Numerics of MITgcm: The 1-D Wave Equation

224 Final Project
Alexander Andriatis

1st June 2020

Introduction

The goal this project is to learn about the numerics of MITgcm. Specifically, I look at the dynamics of the one-dimensional wave equation. A simple numerical solution for the simple second-order wave equation is compared an MITgcm-style solution of the shallow water equations. The pressure method of MITgcm with Adams-Bashforth time stepping and finite difference spatial discretization is used to solve the shallow water equations with a linearized free surface. Simple initial conditions such as a Gaussian pulse and a Gaussian wave packet are simulated and compared.

1 1-D Wave Equation

I begin the investigation of waves in models by considering the simplest case, the 1-dimensional wave equation. I would like to note that while the following work is my own, I am not the first to have a go at this problem; the same material is nicely presented here [Munster, 2020]. The 1-dimensional wave equation is given by

$$\frac{\partial^2 \phi}{\partial t^2} - c^2 \nabla^2 \phi = 0, \tag{1}$$

where ϕ is amplitude and c is the propagation speed of a plane wave,

$$\phi = e^{i(kx - \omega t)}, \tag{2}$$

which obeys the dispersion relationship

$$\omega^2 = c^2 k^2. \tag{3}$$

To discretize 1 in space, I use the usual centered difference

$$\left(\frac{\partial^2 \phi}{\partial x^2} \right)_k = \frac{\phi_{k+1} - 2\phi_k + \phi_{k-1}}{\Delta x^2}. \tag{4}$$

Similarly, centered difference is used for time discretization

$$\left(\frac{\partial^2 \phi}{\partial t^2} \right)^n = \frac{\phi^{n+1} - 2\phi^n + \phi^{n-1}}{\Delta t^2}. \tag{5}$$

The resulting discrete differential wave equation,

$$\frac{\phi_k^{n+1} - 2\phi_k^n + \phi_k^{n-1}}{\Delta t^2} - c^2 \frac{\phi_{k+1}^n - 2\phi_k^n + \phi_{k-1}^n}{\Delta x^2} = 0, \tag{6}$$

is solved for ϕ_k^{n+1} giving

$$\phi_k^{n+1} = 2\phi_k^n - \phi_k^{n-1} + \sigma^2 \left(\phi_{k+1}^n - 2\phi_k^n + \phi_{k-1}^n\right), \tag{7}$$

where $\sigma^2 \equiv \frac{c^2 \Delta t^2}{\Delta x^2}$.

To choose appropriate values of Δt and Δx in the numerical evaluation, a von Neumann stability analysis is performed. Representing $\phi(t, x)$ as a superposition of Fourier components,

$$\phi_k^n = A\varrho^n e^{imk\Delta x}, \quad \text{where} \quad \varrho \equiv e^{-i\omega\Delta t}, \tag{8}$$

equation 7 reduces, after some trig identities, to

$$\varrho = 2 - \varrho^{-1} - 4\sigma^2 \sin^2\left(\frac{m\Delta x}{2}\right). \tag{9}$$

This quadratic equation,

$$\varrho^2 - \alpha\varrho + 1 = 0, \quad \text{where} \quad \alpha \equiv 2 - 4\sigma^2 \sin^2\left(\frac{m\Delta x}{2}\right), \tag{10}$$

can be solved for ϱ,

$$\varrho = \frac{\alpha \pm \sqrt{\alpha^2 - 4}}{2}, \tag{11}$$

where the system is stable for $|\varrho| \leq 1$. This condition does not hold for $|\alpha > 2|$, leaving

$$\varrho = \frac{\alpha}{2} \pm \frac{i\sqrt{4 - \alpha^2}}{2}, \tag{12}$$

for which

$$|\varrho| = \sqrt{\frac{\alpha^2}{4} + \frac{4 - \alpha^2}{4}} = 1. \tag{13}$$

The condition for stability is therefore $|\alpha| \leq 2$, or

$$0 \leq 1 - \sigma^2 \sin^2\left(\frac{m\Delta x}{2}\right) \leq 1, \tag{14}$$

which is satisfied for $\sigma^2 \leq 1$. The numerical solution will therefore be stable as long as

$$\frac{c\Delta t}{\Delta x} \leq 1, \tag{15}$$

which is consistent with the CFL stability criteria for explicit numerical solvers.

To numerically solve the equation, I need initial amplitude $\phi(0, x)$ and rate of change in amplitude $\dot{\phi}(0, x)$, along with boundary conditions $\phi(t, -L)$ and $\phi(t, L)$. I choose to keep the variables dimensional to develop intuition about the behavior of the solution. Before I go any further, for computational simplicity, I'll express 7 in matrix notation,

$$\vec{\phi}^{n+1} = \left(2\boldsymbol{I} + \sigma^2 \boldsymbol{M}_{\mathrm{DD}}\right) \vec{\phi}^n - \vec{\phi}^{n-1}, \tag{16}$$

where

$$\vec{\phi}^n = \begin{bmatrix} \phi_1^n \\ \phi_2^n \\ \vdots \\ \phi_K^n \end{bmatrix}, \tag{17}$$

$\boldsymbol{I}$ and $\boldsymbol{M}_{\mathrm{DD}}$ are $(K \times K)$ matrices, with $\boldsymbol{I}$ the identity matrix and

$$\boldsymbol{M}_{\mathrm{DD}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{18}$$

for Dirichlet boundary conditions $\phi(t, -L) = \phi(t, L) = 0$.

For an initial condition, let's choose a classic example from physics of a wave group in a Gaussian envelope, such that the group is localized in space

$$\phi(0, x) = e^{-x^2/L_0^2} \cos(k_0 x). \tag{19}$$

where L_0 is the characteristic width of the packet. The waves will have an initial velocity such that the packet travels towards positive x,

$$\dot{\phi}(0, x) = \left(2cL_0^{-2} x \cos(k_0 x) + \omega_0 \sin(k_0 x)\right) e^{-x^2/L_0^2}, \tag{20}$$

where $\omega_0 = ck_0$. To implement the initial condition, time rate of change in amplitude at time zero is described through a Leapfrog time difference using an initial ghost time ($k = 0$),

$$\dot{\phi}_k^1 = \frac{\phi_k^2 - \phi_k^0}{2\Delta t} \implies \phi_k^0 = \phi_k^2 - 2\Delta t \dot{\phi}_k^1, \tag{21}$$

giving the first two timesteps of

$$\phi_k^1 = e^{-x_k^2/L_0^2} \cos(k_0 x_k), \tag{22}$$

$$\dot{\phi}_k^1 = \left(2cL_0^{-2} x_k \cos(k_0 x_k) + \omega_0 \sin(k_0 x_k)\right) e^{-x_k^2/L_0^2}, \tag{23}$$

$$\phi_k^2 = \phi_k^1 + \Delta t \dot{\phi}_k^1 + \frac{1}{2}\sigma^2 \left(\phi_{k+1}^1 - 2\phi_k^1 + \phi_{k-1}^1\right). \tag{24}$$

The carrier wave example in physics is interesting when the wave packet is much larger than a wavelength, $L_0 \gg 2\pi k^{-1}$, so for this exercise we'll set $L_0 = \frac{L}{10}$ and $k_0 = \frac{20\pi}{L_0}$. For the other parameters in the evaluation, I set $L = 200\pi$ and $c = 1$, such that $k_0 = 1$ and $\omega_0 = 1$. For sufficient spatial resolution, I estimate that the carrier wave needs to have around 20 points per wavelength, giving $\Delta x = \frac{\pi}{10}$. To run at the limit of CFL stability, $\Delta t = \frac{\pi}{10}$.

Snapshots of the solution at times $t = \left[0, \frac{L}{2}, L, 3L\right]$ are shown in different colors in Figure 1. An animation to the solution for $t = [0, 4L]$ can be accessed here. The wave packet can be seen propagating to the right with speed c, increasing to a height of twice the initial amplitude on reaching the solid boundary on the right, and reflecting back to the left. A small pulse can be seen traveling in the opposite direction of the wave packet due to the truncation error arising from the Leapfrog implementation of the initial condition in 21, which omits terms of $\mathcal{O}\left(\Delta t^2\right)$ and higher.

As another example, let's change the boundary conditions on the right side to a Neumann boundary condition,

$$\frac{\partial \phi}{\partial x}(t, L) = 0. \tag{25}$$

To implement the boundary, a ghost point is added to the right of the domain at $x = L + \Delta x$, where the spatial index $k = K + 1$. The spatial rate of change in amplitude at the boundary is described using a Leapfrog spatial difference as

$$\phi'^n_K = \frac{\phi_{K+1}^n - \phi_{K-1}^n}{2\Delta x} \implies \phi_{K+1}^n = 2\Delta x \phi'^n_K + \phi_{K-1}^n. \tag{26}$$

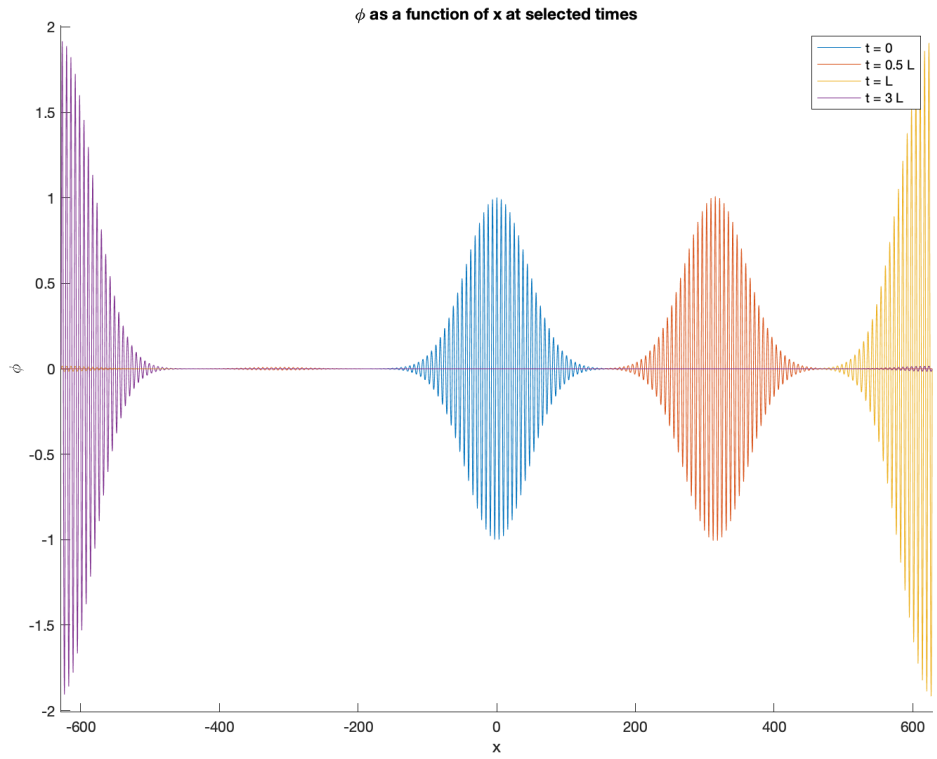

Figure 1: Snapshots of the numerical solution to 16 with initial conditions 19 and20.

This value is substituted into 7 at $k = K$ to find

$$\phi_K^{n+1} = 2\phi_K^n - \phi_K^{n-1} + \sigma^2 \left(2\Delta x {\phi'}_K^n - 2\phi_K^n + 2\phi_{K-1}^n\right). \tag{27}$$

For this example, ${\phi'}_K^n = 0$. It is easiest in the code to implement the boundary condition in a matrix, as in 18, giving

$$\boldsymbol{M}_{\mathrm{DN}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 2 & -2 \end{bmatrix} \tag{28}$$

which operates on the spatial vector of amplitudes in the equation

$$\vec{\phi}^{n+1} = \left(2\boldsymbol{I} + \sigma^2 \boldsymbol{M}_{\mathrm{DN}}\right)\vec{\phi}^n - \vec{\phi}^{n-1}. \tag{29}$$

For an initial condition, let's use a simple Gaussian pulse traveling to the right,

$$\phi_k^1 = e^{-x_k^2/L_0^2}, \tag{30}$$

$$\dot{\phi}_k^1 = 2cL_0^{-2}x_k\phi_k^1, \tag{31}$$

$$\phi_k^2 = \phi_k^1 + \Delta t\dot{\phi}_k^1 + \frac{1}{2}\sigma^2\left(\phi_{k+1}^1 - 2\phi_k^1 + \phi_{k-1}^1\right). \tag{32}$$

To evaluate, I keep the numerical parameters as before, but increase the spatial step to $\Delta x = \frac{L_0}{10} = 2\pi$ since I no longer need to resolve large-wavenumber oscillations. To run at the CFL limit, the timestep is also set to $\Delta t = 2\pi$. Snapshots of the solution at times $t = [0, \frac{1}{2}L, L, \frac{5}{2}L, 3L, \frac{7}{2}L]$ are shown in different colors in Figure 2. An animation to the solution for $t = [0, 4L]$ can be accessed here. The Gaussian pulse starts centered on $x = 0$ and moves to the right with speed c, increasing to a height of twice the initial amplitude as it reflects off the Neumann boundary condition, then travels to the left where it hits the Dirichlet boundary, changing sign as it travels back to the right.

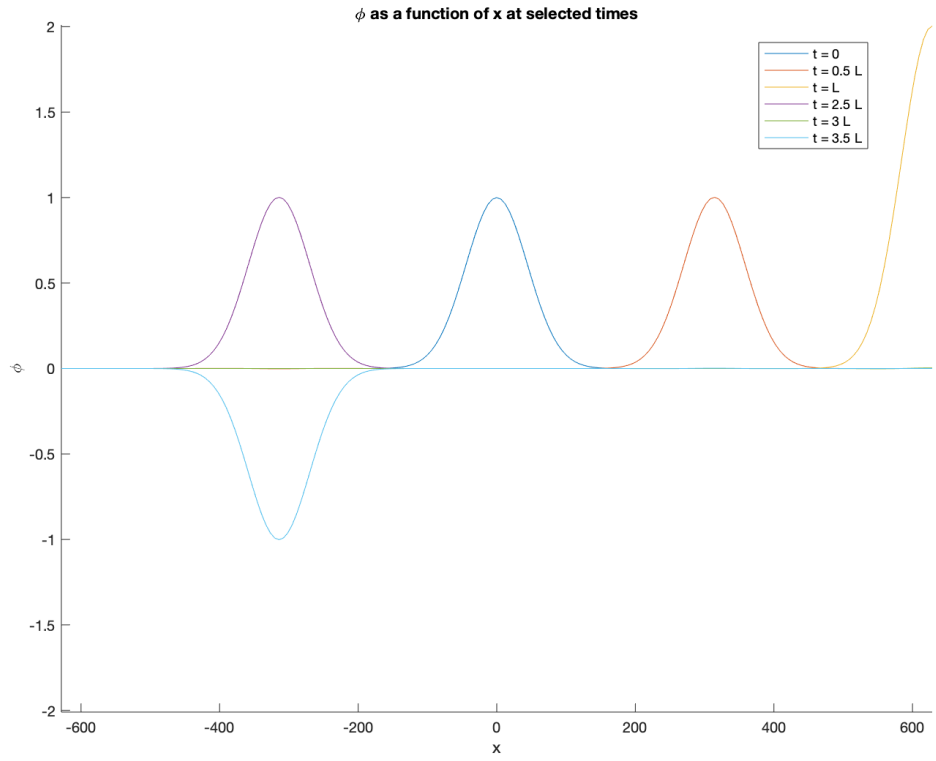

Figure 2: Snapshots of the numerical solution to 29 with initial conditions 30 and 31.

2 1-D Ocean Model

While solving the 1-D wave equation was informative and insightful to understanding the numerics of waves it doesn't help develop an understanding of MITgcm. MITgcm never solves the wave equation, either in 1-D or in all three. Instead, the model solves the governing equations of the ocean - momentum, continuity, the equation of state, temperature, and salinity:

$$\frac{D\vec{v}_h}{Dt} = -\frac{1}{\varrho_0}\nabla_h p - f\hat{k}\times\vec{v}_h + \nu\nabla^2\vec{v}_h, \tag{33}$$

$$0 = -\frac{1}{\varrho_0}\frac{\partial p}{\partial z} - \frac{g\varrho}{\varrho_0}, \tag{34}$$

$$\boldsymbol{\nabla}\cdot\vec{v} = 0, \tag{35}$$

$$\varrho = \varrho(\theta, S), \tag{36}$$

$$\frac{D\theta}{Dt} - Q_\theta, \tag{37}$$

$$\frac{DS}{Dt} = Q_S. \tag{38}$$

For simplicity, the equations here are hydrostatic incompressible Boussinesq equations. To begin to understand how MITgcm solves these equations, I seek to simplify the system as much as possible. Can I make a 1-D ocean model analogous to the above wave equation model? I would need to have an amplitude, a lateral dimension, and time. To reduce the number of dimensions, I turn to the shallow water equations from GFD. While pressure is a perfectly fine amplitude for our equation, I find it more intuitive to work with sea surface height η. I convert one to the other using hydrostatic balance

$$\int_p^{p_a} \mathrm{d}p = \int_z^\eta -\varrho_0 g\,\mathrm{d}z', \tag{39}$$

$$p(x,y,z) = \varrho_0 g\eta(x,y) - \varrho_0 g z + p_a, \tag{40}$$

using a constant density $\varrho = \varrho_0$, p_a is the pressure at the sea surface due to the atmosphere and z is defined positive upward starting at the sea floor $z = -h$. Next, the continuity equation is found by integrating 35

$$\int_{-h}^{\eta}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + w|_{z=\eta} - w|_{z=-h} = 0. \tag{41}$$

At the free surface, the boundary condition is

$$w|_{z=\eta} = \frac{D\eta}{Dt}, \tag{42}$$

while at the bottom,

$$w|_{z=-h} = \frac{D(-h)}{Dt}, \tag{43}$$

satisfying no flow through the bottom boundary. After some algebra (see my solutions to 211A HW 14, Problem 1) and linearizing the equation, which requires an assumption that the wave amplitude is small relative to bottom depth ($\frac{a}{h} \ll 1$), I get the continuity equation

$$\frac{\partial\eta}{\partial t} + \boldsymbol{\nabla}\cdot[h\vec{v}] = 0. \tag{44}$$

This form allows for a changing bottom depth, and therefore wave speed, $h = h(x,y)$. Substituting 40 into 33, replacing 35 with 44, and eliminating the y-coordinate leaves the 1-D equations of motion for sea surface

height $\eta(t,x)$:

$$\frac{\partial u}{\partial t} + g\frac{\partial \eta}{\partial x} = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{45}$$

$$\frac{\partial \eta}{\partial t} + \frac{\partial uh}{\partial x} = 0. \tag{46}$$

Equations 45 and 46, while simple, contain some interesting dynamics. First, it should be noted that they can be combined to give back the 1-D wave equation,

$$\frac{\partial^2 \eta}{\partial t^2} - \frac{\partial}{\partial x} hg \frac{\partial \eta}{\partial x} = \cdots . \tag{47}$$

The variables $u(t,x)$ and $\eta(t,x)$ vary in time and space while $h(x)$ varies in space. Equation 45 features nonlinear momentum advection and diffusion, and 46 describes the linearized free surface. The way I've chosen to write them is not happenstance - they are analogous to the equations for the pressure method in section 2.3 and 2.4 of the MITgcm documentation [Campin et al., 2019], reproduced here:

$$\partial_t u + g\partial_x \eta = G_u, \tag{48}$$

$$\partial_t \eta + \partial_x H\widehat{u} = \mathcal{P} - \mathcal{E} + \mathcal{R}. \tag{49}$$

G_u is a catch-all for all the terms in the momentum equation except the surface pressure gradient, $H\widehat{u}$ is the depth integral of u, which for this barotropic shallow water system is just hu, and the right-hand terms of 49 are fresh-water source terms that I neglect here. The discussion that follows is an attempt to numerically solve the simplified shallow-water model in equations 46 and 45 using the pressure method of MITgcm, keeping as true as possible to the numerical schemes used in the core MITgcm algorithm.

The MITgcm semi-implicit pressure method for hydrostatic equations with an implicit linear free surface, with variables co-located in time and with Adam-Bashforth time-stepping is given by

$$u^* = u^n + \Delta t G_u^{n+1/2}, \tag{50}$$

$$\eta^* = \eta^n - \Delta t \partial_x h u^*, \tag{51}$$

$$\partial_x \left(gh\partial_x \eta^{n+1} \right) - \frac{\eta^{n+1}}{\Delta t^2} = -\frac{\eta^*}{\Delta t^2}, \tag{52}$$

$$u^{n+1} = u^* - \Delta t g \partial_x \eta^{n+1}. \tag{53}$$

from equations 2.12, 2.14, 2.15, and 2.16 in MITgcm documentation [Campin et al., 2019]. Explicit terms in G are extrapolated forward in time to timestep $n+1/2$ using the Adams-Bashforth time stepping method, given by

$$G_\tau^{n+1/2} = (3/2 + \epsilon_{AB})G_\tau^n - (1/2 + \epsilon_{AB})G_\tau^{n-1}, \tag{54}$$

where τ is any prognostic variable and ϵ_{AB} is a small term that adds stability to the solution of oscillatory terms (MITgcm equation 2.19). For the system here, G_u contains two terms - momentum advection, G_u^{adv}, and horizontal dissipation G_u^{h-diss}. The momentum advection term, given by equation 2.90, is discretized using the finite volume method, which simplified for this model gives

$$\left(G_u^{adv}\right)_i = \frac{1}{4\Delta x}\left(u_{i+1}^2 + 2u_{i+1}u_i - 2u_{i-1}u_i - u_{i-1}^2\right). \tag{55}$$

The diffusion term, given in equation 2.108 and 2.110, ends up being simple centered difference

$$\left(G_u^{h-diss}\right)_i = \frac{\nu}{\Delta x^2}\left(u_{i+1} - 2u_i + u_{i-1}\right). \tag{56}$$

Together, 55 and 56 give the right hand side

$$\left(G_u\right)_i = -\left(G_u^{adv}\right)_i + \left(G_u^{h-diss}\right)_i, \tag{57}$$

which we can now plug into 54

$$(G_u)_i^{n+1/2} = (3/2 + \epsilon_{AB})(G_u)_i^n - (1/2 + \epsilon_{AB})(G_u)_i^{n-1}, \tag{58}$$

giving the fully-discretized form for equation 50,

$$u_i^* = u_i^n + \Delta t (G_u)_i^{n+1/2}. \tag{59}$$

The numerical step to calculate u^* requires the velocity field everywhere for the current and previous timestep. Next, to calculate the intermediate sea surface height η^*, I need the divergence of the vertically-integrated velocity hu. The documentation here is somewhat opaque on how the divergence is calculated, so I'll just use a centered difference:

$$(\partial_x hu^*)_i = \frac{1}{2\Delta x}\left(u_i^*(h_{i+1} - h_{i-1}) + h_i\left(u_{i+1}^* - u_{i-1}^*\right)\right), \tag{60}$$

giving the fully-discretized form of equation 51,

$$\eta_i^* = \eta_i^n - \Delta t (\partial_x hu^*)_i. \tag{61}$$

For the implicit free surface, the same centered differencing is first applied to the divergent term,

$$\left(\partial_x\left(gh\partial_x\eta^{n+1}\right)\right)_i = \frac{gh_i}{\Delta x^2}\left(\eta_{i+1}^{n+1} - 2\eta_i^{n+1} + \eta_{i-1}^{n+1}\right) + \frac{g}{4\Delta x^2}(h_{i+1} - h_{i-1})\left(\eta_{i+1}^{n+1} - \eta_{i-1}^{n+1}\right). \tag{62}$$

To solve the linear implicit equation 52, I rewrite it in matrix notation, using the discretized divergence in 62, giving

$$\frac{g}{\Delta x^2}\boldsymbol{M}_{2h}\vec{\eta}^{n+1} + \frac{g}{4\Delta x^2}\boldsymbol{M}_{1h}\vec{\eta}^{n+1} - \frac{1}{\Delta t^2}\boldsymbol{I}\vec{\eta}^{n+1} = -\frac{1}{\Delta t^2}\vec{\eta^*} \tag{63}$$

where $\vec{\eta}^n$ is the spatial vector of sea surface height,

$$\vec{\eta}^n = \begin{bmatrix}\eta_1^n \\ \eta_2^n \\ \vdots \\ \eta_K^n\end{bmatrix}, \tag{64}$$

$\boldsymbol{I}$ is the $(K \times K)$ identity matrix, $\boldsymbol{M}_{2h}$ is the $(K \times K)$ matrix

$$\boldsymbol{M}_{2h} = \begin{bmatrix} h_1 & -2h_1 & h_1 & \cdots & 0 \\ h_2 & -2h_2 & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{K-1} & -2h_{K-1} & h_{K-1} \\ 0 & \cdots & h_K & -2h_K & h_K \end{bmatrix}, \tag{65}$$

and $\boldsymbol{M}_{1h}$ is the $(K \times K)$ matrix

$$\boldsymbol{M}_{1h} = \begin{bmatrix} 4(h_1 - h_2) & 4(h_2 - h_1) & 0 & \cdots & 0 \\ (h_1 - h_3) & 0 & (h_3 - h_1) & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & (h_{K-2} - h_K) & 0 & (h_K - h_{K-2}) \\ 0 & \cdots & 0 & 4(h_{K-1} - h_K) & 4(h_K - h_{K-1}) \end{bmatrix}. \tag{66}$$

In these definitions, I have used forward difference and backward difference at the first and final spatial points respectively. As in the first exercise of a 1-D wave equation, the boundaries can be changed to Neumann

boundaries by changing the first and last rows of $\boldsymbol{M}_{2h}$ and $\boldsymbol{M}_{1h}$. The solution for $\vec{\eta}^{n+1}$ can now easily be obtained from 63 through matrix inversion:

$$\vec{\eta}^{n+1} = -\frac{1}{\Delta t^2}\left[\frac{g}{\Delta x^2}\boldsymbol{M}_{2h} + \frac{g}{4\Delta x^2}\boldsymbol{M}_{1h} - \frac{1}{\Delta t^2}\boldsymbol{I}\right]^{-1}\vec{\eta^*}. \tag{67}$$

Finally, the $n+1$ timestep of velocity is found using centered-difference discretization of 53,

$$u_i^{n+1} = u_i^* - \frac{\Delta t g}{2\Delta x}\left(\eta_{i+1}^{n+1} - \eta_{i-1}^{n+1}\right). \tag{68}$$

Now I'll gather all the numerical steps in a concise set of steps in matrix notation. Before I do so, let's consider the boundary conditions. In global MITgcm, the boundaries either periodic, as in the ACC, or closed, as at continental margins. In regional MITgcm, there can also be open boundaries, where any motions are allowed to propagate freely out without reflection, normally implemented through an area of high viscosity outside the domain of interest. To keep it simple, here I'll use closed boundaries on both sides of my 1-D field, which has length $2L$ from $x = -L$ to $x = L$. A closed boundary requires that there is no flow normal to it, $u(x = -L) = u(x = L) = 0$. Unlike for the 1-D wave equation, I no longer need to set boundary conditions on the wave amplitude - the solution depends on mass conservation at the boundaries. In matrix notation, 50 becomes

$$\vec{u^*} = \vec{u^n} + \Delta t\left[(3/2 + \epsilon_{AB})\,\vec{G_u^n} - (1/2 + \epsilon_{AB})\,\vec{G_u^{n-1}}\right] \quad \text{with} \quad \vec{G_u^n} = -(\vec{G_u^{adv}})^n + (\vec{G_u^{h-diss}})^n. \tag{69}$$

Because of nonlinear momentum advection, $(\vec{G_u^{adv}})^n$ will remain expressed in index notation

$$(\vec{G_u^{adv}})^n = \begin{bmatrix} (G_u^{adv})_1^n \\ (G_u^{adv})_2^n \\ \vdots \\ (G_u^{adv})_K^n \end{bmatrix}, \tag{70}$$

with $\left(G_u^{adv}\right)_i^n$ given in equation 55 and with boundary conditions $u_1 = u_K = 0$. The dissipation term can be written in matrix notation as

$$(\vec{G_u^{h-diss}})^n = \frac{\nu}{\Delta x^2}\boldsymbol{Q}_2\vec{u^n}, \quad \text{where} \quad \boldsymbol{Q}_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{71}$$

The next step, equation 61, is written as

$$\vec{\eta^*} = \vec{\eta^n} - \frac{\Delta t}{2\Delta x}\left(\boldsymbol{H}_1\vec{u^*} + \boldsymbol{H}_2\vec{u^*}\right) \tag{72}$$

where

$$\boldsymbol{H}_1 = \begin{bmatrix} 2(h_2 - h_1) & 0 & \cdots & 0 & 0 \\ 0 & (h_3 - h_1) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & (h_K - h_{K-2}) & 0 \\ 0 & 0 & \cdots & 0 & 2(h_K - h_{K-1}) \end{bmatrix} \tag{73}$$

and

$$\boldsymbol{H}_2 = \begin{bmatrix} -2h_1 & 2h_1 & \cdots & 0 & 0 \\ -h_2 & 0 & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -h_{K-1} & 0 & h_{K-1} \\ 0 & 0 & \cdots & -2h_K & 2h_K \end{bmatrix}, \tag{74}$$

having used forward and backward differencing for the first and last points respectively. The implicit linear free surface condition is solved using equation 67. Finally, equation 68 can be written as

$$\vec{u}^{n+1} = \vec{u^*} - \frac{g\Delta t}{2\Delta x}\boldsymbol{M}_1\vec{\eta}^{n+1}, \tag{75}$$

with

$$\boldsymbol{M}_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \tag{76}$$

where the boundaries have been set to preserve zero velocity for all time.

To choose appropriate numerical parameters for evaluation, I consider as an ansatz satisfying $\frac{c\Delta t}{\Delta x} \leq 1$ in lieu of an analytical stability analysis for this complex system. The characteristic speed c is given by the group speed of a wave solution in the 1-D wave equation 47, which for a zero right hand side is $c_g = \sqrt{gh}$. For some maximum water depth h and some choice of Δx that sufficiently resolves the initial condition, the maximum timestep is given by

$$\Delta t = \Delta x (g h_{max})^{-1/2}. \tag{77}$$

The effect on stability of the extra terms on the right hand side of 47 will be explored numerically further in the text.

As mentioned, the solution domain is defined as going from $x = -L$ to $x = L$. To see the effect of changing bottom depth on the wave, the domain is defined as having a depth

$$h(x) = \begin{cases} 100 & \text{for} \quad -L \leq x \leq 0, \\ -\frac{95}{L}x + 100 & \text{for} \quad 0 < x \leq L, \end{cases} \tag{78}$$

such that the slope shoals linearly until a minimum depth of $h_{min} = 5$. This slope is chosen so that the depth spans two orders of magnitude to explore the depth dependence of the solution, while still weakly satisfying the criterion of $\frac{a}{h} \ll 1$ for a linear free surface.

Finally, we can choose an initial condition for the solution. Let's once again use a simple Gaussian pulse,

$$\eta(0, x) = \begin{cases} e^{-(x+L/4)^2/L_0^2} & \text{for} \quad -L \leq x \leq 0, \\ 0 & \text{for} \quad 0 < x \leq L, \end{cases} \tag{79}$$

so that the pulse is centered around the middle of the deep left region, with amplitude $\eta_0 = 1$ and width $L_0 = \frac{L}{10}$. Using $L = 200\pi$, gives a reasonable spatial step of $\Delta x = \frac{L_0}{10} = 2\pi$. Using $g = 9.81$ with $h_{max} = 100$ gives a wave group speed of $c_g = \sqrt{gh_{max}} = 31.32$. A reasonable initial choice of timestep to satisfy CFL stability is $\Delta t = 0.2$. An initial condition on the velocity is also required. From 46 I can find the initial condition

$$u(0, x) = -\frac{1}{h}\int_{-L}^{L} \frac{\partial \eta}{\partial t}\,\mathrm{d}x = \begin{cases} c_g e^{-(x+L/4)^2/L_0^2} & \text{for} \quad -L \leq x \leq 0, \\ 0 & \text{for} \quad 0 < x \leq L. \end{cases} \tag{80}$$

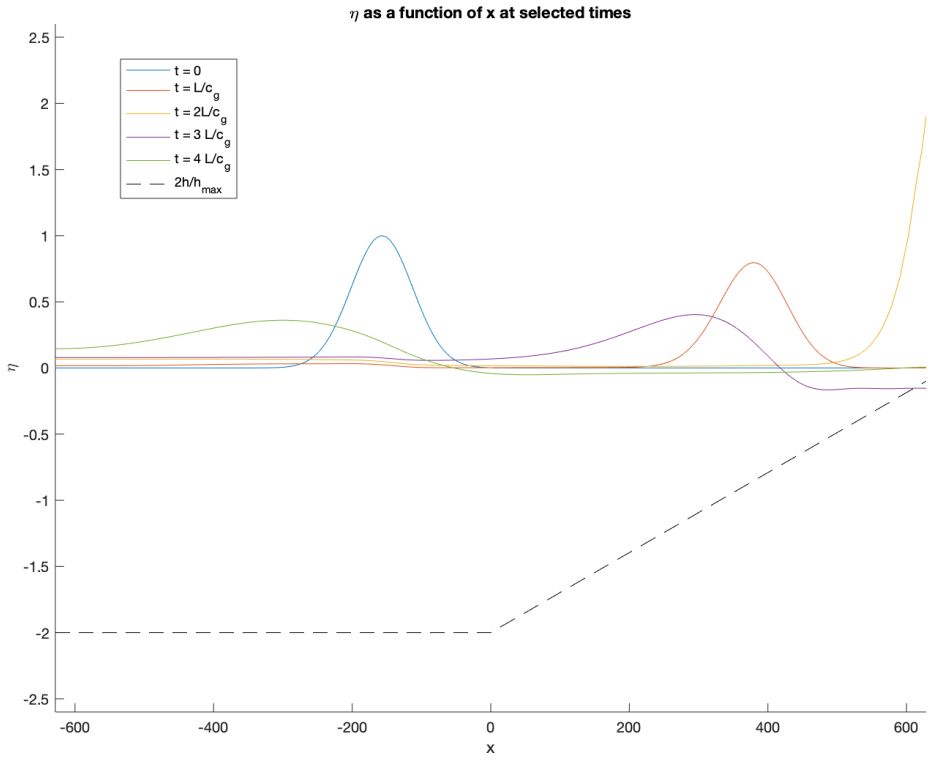

Figure 3: Snapshots of the numerical solution to 69, 72, 67, and 75 with initial conditions 79 and 80. The shape of bottom depth (not to scale with SSH) is shown in the dashed black line.

Snapshots of the solution at times $t = \left[0, \frac{L}{2c_g}, \frac{L}{c_g}, \frac{3L}{c_g}, \frac{4L}{c_g}\right]$ are shown in different colors in Figure 3. An animation to the solution for $t = [0, \frac{8L}{c_g}]$ can be accessed here. The initial Gaussian pulse is seen propagating to the right with speed c_g. As the bottom depth decreases, the wave slows down, and as it encounters the right boundary, the amplitude increases to a height of nearly twice the initial amplitude on reaching the solid boundary on the right, and reflecting back to the left. A small stationary hump pulse can be seen left behind after the initial condition, most likely due to the limits of numerical accuracy in the centered difference method and the presence of the nonlinear momentum advection. The stability of the solution is probed by increasing the timestep until the solution goes unstable. For the chosen parameters, the stability criteria appears to be

$$\frac{c\Delta t}{\Delta x} \lesssim 2.664, \tag{81}$$

but I've found that increasing the duration, decreasing the viscosity, and increasing the amplitude all create a more unstable regime, requiring a smaller Δt to keep the solution stable.

References

Campin, J.-M., Heimbach, P., Losch, M., Forget, G., edhill3, Adcroft, A., amolod, Menemenlis, D., dfer22, Hill, C., Jahn, O., Scott, J., stephdut, Mazloff, M., baylorfk, antnguyen13, Doddridge, E., Fenty, I., Bates, M., Martin, T., Abernathey, R., samarkhatiwala, Smith, T., Lauderdale, J., hongandyan, Deremble, B., raphael dussin, Bourgault, P., dngoldberg, and McRae, A. T. T. (2019). Mitgcm/mitgcm: checkpoint67m.

Munster, W. (2020). Chapter 4: The wave equation.

Matlab Code

```
%% Numerical Project: Waves in MITgcm
% This code solves the 1-D shallow water equations with linearized
% free surface MITgcm-style.
ccc;

% Run Parameters
L = 200*pi; % Half-domain length
L0 = L/10; % Decay scale for gaussian I.C.
g = 9.81; % Gravity
hmax = 100; % Max water depth
hmin = 5; % Min water depth
cg = sqrt(g*hmax); % Shallow water wave speed
dur = L*8/cg; % Run time
dx = L0/20; % Spatial step
dt = 2.7*dx/cg; % Time step
dt = min(dt,dur/600); % Minimum time step for making a movie
epsg = 0.1; % Small stability parameter
nu=1; % Viscosity


% Initialize vectors
x = -L:dx:L; % Length dimension
K = length(x);
t = 0:dt:dur; % Time dimension
T = length(t);
h = hmax*ones(size(x)); % Bottom depth
for i=(K+1)/2:K
    h(i) = (hmin-hmax)/L*x(i)+hmax;
end

% Initial conditions
eta = zeros(K,T); % Sea surface height
eta0 = 0.8*exp(-(x+L/4).^2/L0^2); % Gaussian initial condition
eta(1:(K+1)/2,1) = eta0(1:(K+1)/2)';
u = zeros(K,T); % Water speed
u0 = cg*eta0./h; % Gaussian initial condition
u(1:(K+1)/2,1) = u0(1:(K+1)/2)';
eta(1,1)=0; % removes small leftover from exponential
u(1,1)=0; % removes small leftover from exponential

% Define matrices
Q2 = diag(-2*ones(1,K),0) + diag(ones(1,K-1),1) + diag(ones(1,K-1),-1);
Q2(1,1)=0;
Q2(1,2)=0;
Q2(end,end-1)=0;
Q2(end,end)=0;


H1 = zeros(K);
```

```
for k=2:K-1
    H1(k,k) = h(k+1)-h(k-1);
end
H1(1,1)=2*(h(2)-h(1));
H1(end,end)=2*(h(end)-h(end-1));

H2 = zeros(K);
for k=2:K-1
    H2(k,k-1) = -h(k);
    H2(k,k+1) = h(k);
end
H2(1,1) = -2*h(1);
H2(1,2) = 2*h(1);
H2(end,end-1) = -2*h(K);
H2(end,end) = 2*h(K);

M2h = zeros(K);
for k=2:K-1
    M2h(k,k-1) = h(k);
    M2h(k,k) = -2*h(k);
    M2h(k,k+1) = h(k);
end
M2h(1,1)=h(1);
M2h(1,2)=-2*h(1);
M2h(1,3)=h(1);
M2h(end,end-2)=h(end);
M2h(end,end-1)=-2*h(end);
M2h(end,end)=h(end);

M1h = zeros(K);
for k=2:K-1
    M1h(k,k-1) = h(k-1)-h(k+1);
    M1h(k,k+1) = h(k+1)-h(k-1);
end
M1h(1,1)=4*(h(1)-h(2));
M1h(1,2)=4*(h(2)-h(1));
M1h(end,end-1)=4*(h(end-1)-h(end));
M1h(end,end)=4*(h(end)-h(end-1));

M1 = diag(1*ones(1,K-1),1) + diag(-1*ones(1,K-1),-1);
M1(1,1)=0;
M1(1,2)=0;
M1(end,end-1)=0;
M1(end,end)=0;

% Evaluation

G = zeros(K,2);
for n=1:T-1

    % Equation 69
```

```
    Gadv = zeros(K,1);
    for k=2:K-1
        Gadv(k,1) = 1/(4*dx)*(u(k+1,n)^2+2*u(k+1)*u(k)-2*(u(k-1)*u(k))-u(k-1)^2);
    end
    Gdiss = nu/(dx^2)*Q2*u(:,n);
    G(:,2) = Gadv+Gdiss;
    if n==1
        G12 = G(:,2);
    else
        G12 = (3/2+epsg)*G(:,2)-(1/2+epsg)*G(:,1);
    end
    G(:,1)=G(:,2);
    ustar = u(:,n)+dt*G12;

    % Equation 72
    etastar = eta(:,n) - dt/(2*dx)*(H1*ustar+H2*ustar);

    % Equation 67
    eta(:,n+1) = -1/dt^2*(g/dx^2*M2h + g/(4*dx^2)*M1h-1/dt^2*eye(K))^(-1)*etastar;

    % Equation 75
    u(:,n+1) = ustar - g*dt/(2*dx)*M1*eta(:,n+1);
end

% Plot final state
i=n;
figure
    plot(x,eta(:,i));
    hold on
    plot(x,u(:,i));
    plot(x,etastar);
    plot(x,ustar);
    plot(x,eta(:,i+1));
    plot(x,u(:,i+1));
    plot(x,-1.5*h/hmax,'k--');
    xlabel('x');
    ylabel('\eta');
    tlabel = sprintf('t = %i',t(i));
    title(['\eta as a function of x at ' tlabel]);
    ylim([-2.01,2.01])
    legend('\eta','u','\eta *','u *','\eta+1','u+1');
    set(gcf, 'Position', [576, 252, 768, 576]) % presentation size
    saveas(gcf,'eta.png')

% Plot snapshots in time

for i=round(linspace(1,T,10))
close(gcf)
figure
    plot(x,eta(:,i))
    xlabel('x');
```

```
    ylabel('\eta');
    tlabel = sprintf('t = %i',t(i));
    title(['\eta as a function of x at ' tlabel]);
    ylim([-2.01,2.01])
    set(gcf, 'Position', [576, 252, 768, 576]) % presentation size
    saveas(gcf,'eta.png')
end
```