{eq:atmos-eos} \ c_{v}frac{DT}{Dt}+pfrac{Dalpha }{Dt} &=&mathcal{Q} label{eq:atmos-heat} end{eqnarray} where $vec{mathbf{v}}_{h}=(u,v,0)$ is the ‘horizontal’ (on pressure surfaces) component of velocity, $frac{D}{Dt}=frac{partial}{partial t} +vec{mathbf{v}}_{h}cdot mathbf{nabla }_{p}+omega frac{partial }{partial p}$ is the total derivative, $f=2Omega sin varphi$ is the Coriolis parameter, $phi =gz$ is the geopotential, $alpha =1/rho $ is the specific volume, $omega =frac{Dp }{Dt}$ is the vertical velocity in the $p-$coordinate. Equation(ref {eq:atmos-heat}) is the first law of thermodynamics where internal energy $e=c_{v}T$, T is temperature, Q is the rate of heating per unit mass and $pfrac{Dalpha }{Dt}$ is the work done by the fluid in compressing.

It is convenient to cast the heat equation in terms of potential temperature $theta $ so that it looks more like a generic conservation law. Differentiating (ref{eq:atmos-eos}) we get: begin{equation*} pfrac{Dalpha }{Dt}+alpha frac{Dp}{Dt}=Rfrac{DT}{Dt} end{equation*} which, when added to the heat equation (ref{eq:atmos-heat}) and

using $ c_{p}=c_{v}+R$, gives: begin{equation} c_{p}frac{DT}{Dt}-alpha frac{Dp}{Dt}=mathcal{Q} label{eq-p-heat-interim} end{equation} Potential temperature is defined: begin{equation} theta =T(frac{p_{c}}{p})^{kappa } label{eq:potential-temp} end{equation} where p_{c} is a reference pressure and $kappa =R/c_{p}$. For convenience we will make use of the Exner function $Pi (p)$ which defined by: begin{equation} Pi (p)=c_{p}(frac{p}{p_{c}})^{kappa } label{Exner} end{equation} The following relations will be useful and are easily expressed in terms of the Exner function: begin{equation*} c_{p}T=Pi theta ;;;;;frac{partial Pi }{partial p}=frac{kappa Pi }{p};;;;;alpha =frac{kappa Pi theta }{p}=frac{partial Pi }{ partial p}theta ;;;;;frac{DPi }{Dt}=frac{partial Pi }{partial p} frac{Dp}{Dt} end{equation*} where $b=frac{partial Pi }{partial p}theta $ is the buoyancy.

The heat equation is obtained by noting that begin{equation*} c_{p}frac{DT}{Dt}=frac{D(Pi theta)}{Dt}=Pi frac{Dtheta }{Dt}+theta frac{DPi }{Dt}=Pi frac{Dtheta }{Dt}+alpha frac{Dp}{Dt} end{equation*} and on substituting into (ref{eq-p-heat-interim}) gives: begin{equation} Pi frac{Dtheta }{Dt}=mathcal{Q} label{eq:potential-temperature-equation} end{equation} which is in conservative form.

For convenience in the model we prefer to step forward (ref {eq:potential-temperature-equation}) rather than (ref{eq:atmos-heat}).

subsubsection{Boundary conditions}

The upper and lower boundary conditions are : begin{eqnarray} mbox{at the top:};;p=0 &&text{, }omega =frac{Dp}{Dt}=0 \ mbox{at the surface:};;p=p_{s} &&text{, }phi =phi _{topo}=g~Z_{topo} label{eq:boundary-condition-atmosphere} end{eqnarray} In p-coordinates, the upper boundary acts like a solid boundary ($omega =0 $); in z-coordinates and the lower boundary is analogous to a free surface ($phi $ is imposed and $omega neq 0$).

subsubsection{Splitting the geo-potential} label{sec:hpe-p-geo-potential-split}

For the purposes of initialization and reducing round-off errors, the model deals with perturbations from reference (or ‘‘standard’’) profiles. For example, the hydrostatic geopotential associated with the resting atmosphere is not dynamically relevant and can therefore be subtracted from the equations. The equations written in terms of perturbations are obtained by substituting the following definitions into the previous model equations: begin{eqnarray} theta &=&theta _{o}+theta ^{prime } label{eq:atmos-ref-prof-theta} \ alpha &=&alpha _{o}+alpha ^{prime } label{eq:atmos-ref-prof-alpha} \ phi &=&phi _{o}+phi ^{prime } label{eq:atmos-ref-prof-phi} end{eqnarray} The reference state (indicated by subscript ‘‘0’’) corresponds to horizontally homogeneous atmosphere at rest ($theta _{o},alpha _{o},phi _{o}$) with surface pressure $p_{o}(x,y)$ that satisfies $phi _{o}(p_{o})=g~Z_{topo}$, defined: begin{eqnarray*} theta _{o}(p) &=&f^{n}(p) \ alpha _{o}(p) &=&Pi _{p}theta _{o} \ phi _{o}(p) &=&phi _{topo}-int_{p_{0}}^{p}alpha _{o}dp end{eqnarray*} %begin{eqnarray*} %phi'_alpha & = & int^p_{p_o} (alpha_o -alpha) dp \ %phi'_s(x,y,t) & = & int_{p_o}^{p_s} alpha dp %end{eqnarray*}

The final form of the HPE's in p coordinates is then: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+mathbf{nabla }_{p}phi ^{prime } &=&vec{mathbf{mathcal{F}}} label{eq:atmos-prime} \ frac{partial phi ^{prime }}{partial p}+alpha ^{prime } &=&0 \ mathbf{nabla }_{p}cdot vec{mathbf{v}}_{h}+frac{partial omega }{ partial p} &=&0 \ frac{partial Pi }{partial p}theta ^{prime } &=&alpha ^{prime } \ frac{Dtheta }{Dt} &=&frac{mathcal{Q}}{Pi } end{eqnarray}

section{Appendix OCEAN}

subsection{Equations of motion for the ocean}

We review here the method by which the standard (Boussinesq, incompressible) HPE's for the ocean written in z-coordinates are obtained. The non-Boussinesq equations for oceanic motion are: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho }mathbf{nabla }_{z}p &=&vec{mathbf{mathcal{F}}} \ epsilon _{nh}frac{Dw}{Dt}+g+frac{1}{rho }frac{partial p}{partial z} &=&epsilon _{nh}mathcal{F}_{w} \ frac{1}{rho }frac{Drho }{Dt}+mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{partial w}{partial z} &=&0 label{eq-zns-cont}\ rho &=&rho (theta ,S,p) label{eq-zns-eos}\ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq-zns-heat}\ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq-zns-salt} label{eq:non-boussinesq} end{eqnarray} These equations permit acoustics modes, inertia-gravity waves, non-hydrostatic motions, a geostrophic (Rossby) mode and a thermohaline mode. As written, they cannot be integrated forward consistently - if we step $rho $ forward in (ref{eq-zns-cont}), the answer will not be consistent with that

obtained by stepping (ref{eq-zns-heat}) and (ref {eq-zns-salt}) and then using (ref{eq-zns-eos}) to yield $rho $. It is therefore necessary to manipulate the system as follows. Differentiating the EOS (equation of state) gives:

begin{equation} frac{Drho }{Dt}=left. frac{partial rho }{partial theta }right| _{S,p}frac{Dtheta }{Dt}+left. frac{partial rho }{partial S}right| _{theta ,p}frac{DS}{Dt}+left. frac{partial rho }{partial p}right| _{theta ,S}frac{Dp}{Dt} label{EOSexpansion} end{equation}

Note that $frac{partial rho }{partial p}=frac{1}{c_{s}^{2}}$ is the reciprocal of the sound speed (c_{s}) squared. Substituting into ref{eq-zns-cont} gives: begin{equation} frac{1}{rho c_{s}^{2}}frac{Dp}{Dt}+mathbf{nabla }_{z}cdot vec{mathbf{ v}}+partial _{z}wapprox 0 label{eq-zns-pressure} end{equation} where we have used an approximation sign to indicate that we have assumed adiabatic motion, dropping the $frac{Dtheta }{Dt}$ and $frac{DS}{Dt}$. Replacing ref{eq-zns-cont} with ref{eq-zns-pressure} yields a system that can be explicitly integrated forward: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho }mathbf{nabla }_{z}p &=&vec{mathbf{mathcal{F}}} label{eq-cns-hmom} \ epsilon _{nh}frac{Dw}{Dt}+g+frac{1}{rho }frac{partial p}{partial z} &=&epsilon _{nh}mathcal{F}_{w} label{eq-cns-hydro} \ frac{1}{rho c_{s}^{2}}frac{Dp}{Dt}+mathbf{nabla }_{z}cdot vec{mathbf{ v}}_{h}+frac{partial w}{partial z} &=&0 label{eq-cns-cont} \ rho &=&rho (theta ,S,p) label{eq-cns-eos} \ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq-cns-heat} \ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq-cns-salt} end{eqnarray}

subsubsection{Compressible z-coordinate equations}

Here we linearize the acoustic modes by replacing $rho $ with $rho _{o}(z)$ wherever it appears in a product (ie. non-linear term) - this is the 'Boussinesq assumption'. The only term that then retains the full variation in $rho $ is the gravitational acceleration: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho _{o}}mathbf{nabla }_{z}p &=&vec{mathbf{mathcal{F}}} label{eq-zcb-hmom} \ epsilon _{nh}frac{Dw}{Dt}+frac{grho }{rho _{o}}+frac{1}{rho _{o}} frac{partial p}{partial z} &=&epsilon _{nh}mathcal{F}_{w} label{eq-zcb-hydro} \ frac{1}{rho _{o}c_{s}^{2}}frac{Dp}{Dt}+mathbf{nabla }_{z}cdot vec{ mathbf{v}}_{h}+frac{partial w}{partial z} &=&0 label{eq-zcb-cont} \ rho &=&rho (theta ,S,p) label{eq-zcb-eos} \ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq-zcb-heat} \ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq-zcb-salt} end{eqnarray} These equations still retain acoustic modes. But, because the ''compressible" terms are linearized, the pressure equation ref {eq-zcb-cont} can be integrated implicitly with ease (the time-dependent term appears as a Helmholtz term in the non-hydrostatic pressure equation). These are the emph{truly} compressible Boussinesq equations. Note that the EOS must have the same pressure dependency as the linearized pressure term, ie. $left. frac{partial rho }{partial p}right| _{theta ,S}=frac{1}{ c_{s}^{2}}$, for consistency.

subsubsection{'Anelastic' z-coordinate equations}

The anelastic approximation filters the acoustic mode by removing the time-dependency in the continuity (now pressure-) equation (ref{eq-zcb-cont}). This could be done simply by noting that $frac{Dp}{Dt}approx -grho _{o} frac{Dz}{Dt}=-grho _{o}w$, but this leads to an inconsistency between continuity and EOS. A better solution is to change the dependency on pressure in the EOS by splitting the pressure into a reference function of height and a perturbation: begin{equation*} rho =rho (theta ,S,p_{o}(z)+epsilon _{s}p^{prime }) end{equation*} Remembering that the term $frac{Dp}{Dt}$ in continuity comes from differentiating the EOS, the continuity equation then becomes: begin{equation*} frac{1}{rho _{o}c_{s}^{2}}left(frac{Dp_{o}}{Dt}+epsilon _{s}frac{ Dp^{prime }}{Dt}right) +mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+ frac{partial w}{partial z}=0 end{equation*} If the time- and space-scales of the motions of interest are longer than those of acoustic modes, then $frac{Dp^{prime }}{Dt}<<(frac{Dp_{o}}{Dt}, mathbf{nabla }cdot vec{mathbf{v}}_{h})$ in the continuity equations and $left. frac{partial rho }{partial p}right| _{theta ,S}frac{ Dp^{prime }}{Dt}<<left. frac{partial rho }{partial p}right| _{theta ,S}frac{Dp_{o}}{Dt}$ in the EOS (ref{EOSexpansion}). Thus we set $epsilon _{s}=0$, removing the dependency on $p^{prime }$ in the continuity equation and EOS. Expanding $frac{Dp_{o}(z)}{Dt}=-grho _{o}w$ then leads to the anelastic continuity equation: begin{equation} mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{partial w}{partial z}- frac{g}{c_{s}^{2}}w=0 label{eq-za-cont1} end{equation} A slightly different route leads to the quasi-Boussinesq continuity equation where we use the scaling $frac{partial rho ^{prime }}{partial t}+ mathbf{nabla }_{3}cdot rho ^{prime }vec{mathbf{v}}<<mathbf{nabla } _{3}cdot rho _{o}vec{mathbf{v}}$ yielding: begin{equation} mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{1}{rho _{o}}frac{ partial left(rho _{o}wright) }{partial z}=0 label{eq-za-cont2} end{equation} Equations ref{eq-za-cont1} and ref{eq-za-cont2} are in fact the same equation if: begin{equation} frac{1}{rho _{o}}frac{partial rho _{o}}{partial z}=frac{-g}{c_{s}^{2}}

end{equation} Again, note that if $rho _{o}$ is evaluated from prescribed $theta _{o}$ and S_{o} profiles, then the EOS dependency on p_{o} and the term $frac{ g}{c_{s}^{2}}$ in continuity should be referred to those same profiles. The full set of ‘quasi-Boussinesq’ or ‘anelastic’ equations for the ocean are then: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho _{o}}mathbf{nabla }_{z}p &=&vec{mathbf{mathcal{F}}} label{eq-zab-hmom} \ epsilon _{nh}frac{Dw}{Dt}+frac{grho }{rho _{o}}+frac{1}{rho _{o}} frac{partial p}{partial z} &=&epsilon _{nh}mathcal{F}_{w} label{eq-zab-hydro} \ mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{1}{rho _{o}}frac{ partial left(rho _{o}wright) }{partial z} &=&0 label{eq-zab-cont} \ rho &=&rho (theta ,S,p_{o}(z)) label{eq-zab-eos} \ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq-zab-heat} \ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq-zab-salt} end{eqnarray}

subsubsection{Incompressible z-coordinate equations}

Here, the objective is to drop the depth dependence of $rho _{o}$ and so, technically, to also remove the dependence of $rho $ on p_{o}. This would yield the ‘‘truly” incompressible Boussinesq equations: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho _{c}}mathbf{nabla }_{z}p &=&vec{mathbf{mathcal{F}}} label{eq-ztb-hmom} \ epsilon _{nh}frac{Dw}{Dt}+frac{grho }{rho _{c}}+frac{1}{rho _{c}} frac{partial p}{partial z} &=&epsilon _{nh}mathcal{F}_{w} label{eq-ztb-hydro} \ mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{partial w}{partial z} &=&0 label{eq-ztb-cont} \ rho &=&rho (theta ,S) label{eq-ztb-eos} \ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq-ztb-heat} \ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq-ztb-salt} end{eqnarray} where $rho _{c}$ is a constant reference density of water.

subsubsection{Compressible non-divergent equations}

The above ‘‘incompressible” equations are incompressible in both the flow and the density. In many oceanic applications, however, it is important to retain compressibility effects in the density. To do this we must split the density thus: begin{equation*} rho =rho _{o}+rho ^{prime } end{equation*} We then assert that variations with depth of $rho _{o}$ are unimportant while the compressible effects in $rho ^{prime }$ are: begin{equation*} rho _{o}=rho _{c} end{equation*} begin{equation*} rho ^{prime }=rho (theta ,S,p_{o}(z))-rho _{o} end{equation*} This then yields what we can call the semi-compressible Boussinesq equations: begin{eqnarray} frac{Dvec{mathbf{v}}_{h}}{Dt}+fhat{mathbf{k}}times vec{mathbf{v}} _{h}+frac{1}{rho _{c}}mathbf{nabla }_{z}p^{prime } &=&vec{mathbf{ mathcal{F}}} label{eq:ocean-mom} \ epsilon _{nh}frac{Dw}{Dt}+frac{grho ^{prime }}{rho _{c}}+frac{1}{rho _{c}}frac{partial p^{prime }}{partial z} &=&epsilon _{nh}mathcal{F}_{w} label{eq:ocean-wmom} \ mathbf{nabla }_{z}cdot vec{mathbf{v}}_{h}+frac{partial w}{partial z} &=&0 label{eq:ocean-cont} \ rho ^{prime } &=&rho (theta ,S,p_{o}(z))-rho _{c} label{eq:ocean-eos} \ frac{Dtheta }{Dt} &=&mathcal{Q}_{theta } label{eq:ocean-theta} \ frac{DS}{Dt} &=&mathcal{Q}_{s} label{eq:ocean-salt} end{eqnarray} Note that the hydrostatic pressure of the resting fluid, including that associated with $rho _{c}$, is subtracted out since it has no effect on the dynamics.

Though necessary, the assumptions that go into these equations are messy since we essentially assume a different EOS for the reference density and the perturbation density. Nevertheless, it is the hydrostatic ($epsilon _{nh}=0$ form of these equations that are used throughout the ocean modeling community and referred to as the primitive equations (HPE).

section{Appendix:OPERATORS}

subsection{Coordinate systems}

subsubsection{Spherical coordinates}

In spherical coordinates, the velocity components in the zonal, meridional and vertical direction respectively, are given by (see Fig.2) :

begin{equation*} u=rcos varphi frac{Dlambda }{Dt} end{equation*}

begin{equation*} v=rfrac{Dvarphi }{Dt} end{equation*}

begin{equation*} dot{r}=frac{Dr}{Dt} end{equation*}

Here $varphi $ is the latitude, $lambda $ the longitude, r the radial distance of the particle from the center of the earth, $Omega $ is the angular speed of rotation of the Earth and D/Dt is the total derivative.

The ‘grad’ ($nabla $) and ‘div’ ($nablacdot$) operators are defined by, in spherical coordinates:

begin{equation*} nabla equiv left(frac{1}{rcos varphi }frac{partial }{partial lambda } ,frac{1}{r}frac{partial }{partial varphi },frac{partial }{partial r} right) end{equation*}

begin{equation*} nablacdot vequiv frac{1}{rcos varphi }left{ frac{partial u}{partial lambda }+frac{partial }{partial varphi }left(vcos varphi right) right} +frac{1}{r^{2}}frac{partial left(r^{2}dot{r}right) }{partial r} end{equation*}

CHAPTER

TWO

PHYSICAL PARAMETERIZATIONS - PACKAGES I

In this chapter and in the following chapter, the MITgcm 'packages' are described. While you can carry out many experiments with MITgcm by starting from case studies in section ref{sec:modelExamples}, configuring a brand new experiment or making major changes to an experimental configuration requires some knowledge of the *packages* that make up the full MITgcm code. Packages are used in MITgcm to help organize and layer various code building blocks that are assembled and selected to perform a specific experiment. Each of the specific experiments described in section ref{sec:modelExamples} uses a particular combination of packages.

Figure 2.1 shows the full set of packages that are available. As shown in the figure packages are classified into different groupings that layer on top of each other. The top layer packages are generally specialized to specific simulation types. In this layer there are packages that deal with biogeochemical processes, ocean interior and boundary layer processes, atmospheric processes, sea-ice, coupled simulations and state estimation. Below this layer are a set of general purpose numerical and computational packages. The general purpose numerical packages provide code for kernel numerical alogorithms that apply to many different simulation types. Similarly, the general purpose computational packages implement non-numerical alogorithms that provide parallelism, I/O and time-keeping functions that are used in many different scenarios.

The following sections describe the packages shown in figure ref{fig:package_organigramme}. Section ref{sec:pkg:using} describes the general procedure for using any package in MITgcm. Following that sections ref{sec:pkg:gad}-ref{sec:pkg:monitor} layout the algorithms implemented in specific packages and describe how to use the individual packages. A brief synopsis of the function of each package is given in table ref{tab:package_summary_tab}. Organizationally package code is assigned a separate subdirectory in the MITgcm code distribution (within the source code directory texttt{pkg}). The name of this subdirectory is used as the package name in table ref{tab:package_summary_tab}.

%% In this chapter the schemes for parameterizing processes that are not %% represented explicitly in MITgcm are described. Some of these %% processes are sub-grid scale (SGS) phenomena, other processes, such as %% open-boundaries, are external to the simulation.

begin{table} caption{~} label{tab:package_summary_tab}. end{table}

% Overview newpage input{s_phys_pkgs/text/packages.tex}

2.1 Packages Related to Hydrodynamical Kernel

input{s_phys_pkgs/text/generic_advdiff.tex}

newpage input{s_phys_pkgs/text/shap_filt.tex}

newpage input{s_phys_pkgs/text/zonal_filt.tex}

newpage input{s_phys_pkgs/text/exch2.tex}

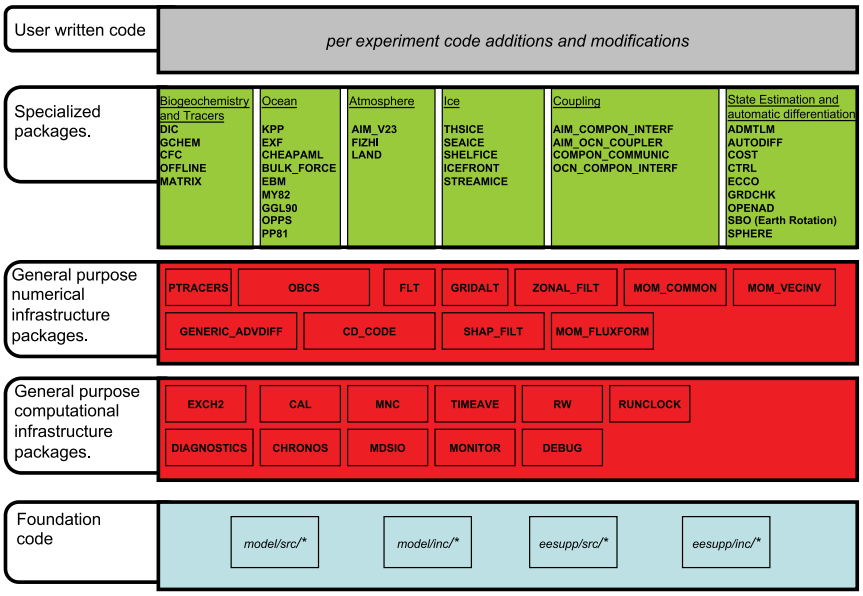

Figure 2.1: Hierarchy of code layers that are assembled to make up an MITgcm simulation. Conceptually (and in terms of code organization) MITgcm consists of several layers. At the base is a layer of core software that provides a basic numerical and computational foundation for MITgcm simulations. This layer is shown marked *Foundation Code* at the bottom of the figure and corresponds to code in the italicised subdirectories on the figure. This layer is not organized into packages. All code above the foundation layer is organized as packages. Much of the code in MITgcm is contained in packages which serve as a useful way of organizing and layering the different levels of functionality that make up the full MITgcm software distribution. The figure shows the different packages in MITgcm as boxes containing bold face upper case names. Directly above the foundation layer are two layers of general purpose infrastructure software that consist of computational and numerical packages. These general purpose packages can be applied to both online and offline simulations and are used in many different physical simulation types. Above these layers are more specialized packages.

newpage input{s_phys_pkgs/text/gridalt.tex}

% Some Mention of Packages that are part of the main model document newpage

2.2 General purpose numerical infrastructure packages

2.2.1 OBCS: Open boundary conditions for regional modeling

Authors: Alistair Adcroft, Patrick Heimbach, Samar Katiwala, Martin Losch

2.2.1.1 Introduction

The OBCS-package is fundamental to regional ocean modelling with the MITgcm, but there are so many details to be considered in regional ocean modelling that this package cannot accomodate all imaginable and possible options. Therefore, for a regional simulation with very particular details, it is recommended to familiarize oneself not only with the compile- and runtime-options of this package, but also with the code itself. In many cases it will be necessary to adapt the obcs-code (in particular code{S/R OBCS_CALC}) to the application in question; in these cases the obcs-package (together with the rbcs-package, section ref{sec:pkg:rbcs}) is a very useful infrastructure for implementing special regional models.

2.2.1.2 OBCS configuration and compiling

As with all MITgcm packages, OBCS can be turned on or off at compile time

- using the `packages.conf` file by adding `obcs` to it,
- or using `genmake2` adding `-enable=obcs` or `-disable=obcs` switches
- *Required packages and CPP options:*
  - Two alternatives are available for prescribing open boundary values, which differ in the way how OB's are treated in time:
    - A simple time-management (e.g. constant in time, or cyclic with fixed fequency) is provided through S/R `obcs_external_fields_load`.
    - More sophisticated 'real-time' (i.e. calendar time) management is available through `obcs_prescribe_read`.
  - The latter case requires packages `cal` and `exf` to be enabled.

(see also Section ref{sec:buildingCode}).

Parts of the OBCS code can be enabled or disabled at compile time via CPP preprocessor flags. These options are set in *OBCS_OPTIONS.h*. Table 2.1 summarizes these options.

Table 2.1: OBCS CPP options

CPP option	Description
ALLOW_OBCS_NORTH	enable Northern OB
ALLOW_OBCS_SOUTH	enable Southern OB
ALLOW_OBCS_EAST	enable Eastern OB
ALLOW_OBCS_WEST	enable Western OB
ALLOW_OBCS_PRESCRIBE	enable code for prescribing OB's
ALLOW_OBCS_SPONGE	enable sponge layer code
ALLOW_OBCS_BALANCE	enable code for balancing transports through OB's
ALLOW_ORLANSKI	enable Orlanski radiation conditions at OB's
ALLOW_OBCS_STEVENS	enable Stevens (1990) boundary conditions at OB's
	(currently only implemented for eastern and
	western boundaries and NOT for ptracers)

2.2.1.3 Run-time parameters

Run-time parameters are set in files `data.pkg`, `data.obcs`, and `data.exf if 'real-time' prescription is requested (i.e. package :code:`exf` enabled). These parameter files are read in S/R `packages_readparms.F`, `obcs_readparms.F`, and `exf_readparms.F`, respectively. Run-time parameters may be broken into 3 categories:

1. switching on/off the package at runtime,
2. OBCS package flags and parameters,
3. additional timing flags in `data.exf`, if selected.

Enabling the package

The OBCS package is switched on at runtime by setting `useOBCS = .TRUE.` in `data.pkg`.

Package flags and parameters

Table 2.2 summarizes the runtime flags that are set in `data.obcs`, and their default values.

Table 2.2: OBCS CPP options

Flag/parameter	default	Description
basic flags & parameters (OBCS_PARM01)		
OB_Jnorth	0	Nx-vector of J-indic
OB_Jsouth	0	Nx-vector of J-indic
OB_Ieast	0	Ny-vector of I-indic
OB_Iwest	0	Ny-vector of I-indic
useOBCSprescribe	`.FALSE.`	
useOBCSsponge	`.FALSE.`	
useOBCSbalance & code{.FALSE.} &		
~ \		
OBCS_balanceFacN/S/E/W & 1 & factor(s) determining the details		

Table 2.2 – continued from previous page

of the balaning code \		
useOrlanskiNorth/South/EastWest & code{.FALSE.} &		
turn on Orlanski boundary conditions for individual boundary\		
useStevensNorth/South/EastWest & code{.FALSE.} &		
turn on Stevens boundary conditions for individual boundary\		
OBtextbf{X}textbf{y}File & ~ &		
file name of OB field \		
~ & ~ &		
textbf{X}: textbf{N}(orth)	textbf{S}(outh)	
textbf{E}(ast)	textbf{W}(est) \	
~ & ~ &		
textbf{y}: textbf{t}(emperature)	textbf{s}(salinity)	
textbf{u}(-velocity)	textbf{v}(-velocity)	\
~ & ~ &		
textbf{w}(-velocity)	textbf{eta}(sea surface height)\	
~ & ~ &		
textbf{a}(sea ice area)	textbf{h}(sea ice thickness)	
textbf{sn}(snow thickness)	textbf{sl}(sea ice salinity)\	
hline		
multicolumn{3}{|c|}{textit{Orlanski parameters} (OBCS_PARM02) } \		
hline		
cvelTimeScale & 2000 sec &		
averaging period for phase speed \		
CMAX & 0.45 m/s &		
maximum allowable phase speed-CFL for AB-II \		
CFIX & 0.8 m/s &		
fixed boundary phase speed \		
useFixedCEast & code{.FALSE.} &		
~ \		
useFixedCWest & code{.FALSE.} &		
~ \		
hline		
multicolumn{3}{|c|}{textit{Sponge-layer parameters} (OBCS_PARM03)} \		
hline		
spongeThickness & 0 &		
sponge layer thickness (in # grid points) \		
Urelaxobcsinner & 0 sec &		
relaxation time scale at the		
innermost sponge layer point of a meridional OB \		
Vrelaxobcsinner & 0 sec &		
relaxation time scale at the		
innermost sponge layer point of a zonal OB \		
Urelaxobcsbound & 0 sec &		
relaxation time scale at the		
outermost sponge layer point of a meridional OB \		
Vrelaxobcsbound & 0 sec &		
relaxation time scale at the		
outermost sponge layer point of a zonal OB \		
hline		
multicolumn{3}{|c|}{textit{Stevens parameters} (OBCS_PARM04) } \		

Table 2.2 – continued from previous page

hline		
T/SrelaxStevens & 0~sec & relaxation time scale for		
temperature/salinity \		
useStevensPhaseVel & code{.TRUE.} & \		
useStevensAdvection & code{.TRUE.} & \		
hline		
hline		
end{tabular}		
}		
caption{pkg OBCS run-time parameters}		
label{tab:pkg:obcs:runtime_flags}		

end{table}

%———————————————————————————-

subsubsection{Defining open boundary positions label{sec:pkg:obcs:defining}}

There are four open boundaries (OBs), a Northern, Southern, Eastern, and Western. All OB locations are specified by their absolute meridional (Northern/Southern) or zonal (Eastern/Western) indices. Thus, for each zonal position $i=1,ldots,N_x$ a meridional index j specifies the Northern/Southern OB position, and for each meridional position $j=1,ldots,N_y$, a zonal index i specifies the Eastern/Western OB position. For Northern/Southern OB this defines an N_x-dimensional ‘‘row” array $tt OB_Jnorth(Nx)$ / $tt OB_Jsouth(Nx)$, and an N_y-dimenisonal ‘‘column” array $tt OB_Ieast(Ny)$ / $tt OB_Iwest(Ny)$. Positions determined in this way allows Northern/Southern OBs to be at variable j (or y) positions, and Eastern/Western OBs at variable i (or x) positions. Here, indices refer to tracer points on the C-grid. A zero (0) element in $tt OB_Ildots$, $tt OB_Jldots$ means there is no corresponding OB in that column/row. For a Northern/Southern OB, the OB V point is to the South/North. For an Eastern/Western OB, the OB U point is to the West/East. For example, begin{tabbing}

code{OB_Jnorth(3)=34} = means that: = \ > code{T(3,34)} > is a an OB point \ > code{U(3,34)} > is a an OB point \ > code{V(3,34)} > is a an OB point \ code{OB_Jsouth(3)=1} > means that: \ > code{T(3,1)} > is a an OB point \ > code{U(3,1)} > is a an OB point \ > code{V(3,2)} > is a an OB point \ code{OB_Ieast(10)=69} > means that: > \ > code{T(69,10)} > is a an OB point \ > code{U(69,10)} > is a an OB point \ > code{V(69,10)} > is a an OB point \ code{OB_Iwest(10)=1} > means that: > \ > code{T(1,10)} > is a an OB point \ > code{U(2,10)} > is a an OB point \ > code{V(1,10)} > is a an OB point

end{tabbing} For convenience, negative values for code{Jnorth}/code{Ieast} refer to points relative to the Northern/Eastern edges of the model eg. $tt OB_Jnorth(3)=-1$ means that the point $tt (3,Ny)$ is a northern OB.

noindenttextbf{Simple examples:} For a model grid with $ N_{x}times N_{y} = 120times144$ horizontal grid points with four open boundaries along the four egdes of the domain, the simplest way of specifying the boundary points in code{data.obcs} is: begin{verbatim}

OB_Ieast = 144*-1,

 # or OB_Ieast = 144*120, OB_Iwest = 144*1, OB_Jnorth = 120*-1,

 # or OB_Jnorth = 120*144, OB_Jsouth = 120*1,

end{verbatim} If only the first 50 grid points of the southern boundary are boundary points: begin{verbatim}

OB_Jsouth(1:50) = 50*1,

end{verbatim}

noindent textsf{Add special comments for case #define NONLIN_FRSURF, see obcs_ini_fixed.F}

%———————————————————————————-

2.2.2 Equations and key routines

paragraph{OBCS_READPARMS:} ~ \ Set OB positions through arrays {tt OB_Jnorth(Nx), OB_Jsouth(Nx), OB_Ieast(Ny), OB_Iwest(Ny)}, and runtime flags (see Table ref{tab:pkg:obcs:runtime_flags}).

paragraph{OBCS_CALC:} ~ \ % Top-level routine for filling values to be applied at OB for T,S,U,V,eta into corresponding ‘‘slice” arrays (x,z), (y,z) for each OB: $tt OB[N/S/E/W][t/s/u/v]$; e.g. for salinity array at Southern OB, array name is $tt OBSt$. Values filled are either % begin{itemize} % item constant vertical T,S profiles as specified in file {tt data} ({tt tRef(Nr), sRef(Nr)}) with zero velocities U,V, % item T,S,U,V values determined via Orlanski radiation conditions (see below), % item prescribed time-constant or time-varying fields (see below). % item use prescribed boundary fields to compute Stevens boundary conditions. end{itemize}

paragraph{ORLANSKI:} ~ \ % Orlanski radiation conditions citep{orl:76}, examples can be found in code{verification/dome} and code{verification/tutorial_plume_on_slope} (ref{sec:eg-gravityplume}).

paragraph{OBCS_PRESCRIBE_READ:} ~ \ % When code{useOBCSprescribe = .TRUE.} the model tries to read temperature, salinity, u- and v-velocities from files specified in the runtime parameters code{OB[N/S/E/W][t/s/u/v]File}. These files are the usual IEEE, big-endian files with dimensions of a section along an open boundary: begin{itemize} item For North/South boundary files the dimensions are

$(N_xtimes N_rtimesmbox{time levels})$, for East/West boundary files the dimensions are $(N_ytimes N_rtimesmbox{time levels})$.

item If a non-linear free surface is used (ref{sec:nonlinear-freesurface}), additional files code{OB[N/S/E/W]etaFile} for the sea surface height eta with dimension $(N_{x/y}timesmbox{time levels})$ may be specified.

item If non-hydrostatic dynamics are used (ref{sec:non-hydrostatic}), additional files code{OB[N/S/E/W]wFile} for the vertical velocity w with dimensions $(N_{x/y}times N_rtimesmbox{time levels})$ can be specified.

item If code{useSEAICE=.TRUE.} then additional files code{OB[N/S/E/W][a,h,sl,sn,uice,vice]} for sea ice area, thickness (code{HEFF}), seaice salinity, snow and ice velocities $(N_{x/y}timesmbox{time levels})$ can be specified.

end{itemize} As in code{S/R external_fields_load} or the code{exf}-package, the code reads two time levels for each variable, e.g.code{OBNu0} and code{OBNu1}, and interpolates linearly between these time levels to obtain the value code{OBNu} at the current model time (step). When the code{exf}-package is used, the time levels are controlled for each boundary separately in the same way as the code{exf}-fields in code{data.exf}, namelist code{EXF_NML_OBCS}. The runtime flags follow the above naming conventions, e.g. for the western boundary the corresponding flags are code{OBCWstartdate1/2} and code{OBCWperiod}. Sea-ice boundary values are controlled separately with code{siobWstartdate1/2} and code{siobWperiod}. When the code{exf}-package is not used, the time levels are controlled by the runtime flags code{externForcingPeriod} and code{externForcingCycle} in code{data}, see code{verification/exp4} for an example.

paragraph{OBCS_CALC_STEVENS:} ~ \ (THE IMPLEMENTATION OF THESE BOUNDARY CONDITIONS IS NOT COMPLETE. PASSIVE TRACERS, SEA ICE AND NON-LINEAR FREE SURFACE ARE NOT SUPPORTED PROPERLY.) \ The boundary conditions following citet{stevens:90} require the vertically averaged normal velocity (originally specified as a stream function along the open boundary) $bar{u}_{ob}$ and the tracer fields chi_{ob} (note: passive tracers are currently not implemented and the code stops when package code{ptracers} is used together with this option). Currently, the code vertically averages the normal velocity as specified in code{OB[E,W]u} or code{OB[N,S]v}. From these prescribed values the code computes the boundary values for the next timestep $n+1$ as follows (as an example, we use the notation for an eastern or western boundary): begin{itemize} item $u^{n+1}(y,z) = bar{u}_{ob}(y) + (u')^{n}(y,z)$, where

$(u')^{n}$ is the deviation from the vertically averaged velocity at timestep n on the boundary. $(u')^{n}$ is computed in the previous time step n from the intermediate velocity u^* prior to the correction step (see section ref{sec:time_stepping}, e.g., eq.,(ref{eq:ustar-backward-free-surface})). % and~(ref{eq:vstar-backward-free-surface})). (This velocity is not available at the beginning of the next time step $n+1$, when S/R~OBCS_CALC/OBCS_CALC_STEVENS are called, therefore it needs to be

saved in S/R~DYNAMICS by calling S/R~OBCS_SAVE_UV_N and also stored in a separate restart files verb+pickup_stevens[N/S/E/W].${iteration}.data+)

% Define CPP-flag OBCS_STEVENS_USE_INTERIOR_VELOCITY to use the % velocity one grid point inward from the boundary. item If u^{n+1} is directed into the model domain, the boudary

value for tracer chi is restored to the prescribed values: [chi^{n+1} = chi^{n} + frac{Delta{t}}{tau_chi} (chi_{ob} - chi^{n}),] where tau_chi is the relaxation time scale texttt{T/SrelaxStevens}. The new chi^{n+1} is then subject to the advection by u^{n+1}.

item If u^{n+1} is directed out of the model domain, the tracer chi^{n+1} on the boundary at timestep $n+1$ is estimated from advection out of the domain with $u^{n+1}+c$, where c is a phase velocity estimated as $frac{1}{2}frac{partialchi}{partial{t}}/frac{partialchi}{partial{x}}$. The numerical scheme is (as an example for an eastern boundary): [chi_{i_{b},j,k}^{n+1} = chi_{i_{b},j,k}^{n} + Delta{t} (u^{n+1}+c)_{i_{b},j,k}frac{chi_{i_{b},j,k}^{n}

- chi_{i_{b}-1,j,k}^{n}}{Delta{x}_{i_{b},j}^{C}}mbox{, if }u_{i_{b},j,k}^{n+1}>0,

] where i_{b} is the boundary index.\ For test purposes, the phase velocity contribution or the entire advection can be turned off by setting the corresponding parameters texttt{useStevensPhaseVel} and texttt{useStevensAdvection} to texttt{.FALSE.}.

end{itemize} See citet{stevens:90} for details. With this boundary condition specifying the exact net transport across the open boundary is simple, so that balancing the flow with (S/R~OBCS_BALANCE_FLOW, see next paragraph) is usually not necessary.

paragraph{OBCS_BALANCE_FLOW:} ~ \ % When turned on (code{ALLOW_OBCS_BALANCE} defined in code{OBCS_OPTIONS.h} and code{useOBCSbalance=.true.} in code{data.obcs/OBCS_PARM01}), this routine balances the net flow across the open boundaries. By default the net flow across the boundaries is computed and all normal velocities on boundaries are adjusted to obtain zero net inflow.

This behavior can be controlled with the runtime flags code{OBCS_balanceFacN/S/E/W}. The values of these flags determine how the net inflow is redistributed as small correction velocities between the individual sections. A value ‘‘code{-1}” balances an individual boundary, values >0 determine the relative size of the correction. For example, the values begin{tabbing}

code{OBCS_balanceFacE}code{ = 1.,} \ code{OBCS_balanceFacW}code{ = -1.,} \ code{OBCS_balanceFacN}code{ = 2.,} \ code{OBCS_balanceFacS}code{ = 0.,}

end{tabbing} make the model begin{itemize} item correct Western code{OBWu} by substracting a uniform velocity to ensure zero net transport through the Western open boundary; item correct Eastern and Northern normal flow, with the Northern

velocity correction two times larger than the Eastern correction, but emph{not} the Southern normal flow, to ensure that the total inflow through East, Northern, and Southern open boundary is balanced.

end{itemize}

The old method of balancing the net flow for all sections individually can be recovered by setting all flags to -1. Then the normal velocities across each of the four boundaries are modified separately, so that the net volume transport across emph{each} boundary is zero. For example, for the western boundary at $i=i_{b}$, the modified velocity is: [u(y,z) - int_{mbox{western boundary}}u,dy,dz approx OBNu(j,k) - sum_{j,k} OBNu(j,k) h_{w}(i_{b},j,k)Delta{y_G(i_{b},j)}Delta{z(k)}.] This also ensures a net total inflow of zero through all boundaries, but this combination of flags is emph{not} useful if you want to simulate, say, a sector of the Southern Ocean with a strong ACC entering through the western and leaving through the eastern boundary, because the value of ‘‘code{-1}” for these flags will make sure that the strong inflow is removed. Clearly, gobal balancing with code{OBCS_balanceFacE/W/N/S} $ge0$ is the preferred method.

paragraph{OBCS_APPLY_*:} ~ \ ~

paragraph{OBCS_SPONGE:} ~ \ % The sponge layer code (turned on with code{ALLOW_OBCS_SPONGE} and code{useOBCSsponge}) adds a relaxation term to the right-hand-side of the momentum and tracer equations. The variables are relaxed towards the boundary values with a relaxation time scale that increases linearly with distance from the boundary [G_{chi}^{mbox{(sponge)}} = - frac{chi - [(L - delta{L}) chi_{BC} + delta{L}chi]/L} {[(L-delta{L})tau_{b}+delta{L}tau_{i}]/L} = - frac{chi - [(1 - l) chi_{BC} + lchi]} {[(1-l)tau_{b}+ltau_{i}]}] where chi is the model variable (U/V/T/S) in the interior, chi_{BC} the boundary value, L the thickness of the sponge layer (runtime parameter code{spongeThickness} in number of grid points), $delta{L}in[0,L]$ ($frac{delta{L}}{L}=lin[0,1]$) the distance from the boundary (also in grid points), and tau_{b} (runtime parameters code{Urelaxobcsbound} and code{Vrelaxobcsbound}) and tau_{i} (runtime parameters code{Urelaxobcsinner} and code{Vrelaxobcsinner}) the relaxation time scales on the boundary and at the interior termination of the sponge layer. The parameters code{Urelaxobcsbound/inner} set the relaxation time scales for the Eastern and Western boundaries, code{Vrelaxobcsbound/inner} for the Northern and Southern boundaries.

paragraph{OB's with nonlinear free surface} ~ \ % ~

%——————————————————————-

subsubsection{Flow chart label{sec:pkg:obcs:flowchart}}

{footnotesize begin{verbatim}

C !CALLING SEQUENCE: c ...

end{verbatim} }

%——————————————————————-

subsubsection{OBCS diagnostics label{sec:pkg:obcs:diagnostics}}

Diagnostics output is available via the diagnostics package (see Section ref{sec:pkg:diagnostics}). Available output fields are summarized in Table ref{tab:pkg:obcs:diagnostics}.

begin{table}[!ht] centering label{tab:pkg:obcs:diagnostics} {footnotesize begin{verbatim} ——————————————————

<-Name->|Levs|grid|<– Units –>|<- Tile (max=80c)

---

end{verbatim} } caption{~} end{table}

%——————————————————————-

subsubsection{Reference experiments} In the directory code{verifcation}, the following experiments use code{obcs}: begin{itemize} item code{exp4}: box with 4 open boundaries, simulating flow over a

Gaussian bump based on citet{adcroft:97}, also tests Stevens-boundary conditions;

item code{dome}: based on the project ''Dynamics of Overflow Mixing and Entrainment'' (url{http://www.rsmas.miami.edu/personal/tamay/DOME/dome.html}), uses Orlanski-BCs;

item code{internal_wave}: uses a heavily modified code{S/R~OBCS_CALC} item code{seaice_obcs}: simple example who to use the sea-ice

related code, based on code{lab_sea};

item code{tutorial_plume_on_slope}: uses Orlanski-BCs, see also section~ref{sec:eg-gravityplume}.

end{itemize}

%——————————————————————-

subsubsection{References}

subsubsection{Experiments and tutorials that use obcs} label{sec:pkg:obcs:experiments}

begin{itemize} item code{tutorial_plume_on_slope} (section~ref{sec:eg-gravityplume}) end{itemize}

%%% Local Variables: %%% mode: latex %%% TeX-master: "../../manual" %%% End:

newpage input{s_phys_pkgs/text/rbcs.tex}

newpage input{s_phys_pkgs/text/ptracers.tex}

% Ocean Packages newpage

2.3 Ocean Packages

input{s_phys_pkgs/text/gmredi.tex}

newpage input{s_phys_pkgs/text/kpp.tex}

newpage input{s_phys_pkgs/text/ggl90.tex}

newpage input{s_phys_pkgs/text/opps.tex}

newpage input{s_phys_pkgs/text/kl10.tex}

newpage input{s_phys_pkgs/text/bulk_force.tex}

newpage input{s_phys_pkgs/text/exf.tex}

newpage input{s_phys_pkgs/text/cal.tex}

newpage section{Atmosphere Packages} input{s_phys_pkgs/text/aim.tex}

newpage input{s_phys_pkgs/text/land.tex}

newpage input{s_phys_pkgs/text/fizhi.tex}

newpage section{Sea Ice Packages} input{s_phys_pkgs/text/thsice.tex}

newpage input{s_phys_pkgs/text/seaice.tex}

newpage input{s_phys_pkgs/text/shelfice.tex}

newpage input{s_phys_pkgs/text/streamice.tex}

newpage section{Packages Related to Coupled Model} input{s_phys_pkgs/text/aim_compon_interf.tex}

newpage input{s_phys_pkgs/text/atm_ocn_coupler.tex}

newpage input{s_phys_pkgs/text/component_communications.tex}

newpage section{Biogeochemistry Packages} input{s_phys_pkgs/text/gchem.tex}

newpage input{s_phys_pkgs/text/dic.tex}

CHAPTER

THREE

GETTING STARTED WITH MITGCM

This chapter is divided into two main parts. The first part, which is covered in sections ref{sec:whereToFindInfo} through ref{sec:testing}, contains information about how to run experiments using MITgcm. The second part, covered in sections ref{sec:eg-baro} through ref{sec:eg-offline}, contains a set of step-by-step tutorials for running specific pre-configured atmospheric and oceanic experiments.

We believe the best way to familiarize yourself with the model is to run the case study examples provided with the base version. Information on how to obtain, compile, and run the code is found here as well as a brief description of the model structure directory and the case study examples. Information is also provided here on how to customize the code when you are ready to try implementing the configuration you have in mind. The code and algorithm are described more fully in chapters ref{chap:discretization} and ref{chap:sarch}.

3.1 Where to find information

There is a web-archived support mailing list for the model that you can email at texttt{MITgcm-support@mitgcm.org} or browse at: begin{rawhtml} <A href=http://mitgcm.org/mailman/listinfo/mitgcm-support/ target="idontexist"> end{rawhtml} begin{verbatim} http://mitgcm.org/mailman/listinfo/mitgcm-support/ http://mitgcm.org/pipermail/mitgcm-support/ end{verbatim} begin{rawhtml} </A> end{rawhtml}

section{Obtaining the code} label{sec:obtainingCode} begin{rawhtml} <!– CMIREDIR:obtainingCode: –> end{rawhtml}

MITgcm can be downloaded from our system by following the instructions below. As a courtesy we ask that you send e-mail to us at begin{rawhtml} <A href=mailto:MITgcm-support@mitgcm.org> end{rawhtml} MITgcm-support@mitgcm.org begin{rawhtml} </A> end{rawhtml} to enable us to keep track of who's using the model and in what application. You can download the model two ways:

begin{enumerate} item Using CVS software. CVS is a freely available source code management tool. To use CVS you need to have the software installed. Many systems come with CVS pre-installed, otherwise good places to look for the software for a particular platform are begin{rawhtml} <A href=http://www.cvshome.org/ target="idontexist"> end{rawhtml} cvshome.org begin{rawhtml} </A> end{rawhtml} and begin{rawhtml} <A href=http://www.wincvs.org/ target="idontexist"> end{rawhtml} wincvs.org begin{rawhtml} </A> end{rawhtml} .

item Using a tar file. This method is simple and does not require any special software. However, this method does not provide easy support for maintenance updates.

end{enumerate}

subsection{Method 1 - Checkout from CVS} label{sec:cvs_checkout}

If CVS is available on your system, we strongly encourage you to use it. CVS provides an efficient and elegant way of organizing your code and keeping track of your changes. If CVS is not available on your machine, you can also download a tar file.

Before you can use CVS, the following environment variable(s) should be set within your shell. For a csh or tcsh shell, put the following begin{verbatim} % setenv CVSROOT :pserver:cvsanon@mitgcm.org:/u/gcmpack end{verbatim} in your texttt{.cshrc} or texttt{.tcshrc} file. For bash or sh shells, put: begin{verbatim} % export CVSROOT=':pserver:cvsanon@mitgcm.org:/u/gcmpack' end{verbatim} in your texttt{.profile} or texttt{.bashrc} file.

To get MITgcm through CVS, first register with the MITgcm CVS server using command: begin{verbatim} % cvs login (CVS password: cvsanon) end{verbatim} You only need to do a ‘‘cvs login” once.

To obtain the latest sources type: begin{verbatim} % cvs co -P MITgcm end{verbatim} or to get a specific release type: begin{verbatim} % cvs co -P -r checkpoint52i_post MITgcm end{verbatim} The CVS command ‘‘texttt{cvs co}” is the abreviation of the full-name ‘‘texttt{cvs checkout}” command and using the option ‘‘-P” (texttt{cvs co -P}) will prevent to download unnecessary empty directories.

The MITgcm web site contains further directions concerning the source code and CVS. It also contains a web interface to our CVS archive so that one may easily view the state of files, revisions, and other development milestones: begin{rawhtml} <A href=”http://mitgcm.org/viewvc/MITgcm/MITgcm/” target=”idontexist”> end{rawhtml} begin{verbatim} http://mitgcm.org/viewvc/MITgcm/MITgcm/ end{verbatim} begin{rawhtml} </A> end{rawhtml}

As a convenience, the MITgcm CVS server contains aliases which are named subsets of the codebase. These aliases can be especially helpful when used over slow internet connections or on machines with restricted storage space. Table ref{tab:cvsModules} contains a list of CVS aliases begin{table}[htb]

centering begin{tabular}[htb]{**|lp{3.25in}|**}hline

textbf{Alias Name} & textbf{Information (directories) Contained} \hline texttt{MITgcm_code} & Only the source code – none of the verification examples. \ texttt{MITgcm_verif_basic} & Source code plus a small set of the verification examples (texttt{global_ocean.90x40x15}, texttt{aim.5l_cs}, texttt{hs94.128x64x5}, texttt{front_relax}, and texttt{plume_on_slope}). \ texttt{MITgcm_verif_atmos} & Source code plus all of the atmospheric examples. \ texttt{MITgcm_verif_ocean} & Source code plus all of the oceanic examples. \ texttt{MITgcm_verif_all} & Source code plus all of the verification examples. \hline

end{tabular} caption{MITgcm CVS Modules} label{tab:cvsModules}

end{table}

The checkout process creates a directory called texttt{MITgcm}. If the directory texttt{MITgcm} exists this command updates your code based on the repository. Each directory in the source tree contains a directory texttt{CVS}. This information is required by CVS to keep track of your file versions with respect to the repository. Don’t edit the files in texttt{CVS}! You can also use CVS to download code updates. More extensive information on using CVS for maintaining MITgcm code can be found begin{rawhtml} <A href=”http://mitgcm.org/public/using_cvs.html” target=”idontexist”> end{rawhtml} here begin{rawhtml} </A> end{rawhtml}. It is important to note that the CVS aliases in Table ref{tab:cvsModules} cannot be used in conjunction with the CVS texttt{-d DIRNAME} option. However, the texttt{MITgcm} directories they create can be changed to a different name following the check-out: begin{verbatim}

% cvs co -P MITgcm_verif_basic % mv MITgcm MITgcm_verif_basic

end{verbatim}

Note that it is possible to checkout code without ‘‘cvs login” and without setting any shell environment variables by specifying the pserver name and password in one line, for example: begin{verbatim}

% cvs -d :pserver:cvsanon:cvsanon@mitgcm.org:/u/gcmpack co -P MITgcm

end{verbatim}

subsubsection{Upgrading from an earlier version}

If you already have an earlier version of the code you can ‘‘upgrade” your copy instead of downloading the entire repository again. First, ‘‘cd” (change directory) to the top of your working copy: begin{verbatim} % cd MITgcm

end{verbatim} and then issue the cvs update command such as: begin{verbatim} % cvs -q update -d -P -r checkpoint52i_post end{verbatim} This will update the ‘‘tag” to ‘‘checkpoint52i_post’‘, add any new directories (-d) and remove any empty directories (-P). The -q option means be quiet which will reduce the number of messages you’ll see in the terminal. If you have modified the code prior to upgrading, CVS will try to merge your changes with the upgrades. If there is a conflict between your modifications and the upgrade, it will report that file with a ‘‘C” in front, e.g.: begin{verbatim} C model/src/ini_parms.F end{verbatim} If the list of conflicts scrolled off the screen, you can re-issue the cvs update command and it will report the conflicts. Conflicts are indicated in the code by the delimites ‘‘$<<<<<<<$’‘, ‘‘=======” and ‘‘$>>>>>>>$’‘. For example, {small begin{verbatim} <<<<<<< ini_parms.F

& bottomDragLinear,myOwnBottomDragCoefficient,

end{verbatim} } means that you added ‘‘myOwnBottomDragCoefficient” to a namelist at the same time and place that we added ‘‘bottomDragQuadratic’‘. You need to resolve this conflict and in this case the line should be changed to: {small begin{verbatim}

& bottomDragLinear,bottomDragQuadratic,myOwnBottomDragCoefficient,

end{verbatim} } and the lines with the delimiters ($<<<<<<$,======,$>>>>>>$) be deleted. Unless you are making modifications which exactly parallel developments we make, these types of conflicts should be rare.

paragraph*{Upgrading to the current pre-release version}

We don’t make a ‘‘release” for every little patch and bug fix in order to keep the frequency of upgrades to a minimum. However, if you have run into a problem for which ‘‘we have already fixed in the latest code” and we haven’t made a ‘‘tag” or ‘‘release” since that patch then you’ll need to get the latest code: begin{verbatim} % cvs -q update -d -P -A end{verbatim} Unlike, the ‘‘check-out” and ‘‘update” procedures above, there is no ‘‘tag” or release name. The -A tells CVS to upgrade to the very latest version. As a rule, we don’t recommend this since you might upgrade while we are in the processes of checking in the code so that you may only have part of a patch. Using this method of updating also means we can’t tell what version of the code you are working with. So please be sure you understand what you’re doing.

subsection{Method 2 - Tar file download} label{sec:conventionalDownload}

If you do not have CVS on your system, you can download the model as a tar file from the web site at: begin{rawhtml} <A href=http://mitgcm.org/download/ target=”idontexist”> end{rawhtml} begin{verbatim} http://mitgcm.org/download/ end{verbatim} begin{rawhtml} </A> end{rawhtml} The tar file still contains CVS information which we urge you not to delete; even if you do not use CVS yourself the information can help us if you should need to send us your copy of the code. If a recent tar file does not exist, then please contact the developers through the begin{rawhtml} <A href=”mailto:MITgcm-support@mitgcm.org“> end{rawhtml} MITgcm-support@mitgcm.org begin{rawhtml} </A> end{rawhtml} mailing list.

section{Model and directory structure} begin{rawhtml} <!– CMIREDIR:directory_structure: –> end{rawhtml}

The ‘‘numerical” model is contained within a execution environment support wrapper. This wrapper is designed to provide a general framework for grid-point models. MITgcmUV is a specific numerical model that uses the framework. Under this structure the model is split into execution environment support code and conventional numerical model code. The execution environment support code is held under the texttt{eesupp} directory. The grid point model code is held under the texttt{model} directory. Code execution actually starts in the texttt{eesupp} routines and not in the texttt{model} routines. For this reason the top-level texttt{MAIN.F} is in the texttt{eesupp/src} directory. In general, end-users should not need to worry about this level. The top-level routine for the numerical part of the code is in texttt{model/src/THE_MODEL_MAIN.F}. Here is a brief description of the directory structure of the model under the root tree (a detailed description is given in section 3: Code structure).

begin{itemize}

item texttt{doc}: contains brief documentation notes.

item texttt{eesupp}: contains the execution environment source code. Also subdivided into two subdirectories texttt{inc} and texttt{src}.

item texttt{model}: this directory contains the main source code. Also subdivided into two subdirectories texttt{inc} and texttt{src}.

item texttt{pkg}: contains the source code for the packages. Each package corresponds to a subdirectory. For example, texttt{gmredi} contains the code related to the Gent-McWilliams/Redi scheme, texttt{aim} the code relative to the atmospheric intermediate physics. The packages are described in detail in chapter ref{chap:packagesI}.

item texttt{tools}: this directory contains various useful tools. For example, texttt{genmake2} is a script written in csh (C-shell) that should be used to generate your makefile. The directory texttt{adjoint} contains the makefile specific to the Tangent linear and Adjoint Compiler (TAMC) that generates the adjoint code. The latter is described in detail in part ref{chap.ecco}. This directory also contains the subdirectory build_options, which contains the ‘optfiles’ with the compiler options for the different compilers and machines that can run MITgcm.

item texttt{utils}: this directory contains various utilities. The subdirectory texttt{knudsen2} contains code and a makefile that compute coefficients of the polynomial approximation to the knudsen formula for an ocean nonlinear equation of state. The texttt{matlab} subdirectory contains matlab scripts for reading model output directly into matlab. texttt{scripts} contains C-shell post-processing scripts for joining processor-based and tiled-based model output. The subdirectory exch2 contains the code needed for the exch2 package to work with different combinations of domain decompositions.

item texttt{verification}: this directory contains the model examples. See section ref{sec:modelExamples}.

item texttt{jobs}: contains sample job scripts for running MITgcm.

item texttt{lsopt}: Line search code used for optimization.

item texttt{optim}: Interface between MITgcm and line search code.

end{itemize}

section[Building MITgcm]{Building the code} label{sec:buildingCode} begin{rawhtml} <!– CMIREDIR:buildingCode: –> end{rawhtml}

To compile the code, we use the texttt{make} program. This uses a file (texttt{Makefile}) that allows us to pre-process source files, specify compiler and optimization options and also figures out any file dependencies. We supply a script (texttt{genmake2}), described in section ref{sec:genmake}, that automatically creates the texttt{Makefile} for you. You then need to build the dependencies and compile the code.

As an example, assume that you want to build and run experiment texttt{verification/exp2}. The are multiple ways and places to actually do this but here let’s build the code in texttt{verification/exp2/build}: begin{verbatim} % cd verification/exp2/build end{verbatim} First, build the texttt{Makefile}: begin{verbatim} % ../../../tools/genmake2 -mods=../code end{verbatim} The command line option tells texttt{genmake} to override model source code with any files in the directory texttt{../code/}.

On many systems, the texttt{genmake2} program will be able to automatically recognize the hardware, find compilers and other tools within the user’s path (‘‘texttt{echo $PATH}’’), and then choose an appropriate set of options from the files (‘‘optfiles’‘) contained in the texttt{tools/build_options} directory. Under some circumstances, a user may have to create a new ‘‘optfile” in order to specify the exact combination of compiler, compiler flags, libraries, and other options necessary to build a particular configuration of MITgcm. In such cases, it is generally helpful to read the existing ‘‘optfiles” and mimic their syntax.

Through the MITgcm-support list, the MITgcm developers are willing to provide help writing or modifing ‘‘optfiles’‘. And we encourage users to post new ‘‘optfiles” (particularly ones for new machines or architectures) to the begin{rawhtml} <A href=”mailto:MITgcm-support@mitgcm.org“> end{rawhtml} MITgcm-support@mitgcm.org begin{rawhtml} </A> end{rawhtml} list.

To specify an optfile to texttt{genmake2}, the syntax is: begin{verbatim} % ../../../tools/genmake2 -mods=../code -of /path/to/optfile end{verbatim}

Once a texttt{Makefile} has been generated, we create the dependencies with the command: begin{verbatim} % make depend end{verbatim} This modifies the texttt{Makefile} by attaching a (usually, long) list of files upon which other files depend. The purpose of this is to reduce re-compilation if and when you start to modify the code. The {tt make depend} command also creates links from the model source to this directory. It is important to note that the {tt make depend} stage will occasionally produce warnings or errors since the dependency parsing tool is unable to find all of the necessary header files (textit{eg.} texttt{netcdf.inc}). In these circumstances, it is usually OK to ignore the warnings/errors and proceed to the next step.

Next one can compile the code using: begin{verbatim} % make end{verbatim} The {tt make} command creates an executable called texttt{mitgcmuv}. Additional make ‘‘targets” are defined within the makefile to aid in the production of adjoint and other versions of MITgcm. On SMP (shared multi-processor) systems, the build process can often be sped up appreciably using the command: begin{verbatim} % make -j 2 end{verbatim} where the ‘‘2” can be replaced with a number that corresponds to the number of CPUs available.

Now you are ready to run the model. General instructions for doing so are given in section ref{sec:runModel}. Here, we can run the model by first creating links to all the input files: begin{verbatim} ln -s ../input/* . end{verbatim} and then calling the executable with: begin{verbatim} ./mitgcmuv > output.txt end{verbatim} where we are re-directing the stream of text output to the file texttt{output.txt}.

subsection{Building/compiling the code elsewhere}

In the example above (section ref{sec:buildingCode}) we built the executable in the {em input} directory of the experiment for convenience. You can also configure and compile the code in other locations, for example on a scratch disk with out having to copy the entire source tree. The only requirement to do so is you have {tt

genmake2} in your path or you know the absolute path to {tt genmake2}.

The following sections outline some possible methods of organizing your source and data.

subsubsection{Building from the {em ../code directory}}

This is just as simple as building in the {em input/} directory: begin{verbatim} % cd verification/exp2/code % ../../../tools/genmake2 % make depend % make end{verbatim} However, to run the model the executable ({em mitgcmuv}) and input files must be in the same place. If you only have one calculation to make: begin{verbatim} % cd ../input % cp ../code/mitgcmuv ./ % ./mitgcmuv > output.txt end{verbatim} or if you will be making multiple runs with the same executable: begin{verbatim} % cd ../ % cp -r input run1 % cp code/mitgcmuv run1 % cd run1 % ./mitgcmuv > output.txt end{verbatim}

subsubsection{Building from a new directory}

Since the {em input} directory contains input files it is often more useful to keep {em input} pristine and build in a new directory within {em verification/exp2/}: begin{verbatim} % cd verification/exp2 % mkdir build % cd build % ../../../tools/genmake2 -mods=../code % make depend % make end{verbatim} This builds the code exactly as before but this time you need to copy either the executable or the input files or both in order to run the model. For example, begin{verbatim} % cp ../input/* ./ % ./mitgcmuv > output.txt end{verbatim} or if you tend to make multiple runs with the same executable then running in a new directory each time might be more appropriate: begin{verbatim} % cd ../ % mkdir run1 % cp build/mitgcmuv run1/ % cp input/* run1/ % cd run1 % ./mitgcmuv > output.txt end{verbatim}

subsubsection{Building on a scratch disk}

Model object files and output data can use up large amounts of disk space so it is often the case that you will be operating on a large scratch disk. Assuming the model source is in {em ~/MITgcm} then the following commands will build the model in {em /scratch/exp2-run1}: begin{verbatim} % cd /scratch/exp2-run1 % ~/MITgcm/tools/genmake2 -rootdir=~/MITgcm

-mods=~/MITgcm/verification/exp2/code

% make depend % make end{verbatim} To run the model here, you’ll need the input files: begin{verbatim} % cp ~/MITgcm/verification/exp2/input/* ./ % ./mitgcmuv > output.txt end{verbatim}

As before, you could build in one directory and make multiple runs of the one experiment: begin{verbatim} % cd /scratch/exp2 % mkdir build % cd build % ~/MITgcm/tools/genmake2 -rootdir=~/MITgcm

-mods=~/MITgcm/verification/exp2/code

% make depend % make % cd ../ % cp -r ~/MITgcm/verification/exp2/input run2 % cd run2 % ./mitgcmuv > output.txt end{verbatim}

subsection{Using texttt{genmake2}} label{sec:genmake}

To compile the code, first use the program texttt{genmake2} (located in the texttt{tools} directory) to generate a Makefile. texttt{genmake2} is a shell script written to work with all ‘‘sh’‘–compatible shells including bash v1, bash v2, and Bourne. %Internally, texttt{genmake2} determines the locations of needed %files, the compiler, compiler options, libraries, and Unix tools. It %relies upon a number of ‘‘optfiles” located in the %texttt{tools/build_options} directory. texttt{genmake2} parses information from the following sources: begin{description} item[-] a {em gemake_local} file if one is found in the current

directory

item[-] command-line options item[-] an “options file” as specified by the command-line option

texttt{–optfile=/PATH/FILENAME}

item[-] a {em packages.conf} file (if one is found) with the specific list of packages to compile. The search path for file {em packages.conf} is, first, the current directory and then each of the “MODS” directories in the given order (see below).

end{description}

subsubsection{Optfiles in texttt{tools/build_options} directory:}

The purpose of the optfiles is to provide all the compilation options for particular ‘‘platforms” (where ‘‘platform” roughly means the combination of the hardware and the compiler) and code configurations. Given the combinations of possible compilers and library dependencies ({it eg.} MPI and NetCDF) there may be numerous optfiles available for a single machine. The naming scheme for the majority of the optfiles shipped with the code is begin{center}

{bf OS_HARDWARE_COMPILER }

end{center} where begin{description} item[OS] is the name of the operating system (generally the

lower-case output of the {tt ‘uname’} command)

item[HARDWARE] is a string that describes the CPU type and corresponds to output from the {tt ‘uname -m’} command: begin{description} item[ia32] is for ‘‘x86” machines such as i386, i486, i586, i686,

and athlon

item[ia64] is for Intel IA64 systems (eg. Itanium, Itanium2) item[amd64] is AMD x86_64 systems item[ppc] is for Mac PowerPC systems end{description}

item[COMPILER] is the compiler name (generally, the name of the FORTRAN executable)

end{description}

In many cases, the default optfiles are sufficient and will result in usable Makefiles. However, for some machines or code configurations, new ‘‘optfiles” must be written. To create a new optfile, it is generally best to start with one of the defaults and modify it to suit your needs. Like texttt{genmake2}, the optfiles are all written using a simple ‘‘sh’‘–compatible syntax. While nearly all variables used within texttt{genmake2} may be specified in the optfiles, the critical ones that should be defined are:

begin{description} item[FC] the FORTRAN compiler (executable) to use item[DEFINES] the command-line DEFINE options passed to the compiler item[CPP] the C pre-processor to use item[NOOPTFLAGS] options flags for special files that should not be

optimized

end{description}

For example, the optfile for a typical Red Hat Linux machine (‘‘ia32” architecture) using the GCC (g77) compiler is begin{verbatim} FC=g77 DEFINES=’-D_BYTESWAPIO -DWORDLENGTH=4’ CPP=’cpp -traditional -P’ NOOPTFLAGS=’-O0’ # For IEEE, use the “-ffloat-store” option if test “x$IEEE” = x ; then

FFLAGS=’-Wimplicit -Wunused -Wuninitialized’ FOPTIM=’-O3 -malign-double -funroll-loops’

else FFLAGS=’-Wimplicit -Wunused -ffloat-store’ FOPTIM=’-O0 -malign-double’

fi end{verbatim}

If you write an optfile for an unrepresented machine or compiler, you are strongly encouraged to submit the optfile to the MITgcm project for inclusion. Please send the file to the begin{rawhtml} <A href=”mail-to:MITgcm-support@mitgcm.org”> end{rawhtml} begin{center}

MITgcm-support@mitgcm.org

end{center} begin{rawhtml} </A> end{rawhtml} mailing list.

subsubsection{Command-line options:}

In addition to the optfiles, texttt{genmake2} supports a number of helpful command-line options. A complete list of these options can be obtained from: begin{verbatim} % genmake2 -h end{verbatim}

The most important command-line options are: begin{description}

item[texttt{–optfile=/PATH/FILENAME}] specifies the optfile that should be used for a particular build.

If no “optfile” is specified (either through the command line or the MITGCM_OPTFILE environment variable), genmake2 will try to make a reasonable guess from the list provided in {em

tools/build_options}. The method used for making this guess is

to first determine the combination of operating system and hardware (eg. “linux_ia32”) and then find a working FORTRAN compiler within the user’s path. When these three items have been identified, genmake2 will try to find an optfile that has a matching name.

item[texttt{–mods=’DIR1 DIR2 DIR3 ...’}] specifies a list of directories containing ‘‘modifications’‘. These directories contain files with names that may (or may not) exist in the main MITgcm source tree but will be overridden by any identically-named sources within the ‘‘MODS” directories.

The order of precedence for this “name-hiding” is as follows: begin{itemize} item ‘‘MODS” directories (in the order given) item Packages either explicitly specified or provided by default

(in the order given)

item Packages included due to package dependencies (in the order that that package dependencies are parsed)

item The “standard dirs” (which may have been specified by the ‘‘-standarddirs” option)

end{itemize}

item[texttt{–pgroups=/PATH/FILENAME}] specifies the file where package groups are defined. If not set, the package-groups definition will be read from {em pkg/pkg_groups}. It also contains the default list of packages (defined as the group ‘‘{it default_pkg_list}” which is used when no specific package list ({em packages.conf}) is found in current directory or in any “MODS” directory.

item[texttt{–pdepend=/PATH/FILENAME}] specifies the dependency file used for packages.

If not specified, the default dependency file {em pkg/pkg_depend} is used. The syntax for this file is parsed on a line-by-line basis where each line containes either a comment (“#”) or a simple “PKGNAME1 (+|-)PKGNAME2” pairwise rule where the “+” or “-” symbol specifies a “must be used with” or a “must not

be used with" relationship, respectively. If no rule is specified, then it is assumed that the two packages are compatible and will function either with or without each other.

item[texttt{–adof=/path/to/file}] specifies the "adjoint" or automatic differentiation options file to be used. The file is analogous to the ‘‘optfile" defined above but it specifies information for the AD build process.

The default file is located in {em tools/adjoint_options/adjoint_default} and it defines the "TAF"

and "TAMC" compilers. An alternate version is also available at {em tools/adjoint_options/adjoint_staf} that selects the newer "STAF" compiler. As with any compilers, it is helpful to have their directories listed in your {tt $PATH} environment variable.

item[texttt{–mpi}] This option enables certain MPI features (using CPP texttt{#define}s) within the code and is necessary for MPI builds (see Section ref{sec:mpi-build}).

item[texttt{–make=/path/to/gmake}] Due to the poor handling of soft-links and other bugs common with the texttt{make} versions provided by commercial Unix vendors, GNU texttt{make} (sometimes called texttt{gmake}) should be preferred. This option provides a means for specifying the make executable to be used.

item[texttt{–bash=/path/to/sh}] On some (usually older UNIX) machines, the ‘‘bash" shell is unavailable. To run on these systems, texttt{genmake2} can be invoked using an ‘‘sh" (that is, a Bourne, POSIX, or compatible) shell. The syntax in these circumstances is: begin{center}

texttt{% /bin/sh genmake2 -bash=/bin/sh [...options...]}

end{center} where texttt{/bin/sh} can be replaced with the full path and name of the desired shell.

end{description}

subsection{Building with MPI} label{sec:mpi-build}

Building MITgcm to use MPI libraries can be complicated due to the variety of different MPI implementations available, their dependencies or interactions with different compilers, and their often ad-hoc locations within file systems. For these reasons, its generally a good idea to start by finding and reading the documentation for your machine(s) and, if necessary, seeking help from your local systems administrator.

The steps for building MITgcm with MPI support are: begin{enumerate}

item Determine the locations of your MPI-enabled compiler and/or MPI libraries and put them into an options file as described in Section ref{sec:genmake}. One can start with one of the examples in: begin{rawhtml} <A

href="http://mitgcm.org/viewvc/MITgcm/MITgcm/tools/build_options/">

end{rawhtml} begin{center}

texttt{MITgcm/tools/build_options/}

end{center} begin{rawhtml} </A> end{rawhtml} such as texttt{linux_ia32_g77+mpi_cg01} or texttt{linux_ia64_efc+mpi} and then edit it to suit the machine at hand. You may need help from your user guide or local systems administrator to determine the exact location of the MPI libraries. If libraries are not installed, MPI implementations and related tools are available including: begin{itemize} item begin{rawhtml} <A

href="http://www-unix.mcs.anl.gov/mpi/mpich/">

end{rawhtml} MPICH begin{rawhtml} </A> end{rawhtml}

item begin{rawhtml} <A

href="http://www.lam-mpi.org/">

end{rawhtml} LAM/MPI begin{rawhtml} </A> end{rawhtml}

item begin{rawhtml} <A

href="http://www.osc.edu/~pw/mpiexec/">

end{rawhtml} MPIexec begin{rawhtml} </A> end{rawhtml}

end{itemize}

item Build the code with the texttt{genmake2} texttt{-mpi} option (see Section ref{sec:genmake}) using commands such as:

{footnotesize begin{verbatim} % ../../../tools/genmake2 -mods=../code -mpi -of=YOUR_OPTFILE % make depend % make

end{verbatim} }

item Run the code with the appropriate MPI ‘‘run" or ‘‘exec"‘ program provided with your particular implementation of MPI. Typical MPI packages such as MPICH will use something like:

begin{verbatim} % mpirun -np 4 -machinefile mf ./mitgcmuv

end{verbatim} Sightly more complicated scripts may be needed for many machines since execution of the code may be controlled by both the MPI library and a job scheduling and queueing system such as PBS, LoadLeveller, Condor, or any of a number of similar tools. A few example scripts (those used for our begin{rawhtml} <A

href="http://mitgcm.org/public/testing.html"> end{rawhtml}regular

verification runsbegin{rawhtml} </A> end{rawhtml}) are available at: begin{rawhtml} <A

href="http://mitgcm.org/viewvc/MITgcm/MITgcm/tools/example_scripts/">

end{rawhtml} {footnotesize tt

http://mitgcm.org/viewvc/MITgcm/MITgcm/tools/example_scripts/ }

begin{rawhtml} </A> end{rawhtml} or at: begin{rawhtml} <A

href="http://mitgcm.org/viewvc/MITgcm/MITgcm_contrib/test_scripts/">

end{rawhtml} {footnotesize tt

http://mitgcm.org/viewvc/MITgcm/MITgcm_contrib/test_scripts/ }

begin{rawhtml} </A> end{rawhtml}

end{enumerate}

An example of the above process on the MITgcm cluster (‘‘cg01’‘) using the GNU g77 compiler and the mpich MPI library is:

{footnotesize begin{verbatim} % cd MITgcm/verification/exp5 % mkdir build % cd build % ../../../tools/genmake2 -mpi -mods=../code

-of=../../../tools/build_options/linux_ia32_g77+mpi_cg01

% make depend % make % cd ../input % /usr/local/pkg/mpi/mpi-1.2.4..8a-gm-1.5/g77/bin/mpirun.ch_gm

-machinefile mf –gm-kill 5 -v -np 2 ../build/mitgcmuv

end{verbatim} }

section[Running MITgcm]{Running the model in prognostic mode} label{sec:runModel} begin{rawhtml} <!–CMIREDIR:runModel: –> end{rawhtml}

If compilation finished succesfully (section ref{sec:buildingCode}) then an executable called texttt{mitgcmuv} will now exist in the local directory.

To run the model as a single process (textit{ie.} not in parallel) simply type: begin{verbatim} % ./mitgcmuv end{verbatim} The ‘‘./" is a safe-guard to make sure you use the local executable in case you have others that exist in your path (surely odd if you do!). The above command will spew out many lines of text output to your screen. This

output contains details such as parameter values as well as diagnostics such as mean Kinetic energy, largest CFL number, etc. It is worth keeping this text output with the binary output so we normally re-direct the texttt{stdout} stream as follows: begin{verbatim} % ./mitgcmuv > output.txt end{verbatim} In the event that the model encounters an error and stops, it is very helpful to include the last few line of this texttt{output.txt} file along with the (texttt{stderr}) error message within any bug reports.

For the example experiments in texttt{verification}, an example of the output is kept in texttt{results/output.txt} for comparison. You can compare your texttt{output.txt} with the corresponding one for that experiment to check that the set-up works.

subsection{Output files}

The model produces various output files and, when using texttt{mnc}, sometimes even directories. Depending upon the I/O package(s) selected at compile time (either texttt{mdsio} or texttt{mnc} or both as determined by texttt{code/packages.conf}) and the run-time flags set (in texttt{input/data.pkg}), the following output may appear.

subsubsection{MDSIO output files}

The ‘‘traditional” output files are generated by the texttt{mdsio} package. At a minimum, the instantaneous ‘‘state” of the model is written out, which is made of the following files:

begin{itemize} item texttt{U.00000nIter} - zonal component of velocity field (m/s

and positive eastward).

item texttt{V.00000nIter} - meridional component of velocity field (m/s and positive northward).

item texttt{W.00000nIter} - vertical component of velocity field (ocean: m/s and positive upward, atmosphere: Pa/s and positive towards increasing pressure i.e. downward).

item texttt{T.00000nIter} - potential temperature (ocean: $^{circ}mathrm{C}$, atmosphere: $^{circ}mathrm{K}$).

item texttt{S.00000nIter} - ocean: salinity (psu), atmosphere: water vapor (g/kg).

item texttt{Eta.00000nIter} - ocean: surface elevation (m), atmosphere: surface pressure anomaly (Pa).

end{itemize}

The chain texttt{00000nIter} consists of ten figures that specify the iteration number at which the output is written out. For example, texttt{U.0000000300} is the zonal velocity at iteration 300.

In addition, a ‘‘pickup” or ‘‘checkpoint” file called:

begin{itemize} item texttt{pickup.00000nIter} end{itemize}

is written out. This file represents the state of the model in a condensed form and is used for restarting the integration. If the C-D scheme is used, there is an additional ‘‘pickup” file:

begin{itemize} item texttt{pickup_cd.00000nIter} end{itemize}

containing the D-grid velocity data and that has to be written out as well in order to restart the integration. Rolling checkpoint files are the same as the pickup files but are named differently. Their name contain the chain texttt{ckptA} or texttt{ckptB} instead of texttt{00000nIter}. They can be used to restart the model but are overwritten every other time they are output to save disk space during long integrations.

subsubsection{MNC output files}

Unlike the texttt{mdsio} output, the texttt{mnc}–generated output is usually (though not necessarily) placed within a subdirectory with a name such as texttt{mnc_test_${DATE}_${SEQ}}.

subsection{Looking at the output}

The ‘‘traditional” or mdsio model data are written according to a ‘‘meta/data” file format. Each variable is associated with two files with suffix names texttt{.data} and texttt{.meta}. The texttt{.data} file contains the data written in

binary form (big_endian by default). The texttt{.meta} file is a ‘‘header” file that contains information about the size and the structure of the texttt{.data} file. This way of organizing the output is particularly useful when running multi-processors calculations. The base version of the model includes a few matlab utilities to read output files written in this format. The matlab scripts are located in the directory texttt{utils/matlab} under the root tree. The script texttt{rdmds.m} reads the data. Look at the comments inside the script to see how to use it.

Some examples of reading and visualizing some output in {em Matlab}: begin{verbatim} % matlab >> H=rdmds(‘Depth’); >> contourf(H’);colorbar; >> title(‘Depth of fluid as used by model’);

>> eta=rdmds(‘Eta’,10); >> imagesc(eta’);axis ij;colorbar; >> title(‘Surface height at iter=10’);

>> eta=rdmds(‘Eta’,[0:10:100]); >> for n=1:11; imagesc(eta(:,:,n)’);axis ij;colorbar;pause(.5);end end{verbatim}

Similar scripts for netCDF output (texttt{rdmnc.m}) are available and they are described in Section ref{sec:pkg:mnc}.

The MNC output files are all in the ‘‘self-describing” netCDF format and can thus be browsed and/or plotted using tools such as: begin{itemize} item texttt{ncdump} is a utility which is typically included

with every netCDF install: begin{rawhtml} <A href=”http://www.unidata.ucar.edu/packages/netcdf/“> end{rawhtml}

begin{verbatim} http://www.unidata.ucar.edu/packages/netcdf/ end{verbatim}

begin{rawhtml} </A> end{rawhtml} and it converts the netCDF binaries into formatted ASCII text files.

item texttt{ncview} utility is a very convenient and quick way to plot netCDF data and it runs on most OSes: begin{rawhtml} <A href=”http://meteora.ucsd.edu/~pierce/ncview_home_page.html“> end{rawhtml}

begin{verbatim} http://meteora.ucsd.edu/~pierce/ncview_home_page.html end{verbatim}

begin{rawhtml} </A> end{rawhtml}

item MatLAB(c) and other common post-processing environments provide various netCDF interfaces including: begin{rawhtml} <A href=”http://mexcdf.sourceforge.net/“> end{rawhtml}

begin{verbatim} http://mexcdf.sourceforge.net/ end{verbatim}

begin{rawhtml} </A> end{rawhtml} begin{rawhtml} <A href=”http://woodshole.er.usgs.gov/staffpages/cdenham/public_html/MexCDF/nc4ml5.html“> end{rawhtml}

begin{verbatim} http://woodshole.er.usgs.gov/staffpages/cdenham/public_html/MexCDF/nc4ml5.html end{verbatim}

begin{rawhtml} </A> end{rawhtml}

end{itemize}

CHAPTER

FOUR

MITGCM EXAMPLE EXPERIMENTS

The full MITgcm distribution comes with a set of pre-configured numerical experiments. Some of these example experiments are tests of individual parts of the model code, but many are fully fledged numerical simulations. Full tutorials exist for a few of the examples, and are documented in sections ref{sec:eg-baro} - ref{sec:eg-tank}. The other examples follow the same general structure as the tutorial examples. However, they only include brief instructions in a text file called {it README}. The examples are located in subdirectories under the directory texttt{verification}. Each example is briefly described below.

subsection{Full list of model examples}

begin{enumerate}

item texttt{tutorial_advection_in_gyre} - Test of various advection schemes in a single-layer double-gyre experiment. This experiment is described in detail in section ref{sec:eg-adv-gyre}.

item texttt{tutorial_baroclinic_gyre} - Four layer, ocean double gyre. This experiment is described in detail in section ref{sec:eg-fourlayer}.

item texttt{tutorial_barotropic_gyre} - Single layer, ocean double gyre (barotropic with free-surface). This experiment is described in detail in section ref{sec:eg-baro}.

item texttt{tutorial_cfc_offline} - Offline form of the MITgcm to study advection of a passive tracer and CFCs. This experiment is described in detail in section ref{sec:eg-offline-cfc}.

item texttt{tutorial_deep_convection} - Non-uniformly forced ocean convection in a doubly periodic box. This experiment is described in detail in section ref{sec:eg-bconv}.

item texttt{tutorial_dic_adoffline} - Offline form of MITgcm dynamics coupled to the dissolved inorganic carbon biogeochemistry model; adjoint set-up.

item texttt{tutorial_global_oce_biogeo} - Ocean model coupled to the dissolved inorganic carbon biogeochemistry model. This experiment is described in detail in section ref{sec:eg-biogeochem_tutorial}.

item texttt{tutorial_global_oce_in_p} - Global ocean simulation in pressure coordinate (non-Boussinesq ocean model). Described in detail in section ref{sec:eg-globalpressure}.

item texttt{tutorial_global_oce_latlon} - 4x4 degree global ocean simulation with steady climatological forcing. This experiment is described in detail in section ref{sec:eg-global}.

item texttt{tutorial_global_oce_optim} - Global ocean state estimation at 4^circ resolution. This experiment is described in detail in section ref{sec:eg-global_state_estimate}.

item texttt{tutorial_held_suarez_cs} - 3D atmosphere dynamics using Held and Suarez (1994) forcing on cubed sphere grid. This experiment is described in detail in section ref{sec:eg-hs}.

item texttt{tutorial_offline} - Offline form of the MITgcm to study advection of a passive tracer. This experiment is described in detail in section ref{sec:eg-offline}.

item texttt{tutorial_plume_on_slope} - Gravity Plume on a continental slope. This experiment is described in detail in section ref{sec:eg-gravityplume}.

item texttt{tutorial_tracer_adjsens} - Simple passive tracer experiment. Includes derivative calculation. This experiment is described in detail in section ref{sec:eg-simple-tracer-adjoint}.\ Also contains an additional set-up using Secon Order Moment (SOM) advection scheme ({it input_ad.som81/}).

item texttt{1D_ocean_ice_column} - Oceanic column with seaice on top.

item texttt{adjustment.128x64x1} - Barotropic adjustment problem on latitude longitude grid with 128x64 grid points (2.8^circ resolution).

item texttt{adjustment.cs-32x32x1} - Barotropic adjustment problem on cube sphere grid with 32x32 points per face (roughly 2.8^circ resolution).\ Also contains a non-linear free-surface adjustment version ({it input.nlfs/}).

item texttt{advect_cs} - Two-dimensional passive advection test on cube sphere grid (32x32 grid points per face, roughly 2.8^circ resolution)

item texttt{advect_xy} - Two-dimensional (horizontal plane) passive advection test on Cartesian grid.\ Also contains an additional set-up using Adams-Bashforth 3 ({it input.ab3_c4/}).

item texttt{advect_xz} - Two-dimensional (vertical plane) passive advection test on Cartesian grid.\ Also contains an additional set-up using non-linear free-surface

with divergent barotropic flow and implicit vertical advection ({it input.nlfs/}).

item texttt{aim.5l_Equatorial_Channel} - 5-levels Intermediate Atmospheric physics, 3D Equatorial Channel configuration.

item texttt{aim.5l_LatLon} - 5-levels Intermediate Atmospheric physics, Global configuration, on latitude longitude grid with 128x64x5 grid points (2.8^circ resolution).

item texttt{aim.5l_cs} - 5-levels Intermediate Atmospheric physics, Global configuration on cube sphere grid (32x32 grid points per face, roughly 2.8^circ).\ Also contains an additional set-up with a slab-ocean and thermodynamic sea-ice ({it input.thSI/}).

item texttt{bottom_ctrl_5x5} - Adjoint test using the bottom topography as the control parameter.

item texttt{cfc_example} - Global ocean with online computation and advection of CFC11 and CFC12.

item texttt{cheapAML_box} - Example using cheap atmospheric mixed layer (cheapAML) package.

item texttt{cpl_aim+ocn} - Coupled Ocean - Atmosphere realistic configuration on cubed-sphere cs32 horizontal grid, using Intermediate Atmospheric physics ({it pkg/aim_v23}) thermodynamic seaice ({it pkg/thsice}) and land packages. on cubed-sphere cs32 in a realistic configuration.

item texttt{cpl_atm2d+ocn} - Coupled Ocean - Atmosphere realistic configuration using 2-D Atmospheric Model ({it pkg/atm2d}).

item texttt{deep_anelastic} - Convection simulation on a giant planet: relax both the Boussinesq approximation (anelastic) and the thin atmosphere approximation (deep atmosphere).

item texttt{dome} - Idealized 3D test of a density-driven bottom current.

item texttt{exp2} - Old version of the global ocean experiment (no GM,

no partial-cells).\

Also contains an additional set-up with rigid-lid ({it input.rigidLid/}).

item texttt{exp4} - Flow over a Gaussian bump in open-water or channel with open boundaries.\ Also contains an additional set-up using non-linear free-surface ({it input.nlfs/}).

item texttt{fizhi-cs-32x32x40} - Global atmospheric simulation with realistic topography, 40 vertical levels, a cubed sphere grid and the full atmospheric physics package.

item texttt{fizhi-cs-aqualev20} - Global atmospheric simulation on an aqua planet with full atmospheric physics. Run is perpetual march with an analytical SST distribution. This is the configuration for the APE (Aqua Planet Experiment) participation experiment.

item texttt{fizhi-gridalt-hs} - Global atmospheric simulation Held-Suarez (1994) forcing, with the physical forcing and the dynamical forcing running on different vertical grids.

item texttt{flt_example} - Example of using float package.

item texttt{front_relax} - Relaxation of an ocean thermal front (test for Gent/McWilliams scheme). 2D (y-z).\ Also contains additional set-ups: begin{enumerate}

item using the Boundary-Value Problem method (Ferrari et al., 2010) ({it input.bvp/}).

item with Mixed-Layer Eddy parameterization (Ferrari & McWilliams, 2007) ({it input.mxl/}).

end{enumerate}

item texttt{global_ocean.90x40x15} - Global ocean simulation at 4x4 degree resolution. Similar to tutorial_global_oce_latlon, but using z^* coordinates with quasi-non-hydrostatic and non-hydrostatic metric terms. This experiment also illustrate the use of SBO package. Also contains additional set-ups: begin{enumerate}

item using down-slope package ({it pkg/down_slope}) ({it input.dwnslp/}) item an Open-AD adjoint set-up ({it code_oad/, input_oad/}). item four TAF adjoint set-ups ({it code_ad/}): begin{enumerate}

item standard experiment ({it input_ad/}). item with bottom drag as a control ({it input_ad.bottomdrag/}). item with kappa GM as a control ({it input_ad.kapgm/}). item with kappa Redi as a control ({it input_ad.kapredi/}).

end{enumerate}

end{enumerate}

item texttt{global_ocean.cs32x15} - Global ocean experiment on the cubed sphere grid.\ Also contains additional forward set-ups: begin{enumerate}

item non-hydrostatic with biharmonic viscosity ({it input.viscA4/}) item using thermodynamic sea ice and bulk force ({it input.thsice/}) item using thermodynamic ({it pkg/thsice}) dynamic ({it pkg/seaice}) sea-ice

and {it exf} package ({it input.icedyn/})

item using thermodynamic - dynamic ({it pkg/seaice}) sea-ice with {it exf} package ({it input.seaice/})

end{enumerate} and few additional adjoint set-ups ({it code_ad/}): begin{enumerate}

item standard experiment without sea-ice ({it input_ad/}). item using thermodynamic - dynamic sea-ice ({it input_ad.seaice/}) item same as above without adjoint sea-ice dynamics ({it input_ad.seaice_dynmix/}) item using thermodynamic sea-ice from {it thsice} package ({it input_ad.thsice/})

end{enumerate}

item texttt{global_ocean_ebm} - Global ocean experiment on a lat-lon grid coupled to an atmospheric energy balance model. Similar to global_ocean.90x40x15 experiment.\ Also contains an adjoint set-up ({it code_ad/, input_ad/}).

item texttt{global_with_exf} - Global ocean experiment on a lat-lon grid using the {it exf} package. Similar to tutorial_global_oce_latlon experiment.\ Also contains a secondary set-up with yearly {it exf} fields ({it input_ad.yearly/}).

item texttt{halfpipe_streamice} - Example using package "streamice".

Also contains adjoint set-ups using TAF ({it code_ad/, input_ad/}) and using Open-AD ({it code_oad/, input_oad/}).

item texttt{hs94.128x64x5} - 3D atmosphere dynamics on lat-lon grid, using Held and Suarez '94 forcing.

item texttt{hs94.1x64x5} - Zonal averaged atmosphere dynamics using Held and Suarez '94 forcing.\ Also contains adjoint set-ups using TAF ({it code_ad/, input_ad/})

and using Open-AD ({it code_oad/, input_oad/}).

item texttt{hs94.cs-32x32x5} - 3D atmosphere dynamics using Held and Suarez (1994) forcing on the cubed sphere, similar to tutorial_held_suarez_cs experiment but using linear free-surface and only 5 levels.\ Also contains an additional set-up with Implicit Internal gravity waves treatment and Adams-Bashforth 3 ({it input.impIGW/}).

item texttt{ideal_2D_oce} - Idealized 2D global ocean simulation on an aqua planet.

item texttt{internal_wave} - Ocean internal wave forced by open boundary conditions.\ Also contains an additional set-up using {it pkg/kl10} (see section

ref{sec:pkg:kl10}, Klymak and Legg, 2010) ({it input.kl10/}).

item texttt{inverted_barometer} - Simple test of ocean response to atmospheric pressure loading.

item texttt{isomip} - ISOMIP like set-up including ice-shelf cavities ({it pkg/shelfice}).\ Also contains additional set-ups: begin{enumerate}

item with "htd" ({it input.htd/}) but only Martin knows what "htd" stands for.

item using package {it icefront} ({it input.icefront})

end{enumerate} and also adjoint set-ups using TAF ({it code_ad/, input_ad/, input_ad.htd/}) or using Open-AD ({it code_oad/, input_oad/}).

item texttt{lab_sea} - Regional Labrador Sea simulation on a lat-lon grid using the sea ice package.\ Also contains additional set-ups: begin{enumerate}

item using the simple "free-drift" assumption for seaice ({it input.fd/}) item using EVP dynamics (instead of LSR solver) and Hibler & Bryan (1987)

sea-ice ocean stress ({it input.hb87/})

item using package {it salt_plume} ({it input.salt_plume/})

end{enumerate} and also 3 adjoint set-ups ({it code_ad/, input_ad/, input_ad.noseaicedyn/,

input_ad.noseaice/}).

item texttt{matrix_example} - Test of experimental method to accelerated convergence towards equilibrium.

item texttt{MLAdjust} - Simple tests for different viscosity formulations. Also contains additional set-ups (see: {it verification/MLAdjust/README}): begin{enumerate}

item ({it input.A4FlxF/}) item ({it input.AhFlxF/}) item ({it input.AhVrDv/}) item ({it input.AhStTn/})

end{enumerate}

item texttt{natl_box} - Eastern subtropical North Atlantic with KPP scheme; 1 month integration

item texttt{obcs_ctrl} - Adjoint test using Open-Boundary conditions as control parameters.

item texttt{offline_exf_seaice} - Seaice on top of oceanic surface layer in an idealized channel. Forcing is computed by bulk-formulae ({it pkg/exf}) with temperature relaxation to prescribed SST (offline ocean).\ Also contains additional set-ups: begin{enumerate}

item sea-ice dynamics-only using JFNK solver and {it pkg/thsice} advection ({it input.dyn_jfnk/})

item sea-ice dynamics-only using LSR solver and {it pkg/seaice} advection ({it input.dyn_lsr/})

item sea-ice thermodynamics-only using {it pkg/seaice} ({it input.thermo/}) item sea-ice thermodynamics-only using {it pkg/thsice} ({it input.thsice/})

end{enumerate} and also 2 adjoint set-ups ({it code_ad/, input_ad/, input_ad.thsice/}).

item texttt{OpenAD} - Simple Adjoint experiment (used also to test Open-AD compiler)

item texttt{rotating_tank} - Rotating tank simulation in cylindrical coordinates. This experiment is described in detail in section ref{sec:eg-tank}.

item texttt{seaice_itd} - Seaice example using Ice Thickness Distribution (ITD). Also contains additional set-ups: begin{enumerate}

item ({it input.thermo/}) item ({it input.lipscomb07/})

end{enumerate}

item texttt{seaice_obcs} - Similar to "lab_sea" ({it input.salt_plume/}) experiment with only a fraction of the domain and open-boundary conditions derived from "lab_sea" experiment.\ Also contains additional set-ups: begin{enumerate}

item ({it input.seaiceSponge/}) item ({it input.tides/})

end{enumerate}

item texttt{short_surf_wave} - Short surface wave adjusment (non-hydrostatic) in homogeneous 2-D vertical section (x-z).

item texttt{so_box_biogeo} - Open-boundary Southern ocean box around

Drake passage, using same model parameters and forcing as experiment "tutorial_global_oce_biogeo" from which initial conditions

and OB conditions have been extracted.

item texttt{solid-body.cs-32x32x1} - Solid body rotation test for cube sphere grid.

item texttt{tidal_basin_2d} - 2-D vertical section (x-z) with tidal forcing (untested)

item texttt{vermix} - Simple test in a small domain (3 columns) for ocean vertical mixing schemes. The standard set-up ({it input/}) uses KPP scheme cite[]{lar-eta:94}.\ Also contains additional set-ups: begin{enumerate}

item with Double Diffusion scheme from KPP ({it input.dd/}) item with cite{gas-eta:90} ({it pkg/ggl90}) scheme ({it input.ggl90/}) item with cite{Mellor:Yamada1982} level 2. ({it pkg/my82}) scheme ({it input.my82/}) item with cite{pal-rom:97} ({it pkg/opps}) scheme ({it input.opps/}) item with cite{Pacanowski:Philander1981} ({it pkg/pp81}) scheme ({it input.pp81/})

end{enumerate}

end{enumerate}

subsection{Directory structure of model examples}

Each example directory has the following subdirectories:

begin{itemize} item texttt{code}: contains the code particular to the example. At a

minimum, this directory includes the following files:

begin{itemize} item texttt{code/packages.conf}: declares the list of packages or

package groups to be used. If not included, the default version is located in texttt{pkg/pkg_default}. Package groups are simply convenient collections of commonly used packages which are defined in texttt{pkg/pkg_default}. Some packages may require other packages or may require their absence (that is, they are incompatible) and these package dependencies are listed in texttt{pkg/pkg_depend}.

item texttt{code/CPP_EEOPTIONS.h}: declares CPP keys relative to the ‘‘execution environment” part of the code. The default version is located in texttt{eesupp/inc}.

item texttt{code/CPP_OPTIONS.h}: declares CPP keys relative to the ‘‘numerical model” part of the code. The default version is located in texttt{model/inc}.

item texttt{code/SIZE.h}: declares size of underlying computational grid. The default version is located in texttt{model/inc}.

end{itemize}

In addition, other include files and subroutines might be present in texttt{code} depending on the particular experiment. See Section 2 for more details.

item texttt{input}: contains the input data files required to run the example. At a minimum, the texttt{input} directory contains the following files:

begin{itemize} item texttt{input/data}: this file, written as a namelist,

specifies the main parameters for the experiment.

item texttt{input/data.pkg}: contains parameters relative to the packages used in the experiment.

item texttt{input/eedata}: this file contains ‘‘execution environment” data. At present, this consists of a specification of the number of threads to use in X and Y under multi-threaded execution.

end{itemize}

In addition, you will also find in this directory the forcing and topography files as well as the files describing the initial state of the experiment. This varies from experiment to experiment. See the verification directories refered to in this chapter for more details.

item texttt{results}: this directory contains the output file texttt{output.txt} produced by the simulation example. This file is useful for comparison with your own output when you run the experiment.

item texttt{build}: this directory is initially empty and is used to compile and load the model, and to generate the executable.

item texttt{run}: this directory is initially empty and is used to run the executable.

end{itemize}

Once you have chosen the example you want to run, you are ready to compile the code.

newpage

4.1 Barotropic Gyre MITgcm Example

begin{center} (in directory: {it verification/tutorial_barotropic_gyre/}) end{center}

This example experiment demonstrates using the MITgcm to simulate a Barotropic, wind-forced, ocean gyre circulation. The files for this experiment can be found in the verification directory tutorial_barotropic_gyre. The experiment

is a numerical rendition of the gyre circulation problem similar to the problems described analytically by Stommel in 1966 cite{Stommel66} and numerically in Holland et. al cite{Holland75}.

In this experiment the model is configured to represent a rectangular enclosed box of fluid, $1200 times 1200 $~km in lateral extent. The fluid is 5~km deep and is forced by a constant in time zonal wind stress, tau_x, that varies sinusoidally in the ‘‘north-south” direction. Topologically the grid is Cartesian and the coriolis parameter f is defined according to a mid-latitude beta-plane equation

$$f(y) = f_0 + \beta y \tag{4.1}$$

where y is the distance along the ‘‘north-south” axis of the simulated domain. For this experiment f_{0} is set to $10^{-4}s^{-1}$ in (4.1) and $beta = 10^{-11}s^{-1}m^{-1}$.

The sinusoidal wind-stress variations are defined according to

$$\tau_x(y) = \tau_0 \sin(\pi \frac{y}{L_y}) \tag{4.2}$$

where L_y is the lateral domain extent (1200~km) and τ_0 is set to $0.1Nm^{-2}$.

Figure 4.1 summarizes the configuration simulated.

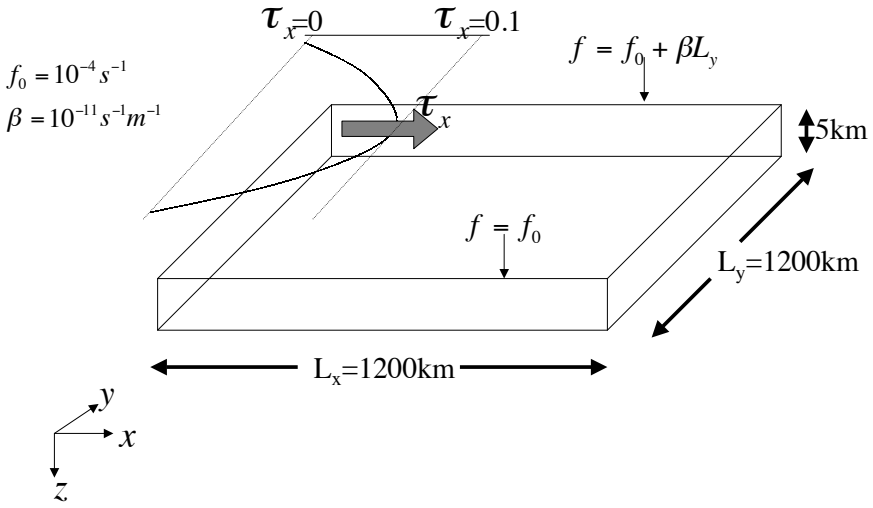

Figure 4.1: Schematic of simulation domain and wind-stress forcing function for barotropic gyre numerical experiment. The domain is enclosed bu solid walls at $x=$~0,1200km and at $y=$~0,1200km.

4.1.1 Equations Solved

The model is configured in hydrostatic form. The implicit free surface form of the pressure equation described in Marshall et. al cite{marshall:97a} is employed. A horizontal Laplacian operator $nabla_{h}^2$ provides viscous dissipation. The wind-stress momentum input is added to the momentum equation for the ‘‘zonal flow’‘, u. Other terms in the model are explicitly switched off for this experiment configuration (see section ref{sec:eg-baro-code_config}), yielding an active set of equations solved in this configuration as follows

begin{eqnarray} label{eq:eg-baro-model_equations} frac{Du}{Dt} - fv +

gfrac{partial eta}{partial x} - A_{h}nabla_{h}^2u

& = & frac{tau_{x}}{rho_{0}Delta z} \ frac{Dv}{Dt} + fu + gfrac{partial eta}{partial y} -

A_{h}nabla_{h}^2v

& = & 0 \ frac{partial eta}{partial t} + nabla_{h}cdot vec{u} &=& 0 end{eqnarray}

noindent where u and v and the x and y components of the flow vector $vec{u}$. \

subsection{Discrete Numerical Configuration} %label{www:tutorials}

The domain is discretised with

a uniform grid spacing in the horizontal set to $Delta x=Delta y=20$~km, so

that there are sixty grid cells in the x and y directions. Vertically the model is configured with a single layer with depth, $Delta z$, of 5000~m.

subsubsection{Numerical Stability Criteria} %label{www:tutorials}

The Laplacian dissipation coefficient, A_{h}, is set to $400 m s^{-1}$. This value is chosen to yield a Munk layer width cite{adcroft:95},

begin{eqnarray} label{eq:eg-baro-munk_layer} M_{w} = pi (frac { A_{h} }{ beta })^{frac{1}{3}} end{eqnarray}

noindent of $approx 100$km. This is greater than the model resolution $Delta x$, ensuring that the frictional boundary layer is well resolved. \

noindent The model is stepped forward with a time step $delta t=1200$secs. With this time step the stability parameter to the horizontal Laplacian friction cite{adcroft:95}

begin{eqnarray} label{eq:eg-baro-laplacian_stability} S_{l} = 4 frac{A_{h} delta t}{{Delta x}^2} end{eqnarray}

noindent evaluates to 0.012, which is well below the 0.3 upper limit for stability. \

noindent The numerical stability for inertial oscillations cite{adcroft:95}

begin{eqnarray} label{eq:eg-baro-inertial_stability} S_{i} = f^{2} {delta t}^2 end{eqnarray}

noindent evaluates to 0.0144, which is well below the 0.5 upper limit for stability. \

noindent The advective CFL cite{adcroft:95} for an extreme maximum horizontal flow speed of $ | vec{u} | = 2 ms^{-1}$

begin{eqnarray} label{eq:eg-baro-cfl_stability} S_{a} = frac{| vec{u} | delta t}{ Delta x} end{eqnarray}

noindent evaluates to 0.12. This is approaching the stability limit of 0.5 and limits $delta t$ to $1200s$.

4.1.2 Code Configuration

The model configuration for this experiment resides under the directory `verification/tutorial_barotropic_gyre/`.

The experiment files

- `input/data`
- `input/data.pkg`
- `input/eedata`
- `input/windx.sin_y`
- `input/topog.box`
- `code/CPP_EEOPTIONS.h`

- code/CPP_OPTIONS.h
- code/SIZE.h

contain the code customizations and parameter settings for this experiments. Below we describe the customizations to these files associated with this experiment.

subsubsection{File {it input/data}} %label{www:tutorials}

This file, reproduced completely below, specifies the main parameters for the experiment. The parameters that are significant for this configuration are

begin{itemize}

item Line 7, begin{verbatim} viscAh=4.E2, end{verbatim} this line sets the Laplacian friction coefficient to $400 m^2s^{-1}$ item Line 10, begin{verbatim} beta=1.E-11, end{verbatim} this line sets $beta$ (the gradient of the coriolis parameter, f) to $10^{-11} s^{-1}m^{-1}$

item Lines 15 and 16 begin{verbatim} rigidLid=.FALSE., implicitFreeSurface=.TRUE., end{verbatim} these lines suppress the rigid lid formulation of the surface pressure inverter and activate the implicit free surface form of the pressure inverter.

item Line 27, begin{verbatim} startTime=0, end{verbatim} this line indicates that the experiment should start from $t=0$ and implicitly suppresses searching for checkpoint files associated with restarting an numerical integration from a previously saved state.

item Line 29, begin{verbatim} endTime=12000, end{verbatim} this line indicates that the experiment should start finish at $t=12000s$. A restart file will be written at this time that will enable the simulation to be continued from this point.

item Line 30, begin{verbatim} deltaTmom=1200, end{verbatim} This line sets the momentum equation timestep to $1200s$.

item Line 39, begin{verbatim} usingCartesianGrid=.TRUE., end{verbatim} This line requests that the simulation be performed in a Cartesian coordinate system.

item Line 41, begin{verbatim} delX=60*20E3, end{verbatim} This line sets the horizontal grid spacing between each x-coordinate line in the discrete grid. The syntax indicates that the discrete grid should be comprise of 60 grid lines each separated by $20 times 10^{3}m$ (20~km).

item Line 42, begin{verbatim} delY=60*20E3, end{verbatim} This line sets the horizontal grid spacing between each y-coordinate line in the discrete grid to $20 times 10^{3}m$ (20~km).

item Line 43, begin{verbatim} delZ=5000, end{verbatim} This line sets the vertical grid spacing between each z-coordinate line in the discrete grid to $5000m$ (5~km).

item Line 46, begin{verbatim} bathyFile='topog.box' end{verbatim} This line specifies the name of the file from which the domain bathymetry is read. This file is a two-dimensional (x,y) map of depths. This file is assumed to contain 64-bit binary numbers giving the depth of the model at each grid cell, ordered with the x coordinate varying fastest. The points are ordered from low coordinate to high coordinate for both axes. The units and orientation of the depths in this file are the same as used in the MITgcm code. In this experiment, a depth of $0m$ indicates a solid wall and a depth of $-5000m$ indicates open ocean. The matlab program {it input/gendata.m} shows an example of how to generate a bathymetry file.

item Line 49, begin{verbatim} zonalWindFile='windx.sin_y' end{verbatim} This line specifies the name of the file from which the x-direction surface wind stress is read. This file is also a two-dimensional (x,y) map and is enumerated and formatted in the same manner as the bathymetry file. The matlab program {it input/gendata.m} includes example code to generate a valid {bf zonalWindFile} file.

end{itemize}

noindent other lines in the file {it input/data} are standard values that are described in the MITgcm Getting Started and MITgcm Parameters notes.

```
# Model parameters
# Continuous equation parameters
 &PARM01
 tRef=20.,
 sRef=10.,
 viscAz=1.E-2,
 viscAh=4.E2,
 diffKhT=4.E2,
 diffKzT=1.E-2,
 beta=1.E-11,
 tAlpha=2.E-4,
 sBeta =0.,
 gravity=9.81,
 gBaro=9.81,
 rigidLid=.FALSE.,
 implicitFreeSurface=.TRUE.,
 eosType='LINEAR',
 readBinaryPrec=64,
 &

# Elliptic solver parameters
 &PARM02
 cg2dMaxIters=1000,
 cg2dTargetResidual=1.E-7,
 &

# Time stepping parameters
 &PARM03
 startTime=0,
#endTime=311040000,
 endTime=12000.0,
 deltaTmom=1200.0,
 deltaTtracer=1200.0,
 abEps=0.1,
 pChkptFreq=2592000.0,
 chkptFreq=120000.0,
 dumpFreq=2592000.0,
 monitorSelect=2,
 monitorFreq=1.,
 &

# Gridding parameters
 &PARM04
 usingCartesianGrid=.TRUE.,
 usingSphericalPolarGrid=.FALSE.,
 delX=60*20E3,
 delY=60*20E3,
 delZ=5000.,
 &

# Input datasets
 &PARM05
 bathyFile='topog.box',
 hydrogThetaFile=,
 hydrogSaltFile=,
 zonalWindFile='windx.sin_y',
 meridWindFile=,
 &
```

begin{small} input{s_examples/barotropic_gyre/input/data} end{small}

subsubsection{File {it input/data.pkg}} %label{www:tutorials}

This file uses standard default values and does not contain customizations for this experiment.

subsubsection{File {it input/eedata}} %label{www:tutorials}

This file uses standard default values and does not contain customizations for this experiment.

subsubsection{File {it input/windx.sin_y}} %label{www:tutorials}

The {it input/windx.sin_y} file specifies a two-dimensional (x,y) map of wind stress ,tau_{x}, values. The units used are Nm^{-2}. Although tau_{x} is only a function of yn in this experiment this file must still define a complete two-dimensional map in order to be compatible with the standard code for loading forcing fields in MITgcm. The included matlab program {it input/gendata.m} gives a complete code for creating the {it input/windx.sin_y} file.

subsubsection{File {it input/topog.box}} %label{www:tutorials}

The {it input/topog.box} file specifies a two-dimensional (x,y) map of depth values. For this experiment values are either $0m$ or {bf -delZ}m, corresponding respectively to a wall or to deep ocean. The file contains a raw binary stream of data that is enumerated in the same way as standard MITgcm two-dimensional, horizontal arrays. The included matlab program {it input/gendata.m} gives a complete code for creating the {it input/topog.box} file.

subsubsection{File {it code/SIZE.h}} %label{www:tutorials}

Two lines are customized in this file for the current experiment

begin{itemize}

item Line 39, begin{verbatim} sNx=60, end{verbatim} this line sets the lateral domain extent in grid points for the axis aligned with the x-coordinate.

item Line 40, begin{verbatim} sNy=60, end{verbatim} this line sets the lateral domain extent in grid points for the axis aligned with the y-coordinate.

end{itemize}

begin{small} input{s_examples/barotropic_gyre/code/SIZE.h} end{small}

subsubsection{File {it code/CPP_OPTIONS.h}} %label{www:tutorials}

This file uses standard default values and does not contain customizations for this experiment.

subsubsection{File {it code/CPP_EEOPTIONS.h}} %label{www:tutorials}

This file uses standard default values and does not contain customizations for this experiment.

newpage input{s_examples/baroclinic_gyre/fourlayer.tex}

newpage input{s_examples/advection_in_gyre/adv_gyre.tex}

newpage input{s_examples/global_oce_latlon/climatalogical_ogcm.tex}

newpage input{s_examples/global_oce_in_p/ogcm_in_pressure.tex}

newpage input{s_examples/held_suarez_cs/held_suarez_cs.tex}

newpage input{s_examples/deep_convection/convection.tex}

newpage input{s_examples/plume_on_slope/plume_on_slope.tex}

newpage input{s_examples/global_oce_biogeo/biogeochem.tex}

newpage input{s_examples/global_oce_optim/global_oce_estimation.tex}

newpage input{s_examples/tracer_adjsens/doc_ad_examples.tex}

newpage input{s_examples/cfc_offline/offline_tutorial.tex}

newpage input{s_examples/rotating_tank/tank.tex}

BIBLIOGRAPHY

[AC04] A. Adcroft and J.-M. Campin. Re-scaled height coordinates for accurate representation of free-surface flows in ocean circulation models. *Ocean Modelling*, 7:269–284, 2004. doi:10.1016/j.ocemod.2003.09.003.

[ACHM04] A. Adcroft, J.-M. Campin, C. Hill, and J. Marshall. Implementation of an atmosphere-ocean general circulation model on the expanded spherical cube. *Mon.~Wea.~Rev.*, 132:2845–2863, 2004. URL: http://mitgcm.org/pdfs/mwr_2004.pdf, doi:10.1175/MWR2823.1.

[AHC+04] A. Adcroft, C. Hill, J.-M. Campin, J. Marshall, and P. Heimbach. Overview of the formulation and numerics of the MITgcm. In *Proceedings of the ECMWF seminar series on Numerical Methods, Recent developments in numerical methods for atmosphere and ocean modelling*, 139–149. ECMWF, 2004. URL: http://mitgcm.org/pdfs/ECMWF2004-Adcroft.pdf.

[AHM97] A.J. Adcroft, C.N. Hill, and J. Marshall. Representation of topography by shaved cells in a height coordinate ocean model. *Mon.~Wea.~Rev.*, 125:2293–2315, 1997. URL: http://mitgcm.org/pdfs/mwr_1997.pdf, doi:10.1175/1520-0493%281997%29125<2293:ROTBSC>2.0.CO;2.

[AM99] Hill C. Adcroft, A. and J. Marshall. A new treatment of the coriolis terms in c-grid models at both high and low resolutions. *Mon.~Wea.~Rev.*, 127:1928–1936, 1999. URL: http://mitgcm.org/pdfs/mwr_1999.pdf, doi:10.1175/1520-0493%281999%29127<1928:ANTOTC>2.0.CO;2.

[CHM99] Daniel Jamous Chris Hill, Alistair Adcroft and John Marshall. A strategy for terascale climate modeling. In *In Proceedings of the Eighth ECMWF Workshop on the Use of Parallel Processors in Meteorology*, 406–425. World Scientific, 1999.

[HM95] C. Hill and J. Marshall. Application of a parallel navier-stokes model to ocean circulation in parallel computational fluid dynamics. In N. Satofuka A. Ecer, J. Periaux and S. Taylor, editors, *Implementations and Results Using Parallel Computers*, pages 545–552. Elsevier Science B.V.: New York, 1995.

[MGZ+99] J. Marotzke, R. Giering, K.Q. Zhang, D. Stammer, C. Hill, and T. Lee. Construction of the adjoint mit ocean general circulation model and application to atlantic heat transport variability. *J.~Geophys.~Res.*, 104, C12:29,529–29,547, 1999.

[MAC+04] J. Marshall, A. Adcroft, J.-M. Campin, C. Hill, and A. White. Atmosphere-ocean modeling exploiting fluid isomorphisms. *Mon.~Wea.~Rev.*, 132:2882–2894, 2004. URL: http://mitgcm.org/pdfs/a_o_iso.pdf, doi:10.1175/MWR2835.1.

[MAH+97] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey. A finite-volume, incompressible navier stokes model for studies of the ocean on parallel computers. *J.~Geophys.~Res.*, 102(C3):5753–5766, 1997. URL: http://mitgcm.org/pdfs/96JC02776.pdf.

[MHPA97] J. Marshall, C. Hill, L. Perelman, and A. Adcroft. Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *J.~Geophys.~Res.*, 102(C3):5733–5752, 1997. URL: http://mitgcm.org/pdfs/96JC02775.pdf.

[MJH98] J. Marshall, H. Jones, and C. Hill. Efficient ocean modeling using non-hydrostatic algorithms. *J.~Mar.~Sys.*, 18:115–134, 1998. URL: http://mitgcm.org/pdfs/journal_of_marine_systems_1998.pdf, doi:10.1016/S0924-7963%2898%2900008-6.