Stokes Drift and Meshless Wave Modeling

by

Adrean A. Webb

B.S., University of Oklahoma, 1998

M.S., University of New Hampshire, 2007

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2013

This thesis entitled:
Stokes Drift and Meshless Wave Modeling
written by Adrean A. Webb
has been approved for the Department of Applied Mathematics

Baylor Fox-Kemper

Keith Julien

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Webb, Adrean A. (Ph.D., Applied Mathematics)

Stokes Drift and Meshless Wave Modeling

Thesis directed by Assistant Professor Baylor Fox-Kemper

This dissertation is loosely organized around efforts to improve vertical ocean mixing in global climate models and includes an in-depth analysis of Stokes drift, optimization of a new global climate model wave component, and development of a meshless spectral wave model.

Stokes drift (hereafter SD) is an important vector component that appears often in wave-averaged dynamics. Mathematically, SD is the mean difference between Eulerian and Lagrangian velocities and intuitively can be thought of as the near-surface ocean current induced from wave motion. Increasingly, spectral wave models are being used to calculate SD globally. These models solve a 5D wave action balance equation and typically require large computational resources to make short to medium-range forecasts of the sea state.

In the first part, a hierarchy of SD approximations are investigated and new approximations that remove systematic biases are derived. A new 1D spectral approximation is used to study the effects of multidirectional waves and directional wave spreading on SD. It is shown that these effects are largely uncorrelated and affect both the magnitude and direction of SD in a nonlinear fashion that is sensitive with depth.

In the second part, efforts to add a wave model component to the NCAR Community Earth System Model are discussed. This coupled component will serve as the backbone to a new Langmuir mixing parameterization and uses a modified version of NOAA WAVEWATCH III (a third-generation spectral wave model). In addition, the governing wave action balance equation is reviewed and several variations are derived and formulated.

In the third part, construction of a monochromatic spectral wave model using RBF-generated finite differences is described. Several numerical test cases are conducted to measure performance and guide further development. In kinematic comparisons with WAVEWATCH III, the meshless prototype is approximately 70–210 times more accurate and uses a factor of 12 to 17 less unknowns.

Dedication

I dedicate this to my family and closest friends. This could not have been completed without their constant encouragement and support.

Acknowledgements

This work was funded by NASA ROSES Physical Oceanography NNX09AF38G. I thank the agency for its support.

In addition, I would like to thank Keith Julien for looking out for me during my stay at the Institute for Pure and Applied Mathematics and for his general support within the Applied Mathematics Department.

I would also like to thank Bengt Fornberg and Natasha Flyer for allowing me to crash their weekly RBF meetings and glean nuggets of useful information. In particular, I would like to thank Natasha for answering my endless questions and helping me set up the meshless wave model.

And finally, I would like to thank my advisor Baylor Fox-Kemper, who in all his wisdom, had me start writing early. I couldn't have picked a better advisor and I am grateful for all his efforts in helping me become a better scholar.

Contents

Chapter

Appendix

Tables

Table

Figures

Figure

Mathematical Notation

Generic vector space:

$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_d)$	d-dimensional vector.
$\partial_{\alpha_i} = \frac{\partial}{\partial \alpha_i}$	Partial derivative with respect to α_i.
$\nabla_{\boldsymbol{\alpha}} = (\,\partial_{\alpha_1},\, \partial_{\alpha_2}, \dots,\, \partial_{\alpha_d})$	d-dimensional gradient.
$C^r(\mathbb{R}^d)$	Space of r-times continuously differentiable scalar functions in a d-dimensional domain.

Cartesian vector space:

$\boldsymbol{x} = (x, y, z)$	3D spatial vector.
$\boldsymbol{x}_h = (x, y)$	2D spatial vector (horizontal).
$\mathbf{k} = (k_x, k_y)$	2D spectral vector (wavevector).
$\nabla = \nabla_{\boldsymbol{x}} = \nabla_{c,\boldsymbol{x}} = (\,\partial_x,\, \partial_y,\, \partial_z)$	3D spatial gradient.
$\nabla_h = \nabla_{\boldsymbol{x}_h} = \nabla_{c,\boldsymbol{x}_h} = (\,\partial_x,\, \partial_y)$	2D spatial gradient (horizontal).
$\nabla_{\mathbf{k}} = \nabla_{c,\mathbf{k}} = \left(\partial_{k_x},\, \partial_{k_y}\right)$	2D spectral gradient (wavevector).
$D_t = \partial_t + \boldsymbol{u} \cdot \nabla$	Material derivative.

Cartesian scalar and vector functions:

$\boldsymbol{u} = \boldsymbol{u}^E = (u_x, u_y, u_z)$	Eulerian velocity.
$\boldsymbol{u}^L = \left(u_x^L, u_y^L, u_z^L\right)$	Lagrangian velocity.
$\varphi = \varphi(\boldsymbol{x}, t)$	Eulerian velocity potential.
$\eta = \eta(\boldsymbol{x}_h, t)$	Ocean surface height perturbation.

- $H = H(\boldsymbol{x}_h)$ Ocean depth.

Cartesian miscellaneous:

- $\mathbf{g} = (0, 0, -\mathrm{g})$ Gravitational acceleration vector for standard gravity g.
- $k = |\mathbf{k}|$ Wavenumber.

In chapter 3, new notation is added and modified to distinguish between analytical and discretized equations.

Modifications:

- $\vec{\alpha} = (\alpha^1, \alpha^2, \ldots, \alpha^d)$ d-dimensional vector.

Discretized vector space:

- a_i ith sampled value.
- $\underline{a} = \underline{a_i} = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix}$ Column vector of size N.
- $\underline{\underline{a}} = \underline{\underline{a_{ij}}} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix}$ Matrix of size $M \times N$ where the subscripts refer to the element in the ith row and jth column.

Terminology

Abbreviations:

LM	Vertical mixing caused by Langmuir circulation and turbulence.
SD	Stokes drift velocity; The mean Eulerian and Lagrangian wave velocity difference.
RBF	Radial basis functions.
RBF-FD	RBF-generated finite differences.
$1\mathrm{D}_h$, $2\mathrm{D}_h$	Horizontally one-, two-dimensional.
$2\mathrm{D}_h$-SD	Horizontally two-dimensional SD approximation; Uses 2D wave spectra.
$1\mathrm{D}_h$-SD	Horizontally one-dimensional SD approximation; Uses 1D wave spectra.
$1\mathrm{D}_h$-DHH-SD	Horizontally one-dimensional SD approximation; Uses 1D wave spectra with the DHH directional distribution to correct for wave spreading.

Chapter 1

Introduction

1.1 Framing the research

This research grew out of a small project to roughly estimate the effects of Langmuir mixing, small wind and wave-driven vertical mixing in the near-surface ocean, in global climate models. Results from a preliminary parametrization demonstrated a need to improve approximations of Stokes drift, near-surface wave-induced currents, and use a prognostic wave field for future parameterizations. Since then, much work has been done. Analysis of Stokes drift has led to better understanding and a hierarchy of approximations. In addition, a meshless prototype using RBF-generated finite differences has been built and shows potential as a viable approach to future spectral wave modeling.

In this chapter, a short background of Langmuir mixing is given and early efforts to estimate its effect in global climate models is discussed. In addition, a review of linear wave theory is presented. The rest of the chapters focus on Stokes drift, spectral wave modeling, and the meshless prototype. For convenience, the preamble contains a description of the mathematical notation used and a brief glossary of common terms and abbreviations.

1.1.1 Motivation for initial research

The oceans play a dominant role in regulating Earth's climate through mechanisms such as heat transport from the equator to the poles and storage of greenhouses gasses. The air-sea interface, or the ocean surface, is a particularly important region as it filters the exchanges of momentum, energy, and gasses (Kump et al., 2004). These exchanges are dependent upon the sea surface state as well as the depth and properties of the surface mixed layer, a homogeneously mixed layer that contains the photic zone where phytoplankton grow (Segar, 2007). The physics (submesoscale and smaller) that create and preserve this environment are unresolved in climate models; therefore it is important to accurately model and parameterize this turbulent region for climate predictions.

Traditionally, the near-surface mixing schemes used in global climate models (GCMs) have focused on convective and shear-driven turbulence. However, surface wave breaking and Langmuir mixing (LM), a type of mixing due to the interaction of wind and waves, also play a role (Tseng and D'Asaro, 2004). Up until now, these latter effects have been included only indirectly through tuned parameterizations that do not use explicit wave information (Wang et al., 1998; Large et al., 1994). The majority of this thesis has been motivated by the need to implement better parameterizations.

1.1.2 Preliminary inclusion of Langmuir mixing

Langmuir cells are small overturning cells (10–100 m wide and 1–10 km long) that form in the near–surface ocean when wind and waves are moving approximately in the same direction (Smith, 2001). Depending on the speed of the wind and waves, these cells can increase greatly the amount of mixing in the mixed layer (Tseng and D'Asaro, 2004; Belcher et al., 2012). Observations indicate that even when these cells are not obvious, Langmuir turbulence – a disordered jumble of Langmuir cells – can lead to near-surface turbulent kinetic energy double what is expected without it (D'Asaro, 2001). Here, LM refers to vertical mixing caused by either Langmuir cells or turbulence. See Figs. 1.1 and 1.2 for images and observations of LM.

Analytical and numerical modeling of LM is a broad and active area of research and is only discussed briefly here (Sullivan and McWilliams, 2010). The broadly accepted theory behind LM is a set of surface-wave filtered Navier-Stokes equations called the Craik-Leibovich equations (Chini, 2008). In the simplified case of wind-wave alignment, much work has been done to simplify the Craik-Leibovich equations (Chini et al., 2009) and develop vertical mixing schemes (McWilliams

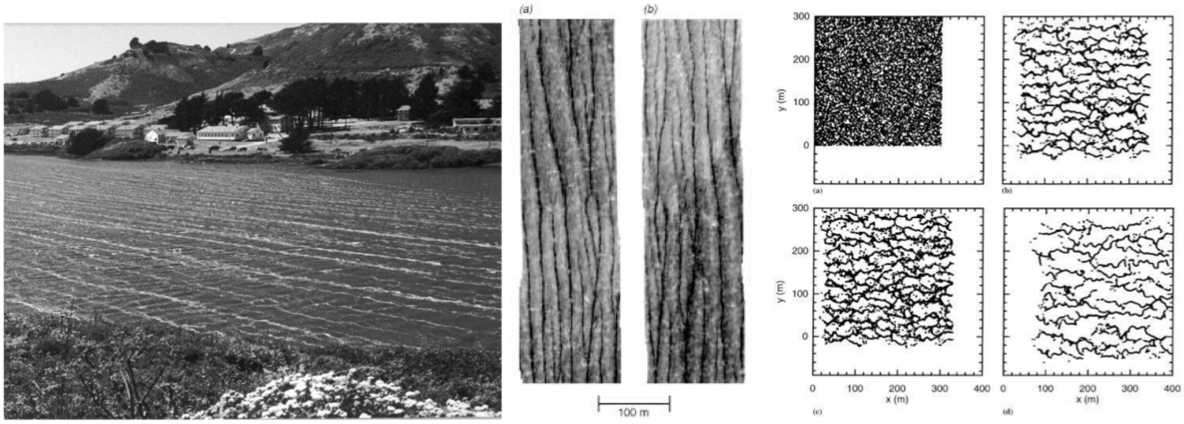


Figure 1.1: Images of Langmuir mixing: (a) a photograph of Rodeo Lagoon in CA (Szeri 1996), (b) an infrared image of the surface of Tama Bay (courtesy of G. Marmorino, NRL, D.C.), and (c) the evolution of surface tracers in a large eddy simulation of Langmuir turbulence (McWilliams et al., 1997). Images are reproduced from Chini et al. (2009).

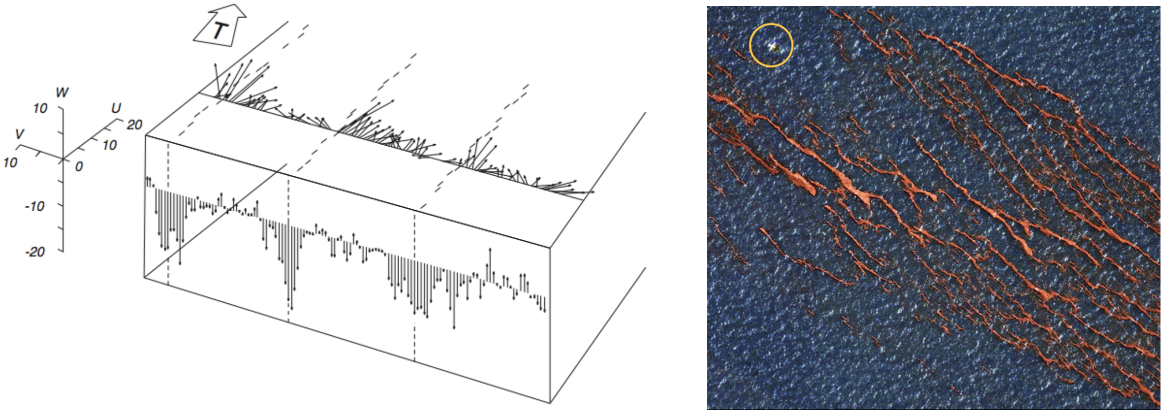


Figure 1.2: Observations of Langmuir mixing (a) from buoy data in the Pacific (Weller et al., 1985) and (b) satellite after the Deepwater Horizon oil spill (DigitalGlobe, 2010). A plane is circled in the satellite image to indicate scale. Images are reproduced from Stewart (2008) and NPR.org respectively.

and Sullivan, 2000; Kantha and Clayson, 2004; Harcourt, 2012). A common measurement in the aligned case of LM strength is the nondimensional parameter $La_t \equiv (u^*/u^s_{z=0})^{1/2}$, known as the turbulent Langmuir number (McWilliams et al., 1997). These aligned unidirectional velocities, u^* and $u^s_{z=0}$, represent the frictional and current velocity scales of the ocean surface due to wind drag (skin friction) and wave motion (Stokes drift) respectively. Intuitively, the turbulent Langmuir number can also be interpreted as the ratio of the production of turbulent kinetic energy due to Eulerian and Stokes shear (Grant and Belcher, 2009). In addition to the turbulent Langmuir number, other alternative parameters have been proposed. One such is the surface layer Langmuir number, which uses a depth average of Stokes drift, the wave-induced current, instead of its surface value (Harcourt and D'Asaro, 2008).

Before research on this dissertation began, it was unclear what role (if any) LM played in determining the depth of the ocean surface mixed layer since mixing occurred in a region already well-mixed. To test its potential importance, a preliminary LM parametrization was added to a GCM[1] to compare with observational data and simulations without the parameterization (a climate version of a "back of the envelope calculation"). This parametrization modified the near-surface mixing scheme of the GCM ocean component by using an energetic scaling and a climatology of the turbulent Langmuir number to determine when to deepen the mixed layer (Webb et al., 2010). The climatology was developed by Webb and Fox-Kemper (2009) and is based on wave spectra data (from an operational forecast ocean wave model)[2] and many simplifications (including a monochromatic approximation of the surface Stokes drift and an ansatz to handle wind-wave misalignment).

Observations (satellites, buoys, field campaigns, etc) of CFC column inventories and mixed layer depths are routinely used to tune and validate ocean circulation models. In most GCMs, there is a persistent, shallow mixed-layer bias (compared with observations) in the northern and southern oceans during their respective winters (when convective mixing is weakest) (Belcher et al.,

[1] NCAR Community Climate System Model version 3.5.

[2] NOAA WAVEWATCH III version 2.22.

2012; Fox-Kemper et al., 2011; Sallee et al., 2013) and as such there is interest among the ocean modeling community in reducing or correcting this bias. In this region, it is plausible that LM and convective mixing are of similar magnitude due to a prevalent combination of strong winds and large swell (Belcher et al., 2012). Initial tests of the LM parametrization in a GCM (NCAR CCSM 3.5) deepened the global mean mixed layer substantially ($\sim$10%) and dramatically improved the Southern Ocean shallow mixed-layer bias (see Fig. 1.3b). However, subsequent tests of a later model (NCAR CCSM 4) revealed that this preliminary parametrization was extremely sensitive to the details of the climatology (far beyond what could be inferred from data) and demonstrated a need for further work to accurately parametrize these effects (Webb et al., 2013). While much work is still needed, inclusion of LM in GCMs has the potential to correct a long-standing open problem in climate modeling.

1.1.3 Focus of research since preliminary results

Since these early investigations, much work has been done to improve the LM parametrization. Three components were identified as essential for improvement: the use of a prognostic wave field, better estimation of Stokes drift, and sounder treatment of wind-wave misalignment. First, to match the high variability of winds, a static LM climatology was replaced with a prognostic one, calculated with an evolving 2D wave field. A modified version of a third-generation wave model[3] has been coupled to a GCM[4] to calculate this field. Configuration, benchmarking, and testing of the wave module were conducted by Webb (Webb et al., 2013). Second, monochromatic approximations of the Stokes drift velocity were replaced with higher order, 2D spectral estimates (Webb and Fox-Kemper, 2011). And third, Van Roekel et al. (2012) derived and validated a new nondimensional parameter, the projected Langmuir number, to assess LM strength during wind-wave misalignment. Progress on all three components can now be combined with more sophisticated mixing schemes that utilize the 2D wave field (McWilliams and Sullivan, 2000; Harcourt and D'Asaro,

[3] NOAA WAVEWATCH III version 3.14. See Chapter 3 and Appendix B.1 for additional information.

[4] NCAR CESM 1.0.

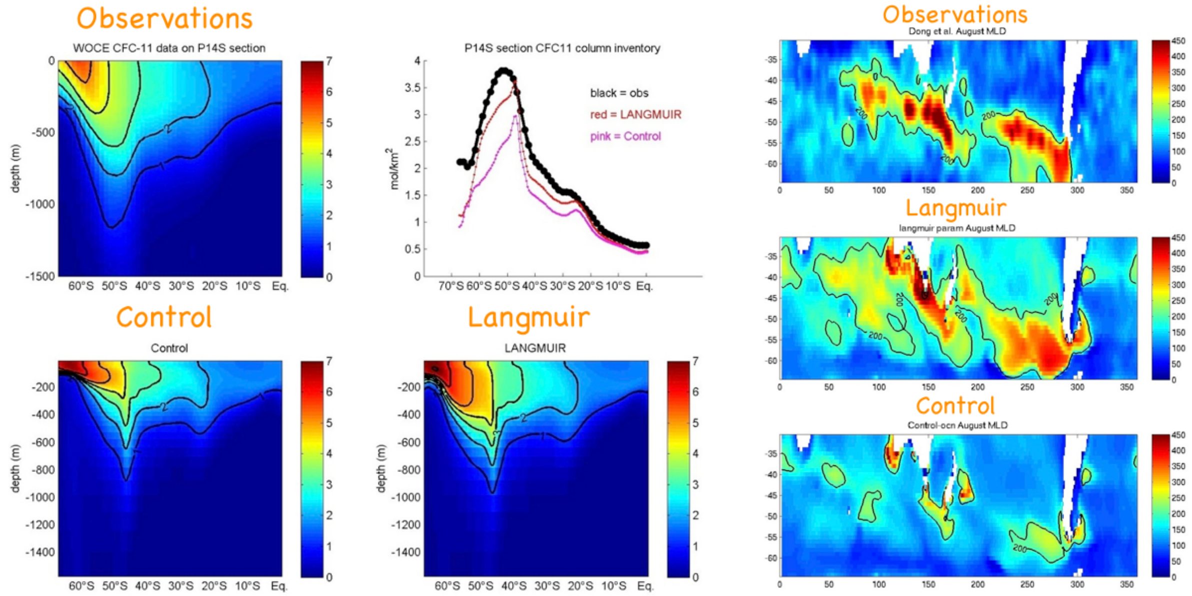


Figure 1.3: Comparison of observations and NCAR CCSM 3.5 output in the Southern Ocean with and without the Langmuir mixing parametrization. Biases are reduced in both the (a) CFC column inventory and (b) mixed layer depth. GCM output and images were generated by S. Peacock and G. Danabasoglu (Webb et al., 2013).

2008; Grant and Belcher, 2009) to build a more robust and accurate LM parametrization.

The majority of this thesis will focus on areas of research that were part of and grew from the aforementioned LM parametrization. Chapter 2 will derive a hierarchy of Stokes drift approximations and analyze their associated error. Chapter 3 will introduce spectral wave modeling, discuss the coupled wave model, and derive wave action balance equations[5] for use with a new spectral wave model. And finally, Chapter 4 covers construction of the meshless spectral wave prototype and analyzes its performance.

1.2 Review of surface gravity waves and linear wave theory

Surface gravity waves and linear wave theory are an important component in this dissertation. In Chapters 2 and 3, a superposition of linear wave trains are used to approximate Stokes drift and derive a wave action balance equation respectively. In addition, a deep-water approximation is used throughout to simplify many calculations. As such, a brief review is warranted and presented here.

1.2.1 Description of setup

Let $\boldsymbol{x} = (\boldsymbol{x}_h, z) \in \mathbb{R}^2 \times \mathbb{R}$. Consider a large basin filled with a fluid of some depth $H = H(\boldsymbol{x}_h)$ and assume the horizontal domain of the basin, $(-L_x/2, L_x/2) \times (-L_y/2, L_y/2)$, is large enough such that a dense set of surface gravity modes exist. Also, denote the upper dynamic boundary interface or **surface height perturbation** as $\eta = \eta(\boldsymbol{x}_h, t)$. Here we will assume the fluid is **ideal** with the following properties: continuous (no bubbles), inviscid (no internal frictional forces), and incompressible (Batchelor, 1967; Currie, 2003). In addition, we will assume any wave motion is irrotational.[6]

[5] The governing equation for spectral wave models.

[6] It will be assumed throughout the dissertation that the fluid is ideal and irrotational.

1.2.2 Balance equations

The governing linear wave equations are essentially a linearized set of balance equations for mass and momentum density combined with boundary conditions. Generalizing the results in Holthuijsen (2007) to include vectors, the following **basic balance equations**[7] for a unit cube of an arbitrary density property (scalar γ or vector $\boldsymbol{v}$) are used to derive the governing equations:

$$B(\gamma) = \partial_t \gamma + \nabla \cdot (\gamma \boldsymbol{u}) = S \tag{1.1}$$

$$\boldsymbol{B}(\boldsymbol{v}) = \partial_t \boldsymbol{v} + \operatorname{div}(\boldsymbol{v} \otimes \boldsymbol{u}) = \boldsymbol{S}. \tag{1.2}$$

Here the term 'div' is used to note that this is the divergence of a tensor. The $\otimes$ denotes the outer product for two vectors with dimensions m, n, given by

$$\boldsymbol{v} \otimes \boldsymbol{u} = \boldsymbol{v}\boldsymbol{u}^T = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \begin{bmatrix} u_1 & \dots & u_n \end{bmatrix} = \begin{bmatrix} v_1 u_1 & \dots & v_1 u_n \\ \vdots & \ddots & \vdots \\ v_m u_1 & \dots & v_m u_n \end{bmatrix} = \begin{bmatrix} u_1 \boldsymbol{v} & \dots & u_n \boldsymbol{v} \end{bmatrix}.$$

In 3D Cartesian space, it follows that

$$\operatorname{div}(\boldsymbol{v} \otimes \boldsymbol{u}) = \begin{bmatrix} \partial_x(v_x u_x) + \partial_y(v_x u_y) + \partial_z(v_x u_z) \\ \partial_x(v_y u_x) + \partial_y(v_y u_y) + \partial_z(v_y u_z) \\ \partial_x(v_z u_x) + \partial_y(v_z u_y) + \partial_z(v_z u_z) \end{bmatrix} = \begin{bmatrix} \partial_x(u_x \boldsymbol{v}) + \partial_y(u_y \boldsymbol{v}) + \partial_z(u_z \boldsymbol{v}) \end{bmatrix}.$$

Let $\boldsymbol{u} \in C^1(\mathbb{R}^3)$ represent the Eulerian fluid velocity. Then for constant mass density and no production of water, the **mass density balance equation** gives

$$B(\rho) = \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0,$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.3}$$

[7] Greater care is required if formulated in other coordinate systems.

which is also known as the **continuity equation** (Holthuijsen, 2007). Likewise, for an arbitrary $\boldsymbol{G}$ but constant ρ, the **momentum density balance equation** gives

$$\begin{aligned} B(\rho\boldsymbol{u}) = \partial_t(\rho\boldsymbol{u}) + \operatorname{div}(\rho\boldsymbol{u}\otimes\boldsymbol{u}) &= \boldsymbol{G}, \\ \partial_t\boldsymbol{u} + \partial_x(u_x\boldsymbol{u}) + \partial_y(u_y\boldsymbol{u}) + \partial_z(u_z\boldsymbol{u}) &= \frac{1}{\rho}\boldsymbol{G}, \\ \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} &= \frac{1}{\rho}\boldsymbol{G}. \end{aligned} \tag{1.4}$$

For an inviscid fluid, the momentum density balance equation only needs to consider the effects of pressure and gravity.[8] Using the D_t $(= \partial_t + \boldsymbol{u}\cdot\nabla)$ material derivative notation (Childress, 2009) and setting $\boldsymbol{G} = -\nabla p + \rho\mathbf{g}$ for an arbitrary pressure field $p = \rho P$, yields

$$D_t\boldsymbol{u} = -\nabla P + \mathbf{g}, \tag{1.5}$$

which with Eq. (1.3), are known as the **incompressible Euler equations with constant density** (Kundu and Cohen, 2008).

1.2.3 Velocity potential

Since the wave motion is irrotational, the velocity vector can be rewritten as

$$\boldsymbol{u} = \nabla\varphi, \tag{1.6}$$

where $\varphi = \varphi(\boldsymbol{x}, t)$ is the **scalar velocity potential** (Kundu and Cohen, 2008). Then Eq. (1.3) becomes

$$\nabla^2\varphi = 0. \tag{1.7}$$

Substituting first the z-component of Eq. (1.4) gives

$$\begin{aligned} \partial_t(\,\partial_z\varphi) + \nabla\varphi\cdot\nabla(\,\partial_z\varphi) &= -\,\partial_z P - \mathrm{g}, \\ \partial_z\left(\,\partial_t\varphi + \frac{1}{2}\nabla\varphi\cdot\nabla\varphi\right) &= -\,\partial_z\left(P + \mathrm{g}z\right). \end{aligned}$$

[8] It should be noted that gravity is aligned with the z-component in 3D Cartesian space.

Combining the other components, switching the order of differentiation,[9] and integrating, we find that

$$\nabla\left(\partial_t\varphi+\frac{1}{2}\nabla\varphi\cdot\nabla\varphi\right)=-\nabla\left(P+\mathrm{g}z\right),$$

$$\partial_t\varphi+\frac{1}{2}\nabla\varphi\cdot\nabla\varphi+P+\mathrm{g}z=F, \tag{1.8}$$

This is known as the **Bernoulli equation for unsteady irrotational motion**, where $F=F(t)$ is some arbitrary integrating function (Currie, 2003).

1.2.4 Boundary conditions

Three boundary conditions, one at the bottom and two at the surface, are employed to close and combine the balance equations. The first condition simply states that the fluid cannot penetrate the bottom. This implies that the velocity component normal to the bottom at the bottom must equal zero. If the bottom boundary is rewritten implicitly as the equation

$$H_{\text{surface}}(x,y,z)=z+H(x,y)=0, \tag{1.9}$$

then a normal to the bottom boundary can be written as $\nabla H_{\text{surface}}=(\,\partial_x H,\,\partial_y H,1)$. Since the normal velocity must vanish (Whitham, 1974), the **bottom boundary condition** can then be derived as:

$$(\nabla H_{\text{surface}})\cdot\boldsymbol{u}=0,\qquad z=-H$$

$$\partial_z\varphi+\nabla_h H\cdot\nabla_h\varphi=0,\qquad z=-H. \tag{1.10}$$

The second condition states that if a fluid parcel is at the surface at some initial time, then it must remain on the surface for all later time (i.e., no spray or cavitation) (Currie, 2003). This implies

$$D_t\left(\eta-z\right)=0,\qquad z=\eta,$$

$$\partial_t\eta+\,\partial_x\varphi\,\partial_x\eta+\,\partial_y\varphi\,\partial_y\eta=\,\partial_z\varphi,\qquad z=\eta, \tag{1.11}$$

[9] Since it is assumed $\boldsymbol{u}\in C^1(\mathbb{R}^3)$, $\partial_{\alpha_i\alpha_j}\varphi\in C^0(\mathbb{R}^3)$ and the order of differentiation can be switched.

and gives the **kinematic boundary condition at the surface**. And thirdly, the **dynamic boundary condition at the surface** is obtained by evaluating the Bernoulli equation for unsteady irrotational motion at the surface (Currie, 2003), giving

$$\partial_t\varphi + \frac{1}{2}\nabla\varphi\cdot\nabla\varphi + P_{\text{atm}} + \mathrm{g}\eta = F, \qquad z = \eta. \tag{1.12}$$

If we split the atmospheric pressure per density $P_{\text{atm}} = P_{\text{atm}}(\boldsymbol{x}, t)$ such that

$$P_{\text{atm}}(\boldsymbol{x}_h, z=\eta, t) = P_{\Delta}(\boldsymbol{x}_h, t) + \frac{1}{L_x L_y}\int_{-L_y/2}^{L_y/2}\int_{-L_x/2}^{L_x/2} P_{\text{atm}}|_{z=0}\, dxdy, \tag{1.13}$$

then Eq. (1.12) can be rewritten as

$$\partial_t\varphi + \frac{1}{2}\nabla\varphi\cdot\nabla\varphi + \mathrm{g}\eta = -P_{\Delta}, \qquad z = \eta, \tag{1.14}$$

since the integrating function is arbitrary.

1.2.5 Governing nonlinear wave equations

For an incompressible and inviscid fluid, the **governing nonlinear wave equations** are then

$$\nabla^2\varphi = 0, \qquad z \in (-H, \eta)\,, \tag{1.15}$$

$$\partial_t\eta + \partial_x\varphi\,\partial_x\eta + \partial_y\varphi\,\partial_y\eta = \partial_z\varphi, \qquad z = \eta, \tag{1.16}$$

$$\partial_t\varphi + \frac{1}{2}\nabla\varphi\cdot\nabla\varphi + \mathrm{g}\eta = -P_{\Delta}, \qquad z = \eta, \tag{1.17}$$

$$\partial_z\varphi + \nabla_h H\cdot\nabla_h\varphi = 0, \qquad z = -H. \tag{1.18}$$

1.2.6 Treatment of atmospheric pressure fluctuations

To be as general as possible, the pressure per density deviation (P_{Δ}) is included in Eq. (1.17). It can be removed however if the horizontal material derivative of the deviation is fairly small (i.e., $D_t^h P_{\Delta} \ll 1$). This can be shown explicitly later if the two surface boundary conditions are combined first by taking the horizontal material derivative of Eq. (1.17) and substituting into Eq. (1.16) as

$$\begin{aligned} D_t^h\left[\partial_t\varphi+\frac{1}{2}\nabla\varphi\cdot\nabla\varphi\right]+\mathrm{g}\,(\,\partial_t\eta+\,\partial_x\varphi\,\partial_x\eta+\,\partial_y\varphi\,\partial_y\eta)= \\ \partial_{tt}\varphi+\,\partial_t\left(\nabla_h\varphi\cdot\nabla_h\varphi\right)+\frac{1}{2}\,\partial_t\,(\,\partial_z\varphi)^2+ \\ \frac{1}{2}\nabla_h\varphi\cdot\nabla_h\left(\nabla\varphi\cdot\nabla\varphi\right)+\mathrm{g}\,\partial_z\varphi=-D_t^h\left[P_\Delta\right], \qquad z=\eta. \end{aligned} \tag{1.19}$$

While the use of Eq. (1.19) is unorthodox and tedious at best, it does permit a single asymptotic expansion between the nonlinear shallow and classical linear regimes.

1.2.7 Non-steep regimes of the governing nonlinear wave equations

With an appropriate nondimensionalization, the governing nonlinear wave equations can be split into different regimes. To simplify the problem, the case of constant depth ($H(\boldsymbol{x}_h)=H_\mathrm{c}$) only is considered here and the notation $\mathcal{P}=D_t^h[P_\Delta]$ is used. Similar to Hendershott et al. (1989), let

$$\boldsymbol{x}_h=L\tilde{\boldsymbol{x}}_h, \qquad z=H_c\tilde{z}, \qquad t=\frac{L}{\sqrt{\mathrm{g}H_c}}\tilde{t}, \qquad \eta=a\tilde{\eta}, \qquad \varphi=aL\sqrt{\frac{\mathrm{g}}{H_c}}\tilde{\varphi}, \qquad \mathcal{P}=\beta\tilde{\mathcal{P}}, \tag{1.20}$$

where the scaling for $\mathcal{P}$ is unspecified yet and φ is based on $\partial_t\varphi+\mathrm{g}\eta\simeq 0$. Potential differences in horizontal length scales are ignored and the scaling for t is chosen for convenience. Substituting into Eqs. (1.15), and (1.18), and (1.19) and dropping tildes, we find

$$\frac{1}{L^2}\nabla^2_{\boldsymbol{x}_h}\varphi+\frac{1}{H_c^2}\,\partial_{zz}\varphi=0, \qquad z\in\left(-1,\frac{a}{H_c}\eta\right), \tag{1.21}$$

$$\begin{aligned} \partial_{tt}\varphi+\frac{a}{H_c}\,\partial_t\left(\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\varphi\right)+\frac{aL^2}{2H_c^3}\,\partial_t\,(\,\partial_z\varphi)^2+ \\ \frac{a^2}{2H_c^2}\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\left[\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\varphi+\frac{L^2}{H_c^2}\,(\,\partial_z\varphi)^2\right]+ \\ \frac{L^2}{H_c^2}\,\partial_z\varphi-\frac{\beta L}{a\sqrt{g^3H_c}}\tilde{\mathcal{P}}=0, \qquad z=\frac{a}{H_c}\eta, \end{aligned} \tag{1.22}$$

$$\partial_z\varphi=0, \qquad z=-1. \tag{1.23}$$

If the following dimensionless quantities are introduced

$$\varepsilon\equiv a/H_c, \qquad \delta\equiv H_c/L, \tag{1.24}$$

then the quantity $\varepsilon\delta$ can be used and differentiate between non-steep and steep waves. In addition, if $\beta = \beta_0(a^2\sqrt{g^3 H_c})/L^2$ for $\beta_0 \in [0,1]$, then $\mathcal{P}$ can be neglected. Substituting into Eqs. (1.21) to (1.23) gives

$$\delta\nabla^2_{\boldsymbol{x}_h}\varphi + \frac{1}{\delta}\,\partial_{zz}\varphi = 0, \qquad z \in (-1, \varepsilon\eta)\,, \tag{1.25}$$

$$\begin{aligned} &\partial_{tt}\varphi + \varepsilon\,\partial_t\left(\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\varphi\right) + \frac{\varepsilon}{2\delta^2}\,\partial_t\left(\,\partial_z\varphi\right)^2 + \\ &\frac{\varepsilon^2}{2}\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\left[\nabla_{\boldsymbol{x}_h}\varphi\cdot\nabla_{\boldsymbol{x}_h}\varphi + \frac{1}{\delta^2}\left(\,\partial_z\varphi\right)^2\right] + \frac{1}{\delta^2}\,\partial_z\varphi = \varepsilon\delta\beta_0\tilde{\mathcal{P}}, \qquad z = \varepsilon\eta, \end{aligned} \tag{1.26}$$

$$\partial_z\varphi = 0, \qquad z = -1. \tag{1.27}$$

For non-steep, intermediate-length waves, or $\varepsilon \ll 1$ and $\delta \sim 1$, Eqs. (1.25) to (1.27) reduce to the dimensionless, governing linear wave equations:

$$\nabla^2_{\boldsymbol{x}}\varphi = 0, \qquad z \in (-1, 0) \tag{1.28}$$

$$\partial_{tt}\varphi + \,\partial_z\varphi = 0, \qquad z = 0 \tag{1.29}$$

$$\partial_z\varphi = 0, \qquad z = -1. \tag{1.30}$$

Similarly, for non-steep but long waves, or $\delta \ll 1$ and $\varepsilon \sim 1$, one can derive the dimensionless, governing nonlinear shallow-water equations (Hendershott et al., 1989).

1.2.8 Governing linear wave equations

Choosing the distinguished limits $\varepsilon \ll 1$, $\delta \sim 1$ and reverting back to dimensional form, yields the **governing linear wave equations** for non-varying depth:

$$\nabla^2\varphi = 0, \qquad z \in (-H_c, 0) \tag{1.31}$$

$$\partial_{tt}\varphi + \mathrm{g}\,\partial_z\varphi = 0, \qquad z = 0 \tag{1.32}$$

$$\partial_z\varphi = 0, \qquad z = -H_c. \tag{1.33}$$

To solve, the linear equations are decomposed into Fourier modes using the following plane wave ansatzes (Ablowitz, 2011):

$$\eta(\boldsymbol{x}_h, t) = \Re\left\{A\, e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\} \tag{1.34}$$

$$\varphi(\boldsymbol{x}, t) = \Re\left\{B(z) e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\} \tag{1.35}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = (\boldsymbol{u}_h(\boldsymbol{x}, t), u_z(\boldsymbol{x}, t)) = \Re\left\{(i\mathbf{k}B(z),\, \partial_z B(z))\, e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\}. \tag{1.36}$$

Here, $\mathbf{k} = (k_x, k_y)$, $A \in \mathbb{R}_+$, and $\omega = \omega(\mathbf{k})$ is the absolute angular frequency for some arbitrary initialization τ. In addition, Eq. (1.17) is linearized to establish

$$B|_{z=0} = \frac{-i\mathrm{g}}{\omega} A. \tag{1.37}$$

Inserting Eq. (1.35) into Eq. (1.31) yields

$$\partial_{zz} B = |\mathbf{k}|^2 B, \qquad z \in (-H_c, 0)\,.$$

Solving and employing Eq. (1.33), we find

$$B(z) = B_c \cosh[|\mathbf{k}|(z + H_c)]\,,$$

for some yet to be determined constant B_c. Finally, employing Eq. (1.32), yields

$$\partial_z B = \frac{\omega^2}{\mathrm{g}} B, \qquad z = 0,$$

and for $\omega = \omega_\pm = \pm\sigma$, gives the following **linear dispersion relation**:

$$\sigma(\mathbf{k}) = (\mathrm{g}\, |\mathbf{k}| \tanh[|\mathbf{k}| H_c])^{1/2}\,. \tag{1.38}$$

Utilizing Eq. (1.37), we finally derive the **monochromatic linear wave solutions**

$$\begin{aligned}
\varphi(\boldsymbol{x}, t) &= \Re\left\{\frac{-i\mathrm{g}\cosh[|\mathbf{k}|(z + H_c)]}{\omega\cosh[|\mathbf{k}|H_c]}\, A e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\} \\
&= \Re\left\{\frac{-i\omega\cosh[|\mathbf{k}|(z + H_c)]}{|\mathbf{k}|\sinh[|\mathbf{k}|H_c]}\, A e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\}, \\
\boldsymbol{u}_h(\boldsymbol{x}, t) &= \Re\left\{\mathbf{k}\,\frac{\omega\cosh[|\mathbf{k}|\,(z + H_c)]}{|\mathbf{k}|\sinh[|\mathbf{k}|H_c]}\, A e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\}, \\
u_z(\boldsymbol{x}, t) &= \Re\left\{\frac{-i\omega\sinh[|\mathbf{k}|(z + H_c)]}{\sinh[|\mathbf{k}|H_c]}\, A e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega t + \tau]}\right\}.
\end{aligned}$$

which simplify to

$$\eta(\boldsymbol{x}_h, t) = A\cos[\mathbf{k}\cdot\boldsymbol{x}_h - \omega(\mathbf{k})t + \tau]\,, \tag{1.39}$$

$$\varphi(\boldsymbol{x}, t) = \frac{\omega(\mathbf{k})\cosh[|\mathbf{k}|(z+H_c)]}{|\mathbf{k}|\sinh[|\mathbf{k}|H_c]}\,A\sin[\mathbf{k}\cdot\boldsymbol{x}_h - \omega(\mathbf{k})t + \tau]\,, \tag{1.40}$$

$$\boldsymbol{u}_h(\boldsymbol{x}, t) = \mathbf{k}\,\frac{\omega(\mathbf{k})\cosh[|\mathbf{k}|(z+H_c)]}{|\mathbf{k}|\sinh[|\mathbf{k}|H_c]}\,A\cos[\mathbf{k}\cdot\boldsymbol{x}_h - \omega(\mathbf{k})t + \tau]\,, \tag{1.41}$$

$$u_z(\boldsymbol{x}, t) = \frac{\omega(\mathbf{k})\sinh[|\mathbf{k}|(z+H_c)]}{\sinh[|\mathbf{k}|H_c]}\,A\sin[\mathbf{k}\cdot\boldsymbol{x}_h - \omega(\mathbf{k})t + \tau]\,. \tag{1.42}$$

Since Eqs. (1.31) to (1.33) are linear, any linear superposition of Eqs. (1.39) through Eqs. (1.42) will also be a solution as long as they do not violate the original assumptions (i.e., $\varepsilon \ll 1$, $\delta \sim 1$).

1.2.9 Linear deep-water approximation

For a large part of the ocean, the deep-water approximation is adequate if the dynamics of interest are away from the coastlines and confined to an upper portion of the vertical domain. The term 'deep' is relative to wavelength and is generally used when depths are greater than half the wavelength (Holthuijsen, 2007). To elucidate, error to the deep-water approximation is bounded for wave motion restricted to the top third of the vertical domain and depths greater than a third of the longest wavelengths.

Since wind-generated surface gravity waves are characterized by wave lengths of 0.1 to 1500 m (Holthuijsen, 2007), let $\lambda \in [0.1, 1500]$. Then for depths greater than 500 m ($\max\{\lambda\}/3 \le H_c$), $|\mathbf{k}|H_c$ ($= 2\pi H_c/\lambda$) can be bounded below as

$$\frac{2\pi}{3} = \frac{1}{3}\min\{|\mathbf{k}|\}\max\{\lambda\} \le |\mathbf{k}|H_c,$$

and $\tanh\left[|\mathbf{k}|H_c\right] \approx 1$, or

$$1 - \left(\tanh\left[|\mathbf{k}|H_c\right]\right)^{1/2} \le 1.5\times 10^{-2}.$$

Then the linear dispersion relation simplifies to the **deep-water dispersion relation** as

$$\sigma(\mathbf{k}) \approx \sqrt{\mathrm{g}\,|\mathbf{k}|}, \tag{1.43}$$

with the error bounded as $\sim 10^{-2}$. In addition, $\boldsymbol{u}$ and φ simplify as well. Let $\mu = -z/H_c > 0$ and $\delta = |\mathbf{k}|H_c \geq 2\pi/3$. Then the z-components of Eqs. (1.40) to (1.42) can be rewritten as

$$\frac{\cosh\left[|\mathbf{k}|\left(z+H_c\right)\right]}{\sinh\left[|\mathbf{k}|H_c\right]} = \frac{\cosh\left[\delta\left(1-\mu\right)\right]}{\sinh\left[\delta\right]} = e^{-\delta\mu}\left[1+\left(\coth\left[\delta\right]-1\right)\cosh\left[\delta\mu\right]e^{\delta\mu}\right],$$

and

$$\frac{\sinh\left[|\mathbf{k}|\left(z+H_c\right)\right]}{\sinh\left[|\mathbf{k}|H_c\right]} = \frac{\sinh\left[\delta\left(1-\mu\right)\right]}{\sinh\left[\delta\right]} = e^{-\delta\mu}\left[1+\left(1-\coth\left[\delta\right]\right)\sinh\left[\delta\mu\right]e^{\delta\mu}\right].$$

Notice that if $\mu \leq 1/3$, then

$$\left(\coth\left[\delta\right]-1\right)\cosh\left[\delta\mu\right]e^{\delta\mu} \leq \left(\coth\left[\delta\right]-1\right)\cosh\left[\delta/3\right]e^{\delta/3} \leq 7.8\times 10^{-2},$$
$$\left(\coth\left[\delta\right]-1\right)\sinh\left[\delta\mu\right]e^{\delta\mu} \leq \left(\coth\left[\delta\right]-1\right)\sinh\left[\delta/3\right]e^{\delta/3} \leq 4.7\times 10^{-2}.$$

This implies that if the dynamics of interest are confined to the top third vertical domain ($z \in [-H_c/3, 0]$), then the z-components of $\boldsymbol{u}$ and φ simplify to $e^{|\mathbf{k}|z}$, with the error bounded as $O(10^{-2})$ as well. Then for depths greater than 500 m ($\max\{\lambda\}/3 \leq H_c$), Eqs. (1.39) to (1.42) reduce to the **monochromatic linear deep-water solutions** as

$$\eta_{\mathbf{k}}(\boldsymbol{x}_h, t) = A\cos\Big(\mathbf{k}\cdot\boldsymbol{x}_h \mp \sqrt{\mathrm{g}|\mathbf{k}|}\,t + \tau\Big), \tag{1.44}$$

$$\varphi_{\mathbf{k}}(\boldsymbol{x}, t) = \pm\sqrt{\frac{\mathrm{g}}{|\mathbf{k}|}}\,A\,e^{|\mathbf{k}|z}\sin\Big(\mathbf{k}\cdot\boldsymbol{x}_h \mp \sqrt{\mathrm{g}|\mathbf{k}|}\,t + \tau\Big), \tag{1.45}$$

$$\boldsymbol{u}_{h,\mathbf{k}}(\boldsymbol{x}, t) = \pm\mathbf{k}\sqrt{\frac{\mathrm{g}}{|\mathbf{k}|}}\,A\,e^{|\mathbf{k}|z}\cos\Big(\mathbf{k}\cdot\boldsymbol{x}_h \mp \sqrt{\mathrm{g}|\mathbf{k}|}\,t + \tau\Big), \tag{1.46}$$

$$u_{z,\mathbf{k}}(\boldsymbol{x}, t) = \pm\sqrt{\mathrm{g}|\mathbf{k}|}\,A\,e^{|\mathbf{k}|z}\sin\Big(\mathbf{k}\cdot\boldsymbol{x}_h \mp \sqrt{\mathrm{g}|\mathbf{k}|}\,t + \tau\Big). \tag{1.47}$$

Chapter 2

Stokes drift

2.1 Introduction and motivation for analysis

The **Stokes drift velocity** (hereafter Stokes drift[1] and abbreviated as SD) is an important vector component that appears often in wave-averaged dynamics. Mathematically, SD is the mean difference between Eulerian and Lagrangian velocities and intuitively can be thought of as the near-surface ocean current induced from wave action. This nonlinear phenomenon was first identified by George G. Stokes in 1847 (Craik, 2005), and the correspondingly named **Stokes transport** is the vertically integrated SD (Kenyon, 1970; Smith, 2006). In the absence of other currents, Fig. 2.1 illustrates the basic principle of SD. For monochromatic linear deep-water waves (see Section 1.2.9), fluid parcel trajectories to leading-order are closed orbits (Fig. 2.1a) and the mean Eulerian velocities (at a point) must be zero due to irrotationality of the fluid. However, the actual orbits of the linear waves are not closed (Fig. 2.1b) and over time there is depth-dependent drift (Fig. 2.1c) (Kundu and Cohen, 2008). Even though this is a second-order effect, the magnitudes of these near-surface currents can be significant (Longuet-Higgins, 1969; Ardhuin et al., 2009).

SD appears naturally in the Craik-Leibovich equations. In this set of surface-wave-filtered Navier-Stokes equations, the waves are assumed to be linear and the Navier-Stokes equations are time-averaged over periods that are long compared to the waves but short compared to other motions. These equations[2] are derived explicitly and completely in Craik and Leibovich (1976) and

[1] Technically, Stokes drift refers to the mean horizontal displacement (not velocity) but the shortened usage is common in literature.

[2] See Appendix A.1 for a brief description of the Craik-Leibovich equations.

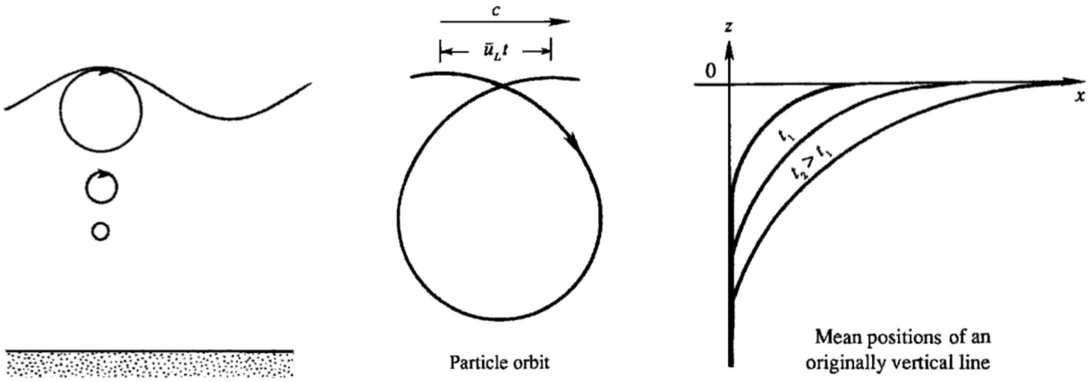


Figure 2.1: The basic principle of SD in 2D and absence of background currents is illustrated here. The (a) leading-order and (b) actual, fluid parcel trajectories (governed by solutions to the linear wave equations) have closed and non-closed orbits respectively. This difference leads to (c) a nonlinear mean drift over time. Cartoons are reproduced from Kundu and Cohen (2008).

in the rotating Boussinesq form in Holm (1996). In large-eddy simulations of upper ocean turbulence (e.g., McWilliams et al., 1997; Grant and Belcher, 2009; Van Roekel et al., 2012), it is generally presumed that the Boussinesq form of the equations are a useful intermediate step between full wave-resolving models (that are expensive) and models that completely neglect surface-wave effects. Essentially all present parameterizations of LM include some sort of SD velocity or equivalent (e.g., McWilliams and Sullivan, 2001; Axell, 2002; Smyth et al., 2002; Kantha and Clayson, 2004; Harcourt and D'Asaro, 2008; Van Roekel et al., 2012).

While our primary interest in SD is due to its role in LM, it is important in other areas as well. It is involved in the upper ocean momentum balance and the transport of tracers, and is closely related to the mass transport by waves, the wave-related pressure, and the wave surface stress correction (McWilliams and Restrepo, 1999; McWilliams et al., 2004; McWilliams and Fox-Kemper, 2013). Any one of these quantities may be of interest for inclusion in large-scale ocean modeling. In addition, SD also plays a role in the emerging field of drift forecasting, for operations such as search-and-rescue and oil spill containment (Hackett et al., 2006; Haza et al., 2012). Thus, it is well established in the literature that SD is an important property of the wave field.

This chapter will go into details on how SD may be derived and estimated from varying degrees of spectral information. Sections 2.2 and 2.3 originally appeared in the appendix of Webb and Fox-Kemper (2011) and are modified and expanded here to fit within the chapter. The main text of Webb and Fox-Kemper (2011) is summarized in Section 2.4. To account for directional spreading of wave energy, a new lower-order SD approximation is proposed in Section 2.5 and is used to diagnose differences between other lower and higher-order approximations.

2.2 Formal definition in Cartesian coordinates

Generally, SD is the mean difference between the Lagrangian velocity $\boldsymbol{u}^L$ (the velocity following the motion of a fluid parcel) and the Eulerian velocity $\boldsymbol{u}^E$. Alternative definitions can be found in Longuet-Higgins (1969), Jansons and Lythe (1998), and Mellor (2011). The leading-order SD velocity will be formally defined here in Cartesian coordinates in a domain similar to the one in Section 1.2.1. While this definition holds for many types of motions (such as tidal and inertial), we will assume the wave motions of interest are to leading-order, periodic surface gravity waves (Longuet-Higgins, 1969).

We will define SD as a time and spatial mean over a period $T \neq 0$ (since the instantaneous value of these velocities is identical at a given location) and a horizontal length scale $\boldsymbol{X}_h = (X_h, Y_h)$ (to remove high frequency variations). Let the position of a fluid parcel at time t be given by $\boldsymbol{x}_p(t)$. Here the interest is in estimating the basic wave-averaged dynamics as set out by Craik and Leibovich (1976) and McWilliams and Restrepo (1999), without higher-order effects. The Lagrangian and Eulerian velocities and fluid parcel displacements can be related through the same Taylor-series expansions,

$$\begin{aligned}
\boldsymbol{u}^L(\boldsymbol{x}_p(t_0), t) &= \boldsymbol{u}^E(\boldsymbol{x}_p(t), t) \\
&= \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), t) + [\boldsymbol{x}_p(t) - \boldsymbol{x}_p(t_0)] \cdot \nabla \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), t) + \boldsymbol{R}_1(\boldsymbol{x}_p(t); \boldsymbol{x}_p(t_0)) \,, \\
\boldsymbol{x}_p(t) - \boldsymbol{x}_p(t_0) &= \int_{t_0}^{t} \boldsymbol{u}^L\big(\boldsymbol{x}_p(t_0), s'\big)\, ds' \\
&= \int_{t_0}^{t} \big[\boldsymbol{u}^E\big(\boldsymbol{x}_p(t_0), s'\big) + \boldsymbol{R}_0\big(\boldsymbol{x}_p(s'); \boldsymbol{x}_p(t_0)\big)\big]\, ds',
\end{aligned}$$

where $\boldsymbol{R}_0$ and $\boldsymbol{R}_1$ are zeroth and first-order remainder vector terms. As previously mentioned, we will formally define SD as

$$\boldsymbol{u}^S(\boldsymbol{x}, t; \boldsymbol{X}_h, T) \equiv \big\langle \boldsymbol{u}^L(\boldsymbol{x}, t) - \boldsymbol{u}^E(\boldsymbol{x}, t) \big\rangle_{\boldsymbol{X}_h, T} \tag{2.1}$$

$$\equiv \frac{1}{T} \int_{t-T/2}^{t+T/2} \big\langle \boldsymbol{u}^L(\boldsymbol{x}, s) - \boldsymbol{u}^E(\boldsymbol{x}, s) \big\rangle_{\boldsymbol{X}_h}\, ds \tag{2.2}$$

$$\equiv \frac{1}{X_h Y_h} \int_{\boldsymbol{x}_h - \boldsymbol{X}_h/2}^{\boldsymbol{x}_h + \boldsymbol{X}_h/2} \big\langle \boldsymbol{u}^L\big(\boldsymbol{x}'_h, z, t\big) - \boldsymbol{u}^E\big(\boldsymbol{x}'_h, z, t\big) \big\rangle_T\, d\boldsymbol{x}'_h, \tag{2.3}$$

where angle brackets denote time or spatial averaging. Ignoring the remainder terms for now and reorganizing, first gives

$$\begin{aligned}
\boldsymbol{u}^L(\boldsymbol{x}_p(t_0), t) - \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), t) &\approx [\boldsymbol{x}_p(t) - \boldsymbol{x}_p(t_0)] \cdot \nabla \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), t) \\
&\approx \left[\int_{t_0}^{t} \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), s') ds'\right] \cdot \nabla \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), t) \,.
\end{aligned} \tag{2.4}$$

Substitution and temporal averaging next yield a leading-order estimate

$$\begin{aligned}\frac{1}{T}\int_{t_0-T/2}^{t_0+T/2}&\left[\boldsymbol{u}^L(\boldsymbol{x}_p(t_0),t)-\boldsymbol{u}^E(\boldsymbol{x}_p(t_0),t)\right]dt\\ &=\frac{1}{T}\int_{t_0-T/2}^{t_0+T/2}\left[\int_{t_0}^{t}\boldsymbol{u}^E\big(\boldsymbol{x}_p(t_0),s'\big)\,ds'\right]\cdot\nabla\boldsymbol{u}^E(\boldsymbol{x}_p(t_0),t)\,dt\\ &=\big\langle\boldsymbol{u}^L(\boldsymbol{x}_p(t_0),t_0)-\boldsymbol{u}^E(\boldsymbol{x}_p(t_0),t_0)\big\rangle_T\,. \end{aligned}\tag{2.5}$$

Finally, spatial averaging gives the leading-order SD,

$$\boldsymbol{u}^S(\boldsymbol{x},t;\boldsymbol{X}_h,T)\approx\left\langle\frac{1}{T}\int_{t-T/2}^{t+T/2}\left[\int_{t}^{s}\boldsymbol{u}^{\mathrm{w}}\big(\boldsymbol{x},s'\big)\,ds'\right]\cdot\nabla\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s)\,ds\right\rangle_{\boldsymbol{X}_h}\,,\tag{2.6}$$

where $\boldsymbol{u}^E$ has been replaced with $\boldsymbol{u}^{\mathrm{w}}$ to distinguish scales.

It should be emphasized that the interval T is sufficient to average over relevant wave displacements by the fast wave velocity $\boldsymbol{u}^{\mathrm{w}}$, but not so long that SD is not a function of time, for example due to wind variability. Similarly, the horizontal length scale $\boldsymbol{X}_h$ is sufficient to remove high frequency fluctuations but not long enough to smooth the frequencies of interest. This smoothing is essential for SD since it removes possible spatially-oscillatory waves that are independent of time (Webb and Fox-Kemper, 2011).

Up until this point, there has been no discussion on the necessary conditions for the approximation to hold. Since for our purposes we have assumed the wave motion to be linear (ideal, irrotational, non-steep waves, etc.), the inclusion of remainder terms $\boldsymbol{R}_0$ and $\boldsymbol{R}_1$ is guaranteed to be third-order or less. This is due to the fact that $\boldsymbol{u}^{\mathrm{w}}$ is periodic to leading-order (Longuet-Higgins, 1969). However for more general motion, this is no longer the case. While it is possible to bound the higher-order terms for a finite period (see Appendix C.1), the results are uninteresting without further assumptions. For a related in-depth treatment, the reader is directed to work by Hasselmann (1971) and McWilliams et al. (2004).

2.3 Derivation of the SD spectral density estimate

A spectral density form of SD for linear surface gravity waves is necessary to estimate LM with a prognostic wave field. Previous derivations of SD in spectral density form can be found in Kenyon (1969), Huang (1971), and McWilliams and Restrepo (1999). To further illustrate, a spectral density estimate based on grid cell averages is presented here for use in a spectral wave model. Since spectral wave modeling will be discussed in detail in Chapter 3, only a cursory description of the modeling approach is given. However, it is prudent to describe the discrete spectra and a thorough treatment of how spectra and SD are formulated within the model is given here.

2.3.1 Wave field decomposition for model inclusion

To illustrate how the wave field is decomposed concretely, consider a spectral linear wave model with an arbitrary domain $\boldsymbol{L}_h$ consisting of grid cells $L \times L$ in size. Furthermore, assume that the wave dynamics being modeled are separable into fast and slow scales, such that the fast dynamics can be represented within each grid cell by a periodic, statistically homogeneous and stationary wave field. Then the slower dynamics can be represented by mean properties of each cell that vary slowly from neighbor to neighbor. For purposes of this derivation, a series approximation will be used to represent the fast dynamics while grid cell averages will serve to model the slower ones. Now within each grid cell, let an arbitrary wave field with a surface displacement η be approximated by η^{w}, a superposition of monochromatic linear deep-water solutions from Section 1.2.9. For reference, the classical solutions in more compact form are

$$\boldsymbol{u}_{\mathbf{k}} = (\boldsymbol{u}_{h,\mathbf{k}}, w_{\mathbf{k}}) = -\nabla\varphi_{\mathbf{k}}, \tag{2.7}$$

$$\varphi_{\mathbf{k}} = -\frac{e^{kz}}{k}\,\partial_t\,\eta_{\mathbf{k}}(\boldsymbol{x}_h, t), \tag{2.8}$$

$$\eta_{\mathbf{k}} = a_{\mathbf{k}} \cos\left[\mathbf{k}\cdot\boldsymbol{x}_h - \omega^{+}_{\mathbf{k}} t + \tau_{\mathbf{k}}\right]. \tag{2.9}$$

Here, $\eta_{\mathbf{k}}$, for a given wavevector $\mathbf{k}$ (and wavenumber $k = |\mathbf{k}|$), has amplitude $a_{\mathbf{k}}$ (slowly varying in space and time), phase shift $\tau_{\mathbf{k}}$, and positive frequency $\omega^{+}_{\mathbf{k}} = \omega^{+}_{k} = \sqrt{gk}$. These solutions and dispersion relation are only appropriate if small wave slope ($ka_{\mathbf{k}} \ll 1$) and deep water ($kD \gg 1$) are assumed. If in addition, the approximate wave field is periodic at the boundary, $\eta_{\mathbf{k}}$ has discrete wavevectors ($\mathbf{k} = \mathbf{k}_{m,n} = (k_{x_m}, k_{y_n}) = \frac{2\pi}{L}(m, n)$, for $m, n = 0, \pm1, \pm2, \ldots$) and can be reformulated as

$$\eta_{\mathbf{k}_{mn}} = c_{\mathbf{k}_{mn}} e^{i[\mathbf{k}_{mn}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}_{mn}} t]} + c^*_{\mathbf{k}_{mn}} e^{-i[\mathbf{k}_{mn}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}_{mn}} t]}, \tag{2.10}$$

where $c_{\mathbf{k}_{mn}}$ corresponds to $\frac{1}{2}a_{\mathbf{k}_{mn}} e^{i\tau_{\mathbf{k}_{mn}}}$. For further simplicity, assume that the grid cell is centered at the origin and η, $\frac{\partial}{\partial t}\eta$ are known at time $t = 0$. Also let all m, n subscripts be implied. Then the approximated surface displacement η^{w} may be rewritten as a finite superposition of linear solutions (discretized in the wavevector domain) with readily determined Fourier coefficients (Tolstov, 1976; Pinkus and Zafrany, 1997):

$$\eta \;\approx\; \eta^{\mathrm{w}}(\boldsymbol{x}_h, t) = \sum_{m,n=-N}^{N} c_{\mathbf{k}}\, e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}} t]} + c^*_{\mathbf{k}}\, e^{-i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}} t]} \tag{2.11}$$

$$\Re\{c_{\mathbf{k}}\} = \frac{1}{2}(c_{\mathbf{k}} + c^*_{\mathbf{k}}) \;=\; \frac{1}{2L^2}\int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \eta(\boldsymbol{x}_h, 0)\, e^{-i[\mathbf{k}\cdot\boldsymbol{x}_h]}\, d\boldsymbol{x}_h \tag{2.12}$$

$$\Im\{c_{\mathbf{k}}\} = \frac{1}{2i}(c_{\mathbf{k}} - c^*_{\mathbf{k}}) \;=\; \frac{1}{2L^2}\int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \frac{1}{\omega_{\mathbf{k}}}\frac{\partial\eta(\boldsymbol{x}_h, 0)}{\partial t}\, e^{-i[\mathbf{k}\cdot\boldsymbol{x}_h]}\, d\boldsymbol{x}_h. \tag{2.13}$$

It then follows that $a_{\mathbf{k}} = 2\,|c_{\mathbf{k}}|$, $\tau_{\mathbf{k}} = \arg(c_{\mathbf{k}})$, and $\langle\eta^{\mathrm{w}}(\boldsymbol{x}_h, t)\rangle_{\boldsymbol{L}_h} = 0$ (since $\eta^{\mathrm{w}}_{\mathbf{k}}$ is horizontally harmonic). It should be noted though that the surface displacement is not a Fourier series in time due to the dispersion relation and in general, $\langle\eta^{\mathrm{w}}(\boldsymbol{x}_{0_h}, t)\rangle_T \neq 0$ for any fixed point $\boldsymbol{x}_{0_h}$ and arbitrary T since

$$\langle\eta^{\mathrm{w}}(\boldsymbol{x}_{0_h}, t)\rangle_T = \frac{1}{T}\int_{t-T/2}^{t+T/2} \eta^{\mathrm{w}}(\boldsymbol{x}_{0_h}, s)\, dt \tag{2.14}$$

$$= \frac{1}{T}\int_{t-T/2}^{t+T/2} \left\{ \sum_{m,n=-N}^{N} \left(c_{\mathbf{k}}\, e^{i[\mathbf{k}\cdot\boldsymbol{x}_{0_h} - \omega_{\mathbf{k}} s]} + \mathrm{c.c} \right) \right\} ds \tag{2.15}$$

$$= \sum_{m,n=-N}^{N} 2\Re\{d_{\mathbf{k}}(t)\}\, \frac{\sin\left(T\omega_{\mathbf{k}}/2\right)}{T\omega_{\mathbf{k}}/2}, \tag{2.16}$$

where $d_{\mathbf{k}}(t) = c_{\mathbf{k}} e^{i[\mathbf{k}\cdot\boldsymbol{x}_{0_h} - \omega_{\mathbf{k}} t]}$ and c.c. denotes the complex conjugate. Although η^{w} is defined deterministically, it can be thought of as a statistically stationary process (in the wide-sense) since the expected mean (for time) is constant ($E\{\eta^{\mathrm{w}}\} = 0$) and the autocorrelation function (for time) is only dependent on one variable ($R(t_1, t_2) = R(t)$ for $t = t_1 - t_2$) (Phillips, 1966; Massel, 1996; Ochi, 1998). To minimize error in this first order approximation, it will be assumed $T \gg \sqrt{2L/\pi g}$ throughout.

Lastly, to sufficiently model wind and swell conditions for SD, L needs to be on the order of 1 km or greater. On a typical $1^\circ \times 1.25^\circ$ latitude-longitude grid, the dimensions of the grid range approximately from 110×140 km (at the equator) to 110×35 km (75° latitude). Capillary waves can be excluded in the summation by ensuring the smallest wavelength is approximately 10 cm, equivalent to $N = O(10^4 L)$ (per km).

2.3.2 Discrete wave spectra

For statistically homogeneous and stationary waves, there is a direct relationship between the expected wave variance (the height deviation squared) and the Fourier transform of the height deviation, magnitude squared, in the frequency and wavevector domain. This latter part is often referred to as the spectral density (Massel, 1996; Ochi, 1998) and can be derived using a modified form of Plancherel's theorem. For simplification, consider the 1D time-frequency relationship for some point $\boldsymbol{x}_{h_0}$ where the surface displacement is ignored outside an interval of length T. Let

$$\eta_T(t) = \begin{cases} \eta(\boldsymbol{x}_{h_0}, t), & |t| \leq T \\ 0, & |t| > T, \end{cases} \tag{2.17}$$

$$\mathcal{F}[\eta_T](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \eta_T(t) e^{-i\omega t} dt. \tag{2.18}$$

Then Plancherel's theorem (Pinkus and Zafrany, 1997) can be used for piecewise continuous η — whether or not it is absolutely and quadratically integrable[3] — to establish a relationship between the variance of Eq. (2.18) and the magnitude square of Eq. (2.17). Taking limits, a general spectral density $\mathcal{S}$ can be defined to satisfy

$$\lim_{T\to\infty} \frac{1}{T} \int_{-\infty}^{\infty} |\eta_T(s)|^2 \, ds = \lim_{T\to\infty} \frac{1}{T} \int_{-\infty}^{\infty} |\mathcal{F}[\eta_T](\omega)|^2 \, d\omega = \int_{-\infty}^{\infty} \mathcal{S}(\omega) d\omega. \tag{2.19}$$

If η is statistically stationary (as previously defined), it can be shown (Ochi, 1998) that

$$\lim_{T\to\infty} \frac{1}{T} |\mathcal{F}[\eta_T](\omega)|^2 = \mathcal{S}(\omega), \tag{2.20}$$

[3] The use of η_T ensures the Fourier transform exists and Eqs. (2.17) and (2.18) are quadratically integrable.

and a discrete frequency form of Eq. (2.19) follows as

$$\lim_{\Delta\omega\to 0} \sum_{i=-\infty}^{\infty} \left(\lim_{T\to\infty} \frac{\left|\mathcal{F}_i[\eta_T^{\mathrm{w}}]\right|^2}{T} \right) \Delta\omega_i = \lim_{\Delta\omega\to 0} \sum_{i=-\infty}^{\infty} \mathcal{S}_i^{\mathrm{w}} \Delta\omega_i, \tag{2.21}$$

where $\mathcal{F}_i$ denotes the Fourier transform for a discrete frequency ω_i of η_T^{w}.

Similarly, a relationship can be derived for the entire domain utilizing the deep–water dispersion relation (and noting η is real):

$$\lim_{T,L\to\infty} \langle \eta(\boldsymbol{x}_h, t)^2 \rangle_{T,\boldsymbol{L}_h} \equiv \iint_{-\infty}^{\infty} G(\mathbf{k}, \omega)\, d\mathbf{k} d\omega \equiv \int_{-\infty}^{\infty} S_{\mathbf{k}}(\mathbf{k})\, d\mathbf{k}, \tag{2.22}$$

where[4]

$$S_{\mathbf{k}}(\mathbf{k}) = 2 \int_0^{\infty} \delta(\omega - \sqrt{gk})\, G(\mathbf{k}, \omega)\, d\omega. \tag{2.23}$$

Using Eq. (2.22), a spectral density estimate now can be defined in the large T and L approach. First note that

$$\begin{aligned} \int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \eta^{\mathrm{w}}(\boldsymbol{x}_h, s)^2\, d\boldsymbol{x} &= \int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \left(\sum_{m,n=-N}^{N} c_{\mathbf{k}}\, e^{i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}} s]} + c_{\mathbf{k}}^*\, e^{-i[\mathbf{k}\cdot\boldsymbol{x}_h - \omega_{\mathbf{k}} s]} \right)^2 d\boldsymbol{x} \\ &= \sum_{m,n=-N}^{N} L^2 \left(2 c_{\mathbf{k}} c_{\mathbf{k}}^* + c_{\mathbf{k}} c_{\text{-}\mathbf{k}}\, e^{-i2\omega_{\mathbf{k}} s} + c_{\mathbf{k}}^* c_{\text{-}\mathbf{k}}^*\, e^{i2\omega_{\mathbf{k}} s} \right), \end{aligned}$$

and

$$\frac{1}{TL^2} \int_{t-\frac{T}{2}}^{t+\frac{T}{2}} \int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \eta^{\mathrm{w}}(\boldsymbol{x}_h, s)^2\, d\boldsymbol{x}_h ds = \sum_{m,n=-N}^{N} 2 c_{\mathbf{k}} c_{\mathbf{k}}^* \left[1 + \frac{\sin(T\omega_{\mathbf{k}})}{T\omega_{\mathbf{k}}} \cos(2\omega_{\mathbf{k}} t) \right].$$

It then follows for L sufficiently large, the **wavevector spectral density** of the surface displacement η can be approximated as a sum of Fourier coefficients of a linear approximation, or

$$\int_{-\infty}^{\infty} \mathcal{S}_{\mathbf{k}}(\mathbf{k}) d\mathbf{k} \;\approx\; \lim_{T,L\to\infty} \langle \eta^{\mathrm{w}}(\boldsymbol{x}_h, t)^2 \rangle_{T,\boldsymbol{L}_h} \;\approx\; \sum_{m,n=-N}^{N} 2 c_{\mathbf{k}} c_{\mathbf{k}}^*. \tag{2.24}$$

[4] If η is real, $G(\mathbf{k}_0, \omega)$ is even (Phillips, 1966).

2.3.3 The cell-averaged SD estimate

With η^{w} in series form, the other series for wave variables and desired forms follow formally:

$$\varphi^{\mathrm{w}}(\boldsymbol{x},t) = \sum_{m,n=-N}^{N} \frac{ic_{\mathbf{k}}\omega_{\mathbf{k}}}{k} e^{kz+i[\mathbf{k}\cdot\boldsymbol{x}_h-\omega_{\mathbf{k}}t]} + \text{c.c.}, \tag{2.25}$$

$$\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},t) = \sum_{m,n=-N}^{N} (k_x,k_y,-ik)\,\frac{c_{\mathbf{k}}\omega_{\mathbf{k}}}{k} e^{kz+i[\mathbf{k}\cdot\boldsymbol{x}_h-\omega_{\mathbf{k}}t]} + \text{c.c.}, \tag{2.26}$$

$$\nabla\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},t) = \sum_{m,n=-N}^{N} (k_x,k_y,-ik)\otimes(ik_x,ik_y,k)\,\frac{c_{\mathbf{k}}\omega_{\mathbf{k}}}{k} e^{kz+i[\mathbf{k}\cdot\boldsymbol{x}_h-\omega_{\mathbf{k}}t]} + \text{c.c.}, \tag{2.27}$$

$$\int_t^s \boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s')ds' = \sum_{m,n=-N}^{N} (ik_x,ik_y,k)\,\frac{c_{\mathbf{k}}}{k} e^{kz+i[\mathbf{k}\cdot\boldsymbol{x}_h]}\left(e^{-i\omega_{\mathbf{k}}s}-e^{-i\omega_{\mathbf{k}}t}\right) + \text{ c.c.}. \tag{2.28}$$

Here, the outer product $\otimes$ emphasizes the tensor rank of $\nabla\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},t)$. It then follows that the difference between the Eulerian and Lagrangian velocities at time s for some fixed initial t is

$$\begin{aligned}
&\left(\int_t^s \boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s')ds'\right)\cdot\nabla\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s)\\
&\quad=\left\{\left[\sum_{m,n=-N}^{N}(ik_x,ik_y,k)\,\frac{c_{\mathbf{k}}}{k}e^{kz+i[\mathbf{k}\cdot\boldsymbol{x}_h]}\left(e^{-i\omega_{\mathbf{k}}s}-e^{-i\omega_{\mathbf{k}}t}\right)+c.c.\right]\right.\\
&\qquad\left.\cdot\;\left[\sum_{m',n'=-N}^{N}\left(\grave{k}_x,\grave{k}_y,-i\grave{k}\right)\otimes\left(i\grave{k}_x,i\grave{k}_y,\grave{k}\right)\frac{\grave{c}_{\mathbf{k}}\grave{\omega}_{\mathbf{k}}}{\grave{k}}e^{\grave{k}z+i\left[\grave{\mathbf{k}}\cdot\boldsymbol{x}_h-\grave{\omega}_{\mathbf{k}}s\right]}+\text{ c.c.}\right]\right\}\\
&\quad=\sum_{m,n,m',n'=-N}^{N}\left\{\left(\grave{k}_x,\grave{k}_y,-i\grave{k}\right)\frac{\grave{c}_{\mathbf{k}}\grave{\omega}_{\mathbf{k}}}{k\grave{k}}e^{(k+\grave{k})z}\left[\vphantom{\int}\right.\right.\\
&\qquad\left(-k_x\grave{k}_x-k_y\grave{k}_y+\grave{k}k\right)c_{\mathbf{k}}e^{i\left[(\mathbf{k}+\grave{\mathbf{k}})\cdot\boldsymbol{x}_h\right]}\left(e^{-i[(\omega_{\mathbf{k}}+\grave{\omega}_{\mathbf{k}})s]}-e^{-i[\omega_{\mathbf{k}}t+\grave{\omega}_{\mathbf{k}}s]}\right)\\
&\qquad\left.\left.+\left(k_x\grave{k}_x+k_y\grave{k}_y+k\grave{k}\right)c^*_{\mathbf{k}}e^{-i\left[(\mathbf{k}-\grave{\mathbf{k}})\cdot\boldsymbol{x}_h\right]}\left(e^{i[(\omega_{\mathbf{k}}-\grave{\omega}_{\mathbf{k}})s]}-e^{i[\omega_{\mathbf{k}}t-\grave{\omega}_{\mathbf{k}}s]}\right)\right]+\text{ c.c.}\right\}.
\end{aligned}$$

The spatial average of the difference over the periodic domain gives

$$\begin{aligned}
&\frac{1}{L^2}\int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2}\left[\left(\int_t^s\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s')ds'\right)\cdot\nabla\boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x},s)\right]d\boldsymbol{x}_h\\
&\qquad=\sum_{m,n=-N}^{N}\left\{(k_x,k_y,ik)\,2c_{\mathbf{k}}c_{\text{-}\mathbf{k}}\omega_{\mathbf{k}}e^{2kz}\left(e^{-i[\omega_{\mathbf{k}}(t+s)]}-e^{-i2\omega_{\mathbf{k}}s}\right)\right.\\
&\qquad\left.+\;(k_x,k_y,-ik)\,2c^*_{\mathbf{k}}c_{\mathbf{k}}\omega_{\mathbf{k}}e^{2kz}\left(1-e^{i[\omega_{\mathbf{k}}(t-s)]}\right)\right\}+\text{c.c.}\,.
\end{aligned}$$

Similarly, integrating in time over the interval $[t - T/2, t + t/2]$, we find

$$\begin{aligned}
\frac{1}{T}\int_{t-T/2}^{t+T/2} &\left\{ \frac{1}{L^2}\int_{-\boldsymbol{L}_h/2}^{\boldsymbol{L}_h/2} \left[\left(\int_t^s \boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x}, s')ds' \right) \cdot \nabla \boldsymbol{u}^{\mathrm{w}}(\boldsymbol{x}, s) \right] d\boldsymbol{x}_h \right\} ds \\
&= \sum_{m,n=-N}^{N} \left\{ 2\omega_{\mathbf{k}} e^{2kz} \left[(k_x, k_y, ik)\, c_{\mathbf{k}} c_{\text{-}\mathbf{k}} \left(\frac{\sin\left(\omega_{\mathbf{k}} T/2\right)}{\omega_{\mathbf{k}} T/2} + \frac{e^{-i2\omega_{\mathbf{k}} t} \sin\left(\omega_{\mathbf{k}} T\right)}{\omega_{\mathbf{k}} T} \right) \right.\right. \\
&\qquad \left.\left. +\ (k_x, k_y, -ik)\, c^*_{\mathbf{k}} c_{\mathbf{k}} \left(1 - \frac{\sin\left(\omega_{\mathbf{k}} T/2\right)}{\omega_{\mathbf{k}} T/2} \right) \right] + c.c. \right\} \\
&\approx \sum_{m,n=-N}^{N} (k_x, k_y, 0)\, 4\, c_{\mathbf{k}} c^*_{\mathbf{k}} \omega_{\mathbf{k}} e^{2kz},
\end{aligned} \tag{2.29}$$

for T sufficiently large ($T \gg \sqrt{2L/\pi g}$). Then the cell-averaged SD centered at the origin for an arbitrary t yields

$$\boldsymbol{u}^S(0, 0, z, t; \boldsymbol{L}_h, T) \approx \sum_{m,n=-N}^{N} 4\, c_{\mathbf{k}} c^*_{\mathbf{k}} \omega_{\mathbf{k}} \mathbf{k} e^{2kz}. \tag{2.30}$$

Now let $\boldsymbol{x}_g$ represent the center of any grid cell (with dimension $\boldsymbol{L}_h$) but with an arbitrary depth z. Then Eq. (2.30) can be generalized as the **cell-averaged SD estimate**:

$$\boldsymbol{u}^S(\boldsymbol{x}_g, t; \boldsymbol{L}_h, T) \approx \sum_{m,n=-N}^{N} 4\, c_{\mathbf{k}} c^*_{\mathbf{k}} \sqrt{gk}\, \mathbf{k}\, e^{2kz}. \tag{2.31}$$

2.3.4 The spectral density SD estimate

Let $\mathcal{S}_{k_r,k_\theta}$ (with $k_r = k$) represent the wavevector spectral density in polar coordinates. Then using the deep-water dispersion relation with the following change of variables,

$$\begin{aligned}
\mathcal{S}_{f\theta}(f, \theta) &= \frac{8\pi^2 f}{g} \mathcal{S}_{k_r k_\theta}\left(k_r = (2\pi f)^2/g, k_\theta = \theta\right), \\
\mathcal{S}_{k_r k_\theta}(k_r, k_\theta) &= k \mathcal{S}_{\boldsymbol{k}}(k_x = k_r \cos k_\theta, k_y = k_r \sin k_\theta),
\end{aligned}$$

the total spectra can be reformulated in terms of the **directional-frequency spectral density**, $\mathcal{S}_{f\theta}$, as

$$\int_{-\infty}^{\infty} \mathcal{S}_{\mathbf{k}}(\mathbf{k})\, d\mathbf{k} = \int_0^{\infty}\int_{-\pi}^{\pi} \mathcal{S}_{k_r k_\theta}(k_r, k_\theta)\, dk_\theta dk_r = \int_0^{\infty}\int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f, \theta)\, d\theta df. \tag{2.32}$$

Using Eq. (2.20), it follows that the cell-averaged SD from Eq. (2.31) can be rewritten in spectral density form (for $L \gg 1km$, $T \gg \sqrt{2L/\pi g}$) as

$$\boldsymbol{u}^S_{2D_h}(\boldsymbol{x}_g, t; \boldsymbol{L}_h, T) \approx \int_{-\infty}^{\infty} 2\sqrt{gk}\,\mathbf{k}\,\mathcal{S}_{\mathbf{k}}(\mathbf{k})d\mathbf{k} \tag{2.33}$$

$$= \frac{16\pi^3}{g}\int_0^{\infty}\int_{-\pi}^{\pi}\Big(\cos\theta, \sin\theta, 0\Big) f^3\mathcal{S}_{f\theta}(f,\theta)e^{\frac{8\pi^2 f^2}{g}z}d\theta df. \tag{2.34}$$

This vector quantity is the spectral density estimate of the leading-order SD for linear surface gravity waves and is the basis for all lower-order spectral density approximations. Here, the horizontally-two-dimensional quantity is termed **2D$_h$-SD** to distinguish it from from later horizontally-one-dimensional approximations.

2.4 Global comparisons of SD

There are numerous nondimensional LM numbers (La, La_t, La_{TKE}, La_{proj}, etc.) used to estimate the vertical velocity of LM and SD is a major component of each. In order to validate any LM parametrization, it is essential to have global estimates of these numbers as well as comparisons of them using different observational and model datasets. Motivated by this, a survey of lower-order SD approximations was conducted and used to estimate and compare global SD using empirical spectra, observations, and models in Webb and Fox-Kemper (2011). These lower-order SD approximations were derived from a unidirectional approximation (see the following section) using spectral moments and empirically-derived relations between them. This further simplification was an essential first step for comparisons since access to 1D spectral data across a wide range of data products was limited and no previous error analyses had been conducted.

2.4.1 The 1D$_h$-SD approximation

Due to a limitation in directional-frequency spectral data from both observations and models, the following unidirectional wave assumption is often used to simplify calculations of SD:

$$\mathcal{S}_{f\theta}(f,\theta) = \delta(\theta - \theta')\mathcal{S}_f(f). \tag{2.35}$$

Here, θ' is used to define the assumed SD direction as $\hat{\boldsymbol{e}}^{\mathrm{w}} = (\cos\theta', \sin\theta')$. This simplifies the interior integral in Eq. (2.34) as

$$\int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f,\theta)\, d\theta = \hat{\boldsymbol{e}}^{\mathrm{w}} \mathcal{S}_f(f), \tag{2.36}$$

and results in a simpler horizontally-one-dimensional (henceforth **1D$_h$** or **1D$_h$-unidirectional**) form of SD and its surface value, given as:

$$\boldsymbol{u}^S_{1\mathrm{D}_h} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{16\pi^3}{\mathrm{g}} \int_0^\infty f^3 \mathcal{S}_f(f) e^{\frac{8\pi^2 f^2}{\mathrm{g}} z}\, df \tag{2.37}$$

$$\boldsymbol{u}^S_{1\mathrm{D}_h}\big|_{z=0} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{16\pi^3}{\mathrm{g}} \int_0^\infty f^3 \mathcal{S}_f(f)\, df \;=\; \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{16\pi^3 m_3}{\mathrm{g}}. \tag{2.38}$$

Notice now that only the third moment[5] of the 1D wave spectrum, m_3, is required to estimate the surface 1D$_h$-SD.

Even though the 1D$_h$-SD approximation is common in literature (Kenyon, 1969; McWilliams and Restrepo, 1999, etc.), it should be pointed out that the assumption of unidirectionality is a strong one that affects both the magnitude and direction of SD. A detailed discussion of the approximation will be delayed until the following section — however it should be noted that typically the direction of SD varies with depth and its magnitude will be overestimated by the unidirectional approximation. Despite this, the assumption is often preferable since access to 2D spectral data has been historically limited.

2.4.2 Lower-order SD approximations

Estimating global SD from observations requires further simplifications since both coverage and spectral data are limited. Satellite radar altimeters provide adequate coverage but only measure wave properties. Buoys calculate wave spectra but are sparsely distributed and data are typically stored in spectral moment form. As a result, lower-order SD approximations are necessary for comparing observational estimates with analytical and model-derived ones.

Traditional measures of wave properties can be defined clearly using 1D spectral moments. The **spectral significant wave height** is a commonly used measure of wave height and is defined

[5] See Appendix A.2 for a definition of moments.

as $H_{m0} = 4\sqrt{m_0}$. Likewise, the ratio of moments $T_n = (m_0/m_n)^{1/n}$ can be used to approximate various wave periods such as the **mean wave period** ($n = 1$) and **zero-crossing wave period** ($n = 2$) (Gommenginger et al., 2003). With these two properties alone, it is possible to estimate the surface $1\mathrm{D}_h$-SD as

$$\boldsymbol{u}^S_{1\mathrm{D}_h}\big|_{z=0} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{\pi^3 \left(16 m_0\right)}{\mathrm{g}\left(m_0/m_3\right)} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{\pi^3 H_{m0}^2}{\mathrm{g} T_3^3}. \tag{2.39}$$

Thus if T_3 were routinely saved in data, it would be straightforward to estimate the surface $1\mathrm{D}_h$-SD. However, T_3 is uncommon in archived data, in comparison to T_1 and T_2, so conversions among wave periods or spectral moments would be valuable.

For a monochromatic spectrum, there is only one wave period, the peak period, and all T_n are equivalent. Replacing T_3 with T_n in Eq. (2.39) yields a surface monochromatic SD approximation which is sometimes used with polychromatic spectra. In Webb and Fox-Kemper (2011), relationships between different moments, period estimates, and SD approximations (both surface and subsurface) were found and tabulated using a wide range of spectra. These relationships included unimodal empirical spectra[6] based on various sea states as well as more complex bimodal spectra generated from NOAA WAVEWATCH III (a third-generation spectral wave model).[7] To account for polychromatic spectra, the following leading order correction to the monochromatic SD approximation (henceforth **spectral-moment-SD**) was proposed:

$$\boldsymbol{u}^S_{\mathrm{mmnt}}\big|_{z=0} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{\mathsf{a}_n \pi^3 H_{m0}^2}{g T_n^3}. \tag{2.40}$$

Values of a_n were calculated for the inverse, first, second, and third moments and are re-tabulated from Webb and Fox-Kemper (2011) in Table 2.1 for convenience. Best estimates using WAVEWATCH III spectra were determined using a linear weighted least-squares fit to minimize the temporal global-mean-square error[8] over an eight year period.

[6] See Webb and Fox-Kemper (2011), Hasselmann et al. (1973), and Pierson and Moskowitz (1964) for a description of the different types of empirically-derived spectra.

[7] See Chapter 3 for a discussion of third-generation spectral wave models and Appendix B.1 for details of the NOAA WAVEWATCH III implementation.

[8] The global mean is an area-weighted mean that adjusts for grid cell size changes due to latitude.

It is clear that different spectral shapes produce different values. Indeed, the reason why multiple 1D spectra were used was to exemplify a realistic range of values. The inverse-moment wave-period estimate is not reliable for 1D$_h$-SD, as it depends sensitively on wave spectrum shape (differences between them are greater than 40%). However, the first and second-moment wave-period estimates differ among the spectra by about 23% and 13% respectively. While this uncertainty is not negligible, the spectral-moment-SD approximation is an improvement to the monochromatic one and differences (of a_1 and a_2 among the spectra) are modest when compared to the discrepancies found between different data sources (as will be discussed in the next subsection). The same method can be applied to estimate subsurface values of SD, given a pair of spectral moments and a prescribed spectral shape (see Webb and Fox-Kemper, 2011).

2.4.3 Summary of spectral-moment-SD approximations and global comparisons

The reliability of the spectral-moment-SD approximation may be judged both by comparisons between different spectral shapes and by comparison to discrepancies between available wave data products. In Webb and Fox-Kemper (2011), a detailed comparison of the spectral-moment-SD approximation is made with the 1D$_h$-SD approximation. At the surface, it was found that the most reliable approximation is based on the second moment, which is usually quite accurate away from coastal areas with a normalized root-global-mean-square error of roughly 10%. For reference, the monochromatic approximation (based on the inverse-moment) used for the preliminary LM parametrization performed the poorest and had normalized errors higher than 100%.

In addition, global SD magnitudes were calculated in Webb and Fox-Kemper (2011). The goal was to use three substantially different estimates of SD – from satellites, a data-assimilating model, and a forward model – to see how reasonable estimates of SD differ. Since 1D spectral information from two of the data sets were not available, the a_2-spectral-moment-SD approximation was used for comparisons. In summary, SD estimates from the different data products disagreed by $30-50\%$, roughly equally divided between discrepancies in significant wave height and wave period. Based on the spectral shapes studied, the a_2-spectral-moment-SD errors would need to be four times larger to rival the contributions from the significant wave height and period discrepancies found among the data products.

While the a_2-spectral-moment-SD is an improvement to the more common monochromatic approximation (McWilliams et al., 1997), there is structure to the error patterns (see Fig. 2.2) and an accurate and full reconstruction of the wave spectrum is recommended to remove the systematic error and fully diagnose SD. At the time of writing, it was concluded that there was no presently well-accepted way to determine both surface and subsurface global SD. It was hoped that the analysis presented would guide future wave data collection and aid in the determination of a global SD climatology and variability.

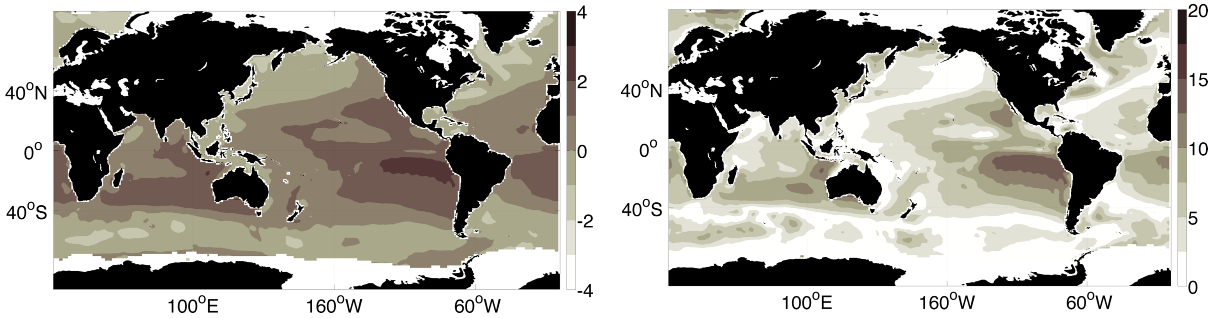
2.5 Comparison of 1D$_h$- and 2D$_h$-SD approximations

Estimating the error in the 1D$_h$-SD approximation without access to the directional-frequency spectra can be challenging since the degree of spreading is unknown and the presence of multidirectional waves can be hidden. These features affect both the magnitude and direction of SD in a nonlinear fashion that is sensitive with depth. By assumption, the 1D$_h$-SD approximation ignores these features and as a result overestimates the magnitude of SD. In addition, the assumed direction (wind seas direction) is often misaligned with the actual direction of SD. These differences can be substantial and are not readily quantifiable.

Recently, there has been a trend toward using the 2D$_h$-SD approximation for calculations that are sensitive to SD despite the necessary exponential increase in computation and storage. While this is appropriate, the role that the missing physics play is still unclear and there is no adequate way yet to compare calculations using 1D$_h$- and 2D$_h$-SD approximations confidently. Here, a first attempt has been made to better understand and quantify the influence of directional spreading and multidirectional waves on SD.

Table 2.1: Proposed coefficients for the (surface) spectral-moment-SD approximation using different mean periods from different spectra. Dots and brackets indicate truncation of an analytical solution and temporal and global means respectively.

	a_{-1}	a_1	a_2	a_3
Monochromatic	1	1	1	1
JONSWAP (empirical)	2.34	1.84	1.49	1
PM (empirical)	2.700...	1.970...	1.537...	1
$\langle$WAVEWATCH III$\rangle_{G,T}$	3.34	2.31	1.69	1



(a) $\left\langle \left(\boldsymbol{u}^S_{\mathrm{1D}_h} - \boldsymbol{u}^S_{\mathrm{mmnt}}\right)|_{z=0}\right\rangle_T$ (cm/s)

(b) $\left\langle \left(\boldsymbol{u}^S_{\mathrm{1D}_h} - \boldsymbol{u}^S_{\mathrm{mmnt}}\right)|_{z=0}\right\rangle_T / \left\langle \boldsymbol{u}^S_{\mathrm{1D}_h}|_{z=0}\right\rangle_{G,T}$ (%)

Figure 2.2: Eight year mean (1994-2001) of the residual and relative-residual surface SD magnitudes between the $1\mathrm{D}_h$-SD and a_2-spectral-moment-SD approximations. Figures are reproduced from Webb and Fox-Kemper (2011).

2.5.1 Pitfalls of the unidirectional assumption

Several simple examples are given below to illustrate some of the challenges of using 1D wave spectra to calculate SD. The last example will provide some insight on how the 1D$_h$-SD approximation can be improved.

2.5.1.1 Multidirectional waves and SD magnitude

When multidirectional waves are present, the magnitude of SD depends on the angle of incidence between the different waves. A monochromatic spectrum for a peak frequency f_p, wave amplitude a, and direction $\tilde{\theta}$, can be defined as

$$\mathcal{S}_{\mathrm{mono},f\theta}(f,\theta) = \frac{a^2}{2}\,\delta(f-f_p)\,\delta(\theta-\tilde{\theta}). \tag{2.41}$$

The resulting SD for the simplest wave is then

$$\boldsymbol{u}^S_{\mathrm{mono}}\big|_{\boldsymbol{x}_0} = \left(\cos\tilde{\theta}, \sin\tilde{\theta}, 0\right)\frac{8\pi^3 a^2 f_p^3}{\mathrm{g}}\,\exp\left[-\frac{8\pi^2\,|z_0|}{\mathrm{g}}f_p^2\right]. \tag{2.42}$$

To illustrate the importance of multidirectional spectra on SD, consider two monochromatic waves passing through the same point from different directions (see Fig. 2.3). For simplicity, let the peak frequency and amplitude of the waves be the same. The bichromatic spectra can then be defined for some mean direction $\bar{\theta}$ and angle of incidence θ' as

$$\mathcal{S}_{\mathrm{bi},f\theta}(f,\theta) = \frac{a^2}{2}\delta(f-f_p)\left[\delta(\theta-\bar{\theta}-\theta') + \delta(\theta-\bar{\theta}+\theta')\right]. \tag{2.43}$$

The resulting SD is then

$$\begin{aligned}\boldsymbol{u}^S_{\mathrm{bi}}\big|_{\boldsymbol{x}_0} &= \left(\cos(\bar{\theta}+\theta') + \cos(\bar{\theta}-\theta'), \sin(\bar{\theta}+\theta') + \sin(\bar{\theta}-\theta'), 0\right)\frac{8\pi^3 a^2 f_p^3}{\mathrm{g}}\,\exp\left[-\frac{8\pi^2|z_0|}{\mathrm{g}}f_p^2\right]\\ &= 2\cos\theta'\,\boldsymbol{u}^S_{\mathrm{mono}}\big|_{\boldsymbol{x}_0}.\end{aligned} \tag{2.44}$$

In terms of approximations, this is equivalent to

$$\boldsymbol{u}^S_{2D_h}\big|_{\boldsymbol{x}_0} = \cos\theta'\,\boldsymbol{u}^S_{1D_h}\big|_{\boldsymbol{x}_0} \tag{2.45}$$

and in general, the 1D$_h$-SD approximation will overestimate SD whenever both wind seas and swell are present.

2.5.1.2 Multidirectional waves and SD direction

In the previous example, the resulting SD magnitude follows simple vector addition. This generalizes for any n-chromatic wave, $\mathcal{S}_{n\text{-chro},f\theta} = \sum_{i=1}^{n} \frac{a_i^2}{2}\delta(f-f_i)\delta(\theta-\theta_i)$, as

$$\boldsymbol{u}^S_{n\text{-chro}}\Big|_{\boldsymbol{x}_0} = \sum_{i=1}^{n} (\cos\theta_i, \sin\theta_i, 0)\, A_i(z_0) \tag{2.46}$$

where $A_i(z) = \frac{8\pi^3 a_i^2 f_i^3}{\mathrm{g}} \exp\left[-\frac{8\pi^2|z|}{\mathrm{g}} f_i^2\right]$ for some amplitude and peak frequency a_i and f_i. Again for simplicity, consider a bichromatic wave but with different peak frequencies such that $a_1 = a_2$, $\theta_1 \neq \theta_2$ and $f_2 = e^1 f_1$. Then for the depth

$$z = \frac{-\mathrm{g}}{8\pi^2\left(e^2-1\right)f_1^2}, \tag{2.47}$$

the SD direction will be the directional average of both waves, $\left(\theta_1+\theta_2\right)/2$. However, for any depth above or below this value, the dominant SD direction will be determined by either the higher- or lower-frequency monochromatic wave respectively. This will be discussed in more detail later, however the direction of SD should be expected to align with wind seas at the surface and swell at greater depths, whenever both are present.

2.5.1.3 Wave spreading and SD magnitude

In addition to multidirectional waves, the degree of wave spreading also plays a large role in determining the magnitude of SD. To illustrate, consider the two following hypothetical unidirectional spectra, one with spreading and one without:

$$\begin{aligned}
\mathcal{S}_{1,f\theta}(f,\theta) &= \sqrt{\frac{2}{\pi}} \exp\left[-2\theta^2\right]\mathcal{S}_f(f),\\
\mathcal{S}_{2,f\theta}(f,\theta) &= \delta(\theta)\mathcal{S}_f(f).
\end{aligned}$$

Notice that both spectra are normalized in θ and directed in the $\hat{e}_1$ direction. However due to spreading, the magnitudes are not equivalent and $\left\|\boldsymbol{u}_1^S\right\|_{\ell^2} = 0.882\left\|\boldsymbol{u}_2^S\right\|_{\ell^2}$.

In all three examples, the magnitude and/or direction of SD is affected by the use of directionally-averaged 1D wave spectra. In the first two examples, the differences between 1D$_h$-

and 2D$_h$-SD are dependent upon properties of a random sea (e.g., how often are the waves multidirectional, what are typical frequency and directional differences, etc.). In this example however, the differences are systematic (i.e., determined by the spreading function) and not a random process. It will be later shown that the error due to wave spreading in 1D$_h$-SD approximations can be removed if the directional (spread) distribution is known.

2.5.2 Directional distribution and SD

To aid understanding, formal relations between 1D$_h$- and 2D$_h$-SD approximations are defined. The relations will be used in the following section to develop an improved 1D$_h$-SD approximation.

2.5.2.1 Defining a generic directional distribution

The 1D frequency spectrum is defined such that

$$\mathcal{S}_f(f) = \int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta)\, d\theta. \tag{2.48}$$

This is equivalent to splitting the directional-frequency spectrum into a frequency component ($\mathcal{S}_f$) and a directional distribution (D) and integrating in θ. For a fixed frequency f_α, a fixed directional distribution can be defined as

$$D(\theta; f_\alpha) = \begin{cases} \frac{\mathcal{S}_{f\theta}(f_\alpha,\theta)}{\int_{-\pi}^{\pi}\mathcal{S}_{f\theta}(f_\alpha,\theta)\, d\theta}, & \mathcal{S}_{f\theta}(f_\alpha,\theta) \neq 0, \\ \frac{1}{2\pi}, & \mathcal{S}_{f\theta}(f_\alpha,\theta) = 0. \end{cases} \tag{2.49}$$

Assuming D can be defined continuously in f, the directional distribution has the property

$$\int_{-\pi}^{\pi} D(f,\theta)\, d\theta = 1, \tag{2.50}$$

for every f, and the directional-frequency and frequency spectrum can be related by

$$\int_0^\infty \int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta)\, d\theta\, df = \int_0^\infty \int_{-\pi}^{\pi} D(f,\theta)\, \mathcal{S}_f(f)\, d\theta\, df = \int_0^\infty \mathcal{S}_f(f)\, df. \tag{2.51}$$

2.5.2.2 Defining a generic directional component

Due to the vector component $(\cos\theta, \sin\theta, 0)$ in the integrand, the behavior of SD is not always intuitive. To identify the role it plays more concretely, let the combined integral of any generic directional distribution and vector component, or

$$\boldsymbol{H}(f) = \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, D(f,\theta)\, d\theta, \tag{2.52}$$

be termed the **directional-SD-component**. Then the 2D$_h$-SD can be rewritten as

$$\boldsymbol{u}^S_{2D_h} = \frac{16\pi^3}{g} \int_0^\infty \boldsymbol{H}(f) f^3 S_f(f) \exp\left[\frac{8\pi^2 f^2}{g} z\right] df. \tag{2.53}$$

Notice that for all unidirectional waves (i.e., $\mathcal{S}_{f,\theta}(f,\theta) = \delta(\theta - \bar{\theta})\mathcal{S}_f(f)$), $\|\boldsymbol{H}(f)\|_{\ell^2} = 1$ for all $f \in \mathbb{R}_+$. In this context, the 1D$_h$-SD can be thought of as a 2D$_h$-SD approximation with $\|\boldsymbol{H}\|_{\ell^2} = 1$ and $\boldsymbol{H}/\|\boldsymbol{H}\|_{\ell^2} = \left(\cos\bar{\theta}, \sin\bar{\theta}, 0\right)$ for some assumed direction $\bar{\theta}$. The directional-SD-component can now be used to quantify the effect of wave spreading.

2.5.3 The 1D$_h$-DHH-SD approximation

Here, an intermediary approximation (between 1D$_h$- and 2D$_h$-SD) is defined to differentiate the importance of spreading and multidirectional waves in SD. This 1D$_h$ approximation will use an empirically-derived directional distribution to include the effects of spreading.

2.5.3.1 The DHH spreading function

Based on observational data, Donelan et al. (1985) derived a frequency-dependent spreading function. Here, the Donelan-Hamilton-Hui (henceforth abbreviated DDH) spreading function has

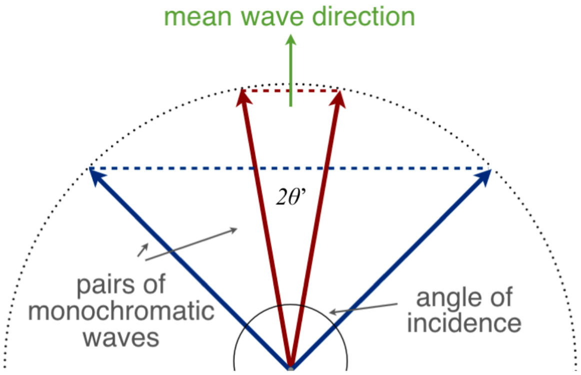


Figure 2.3: Here, pairs of monochromatic waves (red and blue) are shown traveling about a mean direction $\bar{\theta}{=}\pi/2$ with a total directional difference (for each pair) of $2\theta'$. Only the y vector components of the bichromatic waves contribute to SD.

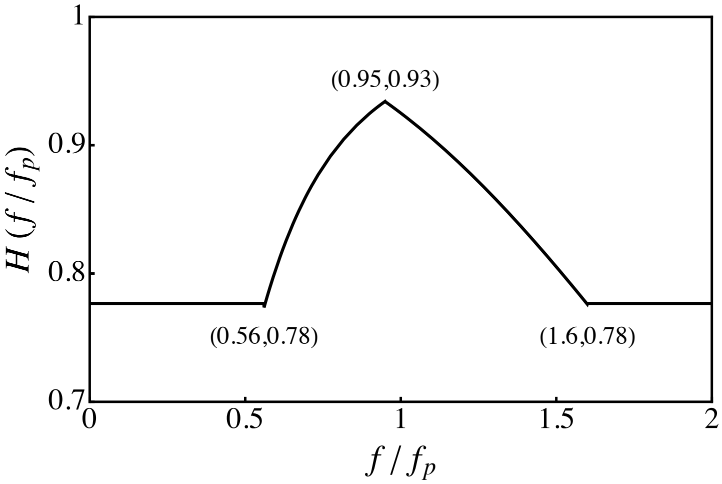


Figure 2.4: The magnitude of the DHH directional-SD-component.

been normalized and the domain shifted to be periodic for θ as

$$D_{\mathrm{DHH}}(f,\theta;f_p,\bar\theta) =$$
$$\frac{\beta(f/f_p)}{2\tanh[\beta(f/f_p)\pi]}\begin{cases}\operatorname{sech}^2[\beta(f/f_p)\,\theta]\,, & \bar\theta(f)=0,\\ \operatorname{sech}^2\left[\beta(f/f_p)\left(\theta-\bar\theta(f)\right)\right], & -\pi+\left|\bar\theta(f)\right|\le\operatorname{sgn}[\bar\theta(f)]\,\theta\le\pi,\\ \operatorname{sech}^2\left[\beta(f/f_p)\left(\theta-\bar\theta(f)+2\pi\operatorname{sgn}[\bar\theta(f)]\right)\right], & -\pi\le\operatorname{sgn}[\bar\theta(f)]\,\theta\le-\pi+\left|\bar\theta(f)\right|.\end{cases} \tag{2.54}$$

Here, f_p is the peak frequency, $\bar\theta(f)\in[-\pi,\pi]$ is the mean direction for a particular f, and β is given by

$$\beta(r)=\begin{cases}2.61\,r^{1.3}, & 0.56<r\le 0.95,\\ 2.28\,r^{-1.3}, & 0.95<r<1.6,\\ 1.24, & \text{otherwise.}\end{cases} \tag{2.55}$$

2.5.3.2 The DHH directional-SD-component

Unfortunately, calculating the directional-SD-component for the DHH spreading function is fairly complicated and requires the use of hypergeometric functions with complex arguments (Luke, 1969). To simplify, a Padé approximate of order [2/2] (Bender and Orszag, 1978) has been used to approximate the exact solution within a relative error of 3×10^{-3}, given by

$$\boldsymbol{H}_{\mathrm{DHH}}(f;f_p,\bar\theta)=\left(\cos\bar\theta,\sin\bar\theta,0\right)\begin{cases}\frac{0.52f_p^2-3.3f_pf+8.9f^2}{f_p^2-3.4f_pf+8.9f^2}, & 0.56<f/f_p\le0.95,\\ \frac{0.98f_p^2-0.19f_pf+0.0058f^2}{f_p^2-0.26f_pf+0.12f^2}, & 0.95<f/f_p<1.6,\\ 0.777, & \text{otherwise.}\end{cases} \tag{2.56}$$

Notice that the DHH directional-SD-component requires knowledge of the peak frequency and the mean direction is still a function in terms of frequency (i.e. the mean direction is not necessarily constant for all frequencies).

The magnitude of the DHH directional-SD-component is depicted in Fig. 2.4. Notice that it is bounded by $0.777\le\|\boldsymbol{H}_{\mathrm{DHH}}\|_{\ell^2}\le0.934$. This implies 1D$_h$-SD approximations may be overestimating 2D$_h$-SD by as much as 30% due to spreading alone.

2.5.3.3 An improved 1D$_h$-SD estimate

As mentioned in the example in Section 2.5.1.3, the 1D$_h$-SD approximation can be improved by using an empirically-derived directional distribution to approximate the full 2D spectra for SD calculations. Here, the Padé-approximated, DHH directional-SD-component (Eq. 2.56) will be used to modify Eq. (2.37) as

$$\boldsymbol{u}^S_{\mathrm{DHH}} = \hat{\boldsymbol{e}}^{\mathrm{w}} \frac{16\pi^3}{\mathrm{g}} \int_0^\infty H(f/f_p) f^3 S_f(f) e^{\frac{8\pi^2 f^2}{\mathrm{g}} z}\, df, \tag{2.57}$$

where $H(f/f_p) = \|\boldsymbol{H}_{\mathrm{DHH}}\|_{\ell^2}$ and $\hat{\boldsymbol{e}}^{\mathrm{w}}$ is a chosen mean direction. The approximation requires knowledge of the peak frequency but this is fairly straight-forward to calculate in a model implementation. As with any 1D$_h$-SD approximation, the choice of $\hat{\boldsymbol{e}}^{\mathrm{w}}$ is not always clear and will be discussed in Section 2.5.5. The improved 1D$_h$ estimate will be termed **1D$_h$-DHH-SD** to distinguish it from the unidirectional **1D$_h$-SD** approximation. The improved estimate is not limited to the use of the DHH directional-SD-component. An alternative directional-SD-component is derived in Appendix A.4 and is based on a modification to the DHH spreading function.

2.5.4 Analysis of SD magnitudes using the 1D$_h$-DHH-SD approximation

Here, the 1D$_h$-DHH-SD will be used with prescribed wave spectra (i.e., empirically formulated) and observational and model output to differentiate the effects of spreading and multidirectional waves in SD. For surface SD magnitudes, it will be shown that estimates from all three types are in close agreement and that it is possible to correct 1D$_h$-SD for directional spreading by multiplying by a constant.

2.5.4.1 Analysis using 1D$_h$-DHH-SD with prescribed wave spectra

Here, spectra from three different empirically-derived formulas will be used to examine the role of spreading. Since empirical spectra require knowledge of the peak frequency, e-folding depths will be used instead of depth to remove the peak frequency dependence and analyze the results.

Let $n \in \mathbb{R}_+$ and $z_n = -ng/(8\pi^2 f_p^2)$. Then for a monochromatic wave,

$$\left\| \boldsymbol{u}^S_{\text{mono}}(z_n) \right\|_{\ell^2} = e^{-n} \left\| \boldsymbol{u}^S_{\text{mono}}(z{=}0) \right\|_{\ell^2} . \tag{2.58}$$

In Table 2.2, three different peak frequencies are chosen (from high to low) to illustrate a typical range of e-folding depths. Notice that at each e-folding depth, the depth for the lowest frequency is approximately 50 times larger than the highest. This will aid an intuitive understanding shortly.

In addition to JONSWAP and PM empirical spectra (previously used in Section 2.4.2), prescribed spectral shapes from Donelan et al. (1985) (abbreviated DHH as well) are also used to analyze the effects of spreading. Fetch-limited and fully-developed cases are chosen for the DHH spectra to mirror JONSWAP and PM shapes respectively. See Webb and Fox-Kemper (2011) for details and a visual comparison of the spectra.

To examine the reduction in SD magnitude due to directional spreading, ratios of 1D$_h$-DHH-to 1D$_h$-SD for e-folding depths are presented in Table 2.3 (for select values) and in Fig. 2.5. For the empirical spectra selected, there is approximately a 10−25% magnitude loss when using 1D$_h$-DHH-SD instead of 1D$_h$-SD. The loss is greatest for all spectra at the surface and flattens out for higher e-folding depths. This is expected since the exponential component of SD will filter mostly higher frequencies initially as depth is increased (see Table 2.2). Eventually however, the majority of SD will be comprised of spectra with frequencies less than half of the peak and the DHH directional-SD-component will no longer influence the magnitude.

2.5.4.2 Analysis using 1D$_h$-DHH-SD with observational data

Since comparisons with 2D$_h$-SD are not possible with empirical spectra, it is necessary to use other data. Wave buoys have commonly been used to measure vertical velocities to infer the frequency spectra at a point (Holthuijsen, 2007). It is also possible for some buoys to estimate wave direction and thus infer the full directional-frequency spectra. This data (both 1D and 2D) is often used for validation of and assimilation in forecast spectral wave models (Tolman, 2009).

Here, observational data from a CDIP[9] directional buoy, stationed in deep water in the

[9] Coastal Data Information Program, Scripps Institution of Oceanography.

Table 2.2: Example e-folding depths $|z_n|$ (m) for the peak frequencies $f_p = 0.05$, 0.16, 0.34 (Hz).

f_p	T_p	$\|z_{0.01}\|$	$\|z_{0.5}\|$	$\|z_1\|$	$\|z_2\|$	$\|z_3\|$
0.34	2.94	0.0107	0.537	1.07	2.15	3.22
0.16	6.25	0.0485	2.43	4.85	9.71	14.6
0.05	20	0.497	24.8	49.7	99.4	149

Table 2.3: Ratio of **1D$_h$-DHH-** to **1D$_h$-SD** magnitudes using empirical spectra for various e-folding depths. Ratios with DHH1 (fetch-limited) and DHH2 (fully-developed) are not provided at z_0 since the spectra is undefined at the surface (see Section 2.5.4.1).

	z_0	$z_{0.01}$	$z_{0.5}$	z_1	z_2	z_3
JONSWAP	0.812	0.820	0.874	0.893	0.908	0.913
DHH1	-	0.803	0.875	0.896	0.911	0.915
PM	0.799	0.805	0.854	0.877	0.898	0.906
DHH2	-	0.795	0.853	0.876	0.897	0.905

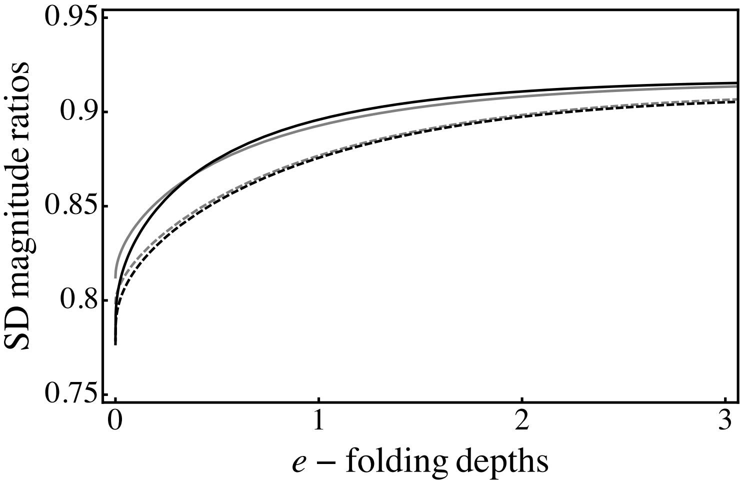


Figure 2.5: Ratios of **1D$_h$-DHH-** to **1D$_h$-SD** magnitudes using empirical spectra for continuous e-folding depths: **JONSWAP** (gray solid), **PM** (gray dashed), **fetch-limited DHH** (black solid), and **fully-developed DHH** (black dashed).

northeastern Pacific Ocean,[10] is used to compare 1D$_h$-, 1D$_h$-DHH-, and 2D$_h$-SD approximations. The 2D spectral data was generated in 30 minute intervals and a 26 month period (2010/7/1–2012/8/31) was selected for comparison.[11] Casual inspection of the 2D spectral data shows that both directional spreading and multidirectional waves are naturally present.

Occasionally, differences between the approximations are starkly noticeable. In Fig. 2.6, a snapshot of SD magnitudes are displayed for the first 10 m of the three different approximations. In the absence of multidirectional waves, SD has an exponentially decaying profile. In this figure interestingly, the surface magnitude of the 2D$_h$-SD is half its value at 10 m. This is can be attributed to the presence of multidirectional waves (likely wind seas opposing a swell) since the 1D$_h$ approximations do not exhibit this behavior.

From examples presented in Section 2.5.1, it is clear that both directional spreading and multidirectional waves can affect the magnitude of SD. To determine if one is more influential, median ratios of the different SD approximations have been calculated for the two year period and displayed in Fig. 2.7. The shaded interval indicates two-thirds of the distribution centered about the median. The ratio of 1D$_h$-DHH-SD to 1D$_h$-SD has been displayed for comparison with the empirical spectra. While it is unclear if the results from empirical spectra fall within the shaded blue region (due to the use of an e-folding depth), the trend is similar with the largest loss at the surface. Since the calculations were limited to depths of 10 m however, it is also unclear if the observational curve will flatten out as well.

Common observational ratios between the 1D$_h$ and 2D$_h$ approximations are illustrated in the other two curves, 2D$_h$-SD to 1D$_h$-SD and 2D$_h$-SD to 1D$_h$-DHH-SD. The (two-third centered) distributions about the median are large in both and are likely caused by multidirectional waves, due to asymmetry and a smaller distribution for the 1D$_h$-DHH-SD to 1D$_h$-SD ratio. In Webb and Fox-Kemper (2011), it was identified that 1D$_h$-SD tended to overestimate the 2D$_h$-SD approximation in the model data analyzed by about 33% at the surface. The unidirectional overestimation is

[10] Datawell directional buoy (O'Reilly et al., 1996); Ocean Station Papa 166 (50°N, 145°W).

[11] A $O(f^{-5})$ spectral tail is used for later comparison with WAVEWATCH III model output (see Appendicies A.5 and B.2).

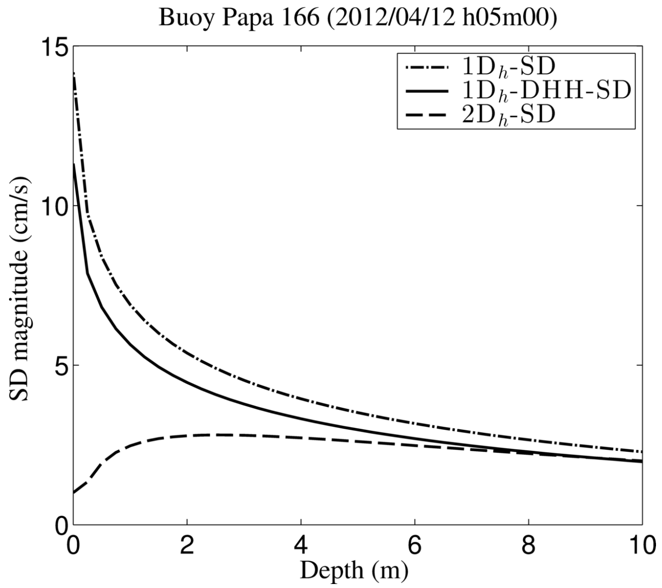


Figure 2.6: Observational buoy data: a snapshot of **SD magnitudes** at depth; the $2D_h$-SD approximation indicates the presence of multidirectional waves.

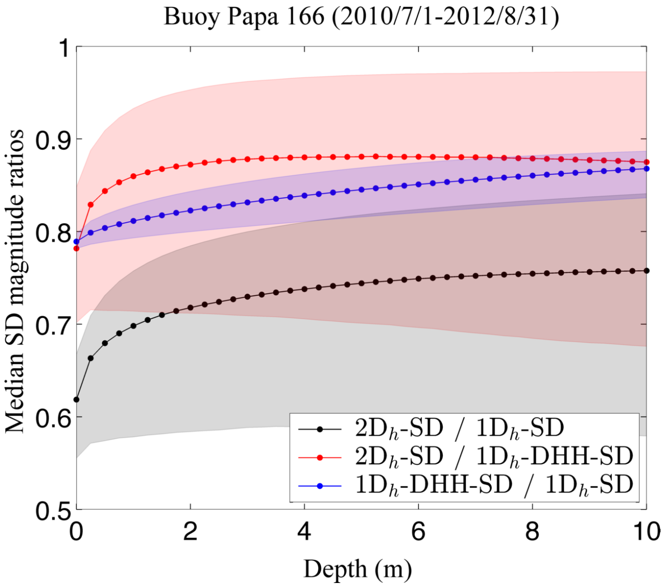


Figure 2.7: Observational buoy data: **median SD magnitude ratios** with the two-thirds centered distribution shaded.

nearly double in these observations at 62%. In comparison, the $1D_h$-DHH-SD overestimates the $2D_h$-SD by less than 28% at the surface and 14% after 3 m. This is still a sizable observational difference but it is a definite improvement. Further discussion on the role of directional spreading and multidirectional waves will be delayed until Section 2.5.6.

2.5.4.3 Analysis using $1D_h$-DHH-SD with model data

While buoys can provide superb spectral data, the data is sparse spatially and it is useful to compare the results with global model-generated data. A 1-year 2D spectral data set[12] has been generated here by WAVEWATCH III and is used to compare differences in SD approximations.

In the observational data analyzed, the majority of uncertainty (in the $1D_h$ approximations) is likely due to the presence of multidirectional waves. To explore why, scatter plots[13] have been generated in Fig. 2.8 to help identify correlations. The dependent values, here surface SD magnitudes (m/s), are shaded by their density distribution. The regions are colored red, green, and blue by the highest 0−30%, 31−60%, and 61−90% centered distributions respectively.

A comparison of Figs. 2.8a and 2.8b, surface magnitudes of $2D_h$-SD (y-axis) versus $1D_h$-SD and $1D_h$-DHH-SD (x-axes) respectively, reveals that the sources of error can be cleanly separated. The shaded regions in both figures are approximately equivalent but oriented about different slopes. In Fig. 2.8b, the upper part of the shaded regions aligns well with the line $y = x$ and indicates that the difference in slopes between the figures is due to directional spreading. It should be pointed out that the alignment in Fig. 2.8b is not guaranteed[14] and the agreement is a testament of the observational studies of Donelan et al. (1985).

Errors due to multidirectional waves are also present. In both figures, the (unnormalized) density distribution normal to the slope of orientation is sharply peaked near the upper part and then gradually descends (in the normal direction with the negative y component). This sharp

[12] Previous analysis of the spectral-moment-SD approximations used an 8-year 1D spectral data set. See Appendix B.1 for details of the 2D spectral data set.

[13] The scatter plots use the total raw (unweighted) data.

[14] As an example, the slope could have easily been higher than unity if the directional distribution had been too diffuse for higher frequencies.

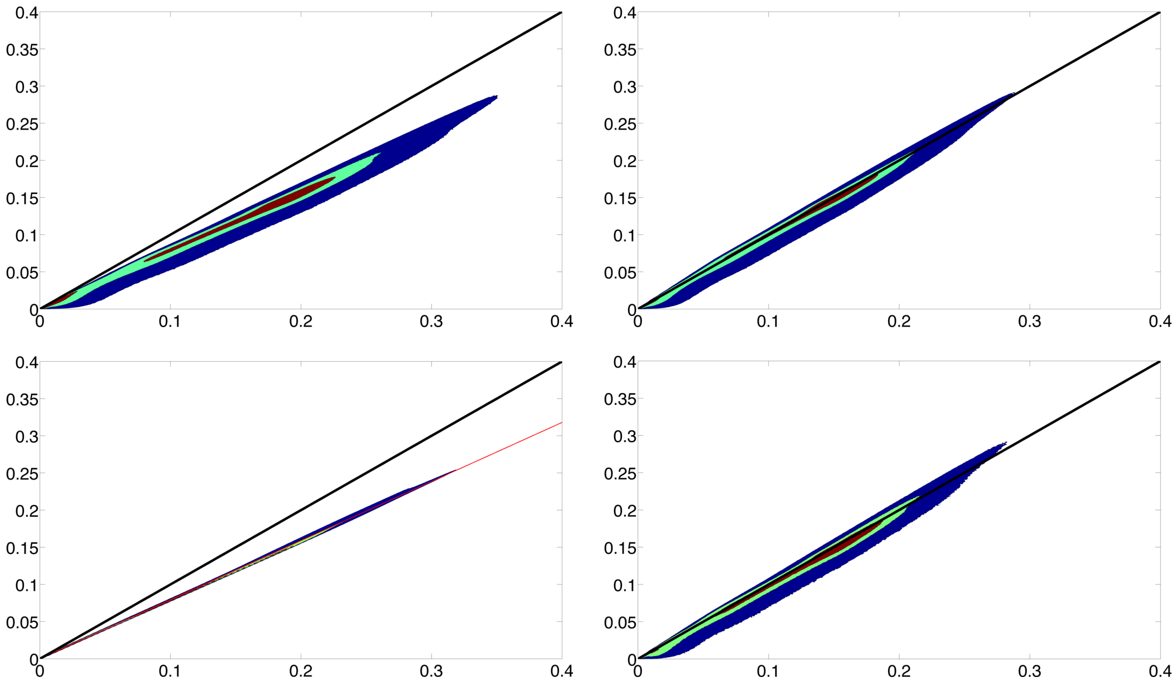


Figure 2.8: Density-shaded scatter plots generated from one year of global model data (WAVEWATCH III). The colors red, green, and blue indicate the highest 0−30%, 31−60%, and 61−90% centered distributions respectively. Surface magnitude (m/s) comparison of (a) **2D$_h$-SD** (y-axis) versus **1D$_h$-SD** (x-axis), (b) **2D$_h$-SD** (y-axis) versus **1D$_h$-DHH-SD** (x-axis), (c) **1D$_h$-DHH-SD** (y-axis) versus **1D$_h$-SD** (x-axis), and (d) **2D$_h$-SD** (y-axis) versus $\boldsymbol{m}\times$**1D$_h$-SD** (x-axis). Here, $m = 0.795$ is the slope of the red line in (c).

peak is expected since the ocean surface is not predominately a mix of wind seas and swell. In addition, the asymmetry is expected as well. Unlike with directional spreading, the presence of multidirectional waves will only cause an overestimation by the 1D$_h$ approximations and never the opposite.

In Fig. 2.8c, surface magnitudes of 1D$_h$-DHH-SD (y-axis) versus 1D$_h$-SD (x-axis), the correlation holds as well. Here the density distribution normal to the slope of orientation is singularly peaked, which is expected since the magnitudes are invariant to the angle of incidence between the multidirectional waves. In addition, the slope of the line the scatter plot falls on (plotted in red with $m = 0.795$) is in close agreement with the values derived from empirical data (0.799, 0.812) and in observations (0.789). This implies that multiplying the surface 1D$_h$-SD magnitudes by approximately 0.8, might be a good rule of thumb to correct for directional spreading (see Fig. 2.8d). This could be very useful since surface magnitudes can be calculated without full knowledge of the 1D wave spectra (as shown in Section 2.4.2). This approach could be extended to include the effects of multidirectional waves but would probably only be useful in a specific region where the statistics of the wave field were well-known.

2.5.5 Analysis of SD direction using the 1D$_h$-DHH-SD approximation

Up until this point, the discussion has largely focused on SD magnitude and not direction. Without full knowledge of the 2D wave spectra, it is not immediately clear what the actual direction should be, particularly for subsurface calculations. At the surface, approximations typically assume the direction of SD is the same as the wind sea. But what about subsurface directions? Should it align with the direction of the wind, the mean wave direction,[15] or a combination of both? Here, this topic is lightly addressed using the same approach as the last section.

In the example in Section 2.5.1.2, the direction of the 2D$_h$-SD (for a bichromatic wave with different peak frequencies) varied with depth, with a preference for the directions of the higher and lower frequencies for (relatively) shallower and deeper depths respectively. This insight is explored

[15] See Appendix A.3.

further in Figs. 2.9 and 2.10 to determine how well the 2D$_h$-SD aligns with the 10 m surface wind and mean (surface) wave directions at various depths.

In the scatter plots, the 2D$_h$-SD aligns best with 10 m surface wind directions at the surface. This result is expected since the higher frequencies of the wind seas contribute the most to the surface SD magnitude. However, this correlation no longer holds as depth is increased. As the higher frequencies are filtered by the exponential component, the direction of 2D$_h$-SD quickly begins to reorientate with the mean wave direction. At a depth of 1 m, the correlations for both wind and mean wave directions are already comparable and by 9 m, the 2D$_h$-SD direction is in near complete agreement with the mean wave direction.

These results are consistent with the notion that wind seas and swell dominate SD for surface and deeper subsurface depths respectively. If the intermediate subsurface SD direction is important however, a spectral wave model is required to calculate this nonlocal, nonlinear vector component.

2.5.6 A qualitative treatment of error in the 1D$_h$-SD approximation

One of the main goals of this investigation is to identify the effects of directional spreading and multidirectional waves on SD. Using three different types of spectra, analyses of the approximations suggest that directional spreading errors are both systematic and removable. Errors due to the interaction of local and nonlocal waves however, appear to be random and add a large degree of uncertainty in both 1D$_h$ approximations. Further quantification of both error types would be useful but only a qualitative treatment will be given here.

In Fig. 2.8c, the scatter plot comparison of 1D$_h$-DHH-SD versus 1D$_h$-SD indicates that the addition of directional spreading does not increase uncertainty in the improved 1D$_h$ approximation. This implies that the physical processes may be independent of each other. In fact, further inspection of Fig. 2.7 reveals that the magnitude ratios, 2D$_h$-SD to 1D$_h$-DHH-SD and 1D$_h$-DHH-SD to 1D$_h$-SD, are largely uncorrelated since the following relation holds to within a few percent:

$$\left\langle \frac{\|\boldsymbol{u}^S_{2\mathrm{D}_h}\|}{\|\boldsymbol{u}^S_{1\mathrm{D}_h}\|} \right\rangle_T = \left\langle \frac{\|\boldsymbol{u}^S_{2\mathrm{D}_h}\|}{\|\boldsymbol{u}^S_{\mathrm{DHH}}\|} \frac{\|\boldsymbol{u}^S_{\mathrm{DHH}}\|}{\|\boldsymbol{u}^S_{1\mathrm{D}_h}\|} \right\rangle_T \approx \left\langle \frac{\|\boldsymbol{u}^S_{2\mathrm{D}_h}\|}{\|\boldsymbol{u}^S_{\mathrm{DHH}}\|} \right\rangle_T \times \left\langle \frac{\|\boldsymbol{u}^S_{\mathrm{DHH}}\|}{\|\boldsymbol{u}^S_{1\mathrm{D}_h}\|} \right\rangle_T . \tag{2.59}$$

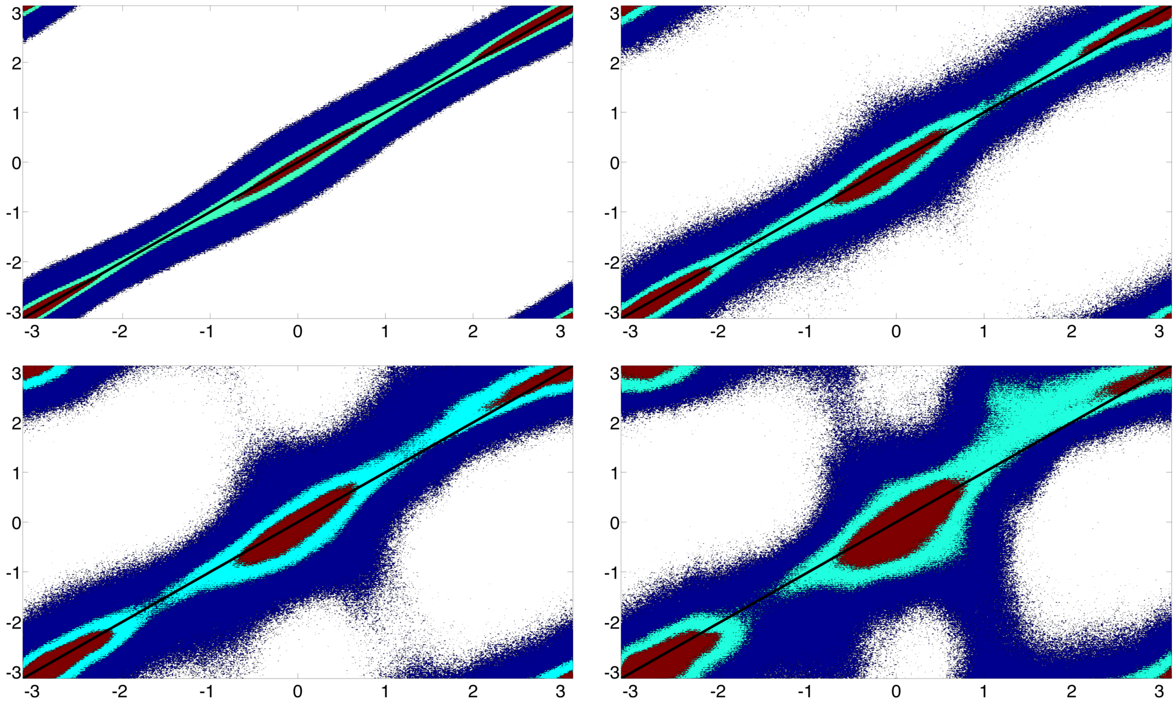


Figure 2.9: Density-shaded scatter plots of the **2D$_h$-SD** (y-axis) versus **10 m surface wind** (x-axis) directions (rad) for different depths (m): (a) z=0, (b) z=1, (c) z=3, and (d) z=9. See Fig. 2.8 for an explanation of the colors.

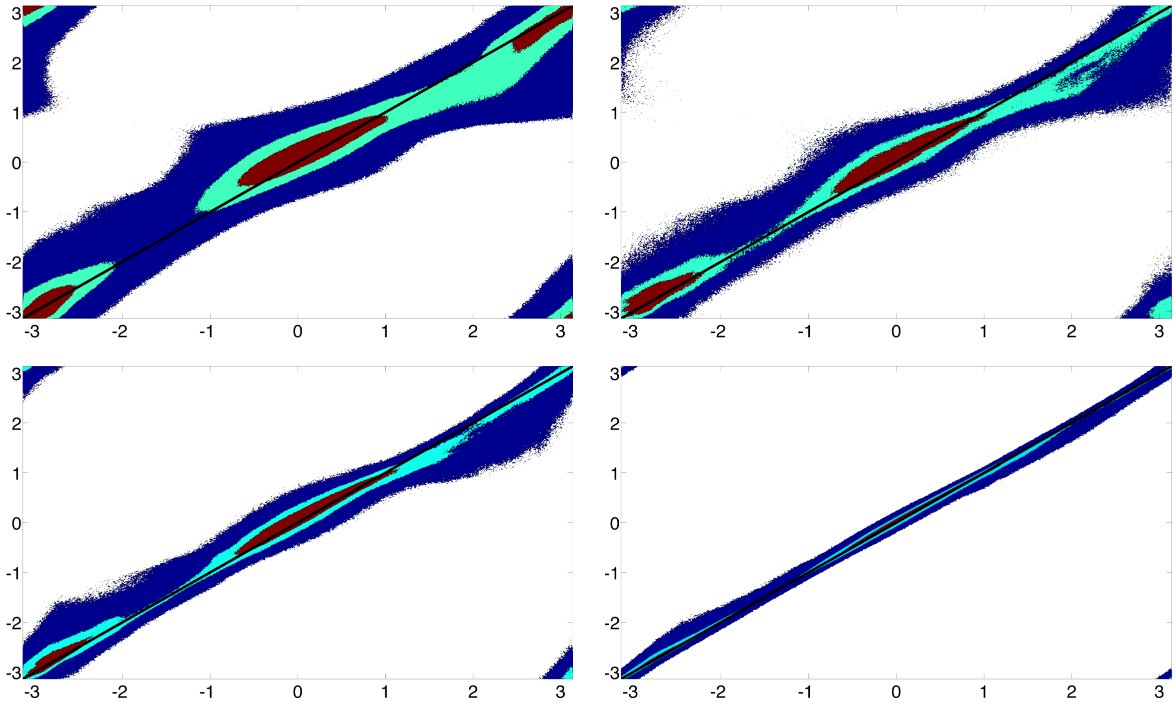


Figure 2.10: Density-shaded scatter plots of the **2D$_h$-SD** (y-axis) versus **surface mean wave** (x-axis) directions (rad) for different depths (m): (a) z=0, (b) z=1, (c) z=3, and (d) z=9. See Fig. 2.8 for an explanation of the colors.

Since both are largely uncorrelated, both can be compared side by side to measure the effects of directional spreading and multidirectional waves separately. Unfortunately, a clear comparison between observational ratios in Fig. 2.7 is not possible since the centered two-thirds distribution is so large for multidirectional waves. However, a comparison of model results is possible at the surface at least and Figs. 2.8b and 2.8c suggest that the errors due to directional spreading will mostly dominate, which agrees with the median curves in Fig. 2.7.

Chapter 3

Spectral wave modeling

3.1 Introduction to spectral wave modeling

A wide range of surface wave phenomenon can be found in the oceans. These can be roughly organized in temporal or physical scale (descending) as trans-tidal waves, tides, storm surges, tsunamis, seiches, infra-gravity waves, wind-generated waves, and capillary waves. The time and length scales of these vary several orders of magnitude with periods of days for tides down to a tenths of a second for capillary waves (see Fig. 3.1) (Holthuijsen, 2007).

Historically there has been much interest in predicting wind-generated waves, which predominantly make up the sea state. The gravity restoring force is characterized by periods shorter than 30 s but greater than 0.25 s (wave lengths of 0.1−1500 m) (Holthuijsen, 2007). They can be separated into two groups, **wind sea**, which are generated locally by wind, and **swell**, which have propagated outside the generation area and have evolved into waves with longer wavelengths and periods.

There are several approaches to modeling wind-generated waves that are based on the scale of interest. On small scales of roughly 10−1000 m, hydrostatic equations of motion can be used to fully model the waves deterministically. This is known as the **phase-resolving** approach. On intermediate scales of roughly 0.1−10 km, phase-resolving quickly becomes computationally prohibitive and a statistical approach is needed. In the **phase-averaging** approach, wind-generated waves are treated as a random process and certain assumptions are made to simplify the modeling. In general, the waves are assumed to have small amplitudes, be away from shallow water, and be statistically homogeneous and stationary (Holthuijsen, 2007). On large basin to global scales, the statistical properties of the waves on intermediate scales are assumed to vary smoothly with location and the generation, propagation, and evolution of wave spectra are modeled deterministically by a **spectral wave modeling** approach. See Fig. 3.2 for an illustration of the concept and the following subsection for more details. Time scales are on the order of hours to days and the latest formulation of the models, third-generation, are typically used to forecast the sea state for commercial and recreational interests. Up until recently, running a third-generation spectral wave model on longer climatological time scales has been computationally infeasible and only statistical wave properties such as significant wave height or period have been calculated using empirical relations or parameterizations.

Here, our primary interest is in extending the capabilities of spectral wave modeling to include longer climatological time scales in order to calculate a prognostic wave field for GCMs. This field is necessary to parameterize LM accurately and will help improve the modeling of air-sea interactions and sea ice formation (Squire, 2007). There are several challenges that hinder implementation though. The first and foremost is that operational forecast third-generation models require significant computational resources and have similar running costs to an entire GCM ocean component. In addition, the governing equations of these models are singular at the poles and require an ice boundary in the northern hemisphere to run. This could be remedied by using a nested polar version[1] of the model and progress is being made on this front (Li, 2012) – but until then, studying an ice-free climate scenario with a global prognostic wave field is not possible.

Running a GCM with a cost-effective prognostic wave field is a challenge and is explored in detail in this chapter. The beginning of the chapter includes an overview of an operational forecast third-generation model, NOAA WAVEWATCH III (Tolman, 2009), with performance benchmarking and progress on the GCM-coupled version. The rest of the chapter is devoted to analyzing the singular governing equations and exploring possible non-singular alternatives.

[1] This requires a new model with the singularities shifted to the equator.

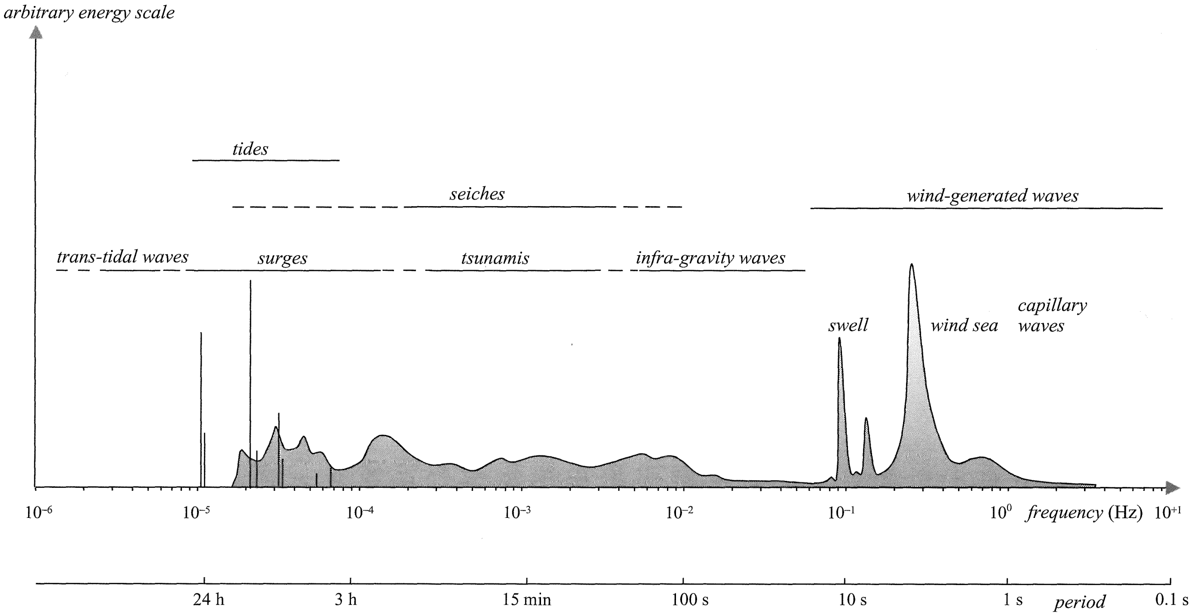


Figure 3.1: Illustration of wave spectra from different types of ocean surface waves. Figure is reproduced from Holthuijsen (2007).

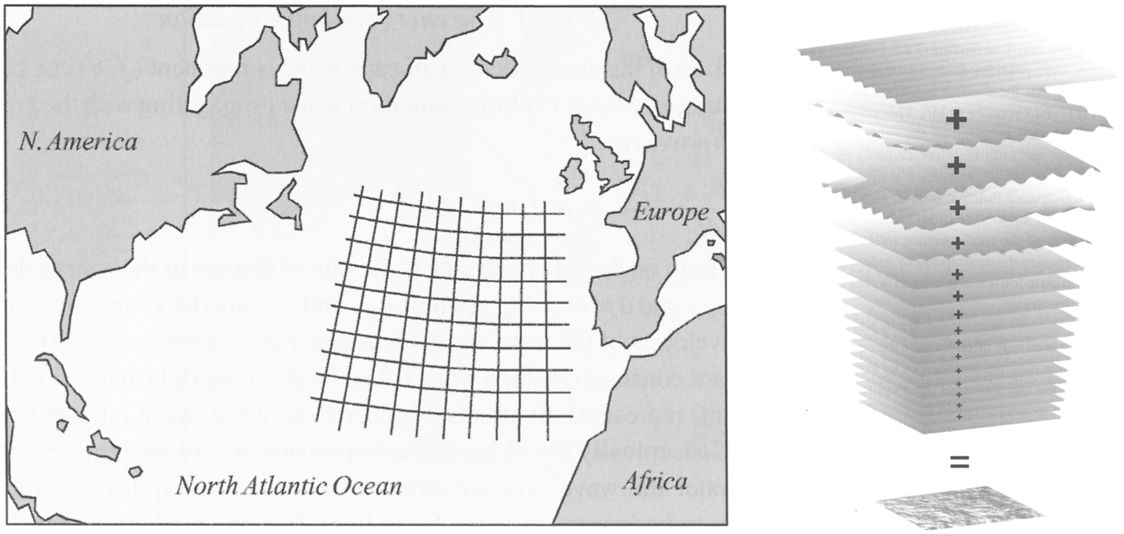


Figure 3.2: Example of a spectral model approach. The random sea of each gridded region in (a) is Fourier decomposed in (b). The statistical differences between neighboring gridded regions are assumed to be small enough such that evolution of wave energy can be modeled by a PDE. Figures are reproduced from Holthuijsen (2007).

3.1.1 Third-generation spectral wave models

As previously mentioned, spectral wave models solve the statistical characteristics of the wave spectrum in a deterministic manner (Holthuijsen, 2007). These models solve an action balance equation with weakly nonlinear sources and sinks to model the generation, dissipation, and evolution of the local spectra of a random sea (Komen et al., 1994). Spectral wave model development over the past 30 years has been dominated by the treatment of these sources and a thorough overview would fill volumes (Komen et al., 1994; Massel, 1996; Janssen, 2004). However, different models can be loosely categorized by their treatment of nonlinear wave-wave interactions and earlier models have either neglected these (first-generation) or have parameterized them (second-generation) (Massel, 1996). In contemporary third-generation wave models, the nonlinear interactions are fully modeled with a 2D wave spectrum.

The **spectral balance equation** is more commonly known as the **wave action balance equation** and is similar to the Boltzmann transport equation found in statistical mechanics (Willebrand, 1975). In the equation, the spectral properties of the physical system are a local property of the spatial domain. As a result, it is a numerically challenging equation to model due to its high dimensionality. Instead of solving directly for the local wave spectrum,[2] the balance equation solves an adiabatic invariant,[3] termed **wave action**, that is defined as

$$\mathcal{W}(\mathbf{k};\boldsymbol{x}_h,t) = \frac{\mathrm{g}\,\mathcal{S}_{\mathbf{k}}(\mathbf{k};\boldsymbol{x}_h,t)}{\sigma(\mathbf{k};\boldsymbol{x}_h)}, \tag{3.1}$$

where σ is the intermediate-water dispersion relation and $\mathcal{S}_{\mathbf{k}}$ is the position and time-dependent (2D) wavevector spectrum. On a 2 x 2 dimensional slab, the balance equation can be written as

$$\partial_t\mathcal{W} + \nabla_{\boldsymbol{x}_h}\cdot(\mathcal{W}\,\nabla_{\mathbf{k}}\Omega) - \nabla_{\mathbf{k}}\cdot(\mathcal{W}\,\nabla_{\boldsymbol{x}_h}\Omega) = \text{Sources}, \tag{3.2}$$

where Ω is a Doppler-shifted dispersion relation and the term 'Sources' encompasses the non-kinematic physics of the waves (generation, dissipation, nonlinear interactions, etc.) (Komen et al.,

[2] Recall from Section 2.3.2 that the wave spectrum is defined here as the time-step averaged, magnitude squared, truncated Fourier transform of the surface height deviation, for each grid cell in the directional-frequency or wavevector domain.

[3] The physical property or quantity remains invariant to slow changes in the system (Whitham, 1974).

1994). See the following section for a detailed discussion of the governing equation.

Third-generation models typically use structured grids in both the spatial and spectral domains (See Fig. 3.3 for examples). Inputs can include surface wind velocities, ocean currents, air and sea interface temperature differences, sea ice concentrations, bathymetry and topography data, and tidal information. In addition to 1D and 2D wave spectra, outputs can include spectral moment quantities (significant wave height, period, etc.) as well as various analyses (wave peak, direction, etc.). Data assimilation also plays a role and data from buoys and altimeters are typically incorporated to improve forecasts.

As previously mentioned, third-generation models can be computationally demanding to run. This is due to the extra spectral domain which typically adds $600-1000$ more unknowns to calculate at each spatial grid cell. The difference in unknowns can be of several orders of magnitude when compared with vertically-layered 3D models ($\sim 30-40$ layers) of similar horizontal spatial resolution. Because of this, global $0.5-1^\circ$ resolution forecasts require massive computing power and need to be run in parallel to produce medium range forecasts. Since it can takes weeks for swell generated by a storm to transverse the oceans, restart capabilities are necessary for models to avoid lengthy spin-up times to account for distant swell. The two main third-generation models in use for global scales are NOAA WAVEWATCH III (Tolman, 2009) and ECMWF WAM (Janssen, 2008).[4]

3.1.2 Coupling WAVEWATCH III to NCAR CESM

In Section 1.1.2, a preliminary LM parameterization showed promise in reducing a Southern Ocean shallow mixed-layer bias in the NCAR CCSM 3.5 model. However, further testing in NCAR CCSM 4 indicated that use of a prognostic wave field is crucial in order to remove uncertainty from the climatological-based parametrization (Webb et al., 2013). To remedy this, a modified form of WAVEWATCH III (version 3.14) is being added and coupled to the NCAR Community Earth System Model (CESM). This third-generation spectral wave model will provide the backbone of

[4] The organizational acronyms stand for the National Oceanic and Atmospheric Administration and the European Centre for Medium-Range Weather Forecasts respectively.

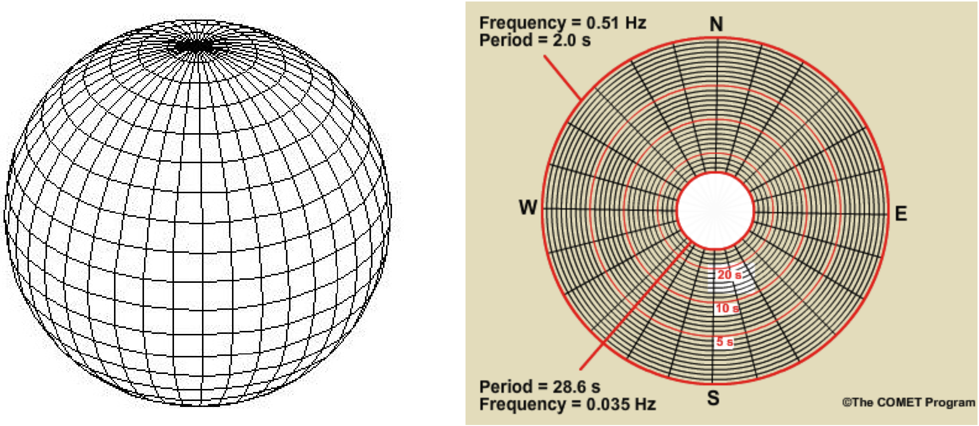


Figure 3.3: Examples of third-generation model grids: (a) spatial latitude-longitude grid spaced equally and non-equally in latitude and longitude (respectively); (b) spectral directional-frequency grid spaced evenly and logarithmically in direction and frequency (respectively).

the new LM parameterization and coupled output from it will eventually be used to improve air-sea interactions and sea ice formation in the GCM.

WAVEWATCH III is generally regarded as the state-of-art in spectral wave modeling and includes extensive physics and parameterizations. It uses a third-order accurate, fractional time stepping and finite difference scheme. In addition to data assimilation, it also has nested grid capabilities to run higher resolution grids within larger coarser ones. Despite these advantages, WAVEWATCH III is ultimately a forecast wave model and modifying it for climate purposes will require addressing two challenges: increasing speed and overcoming polar singularities. It is exactly these challenges that spawned prototype development of a new meshless spectral wave model for climate modeling purposes in Chapter 4. However, until an operational version or another more suitable alternative exists, a modified version of WAVEWATCH III is being used for coupling (see Appendix B.1 for configuration and coupling details).

3.1.2.1 WAVEWATCH III computational costs

Before discussing the previously mentioned challenges, it is useful to know how computational costs are allocated in WAVEWATCH III and how it scales with spatial resolution. In general, the largest costs are attributable to spatial and temporal interpolation of input data, the use of nonlinear source terms, and conversion of output data. A simple analysis of cost versus spatial resolution was performed and is displayed in Fig. 3.4 for three different types of runs: a model run with sources, a model run without sources but with input interpolation, and a baseline run without sources or input.[5] To simplify analysis, the same spectral resolution[6] and number of time steps were used for each different resolution. Since WAVEWATCH III uses a variable time step, the latter was approximately achieved by specifying an arbitrary minimum time step for all runs near or above the maximum CFL value. To verify that the results were not too sensitive to choice of time step, a larger minimum time step (double the original) was also used and is presented as well.

[5] Conversion of output data was not tested since this feature is not used (all output is coupled to the GCM).

[6] The grid consists of 25 frequency and 24 directional bins with an initial and cutoff frequency of 0.0418 and 0.411 respectively.

Using a least squares fit of five different global resolutions (from very coarse to standard), the following cost scaling law was determined based on the number of ocean grid cells ($N_{\boldsymbol{x}}$):

$$\mathrm{Cost}(N_{\boldsymbol{x}}) = \alpha N_{\boldsymbol{x}}^{1.07}, \tag{3.3}$$

for an arbitrary machine-dependent α and $N_{\boldsymbol{x}} \approx \frac{2}{3}(nX \times nY)$.[7] Essentially, the cost is linearly dependent on $N_{\boldsymbol{x}}$ and increasing the time step size by a factor of two has little effect compared with the other run types. Interestingly, processing input and calculating sources approximately doubled and quadrupled the cost respectively. The massive computing power needed to run WAVEWATCH III is often attributed to the need to calculate nonlinear source terms and we can see that running the model with both input and sources is effectively the same as running the model without at double the resolution. However, the importance of nonlinear source terms diminishes somewhat if we account for a changing time step due to grid resolution. Since the cost will change roughly by a factor of $\sqrt{N_{\boldsymbol{x}}/N_{\boldsymbol{x}0}}$ ($N_{\boldsymbol{x}0}$ is the appropriate resolution for the original time step chosen), the adjusted cost will be proportional to $N_{\boldsymbol{x}}^{3/2}$ and implies that the addition of input and sources will be the same as running the model without at an increased resolution of approximately 1.6. While it is now clear that the model scales quadratically with spatial resolution on a single processor, it is not clear how it scales with multiple processors in parallel or with changes to spectral resolution and further benchmarking is needed.

3.1.2.2 Benchmarking WAVEWATCH III for coupling

Since it is not uncommon for climate runs to span several hundred years, a high parallel model throughput of 30 or more simulated years per computational day is ideal. To speed up the model, WAVEWATCH III will be coupled to the CESM on an individual time step to remove the need to interpolate input data. Even as such, a coarsened version of the model is still necessary in order to achieve the high throughput. To estimate the optimal coupled grid for coupling, benchmarking of the uncoupled wave model was performed on two different machines: a Westmere SGI at NASA

[7] nX and nY are the number of longitudinal and latitudinal cells respectively.

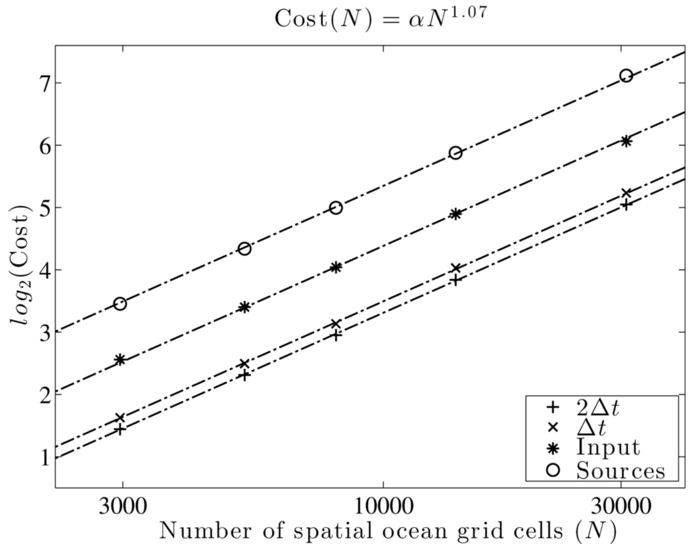


Figure 3.4: A general comparison of WAVEWATCH III cost versus spatial resolution using the same number of time steps and a fixed spectral grid ($25_f \times 24_\theta$). In reverse order of the legend, the four different model runs are (1) with sources, (2) without sources but with input interpolation, (3) without sources and input, and (4) without sources or input but with a larger time step.

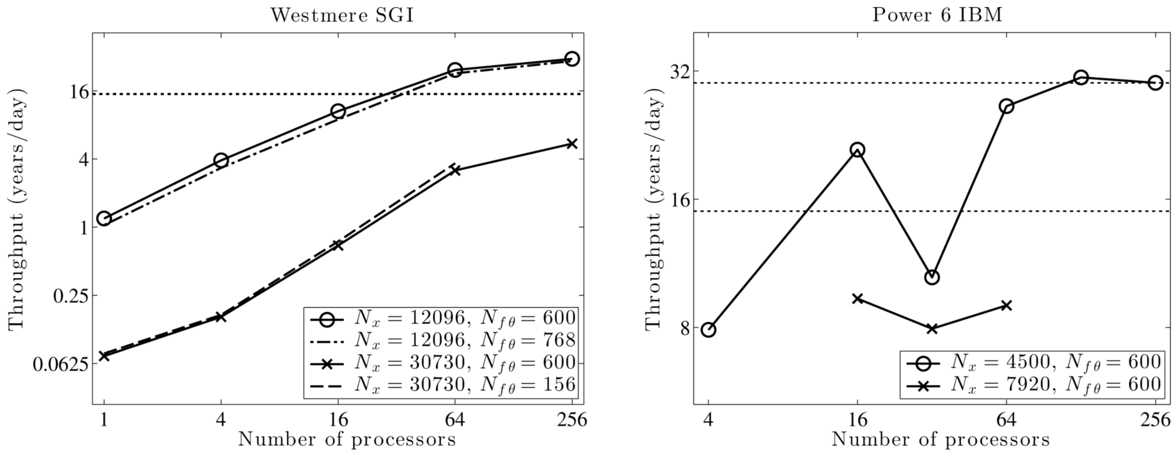


Figure 3.5: Mean WAVEWATCH III grid performance results with benchmarking targets on two different machines. Performance is measured in the number of simulated years per day of running. Benchmarking was performed on (a) NASA Pleaides and (b) NCAR Bluefire on several different spatial ($N_{\boldsymbol{x}}$) and spectral ($N_{f\theta}$) grids. The following spatial lat-lon grids were tested: 1° × 1.25° ($N_{\boldsymbol{x}}$=30730), 1.9° × 2.5° ($N_{\boldsymbol{x}}$=12096), 2.4° × 3° ($N_{\boldsymbol{x}}$=7920), and 3.2° × 4° ($N_{\boldsymbol{x}}$=4500). In addition, the following spectral frequency-direction grids were tested: 32 × 24 ($N_{f\theta}$=768), 25 × 24 ($N_{f\theta}$=600), and 13 × 12 ($N_{f\theta}$=156).

and an IBM Power 6 at NCAR. Benchmarking was performed initially on the faster Westmere SGI to determine the ideal spatial-to-spectral grid ratio and placement of northern and southern ice boundaries. Followup benchmarking was performed on the IBM Power 6 to determine its performance in relation to other components of the NCAR CCSM and CESM.

Benchmarking results are shown in Fig. 3.5. On the Westmere SGI, it was found that adjusting the spectral resolution ($N_{f\theta}$) did not significantly affect model throughput as compared with the spatial resolution ($N_{\boldsymbol{x}}$). This is due to how the coupled domains are split for parallelization in WAVEWATCH III. However, since coarsening the spectral grid quickly rendered the spectral output useless (See Fig. 3.6 for an example), a lower standard grid of $25_f \times 24_\theta$ ($N_{f\theta}$=600) was chosen. A minimum throughput of 15 simulated years per day of running on the Westmere SGI was achieved by reducing from the standard global resolution of $1° \times 1.25°$ ($N_{\boldsymbol{x}}$=30730) to $1.9° \times 2.5°$ ($N_{\boldsymbol{x}}$=12096). On the IBM Power 6, a much coarser grid of $3.2° \times 4°$ ($N_{\boldsymbol{x}}$=4500) was required to reach the minimum and ultimately ideal throughput. In addition, a comparison of machines shows that the (parallelization) efficiencies vary widely between them. On the Westmere SGI, the efficiency roughly scales as $P^{-1/3}$ for P processors up until about 64 and then quickly approaches zero. On the IBM Power 6, the efficiency does not scale as clearly due to a communication lag between nodes (starts at P=32). Even so, the efficiency is also poor and quickly approaches zero for larger P with a recommended max of P=128.

3.1.2.3 The coupled wave model

Using the benchmarking results as a guide, a modified version of WAVEWATCH III (version 3.14) is now coupled to the NCAR CESM 1.2 (Webb et al., 2013). The coupled wave model (WAVE) uses a coarse $3.2° \times 4°$ latitude-longitude grid with latitudinal boundaries of ±78.4° (see next paragraph). The CESM coupler handles all spatial interpolation to and from WAVE and the wave model is coupled on a 30 minute time step (delayed one step) to remove any need for interpolation by the model. Coupled inputs include surface interface temperatures, surface wind velocities, ocean currents, sea ice grid cell concentrations, and mixed layer depths. At the moment, output is limited but will eventually include SD and a new nondimensional LM number (sensitive to wind-wave alignment) to be used for the LM parametrization. Installation testing of WAVE (on Bluefire) has been completed and sample output with relative differences from an uncoupled control run are presented in Fig. 3.7. The actual relative differences are closer to single precision and are higher in the figure ($\sim 10^{-3}$) due to rounding in WAVEWATCH III's output generator. Once the output is fully coupled, additional testing will be necessary to ensure the coarsened model still agrees reasonably well with observational data. While the resolution is less than desired, it is hoped that the prognostic wave field will remove sensitivities in the LM parameterization and help improve other model physics such as air-sea flux calculations and sea ice formation.

Before closing, it should be mentioned that the second challenge of coupling a third-generation model to a GCM is largely unaddressed in the new LM parametrization. Since WAVEWATCH III is both advectively and directionally singular at the poles, it requires the use of northern and southern ice boundaries to run. These polar singularities are due to the numerical methods employed and singularities in the projected version of the wave action balance equation used. While this does make the model ill-suited for studying future polar-ice-free scenarios, there are no current alternatives and mitigation is necessary.[8] Removing the polar singularities from future spectral wave models will require both a nonsingular form of the spherical wave action balance equation and different numerical methods. Both of these issues are discussed further in Section 3.4 and Chapter 4 respectively.

3.2 Wave action balance equation

As previously mentioned, the wave action balance equation is the governing equation used in spectral wave models. It is a similar to the Boltzmann transport equation and uses a coupled spatial-spectral domain to solve an equivalent form of the local wave spectrum for each grid cell location. Several key assumptions are necessary for this large-scale modeling approach and we will

[8] To mitigate, optimal ice lines were determined from Westmere SGI benchmark tests. The ice lines both maximized the region covered and minimized the need for prohibitively small time steps due to shrinking grid cells.

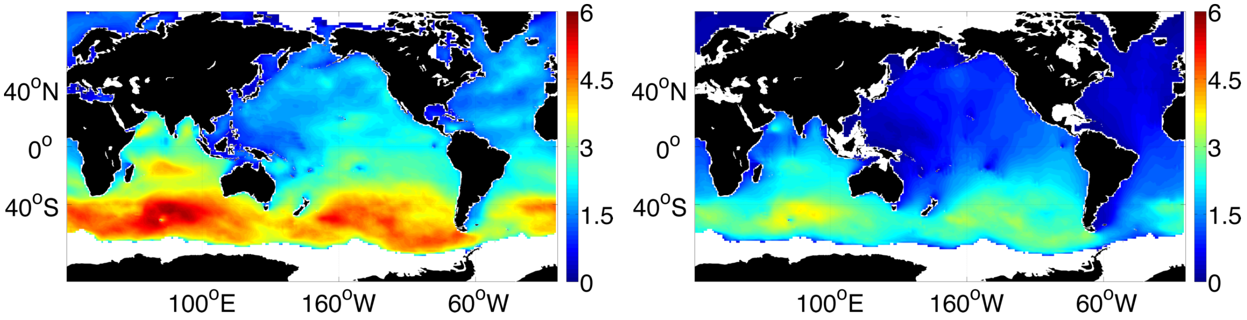


Figure 3.6: An example comparison of significant wave height using a normal ($25_f \times 24_\theta$) and coarsened ($13_f \times 12_\theta$) spectral grid on a standard spatial lat-lon grid (1° × 1.25°).

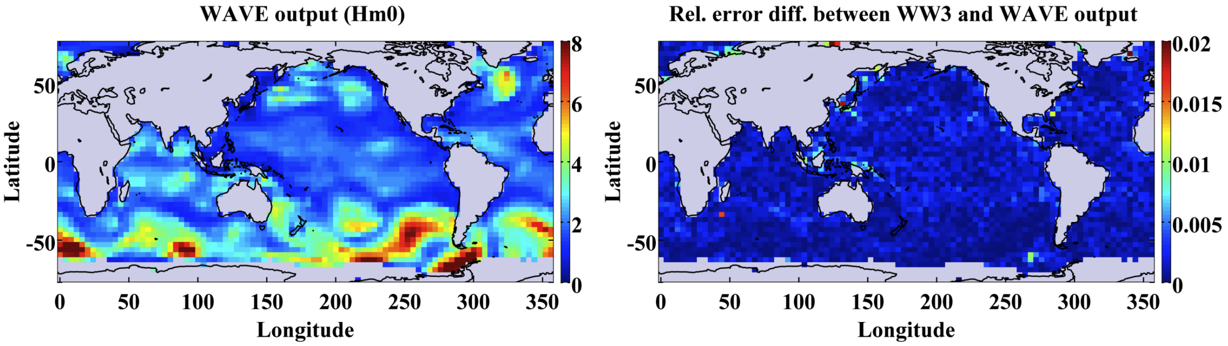


Figure 3.7: Comparison test of the coupled wave model (WAVE) with an uncoupled WAVEWATCH III (WW3) on NCAR Bluefire. Sample output on the new grid is after a 1 day spin-up with seeded spectra of (a) significant wave height (Hm0) and (b) the relative differences (of Hm0) between the models.

be following closely the methods outlined by Mei et al. (2005), Komen et al. (1994), and Janssen (2004). Since work on the subject is broad, this section will focus on the kinematic part of the governing equations only and will not address the source terms. For the sake of brevity, derivation of the main equation will only be outlined and variants of the equation will be derived and discussed in more detail later.

3.2.1 Overview of derivation

In Mei et al. (2005), a wave evolution equation with a slowly-varying amplitude but fast phase speed is derived in Cartesian coordinates using the WKB Method (Bender and Orszag, 1978). Here, the depth, $H = H(\boldsymbol{x}_h, t)$, is allowed to vary slowly in location and time such that a new scale separation parameter can be defined as

$$\mu = \mathrm{O}\left(\frac{\nabla H}{kH}\right) \ll 1. \tag{3.4}$$

Using μ, slow coordinates are introduced (excluding the vertical),

$$\overline{\boldsymbol{x}}_h = \mu \boldsymbol{x}_h, \qquad \bar{t} = \mu t, \tag{3.5}$$

and a new set of governing linear wave equations for non-constant depth are derived for $\overline{\Phi}(\overline{\boldsymbol{x}}_h, z, \bar{t}) = \Phi(\overline{\boldsymbol{x}}_h/\mu, z, \bar{t}/\mu)$[9] and $\overline{H}(\overline{\boldsymbol{x}}_h, \bar{t}) = H(\overline{\boldsymbol{x}}_h/\mu, t/\mu)$. Dropping overbar notation, Eqs. (1.31) through (1.33) become

$$\mu^2 \nabla_h^2 \Phi + \partial_{zz}\Phi = 0, \qquad z \in (-H, 0) \tag{3.6}$$

$$\mu^2\, \partial_{tt}\Phi + \mathrm{g}\, \partial_z \Phi = 0, \qquad z = 0, \tag{3.7}$$

$$\partial_z \Phi + \mu^2 \nabla_h H \cdot \nabla_h \Phi = 0, \qquad z = -H. \tag{3.8}$$

A key component of this approach is the assumption that only the phase varies with the fastest scale $(\boldsymbol{x}_h/\mu, t/\mu)$. As such, the following expansion is introduced:

$$\Phi = \left[\phi_0 + (-i\mu)\phi_1 + (-i\mu)^2\phi_2^2 + \ldots\right] e^{is_{m,n}/\mu}, \tag{3.9}$$

[9] The velocity potential is capitalized in this chapter for asymptotic expansion.

where $s_{m,n} = s_{m,n}(\boldsymbol{x}_h, t)$ and $\phi_j = \phi_j(\boldsymbol{x}_h, z, t)$ are the eikonal and amplitude components respectively.[10] Here, the subscripts m, n have been used to emphasize that the solution is for a wave component with the following dependent wavevector and frequency:

$$\mathbf{k}_{m,n}(\boldsymbol{x}_h, t) = \nabla_h \, s_{m,n}(\boldsymbol{x}_h, t) \tag{3.10}$$

$$\omega_{m,n}(\boldsymbol{x}_h, t) = -\, \partial_t \, s_{m,n}(\boldsymbol{x}_h, t). \tag{3.11}$$

In addition, all dependent relations are stated explicitly for clarity. Note that Eqs. (3.10) and (3.11) imply the following consistency relation:

$$\partial_t \mathbf{k}_{m,n} + \nabla_h \omega_{m,n} = 0. \tag{3.12}$$

Differentiating and collecting powers of μ gives an ordered set of meaningful physical equations. The leading order equation ($O(\mu^0)$) for the wave component gives the **variable depth dispersion relation**,

$$\begin{aligned} \omega_{m,n;\pm}(\boldsymbol{x}_h, t) &= \pm \sigma_{m,n} \left(\boldsymbol{x}_h, \mathbf{k}_{m,n}(\boldsymbol{x}_h, t)\right) \\ &= \pm \left\{ \mathrm{g} \left|\mathbf{k}_{m,n}(\boldsymbol{x}_h, t)\right| \tanh[\, \left|\mathbf{k}_{m,n}(\boldsymbol{x}_h, t)\right| H(\boldsymbol{x}_h)\,] \right\}^{1/2}. \end{aligned} \tag{3.13}$$

This is the same relation as defined in Eq. (1.38) but with variable depth. Collecting the next set of terms ($O(-i\mu)$) gives the energy balance equation

$$\frac{\partial}{\partial t}\left(\frac{E_{m,n}}{\sigma_{m,n}}\right) + \nabla_h \cdot \left(\boldsymbol{c}_{g;m,n} \frac{E_{m,n}}{\sigma_{m,n}}\right) = 0, \tag{3.14}$$

where the **total energy** E and **group velocity** $\boldsymbol{c}_g$ are defined here as

$$E_{m,n}(\boldsymbol{x}_h, t) = \rho \mathrm{g} \left| A_{m,n}(\boldsymbol{x}_h, t)\right|^2, \tag{3.15}$$

and

$$\boldsymbol{c}_{g;m,n}(\boldsymbol{x}_h, \mathbf{k}_{m,n}(\boldsymbol{x}_h, t)) = \left[\nabla_{\mathbf{k}} \, \sigma(\boldsymbol{x}_h, \mathbf{k}(\boldsymbol{x}_h, t))\right]_{\mathbf{k}=\mathbf{k}_{m,n}}. \tag{3.16}$$

[10] Notice that z has not been rescaled.

Here, E/σ in Eq. (3.14) is the wave action since it is a conserved quantity when transported by group velocity (i.e., invariant to slow depth variations) (Bretherton and Garrett, 1968; Mei et al., 2005).

The addition of slowly varying currents leads to a similar approximation. Let $\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{x}_h, t)$ represent the depth-independent current. Then as before, the dispersion relation,

$$\begin{aligned}\omega_{m,n;\pm}(\boldsymbol{x}_h, t) &= \Omega_{m,n;\pm}(\boldsymbol{x}_h, \mathbf{k}_{m,n}(\boldsymbol{x}_h, t)) \\ &= \boldsymbol{U}(\boldsymbol{x}_h, t) \cdot \mathbf{k}_{m,n}(\boldsymbol{x}_h, t) \pm \sigma_{m,n}(\boldsymbol{x}_h, \mathbf{k}_{m,n}(\boldsymbol{x}_h, t))\,, \end{aligned} \tag{3.17}$$

and the energy balance equation,

$$\frac{\partial}{\partial t}\left(\frac{E_{m,n}}{\sigma_{m,n}}\right) + \nabla_h \cdot \left(\boldsymbol{v}_{g;m,n}\frac{E_{m,n}}{\sigma_{m,n}}\right) = 0, \tag{3.18}$$

naturally follow (Komen et al., 1994). Here, Ω and $\boldsymbol{v}_g$ are referred to here as the **Doppler-shifted dispersion relation** and **Doppler-shifted group velocity** respectively, where the latter wave component form is defined as,

$$\boldsymbol{v}_{g;m,n}(\boldsymbol{x}_h, \mathbf{k}_{m,n}(\boldsymbol{x}_h, t)) = \left[\nabla_{\mathbf{k}}\, \Omega(\boldsymbol{x}_h, \mathbf{k}(\boldsymbol{x}_h, t))\right]_{\mathbf{k}=\mathbf{k}_{m,n}}. \tag{3.19}$$

The wave action balance equation for a continuous wave spectrum in an inhomogeneous media (slow depth variations with slowly interacting currents) is given in Eq. (3.1). Derivation is nontrivial and the reader is directed to and Willebrand (1975), Komen et al. (1994), and Janssen (2004) for an in-depth treatment. Essentially, the wave component density is a function of time and this leads to the extra term with $\nabla_{\boldsymbol{x}_h}\Omega$ in the balance equation (Willebrand, 1975; Komen et al., 1994). In addition, unlike in previous Eqs. (3.14) and (3.18), $\mathbf{k}$ is no longer a dependent variable[11] and the spectral and spatial domains are coupled.

3.2.2 Generalized equation and simplifications

While Eq. 3.1 is derived for a 2D spatial and 2D spectral domain, the balance equation can be generalized for other coupled domains. Let $\boldsymbol{x}$, $\boldsymbol{k}$, and $\boldsymbol{\alpha} = (\boldsymbol{x}, \boldsymbol{k})$ define generic spatial, spectral, and

[11] The dispersion relations and group velocities derived for the wave components are still valid for the continuous spectrum however.

coupled spatial-spectral domains respectively such that $\boldsymbol{x}, \boldsymbol{k} \in \mathbb{R}^d$ and $\boldsymbol{\alpha} \in \mathbb{R}^{2d}$ for some dimension d.[12] Then the spatial and spectral gradients for some scalar function $F = F(\boldsymbol{\alpha})$ are defined as

$$\nabla_{\boldsymbol{x}} F(\boldsymbol{\alpha}) = \begin{bmatrix} \partial_{x^1} F(\boldsymbol{\alpha}) & \cdots & \partial_{x^d} F(\boldsymbol{\alpha}) \end{bmatrix}^T, \tag{3.20}$$

$$\nabla_{\boldsymbol{k}} F(\boldsymbol{\alpha}) = \begin{bmatrix} \partial_{k^1} F(\boldsymbol{\alpha}) & \cdots & \partial_{k^d} F(\boldsymbol{\alpha}) \end{bmatrix}^T. \tag{3.21}$$

Here the spectral domain represents the local d-dimensional Fourier decomposition of some filtered region about a point in the d-dimensional spatial domain. Let $\mathcal{W} = \mathcal{W}(\boldsymbol{\alpha}, t)$. Then Eq. (3.1) can be rewritten as

$$\partial_t \mathcal{W}(\boldsymbol{\alpha}, t) + \nabla_{\boldsymbol{x}} \cdot [\mathcal{W}(\boldsymbol{\alpha}, t)\, \nabla_{\boldsymbol{k}}\, \Omega(\boldsymbol{\alpha}, t)] - \nabla_{\boldsymbol{k}} \cdot [\mathcal{W}(\boldsymbol{\alpha}, t)\, \nabla_{\boldsymbol{x}} \Omega(\boldsymbol{\alpha}, t)] = \text{Sources}, \tag{3.22}$$

where $\Omega_{\pm}(\boldsymbol{\alpha}, t) = \boldsymbol{U}(\boldsymbol{x}, t) \cdot \boldsymbol{k} \pm \sigma(\boldsymbol{\alpha})$.

Since our interest lies with the kinematic part of the equation, only spatial and spectral evolution are examined here. As such, it is useful to examine problems without source terms or background currents. On an $(2d)$-dimensional slab, Eq. 3.22 simplifies to

$$\partial_t \mathcal{W}(\boldsymbol{\alpha}, t) \pm \nabla_{\boldsymbol{k}}\, \sigma(\boldsymbol{\alpha}) \cdot \nabla_{\boldsymbol{x}} \mathcal{W}(\boldsymbol{\alpha}, t) \mp \nabla_{\boldsymbol{x}}\, \sigma(\boldsymbol{\alpha}) \cdot \nabla_{\boldsymbol{k}} \mathcal{W}(\boldsymbol{\alpha}, t) = 0. \tag{3.23}$$

For problems of constant bathymetry, $\nabla_{\boldsymbol{x}}\, \sigma(\boldsymbol{\alpha}) = 0$ (since $\sigma(\boldsymbol{\alpha}) = \sigma(\boldsymbol{k})$) and Eq. 3.23 simplifies further to

$$\partial_t \mathcal{W} \pm \boldsymbol{c_g}(\boldsymbol{k}) \cdot \nabla_{\boldsymbol{x}} \mathcal{W} = 0, \tag{3.24}$$

where $\boldsymbol{c_g}(\boldsymbol{k}) = c_g(|\boldsymbol{k}|) \frac{\boldsymbol{k}}{|\boldsymbol{k}|} = \nabla_{\boldsymbol{k}} \sigma(\boldsymbol{k})$ is the spatially invariant group velocity.

3.2.3 Derivation in polar coordinates

One of the key goals of building a new spectral wave model is in removing both the numerical and analytical singularities. This will require a thorough understanding of the kinematic part of the governing equation and its projections. For initial insight, a simplified kinematic wave action

[12] If $d = 2$, the earlier wavevector notation $\mathbf{k} = (k_x, k_y)$ is used instead.

balance equation (similar to Eq. 3.24) is derived here in polar coordinates and is analogous to great-circle propagation on a sphere.

Here an abbreviated cylindrical del notion will be used for polar coordinates with the assumption all cylindrical z-components are null. For clarity, these del operations are denoted ∇_p and are defined and derived in Appendix A.6. In this system, scalar and vector functions are represented as $f = f(r, \lambda)$ and $\boldsymbol{A} = A_r e_r + A_\lambda e_\lambda = (A_r, A_\lambda)$ with $r \in \mathbb{R}^+$ and $\lambda \in (-\pi, \pi]$.

3.2.3.1 Physical model and governing equations

For simplification, consider a horizontally periodic 1D problem such that the propagation can be modeled in polar coordinates with a constant interior radius R and unperturbed ideal fluid with thickness H. Define $\mathbf{g} = (-\mathrm{g}, 0)$ and let $\eta = \eta(\lambda, t)$ represent the surface displacement with $\langle \eta - (R + H) \rangle = 0$.[13] Then using Eq. (A.26), the governing linear wave equations (Eqs. 1.31 to 1.33) can be written in polar form as

$$\frac{1}{r}\,\partial_r \Phi + \,\partial_{rr}\Phi + \frac{1}{r^2}\,\partial_{\lambda\lambda}\Phi = 0, \qquad r \in (R, R + H)\,, \tag{3.25}$$

$$\partial_{tt}\Phi + \mathrm{g}\,\partial_r \Phi = 0, \qquad r = R + H, \tag{3.26}$$

$$\partial_r \Phi = 0, \qquad r = R, \tag{3.27}$$

where η and Φ are related through the linearized polar form of Eq. (1.17) as

$$\partial_t \Phi|_{r=R+H} = -g\eta. \tag{3.28}$$

3.2.3.2 Expansion basis for solution

Here, the same multiple-scales analysis as outlined in Section 3.2.1 is used to derive a wave action balance equation. As such, it is necessary first to determine a basis for the asymptotic expansion. Assume Φ is separable such that $\Phi(r, \lambda, t) = P(r)\Lambda(\lambda)T(t)$. Then Eq. (3.25) gives

$$\nabla_p^2 \Phi = \frac{1}{r} P' \Lambda T + P'' \Lambda T + \frac{1}{r^2} P \Lambda'' T = 0.$$

[13] Notice that for non-constant R, H would need to vary in conjunction.

Since the solution is separable and Λ is periodic,

$$r\frac{P'}{P}+r^2\frac{P''}{P}=m^2=-\frac{\Lambda''}{\Lambda},$$

and $\Lambda \propto e^{im\lambda}$ for $m \in \mathbb{Z}$. In addition, if $P \propto r^q$, then the following relation must hold

$$q^2-|m|^2=0$$

and $P \propto r^m$.

3.2.3.3 Multiple-scales analysis

For $\delta \ll 1$ and $j \in \mathbb{N}_+$, let the slower variables be formally defined as

$$\lambda_j=\delta^j\lambda_f, \quad t_j=\delta^j t_f. \tag{3.29}$$

Here, the subscript f has been added to denote the original fast scale. This implies derivatives with respect to λ and t will be replaced with

$$\begin{aligned}\partial_\lambda &= \partial_{\lambda_f}+\delta\,\partial_{\lambda_1}+\delta^2\,\partial_{\lambda_2}+\dots,\\ \partial_{\lambda\lambda} &= \partial_\lambda\left(\partial_\lambda\right)\\ &= \partial_{\lambda_f\lambda_f}+\delta\left(2\,\partial_{\lambda_f\lambda_1}\right)+\delta^2\left(2\,\partial_{\lambda_f\lambda_2}+\,\partial_{\lambda_1\lambda_1}\right)+\dots\,.\end{aligned}$$

Let $\sigma=\sigma(m)$ be some yet determined dispersion relation. Then for

$$\Phi=\Phi\left(r,\lambda_f,\lambda_1,\lambda_2,\dots,t_f,t_1,t_2,\dots\right), \tag{3.30}$$

$$\psi_\alpha=\psi_\alpha\left(r,\lambda_1,\lambda_2,\dots,t_1,t_2,\dots\right), \qquad \alpha=0,1,2,\dots, \tag{3.31}$$

assume the following solution ansatz:

$$\Phi=\left(\psi_0+\delta\psi_1+\delta^2\psi_2+\dots\right)e^{i\left[m\lambda_f-\omega t_f\right]}. \tag{3.32}$$

Turning the crank, we find

$$\begin{aligned}\partial_r\Phi &= \left(\,\partial_r\psi_0+\delta\,\partial_r\psi_1+\delta^2\,\partial_r\psi_2+\dots\right)e^{i\left[m\lambda_f-\omega t_f\right]},\\ \partial_{rr}\Phi &= \left(\,\partial_{rr}\psi_0+\delta\,\partial_{rr}\psi_1+\delta^2\,\partial_{rr}\psi_2+\dots\right)e^{i\left[m\lambda_f-\omega t_f\right]},\end{aligned}$$

$$\begin{aligned}\partial_{\lambda\lambda}\Phi =& \ \partial_{\lambda_f\lambda_f}\Phi + \delta\left(2\,\partial_{\lambda_f\lambda_1}\Phi\right) + \delta^2\left(2\,\partial_{\lambda_f\lambda_2}\Phi + \partial_{\lambda_1\lambda_1}\Phi\right) + \dots \\ =& \left[-m^2\psi_0 + \delta\left(-m^2\psi_1 + i\,2m\,\partial_{\lambda_1}\psi_0\right)\right. \\ &\left. + \,\delta^2\left(-m^2\psi_2 + i\,2m\,\partial_{\lambda_1}\psi_1 + i\,2m\,\partial_{\lambda_2}\psi_0 + \partial_{\lambda_1\lambda_1}\psi_0\right) + \dots\right] e^{i\left[m\lambda_f - \omega t_f\right]},\end{aligned}$$

and

$$\begin{aligned}\partial_{tt}\Phi =& \ \partial_{t_f t_f}\Phi + \delta\left(2\,\partial_{t_f t_1}\Phi\right) + \delta^2\left(2\,\partial_{t_f t_2}\Phi + \partial_{t_1 t_1}\Phi\right) + \dots \\ =& \left[-\omega^2\psi_0 + \delta\left(-\omega^2\psi_1 - i\,2\omega\,\partial_{t_1}\psi_0\right)\right. \\ &\left. + \,\delta^2\left(-\omega^2\psi_2 - i\,2\omega\,\partial_{t_1}\psi_1 - i\,2\omega\,\partial_{t_2}\psi_0 + \partial_{t_1 t_1}\psi_0\right) + \dots\right] e^{i\left[m\lambda_f - \omega t_f\right]}.\end{aligned}$$

Collecting terms with δ^0 gives the first set of ordered equations:

$$\frac{1}{r}\,\partial_r\psi_0 + \partial_{rr}\psi_0 - \frac{m^2}{r^2}\psi_0 = 0, \qquad r \in (R, R+H) \tag{3.33}$$

$$g\,\partial_r\psi_0 - \omega^2\psi_0 = 0, \qquad r = R+H \tag{3.34}$$

$$\partial_r\psi_0 = 0, \qquad r = R. \tag{3.35}$$

Assume ψ_0 is separable in r; then for $B^{\pm} = B^{\pm}(\lambda_1, \lambda_2, \dots, t_1, t_2, \dots)$, the lowest ordered solution becomes

$$\psi_0 = B^+ r^m + B^- r^{-m}. \tag{3.36}$$

Employing Eq. (3.35), we find

$$\partial_r\psi_0\big|_{r=R} = m\left(B^+ R^{m-1} - B^- R^{-m-1}\right) = 0, \tag{3.37}$$

and implies

$$\psi_0 = B^+\left(r^m + R^{2m} r^{-m}\right).$$

Using Eq. (3.28), B^+ can be redefined such that

$$\psi_0 = -\frac{i\,\mathrm{g}B}{\omega}\left[\frac{r^m + R^{2m} r^{-m}}{(R+H)^m + R^{2m}\,(R+H)^{-m}}\right], \tag{3.38}$$

and

$$\partial_r\psi_0\Big|_{r=R+H} = -\frac{i\,\mathrm{g}mB}{\omega(R+H)}\left[\frac{(R+H)^{2m}-R^{2m}}{(R+H)^{2m}+R^{2m}}\right]. \tag{3.39}$$

After employing Eq. (3.34), a polar linear dispersion relation is derived for $\omega = \omega_{\pm} = \pm\sigma$:

$$\sigma(m) = \left\{\frac{\mathrm{g}m}{R+H}\left[\frac{(R+H)^{2m}-R^{2m}}{(R+H)^{2m}+R^{2m}}\right]\right\}^{1/2} = \left\{\frac{\mathrm{g}\,|m|}{R+H}\left[\frac{\left(1+\frac{H}{R}\right)^{2|m|}-1}{\left(1+\frac{H}{R}\right)^{2|m|}+1}\right]\right\}^{1/2}. \tag{3.40}$$

Repeating and collecting terms with δ^1, gives the second set of ordered equations:

$$\frac{1}{r}\,\partial_r\psi_1 + \partial_{rr}\psi_1 + \frac{1}{r^2}\left(-m^2\psi_1 + i\,2m\,\partial_{\lambda_1}\psi_0\right) = 0, \qquad r\in(R,R+H), \tag{3.41}$$

$$g\,\partial_r\psi_1 + \left(-\omega^2\psi_1 - i\,2\omega\,\partial_{t_1}\psi_0\right) = 0, \qquad r = R+H, \tag{3.42}$$

$$\partial_r\psi_1 = 0, \qquad r = R. \tag{3.43}$$

Notice now that Eqs. (3.41) and (3.42) are inhomogeneous. To solve, first note the following property:

$$\begin{aligned}\frac{1}{r}\,\partial_r\left[r\left(\psi_0\,\partial_r\psi_1 - \psi_1\,\partial_r\psi_0\right)\right] &= \psi_0\left(\frac{1}{r}\,\partial_r\psi_1 + \partial_{rr}\psi_1 - \frac{m^2}{r^2}\psi_1\right) - \psi_1\left(\frac{1}{r}\,\partial_r\psi_0 + \partial_{rr}\psi_0 - \frac{m^2}{r^2}\psi_0\right)\\ &= \psi_0\left(-\frac{i\,2m}{r^2}\,\partial_{\lambda_1}\psi_0\right).\end{aligned} \tag{3.44}$$

Integrating Eq. (3.44) with respect to r, yields

$$\Big[r\left(\psi_0\,\partial_r\psi_1 - \psi_1\,\partial_r\psi_0\right)\Big]_{r=R}^{r=R+H} = -i\,2m\int_R^{R+H}\frac{1}{r}\psi_0\,\partial_{\lambda_1}\psi_0 dr. \tag{3.45}$$

Applying Eqs. (3.42) and (3.43) on the LHS of Eq. (3.45), gives

$$\begin{aligned}\mathrm{LHS} &= (R+H)\left\{-\frac{i\mathrm{g}B}{\omega}\,\partial_r\psi_1 + \frac{i\,\mathrm{g}mB}{\omega(R+H)}\left[\frac{(R+H)^{2m}-R^{2m}}{(R+H)^{2m}+R^{2m}}\right]\psi_1\right\}_{r=R+H}\\ &= (R+H)\left[-\frac{i\,B}{\omega}\left(i\,2\omega\,\partial_{t_1}\psi_0\right)\right]_{r=R+H}\\ &= \left[\partial_{t_1}\!\left(\frac{\mathrm{g}B^2}{\omega}\right)\right]\left[-i\,(R+H)\right].\end{aligned} \tag{3.46}$$

Applying likewise on the RHS, gives

$$\begin{aligned}
\text{RHS} &= (\,\partial_{\lambda_1} B)\left(-\frac{i\,2m}{B}\right)\int_R^{R+H}\frac{1}{r}\psi_0^2 dr \\
&= (\,\partial_{\lambda_1} B)\left(\frac{i\,2m}{B}\right)\int_R^{R+H}\frac{\mathrm{g}^2B^2}{r\omega^2}\left[\frac{r^m + R^{2m}r^{-m}}{(R+H)^m + R^{2m}\,(R+H)^{-m}}\right]^2 dr \\
&= (\,\partial_{\lambda_1} B)\left\{\frac{i\,2m\mathrm{g}^2B(R+H)^{2m}}{\omega^2\left[(R+H)^{2m}+R^{2m}\right]^2}\right\}\int_R^{R+H}\frac{1}{r}\left(r^{2m}+2R^{2m}+R^{4m}r^{-2m}\right)dr \\
&= \left[\partial_{\lambda_1}\left(\frac{\mathrm{g}B^2}{\omega}\right)\right]\left[\frac{i\,\omega(R+H)^{2m+1}}{(R+H)^{4m}-R^{4m}}\right]\int_R^{R+H}\left(r^{2m-1}+2R^{2m}r^{-1}+R^{4m}r^{-2m-1}\right)dr. \qquad (3.47)
\end{aligned}$$

Combining both sides (and multiplying by a constant ρ) yields

$$\begin{aligned}
\left[\partial_{t_1}\left(\frac{\rho\mathrm{g}B^2}{\omega}\right)\right] = \left[\partial_{\lambda_1}\left(\frac{\rho\mathrm{g}B^2}{\omega}\right)\right]&\left[\frac{-\omega(R+H)^{2m}}{(R+H)^{4m}-R^{4m}}\right] \\
&\times\int_R^{R+H}\left(r^{2m-1}+2R^{2m}r^{-1}+R^{4m}r^{-2m-1}\right)dr. \qquad (3.48)
\end{aligned}$$

Recall from earlier that the wave action can be defined as $\mathcal{W}_p = \rho\mathrm{g}B^2/\omega$ (see Eq. 3.15).[14] Assuming $m \neq 0$, then Eq. (3.48) yields the wave action balance equation

$$\begin{aligned}
\partial_{t_1}\mathcal{W}_p &= -\,(\,\partial_{\lambda_1}\mathcal{W}_p)\left[\frac{\omega(R+H)^{2m}}{(R+H)^{4m}-R^{4m}}\right]\left[\frac{1}{2m}r^{2m}-\frac{R^{4m}}{2m}r^{-2m}+2R^{2m}\ln r\right]_R^{R+H} \\
&= -\,(\,\partial_{\lambda_1}\mathcal{W}_p)\;\omega\left\{\frac{1}{2m}+\operatorname{sgn}(m)\left[\frac{2\left(1+\frac{H}{R}\right)^{2|m|}}{\left(1+\frac{H}{R}\right)^{4|m|}-1}\right]\ln\left(1+\frac{H}{R}\right)\right\}. \qquad (3.49)
\end{aligned}$$

3.2.3.4 Asymptotic simplification

For the solution to be useful, it is necessary to simplify about an appropriate scale. First, notice that Eq. (3.49) is only dependent on (λ_1, t_1) and the wave action can be truncated as $\mathcal{W}_p(\lambda_1, \lambda_2, \ldots, t_1, t_2, \ldots) \approx \mathcal{W}_p(\lambda_1, t_1)$. Second, recall the following binomial series expansion:

$$\left(1+\frac{H}{R}\right)^{2|m|} = \sum_{n=0}^{\infty}\begin{pmatrix}2\,|m|\\ \\ n\end{pmatrix}\left(\frac{H}{R}\right)^n = \sum_{n=0}^{2|m|}\frac{(2\,|m|)!}{n!(2\,|m|-n)!}\left(\frac{H}{R}\right)^n.$$

Expanding the squared dispersion relation as a binomial series, gives

$$\sigma^2 = \frac{\mathrm{g}\,|m|}{R+H}\left[\frac{\left(1+\frac{H}{R}\right)^{2|m|}-1}{\left(1+\frac{H}{R}\right)^{2|m|}+1}\right] = \frac{\mathrm{g}\,|m|}{R+H}\left[\frac{2\,|m|\left(\frac{H}{R}\right)+\ldots}{2+2\,|m|\left(\frac{H}{R}\right)+\ldots}\right]. \qquad (3.50)$$

[14] The subscript p is used to denote polar coordinates.

Notice that each binomial term can be bounded as

$$\begin{pmatrix} 2|m| \\ \\ n \end{pmatrix} \leq \frac{(2|m|)^n}{n!}.$$

Let $\varepsilon = 2|m|H/R$ for $m \neq 0$. Then it follows that the binomial series can be bounded as

$$1 + \varepsilon \leq \left(1 + \frac{H}{R}\right)^{2|m|} \leq 1 + \varepsilon + \frac{1}{2}\varepsilon^2 + \cdots + \frac{1}{(2\,|m|)!}\varepsilon^{2|m|}.$$

There are two natural asymptotic limits to this bound. For $\varepsilon \ll 1$, both Eqs. (3.50) and (3.49) respectively simplify to

$$\sigma^2 \approx \frac{\mathrm{g}|m|}{R+H}\left(\frac{\varepsilon}{2+\varepsilon}\right) \approx \frac{\mathrm{g}|m|\varepsilon}{2(R+H)} = \frac{\mathrm{g}m^2 H}{R(R+H)}, \tag{3.51}$$

and

$$\begin{aligned} \partial_{t_1}\mathcal{W}_p &\approx -\left(\partial_{\lambda_1}\mathcal{W}_p\right)\omega\left[\frac{1}{2m} + \operatorname{sgn}(m)\frac{H}{\varepsilon R}\right] = -\left(\partial_{\lambda_1}\mathcal{W}_p\right)\frac{\omega}{m} \\ &\approx \mp\left(\partial_{\lambda_1}\mathcal{W}_p\right)\sqrt{\frac{\mathrm{g}H}{R(R+H)}}. \end{aligned} \tag{3.52}$$

However, if $\varepsilon \gg 1$, then

$$\left(1 + \frac{H}{R}\right)^{2|m|} - 1 \geq \varepsilon \gg 1,$$

and

$$\left[\frac{\left(1+\frac{H}{R}\right)^{2|m|} - 1}{\left(1+\frac{H}{R}\right)^{2|m|} + 1}\right] \approx 1.$$

This implies for $2\,|m| \gg R/H$, both Eqs. (3.50) and (3.49) respectively simplify to the new limits

$$\sigma^2 \approx \frac{\mathrm{g}\,|m|}{R+H}, \tag{3.53}$$

and

$$\begin{aligned} \partial_{t_1}\mathcal{W}_p &\approx -\left(\partial_{\lambda_1}\mathcal{W}_p\right)\omega\left[\frac{1}{2m} + \operatorname{sgn}(m)\frac{2H/R}{\left(1+\frac{H}{R}\right)^{2|m|}}\right] \approx -\left(\partial_{\lambda_1}\mathcal{W}_p\right)\frac{\omega}{2m} \\ &\approx \mp\left(\partial_{\lambda_1}\mathcal{W}_p\right)\frac{1}{2}\sqrt{\frac{\mathrm{g}}{|m|\,(R+H)}}. \end{aligned} \tag{3.54}$$

3.2.3.5 Cartesian solution form

The units for the dispersion relations should be in m/s and rad/s in Cartesian and polar coordinates respectively. In addition, converting from polar to Cartesian coordinates yields the eikonal relation $m = (R + H)k$ and the partial derivative $\partial_\lambda = -y\,\partial_x + x\,\partial_y$ for $x^2 + y^2 = 1$. Since $H/R \sim 5 \times 10^{-4}$ on average in the ocean, Eqs. (3.52) and (3.54) simplify to the following wave action balance equations:

$$\partial_t \mathcal{W} \pm \frac{\sqrt{\mathrm{g}H}}{R} \begin{bmatrix} -y & x \end{bmatrix} \begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} = 0 \tag{3.55}$$

$$\partial_t \mathcal{W} \pm \frac{1}{2R}\sqrt{\frac{\mathrm{g}}{k}} \begin{bmatrix} -y & x \end{bmatrix} \begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} = 0. \tag{3.56}$$

Notice that these equations are similar to Eq. (3.24). Since the bathymetry did not vary (i.e., H and R were held constant), no spectral gradient is expected. In addition, a closer inspection reveals that $\sqrt{\mathrm{g}H}/R$ and $\sqrt{\mathrm{g}}/2R\sqrt{k}$ are in fact the group velocities for linear shallow and deep-water (respectively) on a unit circle! A later comparison in Section 3.3.2 shows that Eqs. (3.55) and (3.56) are indeed equivalent to Eq. (3.24) projected onto a unit circle.

3.3 The wave action balance equation in different geometries

The meshless spectral wave prototype in Chapter 4 uses a different numerical method than WAVEWATCH III (as well as other third-generation wave models such as WAM and SWAN) and as a result requires a different formulation of the tangent-plane approximation of Eq. (3.22) that is used (see Section 3.3.4). In addition, it is beneficial to test the new prototype on simpler wave action balance equations before proceeding to the full problem. As a consequence, it is necessary to formulate wave action balance equations for several different geometries. For simpler geometries, it is unnecessary to derive each balance equation using the WKB method as in Section 3.2.3, and projections of the general wave action balance equation, Eq. (3.22), and its variants can be used. Care is required though since projections into a particular coupled spatial-spectral domain are not guaranteed to exist. For clarity, each balance equation explored in the prototype is derived or stated here separately.

3.3.1 Test geometries

Different projections of the wave action balance equation onto the following geometries are considered here: **ring-point**, **ring-line**, **sphere-ring**, and **sphere-cylinder**. The named components denote the geometry of the spatial and spectral domains respectively. The first geometry, ring-point, is essentially a 1D periodic advection problem on a 2D spatial ring with a fixed frequency. It is primarily considered to aid in later numerical stencil selection for the other geometries. In the ring-line geometry, a continuous spectrum of waves are considered, moving in a clockwise or counter-clockwise fashion on a 2D spatial ring. In this cylindrical geometry, the spatial and spectral domains can either be coupled or uncoupled depending on whether a non-constant bathymetry or frequency dependent source term is used.

In the last two geometries, both domains are always coupled for any projection of Eq. (3.22). In the sphere-ring geometry, only monochromatic waves are considered to propagate on a sphere. However, unlike for the ring-point case, the domains are coupled since the spectral directional component is locally dependent on its spatial position. In the last geometry, sphere-cylinder, the spectral domain can be expanded from a 2D directional ring to include a continuous spectrum of frequencies. Again, coupling of the frequency component with the spatial domain will only occur when a non-constant bathymetry or frequency dependent source term is used.

3.3.2 Spatial 1D periodic with spectral scalar formulation (ring-point)

In Section 3.2.3, a wave action balance equation (in polar coordinates with constant depth) and its asymptotic form were derived for propagation along a great-circle and unit circle respectively. The latter asymptotic balance equation can also be derived directly from Eq. (3.24). Consider a monochromatic wave propagating on a plane with constant direction. Without a loss of generality, let $\mathbf{k}_c = (k_c, 0)$ and $\boldsymbol{c_g}(\mathbf{k}_c) = (c_g(k_c), 0)$. Then the coupled domain collapses into a spatial one only and Eq. (3.24) becomes

$$\partial_t \mathcal{W} = \mp c_g(k_c)\, \partial_x \mathcal{W}. \tag{3.57}$$

Using Eq. (A.31) for some finite length $2\pi R > 0$, the equation can mapped into polar coordinates with $\mathcal{W}(x,t) = \mathcal{W}(x + 2\pi n R, t)$ for $n \in \mathbb{Z}$ as

$$\partial_t \mathcal{W}_p = \mp \frac{c_g(k_c)}{R}\, \partial_\lambda \mathcal{W}_p, \tag{3.58}$$

where $\lambda \in (-\pi, \pi]$. Let $\overline{x} = x/R$. Then a normalized Cartesian form of the equation in ring-point geometry can be formulated as

$$\partial_t \mathcal{W} \pm \frac{c_g(k_c)}{R}\, (-\overline{y}\, \partial_{\overline{x}} \mathcal{W} + \overline{x}\, \partial_{\overline{y}} \mathcal{W}) = 0. \tag{3.59}$$

Comparing Eqs. (3.56) and (3.59), we find the equations match if the deep-water group velocity on a line, $c_g(k) = 1/2\sqrt{\mathrm{g}/k}$, is used.

3.3.3 Spatial 1D periodic with spectral 1D formulation (ring-line)

Expanding upon the previous geometry, consider not one but a continuous spectrum of waves propagating on a plane with the same constant direction. To be general, it is necessary to consider Eq. (3.22) or Eq. (3.23) since $\mathbf{k}$ is not fixed and the wave action can evolve if the bathymetry varies. For simplicity however, initially consider a problem with constant bathymetry. Again without a loss of generality, let $\mathbf{k} = (k, 0)$ and $\boldsymbol{c_g}(\mathbf{k}) = (c_g(k), 0)$. Following Section 3.3.2, the spatially normalized form (dropping overline notation) becomes

$$\partial_t \mathcal{W} \pm \frac{c_g(k)}{R}\, (-y\, \partial_x \mathcal{W} + x\, \partial_y \mathcal{W}) = 0. \tag{3.60}$$

Here we can interpret the new coupled domain $\boldsymbol{\alpha} = (\boldsymbol{x}, k) = (x, y, k)$ as a cylinder oriented in k with radius R. Intuitively, waves are propagating in a clockwise or counter-clockwise fashion on a 2D spatial ring with some variable wavenumber. Since the bathymetry is constant, the group velocity is independent of spatial location and individual waves are decoupled. If there are slow depth variations however, the spatial and spectral domains are coupled and it is necessary to use

Eq. (3.13) (the variable depth dispersion relation) with Eq. (3.23) to project a new balance equation onto the cylinder.

3.3.4 Spatial spherical surface with spectral 1D periodic formulation (sphere-ring)

Third-generation models solve a projected wave action balance equation that is global spatially and local spectrally (Tolman, 2009). The spectral domain is a tangent-plane approximation that is orientated to and singular at the North and South Poles (Groves and Melcer, 1961; Groves, 1966). The projected wave action balance equation from WAMDI Group (1988) and Tolman (2009) can be simplified for a fixed frequency f as

$$\partial_t \mathcal{W} \pm \frac{c_g}{R_{\boldsymbol{x}} \cos\mu} \left[\sin\theta' \, \partial_\lambda \mathcal{W} + \cos\theta' \, \partial_\mu \left(\mathcal{W} \cos\mu\right) + \sin\mu \, \partial_{\theta'} \left(\mathcal{W} \sin\theta'\right)\right] = \text{Sources.} \tag{3.61}$$

where θ' is measured clockwise with respect to true north and c_g is the depth-dependent group velocity. To convert to radians, let $\theta' = -\theta + \frac{\pi}{2}$. Then $\sin\theta' = \cos\theta$, $\cos\theta' = \sin\theta$, and $\partial_{\theta'} = -\partial_\theta$. Rewriting and expanding, gives

$$\begin{aligned} \partial_t \mathcal{W} \pm \frac{c_g}{R_{\boldsymbol{x}} \cos\mu} \left[\cos\theta \, \partial_\lambda \mathcal{W} + \sin\theta \, \partial_\mu \left(\mathcal{W} \cos\mu\right) - \sin\mu \, \partial_\theta \left(\mathcal{W} \cos\theta\right)\right] &= \\ \partial_t \mathcal{W} \pm \frac{c_g}{R_{\boldsymbol{x}}} \left[\frac{\cos\theta}{\cos\mu} \, \partial_\lambda + \sin\theta \, \partial_\mu - \tan\mu \cos\theta \, \partial_\theta\right] \mathcal{W} &= \text{Sources.} \end{aligned} \tag{3.62}$$

To use with the meshless model, it is necessary to convert Eq. (3.62) into a form suitable for the Euclidian norm. Here, the spherical and directional-frequency coordinates, $(\lambda, \mu, r, f, \theta)$, are converted to Cartesian and wavenumber ones, (x, y, z, k, l).[15] Let $\|\boldsymbol{x}\| = R_{\boldsymbol{x}}$ and $\|\boldsymbol{k}\| = R_{\boldsymbol{k}}$ since propagation is confined to the sphere and frequency is fixed. In addition, note the following definitions and transformations:

$$x = R_{\boldsymbol{x}} \cos\lambda \cos\mu, \quad y = R_{\boldsymbol{x}} \sin\lambda \cos\mu, \quad z = R_{\boldsymbol{x}} \sin\mu,$$
$$k = R_{\boldsymbol{k}} \cos\theta, \quad l = R_{\boldsymbol{k}} \sin\theta.$$

This implies

$$\cos\mu = \frac{\sqrt{x^2 + y^2}}{R_{\boldsymbol{x}}}, \qquad \tan\mu = \frac{z}{\sqrt{x^2 + y^2}},$$

[15] The notation $\boldsymbol{k} = (k, l)$ is used for the prototype model instead.

and

$$\begin{aligned}
\partial_\lambda &= \frac{\partial x}{\partial \lambda}\,\partial_x + \frac{\partial y}{\partial \lambda}\,\partial_y + \frac{\partial z}{\partial \lambda}\,\partial_z = -y\,\partial_x + x\,\partial_y, \\
\partial_\mu &= \frac{\partial x}{\partial \mu}\,\partial_x + \frac{\partial y}{\partial \mu}\,\partial_y + \frac{\partial z}{\partial \mu}\,\partial_z = \frac{z}{\sqrt{x^2+y^2}}\left[-x\,\partial_x - y\,\partial_y\right] + \sqrt{x^2+y^2}\,\partial_z, \\
\partial_\theta &= \frac{\partial k}{\partial \theta}\,\partial_k + \frac{\partial l}{\partial \theta}\,\partial_l = -l\,\partial_k + k\,\partial_l.
\end{aligned}$$

Plugging in, a suitable wave action balance equation (on a sphere for fixed frequency) for the new prototype is obtained:

$$\begin{aligned}
\partial_t \mathcal{W} \pm \frac{c_g}{R_{\boldsymbol{x}}} & \left\{ \frac{k}{R_{\boldsymbol{k}}} \frac{R_{\boldsymbol{x}}}{\sqrt{x^2+y^2}} \left[-y\,\partial_x + x\,\partial_y\right] \right. \\
& \left. + \frac{l}{R_{\boldsymbol{k}}} \left[\frac{z}{\sqrt{x^2+y^2}} \left(-x\,\partial_x - y\,\partial_y\right) + \sqrt{x^2+y^2}\,\partial_z \right] - \frac{k}{R_{\boldsymbol{k}}} \frac{z}{\sqrt{x^2+y^2}} \left[-l\,\partial_k + k\,\partial_l\right] \right\} \mathcal{W} = \\
\partial_t \mathcal{W} \pm \frac{c_g}{R_{\boldsymbol{x}} R_{\boldsymbol{k}} \sqrt{x^2+y^2}} & \left\{ \left(-R_{\boldsymbol{x}} yk - xzl\right) \partial_x + \left(R_{\boldsymbol{x}} xk - yzl\right) \partial_y \right. \\
& \left. + l\left(x^2+y^2\right) \partial_z + \left(zkl\right) \partial_k + \left(-zk^2\right) \partial_l \right\} \mathcal{W} = \text{Sources.} \qquad (3.63)
\end{aligned}$$

As a simple check to verify that Eq. (3.63) can reproduce earlier great-circle propagation, let $z = 0$. Then $R_{\boldsymbol{k}} = k$ and

$$\partial_t \mathcal{W} \pm \frac{c_g}{\sqrt{x^2+y^2}} \left\{-y\,\partial_x + x\,\partial_y\right\} \mathcal{W} = \text{Sources.} \qquad (3.64)$$

This is equivalent to Eqs. (3.56) and (3.59) for the deep-water group velocity. This can be repeated for northern and southern propagation only (i.e., $y = k = 0$) with similar results.

3.3.5 Spatial spherical surface with spectral cylindrical surface formulation (sphere-cylinder)

In this dissertation, only aquaplanets with constant bathymetry are studied. As such, consideration of the spectral gradient of the wave action is unnecessary and Eq. (3.63) can be used for continuous frequencies.

3.4 Nonsingular wave action balance equation on a sphere

Deriving a nonsingular form of the wave action balance equation on a sphere is a nontrivial task and there are several main hurdles. The first is the sheer complexity of the problem. Consider a simplified version of the WKB method employed earlier for the Cartesian and polar solutions. For some $f = f(\mu)$, the method essentially uses a scale substitution $\mu = \mu(\mu_1, \mu_2; \delta)$ such that f can be rewritten as

$$f(\mu) = f(\mu(\mu_1, \mu_2; \delta)) = [\delta^{\alpha_0} g_0(\mu_2) + \delta^{\alpha_1} g_1(\mu_2) + \delta^{\alpha_1} g_2(\mu_2) + \ldots]\, h(\mu_1),$$

where h is entirely described by the fast variable μ_1. In Cartesian and polar coordinates, exponentials are used as a solution basis to Laplace's equation and the fast derivatives of f are easily calculable. However, in spherical coordinates this no longer the case since the solution basis includes associated Legendre polynomials, which have mode dependent derivatives that would most likely need to be calculated numerically for higher mode solutions.

There are spherical harmonic solutions to the Boltzmann transport equation, so it is possible a solution also exists for the kinematic part of the wave action balance equation. However, and this is where the second hurdle lies, any asymptotic solution would need to be repeated for each and every source term! Recent work by Hong et al. (2011) suggest that it is possible to project the Cartesian form of Eq. 3.22, the general wave action balance equation (with source terms), onto a sphere and this is worth exploring.

The final hurdle, and perhaps the most serious, is that any projected equation would then be global – both spatially and spectrally. It is not clear yet if a solution of this type would be practical or useful, particularly since the solution would probably require a high number of spherical harmonic modes to resolve the surface gravity waves. In addition, the use of some sort of intermediate spectra, such as the Wigner-Ville spectrum (Martin and Flandrin, 1985), would probably be necessary. As a consequence, wave modelers are stuck with the singular wave action balance equation until these issues can be resolved or a new approach formulated.

Chapter 4

Meshless spectral wave modeling using RBF-generated finite differences

4.1 Introduction

During analysis of SD, WAVEWATCH III was used extensively to generate 1D and 2D spectral data and it quickly became clear that running a spectral wave model within a coupled climate model would be computationally expensive. Several alternatives were considered, including a simplified surface wave model that parametrized the frequency component of the 2D directional-frequency spectrum (Mellor et al., 2008). However, the model was new and untested and did not include wave-wave interactions. Instead, a numerical approach was taken to improve accuracy and speed. Since WAVEWATCH III uses a third-order finite difference scheme, a numerical method better suited for spherical geometry was selected and work on the new prototype began.

For advective problems on a sphere, traditional finite difference methods (abbreviated FD hereafter) typically compare poorly in terms of accuracy with other mesh and meshless methods (e.g., pseudospectral, radial basis functions, finite elements). The traditional FD methods have higher errors and require much finer grids to achieve comparable accuracy with other gridded methods such as pseudospectral-Fourier and spherical harmonic (Fornberg and Merrill, 1997). In addition, application of the FD method in higher dimensions requires a smart stencil selection to minimize error (Iserles, 2009; Fornberg, 1998a,b). For flow on a sphere,[1] the advective direction can be quite important and errors will be large if care is not taken to avoid using nonlocal values to calculate derivatives in off-axes directions. This is not a problem however for radial basis functions

[1] The FD method commonly uses the cylindrical Mercator projection to model flow on a sphere.

(abbreviated as RBF hereafter), a meshless method, since they naturally use local information. Errors are largely independent of direction and only depend on the number of nodes (N) and node layout used (Fornberg et al., 2010).

For many reasons, RBFs are an attractive alternative. In addition to being well-suited for advective problems on a sphere, the method is spectrally accurate (Flyer and Wright, 2007) and requires much fewer unknowns (compared with FDs) to solve problems with smooth solutions. However, the method is global and requires inversion of an $N \times N$ matrix to determine interpolation weights. As a result, the global method does not scale well for full 3D and higher dimensional problems (Fornberg and Lehto, 2011), such as the wave action balance equation.

The RBF-generated finite difference method (hereafter RBF-FD) is a novel compromise between the local FD and global RBF methods. Instead of polynomials, RBFs are used to generate FD stencils that are exact for the linear operator being approximated. These stencils work in a meshless layout and reduce the size of the matrix needed for inversion (Fornberg and Lehto, 2011; Flyer et al., 2012). However, the method may introduce spurious eigenvalues and render the problem unstable unless treated. With the introduction of hyperviscosity (Fornberg and Lehto, 2011; Bollig et al., 2012), it is now possible to stabilize the method and solve problems in higher dimensions.

Here, we will explore the performance of the RBF-FD method on the tangent-plane projection of the wave action balance equation (see Section 3.3.4). This requires adapting RBF-FD to work in a coupled spatial-spectral domain. It is hoped that the new method will permit a scaled model with fewer unknowns but similar accuracy. Since this is a higher dimensional problem, this could drastically reduce computational costs and allow the model to run at a higher resolution. To the author's knowledge, this is the first time a meshless method has been used to model a 3D wave action balance equation.

4.2 Overview of the numerical method

Here a brief overview of the global RBF and local RBF-FD methods are given. For a comprehensive review of the methods applied to geophysical problems on a sphere, the reader is referred to Flyer and Wright (2007) and Flyer et al. (2012). In this chapter, the bold vector notation ($\boldsymbol{\alpha}$) is replaced with the arrow vector notation ($\vec{\alpha}$) to distinguish from numerical arrays, which are denoted $\underline{a} = \underline{a_i}$ and $\underline{\underline{a}} = \underline{\underline{a_{ij}}}$ for column vectors and rectangular matrices respectively (see notation in preamble). In addition, all vector norms are Euclidian. Unless otherwise specified, the sequence norm ℓ_2 is used to analyze relative error.

4.2.1 Global RBF methodology

The global RBF method linearly combines translates of a single radial basis function to interpolate the data. Analytical differentiation of the interpolant is then used to find derivatives at the node locations. Let $\mathcal{W} = \mathcal{W}(\vec{\alpha}, t)$ be a continuous scalar function for $\vec{\alpha} \in \mathbb{R}^d$. Also let $\phi = \phi_\varepsilon : \mathbb{R}_+^* \to \mathbb{R}_+^*$ represent any general RBF for the shape-parameter $\varepsilon \in \mathbb{R}_+^*$. Then for an N finite set of $\vec{\alpha}_i$ points, $\mathcal{W}$ can be decomposed for some fixed time t_f as

$$\mathcal{W}(\vec{\alpha}, t_f) = \sum_{i=1}^{N} \mathsf{w}_i \, \phi(\|\vec{\alpha} - \vec{\alpha}_i\|) = \begin{bmatrix} \phi(\|\vec{\alpha} - \vec{\alpha}_1\|) & \cdots & \phi(\|\vec{\alpha} - \vec{\alpha}_N\|) \end{bmatrix} \underline{\mathsf{w}}. \tag{4.1}$$

This is the continuous RBF decomposition and with an additional constraint, the weights w_i are uniquely determined by the position and number of points $\vec{\alpha}_i$ (Flyer and Wright, 2007).

For notational simplicity, let $\phi_{ji} = \phi(\|\vec{\alpha}_j - \vec{\alpha}_i\|)$ and $\mathcal{W}_j = \mathcal{W}(\vec{\alpha}_j, t_f)$ for any sampled $\vec{\alpha}_j$. Then if $\vec{\alpha}_j$ and $\vec{\alpha}_i$ belong to the same set, a discrete decomposition can be written as

$$\underline{\mathcal{W}} = \begin{bmatrix} \mathcal{W}(\vec{\alpha}_1, t_f) \\ \vdots \\ \mathcal{W}(\vec{\alpha}_N, t_f) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{N1} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} \mathsf{w}_1 \\ \vdots \\ \mathsf{w}_N \end{bmatrix} = \underline{\underline{\phi}} \, \underline{\mathsf{w}}.$$

Due to properties of RBFs, the interpolation matrix, $\underline{\underline{A}} = \underline{\underline{\phi}}$, is nonsingular and can be inverted to determine the set of interpolation weights $\underline{\mathsf{w}}$ (Bochner et al., 1959; Micchelli, 1986) as

$$\underline{\mathsf{w}} = \underline{\underline{A}}^{-1}\,\underline{\mathcal{W}}. \tag{4.2}$$

Now, consider the following linear first-order PDE:

$$\partial_t \mathcal{W}(\vec{\alpha}, t) + \mathcal{L}\mathcal{W}(\vec{\alpha}, t) = 0. \tag{4.3}$$

From Eq. (4.3), it follows

$$\mathcal{L}\,\mathcal{W}(\vec{\alpha}, t_f) = \sum_{i=1}^{N} \mathsf{w}_i \,\mathcal{L}\,\phi(\|\vec{\alpha} - \vec{\alpha}_i\|) = \begin{bmatrix} \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_1\|) & \cdots & \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_N\|) \end{bmatrix} \underline{\mathsf{w}}. \tag{4.4}$$

In discrete form this can be rewritten as

$$\underline{(\mathcal{L}\mathcal{W})} = \begin{bmatrix} \mathcal{L}\mathcal{W}(\vec{\alpha}, t_f)|_{\vec{\alpha}=\vec{\alpha}_1} \\ \vdots \\ \mathcal{L}\mathcal{W}(\vec{\alpha}, t_f)|_{\vec{\alpha}=\vec{\alpha}_N} \end{bmatrix} = \begin{bmatrix} (\mathcal{L}\phi)_{11} & \cdots & (\mathcal{L}\phi)_{1N} \\ \vdots & \ddots & \vdots \\ (\mathcal{L}\phi)_{N1} & \cdots & (\mathcal{L}\phi)_{NN} \end{bmatrix} \begin{bmatrix} \mathsf{w}_1 \\ \vdots \\ \mathsf{w}_N \end{bmatrix} = \underline{\underline{(\mathcal{L}\phi)}}\;\underline{\mathsf{w}},$$

with $(\mathcal{L}\phi)_{ji} = \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_i\|)|_{\vec{\alpha}=\vec{\alpha}_j}$. Let $\underline{\underline{B}} = \underline{\underline{(\mathcal{L}\phi)}}$. Remembering $\underline{\mathsf{w}} = \underline{\underline{A}}^{-1}\underline{\mathcal{W}}$, the differentiation matrix can be defined as $\underline{\underline{D}} = \underline{\underline{B}}\,\underline{\underline{A}}^{-1}$ and the linear PDE can be solved explicitly as

$$\partial_t \underline{\mathcal{W}} = -\underline{\underline{D}}\,\underline{\mathcal{W}}. \tag{4.5}$$

4.2.2 Local RBF-FD methodology

As previously mentioned, RBF-FD is a local method for solving PDEs and is very similar in implementation to the global RBF method. For each node in the global set $\Omega_{\mathrm{G}} = \{\vec{\alpha}_1, \ldots \vec{\alpha}_N\}$, let the subset Ω_s contain $n-1$ selected neighbors[2] to the node $\vec{\alpha}_s$. Then for each Ω_s, the RBF-FD method creates a set of FD weights that are exact at $\vec{\alpha}_s$ for the RBF centered at each node in the subset. To improve accuracy, the $n+1$ weights are calculated indirectly as

$$\begin{bmatrix} \phi(\|\vec{\alpha}_1 - \vec{\alpha}_1\|) & \cdots & \phi(\|\vec{\alpha}_1 - \vec{\alpha}_n\|) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \phi(\|\vec{\alpha}_n - \vec{\alpha}_1\|) & \cdots & \phi(\|\vec{\alpha}_n - \vec{\alpha}_n\|) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_1\|)\big|_{\vec{\alpha}=\vec{\alpha}_s} \\ \vdots \\ \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_n\|)\big|_{\vec{\alpha}=\vec{\alpha}_s} \\ \mathcal{L}1|_{\vec{\alpha}=\vec{\alpha}_s} \end{bmatrix},$$

[2] As it will be shown, the neighbors need not be the closest in the Euclidean norm. This is important for coupled domains where the node densities or group velocities may vary between them.

and the last weight is discarded (Fornberg et al., 2013). Let $\underline{\underline{A}}$ represent the interpolation matrix with an extra row and column and $b_i^s = \mathcal{L}\phi(\|\vec{\alpha} - \vec{\alpha}_i\|)|_{\vec{\alpha}=\vec{\alpha}_s}$. Then the differentiation weights can be written more compactly as

$$\underline{w^s} = \left(\underline{\underline{A}}^{-1}\,\underline{b^s}\right)^*, \tag{4.6}$$

where $(\cdot)^*$ denotes that the last entry is dropped. Let $\mathcal{W}^s \in \{\mathcal{W}(\vec{\alpha}, t_f)\,|\,\vec{\alpha} \in \Omega_s\}$. Then the local solution at each stencil becomes

$$\mathcal{L}\mathcal{W}(\vec{\alpha}, t_f)|_{\vec{\alpha}=\vec{\alpha}_s} = \begin{bmatrix} \mathcal{W}^s(\vec{\alpha}_1, t_f) & \dots & \mathcal{W}^s(\vec{\alpha}_n, t_f) \end{bmatrix} \underline{w^s} = \underline{w^s}^T \underline{\mathcal{W}^s}. \tag{4.7}$$

For efficiency, the locally computed weights are combined into one differentiation matrix Λ with Eq. (4.6) forming one row of the matrix for each $\vec{\alpha}_s$. The approximated linear differential operator becomes

$$\underline{(\mathcal{L}\mathcal{W})} \approx \underline{\underline{\Lambda}}\,\underline{\mathcal{W}}. \tag{4.8}$$

And finally to stabilize the RBF-FD method, a hyperviscosity filter of the form $\underline{\underline{H}} = \gamma\Delta^p$ is applied to the linear differential operator as

$$\partial_t \underline{\mathcal{W}} = -\underline{\underline{\Lambda}}\,\underline{\mathcal{W}} + \underline{\underline{H}}\,\underline{\mathcal{W}}. \tag{4.9}$$

For full details on hyperviscosity, see Fornberg and Lehto (2011), Flyer et al. (2012), and Bollig et al. (2012).

4.3 Problem formulation

To explore performance of the meshless spectral wave model, simplified versions of the wave action balance equation are solved numerically on several different geometries. The key goals here are twofold. The first is to measure and compare the meshless model's performance on the kinematic part of the wave action balance equation (i.e., no source terms). This largely involves studying the spatial and directional evolution of wave action, $\mathcal{W} = \mathcal{S}_{f\theta}/\sigma$, on a sphere. A study of frequency evolution will be postponed until the future since it is influenced by a conjunction of source terms and bathymetry. The second main goal here is to test the suitability of the RBF-FD method applied to both a non-radially-symmetric and higher-dimensional geometry. Care is needed to generate a stencil in the coupled spatial-spectral domain and ensure that it is not too large to implement.

In this section, the wave action balance equations formulated for the ring-point, ring-line, and sphere-ring geometries in Section 3.3 are discretized for implementation. In addition, the method of stencil selection and treatment of boundary conditions are discussed. For simplification, all test cases are conducted on aquaplanets[3] without background currents.

4.3.1 RBF-FD linear operator discretization

To simplify later calculations, the RBF-FD linear operator is discretized here generically. Following Flyer et al. (2012), Gaussian RBFs are used for all RBF-FD calculations; the RBF is defined as

$$\phi(\|\vec{\alpha}-\vec{\alpha}_i\|) = \exp\left[-\varepsilon^2 \|\vec{\alpha}-\vec{\alpha}_i\|^2\right], \tag{4.10}$$

where $\varepsilon \in \mathbb{R}_+^*$ is the shape parameter. For $\vec{\alpha} = (\alpha^1, \dots, \alpha^d)$, the α^γ-th partial derivative can be generalized as

$$\begin{aligned}
\partial_{\alpha^\gamma} \|\vec{\alpha}-\vec{\alpha}_i\| &= \partial_{\alpha^\gamma} \left[\left(\alpha^1-\alpha_i^1\right)^2 + \dots \left(\alpha^d-\alpha_i^d\right)^2\right]^{1/2} = \frac{(\alpha^\gamma-\alpha_i^\gamma)}{\|\vec{\alpha}-\vec{\alpha}_i\|}, \\
\partial_{\alpha^\gamma}\phi(\|\vec{\alpha}-\vec{\alpha}_i\|) &= -2\varepsilon^2 \left(\alpha^\gamma-\alpha_i^\gamma\right)\phi(\|\vec{\alpha}-\vec{\alpha}_i\|).
\end{aligned}$$

Let $\mathcal{L} = \partial_{\alpha^\gamma}$. Then the transpose of $\underline{b^s}$ from Eq. (4.6) can be written as

$$\begin{aligned}
\underline{b^{sT}} &= \begin{bmatrix} \partial_{\alpha^\gamma}\phi(\|\vec{\alpha}-\vec{\alpha}_1\|)|_{\vec{\alpha}=\vec{\alpha}_s} & \dots & \partial_{\alpha^\gamma}\phi(\|\vec{\alpha}-\vec{\alpha}_n\|)|_{\vec{\alpha}=\vec{\alpha}_s} & 0 \end{bmatrix} \\
&= -2\varepsilon^2 \begin{bmatrix} (\alpha_s^\gamma-\alpha_1^\gamma)\,\phi_{s1} & \cdots & (\alpha_s^\gamma-\alpha_n^\gamma)\,\phi_{sn} & 0 \end{bmatrix}.
\end{aligned}$$

This implies

$$\underline{w^{sT}} = \left(\left(\underline{\underline{A}}^{-1}\,\underline{b^s}\right)^*\right)^T = \left(-2\varepsilon^2 \begin{bmatrix} (\alpha_s^\gamma-\alpha_1^\gamma)\,\phi_{s1} & \cdots & (\alpha_s^\gamma-\alpha_n^\gamma)\,\phi_{sn} & 0 \end{bmatrix} \underline{\underline{A}}^{-1}\right)^*, \tag{4.11}$$

[3] This includes the use of ice caps which stabilize the singular equations.

and the local differential operator for each stencil becomes

$$\partial_{\alpha^\gamma}\mathcal{W}(\vec{\alpha},t_f)\big|_{\vec{\alpha}=\vec{\alpha}_s} = \left(\,\partial_{\alpha^\gamma}\mathcal{W}\right)_s = \underline{w^s}^T\underline{\mathcal{W}^s}. \tag{4.12}$$

4.3.2 Equation discretizations

To illustrate concretely what role the coupled domains may or may not play in the test cases, the wave action balance equations are discretized here for reference.[4] To simplify, the extra entries (that were added to reduce the condition number of matrix A) are implied in the single underline notation.

4.3.2.1 Ring-line discretization

Let $\vec{\alpha} = (\vec{x},k) = (x,y,k)$ with $\|\vec{x}\| = 1$ and $k \in \mathbb{R}$. In addition, for an unspecified radius R, assume deep-water conditions with $c_g(k) = \frac{1}{2}\sqrt{\mathrm{g}/k}$. Then from Eq. (3.60), the discretized wave action balance equation for the ring-line geometry is given by

$$\begin{aligned}
\left(\,\partial_t\mathcal{W}\right)_s &= \mp\frac{c_g(k_s)}{R}\left[-y_s\left(\,\partial_x\mathcal{W}\right)_s + x_s\left(\,\partial_y\mathcal{W}\right)_s\right] \\
&= \pm\frac{2\varepsilon^2 c_g(k_s)}{R}\left\{\left[\underline{-y_s\left(x_s-x_i\right)\phi_{si}} + \underline{x_s\left(y_s-y_i\right)\phi_{si}}\right]^T \underline{\underline{A}}^{-1}\right\}^* \underline{\mathcal{W}^s} \\
&= \pm\frac{2\varepsilon^2 c_g(k_s)}{R}\left\{\left[\underline{\left(y_s x_i - x_s y_i\right)\phi_{si}}\right]^T \underline{\underline{A}}^{-1}\right\}^* \underline{\mathcal{W}^s}.
\end{aligned} \tag{4.13}$$

4.3.2.2 Sphere-ring discretization

Assume deep-water conditions and let $\vec{\alpha} = \left(\vec{x},\vec{k}\right) = (x,y,z,k,l)$, R be the radius of Earth, and k_c be the fixed wavevector magnitude. In addition, the spatial and spectral coordinates are rescaled by $1/R$ and $1/k_c$ respectively such that $\|\vec{x}\| = 1$, $\left\|\vec{k}\right\| = 1$, and $c_g\left(\left\|\vec{k}\right\|\right) = c_g(k_c) = \frac{1}{2R}\sqrt{\frac{\mathrm{g}}{k_c}}$. Then from Eq. (3.63), the discretized wave action balance equation for the sphere-ring geometry is

[4] The ring-point discretization was not included since it is a subset of the ring-line geometry.

given by

$$
\begin{aligned}
(\partial_t \mathcal{W})_s &= \mp \frac{c_g(k_c)}{\sqrt{x_s^2+y_s^2}} \Big[(-y_s k_s - x_s z_s l_s)(\partial_x \mathcal{W})_s + (x_s k_s - y_s z_s l_s)(\partial_y \mathcal{W})_s \\
&\qquad\qquad + l_s\left(x_s^2+y_s^2\right)(\partial_z \mathcal{W})_s + z_s k_s l_s (\partial_k \mathcal{W})_s - z_s k_s^2 (\partial_l \mathcal{W})_s \Big] \\
&= \pm \frac{2\varepsilon^2 c_g(k_c)}{\sqrt{x_s^2+y_s^2}} \Bigg\{ \Big[\underline{(-y_s k_s - x_s z_s l_s)\,(x_s - x_j)\,\phi_{sj}} \\
&\qquad\qquad + \underline{(x_s k_s - y_s z_s l_s)\,(y_s - y_j)\,\phi_{sj}} + \underline{l_s\left(x_s^2+y_s^2\right)(z_s - z_j)\,\phi_{sj}} \\
&\qquad\qquad + \underline{(z_s k_s l_s)\,(k_s - k_j)\,\phi_{sj}} - \underline{\left(z_s k_s^2\right)(l_s - l_j)\,\phi_{sj}} \Big]^T \underline{\underline{A}}^{-1} \Bigg\}^* \underline{\mathcal{W}^s} \\
&= \pm \frac{2\varepsilon^2 c_g(k_c)}{\sqrt{x_s^2+y_s^2}} \Bigg\{ \Big[\underline{(y_s k_s + x_s z_s l_s)\,x_j \phi_{sj}} - \underline{(x_s k_s - y_s z_s l_s)\,y_j \phi_{sj}} \\
&\qquad\qquad - \underline{\left(x_s^2+y_s^2\right) l_s z_j \phi_{sj}} - \underline{z_s k_s l_s k_j \phi_{sj}} + \underline{z_s k_s^2 l_j \phi_{sj}} \Big]^T \underline{\underline{A}}^{-1} \Bigg\}^* \underline{\mathcal{W}^s}. \qquad (4.14)
\end{aligned}
$$

4.3.3 Node stencil construction for the sphere-ring geometry

Generation of the RBF-FD stencils plays a large role in the design of the meshless wave model for the sphere-ring geometry. Since the spatial and spectral domains are coupled, care is needed to ensure that evolution of wave action is appropriately resolved in each. Here, the method in Flyer et al. (2012) (i.e., stencil construction for advection on a sphere) is adapted to account for directional evolution of wave action.

In general, RBF-FD stencils are constructed as follows. For each node in a global set Ω_{G}, a subset Ω_s of $n-1$ neighbors are selected for node $\vec{\alpha}_s$. Then for each Ω_s, a unique stencil (a set of FD weights) is calculated beforehand that is exact at $\vec{\alpha}_s$ for each RBF centered at $\vec{\alpha} \in \Omega_s$. In practice, the largest differentiation weights (absolute value) are at nodes closest to $\vec{\alpha}_s$. See Fig. 4.1 for a sample distribution of weights on a sphere.

The nodes in Ω_s do not necessarily need to be the nearest neighbors of α_s however. This is particularly true for problems with inherently different length scales in a single or coupled domain. In the meshless wave model, several different types of node selection criteria were tested, ranging from simple (all spectral nodes included) to sophisticated (use of the Mahalanobis distance).[5] The

[5] The norm is weighted to take into account skewness or anisotropy.

best approach tested was a separate selection process that first selected the spatial locations and then selected nodes based on the desired spectral directions. This guaranteed a consistent number of nodes in each domain and allowed separate tuning to control error convergence rates as needed.

An outline of the stencil construction is as follows. First, global nodes are generated separately for each domain. Maximum determinant (MD) nodes (Sloan and Womersley, 2004) are used for the surface of the sphere while the spectral ring is discretized uniformly. Second, a combined global node set of size $N_{\vec{\alpha}} = N_{\vec{x}} \times N_{\vec{k}}$ is formed. Third, the nearest $n_{\vec{x}} - 1$ spatial locations are chosen for each stencil using the k-D tree package in Matlab (Shechter, 2004). Fourth, the radially nearest $n_{\vec{k}} - 1$ neighbors in the spectral domain are selected. Fifth, the differential weights for a stencil of size $n_{\vec{\alpha}} = n_{\vec{x}} \times n_{\vec{k}}$ are then calculated. And finally, individual stencils are collected (as they are generated) to form a sparse $N_{\vec{\alpha}} \times N_{\vec{\alpha}}$ differentiation matrix. It should be noted that the model can be tuned by altering the sizes $n_{\vec{x}}$ and $n_{\vec{k}}$ while holding $n_{\vec{\alpha}}$ constant.

4.3.4 Boundary attenuation for the sphere-ring geometry

The tangent-plane projection of the wave action balance equation (Eq. 3.63) requires polar caps to prevent evolution of wave action near the singular regions. This is achieved here by attenuating the wave action inside a specified region at each time step. To minimize reflection at the boundary, a smooth transition is used between the unattenuated and attenuated regions.

Since the singularity is due to a tangent-plane projection, it is natural to use geophysical coordinates to define the attenuation. Let (λ, μ, r) represent $\vec{x}$ with longitude $\lambda \in (-\pi, \pi]$ and latitude $\mu \in [-\pi/2, \pi/2]$. The boundary attenuation filter is applied globally after every time step and is defined generically as

$$\mathcal{A}(\mu) = \begin{cases} 1 - \frac{4}{3}\mathcal{B}(\mu), & \mathcal{B}(\mu) \leq \frac{3}{4} \\ 0, & \text{otherwise.} \end{cases} \tag{4.15}$$

for $\mathcal{B} \in [0, 1]$. The 4/3 factor is arbitrarily chosen to ensure that a minimum number of nodes near the poles are fully attenuated. Here a Gaussian bell with approximate diameter d is used to define

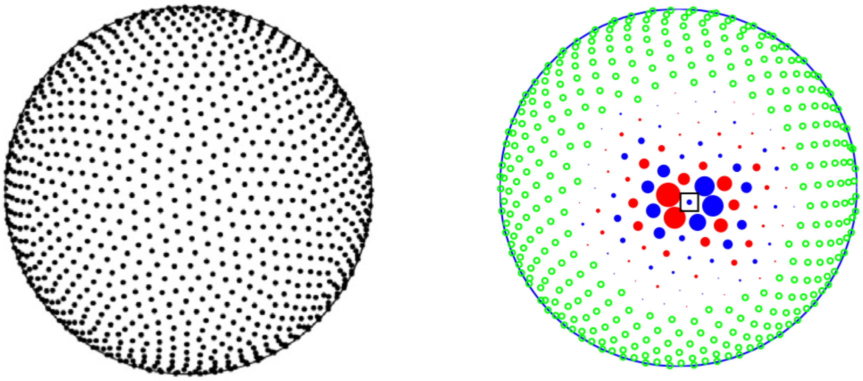


Figure 4.1: (a) Sample minimal energy node distribution (b) overlaid with RBF-FD differentiation weights for a selected node (boxed). The scaled blue and red solid circles correspond to negative and positive values respectively and the green circles represent zero entries in the differentiation matrix. Image is reproduced from Flyer et al. (2012).

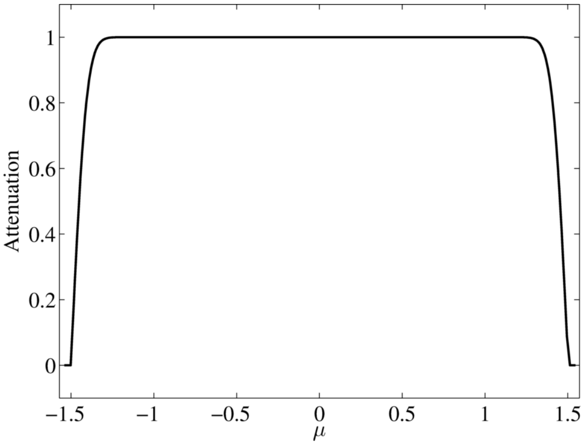


Figure 4.2: A boundary attenuation filter is used to prevent evolution of wave action near the singular poles. The ice lines are approximately at ±75°. See Eq. (4.15) for details.

$\mathcal{B}$ as[6]

$$\mathcal{B}(\mu) = \exp\left[-\left\{\frac{9}{2d}\left[\mu - \text{sgn}^{*}(\mu)\frac{\pi}{2}\right]\right\}^2\right], \quad \mu \in [-\pi/2, \pi/2], \tag{4.16}$$

but any sufficiently smooth curve would suffice. A standard value of $d = \pi/6$ (i.e., ±75° ice line) is used in most tests here. See Fig. 4.2 for an illustration.

4.4 Numerical test case studies

Several different numerical test cases are studied here. The first is a toy problem designed to see how a meshless method might perform on the wave action balance equation and uses RBFs on the ring-line geometry. All test cases afterwards use the RBF-FD method. The second and third cases are primarily performed to guide stencil selection for the meshless wave model. A brief fourth case examines boundary attenuation. The fifth case analyzes model performance for a simple initial condition and a comparison is made with WAVEWATCH III in the final case. Compared to previously mentioned studies, the stencil and global node set sizes are relatively small here and is necessitated by the use of the coupled domain in the sphere-ring geometry.

4.4.1 Case 1: Toy problem

An initial test using RBFs was conducted to test the viability of a meshless wave model. Eq. (4.13) was tested using a sparse $40_{\vec{x}} \times 20_k$ node set (800 unknowns) in the ring-line geometry. The nodes were staggered in the spectral domain both uniformly and logarithmically to account for different group velocities. Interpolated solutions were generated using $100_{\vec{x}} \times 100_k$ Halton nodes to test the practicality of generating output at various resolutions for model coupling (which did not pose any problems). Sample interpolated solutions and error are displayed in Figs. 4.3 and 4.4. As expected, the solution appears to be uncoupled with little evidence of numerical dispersion between the spatial and spectral domains.

[6] The sign function is defined slightly different here: $\text{sgn}^{*}(0) = 1$.

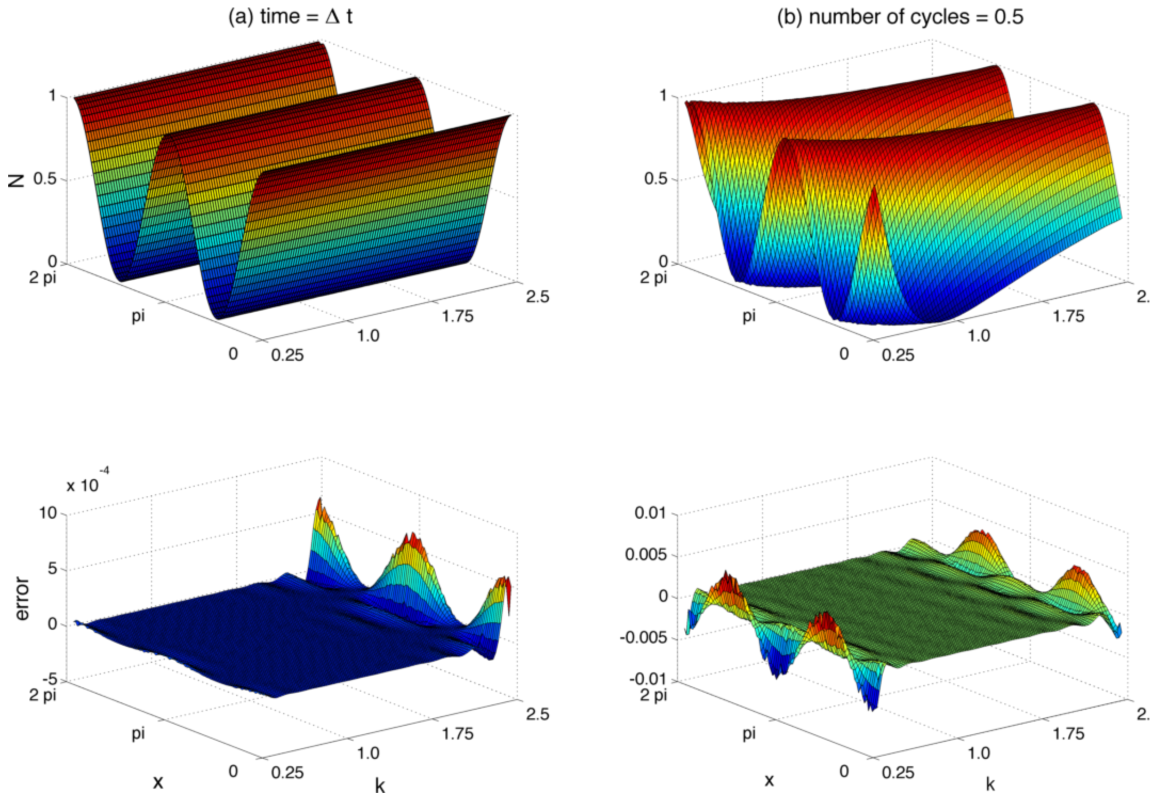


Figure 4.3: Sparse solution and error using $40_{\vec{x}} \times 20_k$ nodes with a staggered layout in k for logarithmically increasing Δk after (a) 1 time step and (b) 1/2 revolution of the fastest wave. The interpolated solution uses $100_{\vec{x}} \times 100_k$ Halton nodes.

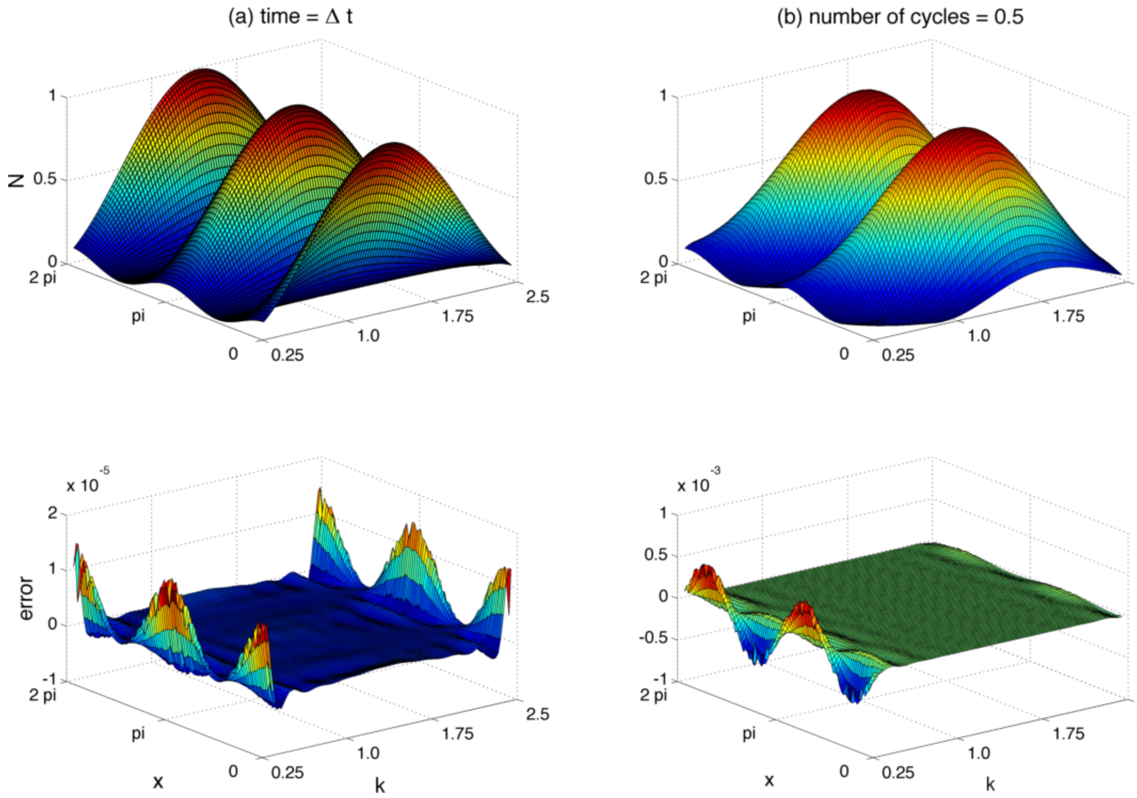


Figure 4.4: Sparse interpolated solution and error using $40_{\vec{x}} \times 20_k$ nodes with a staggered layout in k for fixed Δk after (a) 1 time step and (b) 1/2 revolution of the fastest wave. Interpolated solution uses $100_{\vec{x}} \times 100_k$ Halton nodes.

4.4.2 Case 2: Spectral stencil selection

In the second case, Eq. (4.13) with a constant wavenumber k_c is used to test simple propagation in the ring-point geometry. Results from these tests are later used to guide spectral node selection in the sphere-ring geometry and tuning of the model. For convenience, $\theta = \text{arctan2}[y, x]$ is used for discussion instead of $\vec{x}$.

Three different initial conditions are tested here with varying degrees of smoothness (two in C^∞ and one in C^1) and are displayed in the first column of Fig. 4.5. In each, the diameter (or approximate) of the humps is set by $d_2 = \pi/2$.

- **Cosine squared**: $\mathcal{W}_0(\theta) = \cos^2[\pi\theta/d]$.
- **Gaussian bell**: $\mathcal{W}_0(\theta) = \exp\Big[-(9\theta/2d)^2\Big]$.
- **Cosine bell**: $\mathcal{W}_0(\theta) = \cos^2[\pi\theta/d]$ for $|\theta| < d/2$; 0 otherwise.

The latter two are standard initial conditions for advection on sphere (Flyer and Wright, 2007; Fornberg and Piret, 2008). The Gaussian bell is of most interest here since empirically-derived frequency spectra use exponential functions to fit observational data (see Webb and Fox-Kemper, 2011) and the model is designed to exploit these smooth expected solutions. For each initial condition, optimal values are explored for the time step and shape parameter and node refinement convergence rates are calculated for spectral stencil sizes $n_\theta = \{3, 5, 7, 9, 11\}$.

In the second column of Fig. 4.5, the relative ℓ_2 errors (normalized by the ℓ_2 norm of the analytic solution) are calculated for different time step sizes after 1/4 revolution. In all three comparisons, $\varepsilon = 2$ and $N_\theta = 60$ are used. In addition, the time step is normalized by the propagation speed around the unit circle such that $\Delta t = a\,\Delta\theta/c_g = a\,2\pi/N_\theta c_g$. In Fig. 4.5d, the smooth Gaussian bell exhibits typical behavior. As Δt is reduced, the relative error is reduced until the stencil error dominates. This behavior is not as evident in Fig. 4.5b since the stencil error dominates (mostly) for the larger time steps as well. In Fig. 4.5f, the relative error from approximating the C^1 function likely dominates and is mostly invariant to changes in Δt.

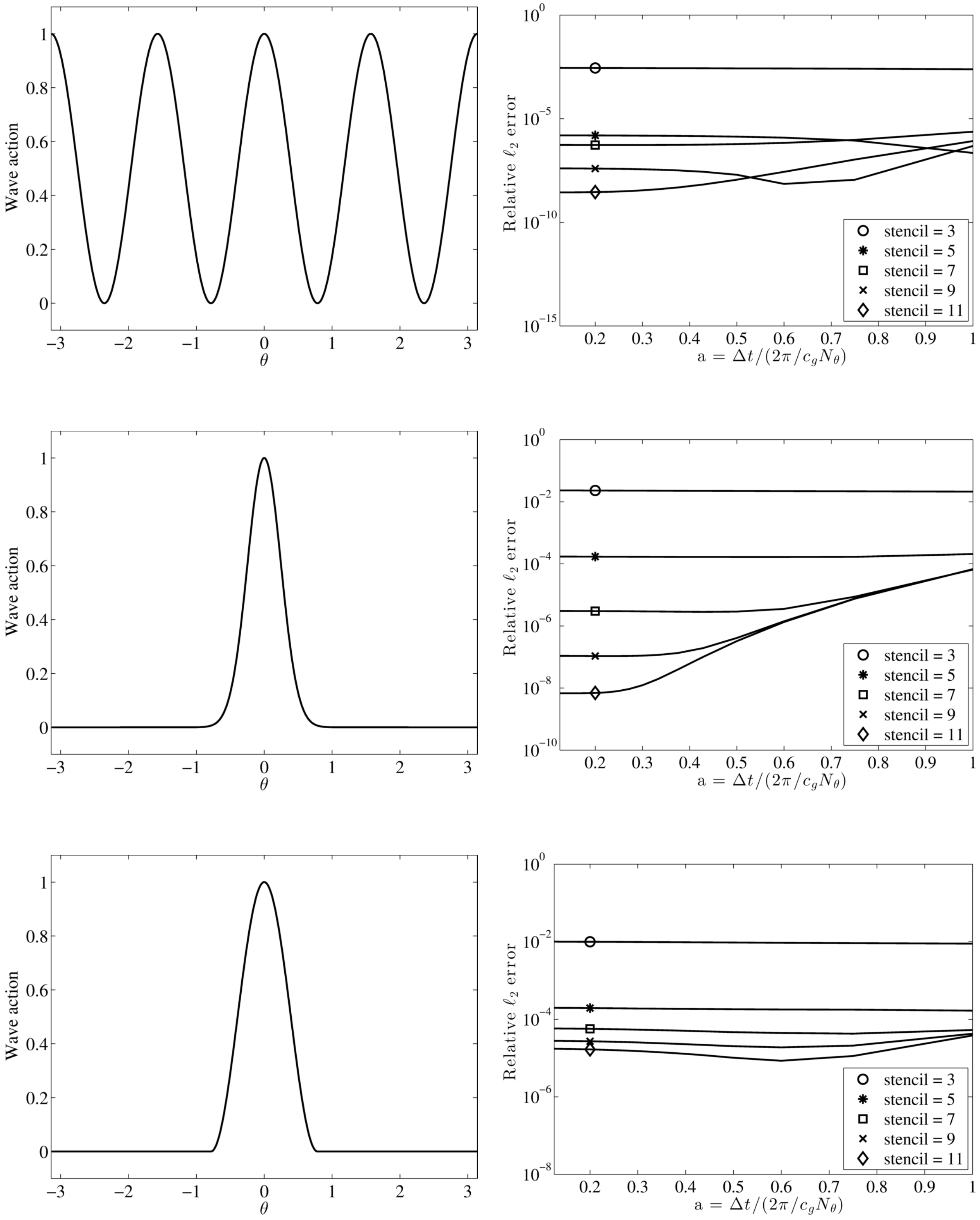


Figure 4.5: **Initial conditions** (first column) for the **ring-point** 1D periodic tests with **relative time step** error after 1/4 a revolution for $N_\theta = 60$ (second column): (a) cosine squared, $\mathcal{W}_0(\theta) = (\cos 2\theta)^2$; (c) Gaussian bell, $\mathcal{W}_0(\theta) = \exp[-(9\theta/\pi)^2]$; (e) cosine bell, $\mathcal{W}_0(\theta) = (\cos 2\theta)^2$ for $|\theta| < \pi/4$ and 0 otherwise. The time step is normalized by the propagation speed in (b), (d), and (f).

For each stencil size in each initial condition, the shape parameter is varied for different global node sizes in Figs. 4.6a–e, 4.7a–e, and 4.8a–e. The dashed and solid lines indicate the relative ℓ_2 error after 1 time step and a 1/4 revolution respectively. In Figs. 4.6f, 4.7f, and 4.8f, the global node refinement convergence rates are also displayed for $N_\theta \in [24, 90]$ and $\varepsilon = 2$. In all three initial conditions, the ratio $a = 0.2$ is used to delineate errors from the choice of Δt. It should be noted that the error and convergence rates are highly dependent on the width d and these results are not meant to be definitive but merely an intuitive guide for later tuning of the wave model.

In Figs. 4.7a–e, the behaviors of the different stencil sizes are similar; this is true not only for d_2 (displayed) but other values as well. In addition, a shape parameter of $\varepsilon = 2$ seems to be ideal for the spectral ring in the meshless wave model. Furthermore, the convergence rates exhibit the same properties of spectrally-accurate RBFs for this initial condition; as N_θ is increased, the convergence rates also increase. Here the rates are approximately 3rd, 7th, 11th, 13th and 15th order for the different stencil sizes.

In Figs. 4.6a–e, the behavior is more complicated but still similar in all. There is a large accuracy gain in moving from $n_\theta = 3$ to $n_\theta = 5$ and $n_\theta = 5$ to $n_\theta = 7$. This is also evident in the convergence rates in Fig. 4.6f. The rates of the first four are approximately 4th, 7th, 16th, and 14th order. For smaller values of d, Fig. 4.6f resembles Fig. 4.7f. Like Figs. 4.7a–e, the behavior of the different stencil sizes in Figs. 4.8a–e are also similar for d_2 and other values. Since the cosine bell is not continuously differentiable, the convergence rates are all approximately the same (as expected) at 4th order for d_2.

4.4.3 Case 3: Spatial stencil selection

In the third case, Eq. (4.14) is used to test propagation along the equator in the sphere-ring geometry. Recall $\vec{\alpha} = \left(\vec{x}, \vec{k}\right)$ with $\vec{k} = (k, l)$ and $\left\|\vec{k}\right\| = k_c$. For $\vec{k}_0 = (\pm k_c, 0)$ and an initial condition $\mathcal{W}_0(\vec{\alpha}_0) \neq 0$, propagation along the equator is largely invariant to the spectral discretization. As a result, this is an ideal case to test spatial stencil selection and guide later model tuning.

Here, modified Gaussian and cosine bells from the previous numerical test case are used for

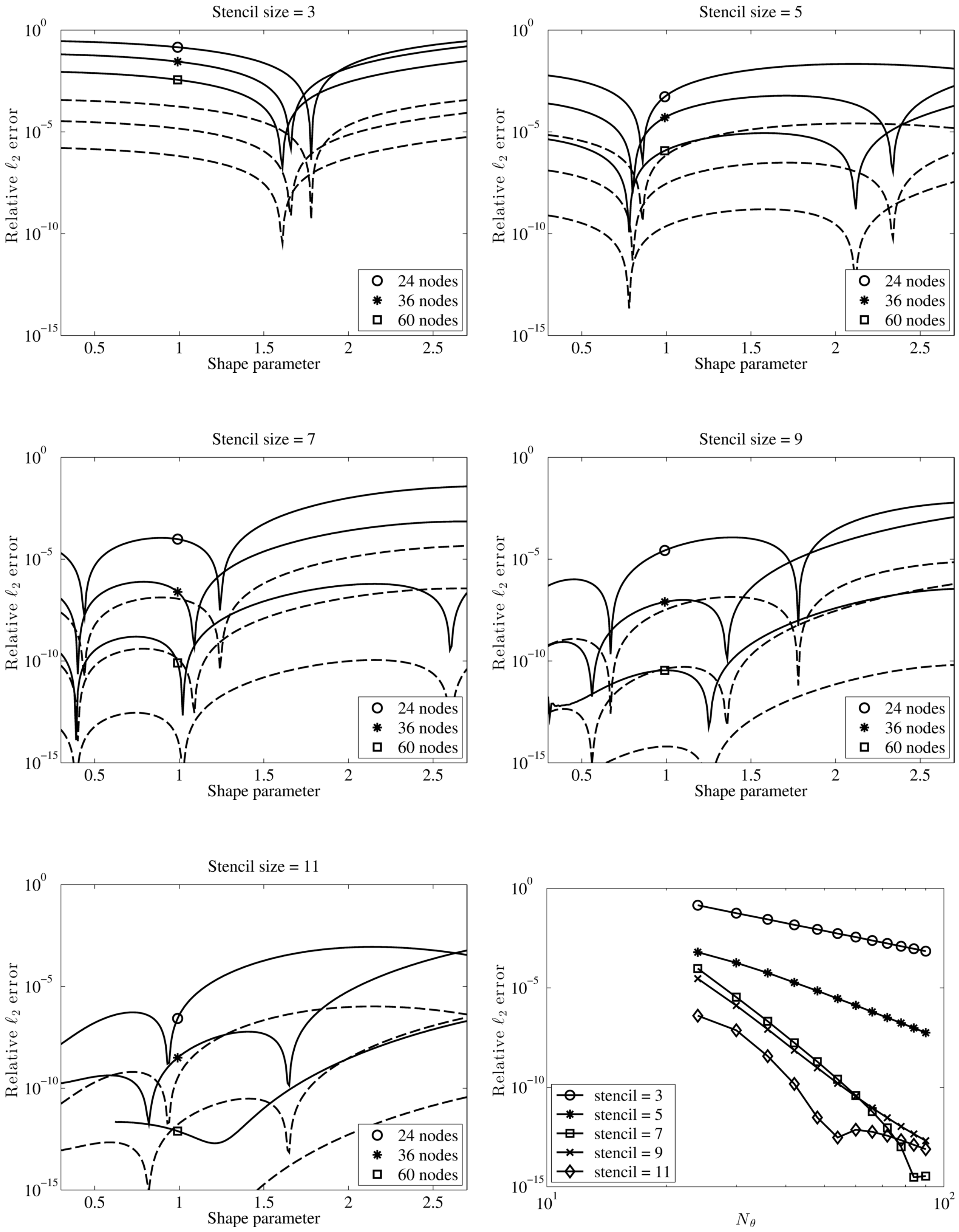


Figure 4.6: **Ring-point** test with initial condition $\mathcal{W}_0(\theta) = (\cos 2\theta)^2$. In (a)–(e), the relative ℓ_2 error after 1 time step (dashed) and 1/4 revolution (solid) are plotted versus **shape parameter** for different spatial nodes. A value of $a = 0.2$ from Fig. 4.5b was used to determine the time step in each. In (f), the relative ℓ_2 error after 1/4 revolution is plotted versus N_x for different stencil sizes.

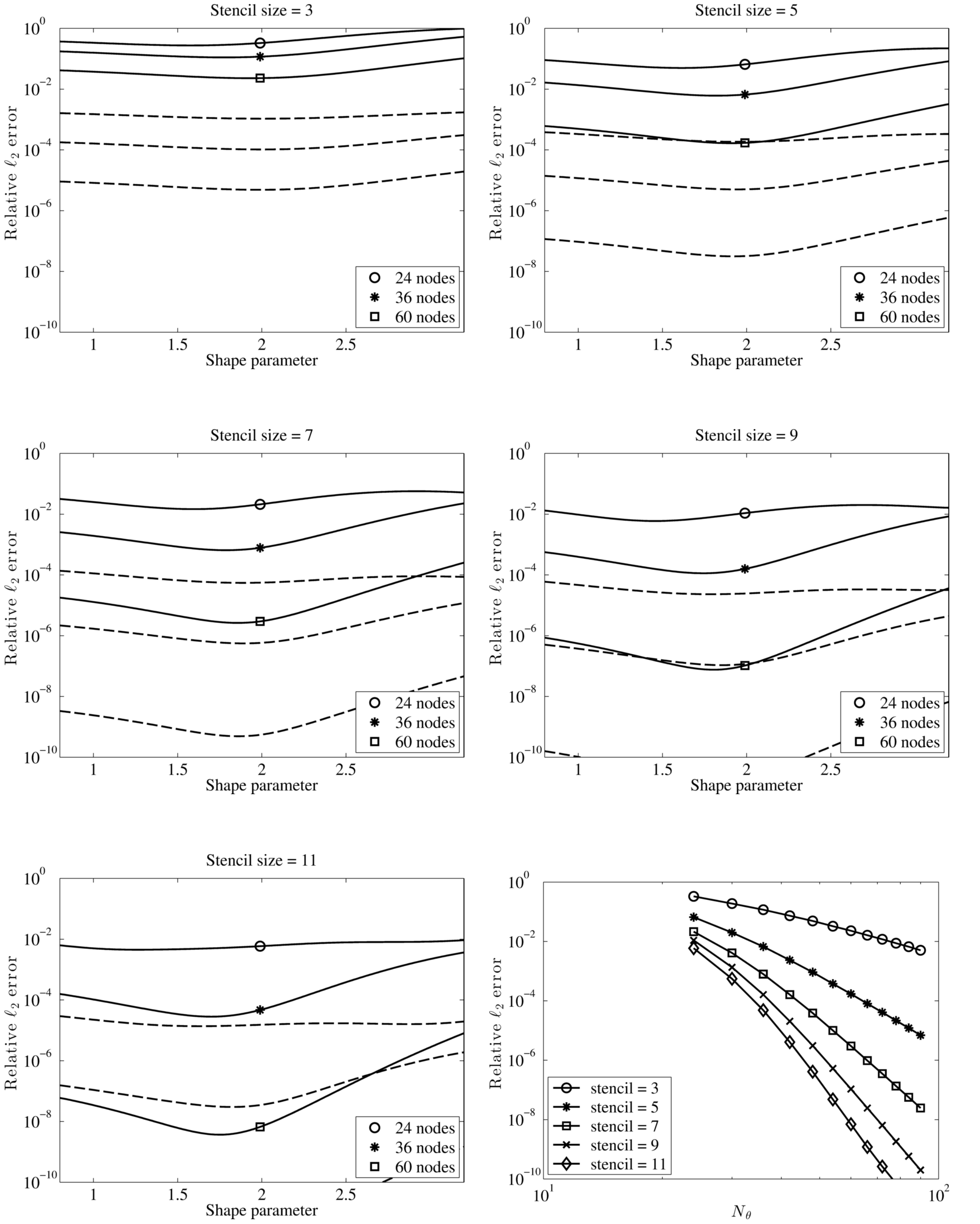


Figure 4.7: **Ring-point** test with initial condition $\mathcal{W}_0(\theta) = \exp[-(9\theta/\pi)^2]$ (Gaussian bell). In (a)–(e), the relative ℓ_2 error after 1 time step (dashed) and 1/4 revolution (solid) are plotted versus **shape parameter** for different spatial nodes. A value of $a = 0.2$ from Fig. 4.5d was used to determine the time step in each. In (f), the relative ℓ_2 error after 1/4 revolution is plotted versus N_x for different stencil sizes.

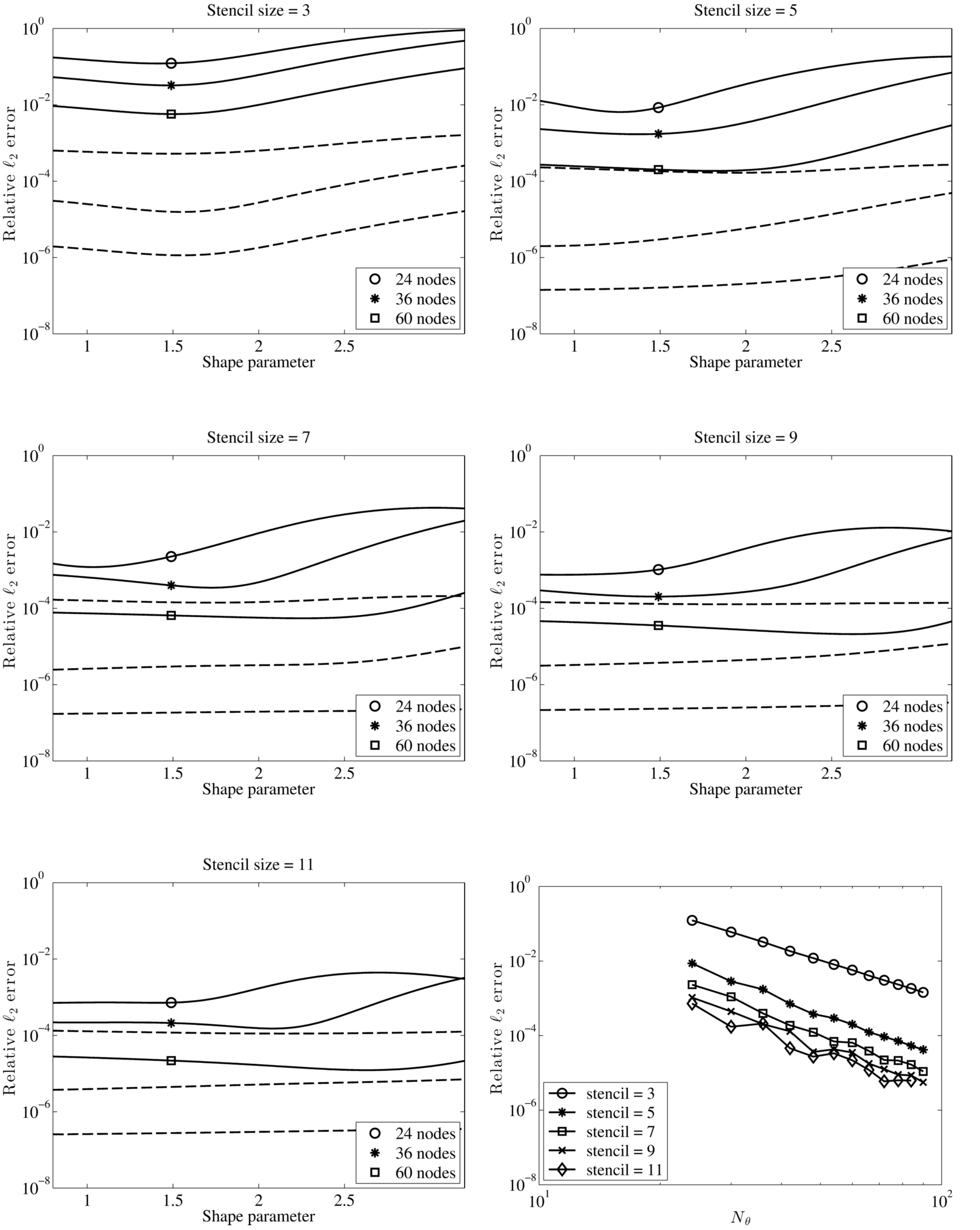


Figure 4.8: **Ring-point** test with initial condition $\mathcal{W}_0(\theta) = (\cos 2\theta)^2$ for $|\theta| < \pi/4$ and 0 otherwise (cosine bell). In (a)–(e), the relative ℓ_2 error after 1 time step (dashed) and 1/4 revolution (solid) are plotted versus **shape parameter** for different spatial nodes. A value of $a = 0.2$ from Fig. 4.5f was used to determine the time step in each. In (f), the relative ℓ_2 error after 1/4 revolution is plotted versus N_x for different stencil sizes.

initial conditions (Flyer and Wright, 2007). For convenience, geophysical spherical coordinates[7] are used here to define the following initial radially-symmetric coordinate centered at longitude λ_0 and latitude μ_0:

$$\xi(\lambda, \mu; \lambda_0, \mu_0) = \arccos[\sin(\mu_0)\sin(\mu) + \cos(\mu_0)\cos(\mu)\cos(\lambda - \lambda_0)]\,.$$

As previously, d specifies the initial bell diameter (or approximate) and is set here as $d_3 = \pi/3$.

- **Gaussian bell**: $\mathcal{W}_0(\xi) = \exp\Big[-(9\xi/2d)^2\Big]$.
- **Cosine bell**: $\mathcal{W}_0(\xi) = \cos^2[\pi\xi/d]$ for $|\xi| < d/2$; 0 otherwise.

Sample initial conditions are displayed in Figs. 4.9a and 4.10a with the propagation path, bell edge, and ice lines marked in red, blue and black respectively. Similar tests are performed as in the last numerical test case for the following commonly used spatial stencil sizes: $n_{\vec{x}} = \{17, 31, 50\}$. Other stencil sizes were tested but the former performed best in this problem and did not need hyperviscosity for stability.

In Figs. 4.9b and 4.10b, the relative ℓ_2 errors are calculated for different time step sizes after a 1/4 revolution with $\varepsilon = 3$ and $N_{\vec{x}} = 3600$. The time step is normalized by the propagation speed along the great circle path such that $\Delta t = a\,\Delta\vec{x}/c_g \approx a\,2\pi/c_g\sqrt{N_{\vec{x}}}$. In Fig. 4.9b, the Gaussian bell initial condition again exhibits typical behavior for $a \lessapprox 0.7$. The model is not expected to perform well for $a > 0.5$ and it is no surprise that relative errors are large for ratios across this threshold. In Fig. 4.10b, the cosine bell initial condition is largely invariant to time step size changes and this is also not surprising. Here, the $n_{\vec{x}} = 50$ stencil has a relative error comparable with $n_{\vec{x}} = 17$ and this is caused by the non-optimal shape parameter (for $n_{\vec{x}} = 50$) used for the comparison.

For each stencil size in each initial condition, the shape parameter is varied again for different global node sizes in Figs. 4.9c–e and 4.10c–e. The dashed and solid lines indicate the relative ℓ_2 error after 1 time step and a 1/4 revolution respectively. In Figs. 4.9f and 4.10f, the global node refinement convergence rates are also displayed for $N_{\vec{x}} \in [1225, 4900]$ and $\varepsilon = 3$. In all comparisons, the ratio $a = 0.2$ is used again to separate errors from the choice of Δt.

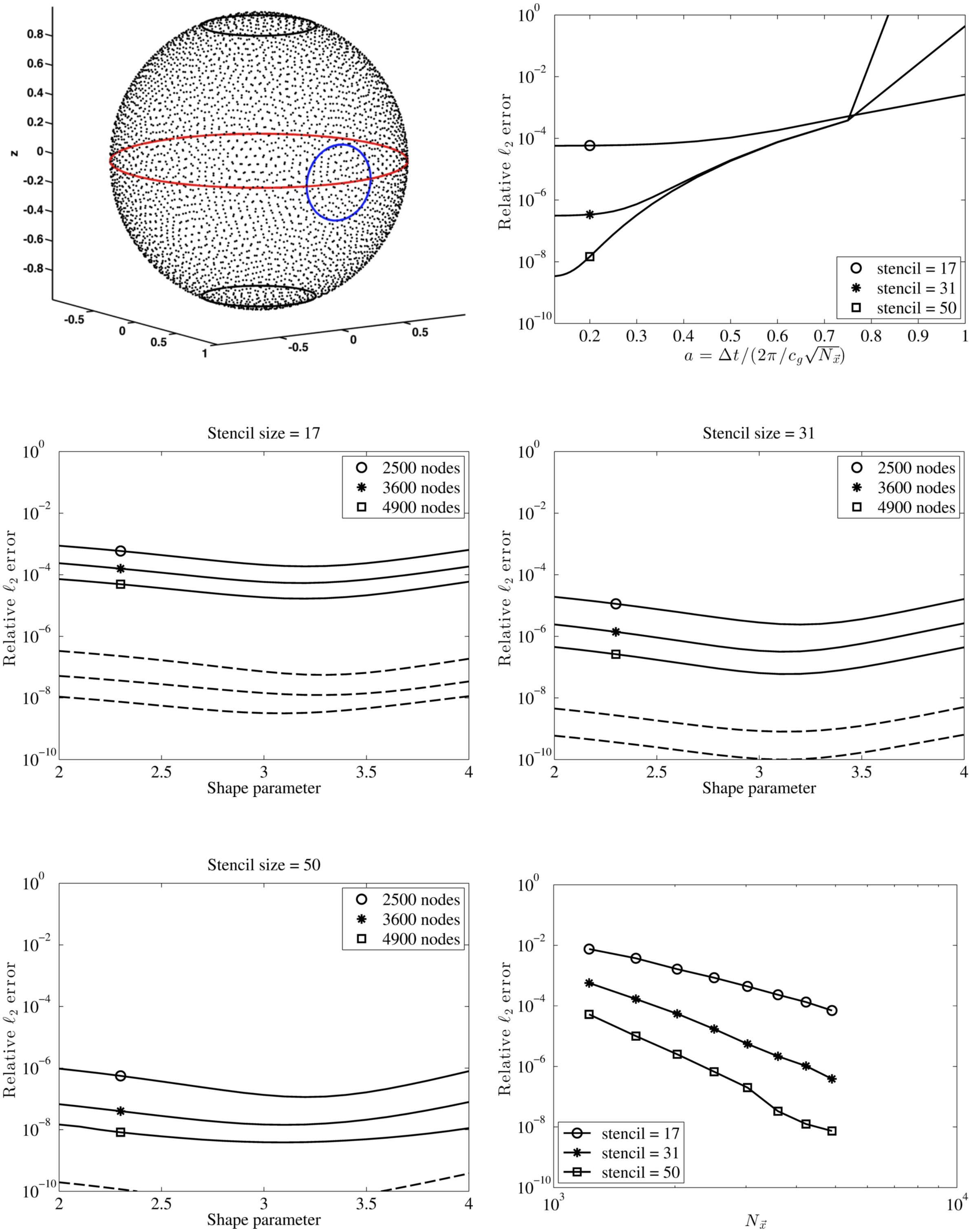


Figure 4.9: **Sphere-ring test** along the equator with a Gaussian bell initial condition, $\mathcal{W}_0(\xi) = \exp[-(27\xi/2\pi)^2]$. In (a), sample node layout with the **test path** (red), (approximate) **initial bell edge** (blue), and **ice cap edges** (black) are marked. Actual tests used a larger Gaussian bell and smaller ice caps than displayed. In (b), relative errors (ℓ_2) after 1/4 revolution are plotted versus a **relative time step** for $N_x = 3600$ and different stencil sizes. In (c)–(e), relative errors (ℓ_2) after 1 time step (dashed) and 1/4 revolution (solid) are plotted versus **shape parameter** for different $N_{\vec{x}}$ with $a = 0.2$. And in (f), the **spatial node convergence rates** are plotted for different stencils sizes.

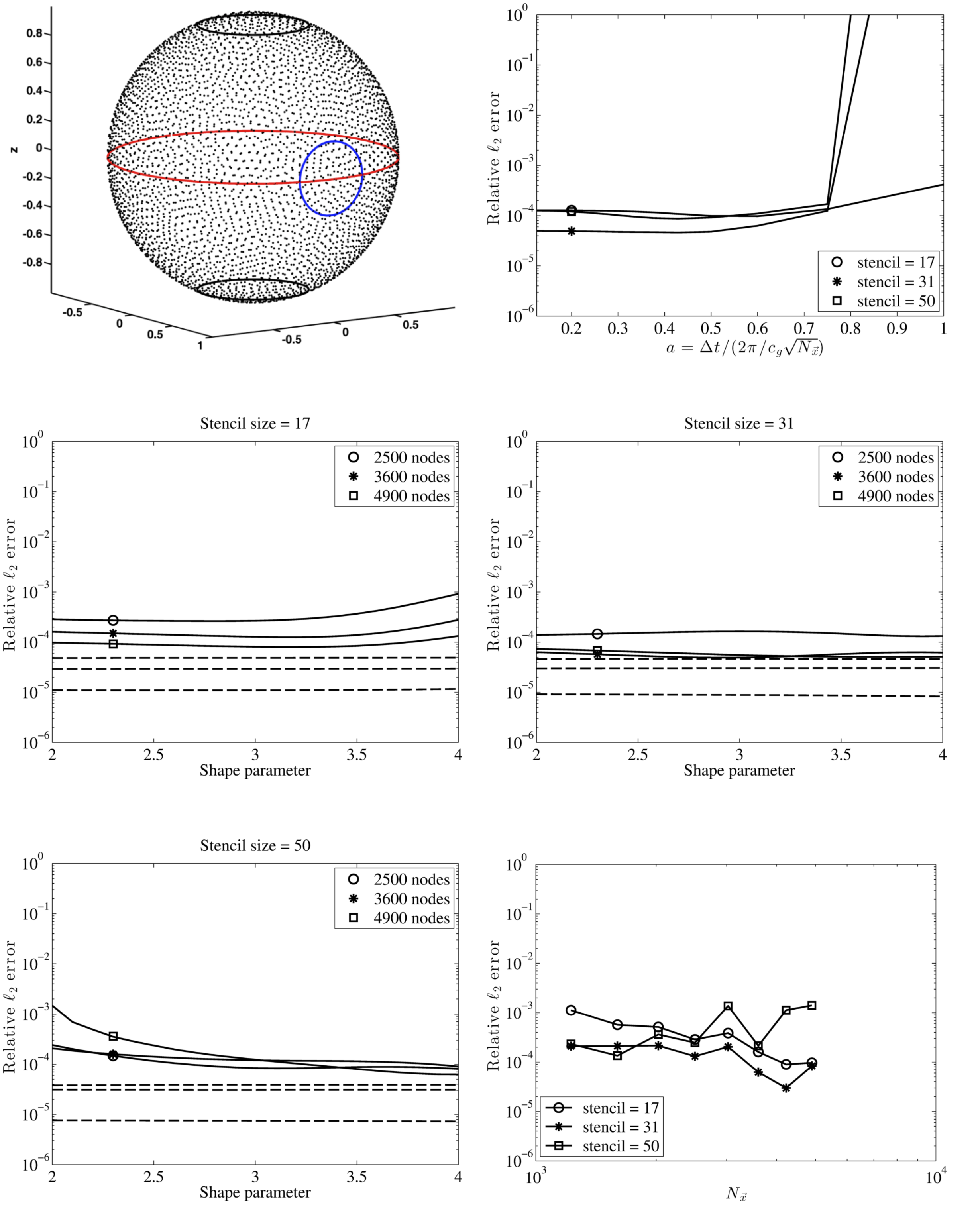


Figure 4.10: **Sphere-ring test** along the equator with a cosine bell initial condition, $\mathcal{W}_0(\xi) = (\cos 3\xi)^2$ for $|\xi < \pi/6$ and 0 otherwise. In (a), sample node layout with the **test path** (red), **initial bell edge** (blue), and **ice cap edges** (black) are marked. Actual tests used a larger Gaussian bell and smaller ice caps than displayed. In (b), relative errors (ℓ_2) after 1/4 revolution are plotted versus a **relative time step** for $N_x = 3600$ and different stencil sizes. In (c)–(e), relative errors (ℓ_2) after 1 time step (dashed) and 1/4 revolution (solid) are plotted versus **shape parameter** for different $N_{\vec{x}}$ with $a = 0.2$. And in (f), the **spatial node convergence rates** are plotted for different stencils sizes.

The choice of $\varepsilon = 3$ used in both the time step and node refinement comparisons is based on results from Figs. 4.9c–e. In Figs. 4.10c–e, there does not appear to be an optimal ε for all three stencils examined. Moreover, the relative errors are similar between the different stencils with no clear order of convergence. However in Figs. 4.9c–e, the relative error differences are evident with approximately 3rd, 5th, and 6th order convergence rates.

4.4.4 Case 4: Boundary attenuation

In the fourth case, the boundary attenuation filter (Section 4.3.4) is briefly tested to ensure that wave action is properly attenuated before reaching the pole singularity in wavenumber space (notice there are no pole singularities in physical space). Here, Eq. (4.14) is used to track wave action with an initial condition $\mathcal{W}_0(\vec{\alpha}_0) \neq 0$ and $\vec{k}_0 = (0, \pm k_c)$ as it traverses through the boundary region. Similar to Case 3, propagation along a strictly northern or southern path is largely invariant to spectral discretization. As such, a $50_{\vec{x}} \times 2_k$ stencil is used here with $4000_{\vec{x}} \times 4_k$ global nodes (16,000 unknowns) and a $\pm 70°$ ice line. Moreover, the C^1 cosine bell with width $d_4 = \pi/6$ is used as an initial condition to ensure the test is stringent (see Section 4.4.3).

Snapshots of the wave action (with direction $\vec{k} = (0, k_c)$) along the great circle path and its associated error are shown for select time steps in the first and second columns respectively of Fig. 4.11. To generate the images and calculate the error along the great circle path, the solution is interpolated to a new grid using 10,000 Halton nodes. In addition, the beginning of the boundary attenuation and the analytic solution are displayed in solid gray lines and curves for reference. The first and last rows show the gridded wave action (and the associated error) before and after (approximately) the wave action is attenuated with steps in between.

The test presented is meant to be illustrative and not comprehensive. Rigorous testing is still needed to ensure that wave action never reaches the singular poles and that reflection is minimal near the boundaries. So far neither are shown to be a problem during general use of the model.

[7] See Sec. 4.3.4 for definition.

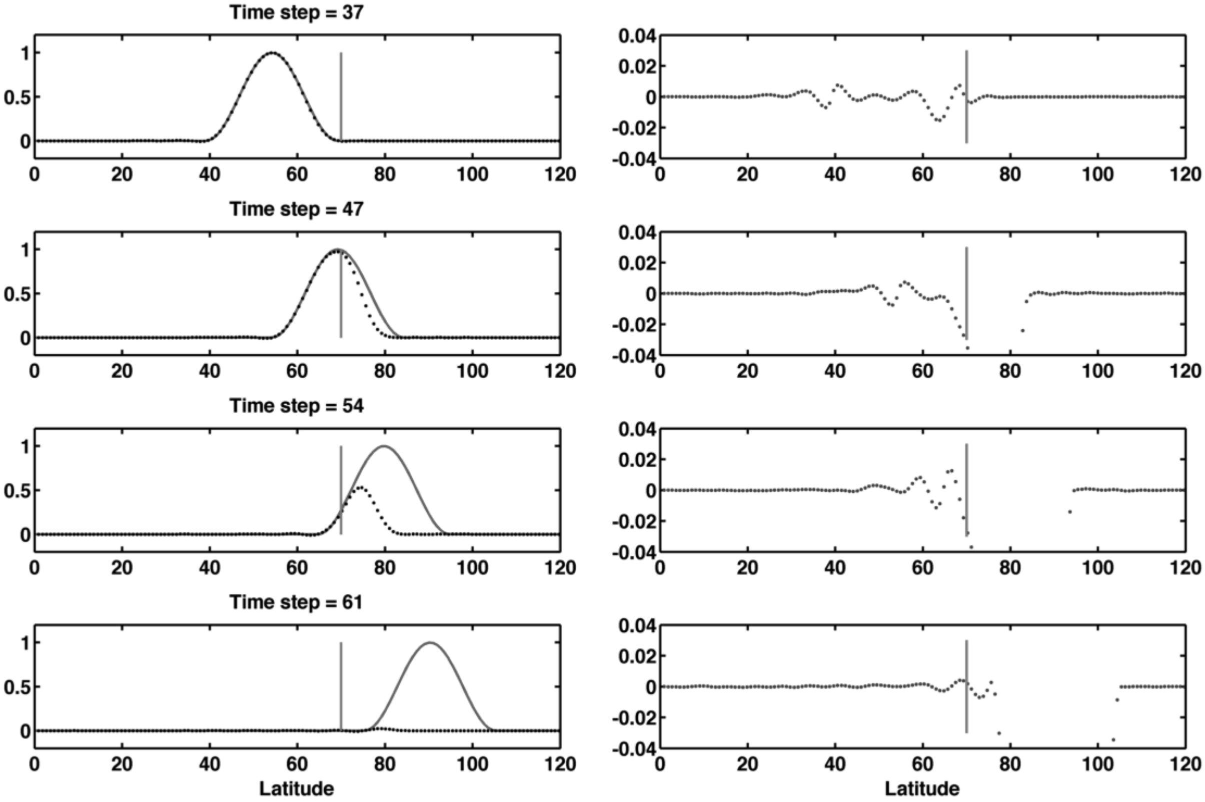


Figure 4.11: The Boundary attenuation filter is tested to ensure that wave action is properly attenuated before reaching the singular pole. Here (a) the wave action and (b) corresponding error are shown at different time steps along a great circle path. The ice line (or edge of the attenuation filter) is situated at ±70° and a cosine bell centered at $(\lambda = 0, \mu = 0)$ with width $d_4 = \pi/6$ and direction $\vec{k} = (0, k_c)$ is used for the initial condition. The solution is generated using a $4000_{\vec{x}} \times 4_k$ global node set with a $50_{\vec{x}} \times 2_k$ stencil. The displayed solution is interpolated to a new grid using $10,000$ Halton nodes. In addition, the ice line and analytical solution are displayed in solid gray for reference.

4.4.5 Case 5: Evolution in the coupled domains

In the fifth numerical test case, spatial and directional evolution of wave action is tested in the meshless wave model[8] using several different initial conditions and model parameters. To simplify discussions, $\theta = \mathrm{arctan2}(k, l)$ is used to describe the direction of wave action propagation. Moreover, spatial initial conditions are held fixed for all tests while the spectral initial conditions are varied to simplify testing of evolution in the coupled domains. Furthermore, directional spreading is added to the spectral initial conditions to mimic natural conditions (see Section 2.5), add stability, and improve convergence rates.

Here a Gaussian bell centered at $(\lambda_0 = 0, \mu_0 = 0)$ and width $d_5 = \pi/3$ is used for the spatial initial conditions (see Section 4.4.3). For simplicity, the same Gaussian bell curve is used for the directional distribution with $\Xi(\theta, \theta_0) = \xi(\theta, 0; \theta_0, 0)$ and in general, the initial dominant direction satisfies $\theta_0 \neq m\pi/2$ for $m \in \mathbb{Z}$. The primary model parameters used for all tests are itemized below.

- Global node size: $129{,}600_{\vec{\alpha}} = 3600_{\vec{x}} \times 36_{\vec{k}}$
- RBF-FD stencil size: $153_{\vec{\alpha}} = 17_{\vec{x}} \times 9_{\vec{k}}$
- Shape parameter: $\epsilon = 3$
- Hyperviscosity type: Δ^4
- Hyperviscosity coefficient: $\gamma = -3.55 \times 10^{-13}$
- Directional spread: $d_{\mathrm{dir}} = \pi/3$
- Ice line: $d_{\mathrm{ice}} = \pi/6$ (i.e., $\pm 75^\circ$)
- Time step ratio: $a = 0.2$

[8] The meshless wave model refers to use of Eq. (4.14) in the sphere-ring geometry.

4.4.5.1 Coupled spatial-spectral evolution

In the absence of source terms, the transport of wave action in Eq. (3.63) will follow a great circle path if the bathymetry is constant. This path can be determined analytically for any given initial $\vec{x}_0$ and θ_0 and compared with the model. However the direction of the wave action is not constant and will follow a sinusoidal pattern as it traverses the great circle path. Here, the directional component of wave action is analyzed as it propagates. To remove the need for interpolation, values are examined only at the directional nodes (multiples of $\pi/18$ here).

In Figs. 4.12 through 4.15, snapshots of the wave action are displayed for select directions over the entire spatial domain. In each figure, the left column (front) and right column (back) are centered about $(1,0,0)$ and $(-1,0,0)$ respectively. In Fig. 4.12, the initial conditions are shown. The initial wave action is distributed in a Gaussian bell with a $\pi/3$ width and a dominant initial direction of $3\pi/18$. In Fig. 4.13, the wave action is shown after a 1/6 revolution. The new dominant direction is $2\pi/18$ and already, most of the wave action previously located at $4\pi/18$ is now at other locations. The same is evident in Fig. 4.14 as well with a new dominant direction of $-2\pi/18$. And finally in Fig. 4.15, the dominant direction is $-3\pi/18$ after a 1/2 revolution. Notice that $\mathcal{W}(\alpha, t_{1/2}) = \mathcal{W}(-\alpha, t_0)$.

4.4.5.2 Initial direction error comparison

As a followup, the spatial-spectral evolution error is analyzed for several different initial directions and presented in Fig. 4.16. Here the total directional ℓ_2 errors after 1/2 revolution are calculated at each spatial location and then normalized by the analytic solution (by the same method). This is a difficult task for FD models to calculate accurately and we can see that the meshless model does well with errors ranging from 2×10^{-4} to 2×10^{-2}. To test initial directions closer to $\pi/2$, either a smaller initial bell width is needed – otherwise part of the wave action will pass through the boundary region and will be attenuated – or the error along a particular track needs to be analyzed instead.

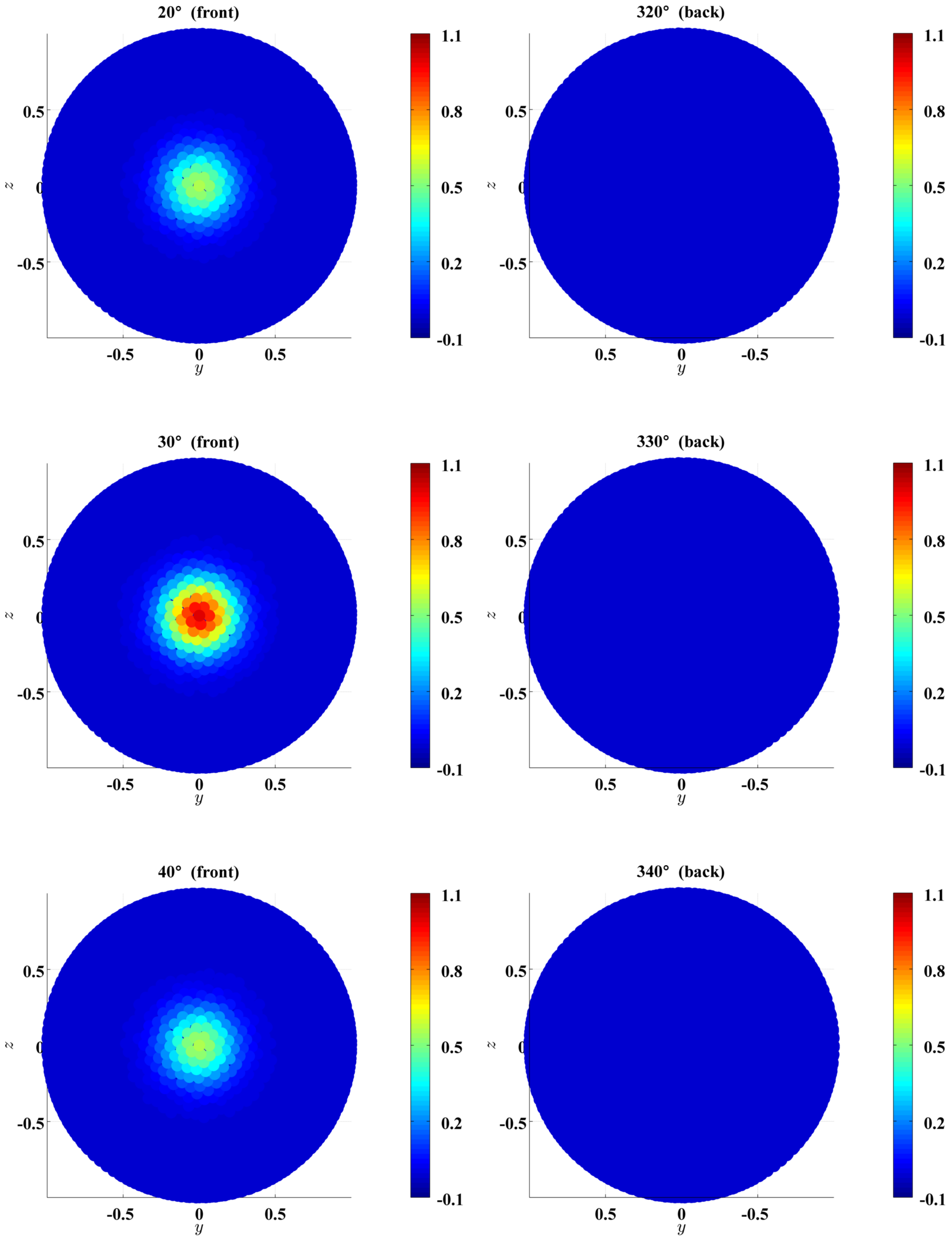


Figure 4.12: The wave action is displayed for select directions at **time** $= 0\Delta t$. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil and a time step ratio $a = 0.2$. The initial condition is a Gaussian bell with width $\pi/3$, direction $3\pi/18$ (30°), and directional spread $\pi/3$.

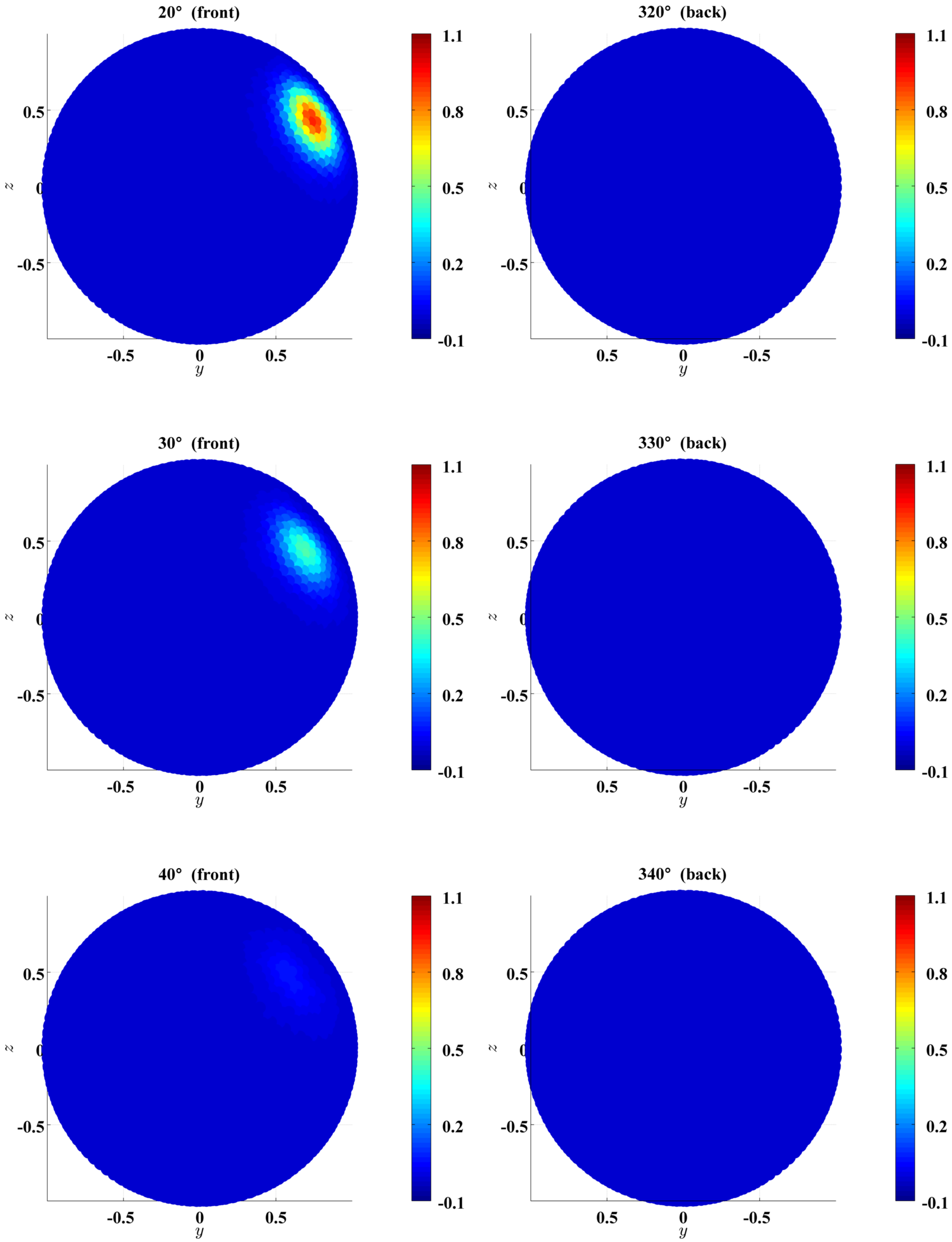


Figure 4.13: The wave action is displayed for select directions at **time** = $50\Delta t$. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil and a time step ratio $a = 0.2$. The initial condition is a Gaussian bell with width $\pi/3$, direction $3\pi/18$ (30°), and directional spread $\pi/3$.

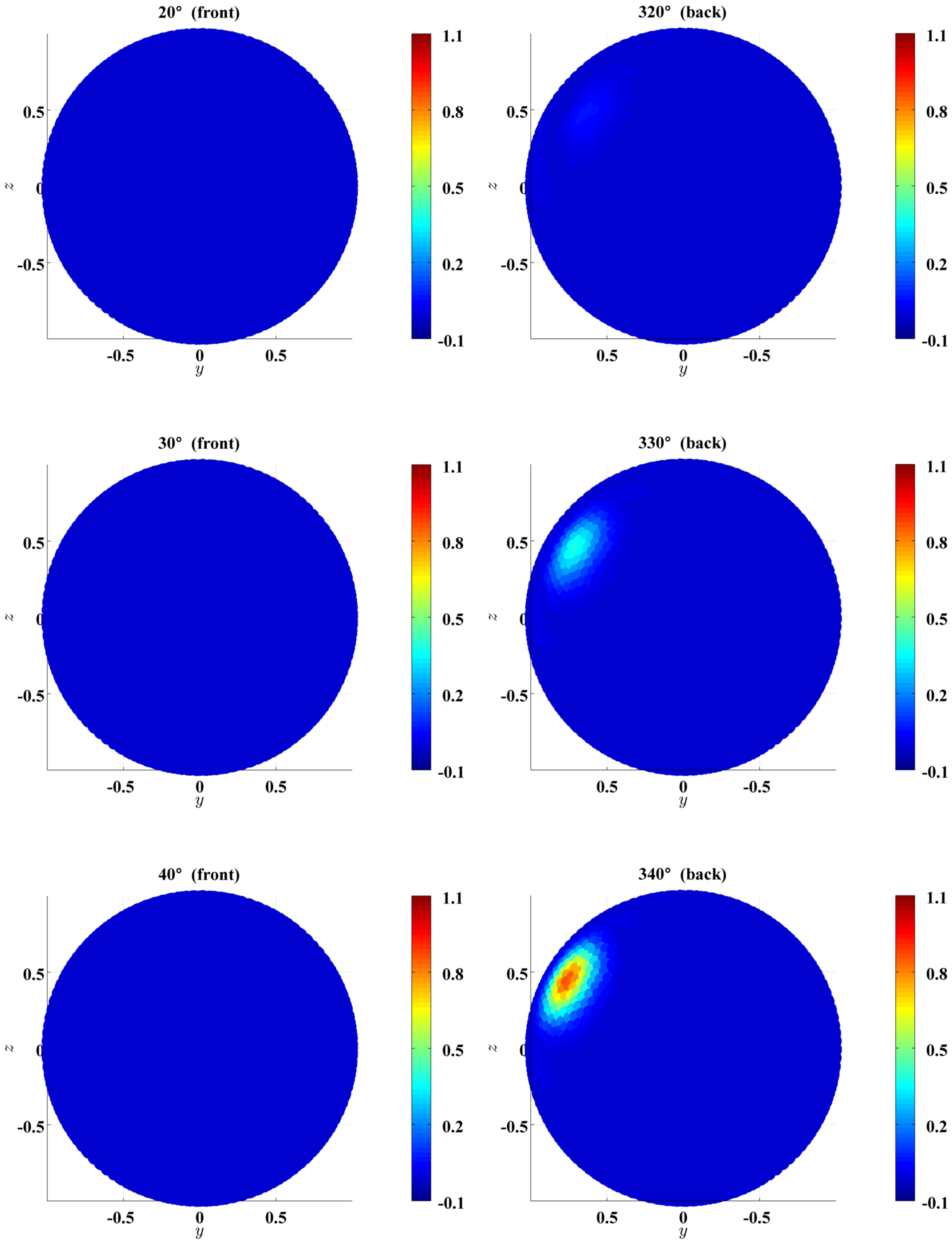


Figure 4.14: The wave action is displayed for select directions at **time** $= 100\Delta t$. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil and a time step ratio $a = 0.2$. The initial condition is a Gaussian bell with width $\pi/3$, direction $3\pi/18$ (30°), and directional spread $\pi/3$.

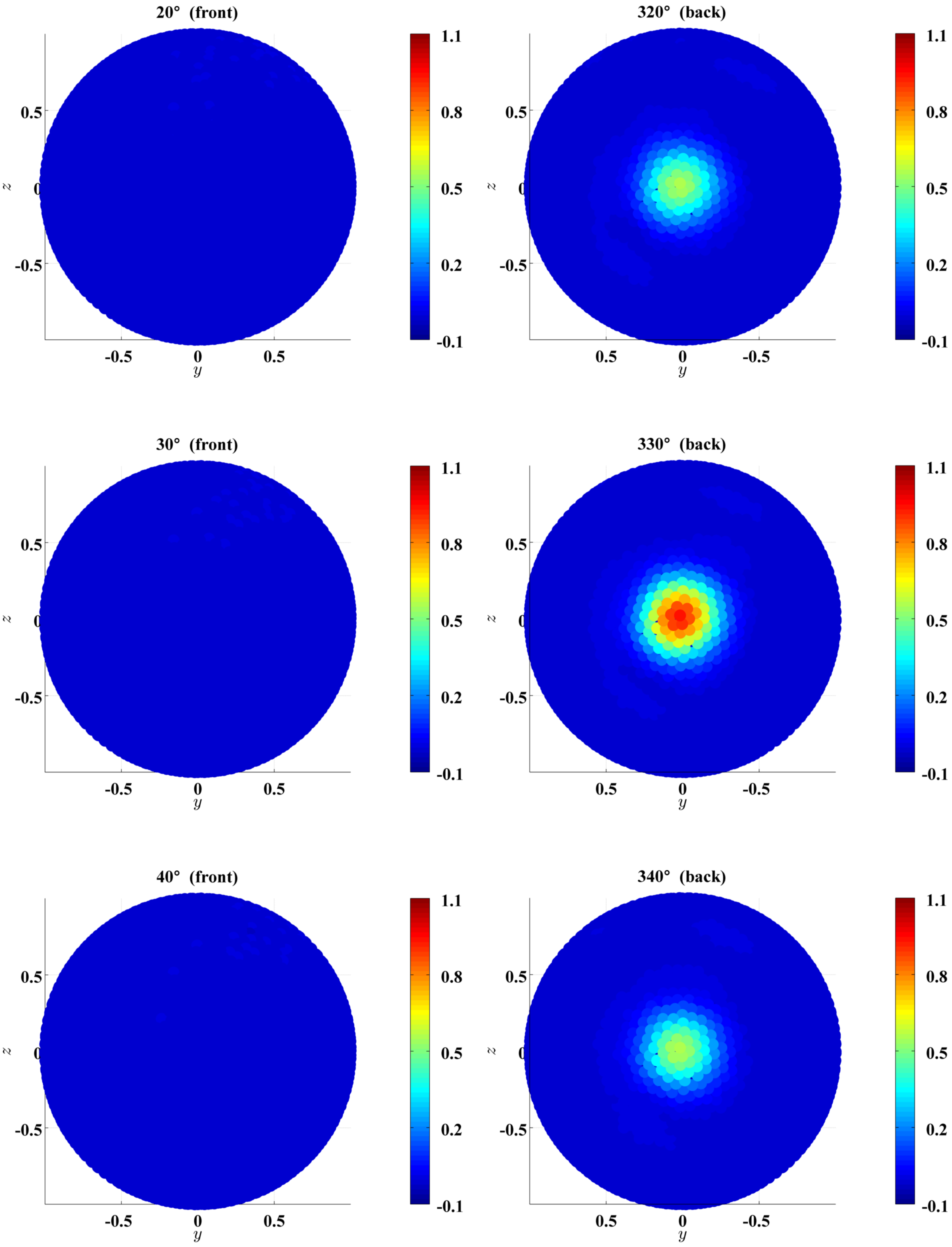


Figure 4.15: The wave action is displayed for select directions at **time** $= 150\Delta t$. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil and a time step ratio $a = 0.2$. The initial condition is a Gaussian bell with width $\pi/3$, direction $3\pi/18$ (30°), and directional spread $\pi/3$.

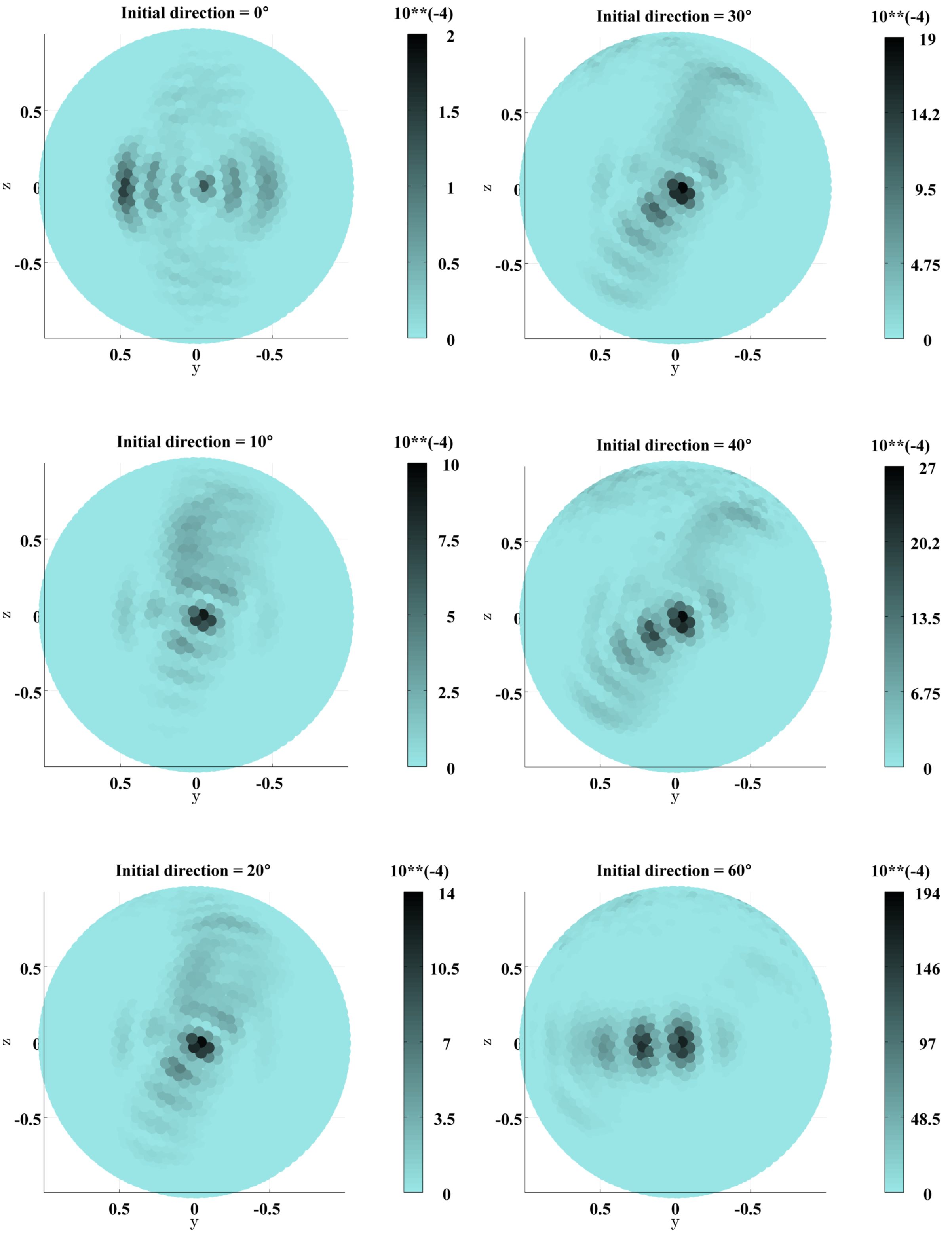


Figure 4.16: The total directional relative ℓ_2 errors after $1/2$ revolution are displayed for select **initial directions**. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil and a time step ratio $a = 0.2$. The initial condition is a Gaussian bell with width $\pi/3$ and directional spread $\pi/3$.

4.4.5.3 Model parameter error comparison

Several parameter and design decisions are based on both presented and omitted test cases (e.g., hyperviscosity). A brief comparison is made here of several of those to explore their effects on the model. In Fig. 4.17, the total directional relative ℓ_2 errors after 1/2 revolution are displayed for several different model settings and initial directions. The primary settings are in the first column.

In Figs. 4.17a and 4.17b, a comparison is made between different orders of hyperviscosity. In the model, hyperviscosity is particularly important since it helps stabilize propagation near the singular poles. As such, an initial direction of $6\pi/18$ is chosen here for comparison. While it is not significant, the higher Laplacian order does reduce error for this initial condition. In addition, overall damping is less noticeable for the filter to be effective.

For Figs. 4.17c–f, an initial direction of $\pi/6$ is used. In Figs. 4.17c and 4.17d, a comparison is made between different shape parameters. From previously, Cases 2 and 3 suggest $\varepsilon = 2$ and $\varepsilon = 3$ are both ideal for the spectral and spatial stencils respectively. However, the relative errors between the two values vary by a factor more than 3. More testing is still needed to determine if $\varepsilon = 3$ is also suitable for different initial conditions.

In the final Figs. 4.17e and 4.17f, a comparison is made between different time step ratios. In all previous test cases presented, a ratio of $a = 0.2$ is sufficient to minimize time stepping errors for larger stencils and this appears to hold for the meshless wave model. However, the relative error quickly grows for $a \leq 0.25$. This differs from previously ($a \lessapprox 0.7$) and needs to be explored further.

4.4.5.4 Global node and stencil comparison

In this model, the higher-ordered coupled stencils and node sets can be quite large and it is important to balance accuracy and size to avoid inverting and storing large matrices. Here, several different node set and stencil combinations are compared for this purpose. The total directional relative ℓ_2 errors after 1/2 revolution are displayed in Fig. 4.18 for $\theta_0 = 3\pi/18$. Many comparisons are possible here and a few have been selected to illustrate two main points.

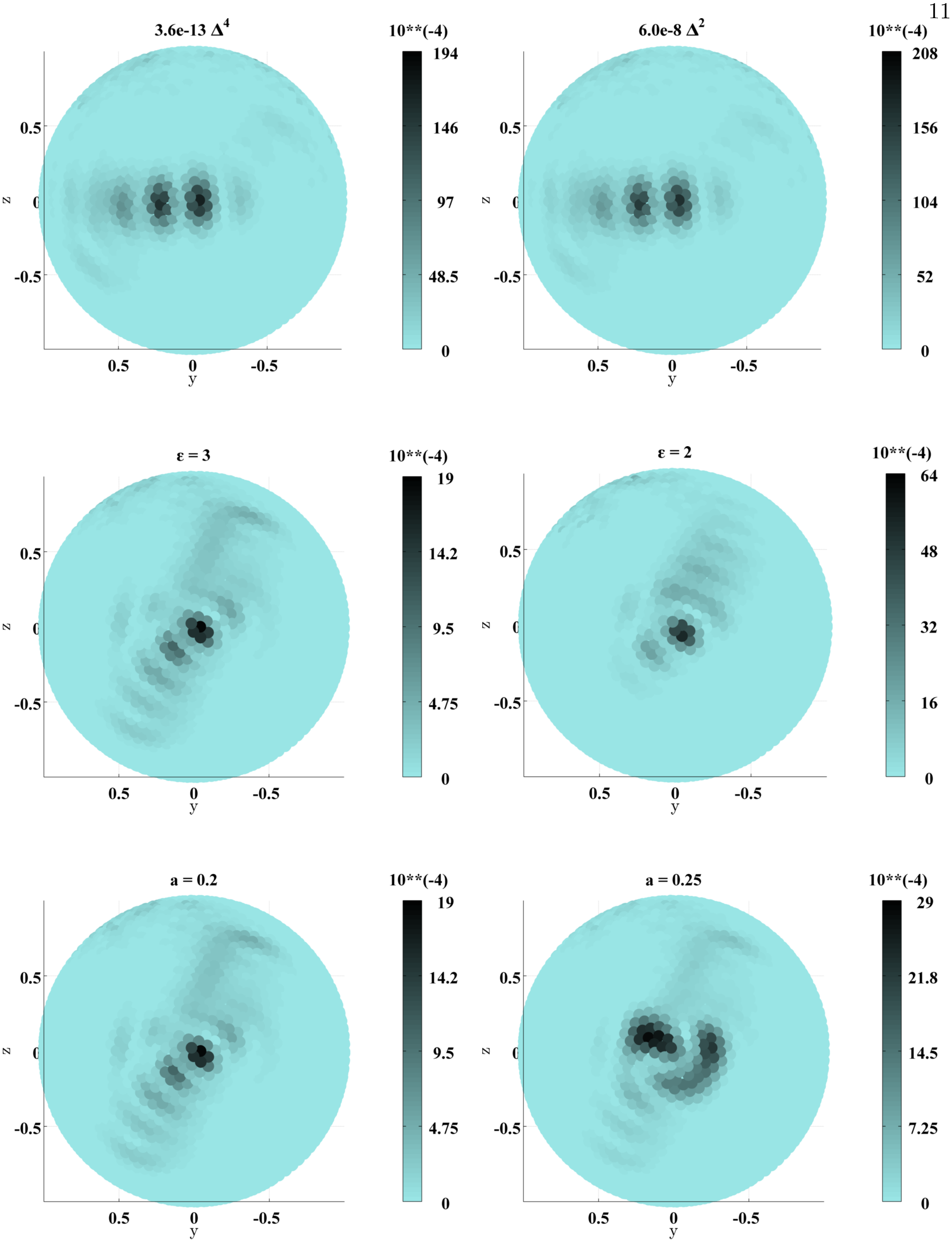


Figure 4.17: The total directional relative ℓ_2 errors after 1/2 revolution are displayed for select **model settings**. The default settings are in the first column. The model uses a $3600_{\vec{x}} \times 36_k$ global node set with a $17_{\vec{x}} \times 9_k$ stencil. The initial condition is a Gaussian bell with width $\pi/3$, and directional spread $\pi/3$. The initial direction is $6\pi/18$ in (a) and (b) and $3\pi/18$ in (c) through (f).

The first is that there are global node size stability and accuracy thresholds in the RBF-FD wave model. While not shown, a spatial resolution of approximately 7.2° or $2500_{\vec{x}}$ nodes is necessary for stability and will limit future spectral resolution in a sphere-cylinder geometry model. For accuracy, there appears to be a directional threshold of $N_{\vec{k}} = 36$ or a 10° resolution. This is evident in Figs. 4.18a and 4.18c or Figs. 4.18b and 4.17c, where the relative errors are more than 15 and 25 times greater respectively for a larger directional resolution. In addition, Figs. 4.18b and 4.18f show that these errors are not resolved by moving from an approximate 6° spatial resolution to a higher spatial 5.1° one (i.e. $3600_{\vec{x}} \times 24_{\vec{k}}$ to $4900_{\vec{x}} \times 24_{\vec{k}}$). On the contrary, Figs. 4.18b and 4.18e show that moving to a lower 7.2° spatial but higher 10° spectral resolution (i.e. $3600_{\vec{x}} \times 24_{\vec{k}}$ to $2500_{\vec{x}} \times 36_{\vec{k}}$) does close the error gap with differences only 1.5 times greater.

The second key point is that a relatively higher-order stencil is needed to accurately resolve directional versus spatial evolution. A comparison here with Figs. 4.17c, 4.18c, and 4.18d demonstrate that the combined stencils $17_{\vec{x}} \times 7_{\vec{k}}$ or $17_{\vec{x}} \times 9_{\vec{k}}$ have lower relative errors than $31_{\vec{x}} \times 5_{\vec{k}}$ by approximately 1/3 and 1/5 respectively. For similar initial conditions in Cases 2 (not shown) and 3, the convergence rates of the directional stencils $n_{\vec{k}} = 7$ and $n_{\vec{x}} = 9$ are approximately 8th and 10th order respectively while the spatial stencil $n_{\vec{x}} = 17$ is 3rd order. Results from these subfigures and other tests indicate that the $17_{\vec{x}} \times 7_{\vec{k}}$ or $17_{\vec{x}} \times 9_{\vec{k}}$ stencils are adequate for use in the sphere-ring RBF-FD wave model.

4.4.6 Case 6: Comparison with WAVEWATCH III

In the final numerical test case, a comparison with WAVEWATCH III is made to quantify the RBF-FD model's performance. For the RBF-FD model, a spatially coarser $2500_{\vec{x}} \times 36_{\vec{k}}$ global node set (90,000 unknowns) with a $17_{\vec{x}} \times 9_{\vec{k}}$ stencil is used with the same shape parameter, hyperviscosity, ice lines, and time-step ratio as in Case 5 (Section 4.4.5). Within WAVEWATCH III, two different resolutions are used for the comparison: the standard 1°×1.25° ($N_{\boldsymbol{x}}$=4320) global latitude-longitude grid and the coarsened 3.2°×4° ($N_{\boldsymbol{x}}$=44, 064) climate grid used in the coupled wave component. The number of spatial WAVEWATCH III grid cells are different here since comparisons are conducted on

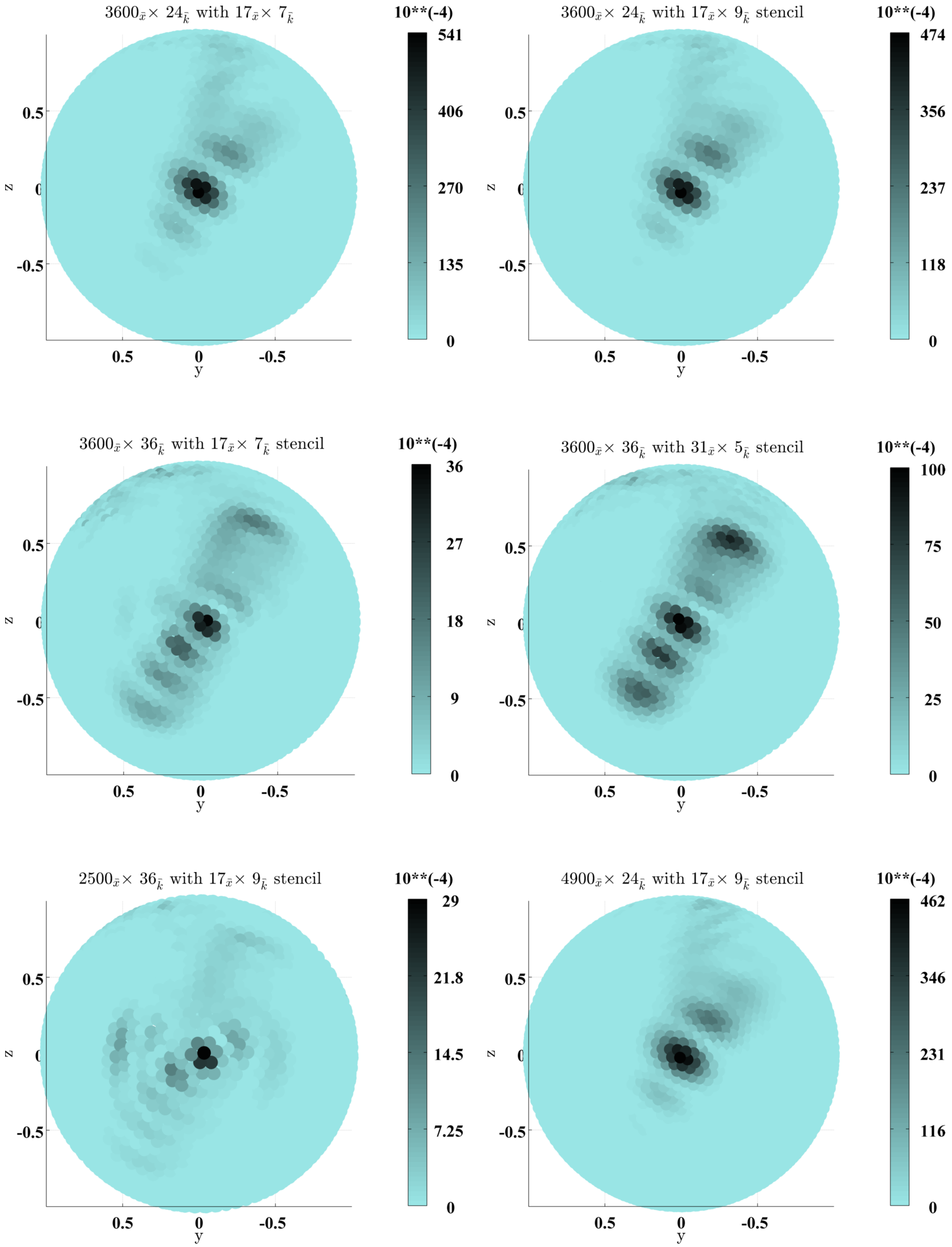


Figure 4.18: The total directional relative ℓ_2 errors after 1/2 revolution are displayed for select **global node** and **stencil sizes**. The model uses a time step ratio $a = 0.2$ and a Gaussian bell initial condition with width $\pi/3$, direction $3\pi/18$ (30°), and directional spread $\pi/3$.

an aqua planet with the same ice lines as the RBF-FD model. In addition, the number of directional bins in WAVEWATCH III have been increased from 24 to 36 to compare similar directional-frequency resolutions.

To simplify testing, optional WAVEWATCH III initial conditions are modified to resemble previous tests as closely as possible.[9] A Gaussian bell centered at $(\lambda_0 = 0, \mu_0 = 0)$ and width $d_6 = 0.31797\pi$ is used for the spatial initial conditions (see Section 4.4.3). To create a spectrum with a single frequency, a sharply peaked Gaussian about $f_p = 0.099$ Hz is used with a cosine power directional spread defined as

$$\mathcal{S}_{f\theta}(f,\theta;\xi,t_0) = \begin{cases} A(\xi,t_0)\,\delta(f-f_p)\cos^{20}[\theta-\theta_0]\,, & |\theta-\theta_0| \le \pi/2 \\ 0, & \text{otherwise.} \end{cases} \tag{4.17}$$

Here, a high 20 cosine power is used approximate a spectral directional bell of width $d = 64\pi/180$. In addition, the spectrum is normalized such that the maximum significant wave height, H_{m0}, at the center of the spatial Gaussian bell is 2.5 m. From Section 2.4.2, this implies the zeroth-moment satisfies

$$m_0 = \int_0^\infty \int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta;\xi{=}0,t_0)\,d\theta\,df = \left(\frac{H_{m0}}{4}\right)^2 = \frac{25}{64}. \tag{4.18}$$

And finally as in the previous subsection, an initial dominant direction $\theta_0 = \pi/3$ is chosen for the comparisons.

In Fig. 4.19, the relative ℓ_2 errors after 1/2 revolution for the new dominant direction $\theta = -\pi/3$ are computed and displayed for each spatial location in both models. The exact and numerical wave action solutions are displayed in the first and second columns respectively and are not interpolated for clarity. For both WAVEWATCH III resolutions (first and second rows), there is extensive diffusion and skewing after 1/2 revolution. In the RBF-FD model (third row), neither are present and there are little visible differences from the exact and numerical solutions aside from trace amounts of wave action west of the final packet.

[9] In WAVEWATCH III, the following settings are used in ww3_strt (initial conditions program): ITYPE=1 with 0.099 0.0005 240. 20. 0. 9. 0. 9. 2.5 (Tolman, 2009).

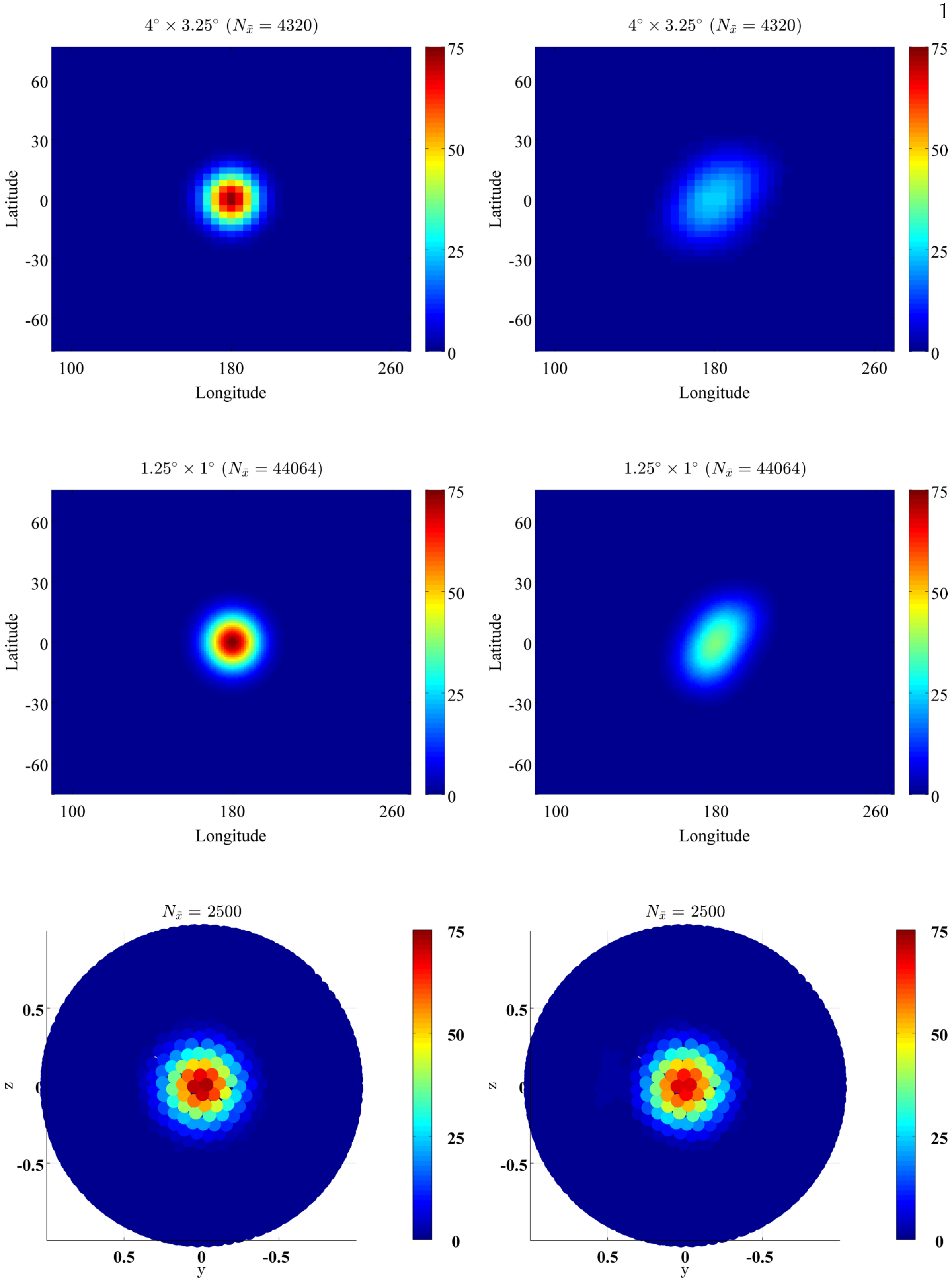


Figure 4.19: **Exact** (first column) and **numerical** (second column) wave action for dominant direction $\theta = -\pi/6$ after 1/2 revolution. Both **WAVEWATCH III** (first and second rows) and the **RBF-FD** model (third row) are initialized with a spatial Gaussian bell with width 0.31797π and a cosine-20-power directional spread ($\approx 64\pi/180$). In addition, the initial wave action are scaled such that the maximum significant wave height is 2.5m. Spatial resolutions are indicated in subfigures.

In Fig. 4.20, the relative ℓ_2 errors after 1/2 revolution for each directional component are summed and displayed for each spatial location in both models. Since initial conditions are slightly different here, error for an RBF-FD model with a $3600_{\vec{x}} \times 36_{\vec{k}}$ global node set and $17_{\vec{x}} \times 9_{\vec{k}}$ stencil is also displayed for comparison with previous tests. In WAVEWATCH III (first row), the maximum relative errors for the $3.2^\circ \times 4^\circ$ ($N_{\boldsymbol{x}}$=4320) and $1^\circ \times 1.25^\circ$ ($N_{\boldsymbol{x}}$=44, 064) resolutions are 3.9726×10^{-1} and 2.1021×10^{-1} respectively. Likewise in the RBF-FD model, the maximum relative errors for $N_{\boldsymbol{x}}$=2500 and $N_{\boldsymbol{x}}$=3600 are 3.0778×10^{-3} and 1.8921×10^{-3} respectively. Here in this test case, the spatially coarser RBF-FD model is approximately 130 to 70 times more accurate than WAVEWATCH III. In addition, the number of unknowns is reduced by a factor more than 17 in the highest resolution WAVEWATCH III run. For $N_{\boldsymbol{x}}$=3600, the RBF-FD model is approximately 210 to 110 times more accurate and reduces the number of unknowns by more than a factor of 12 in the highest resolution run. Comparing results with previous tests, we see that errors are similar for these initial conditions.

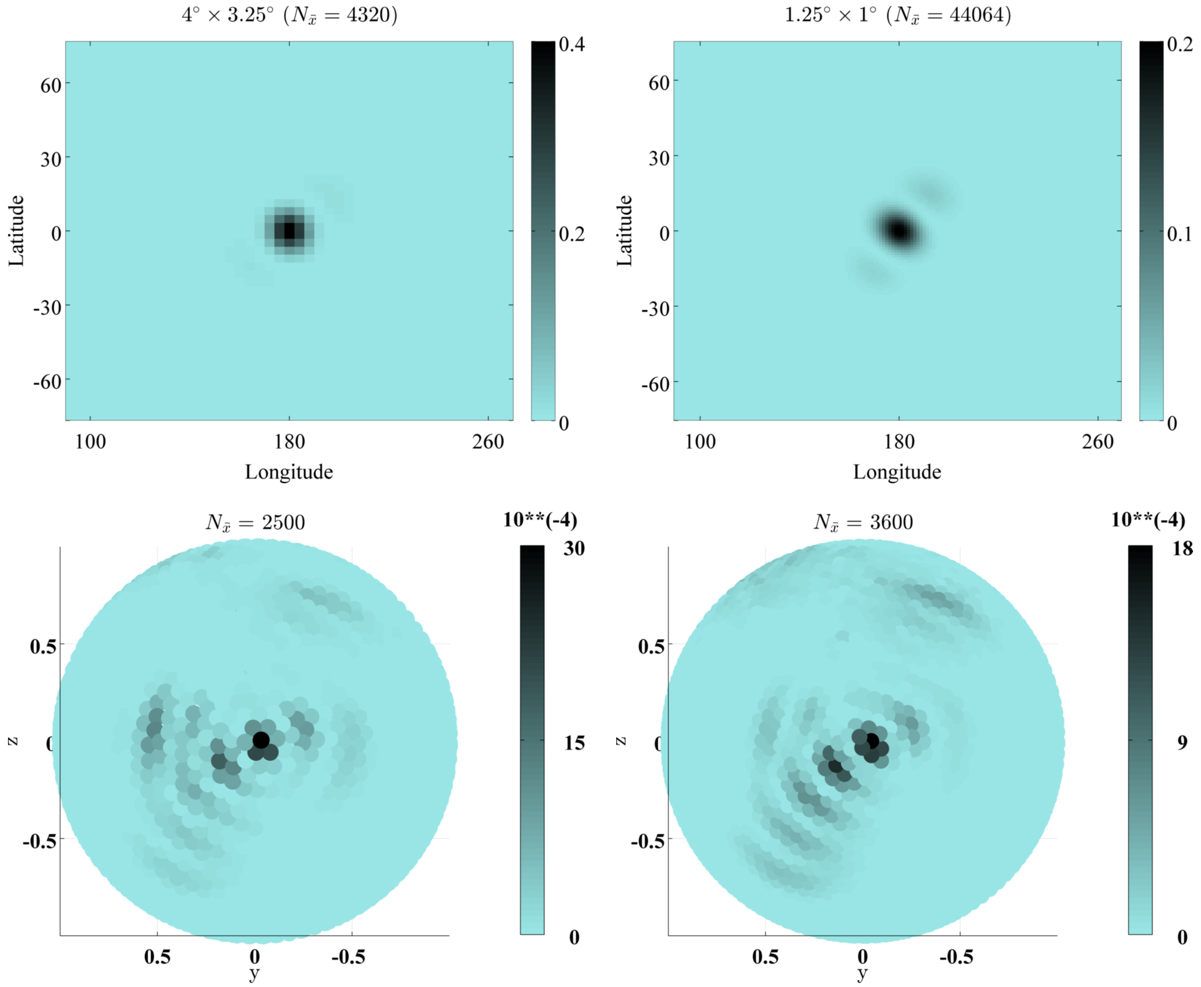


Figure 4.20: The total directional relative ℓ_2 errors after 1/2 revolution are displayed for **WAVEWATCH III** (first row) and the **RBF-FD** model (second row) using different resolutions. The models are initialized with a spatial Gaussian bell with width $64\pi/180$ and a cosine-20-power directional spread ($\approx 64\pi/180$). In addition, the initial wave action are scaled such that the maximum significant wave height is 2.5m. Spatial resolutions are indicated in subfigures.

Chapter 5

Summary and conclusions

This dissertation has been motivated by a desire to estimate the effects of LM in global climate models. Preliminary findings from Webb et al. (2013) demonstrates that inclusion of such mixing has the potential to reduce shallow mixed-layer biases found in many GCMs (in the high-latitude northern and southern oceans during their respective winters). Development of a new parameterization is closer to completion and much work has been done to ensure its analytical and numerical robustness. This dissertation is loosely organized around this work and includes an in-depth analysis of SD, optimization of a new global climate model wave component, and development of a meshless RBF-FD wave prototype.

In Chapter 2, a hierarchy of SD approximations are investigated and derived. These approximations can be classified by data type and in increasing order of accuracy, are based on spectral moments, 1D frequency spectra, and 2D directional-frequency spectra. The approximations derived remove systematic biases and are the most sophisticated and accurate in use.

In the first half of Chapter 2, lower-order approximations are derived using spectral moments and common wave properties (such as significant wave height and period) and are used in Webb and Fox-Kemper (2011) to calculate and compare global estimates of SD magnitudes. It is shown that the a_3-spectral-moment-SD approximation is equivalent to the unidirectional 1D$_h$-SD approximation at the surface and the a_2-spectral-moment-SD approximation roughly agrees to within 10% (of the unidirectional 1D$_h$-SD) away from coastlines. However, a comparison of global estimates shows that SD magnitudes vary significantly between the different (but reliable) data sources. It is hoped that the analysis presented in Webb and Fox-Kemper (2011) will guide future wave data collection and aid in the determination of a global SD climatology and variability.

In the second half of Chapter 2, the effects of directional spreading and multidirectional waves on SD are explored in the higher-order approximations. Since the unidirectional 1D$_h$-SD ignores these features, a new 1D$_h$-DHH-SD approximation is defined to incorporate the systematic effects of wave spreading. This approximation is based on observational studies of Donelan et al. (1985) and uses a Padé approximate to simplify the solution. Both 1D frequency approximations are compared with the 2D$_h$-SD approximation using formulated (empirically-derived), observational, and model generated wave spectra and it is shown that the improved 1D$_h$-DHH-SD reduces error. In addition, the effects of directional spreading and multidirectional waves are largely uncorrelated and affect both the magnitude and direction of Stokes drift in a nonlinear fashion that is sensitive with depth.

In the first half of Chapter 3, the operational forecast wave model WAVEWATCH III (version 3.14) is benchmarked and optimized for inclusion in a global climate model. The WAVE component, the coupled version of WAVEWATCH III in the NCAR CESM, will serve as the backbone of the new parameterization and will calculate SD and other variables needed to estimate the vertical mixing. Furthermore, the prognostic wave field will be useful for improving air-sea fluxes and sea ice formation in the GCM. Additional tests in Chapter 3 indicate that the WAVE component is coupled properly and development of the mixing scheme is ready to proceed.

In the second half of Chapter 3, the governing equation of a third-generation spectral wave model is explored in detail. Eq. (3.22), the wave action balance equation, is similar to the Boltzmann transport equation and is difficult to numerically model due to its high-dimensionality (i.e., uses a coupled spatial-spectral domain) and nonlinear source terms. Derivation of this equation is reviewed and an alternative form for 1D periodic propagation is derived using the WKB method. Different asymptotic limits of the latter yield a balance equation on a unit circle with linear shallow or deep-water group velocities. In addition, several variations of the general wave action balance equation are formulated to test the new meshless spectral wave prototype on different problem geometries.

In Chapter 4, construction of the meshless RBF-FD wave prototype in different geometries is reviewed and several numerical test cases are conducted to guide further development and measure performance. In Case 5.1 (Section 4.4.5.1), evolution in the coupled domains is illustrated for a spatial and spectral Gaussian bell initial condition with width $\pi/3$ and dominant direction $\pi/6$. In Case 5.2 (Section 4.4.5.2), the total directional relative ℓ_2 errors after a 1/2 revolution are calculated for similar initial conditions (different directions) and all errors are within 2%. This is a stringent test and the monochromatic RBF-FD prototype performs well with limited computation. To highlight its performance, a full kinematic comparison is made with WAVEWATCH III in Case 6 (Section 4.4.6). Using similar initial conditions, a coarser RBF-FD model is approximately 70 times more accurate while using less than 17 times the number of unknowns.

Further work is planned in the short term. The first is to continue testing of the prototype to determine robustness to changes in hyperviscosity, initial conditions, etc. The second is to replace the current attenuation filter approach with actual physical boundaries. This could potentially improve accuracy and would remove the need to include nodes within continents. These two combined will give an idea of the viability and benefit of building a parallel RBF-FD wave model for use with GCMs.

Moving further ahead, it will be necessary to determine the proper stencil selection in the spherical-cylinder geometry. It will be useful to resolve both spectral direction and frequency and the stencils will need to balance size versus accuracy. For frequency evolution to occur, this will require a non-constant bathymetry and use of source terms. As a result, a similar study to Cases 2 and 3 in the ring-line geometry will be useful. As an intermediate step, a model without frequency evolution might be useful since this process is commonly turned off in WAVEWATCH III simulations. This intermediate model would be straightforward to build since the frequency components are uncoupled and a monochromatic version could be run in parallel with limited communication.

Bibliography

Ablowitz, M.J., 2011. Nonlinear Dispersive Waves: Asymptotic Analysis and Solitons. Cambridge University Press, Cambridge, UK.

Ardhuin, F., Marié, L., Rascle, N., Forget, P., Roland, A., 2009. Observation and estimation of Lagrangian, Stokes, and Eulerian currents induced by wind and waves at the sea surface. Journal of Physical Oceanography 39, 2820–2838.

Axell, L.B., 2002. Wind-driven internal waves and Langmuir circulations in a numerical ocean model of the southern Baltic Sea. Journal of Geophysical Research: Oceans 107, 25–1–25–20.

Banner, M.L., 1990. Equilibrium spectra of wind-waves. Journal of Physical Oceanography 20, 966–984.

Batchelor, G.K., 1967. An Introduction to Fluid Dynamics. Cambridge University Press, New York, NY.

Belcher, S.E., Grant, A.L.M., Hanley, K.E., Fox-Kemper, B., Van Roekel, L.P., Sullivan, P.P., Large, W.G., Brown, A., Hines, A., Calvert, D., Rutgersson, A., Pettersson, H., Bidlot, J.R., Janssen, P.A.E.M., Polton, J.A., 2012. A global perspective on Langmuir turbulence in the ocean surface boundary layer. Geophysical Research Letters 39, 1–9.

Bender, C.M., Orszag, S.A., 1978. Advanced Mathematical Methods for Scientists and Engineers I: Asymptotic Methods and Perturbation Theory. McGraw-Hill, New York, NY.

Bochner, S., Tenenbaum, M., Pollard, H., 1959. Lectures on Fourier integrals: with an author's supplement on monotonic functions, Stieltjes integrals, and harmonic analysis. Princeton University Press, Princeton, NJ.

Bollig, E.F., Flyer, N., Erlebacher, G., 2012. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. Journal of Computational Physics 231, 7133–7151.

Bouws, E. (Ed.), 1998. Guide to Wave Analysis and Forecasting. Number 702 in WMO, World Meteorological Organization, Geneva. 2nd edition.

Bretherton, F.P., Garrett, C.J.R., 1968. Wavetrains in inhomogeneous moving media. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 302, 529–554.

Childress, S., 2009. An Introduction to Theoretical Fluid Mechanics. American Mathematical Society, Providence, RI.

Chini, G.P., 2008. Strongly nonlinear Langmuir circulation and Rayleigh–Bénard convection. Journal of Fluid Mechanics 614, 39–65.

Chini, G.P., Julien, K., Knobloch, E., 2009. An asymptotically reduced model of turbulent Langmuir circulation. Geophysical and Astrophysical Fluid Dynamics 103, 179–197.

Comiso, J., 1999. Bootstrap sea ice concentrations for NIMBUS-7 SMMR and DMSP SSM/I. Technical Report 2. National Snow and Ice Data Center. Boulder, CO. Updated 2008.

Craik, A.D.D., 2005. George Gabriel Stokes on water wave theory. Annu. Rev. Fluid Mech. 37, 23–42.

Craik, A.D.D., Leibovich, S., 1976. Rational model for Langmuir circulations. Journal of Fluid Mechanics 73, 401–426.

Currie, I.G., 2003. Fundamental Mechanics of Fluids. CRC Press, Boca Raton, FL. 3rd edition.

D'Asaro, E.A., 2001. Turbulent vertical kinetic energy in the ocean mixed layer. Journal of Physical Oceanography 31, 3530–3537.

DigitalGlobe, 2010. Gulf of mexico oil slick. QuickBird Satellite Image.

Donelan, M.A., Hamilton, J., Hui, W.H., 1985. Directional spectra of wind-generated waves. Philosophical Transactions of the Royal Society of London Series A-Mathematical Physical and Engineering Sciences 315, 509–562.

Ewans, K.C., 1998. Observations of the directional spectrum of fetch-limited waves. Journal of Physical Oceanography 28, 495–512.

Flyer, N., Lehto, E., Blaise, S., Wright, G.B., St-Cyr, A., 2012. A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere. Journal of Computational Physics 231, 4078–4095.

Flyer, N., Wright, G.B., 2007. Transport schemes on a sphere using radial basis functions. Journal of Computational Physics 226, 1059–1084.

Flyer, N., Wright, G.B., 2009. A radial basis function method for the shallow water equations on a sphere. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 465, 1949–1976.

Fornberg, B., 1998a. Classroom note: Calculation of weights in finite difference formulas. SIAM Review 40, 685–691.

Fornberg, B., 1998b. A Practical Guide to Pseudospectral Methods. Cambridge University Press, Cambridge, UK.

Fornberg, B., Flyer, N., Russell, J.M., 2010. Comparisons between pseudospectral and radial basis function derivative approximations. IMA journal of numerical analysis 30, 149–172.

Fornberg, B., Lehto, E., 2011. Stabilization of RBF-generated finite difference methods for convective PDEs. Journal of Computational Physics 230, 2270–2285.

Fornberg, B., Lehto, E., Powell, C., 2013. Stable calculation of Gaussian-based RBF-FD stencils. Computers & Mathematics with Applications 65, 627–637.

Fornberg, B., Merrill, D., 1997. Comparison of finite difference-and pseudospectral methods for convective flow over a sphere. Geophysical Research Letters 24, 3245–3248.

Fornberg, B., Piret, C., 2008. On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. Journal of Computational Physics 227, 2758–2780.

Fox-Kemper, B., Danabasoglu, G., Ferrari, R., Griffies, S.M., Hallberg, R.W., Holland, M.M., Maltrud, M.E., Peacock, S., Samuels, B.L., 2011. Parameterization of mixed layer eddies. III: Implementation and impact in global ocean climate simulations. Ocean Modelling 39, 61–78.

Gommenginger, C.P., Srokosz, M.A., Challenor, P.G., Cotton, P.D., 2003. Measuring ocean wave period with satellite altimeters: A simple empirical model. Geophysical Research Letters 30.

Grant, A.L.M., Belcher, S.E., 2009. Characteristics of Langmuir turbulence in the ocean mixed layer. Journal of Physical Oceanography 39, 1871–1887.

Groves, G.W., 1966. Geometric wave propagation through curved media. Journal of Geophysical Research 71, 5271–5274.

Groves, G.W., Melcer, J., 1961. On the propagation of ocean waves on a sphere. Geofisica Internacional 8, 77–93.

Hackett, B., Breivik, Ø., Wettre, C., 2006. Forecasting the drift of objects and substances in the ocean, in: Ocean weather forecasting. Springer, pp. 507–523.

Harcourt, R.R., 2012. A second-moment closure model of Langmuir turbulence. Journal of Physical Oceanography 43, 673–697.

Harcourt, R.R., D'Asaro, E.A., 2008. Large-eddy simulation of Langmuir turbulence in pure wind seas. Journal of Physical Oceanography 38, 1542–1562.

Hasselmann, K., 1971. On the mass and momentum transfer between short gravity waves and larger-scale motions. J. Fluid Mech 50, 189–205.

Hasselmann, K., Barnett, T., Bouws, E., Carlson, H., Cartwright, D.E., Enke, K., Ewing, J.A., Gienapp, H., Hasselmann, D.E., Kruseman, P., Meerburg, A., Muller, P., Olbers, D.J., Richter, K., Sell, W., Walden, H., 1973. Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP). Erganzung zur Deutschen Hydrographischen Zeitschrift, Reihe A (8) 12, 1–95.

Haza, A.C., Özgökmen, T.M., Griffa, A., Garraffo, Z.D., Piterbarg, L., 2012. Parameterization of particle transport at submesoscales in the Gulf Stream region using Lagrangian subgridscale models. Ocean Modelling 42, 31–49.

Hendershott, M.C., Chapman, D.C., Malanotte-Rizzoli, P., 1989. Wave Motions in the Ocean. Unpublished collection of lecture notes by M. Hendershott at Scripps Institution of Oceanography.

Holm, D.D., 1996. The ideal Craik-Leibovich equations. Physica D 98, 415–441.

Holthuijsen, L.H., 2007. Waves in Oceanic and Coastal Waters. Cambridge University Press, New York, NY.

Hong, S.M., Pham, A.T., Jungermann, C., 2011. Deterministic Solvers for the Boltzmann Transport Equation. Springer Wein New York, New York, NY.

Huang, N.E., 1971. Derivation of Stokes drift for a deep-water random gravity wave field, in: Deep Sea Research and Oceanographic Abstracts, Elsevier. pp. 255–259.

Iserles, A., 2009. A First Course in the Numerical Analysis of Differential Equations. Cambridge University Press, Cambridge, UK.

Jansons, K.M., Lythe, G., 1998. Stochastic Stokes drift. Physical review letters 81, 3136–3139.

Janssen, P.A., 2008. Progress in ocean wave forecasting. Journal of Computational Physics 227, 3572–3594.

Janssen, P.A.E.M., 2004. The Interaction of Ocean Waves and Wind. Cambridge University Press, Cambridge, UK.

Kantha, L.H., Clayson, C.A., 2004. On the effect of surface gravity waves on mixing in the oceanic mixed layer. Ocean Modelling 6, 101–124.

Kenyon, K.E., 1969. Stokes drift for random gravity waves. Journal of Geophysical Research 74, 6991–6994.

Kenyon, K.E., 1970. Stokes transport. Journal of Geophysical Research 75, 1133–1135.

Komen, G.J., Cavaleri, L., Donelan, M., Hasselmann, K., Hasselmann, S., Janssen, P.A.E.M., 1994. Dynamics and Modeling of Ocean Waves. Cambridge University Press, Cambridge, UK.

Kump, L.R., Kasting, J.F., Crane, R.G., 2004. The Earth System. Pearson Education, Inc., Upper Saddle River, NJ. 2nd edition.

Kundu, P.K., Cohen, I.M., 2008. Fluid Mechanics. Academic Press, Burlington, MA. 4th edition.

Large, W.G., McWilliams, J.C., Doney, S.C., 1994. Oceanic vertical mixing: A review and a model with a nonlocal boundary layer parameterization. Reviews of Geophysics 32, 363–403.

Large, W.G., Yeager, S.G., 2008. The global climatology of an interannually varying air-sea flux data set. Climate Dynamics 33, 341–364.

Li, J.G., 2012. Propagation of ocean surface waves on a spherical multiple-cell grid. Journal of Computational Physics 231, 8262–8277.

Longuet-Higgins, M.S., 1969. On the transport of mass by time-varying ocean currents. Deep Sea Research and Oceanographic Abstracts 16, 431–447.

Luke, Y.L., 1969. The Special Functions and Their Approximations. Academic Press, New York, NY.

Martin, W., Flandrin, P., 1985. Wigner-Ville spectral analysis of nonstationary processes. Acoustics, Speech and Signal Processing, IEEE Transactions on 33, 1461–1470.

Massel, S.R., 1996. Ocean Surface Waves: Their Physics and Prediction. World Scientific Publishing Co. Pte. Ltd., Singapore.

McWilliams, J.C., Fox-Kemper, B., 2013. Oceanic wave-balanced surface fronts and filaments. Journal of Fluid Mechanics. Submitted.

McWilliams, J.C., Restrepo, J.M., 1999. The wave-driven ocean circulation. Journal of Physical Oceanography 29, 2523–2540.

McWilliams, J.C., Restrepo, J.M., Lane, E.M., 2004. An asymptotic theory for the interaction of waves and currents in coastal waters. Journal of Fluid Mechanics 511, 135–178.

McWilliams, J.C., Sullivan, P.P., 2000. Vertical mixing by Langmuir circulations. Spill & Science Technology Bulletin 6, 225–237.

McWilliams, J.C., Sullivan, P.P., 2001. Surface-wave effects on winds and currents in marine boundary layers, in: Lumley, J.L. (Ed.), Fluid Mechanics and the Environment: Dynamical Approaches. Springer Berlin Heidelberg, pp. 201–224.

McWilliams, J.C., Sullivan, P.P., Moeng, C.H., 1997. Langmuir turbulence in the ocean. Journal of Fluid Mechanics 334, 1–30.

Mei, C.C., Stiassnie, M., Yue, D.K.P., 2005. Theory and Applications of Ocean Surface Waves, Part 1: Linear Aspects. World Scientific Publishing Co. Pte. Ltd., Singapore.

Mellor, G., 2011. Wave radiation stress. Ocean Dynamics 61, 563–568.

Mellor, G.L., Donelan, M.A., Oey, L.Y., 2008. A surface wave model for coupling with numerical ocean circulation models. Journal of Atmospheric and Oceanic Technology 25, 1785–1807.

Micchelli, C.A., 1986. Interpolation of scattered data: Distance matrix and conditionally positive definite functions. Constr. Approx. 2, 11–22.

Ochi, M.K., 1998. Ocean Waves: The Stochastic Approach. Cambridge University Press, Cambridge, UK.

O'Reilly, W.C., Herbers, T.H.C., Seymour, R.J., Guza, R.T., 1996. A comparison of directional buoy and fixed platform measurements of Pacific swell. Journal of Atmospheric and Oceanic Technology 13, 231–238.

Phillips, O.M., 1966. The Dynamics of the Upper Ocean. Cambridge University Press, Cambridge, UK.

Pierson, Jr., W.J., Moskowitz, L., 1964. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. Journal of Geophysical Research 69, 5181–5190.

Pinkus, A., Zafrany, S., 1997. Fourier Series and Integral Transforms. Cambridge University Press, Cambridge, UK.

Rascle, N., Ardhuin, F., Queffeulou, P., Croize-Fillon, D., 2008. A global wave parameter database for geophysical applications. part 1: Wave-current-turbulence interaction parameters for the open ocean based on traditional parameterizations. Ocean Modelling 25, 154–171.

Rayner, N.A., Brohan, P., Parker, D.E., Folland, C.K., Kennedy, J.J., Vanicek, M., Ansell, T.J., Tett, S.F.B., 2006. Improved analyses of changes and uncertainties in sea surface temperature measured in situ since the mid-nineteenth century: The HadSST2 dataset. Journal of Climate 19, 446–469.

Sallee, J.B., Shuckburgh, E., Bruneau, N., Meijers, A.J.S., Bracegirdle, T.J., Wang, Z., 2013. Assessment of Southern Ocean mixed layer depths in CMIP5 models: Historical bias and forcing response. Journal of Geophysical Research: Oceans 118, 1–18.

Segar, D.A., 2007. Introduction to Ocean Sciences. W. W. Norton & Company, Inc., New York, NY.

Shechter, G., 2004. k-D tree package: MathWorks MATLAB Central File Exchange. `http://http://www.mathworks.com/matlabcentral/fileexchange/4586-k-d-tree`.

Sloan, I.H., Womersley, R.S., 2004. Extremal systems of points and numerical integration on the sphere. Advances in Computational Mathematics 21, 107–125.

Smith, J.A., 2001. Observations and theories of Langmuir circulation: A story of mixing, in: Lumley, J. (Ed.), Fluid Mechanics and the Environment: Dynamical Approaches. Springer, pp. 295–314.

Smith, J.A., 2006. Observed variability of ocean wave Stokes drift, and the Eulerian response to passing groups. Journal of Physical Oceanography 36, 1381–1402.

Smyth, W.D., Skyllingstad, E.D., Crawford, G.B., Wijesekera, H., 2002. Nonlocal fluxes and Stokes drift effects in the K-profile parameterization. Ocean Dynamics 52, 104–115.

Squire, V., 2007. Of ocean waves and sea-ice revisited. Cold Regions Science and Technology 49, 110–133.

Stewart, R.H., 2008. Introduction to Physical Oceanography. `http://oceanworld.tamu.edu/resources/ocng_textbook/contents.html`.

Sullivan, P.P., McWilliams, J.C., 2010. Dynamics of winds and currents coupled to surface waves. Annual Review of Fluid Mechanics 42, 19–42.

Tolman, H.L., 2009. User manual and system documentation of WAVEWATCH-III version 3.14. Technical Report 286. NOAA / NWS / NCEP / MMAB.

Tolstov, G.P., 1976. Fourier Series. Dover Publications, Inc., New York, NY.

Tseng, R.S., D'Asaro, E.A., 2004. Measurements of turbulent vertical kinetic energy in the ocean mixed layer from Lagrangian floats. Journal of Physical Oceanography 34, 1984–1990.

Van Roekel, L.P., Fox-Kemper, B., Sullivan, P.P., Hamlington, P.E., Haney, S.R., 2012. The form and orientation of Langmuir cells for misaligned winds and waves. Journal of Geophysical Research: Oceans 117, 1–22.

WAMDI Group, 1988. The WAM model—A third generation ocean wave prediction model. Journal of Physical Oceanography 18, 1775–1810.

Wang, D., McWilliams, J.C., Large, W.G., 1998. Large-eddy simulation of the diurnal cycle of deep equatorial turbulence. Journal of Physical Oceanography 28, 129–148.

Webb, A., Fox-Kemper, B., 2009. Global model sensitivity to parameterizing Langmuir circulation. Poster session presented at: CIRES Scientific Rendezvous, Boulder, CO. http://cires.colorado.edu/science/groups/foxkemper/people/pdfs/WebbFoxKemper09cires.pdf.

Webb, A., Fox-Kemper, B., 2011. Wave spectral moments and Stokes drift estimation. Ocean Modelling 40, 273–288.

Webb, A., Fox-Kemper, B., Baldwin-Stevens, E., Danabasoglu, G., Hamlington, B., Large, W.G., Hemer, M.A., 2013. Global climate model sensitivity to estimated Langmuir mixing. Ocean Modelling. In preparation.

Webb, A., Fox-Kemper, B., Large, W.G., Peacock, S., 2010. Demonstrated sensitivity to Langmuir mixing in a global climate model (CCSM). Oral session presented at: AGU 2010 Ocean Sciences Meeting, Portland, OR. http://cires.colorado.edu/science/groups/foxkemper/pubs/pdfs/WebbFox-Kemper10.pdf.

Whitham, G.B., 1974. Linear and Nonlinear Waves. John Wiley & Sons, Inc., New York, NY.

Willebrand, J., 1975. Energy transport in a nonlinear and inhomogeneous random gravity wave field. J. Fluid Mech 70, 113–126.

Appendix A

Definitions and derivations

A.1 Craik-Leibovich equations

The Craik-Leibovich (CL) equations are a surface wave filtered version of the Navier-Stokes equations (Chini et al., 2009). When wind and waves are aligned (and in absence of density stratification and Coriolis effects), the non-dimensional CL equations can be written as

$$D_t \boldsymbol{u} = -\nabla p + \frac{1}{La_t^2}\left(\hat{\boldsymbol{u}}^s \times \boldsymbol{\omega}\right) + \frac{1}{R^*}\nabla^2 \boldsymbol{u}, \tag{A.1}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{A.2}$$

Here, $\boldsymbol{u}$ is the wave-filtered Eulerian velocity, $\hat{\boldsymbol{u}}^s$ is the normalized Stokes drift velocity (by the magnitude of its surface value), and R^* is the Reynolds number based on the surface friction velocity $\boldsymbol{u}^*$ (due to wind).

A.2 Spectral moments

It is common to summarize 1D wave spectra at a point by their moments. The moments are defined by (Bouws, 1998) as

$$m_n = \int_0^\infty f^n \mathcal{S}_f(f)\, df, \tag{A.3}$$

where the **frequency spectral density**,[1] $\mathcal{S}_f$, is normalized to capture the variance of the surface height displacement, η, for some time scale T such that[2]

$$\lim_{T\to\infty} \langle \eta(t)^2 \rangle_T = \int_0^\infty \mathcal{S}_f(f)\, df. \tag{A.4}$$

Similarly, multidirectional or two-dimensional wave spectra can be summarized as

$$\widehat{m_n} = \int_0^\infty \int_{-\pi}^{\pi} f^n \mathcal{S}_{f\theta}(f,\theta)\, d\theta df, \tag{A.5}$$

where the **directional-frequency spectral density**, $\mathcal{S}_{f\theta}$, is normalized as

$$\lim_{T,L\to\infty} \langle \eta(\boldsymbol{x}_h, t)^2 \rangle_{T,\boldsymbol{L}_h} = \int_0^\infty \int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta)\, d\theta df, \tag{A.6}$$

for some horizontal length scale $\boldsymbol{L}_h = (L, L)$.[3] By definition,

$$\int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta)\, d\theta \equiv \mathcal{S}_f(f). \tag{A.7}$$

In practice, spectral moments are usually calculated statistically using expected values for a particular frequency or deterministically as the limit of a finite sum over a limited area. Since wave amplitude decays exponentially with depth, the 1D and 2D wave moments are expected to decay in z as:

$$\begin{aligned}
\lim_{T,L\to\infty} \langle \eta_{\bar{z}}(\boldsymbol{x}_h, t)^2 \rangle_{T,\boldsymbol{L}_h} &= \lim_{T,L\to\infty} \frac{1}{TL^2} \int_{t-T/2}^{t+T/2} \int_{\boldsymbol{x}_h - \boldsymbol{L}_h/2}^{\boldsymbol{x}_h + \boldsymbol{L}_h/2} \eta_{\bar{z}}(\boldsymbol{x}', t')^2\, d\boldsymbol{x}' dt' && \text{(A.8)}\\
&= \int_0^\infty \int_{-\pi}^{\pi} \mathcal{S}_{f\theta}(f,\theta) e^{\frac{8\pi^2 f^2}{g} z}\, d\theta df && \text{(A.9)}\\
&= \int_0^\infty \mathcal{S}_f(f) e^{\frac{8\pi^2 f^2}{g} z} df. && \text{(A.10)}
\end{aligned}$$

The decay with depth depends on wavenumber k or real frequency f, here related by the dispersion relation for linear deep-water waves ($4\pi^2 f^2 = gk$), where g is the gravitational acceleration.

[1] f is ordinary (not angular) wave frequency.

[2] Angle brackets denote spatial or temporal averaging as indicated by the subscripts.

[3] The h subscript denotes horizontal components.

A.3 Mean Wave Direction

It is convenient to define a **mean wave direction**, θ_{w}, for discussions of SD direction. Following Tolman (2009), it is defined here as

$$\theta_w = \mathrm{atan2}[b, a]\,, \tag{A.11}$$

where

$$(a, b) = \int_0^\infty \int_{-\pi}^{\pi} (\cos\theta, \sin\theta)\, \mathcal{S}_{f,\theta}(f, \theta)\, d\theta\, df. \tag{A.12}$$

Notice that Eq. (A.12) is similar to the zeroth-moment but with a directional component.

A.4 The DHH-B directional-SD-component

Based on observations, Banner (1990) concluded the DHH spreading function was not constant for $f/f_p \geq 1.6$ and proposed the following modification (Ewans, 1998):

$$\beta_{\mathrm{B}}(r) = 10^{-0.4} \exp\left[0.8393 \ln[10.00]\, r^{-1.134}\right], \quad r \geq 1.6. \tag{A.13}$$

Following Section 2.5.3.2, a directional-SD-component was approximated (using a Padé approximate of order [2/2]) for the Banner-modified directional distribution (termed DHH-B here) as

$$\boldsymbol{H}_{\text{DFF-B}}(f; f_p, \bar{\theta}) = \left(\cos\bar{\theta}, \sin\bar{\theta}, 0\right) \tfrac{0.5f_p^2+0.22f_pf+0.19f^2}{f_p^2-0.81f_pf+0.79f^2}, \quad f/f_p \geq 1.6. \tag{A.14}$$

The new directional-SD-component approximation has a maximum relative error of 7×10^{-5}. However, it was not used for testing of the improved $1\mathrm{D}_h$-SD estimate since the ratio of cutoff frequency to peak frequency was often under 1.6 in the observational and model data. To test the accuracy of a **$1\mathrm{D}_h$-DHH-B-SD** approximation, higher cutoff frequencies are needed.

A.5 SD spectral tail calculations

It is necessary to estimate contributions to SD for high frequencies that are outside the model domain. Here, a spectral tail parameterization with cutoff frequency f_c is used to estimate the SD tail. For some positive p and f_c, let the spectral tail be governed by

$$\mathcal{S}_{f\theta}(f,\theta) = \left(\frac{f_c}{f}\right)^p \mathcal{S}_{f\theta}(f_c,\theta), \quad f \geq f_c. \tag{A.15}$$

Then the tail contribution to the 2D$_h$-SD is

$$\boldsymbol{u}^S_{\text{2D,tail}} = \frac{16\pi^3}{\mathrm{g}} f_c^p \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta) \left\{ \int_{f_c}^{\infty} f^{3-p} \, \exp\left[-\frac{8\pi^2\,|z|}{\mathrm{g}} f^2\right] df \right\} d\theta. \tag{A.16}$$

A.5.1 Subsurface SD tail

Note that for $\alpha, f_c > 0$ ($\alpha \in \mathbb{R}$),

$$I = \int_{f_c}^{\infty} f^{3-p} \, \exp\left[-\alpha f^2\right] df = \frac{\alpha^{\frac{p}{2}-2}}{2} \int_{\alpha f_c^2}^{\infty} t^{2-\frac{p}{2}} \, \exp\left[-t\right] dt = \frac{\alpha^{\frac{p}{2}-2}}{2} \Gamma\left[2 - \frac{p}{2}, \alpha f_c^2\right],$$

where $\Gamma\left[s, x\right]$ is the upper incomplete gamma function. Then for $\alpha = 8\pi^2\,|z|\,/\mathrm{g}$,

$$\boldsymbol{u}^S_{\text{2D,tail}}\Big|_{z\neq 0} = \frac{8\pi^3}{\mathrm{g}} \alpha^{\frac{p}{2}-2} \, f_c^p \, \Gamma\left[2 - \frac{p}{2}, \alpha f_c^2\right] \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta. \tag{A.17}$$

Three common values of p are 4, 4.5, and 5. These reduce to

$$\begin{aligned}
\boldsymbol{u}^S_{\text{2D,tail},p=4}\Big|_{z\neq 0} &= \frac{8\pi^3}{\mathrm{g}} f_c^4 \, \Gamma\left[0, \alpha f_c^2\right] \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta, && \text{(A.18)}\\
\boldsymbol{u}^S_{\text{2D,tail},p=4.5}\Big|_{z\neq 0} &= \frac{8\pi^3}{\mathrm{g}} \alpha^{\frac{1}{4}} f_c^{\frac{9}{2}} \, \Gamma\left[-\frac{1}{4}, \alpha f_c^2\right] \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta, && \text{(A.19)}\\
\boldsymbol{u}^S_{\text{2D,tail},p=5}\Big|_{z\neq 0} &= \frac{8\pi^3}{\mathrm{g}} \alpha^{\frac{1}{2}} f_c^5 \, \Gamma\left[-\frac{1}{2}, \alpha f_c^2\right] \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta \\
&= \frac{16\pi^3}{\mathrm{g}} f_c^4 \left[\exp\left[-\alpha f_c^2\right] - \sqrt{\pi\alpha f_c^2}\left(1 - \operatorname{erf}\left[\sqrt{\alpha f_c^2}\right]\right)\right] \\
&\quad \times \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta, && \text{(A.20)}
\end{aligned}$$

where the error function ($\operatorname{erf} : \mathbb{R}_+^* \to [0,1]$) is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left[-t^2\right] dt. \tag{A.21}$$

A.5.2 Surface SD tail

If $z = 0$, then the SD tail is restricted to $p > 4$ and simplifies to

$$\boldsymbol{u}^S_{\text{2D,tail},p>4}\Big|_{z=0} = \frac{16\pi^3 f_c^4}{\mathrm{g}\,(p-4)} \int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta. \tag{A.22}$$

A.5.3 1D spectra simplification

For 1D frequency spectra, $\mathcal{S}_{f\theta}(f_c,\theta) = D(f_c,\theta)\mathcal{S}(f_c)$ and the integrand in the tail simplifies to

$$\int_{-\pi}^{\pi} (\cos\theta, \sin\theta, 0)\, \mathcal{S}_{f\theta}(f_c,\theta)\, d\theta = \boldsymbol{H}(f_c)\, \mathcal{S}_f(f_c), \tag{A.23}$$

where $\boldsymbol{H}$ is the directional-SD-component based on the directional distribution D.

A.6 Del in polar coordinates

Here, various del operations for use in Section 3.2.3 are defined and derived for polar coordinates using an abbreviated cylindrical del notation. It is assumed all cylindrical z-components are null and these del operations are denoted ∇_p for clarity.

Let $r \in \mathbb{R}^+$ and $\lambda \in (-\pi, \pi]$. Then for any scalar and vector function, $f = f(r,\lambda)$ and $\boldsymbol{A} = A_r e_r + A_\lambda e_\lambda = (A_r, A_\lambda)$ respectively, the del operations are defined as

$$\nabla_p f = \left(\partial_r, \frac{1}{r}\,\partial_\lambda\right) f, \tag{A.24}$$

$$\begin{aligned}\nabla_p \cdot (A_r, A_\lambda) &= \frac{1}{r}\left[\partial_r\left(rA_r\right) + \partial_\lambda A_\lambda\right] \\ &= \frac{1}{r}A_r + \partial_r A_r + \frac{1}{r}\,\partial_\lambda A_\lambda,\end{aligned} \tag{A.25}$$

$$\begin{aligned}\nabla_p^2 f &= \nabla_p \cdot \nabla_p f \\ &= \frac{1}{r}\,\partial_r f + \partial_{rr} f + \frac{1}{r^2}\,\partial_{\lambda\lambda} f.\end{aligned} \tag{A.26}$$

In addition, for $\boldsymbol{u} = (u_r(r,\lambda,t), u_\lambda(r,\lambda,t))$, define the velocity potential $\Phi = \Phi(r,\lambda,t)$ such that $\boldsymbol{u} = \nabla_p\Phi$. Then the material derivative on a scalar function in polar coordinates (denoted as D_p) can be rewritten as

$$\begin{aligned}D_p f &= \partial_t f + \boldsymbol{u}\cdot\nabla_p f \\ &= \partial_t f + \partial_r\Phi\ \partial_r f + \frac{1}{r^2}\,\partial_\lambda\Phi\ \partial_\lambda f.\end{aligned} \tag{A.27}$$

Recall that for a vector valued function (A_r, A_λ), the material derivative will have extra terms since $\partial_\lambda e_r = e_\lambda$ and $\partial_\lambda e_\lambda = -e_r$. Calculating each component of the vector separately, we find

$$\begin{aligned} D_p\left(A_r e_r\right) &= \partial_t\left(A_r e_r\right) + \boldsymbol{u}\cdot\nabla_p\left(A_r e_r\right) \\ &= e_r\left(\partial_t A_r + \partial_r\Phi\,\partial_r A_r + \frac{1}{r^2}\partial_\lambda\Phi\,\partial_\lambda A_r\right) + e_\lambda\left(\frac{1}{r^2}A_r\,\partial_\lambda\Phi\right), \end{aligned} \tag{A.28}$$

and

$$\begin{aligned} D_p\left(A_\lambda e_\lambda\right) &= \partial_t\left(A_\lambda e_\lambda\right) + \boldsymbol{u}\cdot\nabla_p\left(A_\lambda e_\lambda\right) \\ &= e_\lambda\left(\partial_t A_\lambda + \partial_r\Phi\,\partial_r A_\lambda + \frac{1}{r^2}\partial_\lambda\Phi\,\partial_\lambda A_\lambda\right) - e_r\left(\frac{1}{r^2}A_\lambda\,\partial_\lambda\Phi\right). \end{aligned} \tag{A.29}$$

Collecting, we find

$$D_p\begin{bmatrix} A_r \\ A_\lambda \end{bmatrix} = \begin{bmatrix} \partial_t A_r + \partial_r\Phi\,\partial_r A_r + \frac{1}{r^2}\left(\partial_\lambda\Phi\,\partial_\lambda A_r - A_\lambda\,\partial_\lambda\Phi\right) \\ \partial_t A_\lambda + \partial_r\Phi\,\partial_r A_\lambda + \frac{1}{r^2}\left(\partial_\lambda\Phi\,\partial_\lambda A_\lambda + A_r\,\partial_\lambda\Phi\right) \end{bmatrix}. \tag{A.30}$$

A.7 Miscellaneous formulations

A.7.1 Normalizing an interval for use with a unit circle

Consider any generic interval $(L_1, L_2]$ with finite length L. Define

$$\theta(l) = \frac{2\pi}{L}(l - L_1) - \pi. \tag{A.31}$$

Then θ maps any point l in the generic interval to $(-\pi, \pi]$.

A.7.2 Mapping 1D functions from an interval to a unit circle

Let $\boldsymbol{a}(\theta) = (\cos\theta, \sin\theta)$. Then for the domains $\Theta = \{\theta : \theta \in (-\pi, \pi]\}$ and $X = \{(x, y) : x, y \in [-1, 1] \text{ and } x^2 + y^2 = 1\}$, the mapping $a : \Theta \to X$ is bijective. For simplicity, consider a continuous periodic function $f : \Theta \to \mathbb{R}$. A new function $\tilde{f}$ can be defined as the mapping of f to the unit circle in Cartesian coordinates as

$$\tilde{f}\left(x, y\right) = f\left(\boldsymbol{a}^{-1}(x, y)\right) = f\left(\text{atan2}(y, x)\right). \tag{A.32}$$

A.7.3 Projection onto a sphere

Let $\boldsymbol{x} = (x_1, x_2, x_3)$ such that $\boldsymbol{x}$ lies on a sphere with radius R. Then for any vector $\boldsymbol{\nu} = (\nu_1, \nu_2, \nu_3)$, the tangential projection on the sphere at point $\boldsymbol{x}$ satisfies (Flyer and Wright, 2009)

$$P_{\boldsymbol{x}}\,\boldsymbol{\nu} = \boldsymbol{\nu} - \hat{\boldsymbol{n}}\,(\hat{\boldsymbol{n}} \cdot \boldsymbol{\nu}) = \boldsymbol{\nu} - \frac{1}{R^2}\boldsymbol{x}\,(\boldsymbol{x} \cdot \boldsymbol{\nu}) = \left(\boldsymbol{I}_3 - \frac{1}{R^2}\boldsymbol{x}\,\boldsymbol{x}^T\right)\boldsymbol{\nu}. \tag{A.33}$$

Likewise, the projected gradient follows similarly:

$$P_{\boldsymbol{x}}\nabla f = \nabla f - \hat{\boldsymbol{n}}\,(\hat{\boldsymbol{n}} \cdot \nabla f) = \left(\boldsymbol{I}_3 - \frac{1}{R^2}\boldsymbol{x}\,\boldsymbol{x}^T\right)\nabla f. \tag{A.34}$$

Note that if a vector is already tangential to the surface at $\boldsymbol{x}$, then this implies $\boldsymbol{x} \cdot \boldsymbol{\nu} = 0$ and $P_{\boldsymbol{x}}\boldsymbol{\nu} = \boldsymbol{\nu}$.

A.7.4 Propagation on a Sphere

Waves travel in a great circle on a sphere. If the path of the great circle (given by the intersection of the plane $\boldsymbol{a} \cdot \boldsymbol{x} = 0$) and the initial velocity vector, $\boldsymbol{v}_0$, for a point, $\boldsymbol{x}_0$, on the circle is known, then the velocity at any point on the circle is given by

$$\boldsymbol{v}_i = (\boldsymbol{x}_0 \times \boldsymbol{v}_0) \times \boldsymbol{x}_i. \tag{A.35}$$

A.7.5 Rotation on a Sphere

Let $\boldsymbol{r}$ and θ be the unit axis and counter-clockwise angle of rotation respectively. Then any point $\boldsymbol{x}_0$ can be decomposed as $\boldsymbol{x}_0 = \boldsymbol{p}_{\|} + (\boldsymbol{x}_0 \cdot \boldsymbol{r})\,\boldsymbol{r}$ where $\boldsymbol{p}_{\|}$ is the projection of $\boldsymbol{x}_0$ onto the great circle with radius $|\boldsymbol{r} \times \boldsymbol{x}_0 \times \boldsymbol{r}|$ and orientation $\boldsymbol{r}$. It then follows that $\boldsymbol{p}_{\perp} = \boldsymbol{r} \times \boldsymbol{x}_0$ is the perpendicular projection on the same great circle and a new point $\boldsymbol{x}_1$ rotated counter-clockwise θ on the circle can be defined as

$$\begin{aligned}
\boldsymbol{x}_1 &= \cos\theta\,\boldsymbol{p}_{\|} + \sin\theta\,\boldsymbol{p}_{\perp} + (\boldsymbol{r} \cdot \boldsymbol{x}_0)\,\boldsymbol{r} \\
&= \cos\theta\,(\boldsymbol{x}_0 - (\boldsymbol{r} \cdot \boldsymbol{x}_0)\,\boldsymbol{r}) + \sin\theta\,(\boldsymbol{r} \times \boldsymbol{x}_0) + (\boldsymbol{r} \cdot \boldsymbol{x}_0)\,\boldsymbol{r} \\
&= \cos\theta\,\boldsymbol{x}_0 + \sin\theta\,(\boldsymbol{r} \times \boldsymbol{x}_0) + (1 - \cos\theta)(\boldsymbol{r} \cdot \boldsymbol{x}_0)\,\boldsymbol{r} \\
&= \Big[\cos\theta\,\boldsymbol{I} + \sin\theta\,\boldsymbol{r}_{\times} + (1 - \cos\theta)\boldsymbol{r} \otimes \boldsymbol{r}\Big]\boldsymbol{x}_0.
\end{aligned} \tag{A.36}$$

Appendix B

Numerical wave modeling

B.1 NOAA WAVEWATCH III details

NOAA WAVEWATCH III (Tolman, 2009) is an operational third–generation wave model that calculates and uses 2D wave spectra to forecast the ocean wave state. Two versions of the model are used here: version 2.22 for spectral moment and SD analysis and version 3.14 for GCM coupling. In version 2.22, operational forecast settings were used to generate 1D wave spectra every six hours for the years 1994−2001 and 2D wave spectra every three hours for the year 2000. Some of the operational settings include 25 frequency and 24 directional bins (with an initial and cutoff frequency of 0.0418 and 0.411 respectively), f^{-5} tail, third order propagation scheme, and Tolman and Chalikov source terms[1]

All WAVEWATCH III simulations are forced with CORE2 (Large and Yeager, 2008) winds[2] with appropriate sea surface temperatures (Hadley SST: Rayner et al., 2006) and sea ice concentrations (Bootstrap Sea Ice Concentrations from Nimbus-7 SMMR and DMSP SSM/I, v2: Comiso, 1999) on a $1^\circ \times 1.25^\circ$ latitude-longitude grid of $(-78:78) \times (0:358.75)$ respectively. In addition, a 50% sea ice threshold for grid point inclusion is typically used for all temporal means (denoted $\langle\cdot\rangle_T$) and an area-weighted global mean (denoted $\langle\cdot\rangle_G$) is used to account for spreading meridians with latitude.

In version 3.14, a new source term package developed and tested by Rascle et al. (2008) is used

[1] For full details, see http://polar.ncep.noaa.gov/waves/implementations.shtml.

[2] Available from http://data1.gfdl.noaa.gov/nomads/forms/mom4/COREv2.html.

with the appropriate recommended settings. The modified coupled version uses a coarser 3.2° × 4° latitude-longitude grid of $(-75.2 : 75.2) \times (0 : 356)$ respectively and a 30 min time step delayed 1 step from the CESM coupler. Coupled inputs include surface interface temperatures, surface wind velocities, ocean currents, sea ice grid cell concentrations, and mixed layer depths. At the moment, output is limited but will eventually include SD, skin friction velocities, and a new nondimensional LM number (sensitive to wind-wave alignment) to be used for the LM parametrization.

B.2 Numerical SD calculations

Discretization of SD for model calculation is outlined here. All spectral tail and subsurface calculations use $p = 5$ and $\alpha = 8\pi^2 \left|z\right| / \mathrm{g}$ respectively. The Matlab functions used for calculating SD are printed in Appendix F.2.

B.2.1 2D$_h$-SD

The **subsurface** and **surface** 2D$_h$-SD numerical approximations are

$$\begin{aligned}\boldsymbol{u}^S_{\mathrm{2D}} \approx \frac{2\pi\alpha}{|z|} \sum_{j=1}^{N_\theta} \Delta\theta_j \left(\cos\theta_j, \sin\theta_j, 0\right) &\left\{ \left[\sum_{i=1}^{N_f} \Delta f_i \, f_i^3 \exp\left[-\alpha f_i^2\right] S_{f\theta}(f_i, \theta_j) \right] \right. \\ &\left. + f_c^4 \left[\exp\left[-\alpha f_c^2\right] - \sqrt{\pi\alpha f_c^2} \left(1 - \operatorname{erf}\left[\sqrt{\alpha f_c^2}\right]\right)\right] \mathcal{S}_{f\theta}(f_c, \theta_j) \right\}, \end{aligned} \tag{B.1}$$

$$\boldsymbol{u}^S_{\mathrm{2D}}\big|_{z=0} \approx \frac{16\pi^3}{\mathrm{g}} \sum_{j=1}^{N_\theta} \Delta\theta_j \left(\cos\theta_j, \sin\theta_j, 0\right) \left\{ \left[\sum_{i=1}^{N_f} \Delta f_i \, f_i^3 S_{f\theta}(f_i, \theta_j) \right] + f_c^4 \mathcal{S}_{f\theta}(f_c, \theta_j) \right\}. \tag{B.2}$$

B.2.2 Directional 1D$_h$-SD

Recall that for a generic $\|\boldsymbol{H}\|_{\ell^2} \neq 1$, the directional 1D$_h$-SD is given by

$$\boldsymbol{u}^S_{\text{1D-dir}} = \frac{16\pi^3}{g} \int_0^\infty \boldsymbol{H}(f) f^3 S_f(f) \, \exp\left[\frac{8\pi^2 f^2}{g} z\right] df. \tag{B.3}$$

The resultant **subsurface** and **surface** directional 1D$_h$-SD numerical approximations are

$$\begin{aligned}\boldsymbol{u}^S_{\text{1D-dir}} \approx \frac{2\pi\alpha}{|z|} \Bigg\{ & \left[\sum_{i=1}^{N_f} \Delta f_i \, \boldsymbol{H}(f_i) \, f_i^3 \, \exp\left[-\alpha f_i^2\right] S_f(f_i) \right] \\ & + f_c^4 \left[\exp\left[-\alpha f_c^2\right] - \sqrt{\pi \alpha f_c^2} \left(1 - \operatorname{erf}\left[\sqrt{\alpha f_c^2}\right]\right)\right] \boldsymbol{H}(f_c) \, S_f(f_c) \Bigg\} , \end{aligned} \tag{B.4}$$

$$\boldsymbol{u}^S_{\text{1D-dir}}\big|_{z=0} \approx \frac{16\pi^3}{\mathrm{g}} \left\{ \left[\sum_{i=1}^{N_f} \Delta f_i \, \boldsymbol{H}(f_i) \, f_i^3 S_f(f_i) \right] + f_c^4 \boldsymbol{H}(f_c) \, S_f(f_c) \right\} . \tag{B.5}$$

B.2.3 Unidirectional 1D$_h$-SD

For unidirectional approximations, $\|\boldsymbol{H}\|_{\ell^2} = 1$ and the **subsurface** and **surface** unidirectional 1D$_h$-SD numerical approximations simplify to

$$\begin{aligned}\boldsymbol{u}^S_{\text{1D-uni}} \approx \frac{2\pi\alpha}{|z|} \Bigg\{ & \left[\sum_{i=1}^{N_f} \Delta f_i \, f_i^3 \, \exp\left[-\alpha f_i^2\right] S_f(f_i) \right] \\ & + f_c^4 \left[\exp\left[-\alpha f_c^2\right] - \sqrt{\pi \alpha f_c^2} \left(1 - \operatorname{erf}\left[\sqrt{\alpha f_c^2}\right]\right)\right] S_f(f_c) \Bigg\} , \end{aligned} \tag{B.6}$$

$$\boldsymbol{u}^S_{\text{1D-uni}}\big|_{z=0} \approx \frac{16\pi^3}{\mathrm{g}} \left\{ \left[\sum_{i=1}^{N_f} \Delta f_i \, f_i^3 S_f(f_i) \right] + f_c^4 S_f(f_c) \right\} . \tag{B.7}$$

Appendix C

Formal SD truncation error

C.1 Nonlinear assumption for convergence

For $f\colon \mathbb{R}^d \to \mathbb{R}$, assume the $d+1$ partial derivatives of f are continuous within a closed ball centered about $\boldsymbol{a}$ and radius $\|\boldsymbol{x}-\boldsymbol{a}\|$. It can be shown that the remainder to a series approximation of a multivariate function satisfies

$$\begin{aligned} f(\boldsymbol{x}) = f(\boldsymbol{a}) &+ \sum_{|\boldsymbol{\alpha}|=1}^{k} \frac{1}{\boldsymbol{\alpha}!} \left. (\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}} D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) \right|_{\boldsymbol{x}=\boldsymbol{a}} \\ &+ \sum_{|\boldsymbol{\alpha}|=k+1} \frac{k+1}{\boldsymbol{\alpha}!} (\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}} \int_0^1 (1-\tau)^k \left. D^{\boldsymbol{\alpha}} f(\boldsymbol{x}') \right|_{\boldsymbol{x}'=\boldsymbol{a}+\tau(\boldsymbol{x}-\boldsymbol{a})} d\tau, \end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is a multi-index notation such that

$$\begin{aligned} \boldsymbol{\alpha}! &= \alpha_1! \ldots \alpha_n!, \\ (\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}} &= (x_1-a_1)^{\alpha_1} \ldots (x_n-a_n)^{\alpha_n}, \\ D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) &= (\,\partial_{x_1})^{\alpha_1} \ldots (\,\partial_{x_n})^{\alpha_n} f(\boldsymbol{x}). \end{aligned}$$

The zeroth and first-order approximations yield

$$f(\boldsymbol{x}) = f(\boldsymbol{a}) + F_0(\boldsymbol{x};\boldsymbol{a}), \tag{C.1}$$

$$f(\boldsymbol{x}) = f(\boldsymbol{a}) + (\boldsymbol{x}-\boldsymbol{a}) \cdot \left. \nabla f(\boldsymbol{x}) \right|_{\boldsymbol{x}=\boldsymbol{a}} + F_1(\boldsymbol{x};\boldsymbol{a}), \tag{C.2}$$

with

$$F_0(\boldsymbol{x};\boldsymbol{a}) = (\boldsymbol{x}-\boldsymbol{a})\cdot\left(\int_0^1 \nabla f(\boldsymbol{x}'(\tau))\big|_{\boldsymbol{x}'(\tau)=\boldsymbol{a}+\tau(\boldsymbol{x}-\boldsymbol{a})}\,d\tau\right), \tag{C.3}$$

$$F_1(\boldsymbol{x};\boldsymbol{a}) = \sum_{|\boldsymbol{\alpha}|=2}\frac{2}{\boldsymbol{\alpha}!}\,(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}}\cdot\left(\int_0^1 (1-\tau)\,D^{\boldsymbol{\alpha}} f(\boldsymbol{x}'(\tau))\big|_{\boldsymbol{x}'(\tau)=\boldsymbol{a}+\tau(\boldsymbol{x}-\boldsymbol{a})}\,d\tau\right). \tag{C.4}$$

Assume the derivatives can be bounded as

$$\left\|(\,\partial_{x_1})^{\alpha_1}\ldots(\,\partial_{x_n})^{\alpha_n}\,f(\boldsymbol{x}')\right\|_{L^\infty} \le M^f_{\boldsymbol{\alpha}} = M^f_{(\alpha_1,\ldots,\alpha_n)}, \qquad x'\in B_{\|\boldsymbol{x}-\boldsymbol{a}\|}[\boldsymbol{a}].$$

Then in three dimensions with $\boldsymbol{a}=(a,b,c)$,

$$\begin{aligned}\|F_0(\boldsymbol{x};\boldsymbol{a})\|_{L^\infty} &\le \left\{|x-a|\,M^f_{(1,0,0)} + |y-b|\,M^f_{(0,1,0)} + |z-c|\,M^f_{(0,0,1)}\right\}\\ &\le \{|x-a|+|y-b|+|z-c|\}\max_{|\boldsymbol{\alpha}|=1} M^f_{\boldsymbol{\alpha}}\\ &= \|\boldsymbol{x}-\boldsymbol{a}\|_{\ell^1}\max_{|\boldsymbol{\alpha}|=1} M^f_{\boldsymbol{\alpha}},\end{aligned}$$

$$\begin{aligned}\|F_1(\boldsymbol{x};\boldsymbol{a})\|_{L^\infty} \le{}& \frac{1}{2}\left\{|x-a|^2\,M^f_{(2,0,0)} + |y-b|^2\,M^f_{(0,2,0)} + |z-c|^2\,M^f_{(0,0,2)}\right.\\ &\left.+\,2\,|x-a|\,|y-b|\,M^f_{(1,1,0)} + 2\,|x-a|\,|z-c|\,M^f_{(1,0,1)} + 2\,|y-b|\,|z-c|\,M^f_{(0,1,1)}\right\}\\ \le{}& \frac{1}{2}\{\,|x-a|^2+|y-b|^2+|z-c|^2+2\,|x-a|\,|y-b|+2\,|x-a|\,|z-c|\\ &+2\,|y-b|\,|z-c|\,\}\max_{|\boldsymbol{\alpha}|=2} M^f_{\boldsymbol{\alpha}}\\ ={}& \frac{1}{2}\,\|\boldsymbol{x}-\boldsymbol{a}\|^2_{\ell^1}\max_{|\boldsymbol{\alpha}|=2} M^f_{\boldsymbol{\alpha}}.\end{aligned}$$

Here, we would like to bound the truncation error used in the leading order approximation of SD. Let $\boldsymbol{R}_j=(R_{j,x},R_{j,y},R_{j,z})$. Then the components of each remainder vector are

$$\begin{aligned}R_{0,i}(\boldsymbol{x}_p(s');\boldsymbol{x}_p(t_0)) ={}& \big(\boldsymbol{x}_p(s')-\boldsymbol{x}_p(t_0)\big)\\ &\cdot\left(\int_0^1 \nabla u_i^E(\boldsymbol{x}'(\tau),s')\big|_{\boldsymbol{x}'(\tau)=\boldsymbol{x}_p(t_0)+\tau(\boldsymbol{x}_p(s')-\boldsymbol{x}_p(t_0))}\,d\tau\right),\end{aligned} \tag{C.5}$$

$$\begin{aligned}R_{1,i}(\boldsymbol{x}_p(t);\boldsymbol{x}_p(t_0)) ={}& \sum_{|\boldsymbol{\alpha}|=2}\frac{2}{\boldsymbol{\alpha}!}\,(\boldsymbol{x}_p(t)-\boldsymbol{x}_p(t_0))^{\boldsymbol{\alpha}}\\ &\cdot\left(\int_0^1 (1-\tau)\,D^{\boldsymbol{\alpha}} u_i^E(\boldsymbol{x}'(\tau),t)\big|_{\boldsymbol{x}'(\tau)=\boldsymbol{x}_p(t_0)+\tau(\boldsymbol{x}_p(t)-\boldsymbol{x}_p(t_0))}\,d\tau\right).\end{aligned} \tag{C.6}$$

For $\boldsymbol{\alpha} = (\alpha_x, \alpha_y, \alpha_z)$ and some neighborhood of s, assume again the derivatives can be bounded as

$$\left\|(\,\partial_x)^{\alpha_x}\,(\,\partial_y)^{\alpha_y}\,(\,\partial_z)^{\alpha_z}\,u_i^E(\boldsymbol{x}', s)\right\|_{L^\infty} \le M_{\boldsymbol{\alpha}}^{u_i^E} = M_{(\alpha_x,\alpha_y,\alpha_z)}^{u_i^E}, \quad x' \in B_{\|\boldsymbol{x}_p(s)-\boldsymbol{x}_p(t_0)\|}[\boldsymbol{x}_p(t_0)].$$

Then for s' and t in the neighborhood of s,

$$\left\|R_{0,i}(\boldsymbol{x}_p(s');\boldsymbol{x}_p(t_0)\right\|_\infty \le \left\|\boldsymbol{x}_p(s') - \boldsymbol{x}_p(t_0)\right\|_{\ell^1} \max_{|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_i^E}, \quad B_{\|\boldsymbol{x}_p(s')-\boldsymbol{x}_p(t_0)\|}[\boldsymbol{x}_p(t_0)], \tag{C.7}$$

$$\|R_{1,i}(\boldsymbol{x}_p(t);\boldsymbol{x}_p(t_0)\|_\infty \le \frac{1}{2}\,\|\boldsymbol{x}_p(t) - \boldsymbol{x}_p(t_0)\|_{\ell^1}^2 \max_{|\boldsymbol{\alpha}|=2} M_{\boldsymbol{\alpha}}^{u_i^E}, \quad B_{\|\boldsymbol{x}_p(t)-\boldsymbol{x}_p(t_0)\|}[\boldsymbol{x}_p(t_0)]. \tag{C.8}$$

Re-examining the first component of the velocity differences, we find that

$$\begin{aligned} &u_x^L(\boldsymbol{x}_p(t_0), t) - u_x^E(\boldsymbol{x}_p(t_0), t) \\ &\quad = \left(\int_{t_0}^t \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), s') + \boldsymbol{R}_0\big(\boldsymbol{x}_p(s');\boldsymbol{x}_p(t_0)\big)\,ds'\right) \cdot \nabla u_x^E(\boldsymbol{x}_p(t_0), t) + R_{1,x}(\boldsymbol{x}_p(t);\boldsymbol{x}_p(t_0))\,, \end{aligned}$$

and

$$\begin{aligned} &\left\|u_x^L(\boldsymbol{x}_p(t_0), t) - u_x^E(\boldsymbol{x}_p(t_0), t)\right\|_{L^\infty} \\ &\quad \le \left\|\left(\int_{t_0}^t \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), s')ds'\right) \cdot \nabla u_x^E(\boldsymbol{x}_p(t_0), t)\right\|_{L^\infty} \\ &\qquad + \left\|\left(\int_{t_0}^t \boldsymbol{R}_0\big(\boldsymbol{x}_p(s');\boldsymbol{x}_p(t_0)\big)\,ds'\right) \cdot \nabla u_x^E(\boldsymbol{x}_p(t_0), t)\right\|_{L^\infty} + \|R_{1,x}(\boldsymbol{x}_p(t);\boldsymbol{x}_p(t_0))\|_{L^\infty} \\ &\quad \le \left\|\left(\int_{t_0}^t \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), s')ds'\right) \cdot \nabla u_x^E(\boldsymbol{x}_p(t_0), t)\right\|_{L^\infty} \\ &\qquad + d\,|t-t_0| \max_{i,s'} \left\|R_{0,i}(\boldsymbol{x}_p(s');\boldsymbol{x}_p(t_0))\right\|_{L^\infty} \max_{|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_x^E} + \|R_{1,x}(\boldsymbol{x}_p(t);\boldsymbol{x}_p(t_0))\|_{L^\infty} \\ &\quad \le \left\|\left(\int_{t_0}^t \boldsymbol{u}^E(\boldsymbol{x}_p(t_0), s')ds'\right) \cdot \nabla u_x^E(\boldsymbol{x}_p(t_0), t)\right\|_{L^\infty} \\ &\qquad + \frac{d\,|t-t_0|}{2} \max_{s'} \left\|\boldsymbol{x}_p(s') - \boldsymbol{x}_p(t_0)\right\|_{\ell^1} \max_{i,|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_i^E} \max_{|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_x^E} \\ &\qquad + \frac{1}{2}\,\|\boldsymbol{x}_p(t) - \boldsymbol{x}_p(t_0)\|_{\ell^1}^2 \max_{|\boldsymbol{\alpha}|=2} M_{\boldsymbol{\alpha}}^{u_x^E}\,. \end{aligned} \tag{C.9}$$

If $L = \max\{X_h, Y_h\}$ and $\|\boldsymbol{x}_p(s) - \boldsymbol{x}_p(t_0)\|_{\ell^1} \le d\,L$, the truncation error for the first vector component of the leading SD estimate can be bounded in the L^∞ norm by

$$\begin{aligned} &\left\|u_x^S(\boldsymbol{x}, t; \boldsymbol{X}_h, T) - \left\langle u_x^L(\boldsymbol{x}, t) - u_x^E(\boldsymbol{x}, t)\right\rangle_{T,\boldsymbol{X}_h}\right\|_\infty \\ &\qquad \le \frac{d^2TL}{2} \max_{i,|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_i^E} \max_{|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_x^E} + \frac{d^2L^2}{2} \max_{|\boldsymbol{\alpha}|=2} M_{\boldsymbol{\alpha}}^{u_x^E}\,. \end{aligned} \tag{C.10}$$

For $\epsilon > 0$, $\max_{|\boldsymbol{\alpha}|=1} M_{\boldsymbol{\alpha}}^{u_i^E} \leq \sqrt{\frac{\epsilon}{d^2 T L}}$, and $\max_{|\boldsymbol{\alpha}|=2} M_{\boldsymbol{\alpha}}^{u_i^E} \leq \frac{\epsilon}{d^2 L^2}$, then

$$\left\| u_i^S(\boldsymbol{x}, t; \boldsymbol{X}_h, T) - \left\langle u_i^L(\boldsymbol{x}, t) - u_i^E(\boldsymbol{x}, t) \right\rangle_{T, \boldsymbol{X}_h} \right\|_\infty \leq \epsilon. \tag{C.11}$$

Appendix D

Analytic solutions to wave action balance equations

D.1 Problem 1

Let the wave action balance $\mathcal{W}$ on a plane (minus the origin) be governed by

$$\partial_t \mathcal{W} - \frac{c_g}{\sqrt{x^2+y^2}} (-y, x) \cdot \nabla_{\boldsymbol{x}} \mathcal{W} = 0, \tag{D.1}$$

for $c_g \in \mathbb{R}^*$ ($|c_g| \neq 0$) and some continuous initial condition $\mathcal{W}(x_0, y_0, t=0) = F(x_0, y_0)$ such that $\sqrt{x_0^2 + y_0^2} = R_{\boldsymbol{x}} > 0$. Applying the method of characteristics, define the characteristic curve as $(x(s), y(s), t(s), \mathcal{W}(s))$, where $\mathcal{W}(s) = \mathcal{W}(x(s), y(s), t(s))$. Then the following system of ODEs is satisfied:

$$\begin{aligned}
\frac{d\mathcal{W}}{ds} &= 0 \\
\frac{dt}{ds} &= 1 \\
\frac{dx}{ds} &= \frac{c_g\, y}{\sqrt{x^2+y^2}} \\
\frac{dy}{ds} &= \frac{-c_g\, x}{\sqrt{x^2+y^2}}.
\end{aligned}$$

For $x(0) = x_0$, $y(0) = y_0$, and $t(0) = 0$, this implies $t(s) = s$. We would like to find two functions, $a(x, y, t) = x_0$ and $b(x, y, t) = y_0$, so that our solution becomes

$$\mathcal{W}(x, y, t) = F(x_0, y_0) = F\left(a(x, y, t), b(x, y, t)\right).$$

To simplify the two coupled ODEs, note that (ignoring the case $x = y = 0$)

$$\frac{dy/ds}{dx/ds} = \frac{-x}{y} \quad \Rightarrow \quad y\, dy = -x\, dx \quad \Rightarrow \quad \sqrt{x^2+y^2} = R_{\boldsymbol{x}} > 0.$$

Our new set of ODEs becomes

$$\frac{dx}{ds} = \frac{c_g}{R_{\boldsymbol{x}}} y, \qquad \frac{dy}{ds} = \frac{-c_g}{R_{\boldsymbol{x}}} x,$$

which implies that

$$\frac{d^2x}{ds^2} = \frac{-c_g^2}{R_{\boldsymbol{x}}^2} x, \qquad \frac{d^2y}{ds^2} = \frac{-c_g^2}{R_{\boldsymbol{x}}^2} y.$$

The following solution satisfies the decoupled ODEs:

$$x(s) = A\cos(\tfrac{c_g}{R_{\boldsymbol{x}}} s + \delta), \qquad y(s) = -A\sin(\tfrac{c_g}{R_{\boldsymbol{x}}} s + \delta).$$

Solving for constants, we find

$$\begin{aligned} A > 0, \quad x(s)^2 + y(s)^2 = A^2 \quad &\Rightarrow \quad A = R_{\boldsymbol{x}}, \\ x_0 \neq 0 \quad \Rightarrow \quad \frac{y_0}{x_0} &= -\tan(\delta), \\ \delta &= -\arctan\left(\frac{y_0}{x_0}\right), \\ x_0 = 0 \quad \Rightarrow \quad \delta = \frac{\pi(2n+1)}{2}, \quad n &= 0, \pm 1, \pm 2, \dots . \end{aligned}$$

Relating the non-parametrized (x, y, t), we find

$$\begin{aligned} \frac{y_0}{x_0} &= \tan\Big[\arctan2(y,x) - \tfrac{c_g}{R_{\boldsymbol{x}}} t\Big], \\ x_0^2 + y_0^2 &= x_0^2 \left\{1 + \tan\Big[\arctan2(y,x) - \tfrac{c_g}{R_{\boldsymbol{x}}} t\Big]^2\right\} \\ &= x_0^2 \sec\Big[\arctan2(y,x) - \tfrac{c_g}{R_{\boldsymbol{x}}} t\Big]^2 = R_{\boldsymbol{x}}^2. \end{aligned}$$

Then

$$a(x,y,t) = x_0 = R_{\boldsymbol{x}} \cos[\arctan2(y,x) + \tfrac{c_g}{R_{\boldsymbol{x}}} t],$$

$$b(x,y,t) = y_0 = R_{\boldsymbol{x}} \sin[\arctan2(y,x) + \tfrac{c_g}{R_{\boldsymbol{x}}} t],$$

and

$$\mathcal{W}(x,y,t) = F(R_{\boldsymbol{x}} \cos[\arctan2(y,x) + \tfrac{c_g}{R_{\boldsymbol{x}}} t], R_{\boldsymbol{x}} \sin[\arctan2(y,x) + \tfrac{c_g}{R_{\boldsymbol{x}}} t]). \tag{D.2}$$

This is a solution for any initial (x_0, y_0) on the circle with radius $R_{\boldsymbol{x}}$ centered at the origin.

D.2 Problem 2

Let the wave action balance $\mathcal{W}$ on a plane (minus the origin) be governed by

$$\partial_t \mathcal{W} - \frac{c_g}{\sqrt{x^2+y^2}}\,(-y,x)\cdot P_{\boldsymbol{x}}\nabla_{\boldsymbol{x}}\mathcal{W} = 0, \tag{D.3}$$

for $c_g \in \mathbb{R}^*$ and some continuous initial condition $\mathcal{W}(x_0, y_0, t=0) = F(x_0, y_0)$ such that $\sqrt{x_0^2+y_0^2} = R_{\boldsymbol{x}} > 0$. Here, $P_{\boldsymbol{x}}$ is the projection onto the unit circle as defined in Eq. (A.34), which simplifies to

$$P_{\boldsymbol{x}}\nabla f = \frac{1}{x^2+y^2}\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix}\begin{bmatrix} \partial_x f \\ \partial_y f \end{bmatrix}.$$

The wave action balance equation can then be rewritten as

$$\begin{aligned}
\partial_t \mathcal{W} - \frac{c_g}{(x^2+y^2)^{3/2}}\begin{bmatrix} -y & x \end{bmatrix}\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix}\begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} &= \\
\partial_t \mathcal{W} - \frac{c_g}{(x^2+y^2)^{3/2}}\begin{bmatrix} -y(x^2+y^2) & x(x^2+y^2) \end{bmatrix}\begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} &= \\
\partial_t \mathcal{W} - \frac{c_g}{(x^2+y^2)^{1/2}}\begin{bmatrix} -y & x \end{bmatrix}\begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} &= 0.
\end{aligned}$$

This is equivalent to the PDE in Problem 1. This is also evident since $P(P\boldsymbol{u}) = P\boldsymbol{u}$ and $P\boldsymbol{u}\cdot\boldsymbol{v} = \boldsymbol{u}\cdot P\boldsymbol{v}$, giving

$$P\boldsymbol{u}\cdot P\boldsymbol{v} = P(P\boldsymbol{u})\cdot\boldsymbol{v} = P\boldsymbol{u}\cdot\boldsymbol{v}.$$

D.3 Problem 3

We would like to consider flow in a coupled spatial and spectral domain. Here, the transport velocity vector, $\boldsymbol{\nu} = \boldsymbol{\nu}(\boldsymbol{x}, \boldsymbol{k})$, is dependent on both $\boldsymbol{x}$ and $\boldsymbol{k}$ and is not necessarily tangent to the unit circle at point $\hat{\boldsymbol{x}}$. To possibly resolve this, consider the following additional constraint: $\hat{\boldsymbol{x}}\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = 0$ Let $\boldsymbol{\nu} \in \mathbb{R}^{2,*}$, then $\boldsymbol{\nu}$ can be decomposed into normal ($\boldsymbol{\nu}_{||}$) and tangent ($\boldsymbol{\nu}_{\perp}$) components for some $(\hat{x}, \hat{y})$ on a unit circle. Previously, we solved

$$\partial_t \mathcal{W} - \boldsymbol{\nu}_{\perp}(\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = 0.$$

Let the transport velocity be governed by the dispersion relation such that

$$\boldsymbol{\nu}(\boldsymbol{x},\boldsymbol{k}) = \nabla_{\boldsymbol{k}}\Omega(\boldsymbol{x},\boldsymbol{k}) = \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}}\begin{bmatrix} k \\ l \end{bmatrix}. \tag{D.4}$$

Then

$$\boldsymbol{\nu}_\perp = P_{\boldsymbol{x}}\nabla_{\boldsymbol{k}}\Omega(\boldsymbol{x},\boldsymbol{k}) = \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)}\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix}\begin{bmatrix} k \\ l \end{bmatrix} = \frac{\sqrt{\mathrm{g}}(lx-ky)}{2(k^2+l^2)^{3/4}(x^2+y^2)}\begin{bmatrix} -y \\ x \end{bmatrix},$$

$$\boldsymbol{\nu}_\parallel = \frac{1}{x^2+y^2}\,\boldsymbol{x}\,\boldsymbol{x}^T\nabla_{\boldsymbol{k}}\Omega(\boldsymbol{x},\boldsymbol{k}) = \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)}\begin{bmatrix} x^2 & xy \\ xy & y^2 \end{bmatrix}\begin{bmatrix} k \\ l \end{bmatrix} = \frac{\sqrt{\mathrm{g}}\,(kx+ly)}{2(k^2+l^2)^{3/4}(x^2+y^2)}\begin{bmatrix} x \\ y \end{bmatrix}.$$

Now consider the following variants.

D.3.1 Variant 1

Let $\gamma \in \mathbb{R}^*$. Consider

$$\partial_t\mathcal{W} - \boldsymbol{\nu}_\perp\cdot\nabla_{\boldsymbol{x}}\mathcal{W} - \gamma\,(\hat{x}\cdot\nabla_{\boldsymbol{x}}\mathcal{W}) = 0. \tag{D.5}$$

Our problem becomes

$$\begin{aligned}\partial_t\mathcal{W} - \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)}(lx-ky)\begin{bmatrix} -y & x \end{bmatrix}\begin{bmatrix} \partial_x\mathcal{W} \\ \partial_y\mathcal{W} \end{bmatrix} - \frac{\gamma}{(x^2+y^2)^{1/2}}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \partial_x\mathcal{W} \\ \partial_y\mathcal{W} \end{bmatrix} \\ = \partial_t\mathcal{W} - \begin{bmatrix} \partial_x\mathcal{W} & \partial_y\mathcal{W} \end{bmatrix}\begin{bmatrix} \frac{-\sqrt{\mathrm{g}}y(lx-ky)}{2(k^2+l^2)^{3/4}(x^2+y^2)} + \frac{\gamma x}{(x^2+y^2)^{1/2}} \\ \frac{\sqrt{\mathrm{g}}x(lx-ky)}{2(k^2+l^2)^{3/4}(x^2+y^2)} + \frac{\gamma y}{(x^2+y^2)^{1/2}} \end{bmatrix} \\ = 0. \end{aligned} \tag{D.6}$$

D.3.2 Variant 2

Let $\gamma = 0$; then

$$\partial_t\mathcal{W} - \boldsymbol{\nu}_\perp\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = \partial_t\mathcal{W} - P_{\boldsymbol{x}}\,\boldsymbol{\nu}\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = \partial_t\mathcal{W} - \boldsymbol{\nu}\cdot P_{\boldsymbol{x}}\nabla_{\boldsymbol{x}}\mathcal{W} = 0.$$

As a side note, also notice that

$$\begin{aligned}\partial_t \mathcal{W} - \boldsymbol{\nu} \cdot P_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} \mathcal{W} =&\ \partial_t \mathcal{W} - \left(\boldsymbol{\nu}_\perp + \boldsymbol{\nu}_\parallel\right) \cdot \left(\nabla_{\boldsymbol{x}} \mathcal{W} - \hat{x}\left(\hat{x} \cdot \nabla_{\boldsymbol{x}} \mathcal{W}\right)\right) \\ =&\ \partial_t \mathcal{W} - \boldsymbol{\nu} \cdot \nabla_{\boldsymbol{x}} \mathcal{W} - |\boldsymbol{\nu}_\parallel| \left(\hat{x} \cdot \nabla_{\boldsymbol{x}} \mathcal{W}\right).\end{aligned}$$

From here, a solution will be derived. Note that

$$\begin{aligned}\partial_t \mathcal{W} - \nabla_{\boldsymbol{k}} \Omega \cdot P_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} \mathcal{W} =& \\ \partial_t \mathcal{W} - \nabla_{\boldsymbol{k}} \Omega \cdot \left(\nabla_{\boldsymbol{x}} \mathcal{W} - \hat{\boldsymbol{x}}\left(\hat{\boldsymbol{x}} \cdot \nabla_{\boldsymbol{x}} \mathcal{W}\right)\right) =& \\ \partial_t \mathcal{W} - \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)} \begin{bmatrix} k & l \end{bmatrix} \begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix} \begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} =& \\ \partial_t \mathcal{W} - \frac{\sqrt{\mathrm{g}}(lx-ky)}{2(k^2+l^2)^{3/4}(x^2+y^2)} \begin{bmatrix} -y & x \end{bmatrix} \begin{bmatrix} \partial_x \mathcal{W} \\ \partial_y \mathcal{W} \end{bmatrix} =&\ 0. \end{aligned} \tag{D.7}$$

As in Problem 1, we find

$$\begin{aligned}\frac{d\mathcal{W}}{ds} &= \frac{dk}{ds} = \frac{dl}{ds} = 0, \\ \frac{dt}{ds} &= 1, \\ \frac{dx}{ds} &= \frac{\sqrt{\mathrm{g}}\,(lx-ky)y}{2(k^2+l^2)^{3/4}(x^2+y^2)}, \\ \frac{dy}{ds} &= \frac{-\sqrt{\mathrm{g}}\,(lx-ky)x}{2(k^2+l^2)^{3/4}(x^2+y^2)}.\end{aligned}$$

The following initial conditions are used: $(x(0), y(0), k(0), l(0), t(0)) = (x_0, y_0, k_0, l_0, 0)$, $\sqrt{x_0^2 + y_0^2} = R_{\boldsymbol{x}} > 0$, and $\sqrt{k_0^2 + l_0^2} = R_{\boldsymbol{k}} > 0$. Solving for t, k, and l, we find

$$t(s) = s, \qquad k(s) = k_0, \qquad l(s) = l_0.$$

We would like to find two functions, $a(x, y, k, l, t) = x_0$ and $b(x, y, k, l, t) = y_0$, such that our solution becomes

$$\mathcal{W}\left(x, y, k, l, t\right) = F(x_0, y_0, k_0, l_0) = F_{kl}(a(x, y, k, l, t), b(x, y, k, l, t)).$$

First note that the relation $l_0 x - k_0 y = 0$ yields a constant solution.

$$\begin{aligned} l_0 x - k_0 y = 0 \quad &\Rightarrow \quad \frac{dx}{ds} = \frac{dy}{ds} = 0 \quad \Rightarrow \quad (x,y) = (x_0, y_0) \neq (0,0), \\ &\Rightarrow \quad (x_0, y_0) = (k_0, l_0)\,\frac{R_{\boldsymbol{x}}}{R_{\boldsymbol{k}}}, \quad \Rightarrow \quad \boldsymbol{x} \parallel \boldsymbol{k}, \quad \mathcal{W}(\boldsymbol{x}, \boldsymbol{k}, t) = c_0 \in \mathbb{R}. \end{aligned}$$

To simplify the last two coupled ODEs, note that (ignoring $x = y = 0$)

$$\frac{dy/ds}{dx/ds} = \frac{-x}{y} \quad \Rightarrow \quad y\,dy = -x\,dx \quad \Rightarrow \quad x^2 + y^2 = R_{\boldsymbol{x}}^2.$$

Let $c_g = \frac{1}{2}\sqrt{\frac{\mathrm{g}}{R_{\boldsymbol{k}}}}$. Then our new set of ODEs becomes

$$\begin{aligned} \frac{dx}{ds} &= \frac{\sqrt{\mathrm{g}}\,(l_0 x - k_0 y) y}{2R_{\boldsymbol{x}}^2 (k_0^2 + l_0^2)^{3/4}} = \frac{\sqrt{\mathrm{g}}\,(l_0 x - k_0 y) y}{2R_{\boldsymbol{k}}^{3/2} R_{\boldsymbol{x}}^2} = \frac{c_g\,(l_0 x - k_0 y) y}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2}, \\ \frac{dy}{ds} &= \frac{-\sqrt{\mathrm{g}}\,(l_0 x - k_0 y) x}{2R_{\boldsymbol{x}}^2 (k_0^2 + l_0^2)^{3/4}} = \frac{-\sqrt{\mathrm{g}}\,(l_0 x - k_0 y) x}{2R_{\boldsymbol{k}}^{3/2} R_{\boldsymbol{x}}^2} = \frac{-c_g\,(l_0 x - k_0 y) x}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2}. \end{aligned}$$

In order to decouple, define two new variables, $u = k_0 x + l_0 y$ and $v = l_0 x - k_0 y$. Then

$$\begin{aligned} \frac{du}{ds} &= k_0 \frac{dx}{ds} + l_0 \frac{dy}{ds} = \frac{-c_g}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2}\,(l_0 x - k_0 y)^2 = \frac{-c_g}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2} v^2, \\ \frac{dv}{ds} &= l_0 \frac{dx}{ds} - k_0 \frac{dy}{ds} = \frac{c_g}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2}\,(l_0 x - k_0 y)\,(k_0 x + l_0 y) = \frac{c_g}{R_{\boldsymbol{k}} R_{\boldsymbol{x}}^2} uv. \end{aligned}$$

First note that

$$u^2 + v^2 = (k_0 x + l_0 y)^2 + (l_0 x - k_0 y)^2 = (k_0^2 + l_0^2) x^2 + (k_0^2 + l_0^2) y^2 = R_{\boldsymbol{k}}^2 R_{\boldsymbol{x}}^2.$$

Then as previously, we see that

$$\frac{du/ds}{dv/ds} = \frac{-v}{u} \quad \Rightarrow \quad u\,du = -v\,dv \quad \Rightarrow \quad u^2 + v^2 = R_{\boldsymbol{x}}^2 R_{\boldsymbol{k}}^2.$$

For simplification, define $R_{\boldsymbol{u}} = R_{\boldsymbol{k}} R_{\boldsymbol{x}}$ and let $\hat{c_g} = c_g R_{\boldsymbol{k}}$. Then the first new ODE can be rewritten and solved as

$$\begin{aligned} \frac{du}{ds} &= \frac{-\hat{c_g}}{R_{\boldsymbol{u}}^2}\left(R_{\boldsymbol{u}}^2 - u^2\right), \\ \frac{du}{R_{\boldsymbol{u}}^2 - u^2} &= \frac{-\hat{c_g}}{R_{\boldsymbol{u}}^2} ds, \\ \frac{du}{1 - \frac{u^2}{R_{\boldsymbol{u}}^2}} &= -\hat{c_g} ds, \end{aligned}$$

$$\begin{aligned}
R_{\boldsymbol{u}} \operatorname{arctanh}\left(\frac{u}{R_{\boldsymbol{u}}}\right) &= -\hat{c_g} s + c_1, \\
u(s) &= R_{\boldsymbol{u}} \tanh\left(\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + c_2\right).
\end{aligned}$$

It then follows

$$\begin{aligned}
\frac{dv}{ds} &= \frac{\hat{c_g}}{R_{\boldsymbol{u}}} \tanh\left(\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + c_2\right) v, \\
v \neq 0, \quad \frac{dv}{v} &= \frac{\hat{c_g}}{R_{\boldsymbol{u}}} \tanh\left(\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + c_2\right) ds, \\
v \neq 0, \quad \log(v) &= -\log\left(\cosh\left(\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + c_2\right)\right) + c_3, \\
v \neq 0, \quad v(s) &= c_4 \operatorname{sech}\left(\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + c_2\right), \quad c_4 > 0.
\end{aligned}$$

Solving we find:

$$\begin{aligned}
c_4 > 0, \quad u^2 + v^2 = R_{\boldsymbol{u}}^2 \quad &\Rightarrow \quad c_4 = R_{\boldsymbol{u}}, \\
\frac{u(0)}{v(0)} = \frac{u_0}{v_0} = \sinh(c_2), \quad &\Rightarrow \quad c_2 = \operatorname{arcsinh}\left(\frac{u_0}{v_0}\right),
\end{aligned}$$

$$\begin{aligned}
u_0 &= k_0 x_0 + l_0 y_0 \\
v_0 &= l_0 x_0 - k_0 y_0
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
x_0 &= \tfrac{1}{R_{\boldsymbol{k}}^2} (k_0 u_0 + l_0 v_0) \\
y_0 &= \tfrac{1}{R_{\boldsymbol{k}}^2} (l_0 u_0 - k_0 v_0)
\end{aligned},$$

$$\begin{aligned}
\frac{u(s)}{v(s)} &= \sinh\left[\frac{-\hat{c_g}}{R_{\boldsymbol{u}}} s + \operatorname{arcsinh}\left(\frac{u_0}{v_0}\right)\right], \\
\frac{u_0}{v_0} &= \sinh\left[\operatorname{arcsinh}\left(\frac{u(s)}{v(s)}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} s\right],
\end{aligned}$$

$$\begin{aligned}
u_0^2 + v_0^2 = \left\{\sinh\left[\operatorname{arcsinh}\left(\frac{u(s)}{v(s)}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} s\right]^2 + 1\right\} v_0^2 &= \cosh\left[\operatorname{arcsinh}\left(\frac{u(s)}{v(s)}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} s\right]^2 v_0^2 = R_{\boldsymbol{u}}^2, \\
\Rightarrow \quad v_0 = R_{\boldsymbol{u}} \operatorname{sech}\left[\operatorname{arcsinh}\left(\frac{u}{v}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right], \quad & u_0 = R_{\boldsymbol{u}} \tanh\left[\operatorname{arcsinh}\left(\frac{u}{v}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right],
\end{aligned}$$

$$\begin{aligned}
x_0 &= \frac{R_{\boldsymbol{u}}}{R_{\boldsymbol{k}}^2} \left\{k_0 \tanh\left[\operatorname{arcsinh}\left(\frac{k_0 x + l_0 y}{l_0 x - k_0 y}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right] + l_0 \operatorname{sech}\left[\operatorname{arcsinh}\left(\frac{k_0 x + l_0 y}{l_0 x - k_0 y}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right]\right\}, \\
y_0 &= \frac{R_{\boldsymbol{u}}}{R_{\boldsymbol{k}}^2} \left\{l_0 \tanh\left[\operatorname{arcsinh}\left(\frac{k_0 x + l_0 y}{l_0 x - k_0 y}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right] - k_0 \operatorname{sech}\left[\operatorname{arcsinh}\left(\frac{k_0 x + l_0 y}{l_0 x - k_0 y}\right) + \frac{\hat{c_g}}{R_{\boldsymbol{u}}} t\right]\right\},
\end{aligned}$$

$$a(x,y,k,l,t) = \frac{R_{\boldsymbol{x}}}{R_{\boldsymbol{k}}}\left\{k\tanh\left[\operatorname{arcsinh}\left(\frac{kx+ly}{lx-ky}\right)+\frac{c_g}{R_{\boldsymbol{x}}}t\right]+l\,\operatorname{sech}\left[\operatorname{arcsinh}\left(\frac{kx+ly}{lx-ky}\right)+\frac{c_g}{R_{\boldsymbol{x}}}t\right]\right\},$$

$$b(x,y,k,l,t) = \frac{R_{\boldsymbol{x}}}{R_{\boldsymbol{k}}}\left\{l\tanh\left[\operatorname{arcsinh}\left(\frac{kx+ly}{lx-ky}\right)+\frac{c_g}{R_{\boldsymbol{x}}}t\right]-k\,\operatorname{sech}\left[\operatorname{arcsinh}\left(\frac{kx+ly}{lx-ky}\right)+\frac{c_g}{R_{\boldsymbol{x}}}t\right]\right\},$$

The final solution becomes

$$\mathcal{W}(x,y,k,l,t) = \begin{cases} c_0, & \boldsymbol{x} \parallel \boldsymbol{k} \\ F_{kl}(a(x,y,k,l,t), b(x,y,k,l,t),k,l)\,, & \text{otherwise} \end{cases}. \tag{D.8}$$

In addition, the asymptotic behavior can be described as

$$t \to \pm\infty, \qquad a \to \pm\frac{R_{\boldsymbol{x}}}{R_{\boldsymbol{k}}}k, \qquad b \to \pm\frac{R_{\boldsymbol{x}}}{R_{\boldsymbol{k}}}l.$$

Checking the solution shows that as expected, the equation

$$\partial_t\mathcal{W} - P_{\boldsymbol{x}}\,\nabla_{\boldsymbol{k}}\Omega\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = 0,$$

automatically satisfies the condition $\hat{\boldsymbol{x}}\cdot\nabla_{\boldsymbol{x}}\mathcal{W} = 0$ since

$$\hat{\boldsymbol{x}}\cdot\nabla_{\boldsymbol{x}}F_{kl}(a,b) = x\,(\,\partial_a F\,\partial_x a + \,\partial_b F\,\partial_x b) + y\,(\,\partial_a F\,\partial_y a + \,\partial_b F\,\partial_y b) = 0.$$

D.3.3 Variant 3

Instead of holding γ fixed, let it be variable. A convenient choice is

$$\partial_t\mathcal{W} - \boldsymbol{\nu}_\perp\cdot\nabla_{\boldsymbol{x}}\mathcal{W} - |\boldsymbol{\nu}_\parallel|\,(\hat{x}\cdot\nabla_{\boldsymbol{x}}\mathcal{W}) = 0.$$

Here

$$|\boldsymbol{\nu}_\parallel| = \frac{\sqrt{\mathrm{g}}(kx+ly)}{2(k^2+l^2)^{3/4}(x^2+y^2)^{1/2}},$$

and our final problem becomes

$$\begin{aligned}\partial_t\mathcal{W} - \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)}\left\{(lx-ky)\begin{bmatrix}-y & x\end{bmatrix}\begin{bmatrix}\partial_x\mathcal{W}\\ \partial_y\mathcal{W}\end{bmatrix} + (kx+ly)\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}\partial_x\mathcal{W}\\ \partial_y\mathcal{W}\end{bmatrix}\right\} &= \\ \partial_t\mathcal{W} - \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}(x^2+y^2)}\begin{bmatrix}\partial_x\mathcal{W} & \partial_y\mathcal{W}\end{bmatrix}\begin{bmatrix}-y(lx-ky)+x(kx+ly)\\ x(lx-ky)+y(kx+ly)\end{bmatrix} &= \\ \partial_t\mathcal{W} - \frac{\sqrt{\mathrm{g}}}{2(k^2+l^2)^{3/4}}\begin{bmatrix}k & l\end{bmatrix}\begin{bmatrix}\partial_x\mathcal{W}\\ \partial_y\mathcal{W}\end{bmatrix} &= 0.\end{aligned} \tag{D.9}$$

Turning the crank, we find

$$\begin{aligned}\frac{d\mathcal{W}}{ds} &= \frac{dk}{ds} = \frac{dl}{ds} = 0,\\ \frac{dt}{ds} &= 1,\\ \frac{dx}{ds} &= \frac{-\sqrt{\mathrm{g}}\,k}{2(k^2+l^2)^{3/4}},\\ \frac{dy}{ds} &= \frac{-\sqrt{\mathrm{g}}\,l}{2(k^2+l^2)^{3/4}}.\end{aligned}$$

Using the same initial conditions, we find $t(s) = s$, $k(s) = k_0$, $l(s) = l_0$. Again, we would like to find two functions such that $a(x, y, k, l, t) = x_0$ and $b(x, y, k, l, t) = y_0$. Let $c_g = \frac{1}{2}\sqrt{\frac{\mathrm{g}}{R_{\boldsymbol{k}}}}$. Then the solution to our new set of ODEs is

$$\begin{aligned}\frac{dx}{ds} &= \frac{-c_g k_0}{R_{\boldsymbol{k}}} &\Rightarrow\quad x(s) &= \frac{-c_g k_0}{R_{\boldsymbol{k}}}s + x_0,\\ \frac{dy}{ds} &= \frac{-c_g l_0}{R_{\boldsymbol{k}}} &\Rightarrow\quad y(s) &= \frac{-c_g l_0}{R_{\boldsymbol{k}}}s + y_0.\end{aligned}$$

Solving, we find

$$\mathcal{W}\,(x, y, k, l, t) = F(x + \frac{c_g k}{R_{\boldsymbol{k}}}t, y + \frac{c_g l}{R_{\boldsymbol{k}}}t, k, l) = F_{kl}(x + \frac{c_g k}{R_{\boldsymbol{k}}}t, y + \frac{c_g l}{R_{\boldsymbol{k}}}t). \tag{D.10}$$

For each $\boldsymbol{k}$, an initial circle of radius $R_{\boldsymbol{x}}$ centered at the origin at time $t = 0$, propagates in the $\pm\hat{\boldsymbol{k}}$ direction on the F_{kl} plane with speed c_g.

Appendix E

Peer reviewed articles

E.1 Webb & Fox-Kemper, 2011

Appendix F

Numerical code

F.1 Matlab SD functions

```
function [stokesDepthArray] = fnc_stokes_uni_magnitude ( ...
    spec1dArray, freqVector, deltaFreqVector, depthVector )

%% CALCULATE 1DH—UNI—SD MAGNITUDE:
% This function uses 1D wave spectra with the unidirectional assumption to
% approximate the Stokes drift (SD) magnitude at some specified depth for
% an arbitrary number of spatial and temporal dimensions.
% Created by Adrean Webb.
%
% Requires: nothing.
%
% function [stokesDepthArray] = fnc_stokes_uni_magnitude ( ...
%    spec1dArray, freqVector, deltaFreqVector, depthVector )
%
% In:   'spec1dArray'       1D spectra with dimensions [nFreq ... ... ].
%       'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'depthVector'       Depth is defined as ABS(z).
%
% Out:  'stokesDepthArray'  1Dh—Uni—SD values with dimensions
%                            [nDepth ... ...].
%
% Version: 1.1 [2012/11/5] — Expanded capability to handle an arbitrary
%                             number of spatial and temporal dimensions.
%                             The script columnizes the extra dimensions in
%                             the 1D spectra and reshapes output to match.
%          1.2 [2012/11/5] — Changed script to run on older versions of
%                             Matlab less than 2011b.


%% 1. CHECK AND CONVERT INPUT IF NECESSARY.

if verLessThan('matlab','7.13')
    nargchk(4,4,nargin);
else
    narginchk(4,4);
end

subfnc_check_column_vector(freqVector,deltaFreqVector,depthVector);

subfnc_check_freq_cutoff(freqVector,deltaFreqVector);

nFreq = length(freqVector);
[flgSpec1d, nDimension] = subfnc_check_array(spec1dArray,nFreq);

[spec1dMatrix,rebuildVector] = subfnc_reduce_dimension(...
    spec1dArray,2,nDimension,flgSpec1d);


%% 2. PRELIMINARY INITIALIZATION AND CALCULATIONS.

G = 9.81;

```

```
nDepth = size(depthVector,1); nPoint = size(spec1dMatrix,2);

stokesSumStaticMatrix = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G );
stokesTailStaticVector = subfnc_SD_tail_static( freqVector(end,1), ...
    depthVector, G );


%% 3. CALCULATE DEPTH—DEPENDENT 1DH—UNI—SD.

stokesDepthMatrix = zeros(nDepth,nPoint);
for indPoint=1:nPoint
    for indDepth=1:nDepth
        stokesDepthMatrix(indDepth,indPoint) = sum( ...
            stokesSumStaticMatrix(:,indDepth) .* ...
            spec1dMatrix(:,indPoint) ) + ...
            stokesTailStaticVector(indDepth) * spec1dMatrix(end,indPoint);
    end
end

stokesDepthArray = subfnc_expand_dimension(stokesDepthMatrix,2,...
    nDimension,rebuildVector,flgSpec1d);

%%%%% END MAIN FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_column_vector( varargin )

%% CHECK COLUMN VECTOR (SPECIFIC):
%
```

```
% In:   'varargin'      Accepts any number of arguments in. Did not use
%                        the iscolumn command since it is not supported in
%                        older versions.


%% 1. VARIFY COLUMN VECTOR INPUT.

for ind=1:nargin
    if ( isvector(varargin{ind}) ~= 1 ) || ...
            ( size(varargin{ind},1) < length(varargin{ind}) )
        error([ 'Check input. Function requires vectors to be in ' ...
            'column form.' ])
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_freq_cutoff( freqVector, deltaFreqVector )

%% CHECK DELTA FREQUENCY CUTOFF (SPECIFIC):
%
% In:   'freqVector'        Frequency bin values.
%       'deltaFreqVector'   Bandwidth of each frequency bin.


%% 1. ENSURE FINAL FREQUENCY VALUE FALLS ON RIGHT EDGE.

freqLength1 = freqVector(end) - freqVector(end-1);
freqLength2 = deltaFreqVector(end) + deltaFreqVector(end-1)/2;
```

```
if (freqLength2 - freqLength1)/freqLength1 > 0.05
    error(['The cutoff frequency bin value should fall on the right ' ...
        'edge. Check the final delta frequency value to ensure it is ' ...
        'not centered.'])
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [flgReshape, nDimension] = subfnc_check_array( spec1dArray, ...
    nFreq )

%% CHECK 1D SPECTRUM INPUT (SPECIFIC):
%
% In:   'spec1dArray'   1D spectral array with frequency as the first
%                        dimension (the rest are location or time).
%       'nFreq'         Size of frequency bin.
% Out:  'flgReshape'    Flag indicates whether input array needs to be
%                        reshaped (1) or not (0).
%       'nDimension'    Number of dimensions in 1D spectral array.


%% 1. CHECK DIMENSIONS AND ASSIGN FLAG.

nDimension = ndims(spec1dArray); flgReshape = 0;
if nDimension == 1
    error( 'The frequency spectrum has a singleton dimension.')
elseif nDimension > 2
    flgReshape = 1;
end
```

```


%% 2. VERIFY ORIENTATION OF ARRAY.

if size(spec1dArray,1) ˜= nFreq
    error([ 'The frequency spectrum must be the first dimension in ' ...
        'the 1D spectral input.' ])
end

for indDimension=2:nDimension
    if size(spec1dArray,1) == size(spec1dArray,indDimension)
        disp(['Warning: Cannot determine if the matrix for the 1D ' ...
            'frequency spectrum is orientated correctly. Ensure the ' ...
            'dimensions are [frequency, location].'])
        pause
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [outputArray, reshapeVector] = ...
    subfnc_reduce_dimension( inputArray, indMin, indMax, flgReshape )

%% REDUCE ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'        The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'        Flag == 0 / 1 indicates no action / action.
%
```

```
% Out:  'outputArray'       The reshaped array (for flag == 1).
%       'shapeVector'       Information containing old array dimensions.


%% 1. REDUCE DIMENSION OF ARRAY IF NECESSARY.

reshapeVector = [];

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nDimension = ndims(inputArray);
    if (indMin < 1) || (indMax > nDimension) || (indMax <= indMin) || ...
            (indMin == 1 && indMax == nDimension)
        error( 'Check dimension delimiters.')
    end
    for ind=1:nDimension
        inputLength{ind} = size( inputArray, ind );
    end
    indReshape = [];
    if indMin > 1
        for ind=1:indMin−1;
            indReshape = [indReshape inputLength{ind}];
        end
    end
    nPoint = 1;
    for ind=indMin:indMax
        nPoint = nPoint * inputLength{ind};
        reshapeVector = [reshapeVector inputLength{ind}];
    end
    indReshape = [indReshape nPoint];
    if indMax < nDimension
        for ind=indMax+1:nDimension
```

```
            indReshape = [indReshape inputLength{ind}];
        end
    end
    outputArray = reshape( inputArray, indReshape );
else
    error( 'Flag incorrectly assigned.' );
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [outputArray] = subfnc_expand_dimension( inputArray, indMin, ...
    indMax, reshapeVector, flgReshape )

%% EXPAND ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'        The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'        Flag == 0 / 1 indicates no action / action.
%       'reshapeVector'     Vector containing sizes of the new dimensions.
%
% Out:  'outputArray'       The reshaped array (for flag == 1).


%% 1. EXPAND DIMENSION OF ARRAY IF NECESSARY.

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nReshapeVector = length(reshapeVector);
    nDimensionInput = ndims(inputArray);
    nDimensionOutput = nDimensionInput + nReshapeVector - 1;

    if (indMin < 1) || ( indMax <= indMin ) || ...
            ( nReshapeVector ~= indMax-indMin+1 )
        error( 'Check dimension delimiters.')
    elseif ( size(inputArray,indMin) ~= prod(reshapeVector) )
        error( 'Number of elements in new array are not the same.')
    end

    for ind=1:nDimensionInput
        inputLength(ind) = size( inputArray, ind );
    end

    expandVector = [];
    if indMin > 1
        for ind=1:indMin-1;
            expandVector = [expandVector size(inputArray,ind)];
        end
    end
    for ind=1:(indMax-indMin+1)
        expandVector = [expandVector reshapeVector(ind)];
    end
    if indMax < nDimensionOutput
        for ind=1:nDimensionOutput-indMax
            expandVector = [expandVector inputLength(ind+indMin)];
        end
    end
    outputArray = reshape( inputArray, expandVector );
else
    error( 'Flag incorrectly assigned.' );
end
```

```
%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sumStaticMatrix] = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G )

%% SD STATIC SUM COMPONENTS (SPECIFIC):
%
% In:   'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'depthVector'       Depth is defined as ABS(z).
%       'G'                 Gravity constant.
%
% Out:  'sumStaticMatrix'   The static components of the 1Dh—Uni—SD sum.
%                             Dimensions are [sumStaticComponents,depth].


%% 1. CALCULATE STATIC SUM COMPONENTS.

INTEGRAND_CONSTANT = 16 * pi^3 / G;
nDepth = size(depthVector,1);
nFreq = size(freqVector,1);

betaMatrix = zeros(nFreq,nDepth);
for indDepth=1:nDepth
    betaMatrix(:,indDepth) = 8 * pi^2 / G * depthVector(indDepth) * ...
        freqVector.^2;
end

sumStaticMatrix = zeros(nFreq,nDepth);
```

```
for indDepth=1:nDepth
    sumStaticMatrix(:,indDepth) = INTEGRAND_CONSTANT * ...
        deltaFreqVector .* freqVector.^3 .* exp(-betaMatrix(:,indDepth));
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tailStaticVector] = subfnc_SD_tail_static( freqCutoff, ...
    depthVector, G )

%% SD STATIC TAIL COMPONENT (SPECIFIC):
%
% In:   'freqCutoff'    The frequency cutoff is the last frequency value.
%       'depthVector'   Depth is defined as ABS(z).
%       'G'             Gravity constant.
%
% Out:  'tailStaticVector'  Static component of the SD tail calculation.


%% 1. CALCULATE STATIC TAIL COMPONENT.

INTEGRAND_CONSTANT = 16 * pi^3 / G ;

alphaVector = 8 * pi^2 * depthVector / G ;
betaVector = alphaVector * freqCutoff^2 ;
erfcVector = 1 - erf( sqrt(betaVector) ) ;

tailStaticVector = INTEGRAND_CONSTANT * freqCutoff^4 * ...
    ( exp(-betaVector) - sqrt( pi * betaVector ) .* erfcVector ) ;
```

```

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [stokesDepthArray] = fnc_stokes_dhh_magnitude ( ...
    spec1dArray, freqVector, deltaFreqVector, depthVector )

%% CALCULATE 1DH-DHH-SD MAGNITUDE:
% This function uses 1D wave spectra with the DHH directional-SD-component
% to approximate the Stokes drift (SD) magnitude at some specified depth
% for an arbitrary number of spatial and temporal dimensions.
% Created by Adrean Webb.
%
% Requires: nothing.
%
% function [stokesDepthArray] = fnc_stokes_dhh_magnitude ( ...
%    spec1dArray, freqVector, deltaFreqVector, depthVector )
%
% In:   'spec1dArray'       1D spectra with dimensions [nFreq ... ... ].
%       'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'depthVector'       Depth is defined as ABS(z).
%
% Out:  'stokesDepthArray'  1Dh-DHH-SD values with dimensions
%                            [nDepth ... ...].
%
% Version: 1.1 [2012/11/5] - Expanded capability to handle an arbitrary
%                             number of spatial and temporal dimensions.
%                             The script columnizes the extra dimensions in
%                             the 1D spectra and reshapes output to match.
%          1.2 [2012/11/5] - Changed script to run on older versions of
%                             Matlab less than 2011b.
```

```


%% 1. CHECK AND CONVERT INPUT IF NECESSARY.

if verLessThan('matlab','7.13')
    nargchk(4,4,nargin);
else
    narginchk(4,4);
end

subfnc_check_column_vector(freqVector,deltaFreqVector,depthVector);

subfnc_check_freq_cutoff(freqVector,deltaFreqVector);

nFreq = length(freqVector);
[flgSpec1d, nDimension] = subfnc_check_array(spec1dArray,nFreq);

[spec1dMatrix,rebuildVector] = subfnc_reduce_dimension(...
    spec1dArray,2,nDimension,flgSpec1d);


%% 2. PRELIMINARY INITIALIZATION AND CALCULATIONS.

G = 9.81;

nDepth = size(depthVector,1); nPoint = size(spec1dMatrix,2);

stokesSumStaticMatrix = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G );
stokesTailStaticVector = subfnc_SD_tail_static( freqVector(end,1), ...
    depthVector, G );


```

```
%% 3. CALCULATE THE DHH DIRECTIONAL-SD-COMPONENT.

hMatrix = zeros(nFreq,nPoint);
for indPoint=1:nPoint
    [specPeak,indSpecPeak] = max( spec1dMatrix(:,indPoint) );
    if ( isnan(specPeak) ~= 1 )
        rVector = freqVector / freqVector(indSpecPeak);
        hMatrix(:,indPoint) = subfnc_SD_directional_component_DHH(rVector);
    else
        hMatrix(:,indPoint) = NaN;
    end
end


%% 4. CALCULATE DEPTH-DEPENDENT 1DH-DHH-SD.

stokesSumDynamicMatrix = hMatrix .* spec1dMatrix;

stokesDepthMatrix = zeros(nDepth,nPoint);
for indPoint=1:nPoint
    stokesTailVector = stokesTailStaticVector * ...
        stokesSumDynamicMatrix(end,indPoint);
    for indDepth=1:nDepth
        stokesDepthMatrix(indDepth,indPoint) = sum( ...
            stokesSumStaticMatrix(:,indDepth) .* ...
            stokesSumDynamicMatrix(:,indPoint) ) + ...
            stokesTailVector(indDepth);
    end
end

stokesDepthArray = subfnc_expand_dimension(stokesDepthMatrix,2,...
    nDimension,rebuildVector,flgSpec1d);

```

```
%%%%% END MAIN FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_column_vector( varargin )

%% CHECK COLUMN VECTOR (SPECIFIC):
%
% In:   'varargin'      Accepts any number of arguments in. Did not use
%                        the iscolumn command since it is not supported in
%                        older versions.


%% 1. VARIFY COLUMN VECTOR INPUT.

for ind=1:nargin
    if ( isvector(varargin{ind}) ~= 1 ) || ...
            ( size(varargin{ind},1) < length(varargin{ind}) )
        error([ 'Check input. Function requires vectors to be in ' ...
            'column form.' ])
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_freq_cutoff( freqVector, deltaFreqVector )

```

```
%% CHECK DELTA FREQUENCY CUTOFF (SPECIFIC):
%
% In:   'freqVector'        Frequency bin values.
%       'deltaFreqVector'   Bandwidth of each frequency bin.


%% 1. ENSURE FINAL FREQUENCY VALUE FALLS ON RIGHT EDGE.

freqLength1 = freqVector(end) - freqVector(end-1);
freqLength2 = deltaFreqVector(end) + deltaFreqVector(end-1)/2;
if (freqLength2 - freqLength1)/freqLength1 > 0.05
    error(['The cutoff frequency bin value should fall on the right ' ...
        'edge. Check the final delta frequency value to ensure it is ' ...
        'not centered.'])
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [flgReshape, nDimension] = subfnc_check_array( spec1dArray, ...
    nFreq )

%% CHECK 1D SPECTRUM INPUT (SPECIFIC):
%
% In:   'spec1dArray'   1D spectral array with frequency as the first
%                        dimension (the rest are location or time).
%       'nFreq'         Size of frequency bin.
% Out:  'flgReshape'    Flag indicates whether input array needs to be
%                        reshaped (1) or not (0).
%       'nDimension'    Number of dimensions in 1D spectral array.
```

```


%% 1. CHECK DIMENSIONS AND ASSIGN FLAG.

nDimension = ndims(spec1dArray); flgReshape = 0;
if nDimension == 1
    error( 'The frequency spectrum has a singleton dimension.')
elseif nDimension > 2
    flgReshape = 1;
end


%% 2. VERIFY ORIENTATION OF ARRAY.

if size(spec1dArray,1) ˜= nFreq
    error([ 'The frequency spectrum must be the first dimension in ' ...
        'the 1D spectral input.' ])
end

for indDimension=2:nDimension
    if size(spec1dArray,1) == size(spec1dArray,indDimension)
        disp(['Warning: Cannot determine if the matrix for the 1D ' ...
            'frequency spectrum is orientated correctly. Ensure the ' ...
            'dimensions are [frequency, location].'])
        pause
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

function [outputArray, reshapeVector] = ...
    subfnc_reduce_dimension( inputArray, indMin, indMax, flgReshape )

%% REDUCE ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'        The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'        Flag == 0 / 1 indicates no action / action.
%
% Out:  'outputArray'       The reshaped array (for flag == 1).
%       'shapeVector'       Information containing old array dimensions.


%% 1. REDUCE DIMENSION OF ARRAY IF NECESSARY.

reshapeVector = [];

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nDimension = ndims(inputArray);
    if (indMin < 1) || (indMax > nDimension) || (indMax <= indMin) || ...
            (indMin == 1 && indMax == nDimension)
        error( 'Check dimension delimiters.')
    end
    for ind=1:nDimension
        inputLength{ind} = size( inputArray, ind );
    end
    indReshape = [];
    if indMin > 1
        for ind=1:indMin−1;
            indReshape = [indReshape inputLength{ind}];
        end
    end
    nPoint = 1;
    for ind=indMin:indMax
        nPoint = nPoint * inputLength{ind};
        reshapeVector = [reshapeVector inputLength{ind}];
    end
    indReshape = [indReshape nPoint];
    if indMax < nDimension
        for ind=indMax+1:nDimension
            indReshape = [indReshape inputLength{ind}];
        end
    end
    outputArray = reshape( inputArray, indReshape );
else
    error( 'Flag incorrectly assigned.' );
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [outputArray] = subfnc_expand_dimension( inputArray, indMin, ...
    indMax, reshapeVector, flgReshape )

%% EXPAND ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'        The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'        Flag == 0 / 1 indicates no action / action.
%       'reshapeVector'     Vector containing sizes of the new dimensions.
```

```
%
% Out:  'outputArray'       The reshaped array (for flag == 1).



%% 1. EXPAND DIMENSION OF ARRAY IF NECESSARY.

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nReshapeVector = length(reshapeVector);
    nDimensionInput = ndims(inputArray);
    nDimensionOutput = nDimensionInput + nReshapeVector - 1;

    if (indMin < 1) || ( indMax <= indMin ) || ...
            ( nReshapeVector ~= indMax-indMin+1 )
        error( 'Check dimension delimiters.')
    elseif ( size(inputArray,indMin) ~= prod(reshapeVector) )
        error( 'Number of elements in new array are not the same.')
    end

    for ind=1:nDimensionInput
        inputLength(ind) = size( inputArray, ind );
    end

    expandVector = [];
    if indMin > 1
        for ind=1:indMin-1;
            expandVector = [expandVector size(inputArray,ind)];
        end
    end
    for ind=1:(indMax-indMin+1)
        expandVector = [expandVector reshapeVector(ind)];
    end
    if indMax < nDimensionOutput
        for ind=1:nDimensionOutput-indMax
            expandVector = [expandVector inputLength(ind+indMin)];
        end
    end
    outputArray = reshape( inputArray, expandVector );
else
    error( 'Flag incorrectly assigned.' );
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [hVector] = subfnc_SD_directional_component_DHH( rVector )

%% DHH DIRECTIONAL-SD-COMPONENT (SPECIFIC):
%
% In:   'rVector'   'r' is defined as r = f/f_peak.
%
% Out:  'hVector'   'h' is the piecewise continuous DHH directional-SD-
%                     component.


%% 1. CREATE PIECEWISE CONTINUOUS H FUNCTION.

nR = size(rVector,1);

hVector = zeros(nR,1);
for ind=1:nR
    if ( rVector(ind) <= 0.56 )
```

```
        hVector(ind) = 0.777;
    end
    if ( rVector(ind) > 0.56 ) && ( rVector(ind) <= 0.95 )
        hVector(ind) = ...
            ( 0.52 - 3.3 * rVector(ind) + 8.9 * rVector(ind)^2) / ...
            ( 1 - 3.4 * rVector(ind) + 8.9 * rVector(ind)^2 );
    end
    if ( rVector(ind) > 0.95 ) && ( rVector(ind) < 1.6 )
        hVector(ind) = ...
            ( 0.98 - 0.19 * rVector(ind) + 0.0058 * rVector(ind)^2) / ...
            ( 1 - 0.26 * rVector(ind) + 0.12 * rVector(ind)^2 );
    end
    if ( rVector(ind) >= 1.6 )
        hVector(ind) = 0.777;
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sumStaticMatrix] = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G )

%% SD STATIC SUM COMPONENTS (SPECIFIC):
%
% In:   'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'depthVector'       Depth is defined as ABS(z).
%       'G'                 Gravity constant.
%
```

```
% Out:  'sumStaticMatrix'   The static components of the 1Dh−DHH−SD sum.
%                             Dimensions are [sumStaticComponents,depth].


%% 1. CALCULATE STATIC SUM COMPONENTS.

INTEGRAND_CONSTANT = 16 * pi^3 / G;
nDepth = size(depthVector,1);
nFreq = size(freqVector,1);

betaMatrix = zeros(nFreq,nDepth);
for indDepth=1:nDepth
    betaMatrix(:,indDepth) = 8 * pi^2 / G * depthVector(indDepth) * ...
        freqVector.^2;
end

sumStaticMatrix = zeros(nFreq,nDepth);
for indDepth=1:nDepth
    sumStaticMatrix(:,indDepth) = INTEGRAND_CONSTANT * ...
        deltaFreqVector .* freqVector.^3 .* exp(−betaMatrix(:,indDepth));
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tailStaticVector] = subfnc_SD_tail_static( freqCutoff, ...
    depthVector, G )

%% SD STATIC TAIL COMPONENT (SPECIFIC):
%
```

```
% In:   'freqCutoff'    The frequency cutoff is the last frequency value.
%       'depthVector'   Depth is defined as ABS(z).
%       'G'             Gravity constant.
%
% Out:  'tailStaticVector'  Static component of the SD tail calculation.


%% 1. CALCULATE STATIC TAIL COMPONENT.

INTEGRAND_CONSTANT = 16 * pi^3 / G ;

alphaVector = 8 * pi^2 * depthVector / G ;
betaVector = alphaVector * freqCutoff^2 ;
erfcVector = 1 - erf( sqrt(betaVector) ) ;

tailStaticVector = INTEGRAND_CONSTANT * freqCutoff^4 * ...
    ( exp(-betaVector) - sqrt( pi * betaVector ) .* erfcVector ) ;

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [stokesMagnitudeArray,stokesDirectionArray] = ...
    fnc_stokes_full_components ( spec2dArrayNd, freqVector, ...
    deltaFreqVector, thetaVector, deltaThetaValue, depthVector )

%% CALCULATE 2DH-SD MAGNITUDE AND DIRECTION:
% This function uses 2D wave spectra to calculate the Stokes drift (SD)
% magnitude and direction at some specified depth for an arbitrary number
% of spatial and temporal dimensions. Created by Adrean Webb.
%
% Requires: nothing.
%
```

```
% function [stokesMagnitudeArray, stokesDirectionArray] = ...
%   fnc_stokes_full_components( spec2dArrayNd, freqVector, ...
%   deltaFreqVector, thetaVector, deltaThetaValue, depthVector )
%
% In:   'spec2dArrayNd'     2D directional—frequency spectra with
%                            dimensions [nFreq nTheta ... ...].
%       'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'thetaVector'       Periodic theta bin vector in radians (centered).
%                            [theta1:deltaTheta:theta1+2pi—deltaTheta]
%       'deltaThetaValue'   Delta theta value in radians (constant).
%       'depthVector'       Depth is defined as ABS(z).
%
% Out:  'stokesMagnitudeArray'     2Dh—SD magnitude with dimensions
%                                    [nDepth ... ...].
%       'stokesDirectionArray'     2Dh—SD direction with dimensions
%                                    [nDepth ... ...].
%
% Version: 1.1 [2012/11/6] — Expanded capability to handle an arbitrary
%                             number of spatial and temporal dimensions.
%                             The script columnizes the extra dimensions in
%                             the 1D spectra and reshapes output to match.
%          1.2 [2012/11/6] — Changed script to run on older versions of
%                             Matlab less than 2011b.


%% 1. CHECK AND CONVERT INPUT IF NECESSARY.

if verLessThan('matlab','7.13')
    nargchk(6,6,nargin);
else
    narginchk(6,6);
end
```

```

subfnc_check_column_vector(freqVector,deltaFreqVector,thetaVector,...
    depthVector);

subfnc_check_freq_cutoff(freqVector,deltaFreqVector);

nFreq = length(freqVector); nTheta = length(thetaVector);

[flgSpec2d,nDimension] = subfnc_check_orientation(spec2dArrayNd,nFreq,...
    nTheta);

[spec2dArray3d,rebuildVector] = subfnc_reduce_dimension(...
    spec2dArrayNd,3,nDimension,flgSpec2d);


%% 2. PRELIMINARY INITIALIZATION AND CALCULATIONS.

G = 9.81;

nDepth = size(depthVector,1); nPoint = size(spec2dArray3d,3);

stokesSumStaticMatrix = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G );
stokesTailStaticVector = subfnc_SD_tail_static( freqVector(end,1), ...
    depthVector, G );


%% 3. INTEGRATE FREQUENCY COMPONENT.

stokesStepArray = zeros(nTheta,nDepth,nPoint);
for indPoint=1:nPoint
    for indDepth=1:nDepth
        for indTheta=1:nTheta
```

```
            stokesStepArray(indTheta,indDepth,indPoint) = ...
                sum( stokesSumStaticMatrix(:,indDepth) .* ...
                spec2dArray3d(:,indTheta,indPoint) ) + ...
                stokesTailStaticVector(indDepth) * ...
                spec2dArray3d(end,indTheta,indPoint);
        end
    end
end


%% 4. INTEGRATE THETA COMPONENT.

stokesDepthXMatrix = zeros(nDepth,nPoint);
stokesDepthYMatrix = zeros(nDepth,nPoint);
for indPoint=1:nPoint
    for indDepth=1:nDepth
        stokesDepthXMatrix(indDepth,indPoint) = deltaThetaValue * ...
            sum( cos(thetaVector) .* stokesStepArray(:,indDepth,indPoint) );
        stokesDepthYMatrix(indDepth,indPoint) = deltaThetaValue * ...
            sum( sin(thetaVector) .* stokesStepArray(:,indDepth,indPoint) );
    end
end
stokesMagnitudeMatrix = sqrt( stokesDepthXMatrix.^2 + ...
    stokesDepthYMatrix.^2 );
stokesDirectionMatrix = atan2( stokesDepthYMatrix, stokesDepthXMatrix );

stokesMagnitudeArray = subfnc_expand_dimension(stokesMagnitudeMatrix,2,...
    nDimension-1,rebuildVector,flgSpec2d);
stokesDirectionArray = subfnc_expand_dimension(stokesDirectionMatrix,2,...
    nDimension-1,rebuildVector,flgSpec2d);


%%%%% END MAIN FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_column_vector( varargin )

%% CHECK COLUMN VECTOR (SPECIFIC):
%
% In:   'varargin'      Accepts any number of arguments in. Did not use
%                        the iscolumn command since it is not supported in
%                        older versions.


%% 1. VARIFY COLUMN VECTOR INPUT.

for ind=1:nargin
    if ( isvector(varargin{ind}) ~= 1 ) || ...
            ( size(varargin{ind},1) < length(varargin{ind}) )
        error([ 'Check input. Function requires vectors to be in ' ...
            'column form.' ])
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function subfnc_check_freq_cutoff( freqVector, deltaFreqVector )

%% CHECK DELTA FREQUENCY CUTOFF (SPECIFIC):
```

```
%
% In:   'freqVector'        Frequency bin values.
%       'deltaFreqVector'   Bandwidth of each frequency bin.


%% 1. ENSURE FINAL FREQUENCY VALUE FALLS ON RIGHT EDGE.

freqLength1 = freqVector(end) - freqVector(end-1);
freqLength2 = deltaFreqVector(end) + deltaFreqVector(end-1)/2;
if (freqLength2 - freqLength1)/freqLength1 > 0.05
    error(['The cutoff frequency bin value should fall on the right ' ...
        'edge. Check the final delta frequency value to ensure it is ' ...
        'not centered.'])
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [flgReshape,nDimension] = subfnc_check_orientation( ...
    spec2dArrayNd, nFreq, nTheta )

%% CHECK ARRAY ORIENATION OF 2D DIRECTIONAL-FREQUENCY SPECTRA (SPECIFIC):
%
% In:   'spec2dArrayNd'     2D directional-frequency wave spectra with
%                            dimensions [nFreq nTheta ... ...].
%       'nFreq'             Size of frequency bin.
%       'nTheta'            Size of theta bin.
%
% Out:  'flgReshape'        Flag indicates whether input array needs to be
%                            reshaped (1) or not (0).
%       'nDimension'        Number of dimensions in 1D spectral array.


%% 1. CHECK DIMENSIONS AND ASSIGN FLAG.

nDimension = ndims(spec2dArrayNd); flgReshape = 0;

if nDimension == 1
    error( 'The frequency—directional spectrum has a singleton dimension.')
elseif ( nDimension == 2 ) && ( min( size(spec2dArrayNd) ) == 1 )
    error([ 'The frequency or directional component has a singleton ' ...
        'dimension.' ])
elseif nDimension > 2
    flgReshape = 1;
end


%% 2. VERIFY ORIENTATION OF ARRAY.

if size(spec2dArrayNd,1) ~= nFreq
    error([ 'The frequency spectrum must be the first dimension in ' ...
        'the 2D spectral input array.' ])
elseif size(spec2dArrayNd,2) ~= nTheta
    error([ 'The directional spectrum must be the second dimension in ' ...
        'the 2D spectral input array.' ])
end

for indDimension=2:nDimension
    if size(spec2dArrayNd,1) == size(spec2dArrayNd,indDimension)
        disp(['Warning: Cannot determine if the array for the 2D ' ...
            'directional—frequency spectrum is orientated correctly. ' ...
            'Ensure the dimensions are [frequency, direction ... ...].' ])
        pause
    end
end

if nDimension >=3
    for indDimension=3:nDimension
        if size(spec2dArrayNd,2) == size(spec2dArrayNd,indDimension)
            disp(['Warning: Cannot determine if the array for the 2D ' ...
                'directional—frequency spectrum is orientated ' ...
                'correctly. Ensure the dimensions are [frequency, ' ...
                'direction ... ...].' ])
            pause
        end
    end
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [outputArray, reshapeVector] = ...
    subfnc_reduce_dimension( inputArray, indMin, indMax, flgReshape )

%% REDUCE ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'         The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'         Flag == 0 / 1 indicates no action / action.
%
% Out:  'outputArray'        The reshaped array (for flag == 1).
%       'shapeVector'        Information containing old array dimensions.

```

```

%% 1. REDUCE DIMENSION OF ARRAY IF NECESSARY.

reshapeVector = [];

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nDimension = ndims(inputArray);
    if (indMin < 1) || (indMax > nDimension) || (indMax <= indMin) || ...
            (indMin == 1 && indMax == nDimension)
        error( 'Check dimension delimiters.')
    end
    for ind=1:nDimension
        inputLength{ind} = size( inputArray, ind );
    end
    indReshape = [];
    if indMin > 1
        for ind=1:indMin-1;
            indReshape = [indReshape inputLength{ind}];
        end
    end
    nPoint = 1;
    for ind=indMin:indMax
        nPoint = nPoint * inputLength{ind};
        reshapeVector = [reshapeVector inputLength{ind}];
    end
    indReshape = [indReshape nPoint];
    if indMax < nDimension
        for ind=indMax+1:nDimension
            indReshape = [indReshape inputLength{ind}];
        end
    end
    outputArray = reshape( inputArray, indReshape );
else
    error( 'Flag incorrectly assigned.' );
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [outputArray] = subfnc_expand_dimension( inputArray, indMin, ...
    indMax, reshapeVector, flgReshape )

%% EXPAND ARRAY DIMENSION (GENERAL):
%
% In:   'inputArray'        The array to be reshaped.
%       'indMin', 'indMax'  The dimension delimiters for reshaping.
%       'flgReshape'        Flag == 0 / 1 indicates no action / action.
%       'reshapeVector'     Vector containing sizes of the new dimensions.
%
% Out:  'outputArray'       The reshaped array (for flag == 1).


%% 1. EXPAND DIMENSION OF ARRAY IF NECESSARY.

if flgReshape == 0
    outputArray = inputArray;
elseif flgReshape == 1
    nReshapeVector = length(reshapeVector);
    nDimensionInput = ndims(inputArray);
    nDimensionOutput = nDimensionInput + nReshapeVector - 1;

    if (indMin < 1) || ( indMax <= indMin ) || ...
            ( nReshapeVector ~= indMax-indMin+1 )
        error( 'Check dimension delimiters.')
    elseif ( size(inputArray,indMin) ~= prod(reshapeVector) )
        error( 'Number of elements in new array are not the same.')
    end

    for ind=1:nDimensionInput
        inputLength(ind) = size( inputArray, ind );
    end

    expandVector = [];
    if indMin > 1
        for ind=1:indMin-1;
            expandVector = [expandVector size(inputArray,ind)];
        end
    end
    for ind=1:(indMax-indMin+1)
        expandVector = [expandVector reshapeVector(ind)];
    end
    if indMax < nDimensionOutput
        for ind=1:nDimensionOutput-indMax
            expandVector = [expandVector inputLength(ind+indMin)];
        end
    end
    outputArray = reshape( inputArray, expandVector );
else
    error( 'Flag incorrectly assigned.' );
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sumStaticMatrix] = subfnc_SD_sum_static( freqVector, ...
    deltaFreqVector, depthVector, G )

%% SD STATIC SUM COMPONENTS (SPECIFIC):
%
% In:   'freqVector'        Frequency bin values (centered except for end).
%       'deltaFreqVector'   Bandwidth of each frequency bin.
%       'depthVector'       Depth is defined as ABS(z).
%       'G'                 Gravity constant.
%
% Out:  'sumStaticMatrix'   The static components of the 2Dh-SD sum.
%                             Dimensions are [sumStaticComponents,depth].


%% 1. CALCULATE STATIC SUM COMPONENTS.

INTEGRAND_CONSTANT = 16 * pi^3 / G;
nDepth = size(depthVector,1);
nFreq = size(freqVector,1);

betaMatrix = zeros(nFreq,nDepth);
for indDepth=1:nDepth
    betaMatrix(:,indDepth) = 8 * pi^2 / G * depthVector(indDepth) * ...
        freqVector.^2;
end

sumStaticMatrix = zeros(nFreq,nDepth);
for indDepth=1:nDepth
    sumStaticMatrix(:,indDepth) = INTEGRAND_CONSTANT * ...
        deltaFreqVector .* freqVector.^3 .* exp(-betaMatrix(:,indDepth));
```

```
end

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%% START SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tailStaticVector] = subfnc_SD_tail_static( freqCutoff, ...
    depthVector, G )

%% SD STATIC TAIL COMPONENT (SPECIFIC):
%
% In:   'freqCutoff'    The frequency cutoff is the last frequency value.
%       'depthVector'   Depth is defined as ABS(z).
%       'G'             Gravity constant.
%
% Out:  'tailStaticVector'  Static component of the SD tail calculation.


%% 1. CALCULATE STATIC TAIL COMPONENT.

INTEGRAND_CONSTANT = 16 * pi^3 / G ;

alphaVector = 8 * pi^2 * depthVector / G ;
betaVector = alphaVector * freqCutoff^2 ;
erfcVector = 1 - erf( sqrt(betaVector) ) ;

tailStaticVector = INTEGRAND_CONSTANT * freqCutoff^4 * ...
    ( exp(-betaVector) - sqrt( pi * betaVector ) .* erfcVector ) ;

%%%%% END SUBFUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

F.2 Matlab RBF-FD model scripts and functions

```
%% STENCIL GENERATOR FOR THE RBF-FD WAVE MODEL: core_generate_stencil.m
% This program generates a stencil for the RBF-FD wave model on the sphere-ring
% geometry. Created by Adrean Webb.
%
% Requires: 'kdtree' package in addition to model files and MD node data.


%% 1. INTIALIZE MODEL WITH NON-VARIABLE SETTINGS.

close all; clear all; tic
model_settings_initialize;


%% 2. VARIABLE SETTINGS.

nXApprox = 3600;  % APPROXIMATE NUMBER OF SPATIAL NODES.
nDir = 36  % NUMBER OF DIRECTIONS PER SPATIAL NODE.

epsilon0D = 3; % SHAPE PARAMETER
nXLocal = 17; nKLocal = 9; laplacianOrder0D = 4; laplacianDimension0D = 4;

% FINITE DIFFERENCE SETTINGS.
stencilSettings1D = [epsilon0D; nXLocal; nKLocal; laplacianOrder0D; ...
    laplacianDimension0D];


%% 3. CREATE STRING NAME FOR STENCIL FILE.

FID_SAVE = [ PATH.STENCIL 'rbffd_' num2str(nXApprox) 'x' num2str(nXLocal) ...
```

```
    '_' fnc_integer_to_string(nDir,2) 'd' num2str(nKLocal) ...
    '_h' num2str(laplacianOrder0D) '_e' num2str(epsilon0D-mod(epsilon0D,1)) ...
    'pt' num2str(10*mod(epsilon0D,1)) ]


%% 4. CREATE NORMALIZED SPATIAL NODES ON A SPHERE.

[xVector2D, nX] = fnc_generate_node_spherical(nXApprox); nX
xVector2D = [xVector2D(:,3), xVector2D(:,1), xVector2D(:,2)];

xVectorSort2D = zeros(nX,4); xVectorSort2D(:,1:3) = xVector2D;
for ii=1:nX
    xVectorSort2D(ii,4) = fnc_norm_great_circle(xVectorSort2D(1,1:3), ...
        xVectorSort2D(ii,1:3));
end

distanceFromReferenceNode = sortrows(xVectorSort2D,4);
deltaXApprox = distanceFromReferenceNode(2,4);


%% 5. CREATE NORMALIZED SPECTRAL NODES ON A RING.

deltaDir = 2*pi / nDir;  % periodic
dirVector1D = [0:deltaDir:2*pi-deltaDir]';

nK = nDir; % REDUNDANT BUT WILL BE IMPORTANT FOR POLYCHROMATIC MODEL
kVector2D = [cos(dirVector1D) sin(dirVector1D)]; % UNIT VECTOR


%% 6. CREATE ALPHA VECTOR.

nAlpha = nX* nK;
alphaVector2D = zeros(nAlpha,5);
```

```

for indX=1:nX
    indColumn = (nK * (indX - 1) + 1):(nK * indX)';
    alphaVector2D(indColumn,:) = [xVector2D(indX,1) * ones(nK,1) ...
        xVector2D(indX,2) * ones(nK,1) xVector2D(indX,3) * ones(nK,1) ...
        kVector2D(:,1) kVector2D(:,2)];
end


%% 7. CREATE RBF DIFFERENTIAL MATRIX.

sparseStencil = fnc_rbffd_sphere_ring( xVector2D, kVector2D, ...
    alphaVector2D, stencilSettings1D );
toc

nodeSettings.nXApprox = nXApprox; nodeSettings.nX = nX;
nodeSettings.nDir = nDir; nodeSettings.nK = nK; nodeSettings.nAlpha = nAlpha;
nodeSettings.deltaXApprox = deltaXApprox; nodeSettings.deltaDir = deltaDir;

save( FID_SAVE, 'xVector2D', 'dirVector1D', 'kVector2D', 'alphaVector2D', ...
    'nodeSettings', 'stencilSettings1D', 'sparseStencil', '-v7.3' )
```

```
%% RUN RBF-FD WAVE MODEL: core_run_gaussian_half_revolution.m
% This program runs the RBF-FD wave model on the sphere-ring geometry using a
% previously generated stencil. Created by Adrean Webb.


%% 1. INTIALIZE MODEL WITH NON-VARIABLE SETTINGS.

close all; clear all; tic
model_settings_initialize;
```

```


%% 2. VARIABLE SETTINGS FOR LOADING STENCILS.

nXApprox = 3600;  % APPROXIMATE NUMBER OF SPATIAL NODES.
nDir = 36  % NUMBER OF DIRECTIONS PER SPATIAL NODE.
epsilon0D = 3; % SHAPE PARAMETER
nXLocal = 17; nKLocal = 9; laplacianOrder0D = 4; laplacianDimension0D = 4;


%% 3. VARIABLE SETTINGS FOR RUNNING.

gamma0D = -1e8; strGamma0D = 'n1p8'
timeScale0D = 0.2;
initialWidth0D = pi/3; boundaryWidth0D = pi/6;
initialLongitude0D = 0; initialLatitude0D = 0;
initialSpread0D = pi/3; initialDirection0D = 3*pi/18

% DETERMINE GROUP VELOCITY.
% kRingNorm0D = 4e-2 ; % (M/S) TYPICAL SWELL WITH PERIOD OF 10 SECONDS.
% groupVelocity0D = 0.5* sqrt( G / kRingNorm0D ) ; % APPROXIMATELY 28 KM/HR.
% groupVelocity0D = groupVelocity0D / RADIUS_EARTH
groupVelocity0D = 1 % TO SPEED UP TESTING; NTIME IS AUTOMATICALLY ADJUSTED.


%% 4. LOAD STENCIL DATA.

FID_STENCIL = [ PATH.STENCIL 'rbffd_' num2str(nXApprox) 'x' num2str(nXLocal) ...
    '_' fnc_integer_to_string(nDir,2) 'd' num2str(nKLocal) '_h' num2str(laplacianOrder0D) ...
    '_e' num2str(epsilon0D-mod(epsilon0D,1)) 'pt' num2str(10*mod(epsilon0D,1)) 'b']

load(FID_STENCIL)

```

```
nX = nodeSettings.nX; nK = nodeSettings.nK; nAlpha = nodeSettings.nAlpha;
deltaXApprox = nodeSettings.deltaXApprox; deltaDir = nodeSettings.deltaDir;


%% 5. CREATE RBF-FD SPARSE MATRICES.

dXMatrix2D = sparseStencil.dX1 + sparseStencil.dX2 + sparseStencil.dX3 ...
    + sparseStencil.dK1 + sparseStencil.dK2 ;
dXMatrix2D = groupVelocity0D * dXMatrix2D;

damping0D = gamma0D * (nAlpha)^(-laplacianOrder0D);
dissipation2D = damping0D * sparseStencil.dissipation;
toc


%% 6. DISCRETIZE TIME.

deltaTime =  timeScale0D * 2*pi/sqrt(nXApprox) / groupVelocity0D;
nTime = ceil(sqrt(nXApprox) / 2 / timeScale0D / groupVelocity0D + 1)


%% 7. CREATE INITIAL CONDITIONS.

initialWaveAction1D = fnc_initial_conditions_gaussian( alphaVector2D, nDir, ...
    initialWidth0D, initialLongitude0D, initialLatitude0D, ...
    initialDirection0D, initialSpread0D );

% % % CHECK INITIAL CONDITIONS
% % figure; plot3k(xVector2D,'ColorData',initialWaveAction1D(indDirection1D), ...
% %     'ColorRange',[-.1 1.1],'Marker',{'o',10}); axis equal;
% % view([cos(initialLongitude0D) sin(initialLongitude0D) sin(initialLatitude0D)])
% % title('Initial conditions'); xlabel('x'); ylabel('y'); zlabel('z'); pause

```

```

%% 8. CREATE POLAR BOUNDARIES.

boundary1D = fnc_boundary_attenuation_gaussian(alphaVector2D,boundaryWidth0D);

% % % CHECK BOUNDARY ATTENUATION
% % figure; plot3( atan2(alphaVector2D(indDirection1D,2), ...
% %     180/pi* alphaVector2D(indDirection1D,1)), ...
% %     180/pi* asin(alphaVector2D(indDirection1D,3)), ...
% %     boundary1D(indDirection1D),'.'); title('Boundary attenuation');
% % xlabel('longitude'); ylabel('latitude'); zlabel('attenuation'); view([1 0 0]);
% % figure; plot3k(xVector2D, 'ColorData', boundary1D(indDirection1D), ...
% %     'ColorRange', [-.1 1.1], 'Marker', {'o',10}); title('Boundary attenuation');
% % xlabel('x'); ylabel('y'); zlabel('x'); pause


%% 9. TIME STEP USING EXPLICIT RK4.

rhsOperator = @(currentSolution2D) ( dXMatrix2D * currentSolution2D + ...
    dissipation2D * currentSolution2D );

waveAction2D = zeros(nAlpha,nTime); waveAction2D(:,1) = initialWaveAction1D;
waveAction2D(:,1) = boundary1D .* waveAction2D(:,1);

for indTime=2:nTime
    stage1 = deltaTime * rhsOperator( waveAction2D(:,indTime-1) );
    stage2 = deltaTime * rhsOperator( waveAction2D(:,indTime-1) + 0.5*stage1 );
    stage3 = deltaTime * rhsOperator( waveAction2D(:,indTime-1) + 0.5*stage2 );
    stage4 = deltaTime * rhsOperator( waveAction2D(:,indTime-1) + stage3 );

    waveAction2D(:,indTime) = waveAction2D(:,indTime-1) + 1/6 * ...
                (stage1 + 2*stage2 + 2*stage3 + stage4);
    waveAction2D(:,indTime) = boundary1D .* waveAction2D(:,indTime);
```

```
end
toc


%% 10. SAVE OUTPUT.

runSettings = struct('initialWidth0D', initialWidth0D, ...
    'boundaryWidth0D', boundaryWidth0D, ...
    'initialLongitude0D', initialLongitude0D,  ...
    'initialLatitude0D', initialLatitude0D, ...
    'initialDirection0D', initialDirection0D, ...
    'initialSpread0D', initialSpread0D, ...
    'groupVelocity0D', groupVelocity0D, 'gamma0D', gamma0D, ...
    'damping0D', damping0D, 'deltaTime', deltaTime, 'nTime', nTime);

FID_SAVE = [ PATH.WORK 'waveAction2D_B_Half_Exp_' num2str(nXApprox) 'x' ...
    num2str(nXLocal) '_' fnc_integer_to_string(nDir,2) 'd' num2str(nKLocal) ...
    '_h' num2str(laplacianOrder0D) strGamma0D '_e' num2str(epsilon0D-mod(epsilon0D,1)) ...
    'pt' num2str(10*mod(epsilon0D,1)) ...
    '_i' fnc_integer_to_string(int8(initialDirection0D*180/pi),3) ...
    '_s' num2str(round(initialSpread0D*180/pi)) '_a' num2str(round(timeScale0D*10)) ...
    '_b' num2str(ceil(boundaryWidth0D/pi*180))]

save(FID_SAVE, 'xVector2D', 'dirVector1D', 'kVector2D', 'alphaVector2D', ...
    'nodeSettings', 'stencilSettings1D', 'runSettings', 'waveAction2D')

totalMinutes = toc/60
```

```
function attenuation1D = fnc_boundary_attenuation_gaussian( alphaVector2D, ...
    width0D )

```

```
%% CREATE POLAR BOUNDARIES:
% This function generates an attenuation filter for the polar boundaries in the
% sphere-ring geometry. Created by Adrean Webb.
%
% Requires: Nothing.


%% 1. INITIALIZE.

nAlpha = length(alphaVector2D);
attenuation1D = zeros(nAlpha,1);


%% 2. CONVERT TO GEOPHYSICAL COORDINATES.

lambdaVector1D = atan2(alphaVector2D(:,2),alphaVector2D(:,1));
phiVector1D = asin(alphaVector2D(:,3));


%% 3. CREATE RHO VECTORS.

northRhoVector1D = acos( sin(phiVector1D) );
southRhoVector1D = acos( - sin(phiVector1D) );


%% 4. CREATE GAUSSIAN BELL WITH SPECIFIED WIDTH.

northExponentialTerm1D = 9/2/ width0D* northRhoVector1D;
southExponentialTerm1D = 9/2/ width0D* southRhoVector1D;

northGaussianBell1D = exp( - northExponentialTerm1D.^2 );
southGaussianBell1D = exp( - southExponentialTerm1D.^2 );

```

```

%% 5. CREATE ATTENUATION VECTOR.

northNorm0D = max(northGaussianBell1D(alphaVector2D(:,3) > 0));
southNorm0D = max(southGaussianBell1D(alphaVector2D(:,3) < 0));

attenuation1D(alphaVector2D(:,3)>0) = ...
    northGaussianBell1D(alphaVector2D(:,3)>0) / northNorm0D * 4/3;
attenuation1D(alphaVector2D(:,3)<0) = ...
    southGaussianBell1D(alphaVector2D(:,3)<0) / southNorm0D * 4/3;

attenuation1D = 1 - attenuation1D;
attenuation1D(attenuation1D<0) = 0;


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [nodeMatrix, nNode] = fnc_node_generate_spherical(nApproxNode, radius)

%% SPHERICAL NODE GENERATOR:
% Creates maximal determinant (MD) nodes (on a sphere) with a specifed radius in
% Cartesian coordinates based on the approximate number of nodes needed.
%
% Requires: 'kdtree' package, MD node data, and 'model_settings.mat' file.
%
% function [NODEMATRIX, NNODE] = node_generate_spherical_md(NAPPROXNODE, RADIUS)
%
% IN:   'NAPPROXNODE'   Approximate number of nodes to use.
%       'RADIUS'        Radius of the sphere. Default is '1'.
%
% OUT:  'NODEMATRIX'    Nodes returned in Cartesian coordinate form as
%                        [x1Vector x2Vector x3Vector].
%       'NNODE'         Number of actual nodes used (optional)
%
% NOTES:    Path to MD node data directory is set in file
%            'model_settings_initialize.m'.


%% 1. CHECK INPUT.

if exist('radius') ~= 1; radius = 1; end


%% 2. DETERMINE CORRECT FILE TO LOAD.

% POLYNOMIAL DEGREE USED TO SOLVE THE MAXIMUM DETERMINANT
polyDegree = ceil(sqrt(nApproxNode)) - 1;
polyDegreeString = sprintf('%03d',polyDegree);

nNode = (polyDegree + 1)^2;
nNodeString = sprintf('%05d', nNode);


%% 3. LOAD DATA.

load('model_settings', 'PATH')
fileDataString = [ PATH.DATA_MD 'md' polyDegreeString '.' nNodeString ]
fidData = fopen(fileDataString,'r');

textBody = textscan(fidData,'%f %f %f %f',nNode);

x1Vector = textBody{1}; x2Vector = textBody{2}; x3Vector = textBody{3};
xMatrix = [x1Vector x2Vector x3Vector];

```

```
fclose(fidData);


%% 4. DEFINE OUTPUT.

nodeMatrix = radius * xMatrix;


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function initialConditions1D = fnc_initial_conditions_gaussian( ...
    alphaVector2D, nDir, width0D, centerLongitude0D, centerLatitude0D, ...
    initialDirection0D, initialSpread0D )

%% CREATE INITIAL CONDITIONS:
% This function generates spatial and spectral Gaussian bell initial conditions
% for the sphere—ring geometry. Created by Adrean Webb.
%
% Requires: Nothing.


%% 1. INITIALIZE.

nAlpha = length(alphaVector2D); nX = nAlpha / nDir;
initialConditions1D = zeros(nAlpha,1);
deltaDir = 2*pi / nDir; dirVector1D = [0:deltaDir:2*pi—deltaDir]';

% CONVERT TO GEOPHYSICAL COORDINATES.
lambdaVector1D = atan2(alphaVector2D(:,2),alphaVector2D(:,1));
phiVector1D = asin(alphaVector2D(:,3));

```

```
% CREATE RHO VECTOR.
rhoVector1D = acos( sin(centerLatitude0D) * sin(phiVector1D) + ...
    cos(centerLatitude0D) * cos(phiVector1D) .* ...
    cos( lambdaVector1D - centerLongitude0D * ones(nAlpha,1)) ) ;


%% 2. CREATE GAUSSIAN BELL WITH SPECIFIED CENTER AND WIDTH.

exponentialTermAmplitude1D = 9/2/ width0D * rhoVector1D ;
gaussianBellAmplitude1D = exp( - exponentialTermAmplitude1D.^2 );


%% 3. CREATE GAUSSIAN BELL IN SPECTRAL DOMAIN.

spreadVector1D = dirVector1D - initialDirection0D * ones(nDir,1);
spreadVector1D(spreadVector1D < -pi) = ...
    spreadVector1D(spreadVector1D < -pi) + 2*pi;
spreadVector1D(spreadVector1D > pi) = ...
    spreadVector1D(spreadVector1D > pi) - 2*pi;

exponentialTermSpread1D = 9/2/ initialSpread0D * spreadVector1D ;
gaussianBellSpread1D = exp( - exponentialTermSpread1D.^2 );

%% 4. COMBINE GAUSSIAN BELLS.

for ind=1:nX
    indSpread = ((nDir*(ind-1)+1):nDir*ind)';
    initialConditions1D(indSpread,1) = ...
        gaussianBellAmplitude1D(indSpread,1) .* gaussianBellSpread1D;
end


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [distance] = fnc_norm_great_circle(node1, node2)

%% GREAT—CIRCLE DISTANCE:
% Calculates the shortest distance on a sphere between any two nodes. Nodes are
% in Cartesian [x y z] form.
%
% Requires: nothing.
%
% function [DISTANCE] = norm_great_circle(NODE1, NODE2)
%
% IN:   'NODE*'         A matrix of Cartesian points with size(NODE*) = (N,3).
%
% OUT:  'DISTANCE'      The shortest great—circle distance between two Cartesian
%                       node matrices with size(DISTANCE) = (N,1).


%% 1. CALCULATE DISTANCE.

crossProduct = cross(node1,node2);
crossProductMagnitude = sqrt(sum(crossProduct.^2,2));
dotProduct = sum(node1.*node2,2);

distance = atan2(crossProductMagnitude,dotProduct);


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [strPad] = fnc_integer_to_string( integerValue, nChar )
```

```

%% CONVERT INTEGER TO PADDED STRING:
% This function saves the hassle of remembering the necessary format for
% converting an integer into a padded string for file loading and saving.
% Created by Adrean Webb.
%
% Requires: Nothing.
%
% function [strPad] = fnc_integer_to_string( integerValue, nChar )
%
% In:   'integerValue'  Integer value to be checked/converted.
%       'nChar'         Number of characters for final string.
%
% Out:  'strPad'        Numeric string padded with zeros.


%% 1. CHECK INPUT ARGUMENTS.

if (mod(integerValue,1) ~= 0)
    error('Input must be an integer value.')
elseif (integerValue >= 10^nChar)
    error('Input is larger than output.')
end


%% 2. CONVERT TO STRING AND PAD WITH ZEROS IF NECESSARY.

strFormat = ['%0' num2str(nChar) '.0f'];
strPad = num2str(integerValue,strFormat);


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function sparse2D = fnc_rbffd_sphere_ring( xVector2D, kVector2D, ...
    alphaVector2D, stencilSettings1D )

%% CREATE RBF DIFFERENTIAL MATRIX FOR RING–POINT GEOMETRY.
% This function generates the differential matrix for the wave action balance
% equation on the ring–point geometry.  Created by Adrean Webb.
%
% Requires: 'kD–tree' package.
%
% function [DXMATRIX2D, ACONDITION0D] = FNC_RBF_RING_POINT( XVECTOR2D, ...
%       RSQUARED2D, EPSILON, GROUPVELOCITY )
%
% IN:   'XVECTOR2D'         2D spatial nodes.
%                             Given as radius * [ cos(theta) sin(theta) ].
%       'EPSILON'           The shape parameter value.
%       'GROUPVELOCITY'     The group velocity for propagation.
%
% OUT:  'DXMATRIX2D'        The combined dX and dY differential matrix.
%       'ACONDITION0D'      The condition number of the 'A Matrix' to invert.


%% 1. CHECK INPUT AND DEFINE VARIABLES.

epsilon0D = stencilSettings1D(1); nXLocal = stencilSettings1D(2);
nKLocal = stencilSettings1D(3); laplacianOrder0D = stencilSettings1D(4);
laplacianDimension0D = stencilSettings1D(5);

nX = size(xVector2D,1); nK = size(kVector2D,1); nAlpha = size(alphaVector2D,1);
nAlphaLocal = nXLocal * nKLocal

if (nAlphaLocal > nAlpha)
    error('Stencil size must be smaller than number of nodes.')
```

```
end


%% 2. DEFINE RBFS.

rbf = @(epsilon,rSquared) exp(-epsilon^2 * rSquared);
dRbfBase = @(epsilon,rSquared) -2*epsilon^2 * exp(-epsilon^2 *rSquared);


%% 3. INITIALIZE SPARSE ARRAYS.

aMatrix2D = ones(nAlphaLocal+1,nAlphaLocal+1); aMatrix2D(end,end) = 0;
bVectorRow1D = zeros(1,nAlphaLocal+1);

sparseColumn1D = zeros(nAlpha*nAlphaLocal,1);
sparseRow1D = zeros(nAlpha*nAlphaLocal,1);

dX1Vector1D = zeros(nAlpha*nAlphaLocal,1);
dX2Vector1D = zeros(nAlpha*nAlphaLocal,1);
dX3Vector1D = zeros(nAlpha*nAlphaLocal,1);
dK1Vector1D = zeros(nAlpha*nAlphaLocal,1);
dK2Vector1D = zeros(nAlpha*nAlphaLocal,1);
dissipation1D = zeros(nAlpha*nAlphaLocal,1);

aCondition1D = zeros(nAlpha,1);


%% 4. CREATE INDIVIDUAL STENCILS.

nearestKHalf = (nKLocal - 1)/2;
treeRoot = kdtree_build(xVector2D);

for indX=1:nX
```

```

% FIND NEAREST NEIGHBORS.
    nearestXIndex = kdtree_k_nearest_neighbors( treeRoot, ...
        xVector2D(indX,:)', nXLocal );
    nearestXIndex = nearestXIndex( nXLocal:-1:1 );

    for indK=1:nK
        nearestKIndex = (indK:(indK + nearestKHalf))';
        nearestKIndex = [nearestKIndex;[(indK - nearestKHalf):(indK - 1)]'];
        nearestKIndex(nearestKIndex<1) = nearestKIndex(nearestKIndex<1) + nK;
        nearestKIndex(nearestKIndex>nK) = nearestKIndex(nearestKIndex>nK) - nK;

        nearestAlphaIndex = [];
        for indXLocal=1:nXLocal
            nearestAlphaIndex = [nearestAlphaIndex; ...
                (nK*(nearestXIndex(indXLocal) - 1)*ones(nKLocal,1) + ...
                nearestKIndex) ];
        end

        indAlpha = nK*(indX - 1) + indK;
        sparseIndex = (((indAlpha-1)* nAlphaLocal + 1):(indAlpha* nAlphaLocal))';
        sparseColumn1D(sparseIndex) = indAlpha;
        sparseRow1D(sparseIndex) = nearestAlphaIndex;

% CREATE LOCAL X VECTOR.
        alphaVectorLocal2D = alphaVector2D(nearestAlphaIndex,:);
        alphaStencil1D = alphaVectorLocal2D(1,:)';

% CREATE LOCAL DIFFERENCE MATRICES.
        [x1Vertical2D, x1Horizontal2D] = ndgrid( alphaVectorLocal2D(:,1), ...
            alphaVectorLocal2D(:,1) );
        [x2Vertical2D, x2Horizontal2D] = ndgrid( alphaVectorLocal2D(:,2), ...
            alphaVectorLocal2D(:,2) );
        [x3Vertical2D, x3Horizontal2D] = ndgrid( alphaVectorLocal2D(:,3), ...
            alphaVectorLocal2D(:,3) );
        [k1Vertical2D, k1Horizontal2D] = ndgrid( alphaVectorLocal2D(:,4), ...
            alphaVectorLocal2D(:,4) );
        [k2Vertical2D, k2Horizontal2D] = ndgrid( alphaVectorLocal2D(:,5), ...
            alphaVectorLocal2D(:,5) );

% CALCULATE RADIAL DISTANCE FOR THE RBF DIFFERENCE MATRIX.
        rSquared2D = (x1Vertical2D — x1Horizontal2D).^2 + ...
            (x2Vertical2D — x2Horizontal2D).^2 + ...
            (x3Vertical2D — x3Horizontal2D).^2 + ...
            (k1Vertical2D — k1Horizontal2D).^2 + ...
            (k2Vertical2D — k2Horizontal2D).^2;
        rSquaredRow1D = rSquared2D(1,:);

        x1VerticalRow1D = x1Vertical2D(1,:); x1HorizontalRow1D = ...
            x1Horizontal2D(1,:);
        x2VerticalRow1D = x2Vertical2D(1,:); x2HorizontalRow1D = ...
            x2Horizontal2D(1,:);
        x3VerticalRow1D = x3Vertical2D(1,:); x3HorizontalRow1D = ...
            x3Horizontal2D(1,:);
        k1VerticalRow1D = k1Vertical2D(1,:); k1HorizontalRow1D = ...
            k1Horizontal2D(1,:);
        k2VerticalRow1D = k2Vertical2D(1,:); k2HorizontalRow1D = ...
            k2Horizontal2D(1,:);

        clear x1Vertical2D x2Vertical2D x3Vertical2D k1Vertical2D ...
            k2Vertical2D x1Horizontal2D x2Horizontal2D x3Horizontal2D ...
            k1Horizontal2D k2Horizontal2D

% CALCULATE NORM FOR DIFFERENTIATION VECTOR.
        normXY0D = sqrt( alphaStencil1D(1)^2 + alphaStencil1D(2)^2 );

```

```
% CALCULATE DIFFERENTIATION VECTORS.
        aMatrix2D(1:nAlphaLocal,1:nAlphaLocal) = rbf(epsilon0D,rSquared2D);
        [lowerAMatrix2D,upperAMatrix2D,permutationAMatrix2D] = lu(aMatrix2D);

        bVectorRow1D(1,1:nAlphaLocal) = ...
            ( alphaStencil1D(2) * alphaStencil1D(4) + ...
            alphaStencil1D(1) * alphaStencil1D(3) * alphaStencil1D(5) ) / ...
            normXY0D * x1HorizontalRow1D ;
        bVectorRow1D(1,1:nAlphaLocal) = bVectorRow1D(1,1:nAlphaLocal) .* ...
            dRbfBase(epsilon0D,rSquaredRow1D);
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dX1Vector1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

        bVectorRow1D(1,1:nAlphaLocal) = ...
            − ( alphaStencil1D(1) * alphaStencil1D(4) − ...
            alphaStencil1D(2) * alphaStencil1D(3) * alphaStencil1D(5) ) / ...
            normXY0D * x2HorizontalRow1D ;
        bVectorRow1D(1,1:nAlphaLocal) = bVectorRow1D(1,1:nAlphaLocal) .* ...
            dRbfBase(epsilon0D,rSquaredRow1D);
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dX2Vector1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

        bVectorRow1D(1,1:nAlphaLocal) = − alphaStencil1D(5) * normXY0D * ...
            x3HorizontalRow1D ;
        bVectorRow1D(1,1:nAlphaLocal) = bVectorRow1D(1,1:nAlphaLocal) .* ...
            dRbfBase(epsilon0D,rSquaredRow1D);
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dX3Vector1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

        bVectorRow1D(1,1:nAlphaLocal) = ...
```

```
            — alphaStencil1D(3) * alphaStencil1D(4) * alphaStencil1D(5) / ...
            normXY0D * k1HorizontalRow1D;
        bVectorRow1D(1,1:nAlphaLocal) = bVectorRow1D(1,1:nAlphaLocal) .* ...
            dRbfBase(epsilon0D,rSquaredRow1D);
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dK1Vector1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

        bVectorRow1D(1,1:nAlphaLocal) = ...
            alphaStencil1D(3) * alphaStencil1D(4)^2 / ...
            normXY0D * k2HorizontalRow1D ;
        bVectorRow1D(1,1:nAlphaLocal) = bVectorRow1D(1,1:nAlphaLocal) .* ...
            dRbfBase(epsilon0D,rSquaredRow1D);
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dK2Vector1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

% CALCULATE DISSIPATION VECTOR.
        bVectorRow1D(1:nAlphaLocal) = epsilon0D^(2*laplacianOrder0D) * ...
            fnc_hyperviscosity( epsilon0D^2 * rSquaredRow1D', ...
            laplacianOrder0D, laplacianDimension0D) .* ...
            exp( —epsilon0D^2 * rSquaredRow1D )';
        weight1D = upperAMatrix2D \ ( lowerAMatrix2D \ ...
            ( permutationAMatrix2D * bVectorRow1D' ) );
        dissipation1D(sparseIndex) = weight1D(1:nAlphaLocal,1);

        aCondition1D(indAlpha,1) = cond(aMatrix2D);
    end
end


%% 5. ASSEMBLE INDIVIDUAL STENCILS INTO SPARSE MATRICES.

```

```
sparse2D.dX1 = sparse(sparseColumn1D, sparseRow1D, dX1Vector1D, nAlpha, nAlpha);
sparse2D.dX2 = sparse(sparseColumn1D, sparseRow1D, dX2Vector1D, nAlpha, nAlpha);
sparse2D.dX3 = sparse(sparseColumn1D, sparseRow1D, dX3Vector1D, nAlpha, nAlpha);
sparse2D.dK1 = sparse(sparseColumn1D, sparseRow1D, dK1Vector1D, nAlpha, nAlpha);
sparse2D.dK2 = sparse(sparseColumn1D, sparseRow1D, dK2Vector1D, nAlpha, nAlpha);

sparse2D.dissipation = sparse(sparseColumn1D, sparseRow1D, dissipation1D, ...
    nAlpha, nAlpha);

sparse2D.condition0D = mean(aCondition1D);


%%%%% END OF FUNCTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% ENVIRONMENT INITIALIZATION: model_settings_initialize.m
% Defines environment necessary to run scripts. Data is saved in
% 'model_settings.mat' for quick matlab reference. Created by Adrean Webb.


%% 1. CONSTANTS.

G = 9.81;  % gravity (m/s2)
RADIUS_EARTH = 6.37e6; % (M)


%% 2. FILE PATHS.

PATH.MAIN = '/Users/adrean/Research/Projects/RBF_Spectral_Wave_Model/';
PATH.DATA_MD = [ PATH.MAIN 'Data/MD_Nodes/'];

PATH.EXEC = [ PATH.MAIN 'Matlab/Dissertation_Code/' ];
PATH.FIGURE = [ PATH.EXEC '[Figures]/' ];
PATH.WORK = [ PATH.EXEC '[Work]/' ];
PATH.STENCIL = [ PATH.EXEC '[Stencils]/' ];


%% 3. SAVE SETTINGS.

save model_settings
```

```
%% PLOT RBF—FD WAVE MODEL OUTPUT: plot_spectra_gaussian_error_initial_direction.m
% This sample program plots output from the RBF—FD wave model.
% Created by Adrean Webb.


%% 1. LOAD WAVE ACTION DATA.

clear all; close all
model_settings_initialize;

initialDirectionDeg0D = 30

strHyper = 'h4n1p8';
strAValue = 'a2';
strX = '3600x17';
strK = '36d9';
strEpsilon = 'e3pt0';
strSpread = 's60';
strBoundary = 'b30';

strInitialDirection = [ 'i' fnc_integer_to_string(initialDirectionDeg0D,3) ];
initialDirectionBin0D = initialDirectionDeg0D/10 + 1;
```

```

FID_DATA = ['[Work]/waveAction2D_B_Half_Exp_' strX '_' strK '_' strHyper '_' ...
    strEpsilon '_' strInitialDirection '_' strSpread '_' strAValue '_' ...
    strBoundary '.mat']
load(FID_DATA)

nAlpha = size(alphaVector2D,1); nX = size(xVector2D,1);
nDir = size(dirVector1D,1); nK = size(kVector2D,1);


%% 2. LOAD RUN SETTINGS.

initialWidth0D = runSettings.initialWidth0D;
initialLongitude0D = runSettings.initialLongitude0D;
initialLatitude0D = runSettings.initialLatitude0D;
initialDirection0D = runSettings.initialDirection0D;
initialSpread0D = runSettings.initialSpread0D;
groupVelocity0D = runSettings.groupVelocity0D;
deltaTime = runSettings.deltaTime;
nTime = runSettings.nTime;


%% 3. DEFINE WAVE ACTION.

waveAction2D(:,nTime+1) = zeros(nAlpha,1);
waveAction2D(:,2:nTime+1) = waveAction2D(:,1:nTime);


%% 4. CREATE INITIAL CONDITIONS.

finalLongitude0D = initialLongitude0D + pi;
finalLatitude0D = initialLatitude0D;
finalDirection0D = -initialDirection0D;
```

```
finalDirectionBin0D = mod(nDir + 1 - initialDirectionBin0D,nDir) + 1;

analyticWaveAction1D = fnc_initial_conditions_gaussian( alphaVector2D, nDir, ...
    initialWidth0D, finalLongitude0D, finalLatitude0D, ...
    finalDirection0D, initialSpread0D );


%% 5. CALCULATE ERROR.

residual1D = waveAction2D(:,end) - analyticWaveAction1D;

for tt=1:nX
    indDir = nDir*(tt-1)+1:nDir*tt;
    analyticNormInf1D(tt,1) = max( abs( analyticWaveAction1D(indDir) ) );
    analyticNormTwo1D(tt,1) = sum( analyticWaveAction1D(indDir).^2 );
    residualInf1D(tt,1) = max( abs(residual1D(indDir)) );
    residualTwo1D(tt,1) = sum( residual1D(indDir).^2 );
end

relativeErrorInf1D = residualInf1D ./ max(analyticNormInf1D);
relativeErrorTwo1D = residualTwo1D ./ max(analyticNormTwo1D);

max(relativeErrorTwo1D)*10000
colorScaleTwo0D = ceil(max(relativeErrorTwo1D)*10000);

handleFigure1 = figure; strLegend = [];
set(gcf, 'Position', [500 400 600 400], 'Color', 'w') % [l b w h]


%% 6. CREATE COLORMAP.

redratio = 0.6; blueratio = 0.9; greenratio = 0.9;
nRows = 64; cmap =zeros(nRows,3);
```

```
greenness = 0; redness = 0; blueness = 0; stepsize = 1/(nRows-1);
for i=2:nRows
    greenness = greenness + stepsize*greenratio;  cmap(i,2) = greenness;
    redness = redness + stepsize*redratio; cmap(i,[1]) = redness;
    blueness = blueness + stepsize*blueratio; cmap(i,[3]) = blueness;
end
cmap = cmap(end:-1:1,:);
colormap(cmap)


%% 7. PLOT AND SAVE OUTPUT.

plot3k(xVector2D, 'ColorData', relativeErrorTwo1D*10000, ...
'ColorRange', [0 colorScaleTwo0D], 'CBLabels', 5, 'Marker', {'o',12}, 'Labels',{'','','','','
10**(-4)'});
axis equal; view([-1 0 0])

xlabel('x', 'interpreter', 'latex');
ylabel('y', 'interpreter', 'latex');
zlabel('z', 'interpreter', 'latex');

set(gca,'XTick',[-1 -0.5 0 0.5 1])
set(gca,'YTick',[-1 -0.5 0 0.5 1])
set(gca,'ZTick',[-1 -0.5 0 0.5 1])

title([ 'Initial direction = ' num2str(initialDirectionDeg0D) '{\circ}' ]);

set(gca,'FontSize',18, 'FontName', 'Times New Roman');
set(findall(gcf,'type','text'),'FontSize',18, 'FontName', 'Times New Roman');

FID_SAVE = ['[Figures]/Ch4_Case5_Error_B_' strX '_' strK '_' strHyper '_' ...
    strEpsilon '_' strInitialDirection '_' strSpread '_' strAValue '_' ...
    strBoundary '.png']
```

```

export_fig( FID_SAVE, '-m2')
```