# Adjoint-Guided Adaptive Mesh Refinement for Hyperbolic Systems of Equations

Brisa N. Davis

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Randall J. LeVeque, Chair

Loyce Adams

Frank Gonzalez

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

## Abstract

Adjoint-Guided Adaptive Mesh Refinement for Hyperbolic Systems of Equations

Brisa N. Davis

Chair of the Supervisory Committee:
Professor Randall J. LeVeque
Applied Mathematics

One difficulty in developing numerical methods for time-dependent partial differential equations is the fact that solutions contain time-varying regions where much higher resolution is required than elsewhere in the domain. The open source Clawpack software implements block-structured adaptive mesh refinement to selectively refine around propagating waves in the AMRClaw and GeoClaw packages. In particular, GeoClaw is widely used for tsunami modeling, the application that motivated this work.

For problems where the solution must be computed over a large domain but is only of interest in one small area (e.g. one coastal community when doing tsunami modeling, or the location of a pressure gauge when doing acoustics modeling), a method that allows identifying and refining the grid only in regions that influence this target area would significantly reduce the computational cost of finding a solution. The adaptive mesh refinement approach currently implemented in AMRClaw and GeoClaw often refines waves that will not impact the target area. To remedy this, we seek a method that enables the identification and refinement of only the waves that will influence the location of interest.

In this work we show that solving the time-dependent adjoint equation and using a suitable inner product with either the forward solution, or the estimated one-step error in the forward solution, allows for a more precise refinement of the relevant waves. We present the adjoint methodology first in one space dimension for illustration and in a broad context since it could also be used in other adaptive software, and for other tsunami applications beyond adaptive mesh refinement. We then show how this adjoint method has been integrated into the adaptive mesh refinement strategy of the open source AMRClaw and GeoClaw software and present linear variable coefficient acoustics and tsunami modeling results showing that the accuracy of the solution is maintained and the computational time required is significantly reduced through the integration of the adjoint method into adaptive mesh refinement. The adjoint method is compared to adaptive mesh refinement methods already available in the AMRClaw software, and the advantages and disadvantages of using the adjoint method are discussed. Other capabilities of the adjoint method such as focusing on specific time ranges of interest, sensitivity analysis, and source impact analysis and design are also presented. The new algorithms are incorporated in Clawpack and code for the examples presented in this work is archived on Github.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

## ACKNOWLEDGMENTS

I would like to express my gratitude to all of the people who have made this thesis possible. First I would like to thank my advisor Randy LeVeque for his continuous guidance and wealth of research ideas. Randy has always been incredibly encouraging, helpful, and supportive throughout my time at the University of Washington. Even when I threw in unexpected constraints, like the birth of my son, Randy was extremely patient and flexible. He has taught me so much, and I would certainly not have made it to this point without his help.

I would like to thank Loyce Adams, who had been a constant source of encouragement and interesting ideas. She has been instrumental in providing motivation, testing, and rigor to the code that I have developed in my time at UW. Thank you to the other professors on my committee, Frank Gonzalez, Anne Greenbaum, and Jody Bourgeois, who have all contributed significantly to my success. Thank you for all of the time you have sacrificed to be on my committee, and for all of the feedback and suggestions that you have provided.

Thank you to the many professors I have had throughout the years who have worked to train my mind and character. Special thanks to the professors in the Applied Mathematics department at UW. I have learned a lot, and have greatly enjoyed my time here. Thank you also to all of my peers, for the years of friendship and conversations. Special thanks to my cohort, it was fun to start and finish with you.

Finally, I am extremely grateful to my family. Thank you to my father and mother for offering constant support and encouragement, and for all of the time and effort you have spent in training me to be the person I am today. Thank you also for helping me to grow in my relationship with God. Without your friendship and backing, and without the great strength and motivation that comes from the faith you first introduced me to, I would not be finishing this degree. Thank you also to my siblings, for being there to talk to and offering refreshment and entertainment. And thank you to my son Nathan, for always being willing to offer a good excuse to ignore a research problem I am currently frustrated with. Lastly I would like to express my gratitude to my husband Andrew, for always being there with me through this long endeavor. The many conversations we have had hashing out details of my research on multiple whiteboards were both extremely helpful and a lot of fun. The support you have offered me has been invaluable, and I am honored to have been able to share this journey at UW with you.

## DEDICATION

to my husband, Andrew

# Chapter 1
# INTRODUCTION

## *1.1 Motivation*

Hyperbolic systems of partial differential equations appear in the study of numerous physical phenomena involving wave propagation. Methods for numerically calculating solutions to these systems of PDEs have broad applications in many disciplines. Complicating the development of numerical methods for solving these systems is the fact that solutions often contain discontinuities or localized steep gradients. A variety of adaptive mesh refinement (AMR) techniques have been developed to allow the use of much finer grids around discontinuities or localized regions needing higher resolution, which generally propagate as the solution evolves. Nonlinear hyperbolic problems that produce shock waves often require AMR in regions that can be easily identified by computing the local gradient. But AMR is also often extremely useful for linear hyperbolic equations with smooth solutions, particularly when the solution is of interest over some small region relative to the size of the computational domain. Examples motivating this work include modeling tsunami propagation over the ocean (for which linearized shallow water equations are suitable until the waves reach shore) and earthquake modeling, where linear elasticity equations are generally used. In both cases spatially varying coefficients in the PDE lead to scattering and reflections that can make it difficult to predict which portions of the domain must be refined at any given time in order to capture the waves that will ultimately reach the location of interest.

In this work I focus on the implementation of AMR in the Clawpack software [30, 77], and on methods for achieving targeted AMR by using the adjoint method. This open source software has been developed since 1994 and can be applied to almost any hyperbolic PDE in 1, 2, or 3 space dimensions by providing a Riemann solver, the basic building block of the high-resolution

Godunov-type finite volume methods that are employed [10, 66, 68, 69]. The AMRClaw package of Clawpack uses block structured AMR as developed in [16, 13] and adapted to the wave-propagation algorithms used in Clawpack in [15]. The AMR software is also used in the GeoClaw variant of Clawpack developed for modeling tsunamis, storm surge, and other geophysical flows (e.g. [14, 71, 78]).

A key component of any AMR algorithm is the criterion for deciding which grid cells should be refined. Before this work, four different criteria were available in AMRClaw and GeoClaw for flagging cells that need refinement from one level to the next finer level. These are:

- a Richardson extrapolation error estimation procedure that compares the solution on the existing grid with the solution on a coarser grid and refines cells where this error estimate is greater than a specified tolerance,
- refining cells where the gradient of the solution (or an undivided difference of neighboring cell values) is large,
- refining cells where the surface elevation differs significantly from sea level (this refinement criteria is specific to GeoClaw for tsunami modeling), or
- some other user-specified criterion that examines the current solution locally.

In general these approaches will flag cells for refinement anywhere that a specified refinement tolerance is exceeded, irrespective of the fact that the area of interest may be only a subregion of the full solution domain. To address this, recent versions of AMRClaw and GeoClaw also allow specifying "refinement regions," space-time subsets of the computational domain where refinement above a certain level can be either required or forbidden. This is essential in many GeoClaw applications where only a small region along the coast (some community of interest) must be refined down to a very fine resolution (often 1/3 arcsecond, less than 10 meters) as part of an ocean-scale simulation. GeoClaw simulations often use 6 or 7 nested levels of refinement, starting with a resolution of 1 or 2 degrees of latitude/longitude over the entire computational domain.

This might be refined to 4-minute or 1-minute resolution around the propagating waves, and then refined to successively higher resolution around the target region, where the finest grids may be 1/3 arcsecond for inundation studies. The target location where the highest levels of refinement is required can be specified directly by the user. These AMR regions can also be used to ensure that the simulation only refines around the waves of interest. However, since the regions are user-specified, placing them optimally often requires multiple attempts and careful examination of how the solution is behaving. This generally necessitates the use of coarser grid runs for guidance, which adds computational and user time requirements. This manual guiding of AMR may also fail to capture some waves that are important. For example, a user may determine, based on a coarser grid run, that a portion of a wave is heading away from the region of interest and therefore forbid excessive refinement in that area in an effort to save computational time. However, this portion of the wave may later reflect off a distant boundary or material heterogeneity, causing it to have an unexpected impact on the region of interest. Alternatively, edge waves may be excited that propagate back and forth along the continental shelf for hours after the primary wave has passed.

This challenge in tsunami modeling was the original motivation for our work on adjoint-based refinement, which we are also incorporating into the more general Clawpack software [30]. GeoClaw is based on the AMRClaw branch of Clawpack, a more general code that implements adaptive mesh refinement in both two and three space dimensions (which has now been extended to one space dimension, as a part of this work). Other applications where adjoint-based refinement could be very useful include earthquake simulation, for example, where the desire might be to efficiently refine only the seismic waves that will reach a particular location. For any problem where only a particular area of the total solution is of interest, a method that allows specifically targeting and refining the grid in regions that influence this area of interest would allow for computational savings while also ensuring that the accuracy of the solution is preserved.

## *1.2 Contributions of this work*

In this work we are concerned with the question of how best to refine the computational domain to capture the portions of the propagating waves that will eventually impact a target location, without over-refining waves that will not. We present a general approach to using the adjoint equation to efficiently guide adaptive refinement in this situation, and describe an implementation of this approach in the Clawpack software. In this work we show acoustics examples, where the region of interest might be dictated by the placement of a pressure gauge or other measurement device. We also explore the use of adjoint methods in tsunami modeling by incorporating them into the open source GeoClaw software that is widely used for tsunami simulation, see e.g. [14, 49, 71], where the target location is likely dictated by a community of interest or the location of a tide gauge. However, the methods presented here could be applied to any problem or software that uses time-dependent AMR.

For a hyperbolic PDE, the adjoint equation is another (closely related) hyperbolic PDE that must be solved backwards in time, starting with a disturbance at the location of interest. Waves propagating outward in this adjoint solution indicate the portion of the domain that can affect the solution at the location and time of interest. Taking a suitable inner product between the forward solution (or an estimate of the local error in the forward solution) and the adjoint solution allows us to determine what regions need to be refined at each earlier time.

Adjoint methods are often used in conjunction with the numerical solution of differential equations for a variety of purposes, such as computing sensitivities of the solution to input data, solving inverse problems, estimating errors in the solution, or guiding the design of a computational grid to most efficiently compute particular quantities of interest. We present a detailed description of the adjoint method and its background in chapter 2, but in this work we focus primarily on one particular use: guiding adaptive mesh refinement (AMR) to efficiently capture the waves that will impact a particular "target location," by which we mean a specific location we are interested in. In the case of a tsunami modeling example this could be a location where we want to compare with available DART buoy or tide gauge data, or a portion of the coastline where we wish to compute inundation.

We show linear variable coefficient acoustics problems in this work because in this case the adjoint equation is independent of the forward solution, and can be computed *a priori* before solving the forward problem. We also develop the adjoint method for the shallow water equations, since these can be linearized about the ocean at rest to yield an equation that is valid in deep water. Since in this case a single linearized equation is valid across the ocean in deep water, the adjoint is again independent of the forward solution and can be solved before solving the forward problem. The adjoint problem is generally solved on a much coarser grid than will be used for the forward problem, so that this adds relatively little computational expense. For a nonlinear hyperbolic PDE, an adjoint problem can still be defined, but involves linearizing about a particular solution of the forward problem. For nonlinear problems, making full use of the adjoint to guide adaptive refinement might require iterating between approximations to the forward and adjoint problem as the forward problem is better resolved in appropriate regions. The work reported here could form the basis for such an extension, and has already proved very useful in its own right for the applications mentioned above. We also assume that the coefficients of the equations we consider vary in space but not in time. Time-varying coefficients could be handled in much the same way if the focus is on a single output time of interest, but in most practical cases the coefficients are time-invariant and hence the equation is autonomous in time. This makes it easy to extend the approach to handle problems where the solution at one spatial location is of interest over a range of times, which is the situation in the examples mentioned above, for example. The adjoint solution needs to be computed only once over a sufficiently long time period, and snapshots in time of this one solution can be used to guide the adaptive refinement regardless of the time period of interest.

In two space dimensions (with obvious modifications for 1 or 3 dimensions), a linear hyperbolic system with coefficients varying in space typically takes one of two forms, either the conservative form

$$q_t(x, y, t) + (A(x, y)q(x, y, t))_x + (B(x, y)q(x, y, t))_y = 0, \tag{1.1}$$

or the non-conservative form

$$q_t(x, y, t) + A(x, y)q_x(x, y, t) + B(x, y)q_y(x, y, t) = 0. \tag{1.2}$$

Of course eq. (1.2) differs from eq. (1.1) only if the coefficient matrices, $A(x, y)$ and $B(x, y)$, vary in space. Often the same physical system can be modeled by equations of either form, depending on what variables are chosen to make up the vector $q(x, y, t)$. For example, linear acoustics can be written in the form eq. (1.1) as a system of 3 equations where $q$ models small disturbances of density and momenta in the $x$- and $y$-directions. Alternatively the same acoustics problem can be modeled in the form eq. (1.2) (with different coefficient matrices) if instead the system is written in terms of pressure and velocities. Clawpack can be used to solve equations in either form. This is important to note since if the original equation is in one form then the adjoint equation takes the other form. As derived below in chapter 2, the adjoint equations arise from an integration by parts and hence if the original equation is eq. (1.1) then the adjoint equation is

$$\hat{q}_t(x, y, t) + A(x, y)^T \hat{q}_x(x, y, t) + B(x, y)^T \hat{q}_y(x, y, t) = 0, \tag{1.3}$$

while if the original equation is eq. (1.2) then the adjoint equation is

$$\hat{q}_t(x, y, t) + (A(x, y)^T \hat{q}(x, y, t))_x + (B(x, y)^T \hat{q}(x, y, t))_y = 0. \tag{1.4}$$

When solving the original time-dependent PDE, which we will call the "forward problem", generally we have initial data specified at some initial time $t_0$ and the problem is solved forward in time to find the solution at the target location at some later time $t_f$ (or, more typically, over some time range of interest). The adjoint equation (derived in chapter 2, and more specifically for particular forward problems in chapter 6) must then be solved backwards in time from the final time $t_f$ to the initial time $t_0$. The "initial " data for the adjoint equation, which must be specified at the final time, is non-zero at the target location and then spreads out into waves as the adjoint equation is solved. The key idea is that at any intermediate "regridding" time $t_r$, between $t_0$ and $t_f$, the only regions in space where the forward solution could possibly affect the target location at time $t_f$ are regions where the adjoint solution is nonzero. Moreover, by computing a suitable inner product of the forward and adjoint solutions at time $t_r$ it is possible to determine whether the forward solution at a given spatial point will actually impact the target location at the final time $t_f$, or whether it can be safely ignored. Alternatively, by computing a suitable inner product of an error estimate in the forward solution and the adjoint solution at time $t_r$, it is possible to estimate how much the error at a given spatial point will affect final accuracy at the target location. This information can then be used to decide whether or not to refine this spatial location in the forward solution.

In short, the contributions of this work are:

- The development of the use of the continuous adjoint method to guide adaptive mesh refinement (AMR).
- Implementation of this *adjoint guided* AMR in the context of AMRClaw.
- Implementation of adjoint guided AMR in the context of GeoClaw.
- Novel use of the continuous adjoint for the shallow water equations to:
  - guide AMR,
  - identify waves that contribute to the data seen at a tide gauge in a particular time interval of interest, and
  - do sensitivity analysis and source design.
- The development of a one-dimensional version of AMRClaw, to enable better performance analysis on the code base.
- Analysis of computational performance of the adjoint guided AMR methods relative to AMR methods previously used in AMRClaw and GeoClaw.

## 1.3 Overview

To develop a framework for the work presented here, I first introduce some of the background of the adjoint method in chapter 2. This chapter introduces the uses of the adjoint in literature, some of the terminology and important distinctions that are present in the adjoint method work, and the mathematical derivation of the adjoint equation first in the context of an algebraic system of equations and then for a general time-dependent linear equation. Both the forward and the adjoint problem are solved using finite volume methods, so a short introduction to the basic terminology and methods is presented in chapter 3. This chapter introduces the forms equations must take for them to be solvable using the methods presented in this work, and the algorithms used to solve the equations.

We first introduce various methods for determining where to apply AMR in chapter 4, some of which have been implemented in AMRClaw and GeoClaw already and some of which were newly implemented in the Clawpack framework for this work. This chapter presents more detail on the adjoint equation and how it can be used to guide AMR, and expands the basic adjoint method to take into account the approximated error in the calculated solution at each time step. The material throughout this chapter is generally applicable to any adaptive mesh refinement software.

Specifics on implementing this new method in the context of Clawpack are presented in chapter 5. This includes various considerations that need to be taken into account due to the nature of the algorithms in Clawpack and AMRClaw, and describes the new algorithms that were developed to solve and use the adjoint problem to guide adaptive mesh refinement.

The governing equations for the examples shown in this work are given in chapter 6. These are the linear variable coefficient acoustics equations in one and two dimensions as well as the shallow water equations. The derivation of the continuous adjoint problem for these equations is shown, and a discussion of how to solve the Riemann problem for these forward problems and their adjoints is given. Examples of the adjoint method applied to the one- and two-dimensional variable coefficient linear acoustics equations are given in chapter 7, as well as an analysis of the computational performance of this method relative to the previously available AMR methods in Clawpack. I first consider one-dimensional problems (one space dimension plus time) for ease of exposition and because it is informative to view the solutions in the $x$–$t$ plane in this case. The ideas extend directly to more space dimensions, as illustrated by the two-dimensional examples presented. This work could serve as a basis for an extension to three-dimensional problems, which would follow directly from the work presented here. Finally, examples of the adjoint method applied to tsunami modeling are given in chapter 8. This chapter not only presents the novel use of the continuous adjoint method to guide adaptive mesh refinement for tsunami modeling, but also has examples that highlight various different capabilities of the adjoint method such as focusing on time ranges of interest, sensitivity analysis, and source design.

Much of the new work presented in this thesis appears in publications as well. Large portions of chapter 2, the basics of the methods presented in chapters 4 and 5, and the tsunami example given in section 8.1 appeared in [35]. That publication also contained a one dimensional shallow water equation example, which was used to illustrate the areas that would be refined by using adjoint-flagging even though no adaptive mesh refinement was actually utilized. The one-dimensional version of AMRClaw that was developed as part of this thesis was completed after [35] was published. Therefore, rather than including the one dimensional example from [35] that did not use AMR, one dimensional examples using adaptive mesh refinement are included in section 7.1 of this thesis. A second publication which contains the majorities of chapters 4, 5 and 7 is currently in preparation [34].

# Chapter 2

# ADJOINT METHOD

In this chapter, we introduce the adjoint method in the context of linear algebraic equations and then extend the method to more general systems of equations. Common uses of the adjoint method in literature are presented, and the basic framework for how we plan on using the adjoint in this work is developed.

## *2.1 Use of the adjoint in literature*

Adjoint equations have been used computationally for many years in a variety of different fields, with wide ranging applications. A few examples include weather model tuning [55], automobile aerodynamics [89], and geodynamics [23]. The adjoint equations have been used in design methods for aeronautics, like laminar flow control [2]. A good overview of adjoint design methods for aeronautical applications can be found in [86]. Adjoint equations have even been used for seismic and tsunami inversion [3, 19].

Because of the wide use of the adjoint equations, and the great variability of the fields that it has been accepted into, the adjoint equations have also been used with a wide variety of numerical methods. For example, adjoint equations have been used with Monte Carlo methods to follow a tracer in an advection-diffusion equation [22], with finite volume methods for rotor blade design [81], and with spectral-element methods for seismic tomography [109].

The majority of the work with the adjoint equations, however, can be slit up into three main categories: sensitivity analysis, error correction, and adaptive mesh refinement.

### *2.1.1 Sensitivity analysis*

The mathematics behind how the adjoint can be used for sensitivity analysis is discussed in section 2.3 for an algebraic system of equations and in section 2.4 for a system of linear equations, but for now note that if the adjoint method gives the sensitivities of an output functional of interest to any changes in the input then this can determine which variables in the input need to be optimized. The use of the adjoint equation for optimization had its beginning in the 1970s [74]. It was first used in the field of fluid mechanics by Pironneau [95], who referred to the vector containing the variables for the adjoint equation as the *co-state vector*. He also used the adjoint in the areas of profile optimization in Stokes flow and shape design for elliptic systems [94, 96].

Jameson pioneered the effort to use the adjoint for potential flow equations in the field of aerodynamics design optimization [57]. He has since expanded his work to also consider shape design in viscous compressible flow [59, 58]. The adjoint equation has also been used for optimization in the context of the Navier-Stokes equations in 2-dimensions [39], and even on unstructured grids [87]. It has also been used for optimization with the 3-dimensional Euler equations [39]. All of these, however, were focused on steady state (time-independent) problems.

More recently, there had been some advances in using the adjoint method for time-dependent optimization problems. Rumpfkeil and Zingg developed a general framework for using the discrete adjoint for unsteady flows [101], and Nielsen used the adjoint method for time-dependent unsteady flows on dynamic grids [88]. A time-dependent adjoint problem has also been used in the context of aerodynamics design optimization [63].

The capability of the adjoint to compute sensitivities of a functional of interest to changes in the input data has also lead to adjoint equations being utilized for system control in a wide variety of applications. A general approach for using the adjoint method for optimal control of free boundary problems is presented by Marburger [80], who tests his method against a Navier-Stokes equation problem. Work with time-dependent flow control problems has also begun to appear in the literature recently. For the time-dependent case, Bewley gives an excellent review for flow-control problems [18]. Of particular interest to this dissertation, Sanders and Katopodes use the adjoint method with the shallow water equations to compute the sensitivities of the problem to changes in the input data in the context of water wave control in rivers and estuarine systems [104].

### *2.1.2 Error estimation and reduction*

In the context of a priori error estimation the use of adjoint equations, or *duality arguments* as they are sometimes called in this field, can be traced to the work of Aubin [7]. More recently, the literature has focused on a posteriori error estimates, with the goal being to improve the accuracy of the solution by finding appropriate correction terms. As in the field of sensitivity analysis, the majority of the work using the adjoint method has been for steady-state problems. Giles and Suli provide a good overview of the use of the adjoint method for a posteriori error analysis, with an emphasis on finding appropriate correction terms for the calculated solution or on finding error bounds on the final solution accuracy [52].

A technique based on solving an adjoint problem to estimate the error in a quantity of interest given a functional of the solution was developed by Eriksson et al. [40, 41, 42]. Also, motivated by the applications in computational fluid dynamics, Pierce and Giles used the adjoint equations to relate the local errors in approximating the solution to a flow problem to the corresponding global errors in the final functional of interest [92]. Using this in combination with an estimation of the local residual error produced a correction for the solution approximation that resulted in significantly improved accuracy. They applied this method to the Poisson equation in one and two dimensions and for the nonlinear quasi-one-dimensional Euler equations. This approach has since also been used in the field of aerodynamics [11].

### *2.1.3 Guiding adaptive mesh refinement*

The adjoint equation has also been used to guide adaptive mesh refinement. Typically the basic approach is to estimate the error in the calculated solution, determine the significance of that error by using the adjoint equation, and then refine the mesh appropriately.

Drawing from the use of the adjoint in optimal control, Becker and Rannacher used the adjoint equation to relate the global error to errors in the physical quantities of interest for the particular problem being considered, and then used these a posteriori error estimates to refine or coarsen the mesh [11]. The field of computational fluid dynamics motivated Venditti and Darmofal, who used this same basic approach for a standard, second-order, finite volume discretization of the quasi-one-dimensional Euler equations [110]. Both isentropic and shocked flows are considered. They then extended their approach to two-dimensional inviscid and viscous flows [111, 112]. Park extended the methods presented in [111] to three-dimensional problems in the area of computational fluid dynamics [90].

Notably, the adjoint method has been used to guide adaptive mesh refinement in connection with finite volume methods. In the finite volume literature this approach is known as output-based mesh adaptation, although perhaps a clearer term would be *adjoint-error* based mesh adaptation. Nemec et al. apply the adjoint method to mesh refinement for a Cartesian finite volume code [85, 84]. Park presents adaptive mesh refinement results for compressible flows, achieved by merging the adjoint method and a finite volume method [91]. However, in all these works steady state problems are solved and the adjoint problem takes the form of another steady state problem, typically yielding a large sparse system of linear equations to be solved.

By contrast, in our time-dependent problem the adjoint equation will also be time-dependent, and must be solved backwards in time. Note that this is a more expensive problem than the time-invariant problems the adjoint equation has been working with previously, due to the fact that there is not a single time-invariant adjoint solution but rather the problem requires computing and saving adjoint snapshots at various times.

Within the finite volume community adjoint-error based mesh refinement has begun to be used for unsteady problems. For example, temporal-only adaptation and space-time adaptation in the context of aerodynamics have been explored in [79] and [44] respectively. Similarly, the adjoint method has been used to calculate the error in the solution in an effort to determine how often to conduct blockwise time and space refinement [25].

Work with the compressible Navier-Stokes equations has been done for both static domains [76], and deforming domains [62]. This method has even been extended into the realm of general coupled time-dependent systems [6]. They derived an adjoint-based a posteriori error representation for parabolic problems, and a time-space refinement strategy based on the fact that they can distinguish between temporal and spatial errors.

Other examples include [62] and [76] who use the adjoint method to guide AMR for both spatial and temporal grid refinement for the Navier-Stokes equations and [64] who applies this method to the compressible Euler equations. All of these examples work with the discrete adjoint formulation, whereas in this work we derive the continuous adjoint and then discretize this hyperbolic system using the same finite volume methods as used for the forward problem. A discussion of the benefits and disadvantages of the continuous and discrete adjoint approaches are in section 2.2.

### *2.1.4 Geosciences and the adjoint method*

The adjoint method has been used in the geosciences for data assimilation [75, 26, 48, 27], parameter identification [9, 37], wave and flood control [38, 103, 102], seismic and tsunami source inversion [3, 19], and seismic tomography [109]. [93] presents an approach for using the adjoint method for tsunami source inversion. Since then, work has been done on finding the tsunami source based on buoy measurements and water heights [19, 61]. The discrete adjoint method has even been extended to do tsunami source inversion given inundation data [47]. Using the adjoint method for goal-oriented error estimation was first adapted from the field of computational fluid dynamics to geophysical models by [99]. They followed the methods used in computational fluid dynamics rather closely, adhering to the use of the discrete adjoint and using the calculated errors as correction terms for the forward solution. In fact, all of these works using the adjoint method in the field of geosciences used the discrete adjoint formulation.

The use of the continuous adjoint method in the context of geosciences to guide adaptive mesh refinement is novel to the work presented in this dissertation. Previously, the continuous adjoint approach had been used for the shallow water equations to perform sensitivity analysis with the goal of wave control [104]. Since the publication of our first journal paper using the continuous adjoint equations for the shallow water equations, Lacasta et al. has published work also using the continuous adjoint equations for the shallow water equations — which they used to control their internal boundary conditions [65].

## *2.2 Discrete and continuous adjoint methods*

Computationally there are two different varieties of the adjoint method: the *discrete adjoint* approach, where the PDE is first discretized and the adjoint system is derived by algorithmic differentiation of this discretization, and the *continuous adjoint* approach, where the adjoint of the original continuous mathematical equation is first derived and then discretized. The use of the discrete and the continuous adjoint systems varies by field, but generally computational work has focused on the use of the discrete adjoint. For instance, optimality conditions for aerodynamic applications have been derived from a continuous approach [17, 5] but the computer implementations have generally followed the discrete approach in this field [4].

[46] presents a comparison between the continuous and discrete adjoint for quasi-one-dimensional flow, and [82] develops a comparison between the discrete and continuous adjoint formulations for automatic aerodynamic optimization. [4] conducted a performance analysis between the discrete and continuous adjoint methods and found that they agreed far from the surface being considered – recall that this is in the context of aerodynamics so they were considering a surface in both inviscid and viscous flows. Based on their analysis they posited that in the limit of infinitely small mesh size the two approaches would agree throughout the computational domain.

In general, one of the main advantages of the discrete adjoint approach is that the equations are discretely adjoint to the discrete versions of the original (forward problem) equations. This means that the derivatives obtained are consistent with the finite-difference gradients. However, [82] found that the difference between the finite-difference gradients calculated by these two adjoint methods is small and that the cost of deriving the discrete adjoint is greater than the corresponding cost of deriving the continuous adjoint. Therefore, they argue for the use of the continuous adjoint method. Moreover, the continuous and discrete adjoint approaches have been shown to give comparable results in the area of aerodynamic shape optimization [83].

Nevertheless, the discrete adjoint formulation is still most commonly used and algorithms for this formulation continue to be developed [51]. In fact, for some problems (e.g. to obtain precise error estimates, or for optimization or control problems), it is crucial that the adjoint equations solved be the adjoint of the discretized forward problem. For this reason the majority of the existing mesh adaptation algorithms using the adjoint method are based on the discrete adjoint approach — since most mesh adaptation algorithms rely on relating the local error in the solution to the global error in a functional of interest, and doing so arises more naturally in the context of the discrete adjoint [43]. However, in the context of mesh adaptation for steady state problems the continuous and discrete adjoint approaches have been shown to give comparable results [72].

Determining the boundary conditions for the adjoint PDE can be difficult for some problems and limits the ability to use the continuous adjoint approach, although progress has been made in simplifying the derivation of the boundary conditions for this approach [72]. As a work-around, [73] introduces a hybrid approach for computing sensitivities in time-dependent problems where AMR is used. They propose a nested system, where the continuous adjoint formulation is used to compute the inner grid and the discrete adjoint computes the outer grid. This approach also allows them to use mesh refinement when solving the adjoint problem, which cannot be used when solving the adjoint problem that arises from the discrete adjoint method since the problem does not remain consistent with the original forward problem if the mesh is refined.

In our work we are primarily interested in identifying regions in space where the grid for the forward problem should be refined and for this the continuous adjoint approach appears to be sufficient and is much easier to implement in the context of Clawpack. For the hyperbolic equations we consider we will see that the boundary conditions can be easily obtained analytically, which also allows us to use the continuous adjoint. Moreover, we will show that the adjoint equation for a hyperbolic system of equations written as a PDE is also a hyperbolic PDE, which can be solved by the same finite volume methods using Clawpack as are used for the forward problem.

## 2.3 Adjoint method for linear systems

For readers not familiar with the concept of an adjoint equation, it may be easiest to appreciate the power and limitations of this approach by first considering the solution to an algebraic system of equations, beginning with a linear system of the form $Ax = b$, where $A$ is an invertible $n \times n$

matrix, $b$ is a given vector with $n$ components, and the solution is $x = A^{-1}b$. In practice such a system is best solved by Gaussian elimination, requiring $O(n^3)$ operations when A is dense for each system solve. When A is sparse with semi-bandwith $\beta$, Gaussian elimination without pivoting would require $O(n\beta^2)$ for each solve. Suppose that we are not interested in the full solution $x$ but only in one component, say $x_k$. In general we must still solve the full system to determine $x_k$. But now suppose we want to do this for many different sets of data $b$, or that we wish to determine the sensitivity of $x_k$ to changes in any component of $b$. In these situations the adjoint equation can be very useful since it requires solving only a single system of equations rather than many systems.

More generally, suppose that we only care about $J = \phi^T x = \sum_{i=1}^{n} \phi_i x_i$, where $\phi$ is some specified vector with $n$ components. In particular if $\phi$ is the unit vector with $\phi_k = 1$ and $\phi_i = 0$ for $i \neq k$, then $\phi^T x = x_k$, the case considered in the previous paragraph. The adjoint approach works for more general $\phi$, i.e. when we only care about some scalar quantity of interest that can be defined as a linear functional applied to $x$.

For the linear system $Ax = b$, the adjoint equation is the linear system $A^T \hat{x} = \phi$, where the vector $\phi$ is now used as the data on the right hand side and we solve for $\hat{x}$, the adjoint solution. The matrix $A^T$ is the transpose (also called the adjoint) of the matrix $A$, with elements $(A^T)_{ij} = A_{ji}$. This is also an $n \times n$ invertible matrix so this problem has a unique solution $\hat{x} = A^{-T}\phi$. The matrix $A^{-T}$ is the inverse of the transpose, which agrees with the transpose of the inverse.

The adjoint solution can now be used to compute $J(b)$, the value of the functional $\phi^T x$, where $x$ solves $Ax = b$, by using elementary linear algebra:

$$J(b) = \phi^T x = \phi^T A^{-1} b = (A^{-T}\phi)^T b = \hat{x}^T b. \tag{2.1}$$

Note that once we have solved the adjoint equation for $\hat{x}$, we can compute $J(b)$ for *any* data $b$ without solving additional linear systems. We need only compute the inner product $\hat{x}^T b = \sum_{i=1}^{n} \hat{x}_i b_i$, which requires only $O(n)$ operations.

Moreover, we can also compute the sensitivity of $J(b)$ to a change in any component $b_i$ of the data. Differentiating eq. (2.1) with respect to $b_i$ shows that

$$\frac{\partial J(b)}{\partial b_i} = \hat{x}_i, \tag{2.2}$$

in other words the components of the adjoint solution are exactly the sensitivities of $J$ to changes in the corresponding component of $b$. We could have estimated the sensitivity $\partial J(b)/\partial b_i$ by varying $b_i$ slightly and solving a perturbed linear system, but we would have had to solve $n$ such linear systems to estimate all the sensitivities. The adjoint equation computes them all simultaneously through the solution of a single linear system.

This can be useful in a variety of different fields. As an example, in tsunami modeling we may wish to compute the sensitivity of the tsunami observed at our target location to changes in the data, e.g. to changes in the seafloor deformation if we are using a gradient-based optimization algorithm to solve an inverse problem to match observations (see [19] for one such application to tsunami source inversion). Or we may want to determine what potential source regions around the Pacific Rim give the largest tsunami response at a particular target location (such as Pearl Harbor, as considered in a study by [105]). Rather than solving many forward problems, this can be determined with a single adjoint solution.

One limitation of the adjoint approach is that changing the target location is analogous to changing the vector $\phi$ defining the quantity of interest $J(b)$ in the linear system problem, and a new adjoint solution must be computed for each location of interest.

Another limitation is that the adjoint approach is most easily applied to a linear problem. If we replace the linear system $Ax = b$ by a nonlinear system of equations $f(x) = b$ that defines $x$ for data $b$, then we can still use an adjoint approach to compute sensitivities of $J(b) = \phi^T x$ to changes in $b$, but we must first linearize about a particular set of data $\bar{b}$ with solution $\bar{x}$ and can only compute sensitivities due to small changes in $b$ around $\bar{b}$. The adjoint equation then takes the form of a linear system where the matrix $A$ is replaced by the Jacobian matrix of the function $f$ evaluated at $\bar{x}$.

## 2.4 Developing the continuous adjoint equation

Suppose $q(x,t)$ is the solution to the time-dependent linear equation (with spatially varying coefficients)

$$q_t(x,t) + A(x)q_x(x,t) = 0, \quad a \le x \le b, \quad t_0 \le t \le t_f \tag{2.3}$$

subject to some known initial conditions, $q(x, t_0)$, and some boundary conditions at $x = a$ and $x = b$. Here $q(x, t) \in \mathbb{R}^m$ for a system of $m$ equations and we assume $A(x) \in \mathbb{R}^{m\times m}$ is diagonalizable with real eigenvalues at each $x$, so that eq. (2.3) is a hyperbolic system of equations.

Now suppose we are interested in calculating the value of a functional

$$J = \int_a^b \varphi^T(x)q(x, t_f)dx \tag{2.4}$$

for some given $\varphi(x)$. For example, if $\varphi(x) = \delta(x - x_0)$ then $J = q(x_0, t_f)$ is the solution value at the point $x = x_0$ at the final time $t_f$. This is in fact the situation we consider here, with the delta function smeared out around the region of interest for the computational approach.

If $\hat{q}(x, t) \in \mathbb{R}^m$ is any other function then multiplying this by eq. (2.3) and integrating yields

$$\int_{t_0}^{t_f} \int_a^b \hat{q}^T(x, t)\,(q_t(x, t) + A(x)q_x(x, t))\,dx\,dt = 0 \tag{2.5}$$

for any time $t_0 < t_f$. Then integrating by parts twice yields the equation

$$\int_a^b \hat{q}^T q\Big|_{t_0}^{t_f} dx + \int_{t_0}^{t_f} \hat{q}^T A q\Big|_a^b dt - \int_{t_0}^{t_f} \int_a^b q^T \left(\hat{q}_t + \left(A^T \hat{q}\right)_x\right) dx\,dt = 0. \tag{2.6}$$

By defining the adjoint equation,

$$\hat{q}_t(x, t) + (A^T(x)\hat{q}(x, t))_x = 0, \tag{2.7}$$

setting $\hat{q}(x, t_f) = \varphi(x)$, and selecting the appropriate boundary conditions for $\hat{q}(x, t)$ such that the integral in time vanishes, we can eliminate all terms from eq. (2.6) except the first term, to obtain

$$\int_a^b \hat{q}^T(x, t_f)q(x, t_f)dx = \int_a^b \hat{q}^T(x, t_0)q(x, t_0)dx. \tag{2.8}$$

The boundary conditions necessary for the integral in time to vanish vary depending on the problem being studied, and are discussed in more detail below. Given eq. (2.8), the integral of the inner product between $\hat{q}$ and $q$ at the final time is equal to the integral at the initial time $t_0$:

$$J = \int_a^b \hat{q}^T(x, t_0)q(x, t_0)dx. \tag{2.9}$$

Note that we can replace $t_0$ in eq. (2.5) with any $t$ so long as $t_0 \leq t \leq t_f$, which would yield eq. (2.9) with $t_0$ replaced by $t$.

Note that it is also possible to use adjoint equations to compute sensitivities of $J$ to changes in the input data, by using the equation

$$\delta J = \int \hat{q}^T(x,t_0)\,\delta q(x,t_0)\,dx. \tag{2.10}$$

This has led to the adjoint equations being utilized for system control in a wide variety of applications such as shallow-water wave control [104] and optimal control of free boundary problems [80]. As a simple example, if we take $\varphi(x) = \hat{q}(x,t_f)$ to be a delta function then this adjoint method approach would provide us with the sensitivity of a single solution point to changes in the data $q(x,t_0)$. This is also useful in solving inverse problems and potential applications of this approach in tsunami modeling are being studied separately.

Regardless of whether we are interested in the value of $J$ or $\delta J$, note that we are interested in computing the inner product between $\hat{q}$ and some other vector (specifically $q$ and $\delta q$, respectively) at some time $t$. Therefore, more generally, we are interested in the integral

$$\int \hat{q}^T(x,t)\,g(x,t)\,dx \tag{2.11}$$

where the vector $g(x,t)$ is determined by what we are trying to calculate. In this work we will use the adjoint equation developed above to guide adaptive mesh refinement, as is described in detail in chapter 4.

# Chapter 3

# FINITE VOLUME METHODS

In this chapter, we review the fundamentals of finite volume methods for hyperbolic conservation laws. This will provide a framework for the numerical methods that will be used for the methods developed in chapter 5 as well as the applications presented in chapters 7 and 8. More details on the topics presented briefly here can be found in [69, 67, 70, 10].

## ***3.1 Hyperbolic systems of conservation laws***

### *3.1.1 Systems of conservation laws*

The work presented here focuses on an important class of homogeneous hyperbolic equations called conservation laws. As a simple example, consider the one-dimensional conservation law (written as a PDE)

$$q_t(x,t) + f(q(x,t))_x = 0 \tag{3.1}$$

were $f(q)$ is called the *flux function*. Conservation laws such as this one typically arise from physical laws in an integral form, stating that for any two points $x_1$ and $x_2$,

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = f(q(x_1,t)) - f(q(x_2,t)). \tag{3.2}$$

Simply put, each component of $q$ measures the density of some conserved quantity, and eq. (3.2) states that the total mass of this quantity between any two points can only change due to the flux past the endpoints. Provided that $q$ and $f(q)$ are sufficiently smooth, the PDE in eq. (3.1) can be derived from the integral equation, eq. (3.2). For details on this, the reader is referred to [69].

Now consider a system of $m$ conservation laws in $d$ dimensions,

$$\frac{\partial q}{\partial t} + \sum_{j=1}^{d} \frac{\partial f_j(q)}{\partial x_j} = 0, \tag{3.3}$$

where $q \in \mathbb{R}^m$ is a vector with $m$ components representing the unknowns we wish to determine, $f(q(x,t)) \in \mathbb{R}^m$ is a vector containing the flux for each of those unknowns, and $x \in \mathbb{R}^d$ corresponds to the spatial dimension.

### *3.1.2 Hyperbolic systems of conservation laws*

We are interested in the wave propagation in conservation laws of the form eq. (3.3), which is typically observed when the system is hyperbolic. To define the hyperbolicity in the context of the systems of conservation laws we are considering, we being by rewriting eq. (3.3) as a quasi-linear system

$$\frac{\partial q}{\partial t} + \sum_{j=1}^{d} A(\hat{n}, q)\frac{\partial q}{\partial x_j} = 0 \tag{3.4}$$

where $\hat{n} = (n_1, \cdots, n_d) \in \mathbb{R}^d$ is a unit vector,

$$A(\hat{n}, q) = \sum_{j=1}^{d} n_j A_j(q), \tag{3.5}$$

and

$$A_j(q) = \left(\frac{\partial f_j}{\partial q}\right) \tag{3.6}$$

is the Jacobian of the $j^{th}$ dimension flux $f_j(q)$.

Consider the linear case, $f(q) = Aq$ where $A$ is a constant $m \times m$ real matrix. Then our conservation law eq. (3.1) becomes

$$q_t(x,t) + Aq(x,t)_x = 0 \tag{3.7}$$

which is a constant-coefficient linear system. This system is called *hyperbolic* if the matrix $A$ has real eigenvalues and a corresponding set of $m$ linearly independent eigenvectors.

For the nonlinear case, the system given by eq. (3.4) is hyperbolic at a state $q$ provided that for all unit vectors $\hat{n}$ the matrix $A(\hat{n}, q)$ is diagonalizable with $m$ real eigenvalues and a complete set of $m$ linearly independent eigenvectors. In this case, any vector in $\mathbb{R}^m$ can be uniquely decomposed as a linear combination of these eigenvectors. This is used extensively in Clawpack to decompose a problem into distinct waves, which is discussed in more detail in section 3.3. The eigenvalues must be real because they represent the corresponding wave speeds. This decomposition capability is also used to solve the adjoint problem, which is explained in detail for linear variable coefficient acoustics equations and for the shallow water equations in chapter 6.

### 3.1.3 Conservation vs. nonconservation form

There are several forms that a hyperbolic system might take, which present themselves naturally in different contexts. For instance, consider a variable coefficient system

$$q_t + A(x)q_x = 0. \tag{3.8}$$

A similar system is

$$q_t + (A(x)q)_x = 0, \tag{3.9}$$

where now the matrix $A$ appears in the $x$ derivative. If this were a constant-coefficient problem, $A(x) = A$, then these two equations would be identical. However, in the variable coefficient case the equations have different solutions. For a particular physical problem it may be possible to derive an equation of either form eq. (3.8) or eq. (3.9) by varying how the vector $q$ is defined. For examples and more details, the reader is referred to [69].

Equations of the form eq. (3.9) are said to be in *conservation form*, in contrast to equations of the form eq. (3.8) which are said to be in *non-conservation* form. Two-dimensional equations in conservative and non-conservative form were presented and discussed in eqs. (1.1) and (1.2). The significance of this distinction is explained in section 3.3.

## 3.2 Riemann problems and shock waves

A *Riemann problem* is a hyperbolic equation together with initial data that is piecewise constant with a single jump discontinuity. For a one dimensional system it takes the form

$$\begin{gathered} q_t(x,t) + f(q(x,t))_x = 0 \\ q_0(x) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases} \end{gathered}$$

where $q_l$ and $q_r$ are constant state vectors.

One of the solutions that can arise from Riemann problems is propagating shock waves. We can use the integral form of the conservation law to determine the shock speed in terms of the two states $q_l$ and $q_r$ that are to the left and the right of the shock. Suppose that the solution exhibits a single shock at $x = s(t)$. Also, assume that $x_1 < s(t) < x_2$ where $x_1$ and $x_2$ are the two points in eq. (3.2). Finally, let $q(x_1) = q_l$ and $q(x_2) = q_r$. Then we can rewrite eq. (3.2) as follows

$$\frac{d}{dt}\left(\int_{x_1}^{s(t)} q(x,t)dx + \int_{s(t)}^{x_2} q(x,t)dx\right) = f(q_l) - f(q_r)$$
$$q_l s'(t) - q_r s'(t) + \int_{x_1}^{s(t)} q_t(x,t)dx + \int_{s(t)}^{x_2} q_t(x,t)dx = f(q_l) - f(q_r),$$

where I have used the Leibniz's rule for differentiation under the integral sign. If we let $x_1 \to s(t)$ and $x_2 \to s(t)$ then both of the remaining integral terms vanish, since $q_t$ is smooth before and after the shock. Therefore

$$s'(t)\,(q_l - q_r) = f(q_l) - f(q_r) \tag{3.10}$$

is the equation for the shock speed, $s'(t)$. This is called the *Rankine-Hugoniot jump condition*.

If the system of equations is linear, $f(q) = Aq$, the Rankine-Hugoniot condition becomes

$$s'(t)\,(q_l - q_r) = A\,(q_l - q_r) \tag{3.11}$$

which means that $q_l - q_r$ must be an eigenvector of the matrix $A$, and the shock speed $s'(t)$ is the corresponding eigenvalue. This is the motivation behind requiring that $A$ be hyperbolic: the eigenvalues must be real because they represent the physical propagation velocities for waves, and the corresponding eigenvectors must be linearly independent so that waves can be uniquely decomposed as linear combinations of the eigenvectors.

It is also in this context that the importance of having an equation in conservation form becomes clear: it can be written as the integral form of a conservation law, as in eq. (3.2). For solutions involving shock waves, the integral form is more fundamental than the differential equation and forms the basis for the derivation of the Rankine-Hugoniot conditions that govern the form and speed of shock waves. For nonlinear problems, if the equation is not in conservation form then extra conditions are needed to be able to decompose the waves as unique linear combinations of the eigenvectors.

In the case of a system of equations eq. (3.3) can be rewritten in integral form by integrating over an arbitrary domain $\Omega$ in $d$-dimensional space. Let $\omega = (\omega_1, \cdots, \omega_d)$ be an outward unit normal vector to $\Omega$. Then applying the divergence theorem to eq. (3.3) gives

$$\frac{d}{dt}\int_{\Omega} q\,dx + \sum_{j=1}^{d}\int_{\partial\Omega} f_j(q)\omega_j\,dS = 0. \tag{3.12}$$

This equation implies that the change of the quantity $q$ in time within $\Omega$ is determined by the flux of $q$ through the boundary $\partial\Omega$.

## *3.3 Godunov's method*

A finite volume method is based on dividing the domain $\Omega$ into grid cells. Consider the one dimensional case, for the sake of simplicity. The goal is to keep track of the integral of $q$ over each of these grid cells, and update it appropriately using the calculated flux through the endpoints of the cell. Let the $i$th grid cell be denoted by

$$C_i = (x_{i-1/2}, x_{i+1/2})$$

where the discrete points $(x_i, t_n)$ are given by $x_i = i\Delta x$, $t_n = n\Delta t$. The numerical solution $Q_i^n$ at $(x_i, t_n)$ will approximate the average value over the $i$th cell at time $t_n$:

$$Q_i^n = \frac{1}{\Delta x}\int_{C_i} q(x, t_n)\,dx \tag{3.13}$$

where $\Delta x = x_{i+1/2} - x_{i-1/2}$ is the length of the cell.

Applying the integral form of the conservation law eq. (3.2) to the grid cell $C_i$ gives

$$\frac{d}{dt}\int_{C_i} q(x,t)\,dx = f\left(q\left(x_{i-1/2}, t\right)\right) - f\left(q\left(x_{i+1/2}, t\right)\right). \tag{3.14}$$

Using this equation we wish to approximate $Q_i^{n+1}$ given $Q_i^n$, and thereby be able update the approximate cell average $Q_i^n$ from one time step to the next. Integrating eq. (3.14) in time gives

$$\int_{C_i} q(x, t_{n+1})\,dx - \int_{C_i} q(x, t_n)\,dx = \int_{t_n}^{t_{n+1}} f\left(q\left(x_{i-1/2}, t\right)\right)\,dt - \int_{t_n}^{t_{n+1}} f\left(q\left(x_{i+1/2}, t\right)\right)\,dt.$$

Dividing by $\Delta x$ gives

$$\begin{aligned}\frac{1}{\Delta x}\int_{C_i} q(x,t_{n+1})\,dx - \frac{1}{\Delta x}\int_{C_i} q(x,t_n)\,dx \\ = \frac{1}{\Delta x}\left[\int_{t_n}^{t_{n+1}} f\left(q\left(x_{i-1/2},t\right)\right)\,dt - \int_{t_n}^{t_{n+1}} f\left(q\left(x_{i+1/2},t\right)\right)\,dt\right].\end{aligned} \tag{3.15}$$

Let the numerical flux function $F^n_{i-1/2}$ be some approximation of the average flux at $x_{i-1/2}$:

$$F^n_{i-1/2} \approx \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f\left(q\left(x_{i-1/2},t\right)\right)\,dt. \tag{3.16}$$

Then using our numerical solution $Q^n_i$ and rearranging allows us to rewrite eq. (3.15) as follows:

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}\left(F^n_{i+1/2} - F^n_{i-1/2}\right). \tag{3.17}$$

The specific method obtained depends on how we select the formula for the numerical flux function. For hyperbolic problems information propagates at finite speeds, so it is reasonable to assume that the flux will depend on the values $Q^n_i$ and $Q^n_{i-1}$. Therefore, we will consider formulas of the form

$$F^n_{i-1/2} = \mathbb{F}\left(Q^n_{i-1}, Q^n_i\right). \tag{3.18}$$

For more details on the implications of selecting different numerical flux functions, please see [69]. To actually calculate the solution $Q^{n+1}$ given the solution at the previous time step we will be considering the Riemann problem between each pair of grid cells. Solving these Riemann problems will require describing the jump in the solution vector between neighboring cells in terms of the eigenvectors of the matrix $A$ of the conservation law.

### *3.3.1 Wave propagation form*

For a linear system of equations, we can develop simple equations describing the jump in $Q$ as a sum of waves that will propagate into the cells being considered. This is particularly useful when solving hyperbolic systems that are not in conservation form. Consider the nonconservative system given in eq. (3.8). If the system is linear, and we assume that $A(x)$ is diagonalizable at each point $x$, we get

$$A(x_{i-1/2}) = R(x_{i-1/2})\Lambda(x_{i-1/2})R^{-1}(x_{i-1/2})$$

where $R(x_{i-1/2})$ is the matrix of eigenvectors and $\Lambda(x_{i-1/2})$ is the diagonal matrix of eigenvalues at the grid point $x_{i-1/2}$. Recall that, since we are focusing on hyperbolic systems of equations, all of the eigenvalues are real and we have a full set of linearly independent eigenvectors. Then the jump in the solution vector can be expressed as a linear combination of the eigenvectors, $r^P_{i-1/2}$, of $A(x_{i-1/2})$,

$$Q_i - Q_{i-1} = \sum_{p=1}^{m} \alpha^p_{i-1/2} r^p_{i-1/2} \equiv \sum_{p=1}^{m} \mathcal{W}^p_{i-1/2}, \tag{3.19}$$

where we have defined the set of waves $\mathcal{W}^p_{i-1/2}$ that will be advected with the velocity given by the corresponding eigenvalue. The coefficients $\alpha^p_{i-1/2}$ are given by

$$\alpha^p_{i-1/2} = R^{-1}(x_{i-1/2})\,(Q_i - Q_{i-1})\,.$$

Note that for this approach to yield a conservative algorithm it must be the case that

$$A_{i-1/2}\,(Q_i - Q_{i-1}) = f_i\,(Q_i) - f_{i-1}\,(Q_{i-1}) \tag{3.20}$$

which is just a rewriting of the Rankine-Hugoniot condition we saw in eq. (3.11).

### *3.3.2 Flux-based wave decomposition*

For spatially varying fluxes, or for systems in conservation form like eq. (3.9), we can instead consider using a flux-based wave decomposition. The basic idea is to decompose the flux difference $f_i(Q_i) - f_{i-1}(Q_{i-1})$ into waves, rather than the difference in the solution vector $Q$ as was used in eq. (3.19). Similarly to how we decomposed the difference in the solution vector, here we will write the flux difference as a linear combination of the eigenvectors $r^p_{i-1/2}$,

$$f_i\,(Q_i) - f_{i-1}\,(Q_{i-1}) = \sum_{p=1}^{m} \beta^p_{i-1/2} r^p_{i-1/2} \equiv \sum_{p=1}^{m} \mathcal{Z}^p_{i-1/2} \tag{3.21}$$

where coefficients $\beta^p_{i-1/2}$ are given by

$$\beta^p_{i-1/2} = R^{-1}(x_{i-1/2})\,(f_i\,(Q_i) - f_{i-1}\,(Q_{i-1}))\,.$$

For an example of when using the wave propagation form will give an incorrect solution to the Riemann problem while the flux-based wave decomposition gives the correct solution see [10].

It is interesting to note the relationship between this approach and the wave-propagation decomposition described above. Note that if eq. (3.20) is satisfied then multiplying eq. (3.19) by $A_{i-1/2}$ yields

$$f_i(Q_i) - f_{i-1}(Q_{i-1}) = \sum_{p=1}^{m} \alpha_{i-1/2}^{p} \lambda_{i-1/2}^{p} r_{i-1/2}^{p} \tag{3.22}$$

since $r_{i-1/2}^p$ is an eigenvector of $A_{i-1/2}$. Therefore, $\mathcal{Z}_{i-1/2}^p = \lambda_{i-1/2}^p \mathcal{W}_{i-1/2}^p$.

## 3.4 Riemann solvers

Clawpack [30] provides an implementation of these algorithms, along with various high order modifications and limiters to prevent nonphysical oscillations. The main building block for solving the system of hyperbolic equations is the *Riemann solver*, where the jump in the solution vector or the flux is decomposed into a linear combination of the eigenvectors. Once we have that decomposition, we have said that

$$f_i(Q_i) - f_{i-1}(Q_{i-1}) = \sum_{p=1}^{m} \mathcal{Z}_{i-1/2}^{p} = \sum_{p=1}^{m} \lambda_{i-1/2}^{p} \mathcal{W}_{i-1/2}^{p} \tag{3.23}$$

This allows us to write Godunov's updating formula eq. (3.17) as

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(A^+\Delta Q_{i-1/2} + A^-\Delta Q_{i+1/2}\right), \tag{3.24}$$

with

$$\begin{aligned} A^+\Delta Q_{i-1/2} &= \sum_{p=1}^{m} \left(\lambda_{i-1/2}^{p}\right)^{+} \mathcal{W}_{i-1/2}^{p} \\ A^-\Delta Q_{i-1/2} &= \sum_{p=1}^{m} \left(\lambda_{i-1/2}^{p}\right)^{-} \mathcal{W}_{i-1/2}^{p} \end{aligned}$$

if we are using wave propagation form eq. (3.19), where $\lambda^+ = \max(\lambda, 0)$ and $\lambda^- = \min(\lambda, 0)$, or

$$\begin{aligned} A^+\Delta Q_{i-1/2} &= \sum_{p=1}^{m} \delta^+\left(\lambda_{i-1/2}^{p}\right) \mathcal{Z}_{i-1/2}^{p} \\ A^-\Delta Q_{i-1/2} &= \sum_{p=1}^{m} \delta^-\left(\lambda_{i-1/2}^{p}\right) \mathcal{Z}_{i-1/2}^{p} \end{aligned}$$

if we are using flux-based wave decomposition eq. (3.21), where

$$\delta^{+}(\lambda) = \begin{cases} 1 & \text{if } \lambda > 0 \\ 0 & \text{if } \lambda \leq 0 \end{cases}, \qquad \delta^{-}(\lambda) = \begin{cases} 1 & \text{if } \lambda < 0 \\ 0 & \text{if } \lambda \geq 0 \end{cases}.$$

# Chapter 4
# ADAPTIVE MESH REFINEMENT

This chapter contains the basics on how adaptive mesh refinement is accomplished in Clawpack, as well as the methods that are currently available for selecting which cells to refine (referred to as *flagging* the cells for refinement). How the adjoint method can be used for flagging cells for refinement is also presented.

## *4.1 Adaptive mesh refinement in Clawpack*

The block-structured mesh refinement used in AMRClaw consists of a set of logically rectangular grid patches at multiple levels of refinement. The coarsest level contains grids that cover the entire domain, with subsequent levels representing progressively finer mesh resolutions. Generally each level is refined in both time and space to preserve the stability of the explicit finite volume method. Each level, other than the coarsest level, is properly nested within the grids that comprise the next coarsest level. For each time step on Level $L$ starting with $L = 0$ for the coarsest level (and recursively applying to the finest level $L_{\max}$), the following steps are performed:

1. If $L > 0$ then fill ghost cells around each patch at this level (from neighboring Level $L$ patches, or by interpolating from Level $L - 1$).

2. Take a time step of length $\Delta t_L$ appropriate to this level.

3. If $L < L_{\max}$, then

   (a) If it is time to regrid, flag cells on Level $L$ that need refining, and cluster these into rectangular patches to define the new Level $L + 1$ grid patches.

(b) Take $\Delta t_L/\Delta t_{L+1}$ time steps on all grid patches at Level $L+1$ by recursively applying this procedure.

(c) Update the Level $L$ solution in regions covered by Level $L+1$ patches by cell-averaging the finer grid solution, and in cells directly neighboring these patches as needed to preserve conservation.

We assume the refinement ratio $\Delta t_L/\Delta t_{L+1}$ is an integer, generally equal to the refinement ratio in space from Level $L$ to $L+1$ in order to preserve stability, since Clawpack uses explicit finite volume methods that require the Courant number be bounded by 1. See [15] for more details on the steps above, and [16, 13] for general background on this approach. Here, we are concerned only with step 3(a), and in particular the manner in which cells at one level are flagged for refinement to the next level. The clustering is then done using an algorithm of Berger and Rigoutsos presented in [12], which attempts to limit the number of unflagged cells contained in the rectangular patches while also not introducing too many separate patches. Some buffering is also done so that the refined patches extend a few grid cells out from the flagged cells. Waves can travel at most one grid cell per time step, so this ensures that waves needing refinement will not escape from the refined patches in a few time steps, but in general for a wave propagation problem it is necessary to regrid every few time steps on each level. The regridding interval and width of the buffer zone are generally related, and can each be set as input parameters in the code. Because of the need for frequent regridding, and the desire to minimize the number of needlessly refined cells, the methodology utilized for selecting which cells to flag will have a significant impact on both the accuracy of the results and the time required for the computation to run.

### *4.1.1 Flagging methods available in Clawpack*

Prior to this work there were two built-in flagging methods available in AMRClaw, undivided differences and Richardson extrapolation. Both of these methods flag a cell if the quantity being evaluated is greater than some specified tolerance. A third flagging method, specific to GeoClaw, determines whether to flag a cell based on the surface elevation of the water. For all of the methods of refinement, if the user has specified limitations on certain regions of the domain (either requiring or forbidding flagging to occur) then these limitations are also taken into account at the flagging step in the code.

*Undivided Differences.* This is the default flagging routine that is used in AMRClaw. This routine evaluates the maximum max-norm of the undivided differences between a given grid cell and its two, four, or six neighbors in one, two, or three space dimensions respectively. If this maximum max-norm is greater than the specified tolerance then the cell is flagged for refinement. Therefore, this flagging method refines the mesh wherever the solution is not sufficiently smooth. This approach often works very well for problems with shock waves where the goal is to refine around all shocks. In this work, we will refer to this flagging method as *difference-flagging*.

*Richardson Extrapolation.* This second approach to flagging is based on using Richardson extrapolation to estimate the error in each cell. For each grid on Level $L$, this is done by

- advancing the current grid a time step, as normal,
- taking an extra time step on the current grid,
- generating a coarsened grid (by a factor of 2 in each direction) and taking one time step on this grid,
- comparing the solution on these two grids to estimate the one-step error introduced in a single time step with the current Level $L$ mesh size. Cells are flagged for refinement to Level $L + 1$ where this estimate is above a given tolerance.

Note that this approach requires one additional time step on the Level $L$ grid and one time step on the coarsened grid, relative to difference-flagging. Therefore, it is more expensive than difference-flagging, but has the advantage of refining based on estimates of the error in the solution rather than simply anywhere that the solution is not sufficiently smooth. In this work we will refer to this flagging method as *error-flagging*.

*Surface Elevation.* This flagging method is specific to GeoClaw, and is in fact the default for that code. It evaluates the surface height of the water in each grid cell (the sum of water depth and underlying topography), and flags the cell if the difference between the surface height and sea level for the ocean at rest is above some tolerance. In this work we will refer to this flagging method as *surface-flagging*. As described above, cells can also be flagged if they are in a region that enforces refinement. Defining regions where refinement is either required or forbidden is common for GeoClaw simulations, due to the large total simulation area (e.g., the Pacific Ocean) and the level of resolution required to accurately model small regions of interest (e.g., one harbor at 10 m resolution). In addition, another option in GeoClaw allows flagging where the fluid speed is above a given speed tolerance.

### *4.1.2 Flagging using the adjoint method*

The use of the adjoint method to guide the flagging of cells for refinement is the focus of this work. Here a very brief description of how the method can be used for AMR is given, to provide a general framework for understanding our goal for the adjoint equations. Section 2.4 outlined the derivation of the equations required for this method in one dimension, which are expanded to two dimensions in section 4.2. Section 4.3 expands the basic adjoint method to take into account the approximated error in the calculated solution of the forward problem at each time step. Chapter 5 contains a detailed description of the algorithms used to implement this method. The end goal of this work is to determine which regions of the domain need to be refined at any given time by using the adjoint solution to determine what portions of the current forward solution will affect the target location.

The steps needed to use this flagging method are the following:

1. Determine the appropriate adjoint problem based on the forward problem.

2. Solve the adjoint problem backward in time, saving snapshots of the solution at various times.

3. When solving the forward problem, at each regridding time:

(a) identify the snapshot(s) of the adjoint solution that bracket the current time in the forward solution,

(b) find the value of each of the identified adjoint snapshots at the spatial location of each grid cell by interpolation,

(c) take a suitable inner product between the forward solution and each of the identified adjoint snapshots,

(d) and flag each cell if the maximum of these inner products for that grid cell is above a certain tolerance.

Note that there are several extra steps required for this flagging method, which we will refer to as *adjoint-flagging*, when compared to the flagging methods currently available in AMRClaw. However, much of this has been automated in the work described here, and if the adjoint problem is solved on a relatively coarse grid then the computational time added by the adjoint solution and inner products may be small relative to the time potentially saved by refining in a more optimal manner. Also note that, rather than considering the inner product between the forward problem and various selected adjoint snapshots, we could instead start from the appropriate adjoint snapshot and take a few small time steps to get to the time corresponding to the current regridding time for the forward problem. Since we are trying to minimize the amount of work required while solving the forward problem, we have selected to go with the method presented above — as it requires less computations. However, even though the method we have selected is less computationally intensive, it has proven to be extremely effective in enabling us to guide AMR for the forward problem.

Finding the appropriate adjoint problem analytically and implementing the adjoint solver is a one-time cost for each type for forward problem. For example, once the appropriate adjoint problem is found for the acoustics equations, any acoustics forward problem can use the same adjoint problem. Note that the adjoint equation is also a linear hyperbolic equation, and can therefore be solved using the same software as the forward problem. To see how this can be accomplished see chapter 6 where the adjoint problems are found for one- and two-dimensional linear acoustics problems as well as for the shallow water equations.

When using adjoint-flagging we consider two different options when it comes to taking the inner product, stemming from a slightly different refinement criterion analogous to the difference-flagging vs. error-flagging described above.

*Considering the forward solution.* The first option focuses on the magnitude of the forward solution, and asks the question:

> At a given regridding time $t_r$, what portions of the forward solution will eventually affect the target location?

To answer this question we take the inner product between the forward solution and the adjoint solution at the appropriate complementary time and flag the cells where this inner product is above the given tolerance. This has the advantage of flagging only cells that contain information that is pertinent to the target location and time interval, which is not possible with any of the flagging methods currently available in AMRClaw and GeoClaw. In this work we will refer to this flagging method as *adjoint-magnitude flagging*, which we describe in detail in sections 4.2 and 5.3. While this is often quite effective, it can be difficult to choose a suitable magnitude tolerance for flagging cells (relative to the desired accuracy of the solution). Moreover, regions where the inner product is largest do not necessarily contribute the most error to the final computed solution at the location of interest. Whether these regions actually require more refinement depends also on the smoothness of the solution and hence the accuracy of the finite volume solution.

*Considering the error in the forward solution.* The second option focuses on estimating the error in the forward solution, and asks the question:

> At a given regridding time $t_r$, what portions of the forward solution will introduce a significant amount of error that will eventually affect the target location?

This is the question we really want to answer in deciding where to refine, but requires more work. To answer this question we estimate the one-step error in the forward solution using the Richardson extrapolation algorithm of AMRClaw, take the inner product between this error estimation and the adjoint solution at the appropriate complementary time, and flag the cells where this inner product is above a given tolerance. This method still has the advantage of flagging only cells that contain information that will reach the target location, but also only if the error estimate indicates that the error in this part of the solution is signficant on the current grid resolution. However, this flagging method is slower than adjoint-magnitude flagging because it requires the estimation of the one-step error in each grid cell using Richardson extrapolation. We will refer to this flagging method as *adjoint-error flagging*, and consider it further in sections 4.3 and 5.4 after describing the basic ideas in the simpler context of adjoint-magnitude flagging.

## 4.2 The adjoint equation and adjoint-magnitude flagging

Recall that the adjoint equation in one-dimension was presented in section 2.4. Here, we extend that work to find the adjoint equation in the general context of two-dimensional time-dependent linear hyperbolic equations, and discuss how the adjoint equation can be used to guide adaptive mesh refinement. Chapter 5 describes the algorithms necessary to implement this method in AMRClaw and GeoClaw.

Suppose $q(x,t)$ is the solution to the time-dependent linear equation (with spatially varying coefficients)

$$q_t(x,y,t) + A_1(x,y)q_x(x,y,t) + A_2(x,y)q_y(x,y,t) = 0, \quad a \le x \le b, \quad \alpha \le y \le \beta, \quad t_0 \le t \le t_f \tag{4.1}$$

subject to some known initial conditions $q(x,y,t_0)$ and some boundary conditions at $x = a$, $x = b$, $y = \alpha$ and $y = \beta$. Here $q(x,t) \in \mathbb{R}^m$ for a system of $m$ equations, and we assume $A_1(x,y) \in \mathbb{R}^{m\times m}$ and $A_2(x,y) \in \mathbb{R}^{m\times m}$ are diagonalizable with real eigenvalues at each $x, y$, so that eq. (4.1) is a hyperbolic system of equations. Note that we have assumed that we start with an equation in the form eq. (1.2), but we could equally well start with an equation in the form eq. (1.1).

Now suppose we do not care about the solution everywhere, but only about the value of a linear functional

$$J = \int_a^b \int_\alpha^\beta \varphi^T(x,y)q(x,y,t_f)dy\,dx \tag{4.2}$$

for some given $\varphi(x,y)$ at the final time (or, more typically, over a range of times).

If $\hat{q}(x, y, t)$ is an appropriately sized vector of functions then note that

$$\int_{t_0}^{t_f} \int_a^b \int_\alpha^\beta \hat{q}^T \left(q_t + A_1(x, y)q_x + A_2(x, y)q_y\right) dy\, dx\, dt = 0. \tag{4.3}$$

Integrating by parts three times (in $x$, $y$, and $t$) yields the equation

$$\int_a^b \int_\alpha^\beta \hat{q}^T q|_{t_0}^{t_f} dy\, dx + \int_{t_0}^{t_f} \int_\alpha^\beta \hat{q}^T A_1(x, y) q|_a^b dy\, dt + \int_{t_0}^{t_f} \int_a^b \hat{q}^T A_2(x, y) q|_\alpha^\beta dx\, dt \\ - \int_{t_0}^{t_f} \int_a^b \int_\alpha^\beta q^T \left(\hat{q}_t + \left(A_1^T(x, y)\hat{q}\right)_x + \left(A_2^T(x, y)\hat{q}\right)_y\right) dy\, dx\, dt = 0, \tag{4.4}$$

and if we can define an adjoint problem such that all but the first term in this equation vanishes then we are left with

$$\int_a^b \int_\alpha^\beta \hat{q}^T(x, y, t_f) q(x, y, t_f) dy\, dx = \int_a^b \int_\alpha^\beta \hat{q}^T(x, y, t_0) q(x, y, t_0) dy\, dx, \tag{4.5}$$

which is analogous to eq. (2.8) in the one dimensional case. As in the one dimensional case, this expression allows us to use the inner product of the adjoint and forward problems at each time step to determine what regions will influence the point of interest at the final time. This requires that the adjoint equation have the form

$$\hat{q}_t + \left(A_1^T(x, y)\hat{q}\right)_x + \left(A_2^T(x, y)\hat{q}\right)_y = 0 \tag{4.6}$$

over the same domain as the forward problem.

Therefore, the integral of the inner product between $\hat{q}$ and $q$ at the final time is equal to the integral at the initial time $t_0$:

$$J = \int_a^b \int_\alpha^\beta \hat{q}^T(x, y, t_0) q(x, y, t_0) dy\, dx. \tag{4.7}$$

Note that we can replace $t_0$ in eq. (4.3) with any $t_r$ at which we wish to do regridding ($t_0 \leq t_r \leq t_f$), which would yield eq. (4.7) with $t_0$ replaced by $t_r$. From this we observe that the locations where the inner product $\hat{q}(x, y, t_r)^T q(x, y, t_r)$ is large at the regridding time are the areas that will have a significant effect on the functional $J$. These are the areas where the solution should be refined at time $t_r$ if we are using adjoint-magnitude flagging. To make use of this, we must first solve the adjoint equation eq. (4.6) for $\hat{q}(x, y, t)$. Equivalently, if we were solving a one-dimensional problem then we would be interested in the functional $J$ given by eq. (2.4), and from eq. (2.9) we see that the locations where the inner product $\hat{q}(x, t_r)^T q(x, t_r)$ is large at the regridding time are the areas that will have a significant effect on the functional $J$. This requires using "initial" data $\hat{q}(x, y, t_f) = \varphi(x, y)$, or $\hat{q}(x, t_f) = \varphi(x)$ if we are in one spatial dimension, so the adjoint problem must be solved backward in time. The strategy used for this is discussed in chapter 5.

## 4.3 *Adjoint-error flagging*

To extend the adjoint approach to use adjoint-error flagging, we need to consider the errors being made at each time step. For the sake of simplicity we will consider the one spatial dimension case, although the equations presented here extend easily into more spatial dimensions. Let $Q(x, t_n)$ be the calculated solution at $t_n$ that approximates $q(x, t_n)$. Then the error in the functional $J$ at the final time is given by

$$\int \hat{q}^T(x, t_f) \left[ Q(x, t_f) - q(x, t_f) \right] dx.$$

Recall that the PDE we solve is assumed to be linear and autonomous in time, and let $\mathcal{L}(\Delta t)$ represent the solution operator over time $\Delta t$, so we can write $q(\cdot, t + \Delta t) = \mathcal{L}(\Delta t) q(\cdot, t)$. For notational convenience we will also write this as $q(x, t + \Delta t) = \mathcal{L}(\Delta t) q(x, t)$, but note that $\mathcal{L}(\Delta t)$ cannot be applied pointwise, so this is really shorthand for $q(x, t + \Delta t) = [\mathcal{L}(\Delta t) q(\cdot, t)](x)$. Also note that $\mathcal{L}$ satisfies the semigroup property, $\mathcal{L}(t_2 - t_1)\mathcal{L}(t_1 - t_0) = \mathcal{L}(t_2 - t_0)$.

Now let $\tau(x, t_n)$ be the one-step error at time $t_n$, defined as the error that would be incurred by taking a single time step of length $\Delta t_n = t_n - t_{n-1}$, starting with data $Q(x, t_{n-1})$:

$$\tau(x, t_n) = Q(x, t_n) - \mathcal{L}(\Delta t_n) Q(x, t_{n-1}),$$

which is approximately $\Delta t_n$ times the local truncation error (LTE). Then the global error grows according to the recurrence

$$Q(x, t_n) - q(x, t_n) = \mathcal{L}(\Delta t_n)(Q(x, t_{n-1}) - q(x, t_{n-1})) + \tau_n,$$

which, together with the assumption of no error in the initial data yields, after $N$ steps,

$$Q(x,t_N) - q(x,t_N) = \sum_{n=1}^{N} \mathcal{L}(t_N - t_n)\tau(x,t_n). \tag{4.8}$$

Now note that eq. (2.8) implies that $\int \hat{q}^T(x,t_N)\mathcal{L}(t_N - t_n)q(x,t_n)dx = \int \hat{q}^T(x,t_n)q(x,t_n)dx$. This same expression applies if we replace $q(x,t_n)$ by any other function of $x$, since the solution operator $\mathcal{L}$ then evolves this data according to the original PDE. Hence, in particular,

$$\int \hat{q}^T(x,t_N)\mathcal{L}(t_N - t_n)\tau(x,t_n)dx = \int \hat{q}^T(x,t_n)\tau(x,t_n)dx. \tag{4.9}$$

Bearing this in mind, we turn back to considering the error in our functional of interest, given by eq. (2.4). Let $E_J$ be the error in our functional at the final time $t_N \equiv t_f$:

$$E_J = \int \hat{q}^T(x,t_N)Q(x,t_N)dx - \int \hat{q}^T(x,t_N)q(x,t_N)dx.$$

Using eqs. (4.8) and (4.9) gives us

$$\begin{aligned} E_J &= \int \hat{q}^T(x,t_N)\left[Q(x,t_N) - q(x,t_N)\right]dx \\ &= \int \hat{q}^T(x,t_N)\sum_{n=1}^{N}\mathcal{L}(t_N - t_n)\tau(x,t_n)dx \\ &= \sum_{n=1}^{N}\int \hat{q}^T(x,t_N)\mathcal{L}(t_N - t_n)\tau(x,t_n)dx \\ &= \sum_{n=1}^{N}\int \hat{q}^T(x,t_n)\tau(x,t_n)dx. \end{aligned} \tag{4.10}$$

Therefore the error in our functional at the final time is equal to the sum of integrals of the inner product of the adjoint and the one-step error at each time step. From this we can observe that the locations where the inner product $\hat{q}^T(x,t_n)\tau(x,t_n)$ is large at the time $t_n$ are the areas that will have a significant effect on the final error. Moreover we can attempt to use estimates of the one-step error to flag only the cells that contribute excessively to the estimated error in $J$.

Now suppose we wish to limit the error $E_J$ to a maximum value of $\epsilon$. Then we want

$$\left|\sum_{n=1}^{N}\int \hat{q}^T(x,t_n)\tau(x,t_n)dx\right| \leq \epsilon.$$

Note that

$$\left|\sum_{n=1}^{N} \int \hat{q}^T(x,t_n)\tau(x,t_n)dx\right| \leq \sum_{n=1}^{N} \Delta t_n \left|\frac{1}{\Delta t_n} \int \hat{q}^T(x,t_n)\tau(x,t_n)dx\right| \tag{4.11}$$

$$\leq (t_N - t_0) \max_n \frac{1}{\Delta t_n} \int \left|\hat{q}^T(x,t_n)\tau(x,t_n)\right| dx. \tag{4.12}$$

We can enforce this bound by requiring that

$$\int \left|\hat{q}^T(x,t_n)\tau(x,t_n)\right| dx \leq \epsilon \Delta t_n/(t_N - t_0) \equiv \epsilon_n \tag{4.13}$$

for each $n$.

# Chapter 5
# IMPLEMENTING THE ADJOINT METHOD IN CLAWPACK

In principle the adjoint-flagging methodology could be used with any software that uses time-dependent AMR. In this chapter I examine the specifics of implementing it in the context of AMRClaw. This chapter discusses the special considerations required for coding the adjoint method in Clawpack, and the resulting algorithms.

## 5.1 *Solving the adjoint equation*

Consider the one dimensional problem eq. (2.3) and recall that the adjoint equation eq. (2.7) has the form

$$\hat{q}_t + \left(A^T(x)\hat{q}\right)_x = 0,$$

where the initial condition for $\hat{q}$ is given at the final time, $\hat{q}(x, t_f) = \varphi(x)$, and is selected to highlight the impact of the forward solution on some region of interest.

Clawpack is designed to solve equations forward in time, so slight modifications must be made to the adjoint problem to make it compatible with our software. Define the function

$$\tilde{q}(x, \tilde{t}) \equiv \hat{q}(x, t_f - \tilde{t})$$

for $\tilde{t} > 0$. This function satisfies the *modified adjoint equation*

$$\begin{aligned} &\tilde{q}_{\tilde{t}} - \left(A^T(x)\tilde{q}\right)_x = 0 && x \in [a, b], \quad \tilde{t} > 0 \\ &\tilde{q}(a, \tilde{t}) = \hat{q}(a, t_f - \tilde{t}) && 0 \le \tilde{t} \le t_f - t_0 \\ &\tilde{q}(b, \tilde{t}) = \hat{q}(b, t_f - \tilde{t}) && 0 \le \tilde{t} \le t_f - t_0 \end{aligned} \tag{5.1}$$

with initial condition $\tilde{q}(x, 0) = \varphi(x)$ given at the initial time $\tilde{t}_0 = 0$. This problem is then solved using the Clawpack software. Snapshots of this solution are saved at regular time intervals, $\tilde{t}_0 =$

$0,\ \tilde{t}_1,\ \cdots,\ \tilde{t}_f = t_f - t_0$. Note that $\tilde{t}$ has the interpretation of time remaining to $t_f$, which will be useful when selecting which adjoint snapshots to consider at a given regridding time $t_r$. This is discussed in the next section.

AMRClaw and GeoClaw can generate output in a variety of different formats, but the current code assumes that the adjoint solution is being output in the *binary* format because the files in that format are significantly smaller. When solving the forward problem, the saved snapshots of the adjoint solution are retrieved from the appropriate output folder and the adjoint solution $\tilde{q}$ at each snapshot is saved in an 'adjoints' data structure. This structure is then utilized throughout the code whenever we need to take the inner product between the adjoint solution and either the forward solution or the Richardson error estimate of the forward solution.

The basic mechanism through which Clawpack solves hyperbolic problems is by using a Riemann solver to calculate the waves generated between each set of adjacent cells at any point in time. See [69] for more details of the wave propagating algorithms that are used. Therefore, an appropriate Riemann solver is needed for any problem being solved using this software package. For linear systems the Riemann solution is generally easy to compute in terms of the eigenstructure of the coefficient matrix, for either nonconservative or conservative linear systems. We provide details for the linear acoustics equations in one and two dimensions and for the shallow water equations, as well as the corresponding adjoint equations, in chapter 6, and the software implementation of these solvers are available at [33].

## 5.2 Using adjoint snapshots

With the adjoint solution in hand, we now turn to solving the forward problem. During this solution process it is unlikely that solution data for the adjoint problem will be available at all the times needed for regridding, nor will it generally be available on as fine a grid as the forward solution, since the adjoint solution is generally solved only on a coarse grid. As refinement occurs in space for the forward problem, maintaining the stability of the finite volume method requires that refinement must also occur in time, which will further exacerbate the issue of adjoint solution data not being available at the needed spatial and temporal locations. To address this issue, the solution for the adjoint problem at the necessary locations is approximated using linear or bilinear interpolation from the data present on the coarser grid for one- or two-dimensional problems, respectively.

Typically we are interested in a time range, between $t_s$ and $t_f$, at the location of interest rather than a single point in time. For example, if we were modeling an acoustic wave we could be interested in the time range between when the first and last waves reach our pressure gauge, or in a tsunami simulation we are interested in accurately capturing the waves reaching a tide gauge over some time range. See chapter 8 for more details and examples for this latter application. This time range comes into play when we are considering which adjoint snapshots to take into account at each time step of the forward problem. Suppose that we are solving the forward problem from $t_0$ to $t_f$ and that we are currently at regridding time $t_r$ in the solution process. Since the time for our modified adjoint problem, $\tilde{t}$, has the interpretation of being the time remaining to $t_f$, when evaluating which portions of the current solution will affect the target location at $t_f$ we need to consider the modified adjoint at $\tilde{t} = t_f - t_r$. Similarly, when evaluating which portions of the current solution will affect the target location at $t_s$ we need to consider the adjoint at $\tilde{t} = t_s - t_r$ (using the fact that the equations are autonomous in time). Because we are actually interested in evaluating which portions of the current solution will affect the target location in the time range between $t_s$ and $t_f$, we need to consider the adjoint solutions snapshots for which $t_s - t_r \leq \tilde{t} \leq t_f - t_r$. This results in a list of adjoint snapshots, $\tilde{t}_m$ for $m = m_1, m_2, \cdots, M$ that need to be considered for each time $t_r$ that we wish to preform flagging for the forward problem. To be conservative, we expand the list to include $\tilde{t}_{m_1 - 1}$ (if $\tilde{t}_{m_1} > 0$) and $\tilde{t}_{M+1}$ (if $\tilde{t}_M < t_f - t_0$).

## 5.3 Implementing adjoint-magnitude flagging

Now that we have a list of adjoint snapshots that we should consider at the regridding time $t_r$, we take the inner product between the forward solution and each adjoint snapshot on our list since we know from eq. (4.7) that the locations where this inner product is large are the areas that will have a significant impact on our functional of interest $J$. (In section 5.4 we modify this to use the one-step error estimate rather than the forward solution itself.) Since we are concerned with flagging the cell if any of these inner products exceeds the specified tolerance, we just keep track of the maximum inner product calculated at the grid cell we are considering:

$$\max_{\tilde{t}_m} \left| \hat{q}^T(x, y, \tilde{t}_m) q(x, y, t) \right| \tag{5.2}$$

for $m = m_1 - 1, m_1, m_2, \cdots, M, M+1$. Note that we must save snapshots of the adjoint at sufficiently dense output times for this to be sufficient.

If this value exceeds the tolerance, the cell is flagged. Algorithm 1 describes in pseudo-code the basic flagging procedure, although in the actual code base this functionality is spread throughout various files.

As in the standard AMRClaw and GeoClaw codes, the user must choose a tolerance and some experimentation may be required to choose a suitable tolerance, related to the scaling of the solution. This form of adjoint-magnitude flagging allows the code to avoid refining regions of the solution that cannot possibly affect $J$ (the inner product will be indentically zero in such regions). Moreover, the inner product $|\hat{q}^T(x, t_r) q(x, t_r)|$ can be viewed as a measure of the sensitivity of $J$ to changes to the solution near $x$ at the regridding time, and hence regions where this is large may be important to refine. However, just because this is large at some $x$ does not necessarily mean that the solution is inaccurate at this location and needs refinement. The next section explores the extension of this approach to incorporate error estimation.

## 5.4 *Implementing adjoint-error flagging*

We know from eq. (4.13) that if we wish to limit the amount of error in our functional of interest $J$ we need to enforce a limit on the integral of the inner product $\hat{q}^T(x, t_n)\tau(x, t_n)$, since the areas where this inner product is large at the time $t_n$ are the areas that will have a significant effect on the final error in our functional of interest. In the AMRClaw software the Richardson error estimation procedure described in section 4.1.1 provides an estimate of the one-step error, and this can be used to approximate the contribution to the error $E_J$ (from eq. (4.10)) in the functional that results from the step at $t_n$.

Let $Q(x_i, t_n; L)$ represent the numerical solution in a grid cell at level $L$, where $i$ ranges over an index set $i \in I_L$ giving the indices of cells at this level that are not covered by finer grid patches.

Only these cells are part of the best approximation to $q(x, t_n)$ that is composed of the finest-level approximation available at each point in the domain. Similarly, let $\tau(x_i, t_n; L)$ be the estimated one-step error in such a cell, and let $\hat{Q}(x_i, t_n; L)$ be the numerical approximation of the adjoint solution interpolated to $(x_i, t_n)$. Then the error estimate eq. (4.10) for $E_J$ can be approximated by

$$E_J \approx E_J^n = \sum_{L=1}^{L_{max}} \sum_{i \in I_L} \hat{Q}^T(x_i, t_n; L)\tau(x_i, t_n; L)\,\Delta x_L. \tag{5.3}$$

In practice we choose the maximum refinement level $L_{max}$ in advance along with some desired tolerance $\epsilon$ and hope to achieve an error in $J$ that is no larger than $\epsilon$. This makes the implicit assumption that if we refined everywhere to the finest level then we would be able to obtain at least this much accuracy in $J$. Our goal is to acheive $|E_J| < \epsilon$ without refining everywhere, by refining only the areas on each level $L$ where the error estimate is too large. Choosing the optimal refinement pattern would be an unsolvable global optimization problem over all grids, so we must use some heuristic to decide what is "too large" as we sequentially refine each level to the next resolution. We take the approach of trying to enforce

$$\left| \sum_{i \in I_L} \hat{Q}^T(x_i, t_n; L)\tau(x_i, t_n; L)\,\Delta x_L \right| < \epsilon_n / L_{max} \tag{5.4}$$

on each level, so that the allowable error is equally distributed among levels. This suggests a way to flag cells at level $L$ for refinement to level $L+1$: flag cells starting with the one having the largest error estimate and continue flagging (i.e., removing cells from the index set $I_L$) until the sum of the remaining estimates satisfies eq. (5.4).

This is still impossible to implement in the context of AMRClaw, where there may be many patches at each level that are handled sequentially (or in parallel with OpenMP), and instead we require a pointwise tolerance $\epsilon_n^L$ so that we can simply flag any cell where

$$\left| \hat{Q}^T(x_i, t_n; L)\tau(x_i, t_n; L)\,\Delta x_L \right| > \epsilon_n^L. \tag{5.5}$$

One simple choice is $\epsilon_n^L = (\epsilon_n / L_{max})/(b - a)$, where $(b - a)$ is the length of the domain, since $\sum_{I_L} \Delta_x \leq (b - a)$. This ignores the fact that in general only part of the domain is refined to level $L$, and so this can be improved by replacing $(b - a)$ by the total length of level $L$ grid patches. This is the approach we use in the first time step $n = 0$ to choose the initial grid patches at each level. This still ignores the fact that in some cells the error will be much smaller than the tolerance, and hence in other cells a larger error should be allowed. So at later regridding times we choose $\epsilon_n^L$ based on the error estimates from the previous regridding step, assuming a similar distribution of errors (generally valid since we regrid every few time steps on each level). We save all the error estimates as each grid patch is processed. At the next regridding time for each refinement level, $L$, we sort the error estimates from all of the grids corresponding to level $L$ from smallest to largest and sum them up until the cumulative sum of the first $j$ sorted errors reaches $\epsilon_n/L_{max}$, and then set $\epsilon_n^L$ to be the last included term,

$$\epsilon_n^L = \left|\hat{Q}^T(x_j, t_n; L)\tau(x_j, t_n; L)\,\Delta x_L\right| \tag{5.6}$$

In the next regridding step we flag any cell on level $L$ for which the pointwise error estimate exceeds this value.

**Algorithm 1:** Flagging Cells For Refinement

```
Input: Forward solution on some level $L < L_{\max}$ at a particular regridding time $t_r$, and
        adjoint solution snapshots
Output: Flagged cells needing refinement to level $L + 1$
list = adjoint snapshots at times $\tilde{t}_{m_1}, \tilde{t}_{m_2}, \cdots \tilde{t}_M$ such that $t_s - t_r \leq \tilde{t}_m \leq t_f - t_r$
if $\tilde{t}_{m_1} > 0$ add adjoint snapshot at time $t_{m_1 - 1}$ to list
if $\tilde{t}_M < t_f - t_0$ add adjoint snapshot at time $t_{M+1}$ to list
for each grid patch g at level L do
    initialize max inner product to 0 at each grid point
    for each adjoint snapshot in list do
        for each grid patch in adjoint snapshot do
            if the adjoint and forward patches overlap then
                for each cell in grid patch g do
                    interpolate the adjoint snapshot in space and time to cell center
                    if using adjoint-error flagging: estimate error in forward solution
                    calculate appropriate inner product and save if greater than current
                      recorded maximum
                end
            end
        end
    end
    for each cell on grid patch g do
        if max inner product exceeds tolerance: flag cell
    end
end
```

# Chapter 6
# GOVERNING EQUATIONS

## 6.1 One dimensional variable coefficient linear acoustics

Consider the linear acoustics equations in one dimension in a piecewise constant medium,

$$\begin{aligned} p_t(x,t) + K(x)u_x(x,t) &= 0, \\ \rho(x)u_t(x,t) + p_x(x,t) &= 0, \end{aligned} \tag{6.1}$$

in the domain $x \in [a,b], t > t_0$. Setting

$$A(x) = \begin{bmatrix} 0 & K(x) \\ 1/\rho(x) & 0 \end{bmatrix} \quad \text{and} \quad q(x,t) = \begin{bmatrix} p(x,t) \\ u(x,t) \end{bmatrix}, \tag{6.2}$$

gives us the equation $q_t(x,t) + A(x)q_x(x,t) = 0$. Here $\rho(x)$ is the density and $K(x)$ is the bulk modulus.

The adjoint equation for this forward problem is given by eq. (2.7), but recall that in Clawpack we will instead solve the modified adjoint equation $\tilde{q}_t - (A(x)^T\tilde{q})_x = 0$ that is solved forward in time (see eq. (5.1)).

### 6.1.1 Riemann solvers

Recall from chapter 3 that a Riemann problem is the hyperbolic equation of interest together with initial data that is piecewise constant with a single jump discontinuity, say,

$$q_0(x) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

where $q_l$ and $q_r$ are constant state vectors. For variable coefficient linear problems, the Riemann problem also assumes that the variable coefficients have a jump discontinuity at $x = 0$, e.g., from $\rho_l$ to $\rho_r$.

For the acoustics equations $q_t + A(x)q_x = 0$ with $A(x)$ given by eq. (6.2), it is easy to compute that the eigenvalues are

$$\lambda^1 = -c(x) \qquad \text{and} \qquad \lambda^2 = c(x)$$

where $c(x) = \sqrt{K(x)/\rho(x)}$ is the speed of sound in material. We also define the impedance $Z(x) = \sqrt{K(x)\rho(x)} = \rho(x)c(x)$. The eigenvectors for this matrix $A(x)$ are

$$r^1 = \begin{bmatrix} -Z(x) \\ 1 \end{bmatrix} \qquad \text{and} \qquad r^2 = \begin{bmatrix} Z(x) \\ 1 \end{bmatrix}. \tag{6.3}$$

Note that any scalar multiple of these vectors would still be an eigenvector of $A(x)$. This particular normalization was chosen for consistency with [69], where many more details are given about Riemann solutions and the manner in which these are used as the building block for the high-resolution finite volume methods implemented in Clawpack.

The solution to the Riemann problem for acoustics consists of two waves, a left-going sound wave with speed $-c_l$ and a right-going sound wave with speed $c_r$. The left-going wave moves into a homogeneous material with impedance $Z_l$ and hence the jump in $q$ across this wave is a multiple of the eigenvector $r_l^1 = [-Z_l, 1]^T$, while the jump in $q$ across the right-going wave must be a multiple of $r_r^1 = [Z_r, 1]^T$. Between these two waves the state $q_m$ must be constant across the jump in material properties at $x = 0$, or else the resulting stationary jump in $q$ would lead to a delta function singularity in $q_t$ according to the PDE. This structure is illustrated in fig. 6.1.

Therefore, we wish to find $q_m$ such that

$$q_m - q_l = \alpha^1 \begin{bmatrix} -Z_l \\ 1 \end{bmatrix} \equiv \mathcal{W}_{rl}^1 \qquad \text{and} \qquad q_r - q_m = \alpha^2 \begin{bmatrix} Z_r \\ 1 \end{bmatrix} \equiv \mathcal{W}_{rl}^2$$

for some scalar coefficients $\alpha^1$ and $\alpha^2$. If we add these two equations together we get

$$q_r - q_l = \alpha^1 \begin{bmatrix} -Z_l \\ 1 \end{bmatrix} + \alpha^2 \begin{bmatrix} Z_r \\ 1 \end{bmatrix}.$$

This gives us a linear system of two equations to solve for $\alpha^1$ and $\alpha^2$. The solution of this gives us the decomposition of the jump in $q$ as the sum of the two acoustic waves,

$$q_r - q_l = \sum_{p=1}^{2} \alpha^p r^p \equiv \sum_{p=1}^{2} \mathcal{W}_{rl}^p \tag{6.4}$$

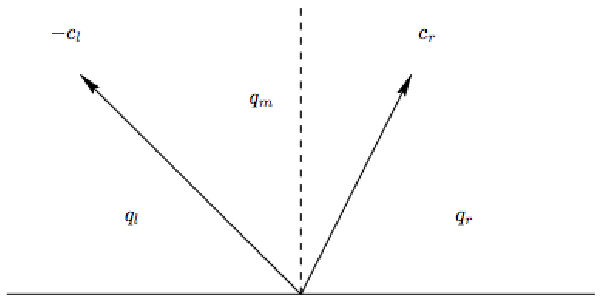


Figure 6.1: Structure of the solution to the Riemann problem for the forward problem, in the $x - t$ plane. The dashed line depicts the interface between two different cells. Between the waves is a single state $q_m$. For the adjoint problem with initial states $\tilde{q}_l$ and $\tilde{q}_r$ the wave speeds are the same but there will be two intermediate states $\tilde{q}_{ml}$ and $\tilde{q}_{mr}$ to either side of the dashed line. Figure from [69].

When solving the Riemann problem for the modified adjoint equation $\tilde{q}_t - (A(x)^T\tilde{q})_x = 0$ that is solved forward in time (see eq. (5.1)), the eigenvalues of $-A(x)^T$ are the same as the eigenvalues of $A(x)$, but the eigenvectors are now

$$\tilde{r}^1 = \begin{bmatrix} 1 \\ Z(x) \end{bmatrix} \quad \text{and} \quad \tilde{r}^2 = \begin{bmatrix} 1 \\ -Z(x) \end{bmatrix}. \tag{6.5}$$

Since the adjoint equation is in conservation form, it is the flux $\tilde{f}(\tilde{q}, x) \equiv -A(x)^T\tilde{q}$ that must be continuous across $x = 0$ in order to avoid a singularity, and so $\tilde{q}$ will generally have have a jump from $\tilde{q}_{ml}$ to $\tilde{q}_{mr}$ across $x = 0$, while $\tilde{f}(\tilde{q}_{ml}) = \tilde{f}(\tilde{q}_{mr}) \equiv \tilde{f}_m$.

As before, the jump across each wave is given by an eigenvector of the coefficient matrix from the appropriate material. Therefore, we have that

$$\tilde{f}_m - \tilde{f}_l = \beta^1 \begin{bmatrix} 1 \\ Z_l \end{bmatrix} \equiv \mathcal{Z}^1_{rl} \qquad \text{and} \qquad \tilde{f}_r - \tilde{f}_m = \beta^2 \begin{bmatrix} 1 \\ -Z_r \end{bmatrix} \equiv \mathcal{Z}^2_{rl}$$

are the left-going and right-going waves, for some scalar coefficients $\beta^1$ and $\beta^2$. If we add these two equations together we get

$$\tilde{f}_r - \tilde{f}_l = \beta^1 \begin{bmatrix} 1 \\ Z_l \end{bmatrix} + \beta^2 \begin{bmatrix} 1 \\ -Z_r \end{bmatrix}.$$

This gives us a linear system of equations to solve for $\beta^1$ and $\beta^2$. If we define

$$\delta^1 = -u_r/\rho_r + u_l/\rho_l, \qquad \delta^2 = -K_r p_r + K_l p_l$$

then

$$\beta^1 = \frac{Z_r\delta^1 + \delta^2}{Z_l + Z_r}, \qquad \beta^2 = \frac{Z_l\delta^1 - \delta^2}{Z_l + Z_r}.$$

The solution of this gives us the decomposition of the jump in $\tilde{f}$ as the sum of the two acoustic waves,

$$\tilde{f}_r - \tilde{f}_l = \sum_{p=1}^{2} \beta^p r^p \equiv \sum_{p=1}^{2} \mathcal{Z}^p_{rl} \tag{6.6}$$

These so-called f-waves can be used directly in the f-wave version of the wave-propagation algorithms used in Clawpack, as discussed further in [10], or these can be converted into jumps in $\tilde{q}$ as follows. Since the left-going and right-going waves propagate through constant materials, we have (by the Rankine-Hugoniot jump conditions) that

$$\tilde{q}_{ml} - \tilde{q}_l = \frac{\tilde{f}_m - \tilde{f}(\tilde{q}_l)}{\lambda^1_l} = -\frac{\beta^1}{c_l}\begin{bmatrix} 1 \\ Z_l \end{bmatrix}, \qquad \tilde{q}_r - \tilde{q}_{ml} = \frac{\tilde{f}(\tilde{q}_r) - \tilde{f}_m}{\lambda^2_r} = \frac{\beta^2}{c_r}\begin{bmatrix} 1 \\ Z_r \end{bmatrix},$$

which allows determining the jumps in $\tilde{q}$ across each propagating wave.

Note that $\tilde{r}^k$, defined in eq. (6.5), is orthogonal to $r^k$, defined in eq. (6.3), for $k = 1, 2$. Therefore, the inner product between the left-going waves for the modified adjoint problem and the forward problem will be zero. Similarly, the inner product between the right-going waves for the modified adjoint problem and the forward problem will also be zero. As we will see in later examples, this property allows us to correctly ignore parts of the waves from the forward problem that pass through the waves from the modified adjoint problem but are not headed in the correct direction to have an impact on our region of interest. This property is also true for the two dimensional systems of equations that we present in this work.

More complete details can be found in the references cited above, and in the Clawpack code implementing the examples shown in this thesis which can be found in [32, 33]. In this code base, the adjoint problem is solved using the f-wave version of the wave-propagation algorithms.

## *6.2 Two dimensional variable coefficient linear acoustics*

In two dimensions the variable coefficient linear acoustics equations are

$$\begin{aligned} p_t(x,y,t) + K(x,y)\left(u_x(x,y,t) + v(x,y,t)_y\right) &= 0, \\ \rho(x,y)u_t(x,y,t) + p_x(x,y,t) &= 0, \\ \rho(x,y)v_t(x,y,t) + p_y(x,y,t) &= 0, \end{aligned} \tag{6.7}$$

in the domain $x \in [a,b],\ y \in [\alpha,\beta],\ t > t_0$. Setting

$$A(x,y) = \begin{bmatrix} 0 & K(x,y) & 0 \\ 1/\rho(x,y) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B(x,y) = \begin{bmatrix} 0 & 0 & K(x,y) \\ 0 & 0 & 0 \\ 1/\rho(x,y) & 0 & 0 \end{bmatrix}, \tag{6.8}$$

and

$$q(x,y,t) = \begin{bmatrix} p(x,y,t) \\ u(x,y,t) \\ v(x,y,t) \end{bmatrix},$$

gives us the equation $q_t(x,y,t) + A(x,y)q_x(x,y,t) + B(x,y)q_y(x,y,t) = 0$.

The form of the adjoint equation for this forward problem is given by eq. (4.6), but recall that, as in the one dimensional case, in Clawpack we will actually solve the modified adjoint equation

$$\tilde{q}_t - \left(A^T(x,y)\tilde{q}\right)_x - \left(B^T(x,y)\tilde{q}\right)_y = 0 \qquad x \in [a,b], y \in [\alpha,\beta], \quad t_f \geq t \geq t_0 \tag{6.9}$$

that is solved forward in time (see eq. (5.1)).

### 6.2.1 *Riemann solvers*

For the acoustics equations $q_t(x,y,t) + A(x,y)q_x(x,y,t) + B(x,y)q_y(x,y,t) = 0$ with $A(x,y)$ and $B(x,y)$ given by eq. (6.8), it is easy to compute that the eigenvalues for both $A(x,y)$ and $B(x,y)$ are

$$\lambda^1 = -c(x,y), \qquad \lambda^2 = 0, \qquad \lambda^3 = c(x,y),$$

where $c(x,y) = \sqrt{K(x,y)/\rho(x,y)}$ is the speed of sound in material. The eigenvectors for $A(x,y)$ are

$$r_A^1 = \begin{bmatrix} -Z(x,y) \\ 1 \\ 0 \end{bmatrix}, \qquad r_A^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad r_A^3 = \begin{bmatrix} Z(x,y) \\ 1 \\ 0 \end{bmatrix},$$

and the eigenvectors for $B(x,y)$ are

$$r_B^1 = \begin{bmatrix} -Z(x,y) \\ 0 \\ 1 \end{bmatrix}, \qquad r_B^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad r_B^3 = \begin{bmatrix} Z(x,y) \\ 0 \\ 1 \end{bmatrix},$$

where $Z(x,y) = \rho(x,y)c(x,y)$ is the impedance.

The solution to the Riemann problem normal to each cell interface for acoustics in two dimensions consists of three waves. The Riemann problem at the $(i-1/2, j)$ edge, for example, has a left-going sound wave with speed $-c_{i-1,j}$, a right-going sound wave with speed $c_{i,j}$, and a stationary wave with speed 0. The left-going wave moves into a homogeneous material with impedance $Z_{i-1,j}$ and hence the jump in $q$ across this wave is a multiple of the eigenvector $r_{i-1,j}^1 = [-Z_{i-1,j}, 1, 0]^T$, while the jump in $q$ across the right-going wave must be a multiple of $r_{i,j}^1 = [Z_{i,j}, 1, 0]^T$.

The decomposition of the jump from $q_{i,j}$ to $q_{i-1,j}$ is computed exactly as in the one-dimensional case, giving us

$$q_{i,j} - q_{i-1,j} = \alpha^1 \begin{bmatrix} -Z_{i-1,j} \\ 1 \\ 0 \end{bmatrix} + \alpha^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \alpha^3 \begin{bmatrix} Z_{i,j} \\ 1 \\ 0 \end{bmatrix}.$$

for some scalar coefficients $\alpha^1$, $\alpha^2$, and $\alpha^3$. This gives us a linear system of three equations to solve for the scalar coefficients $\alpha^1$, $\alpha^2$, $\alpha^3$. The solution of this gives us the decomposition of the jump in $q$ as the sum of the three acoustic waves,

$$q_{i,j} - q_{i-1,j} = \sum_{p=1}^{3} \alpha^p r_A^p \equiv \sum_{p=1}^{3} \mathcal{W}^p_{i-1/2,j}. \tag{6.10}$$

Decomposing the jump in $q$ when sweeping in the $y$-direction, at say the $(i, j-1/2)$ edge, follows the same procedure and can be found in detail in [69].

When solving the Riemann problem normal to each cell interface for the modified adjoint equation $\tilde{q}_t - (A(x,y)^T\tilde{q})_x - (B(x,y)^T\tilde{q})_y = 0$ that is solved forward in time (see eq. (5.1)), the eigenvalues of $\tilde{A}(x,y) \equiv -A(x,y)^T$ and $\tilde{B}(x,y) \equiv -B(x,y)^T$ are the same as the eigenvalues of $A(x,y)$ and $B(x,y)$, but the eigenvectors for $\tilde{A}(x,y)$ are

$$\tilde{r}_A^1 = \begin{bmatrix} 1 \\ Z(x,y) \\ 0 \end{bmatrix}, \qquad \tilde{r}_A^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \tilde{r}_A^3 = \begin{bmatrix} 1 \\ -Z(x,y) \\ 0 \end{bmatrix},$$

and the eigenvectors for $\tilde{B}(x,y)$ are

$$\tilde{r}_B^1 = \begin{bmatrix} 1 \\ 0 \\ Z(x,y) \end{bmatrix}, \qquad \tilde{r}_B^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \tilde{r}_B^3 = \begin{bmatrix} 1 \\ 0 \\ -Z(x,y) \end{bmatrix}.$$

As in the one dimensional case, since the adjoint equation is in conservation form, it is the flux $\tilde{f}(\tilde{q}, x, y) \equiv \tilde{A}(x,y)\tilde{q}$ that must be continuous across the $(i-1/2, j)$ edge when sweeping in the $x$-direction in order to avoid a singularity, and so between the two waves $\tilde{f}_m$ will be constant across the $(i-1/2, j)$ edge, while $\tilde{q}$ will generally have a jump at the $(i-1/2, j)$ edge.

As before, the jump across each wave is given by an eigenvector of the coefficient matrix from the appropriate material. Therefore, following the same steps as in the one dimensional case, we have that

$$\tilde{A}_{i,j}\tilde{Q}_{i,j} - \tilde{A}_{i-1,j}\tilde{Q}_{i-1,j} = \beta_A^1 \begin{bmatrix} 1 \\ Z_{i-1,j} \\ 0 \end{bmatrix} + \beta_A^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \beta_A^3 \begin{bmatrix} 1 \\ -Z_{i,j} \\ 0 \end{bmatrix} \tag{6.11}$$

for some scalar coefficients $\beta_A^1$, $\beta_A^2$, and $\beta_A^3$. This gives us a linear system of equations to solve for $\beta_A^1$, $\beta_A^2$, and $\beta_A^3$. If we define

$$\delta_A^1 = -u_{i,j}/\rho_{i,j} + u_{i-1,j}/\rho_{i-1,j}, \qquad \delta_A^2 = -K_{i,j}p_{i,j} + K_{i-1,j}p_{i-1,j}$$

then

$$\beta_A^1 = \frac{Z_{i,j}\delta_A^1 + \delta_A^2}{Z_{i,j} + Z_{i-1,j}}, \qquad \beta_A^2 = 0, \qquad \beta_A^3 = \frac{Z_{i-1,j}\delta_A^1 - \delta_A^2}{Z_{i,j} + Z_{i-1,j}}.$$

This gives us the decomposition of the jump in the flux $\tilde{f}$ as the sum of the three acoustic waves,

$$\tilde{A}_{i,j}\tilde{Q}_{i,j} - \tilde{A}_{i-1,j}\tilde{Q}_{i-1,j} = \sum_{p=1}^{3} \beta_A^p \tilde{r}_A^p \equiv \sum_{p=1}^{3} \mathcal{Z}_{i-1/2,j}^p. \tag{6.12}$$

The left-going fluctuation is given by

$$\mathcal{A}^- \Delta\tilde{Q}_{i-1/2,j} \equiv \mathcal{Z}_{i-1/2,j}^1, \tag{6.13}$$

and the right-going fluctuation is given by

$$\mathcal{A}^+ \Delta\tilde{Q}_{i-1/2,j} \equiv \mathcal{Z}_{i-1/2,j}^3. \tag{6.14}$$

These so-called f-waves can be used directly in the f-wave version of the wave-propagation algorithms used in Clawpack, as mentioned previously, or these can be converted into jumps in $\tilde{Q}$ as follows. Since the left-going and right-going waves propagate through constant materials, we have (by the Rankine-Hugoniot jump conditions) that

$$\tilde{Q}_{i,j} - \tilde{Q}_{i-1,j} = -\frac{\beta_A^1}{c_{i-1,j}} \begin{bmatrix} 1 \\ Z_{i-1,j} \\ 0 \end{bmatrix} + \left(v_{i,j} - v_{i-1,j}\right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{\beta_A^3}{c_{i,j}} \begin{bmatrix} 1 \\ -Z_{i,j} \\ 0 \end{bmatrix}.$$

Note that for the coefficients of $\tilde{r}^1$ and $\tilde{r}^3$ we have simply divided $\beta_A^1$ and $\beta_A^3$ by the corresponding wave speed. However, given that the wave speed corresponding to $\tilde{r}_A^2$ is equal to zero, some care must be given for the stationary wave. For more details on how the coefficient for this stationary wave is found when dealing with the jump in $q$ see [69].

When sweeping in the $y$-direction it is the flux $\tilde{g}(\tilde{q}, x, y) \equiv \tilde{B}(x, y)^T \tilde{q}$ that must be continuous across the $(i, j-1/2)$ edge in order to avoid a singularity, and so between the two waves $\tilde{g}_m$ will be constant across the $(i, j-1/2)$ edge, while $\tilde{q}$ will generally have a jump at the $(i, j-1/2)$ edge.

As before, the jump across each wave is given by an eigenvector of the coefficient matrix from the appropriate material. Therefore, we have that

$$\tilde{B}_{i,j}\tilde{Q}_{i,j} - \tilde{B}_{i,j-1}\tilde{Q}_{i,j-1} = \beta_B^1 \begin{bmatrix} 1 \\ 0 \\ Z_{i,j-1} \end{bmatrix} + \beta_B^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta_B^3 \begin{bmatrix} 1 \\ 0 \\ -Z_{i,j} \end{bmatrix}$$

for some scalar coefficients $\beta_B^1$, $\beta_B^2$, and $\beta_B^3$. This gives us a linear system of equations to solve for $\beta_B^1$, $\beta_B^2$, and $\beta_B^3$. If we define

$$\delta_B^1 = -v_{i,j}/\rho_{i,j} + v_{i,j-1}/\rho_{i,j-1}, \qquad \delta_B^2 = -K_{i,j}p_{i,j} + K_{i,j-1}p_{i,j-1}$$

then

$$\beta_B^1 = \frac{Z_{i,j}\delta_B^1 + \delta_B^2}{Z_{i,j} + Z_{i,j-1}}, \qquad \beta_B^2 = 0, \qquad \beta_B^3 = \frac{Z_{i,j-1}\delta_B^1 - \delta_B^2}{Z_{i,j} + Z_{i,j-1}}.$$

This gives us the decomposition of the jump in $\tilde{g}$ as the sum of the three acoustic waves,

$$\tilde{B}_{i,j}\tilde{Q}_{i,j} - \tilde{B}_{i,j-1}\tilde{Q}_{i,j-1} = \sum_{p=1}^{3} \beta_B^p \tilde{r}_B^p \equiv \sum_{p=1}^{3} \mathcal{Z}_{i,j-1/2}^p. \tag{6.15}$$

The down-going fluctuation is given by

$$\mathcal{B}^- \Delta\tilde{Q}_{i,j-1/2} \equiv \mathcal{Z}_{i,j-1/2}^1, \tag{6.16}$$

and the up-going fluctuation is given by

$$\mathcal{B}^+ \Delta\tilde{Q}_{i,j-1/2} \equiv \mathcal{Z}_{i,j-1/2}^3. \tag{6.17}$$

Since the down-going and up-going waves propagate through constant materials, we have (by the Rankine-Hugoniot jump conditions) that

$$\tilde{Q}_{i,j} - \tilde{Q}_{i,j-1} = -\frac{\beta_B^1}{c_{i,j-1}}\begin{bmatrix} 1 \\ 0 \\ Z_{i,j-1} \end{bmatrix} + \left(u_{i,j} - u_{i,j-1}\right)\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \frac{\beta_B^3}{c_{i,j}}\begin{bmatrix} 1 \\ 0 \\ -Z_{i,j} \end{bmatrix}.$$

Note that the formulas given by eqs. (6.12) and (6.15) are essentially the same, except that the use of the second and third components of $Q$ are switched in the definitions of the coefficients $\beta^1$, $\beta^2$, and $\beta^3$, depending on which velocity ($u$ or $v$) is normal to the surface being considered. In the code base [33] the f-wave version of the algorithms is used when solving the adjoint problem.

### 6.2.2 *Transverse Riemann solvers*

Since we are dealing with two-dimensional problems, we must also consider transverse propagation. Consider the f-wave splitting given by eq. (6.12) when sweeping in the $x$-direction. Since to arrive at eq. (6.12) we were considering the Riemann problem at the $(i-1/2, j)$ edge, the right-going fluctuation $\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$, given by eq. (6.14), is flowing into $(i, j)$ cell. This right-going fluctuation, $\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$, is split into up-going and down-going fluxes $\mathcal{B}^+\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$ and $\mathcal{B}^-\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$ that modify the fluxes above and below the cell $(i, j)$ respectively.

To compute the up-going flux $\mathcal{B}^+\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$ we decompose the right-going flux into eigenvectors corresponding to the up-going and down-going waves at the interface $(i, j+1/2)$. So, we need to compute

$$\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j} = \beta^1\begin{bmatrix} 1 \\ 0 \\ Z_{i,j} \end{bmatrix} + \beta^2\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta^3\begin{bmatrix} 1 \\ 0 \\ -Z_{i,j+1} \end{bmatrix}$$

for some scalar coefficients $\beta^1$, $\beta^2$, and $\beta^3$, where the corresponding wave speeds are $-c_{i,j}$, 0, and $c_{i,j+1}$ respectively. This gives us a linear system of equations to solve for $\beta^1$, $\beta^2$, and $\beta^3$. Solving this linear system gives

$$\begin{aligned}
\beta^1 &= \frac{Z_{i,j+1}\left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + \left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{Z_{i,j+1}+Z_{i,j}} \\
\beta^2 &= 0 \\
\beta^3 &= \frac{Z_{i,j}\left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 - \left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{Z_{i,j+1}+Z_{i,j}}
\end{aligned}$$

where $\left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^p$ is the $p$th element in the vector $\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$. Then the up-going fluctuation is given by

$$\mathcal{B}^+\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j} = c_{i,j+1}\beta^3 \begin{bmatrix} 1 \\ 0 \\ -Z_{i,j+1} \end{bmatrix}.$$

Similarly, to compute the down-going flux $\mathcal{B}^-\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}$ we decompose the right-going flux flowing from the $(i-1/2, j)$ edge into the $(i, j)$ cell, given by eq. (6.14), into eigenvectors corresponding to the up-going and down-going waves at the interface $(i, j-1/2)$. So, we need to compute

$$\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j} = \beta^1 \begin{bmatrix} 1 \\ 0 \\ Z_{i,j-1} \end{bmatrix} + \beta^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta^3 \begin{bmatrix} 1 \\ 0 \\ -Z_{i,j} \end{bmatrix}$$

for some scalar coefficients $\beta^1, \beta^2$, and $\beta^3$, where the corresponding wave speeds are $-c_{i,j-1}$, 0, and $c_{i,j}$ respectively. This gives us a linear system of equations to solve for $\beta^1, \beta^2$, and $\beta^3$. Solving this linear system gives

$$\begin{aligned}
\beta^1 &= \frac{Z_{i,j}\left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + \left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{Z_{i,j}+Z_{i,j-1}} \\
\beta^2 &= 0 \\
\beta^3 &= \frac{Z_{i,j-1}\left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 - \left(\mathcal{A}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{Z_{i,j}+Z_{i,j-1}}.
\end{aligned}$$

Then the down-going fluctuation is given by

$$\mathcal{B}^{-}\mathcal{A}^{+}\Delta\tilde{Q}_{i-1/2,j} = -c_{i,j-1}\beta^{1}\begin{bmatrix} 1 \\ 0 \\ Z_{i,j-1} \end{bmatrix}.$$

The left-going flux $\mathcal{A}^{-}\Delta\tilde{Q}_{i-1/2,j}$ flowing from the $(i-1/2, j)$ edge into the $(i-1, j)$ cell, given by eq. (6.13), must be similarly decomposed into up-going and down-going fluxes. When sweeping in the $y$-direction it is the up-going and down-going fluxes, $\mathcal{B}^{+}\Delta\tilde{Q}_{i,j-1/2}$ and $\mathcal{B}^{-}\Delta\tilde{Q}_{i,j-1/2}$, given by eqs. (6.16) and (6.17) and flowing from the $(i, j-1/2)$ edge into the $(i, j)$ and $(i, j-1)$ cells respectively, that must be similarly decomposed into left-going and right-going fluxes. Therefore, at each cell interface three Riemann problems must be solved:

- one Riemann problem splitting the jump in flux into waves traveling normal to the cell interface, and

- two Riemann problems splitting each of these waves into the two waves traveling parallel to the cell interface.

Again, more details can be found in the references cited above, and in the Clawpack code implementing the examples shown in this paper which can be found in [32, 33].

## *6.3 Two dimensional shallow water equations*

One limitation to using the adjoint method is that the adjoint approach is most easily applied to a linear problem. If instead of the linear systems that we have been considering we have a nonlinear system of equations then we can still use an adjoint approach, but we must first linearize about a particular forward solution. Since our linearization will only be valid for small perturbations from that particular solution, our adjoint problem will also only be valid for small perturbations from that particular forward solution. Therefore, a nonlinear problem would in principle require some kind of automated iterative process that linearizes about the current forward solution at regular time intervals to find the appropriate adjoint problem for that time.

The GeoClaw software solves the nonlinear shallow water equations, but in this work we restrict our attention to the application of tracking waves in the ocean that will reach the target location. Since a tsunami in the ocean typically has an amplitude that is very small compared to the ocean depth, these equations essentially reduce to the linear shallow water equations and the adjoint equation linearized about the ocean at rest is sufficient for our needs. This signifies that a single adjoint equation, found by linearizing about the ocean at rest, can be used for the entire computation (rather than needing to linearize about the forward solution at regular time intervals, as would likely be required for a different nonlinear problem). We will see that these adjoint equations take a very similar form to the linearized shallow water equations, although with slightly different boundary conditions. If we wanted to compute sensitivies of the nonlinear onshore inundation to changes in data then we would have to linearize about a particular forward solution, which is not completed in this work but is an interesting area for future work.

In two space dimensions the shallow water equations take the form

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0 \\ (hu)_t + (hu^2 + \tfrac{1}{2}gh^2)_x + (huv)_y &= -ghB_x \\ (hv)_t + (huv)_x + (hv^2 + \tfrac{1}{2}gh^2)_y &= -ghB_y. \end{aligned} \tag{6.18}$$

Here, $u(x, y, t)$ and $v(x, y, t)$ are the depth-averaged velocities in the two horizontal directions and $B(x, y)$ is the bottom surface elevation relative to mean sea level. Let the water surface elevation be given by

$$\eta(x, y, t) = h(x, y, t) + B(x, y).$$

The shallow water equations are a special case of a hyperbolic system of equations,

$$q_t(x, y, t) + f(q)_x + g(q)_y = \psi(q, x, y) \tag{6.19}$$

in two dimensions, where $q$ is a vector of unknowns, $f(q)$ and $g(q)$ are the vectors of corresponding fluxes, and $\psi$ is a vector of source terms. Equations of this form appear in the study of numerous physical phenomena where wave motion is important, and hence methods for numerically calculating solutions to these systems of partial differential equations have broad applications over multiple disciplines.

As mentioned above, when tracking a tsunami in the ocean the nonlinear shallow water equations essentially reduce to the linear shallow water equations. To linearize the shallow water equations about the ocean at rest, we begin by letting $\mu = hu$ and $\gamma = hv$ represent the momentum in $x$ and $y$, respectively, and noting that the momentum equations from eq. (6.18) can be rewritten as

$$\mu_t + (hu^2)_x + gh(h + B)_x + (huv)_y = 0$$
$$\gamma_t + (huv)_x + (hv^2)_y + gh(h + B)_y = 0$$

Linearizing these equations as well as the continuity equation about an ocean at rest, given by a flat surface $\bar{\eta}$ and zero velocity $\bar{u} = 0, \bar{v} = 0$, with $\bar{h}(x) = \bar{\eta} - B(x)$ gives

$$\tilde{\eta}_t + \tilde{\mu}_x + \tilde{\gamma}_y = 0$$
$$\tilde{\mu}_t + g\bar{h}(x,y)\tilde{\eta}_x = 0$$
$$\tilde{\gamma}_t + g\bar{h}(x,y)\tilde{\eta}_y = 0$$

for the perturbation $(\tilde{\eta}, \tilde{\mu}, \tilde{\gamma})$ about $(\bar{\eta}, 0, 0)$. Dropping tildes and setting

$$F(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ g\bar{h}(x,y) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad G(x,y) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ g\bar{h}(x,y) & 0 & 0 \end{bmatrix}, \quad q(x,y,t) = \begin{bmatrix} \eta \\ \mu \\ \gamma \end{bmatrix}, \tag{6.20}$$

gives us a system of the form of eq. (4.1): $q_t(x,y,t) + F(x,y)q_x(x,y,t) + G(x,y)q_y(x,y,t) = 0$. The form of the adjoint equation for this forward problem is given by eq. (4.6), but recall that in Clawpack we will actually solve the modified adjoint equation

$$\tilde{q}_t - \left(F^T(x,y)\tilde{q}\right)_x - \left(G^T(x,y)\tilde{q}\right)_y = 0 \qquad x \in [a,b], y \in [\alpha,\beta], \quad t_f \geq t \geq t_0 \tag{6.21}$$

that is solved forward in time (see eq. (5.1)). When modeling tsunamis the correct boundary conditions to use for the adjoint problem are zero normal velocity at all interfaces between any wet cell and dry cell so that the boundary terms also drop out of expression eq. (4.4) to obtain eq. (4.5).

### *6.3.1 Riemann solvers*

In general, GeoClaw solves the two-dimensional nonlinear shallow water equations in the form of a nonlinear system of hyperbolic conservation laws for depth and momentum. The details of the Riemann solver in GeoClaw can be found in [50]. Away from coastlines, this solver reduces to a Roe solver for the shallow water equations plus bathymetry, which means that the eigenstructure of a locally linearized Riemann problem is solved at each cell interface, making it no more expensive in the deep ocean than simply solving the linearized equations, but also capable of robustly handling nonlinearity near shore and inundation.

Here, we will focus on describing the Riemann solver used for the linear adjoint problem. When solving the Riemann problem normal to each cell interface for the modified adjoint equation $\tilde{q}_t - (F(x,y)^T\tilde{q})_x - (G(x,y)^T\tilde{q})_y = 0$ that is solved forward in time (see eq. (5.1)), with $F(x,y)$ and $G(x,y)$ given by eq. (6.20), it is easy to compute that the eigenvalues for both $\tilde{F} \equiv -F(x,y)^T$ and $\tilde{G} \equiv -G(x,y)^T$ are given by

$$\lambda^1 = -c(x,y), \qquad \lambda^2 = 0, \qquad \lambda^3 = c(x,y),$$

where $c(x,y) = \sqrt{g\bar{h}(x,y)}$ is the speed of gravity waves.

The eigenvectors for $\tilde{F}$ are

$$\tilde{r}_F^1 = \begin{bmatrix} c(x,y) \\ 1 \\ 0 \end{bmatrix}, \qquad \tilde{r}_F^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \tilde{r}_F^3 = \begin{bmatrix} -c(x,y) \\ 1 \\ 0 \end{bmatrix},$$

and the eigenvectors for $\tilde{G}$ are

$$\tilde{r}_G^1 = \begin{bmatrix} c(x,y) \\ 0 \\ 1 \end{bmatrix}, \qquad \tilde{r}_G^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \tilde{r}_G^3 = \begin{bmatrix} -c(x,y) \\ 0 \\ 1 \end{bmatrix}.$$

Since the adjoint equation is in conservation form it is the flux $\tilde{f}(\tilde{q},x,y) \equiv \tilde{F}(x,y)\tilde{q}$ that must be continuous across the $(i-1/2,j)$ edge when sweeping in the $x$-direction in order to avoid a singularity, and so between the two waves $\tilde{f}_m$ will be constant across the $(i-1/2, j)$ edge, while $\tilde{q}$ will generally have a jump at the $(i-1/2, j)$ edge.

As before, the jump across each wave is given by an eigenvector of the coefficient matrix from the appropriate material. Therefore, following the same steps above, we have that

$$\tilde{F}_{i,j}\tilde{Q}_{i,j} - \tilde{F}_{i-1,j}\tilde{Q}_{i-1,j} = \beta_F^1 \begin{bmatrix} c_{i-1,j} \\ 1 \\ 0 \end{bmatrix} + \beta_F^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \beta_F^3 \begin{bmatrix} -c_{i,j} \\ 1 \\ 0 \end{bmatrix}. \tag{6.22}$$

for some scalar coefficients $\beta_F^1$, $\beta_F^2$, and $\beta_F^3$. This gives us a linear system of equations to solve for $\beta_F^1$, $\beta_F^2$, and $\beta_F^3$. If we define

$$\delta_F^1 = -g\left(\bar{h}_{i,j}\mu_{i,j} - \bar{h}_{i-1,j}\mu_{i-1,j}\right), \qquad \delta_F^2 = -\eta_{i,j} + \eta_{i-1,j}$$

then

$$\beta_F^1 = \frac{\delta_F^1 + c_{i,j}\delta_F^2}{c_{i,j} + c_{i-1,j}}, \qquad \beta_F^2 = 0, \qquad \beta_F^3 = \frac{-\delta_F^1 + c_{i-1,j}\delta_F^2}{c_{i,j} + c_{i-1,j}}.$$

This gives us the decomposition of the jump in the flux $\tilde{f}$ as the sum of the three acoustic waves,

$$\tilde{F}_{i,j}\tilde{Q}_{i,j} - \tilde{F}_{i-1,j}\tilde{Q}_{i-1,j} = \sum_{p=1}^{3} \beta_F^p \tilde{r}_F^p \equiv \sum_{p=1}^{3} \mathcal{Z}_{i-1/2,j}^p. \tag{6.23}$$

The left-going fluctuation is given by

$$\mathcal{F}^- \Delta\tilde{Q}_{i-1/2,j} \equiv \mathcal{Z}_{i-1/2,j}^1, \tag{6.24}$$

and the right-going fluctuation is given by

$$\mathcal{F}^+ \Delta\tilde{Q}_{i-1/2,j} \equiv \mathcal{Z}_{i-1/2,j}^3. \tag{6.25}$$

When sweeping in the $y$-direction it is the flux $\tilde{g}(\tilde{q}, x, y) \equiv \tilde{G}(x,y)^T\tilde{q}$ that must be continuous across the $(i, j-1/2)$ edge in order to avoid a singularity, and so between the two waves $\tilde{g}_m$ will be constant across the $(i, j-1/2)$ edge, while $\tilde{q}$ will generally have have a jump at the $(i, j-1/2)$ edge.

As before, the jump across each wave is given by an eigenvector of the coefficient matrix from the appropriate material. Therefore, we have that

$$\tilde{G}_{i,j}\tilde{Q}_{i,j} - \tilde{G}_{i,j-1}\tilde{Q}_{i,j-1} = \beta_G^1 \begin{bmatrix} c_{i,j-1} \\ 0 \\ 1 \end{bmatrix} + \beta_G^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta_G^3 \begin{bmatrix} -c_{i,j} \\ 0 \\ 1 \end{bmatrix}$$

for some scalar coefficients $\beta_G^1$, $\beta_G^2$, and $\beta_G^3$. This gives us a linear system of equations to solve for $\beta_G^1$, $\beta_G^2$, and $\beta_G^3$. If we define

$$\delta_G^1 = -g\left(\bar{h}_{i,j}\gamma_{i,j} - \bar{h}_{i,j-1}\gamma_{i,j-1}\right), \qquad \delta_G^2 = -\eta_{i,j} + \eta_{i,j-1}$$

then

$$\beta_G^1 = \frac{\delta_G^1 + c_{i,j}\delta_G^2}{c_{i,j} + c_{i,j-1}}, \qquad \beta_G^2 = 0, \qquad \beta_G^3 = \frac{-\delta_G^1 + c_{i,j-1}\delta_G^2}{c_{i,j} + c_{i,j-1}}.$$

This gives us the decomposition of the jump in the flux $\tilde{g}$ as the sum of the three acoustic waves,

$$\tilde{G}_{i,j}\tilde{Q}_{i,j} - \tilde{G}_{i,j-1}\tilde{Q}_{i,j-1} = \sum_{p=1}^{3} \beta_G^p \tilde{r}_G^p \equiv \sum_{p=1}^{3} \mathcal{Z}_{i,j-1/2}^p. \tag{6.26}$$

The down-going fluctuation is given by

$$\mathcal{G}^- \Delta\tilde{Q}_{i,j-1/2} \equiv \mathcal{Z}_{i,j-1/2}^1, \tag{6.27}$$

and the up-going fluctuation is given by

$$\mathcal{G}^+ \Delta\tilde{Q}_{i,j-1/2} \equiv \mathcal{Z}_{i,j-1/2}^3. \tag{6.28}$$

### *6.3.2 Transverse Riemann solvers*

Since we are dealing with a two-dimensional problem, we must also consider transverse propagation. Consider the f-wave splitting given by eq. (6.23) when sweeping in the $x$-direction. Since to arrive at eq. (6.23) we were considering the Riemann problem at the $(i-1/2, j)$ edge, the right-going fluctuation $\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$, given by eq. (6.25), is flowing into $(i, j)$ cell. This right-going fluctuation

$\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$, is split into up-going and down-going fluxes $\mathcal{G}^+\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$ and $\mathcal{G}^-\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$ that modify the fluxes above and below the cell $(i, j)$ respectively.

To compute the up-going flux $\mathcal{G}^+\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$ we decompose the right-going flux into eigenvectors corresponding to the up-going and down-going waves at the interface $(i, j + 1/2)$. So, we need to compute

$$\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j} = \beta^1 \begin{bmatrix} c_{i,j} \\ 0 \\ 1 \end{bmatrix} + \beta^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta^3 \begin{bmatrix} -c_{i,j+1} \\ 0 \\ 1 \end{bmatrix}$$

for some scalar coefficients $\beta^1$, $\beta^2$, and $\beta^3$, where the corresponding wave speeds are $-c_{i,j}$, 0, and $c_{i,j+1}$ respectively. This gives us a linear system of equations to solve for $\beta^1$, $\beta^2$, and $\beta^3$. Solving this linear system gives

$$\begin{aligned}
\beta^1 &= \frac{\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + c_{i,j+1}\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{c_{i,j+1} + c_{i,j}} \\
\beta^2 &= 0 \\
\beta^3 &= \frac{-\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + c_{i,j}\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{c_{i,j+1} + c_{i,j}}
\end{aligned}$$

where $\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^p$ is the $p$th element in the vector $\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$. Then the up-going fluctuation is given by

$$\mathcal{G}^+\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j} = c_{i,j+1}\beta^3 \begin{bmatrix} -c_{i,j} \\ 0 \\ 1 \end{bmatrix}.$$

Similarly, to compute the down-going flux $\mathcal{G}^-\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}$ we decompose the right-going flux flowing from the $(i-1/2, j)$ edge into the $(i, j)$ cell, into eigenvectors corresponding to the up-going and down-going waves at the interface $(i, j - 1/2)$. So, we need to compute

$$\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j} = \beta^1 \begin{bmatrix} c_{i,j-1} \\ 0 \\ 1 \end{bmatrix} + \beta^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \beta^3 \begin{bmatrix} -c_{i,j} \\ 0 \\ 1 \end{bmatrix}$$

for some scalar coefficients $\beta^1$, $\beta^2$, and $\beta^3$, where the corresponding wave speeds are $-c_{i,j-1}$, 0, and $c_{i,j}$ respectively. This gives us a linear system of equations to solve for $\beta^1$, $\beta^2$, and $\beta^3$. Solving this linear system gives

$$\begin{aligned}\beta^1 &= \frac{\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + c_{i,j}\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{c_{i,j}+c_{i,j-1}}\\ \beta^2 &= 0\\ \beta^3 &= \frac{-\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^1 + c_{i,j-1}\left(\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j}\right)^2}{c_{i,j}+c_{i,j-1}}.\end{aligned}$$

Then the down-going fluctuation is given by

$$\mathcal{G}^-\mathcal{F}^+\Delta\tilde{Q}_{i-1/2,j} = -c_{i,j-1}\beta^1\begin{bmatrix} c_{i,j-1}\\ 0\\ 1\end{bmatrix}.$$

The left-going flux $\mathcal{F}^-\Delta\tilde{Q}_{i-1/2,j}$ flowing from the $(i-1/2, j)$ edge into the $(i-1, j)$ cell, given in eq. (6.24), must be similarly decomposed into up-going and down-going fluxes. When sweeping in the $y$-direction it is the up-going and down-going fluxes, $\mathcal{G}^+\Delta\tilde{Q}_{i,j-1/2}$ and $\mathcal{G}^-\Delta\tilde{Q}_{i,j-1/2}$, given in eqs. (6.27) and (6.28) and flowing from the $(i, j-1/2)$ edge into the $(i, j)$ and $(i, j-1)$ cells respectively, that must be similarly decomposed into left-going and right-going fluxes. So, as with the two-dimensional linear acoustics, at each cell interface multiple Riemann problems must be solved.

Again, more details can be found in the references cited above, and in the Clawpack code implementing the examples shown in this paper which can be found in [32, 33].

# Chapter 7
# RESULTS FROM VARIABLE COEFFICIENT LINEAR ACOUSTICS

This chapter presents some examples of the performance of the algorithms developed in this work relative to the older AMRClaw algorithms. We begin with two one-dimensional linear acoustics examples for ease of visualization. Our first example showcases the capabilities of adjoint-flagging in a rather simple setting, and the second showcases a more complex scenario. We then present two two-dimensional linear acoustics examples. These four examples highlight the strengths of adjoint-flagging when compared to the flagging methods currently available in AMRClaw. For some examples using adjoint-flagging in a two dimensional shallow water equations context see [21], [32], and chapter 8.

## *7.1 One-dimensional variable coefficient linear acoustics*

The linear acoustics equations in one dimension in a piecewise constant medium are given by eq. (6.1), and they can be written in the form $q_t(x,t) + A(x)q_x(x,t) = 0$ with $q(x,t)$ and $A(x)$ given by eq. (6.2). Assume that the problem is defined in the domain $x \in [a,b], t > t_0$, with solid wall (reflecting) boundary conditions at each boundary,

$$u(a,t) = 0, \quad u(b,t) = 0, \qquad t \geq t_0. \tag{7.1}$$

### *7.1.1 Example 1: constant impedance in one dimension*

Let $t_0 = 0$, $t_f = 34$, $a = -12$, $b = 12$,

$$\rho(x) = \begin{cases} 1 & \text{if } x < 0, \\ 4 & \text{if } x > 0, \end{cases} \quad \text{and} \quad K(x) = \begin{cases} 4 & \text{if } x < 0, \\ 1 & \text{if } x > 0, \end{cases}$$

so that

$$c(x) = \begin{cases} 2 & \text{if } x < 0, \\ 0.5 & \text{if } x > 0, \end{cases} \qquad \text{and} \qquad Z(x) \equiv 2.$$

Since the impedance is constant across $x = 0$, so are the eigenvectors of the coefficient matrix $A$, which are given by eq. (6.3). Therefore, the sound speed changes and the waves are deformed as they pass through the interface, but there will be no reflected waves. As initial data for $q(x, t)$ we take two wave packets in pressure, one on each side of the interface, and zero velocity everywhere. The wave packets in pressure are given by

$$p(x, 0) = e^{-\beta_r(x-3)^2} \sin(f_r x) + e^{-\beta_l(x+2.5)^2} \sin(f_l x) \tag{7.2}$$

with $\beta_l = 20$, $\beta_r = 5$, $f_l = 20$ and $f_r = 3$. As time progresses, the wave packets split into equal left-going and right-going waves which interact with the walls and the interface giving reflected and transmitted waves (recall that there will be no reflected waves off of the interface because the impedance is constant throughout the domain).

Suppose that we are interested in the accurate estimation of the pressure near some point $x_p$. Then we can use the functional $J = \int_a^b \alpha \exp\left(-\hat{\beta}\left(x - x_p\right)^2\right) p(x, t_f)\, dx$, which corresponds to eq. (2.4) with

$$\varphi(x) = \begin{bmatrix} \alpha \exp\left(-\hat{\beta}\left(x - x_p\right)^2\right) \\ 0 \end{bmatrix}. \tag{7.3}$$

For this example we take $x_p = 7.5$, $\hat{\beta} = 50$, and

$$\alpha = \sqrt{\hat{\beta}/\pi} \tag{7.4}$$

to normalize the Gaussian so it has mass 1 and represents an averaging of the pressure in a small region around $x_p$. If we define the adjoint solution by solving

$$\begin{aligned} &\hat{q}_t + \left(A^T(x)\hat{q}\right)_x = 0 && x \in [a, b], \quad t_f \geq t \geq t_0 \\ &\hat{u}(a, t) = 0, \quad \hat{u}(b, t) = 0 && t_f \geq t \geq t_0, \end{aligned} \tag{7.5}$$

then the second and third terms in eq. (2.6) vanish and we are left with eq. (2.8), which is the expression that allows us to use the inner product of the adjoint and forward problems at each time step to determine what regions will influence the point of interest at the final time.

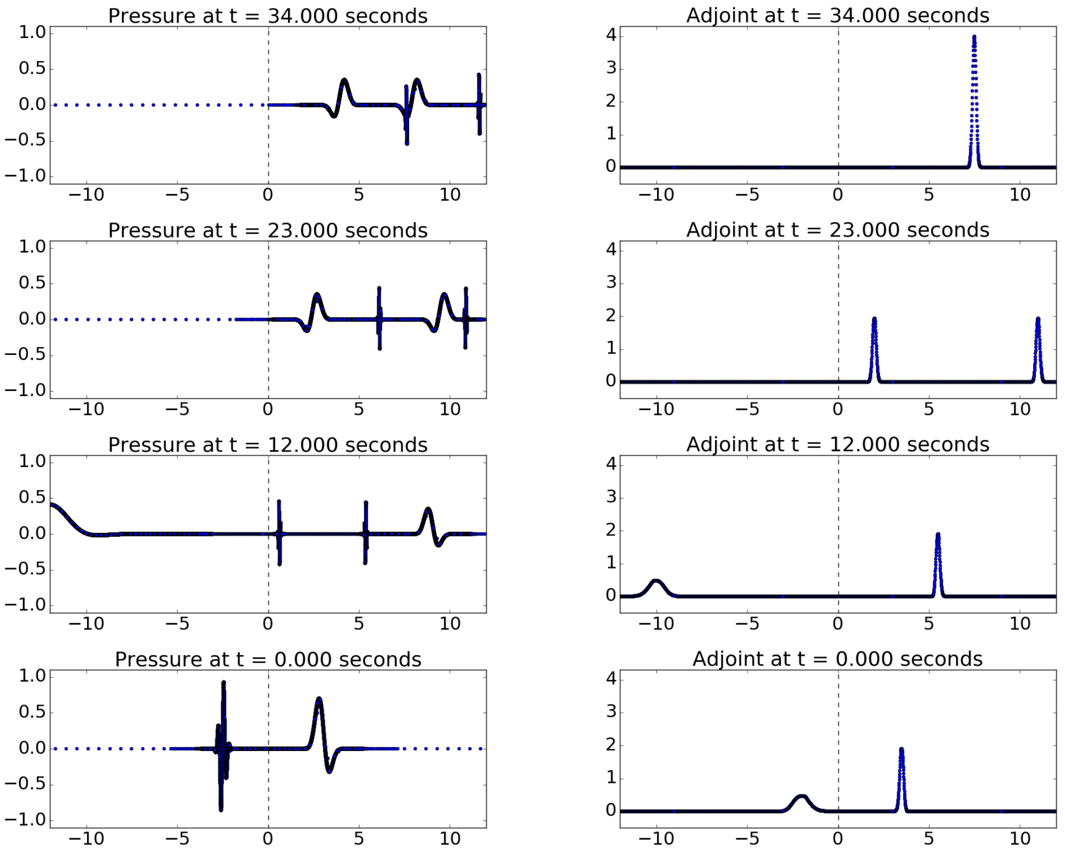


Figure 7.1: Two wave packets in the forward problem (on the left, with initial data at $t = 0$, in the bottom plot) and a Gaussian hump in the adjoint problem (on the right, with "initial" data at $t = 34$, in the top plot), interacting with an interface at $x = 0$ (dashed line). There is a change of sound speed at the interface, but the impedance is constant throughout the domain. The solutions are plotted as points, whose density varies due to AMR in the left plots.

As the "initial" data for this problem we set $\hat{q}(x, t_f) = \varphi(x)$. As time progresses backwards, the Gaussian hump splits into equal left-going and right-going waves which interact with the walls and the interface giving reflected waves off of the walls and transmitted waves through the interface. Figure 7.1 shows the AMRClaw results for the pressure of both the forward and adjoint solutions at four different times. Each data point represents the pressure (the $y$ axis) at the center of a grid point in space (the $x$ axis). The density of data points in $x$ is indicative of the resolution of the level of refinement in that location. For instance, in the top plot on the left side of fig. 7.1, the widely spaced data points on the left side of the domain indicate that the region is covered by a very coarse grid while the tightly spaced data points on the right side of the domain indicate that the region is covered by a very fine grid. Both the forward and adjoint problems are run using $t_f = 34$, so the forward problem is run from $t = 0$ to $t = 34$ and the adjoint problem is run from $t = 34$ to $t = 0$.

To better visualize how the waves are moving through the domain, it is helpful to look at the data in the $x$-$t$ plane as shown in fig. 7.2. For fig. 7.2, the horizontal axis is the position, $x$, and the vertical axis is time. The left plot shows in blue the locations where $1-$norm of $q(x,t)$ is greater than or equal to $10^{-2}$ and in red the locations where the $1-$norm of $\hat{q}(x,t)$ is greater than or equal to $10^{-2}$. The right plot shows in green the locations where the inner product $\hat{q}^T(x,t)q(x,t)$ is greater than or equal to $10^{-2}$. This indicates which portions of the wave in the forward problem will actually reach our point of interest, and are exactly the regions that will be refined when using the adjoint-magnitude flagging method.

### *7.1.2 Example 2: variable impedance in one dimension*

As a slightly more complex example, let $t_0 = 0$, $t_f = 52$, $a = -12$, $b = 12$,

$$\rho(x) = \begin{cases} 1 & \text{if } x < 0, \\ 2 & \text{if } x > 0, \end{cases} \quad \text{and} \quad K(x) = \begin{cases} 4 & \text{if } x < 0, \\ 1 & \text{if } x > 0, \end{cases}$$

so that

$$c(x) = \begin{cases} 2 & \text{if } x < 0 \\ 0.5 & \text{if } x > 0 \end{cases} \quad \text{and} \quad Z(x) = \begin{cases} 4 & \text{if } x < 0, \\ 0.5 & \text{if } x > 0. \end{cases}$$

In this case there is a jump in impedance at $x = 0$, which will result in there being both transmitted and reflected waves at the interface. As initial data for $q(x,t)$ we take two wave packets in pressure

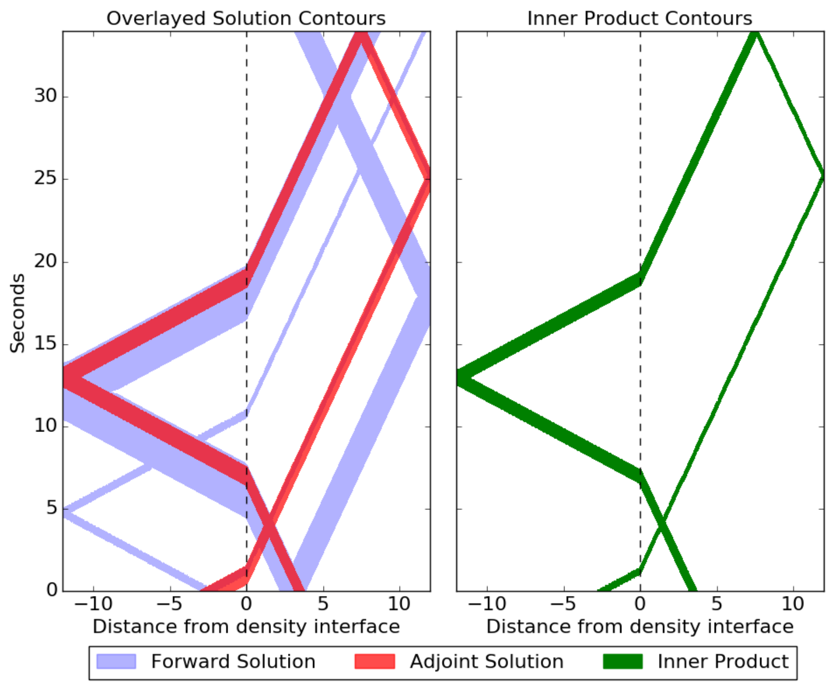


Figure 7.2: Space-time plots showing the forward and adjoint solutions from Example 1 overlaid in the left plot. Colored regions are where the magnitude of the first component of $q(x,t)$ or $\hat{q}(x,t)$ is above some threshold. The plot on the right shows regions where $|\hat{q}^T(x,t)q(x,t)|$ is above some tolerance, indicating where the forward problem should be refined if we are using adjoint-magnitude flagging. The time axis is the same for both plots.

one on each side of the interface, and zero velocity. The wave packets in pressure are the same as in example 1, and given by eq. (7.2). As time progresses, the wave packets split into equal left-going and right-going waves which interact with the walls and the interface giving reflected and transmitted waves. In contrast to example 1, many more waves arise in the solution as time evolves due to the generation of new waves at each reflection.

For this example we suppose that we are interested in the accurate estimation of the pressure around $x_p = 4.5$ at $t = 52$, chosen so that several waves in the forward problem must be resolved to accurately capture the solution (while many other waves do not need to be resolved; see fig. 7.3).

As "initial" data for the adjoint problem, $\hat{q}(x, t_f) = \varphi(x)$, we can use the same functional eq. (7.3) as in Example 1, with this value of $x_p$, and with $\alpha$, and $\hat{\beta}$ defined as in the previous example. As time progresses backwards in the adjoint solution, the hump splits into equal left-going and right-going waves that interact with the walls and the interface giving both reflected and transmitted waves. Both the forward and adjoint problems are run using $t_f = 52$, so the forward problem is run from $t = 0$ to $t = 52$ and the adjoint problem is run from $t = 52$ to $t = 0$.

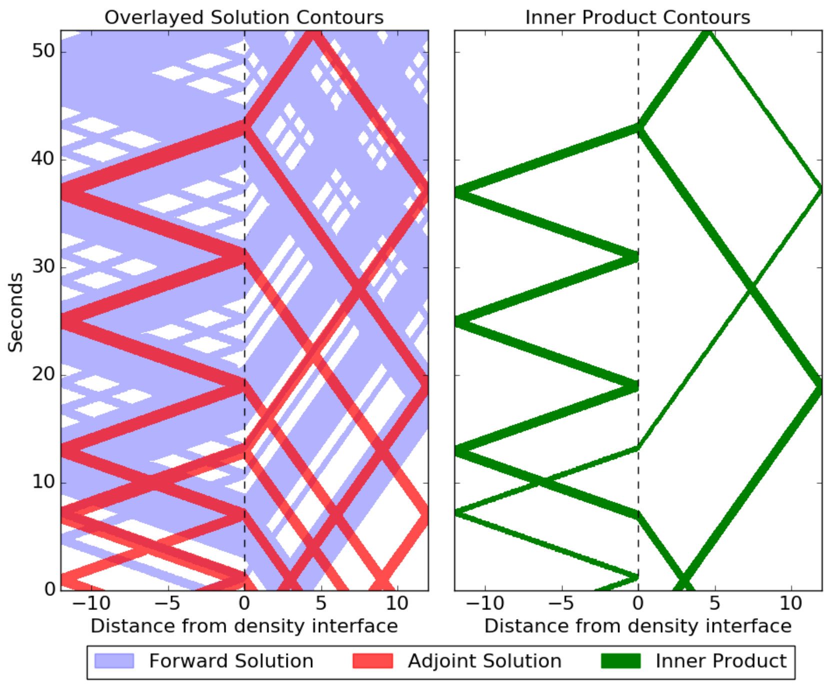


Figure 7.3: Space-time plots showing the forward and adjoint solutions from Example 2 overlaid in the left plot. Colored regions are where the magnitude of the first component of $q(x, t)$ or $\hat{q}(x, t)$ is above some threshold. The plot on the right shows regions where $|\hat{q}^T(x, t)q(x, t)|$ is above some tolerance, indicating where the forward problem should be refined if we are using adjoint-magnitude flagging. The time axis is the same for both plots.

We will again visualize how the waves are moving through the domain by looking at the data in the $x$-$t$ plane as shown in fig. 7.3. For fig. 7.3, the horizontal axis is the position, $x$, and the vertical axis is time. The left plot shows in blue the locations where 1−norm of $q(x,t)$ is greater than or equal to $10^{-5}$ and in red the locations where the 1−norm of $\hat{q}(x,t)$ is greater than or equal to $10^{-3}$. The right plot shows in green the locations where the inner product $\hat{q}^T(x,t)q(x,t)$ is greater than or equal to $10^{-3}$. This indicates which portions of the wave in the forward problem will actually reach our point of interest, and are exactly the regions that will be refined when using the adjoint-magnitude flagging method.

To better visualize the impact these different flagging methods have on the final solution, fig. 7.4 shows the final solution when using difference flagging and when using adjoint-magnitude flagging to achieve the same level of accuracy for our functional $J$. Similarly, fig. 7.5 shows the final solution when using error flagging and when using adjoint-error flagging to achieve the same level of accuracy in $J$. Note that the adjoint flagging methods have only focused on accurately capturing the waves at the location of interest, whereas the difference and error flagging techniques have captured all of the waves in the domain.

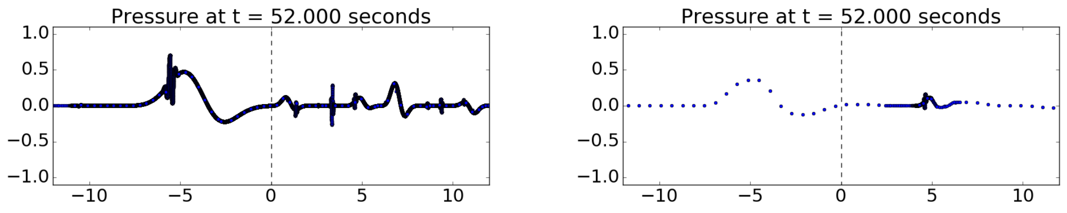


Figure 7.4: The final solution computed in example 2 when using two different refinement strategies. On the left: using difference-flagging. On the right: using adjoint-magnitude flagging, with a functional $J$ chosen to resolve only the waves that affect the solution near $x_p = 4.5$ at this time.

As an aid to visualizing how the different methods capture the solution at the region of interest, fig. 7.6 shows the solution for the four different flagging methods for a zoomed-in area around the region of interest. Difference-flagging is shown in blue, error-flagging is shown in red, adjoint-magnitude is flagging shown in green, and adjoint-error flagging is shown in black. Note that although all four solutions (from the four different flagging methods) are shown, they are almost

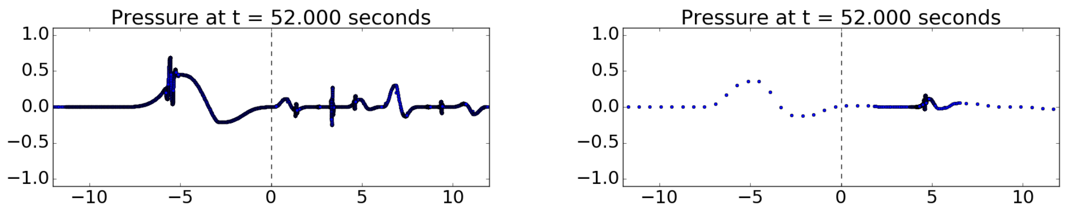


Figure 7.5: The final solution computed in example 2 when using two different refinement strategies. On the left: using error-flagging. On the right: using adjoint-error flagging, with a functional $J$ chosen to resolve only the waves that affect the solution near $x_p = 4.5$ at this time.

impossible to distinguish from one another in the figure. In this figure, only the solution on refinement levels 4 and 5 is shown. Note that the area of the domain that is resolved to refinement levels 4 and 5 is smaller for the adjoint-flagging methods than for their non-adjoint flagging counterparts, as expected.

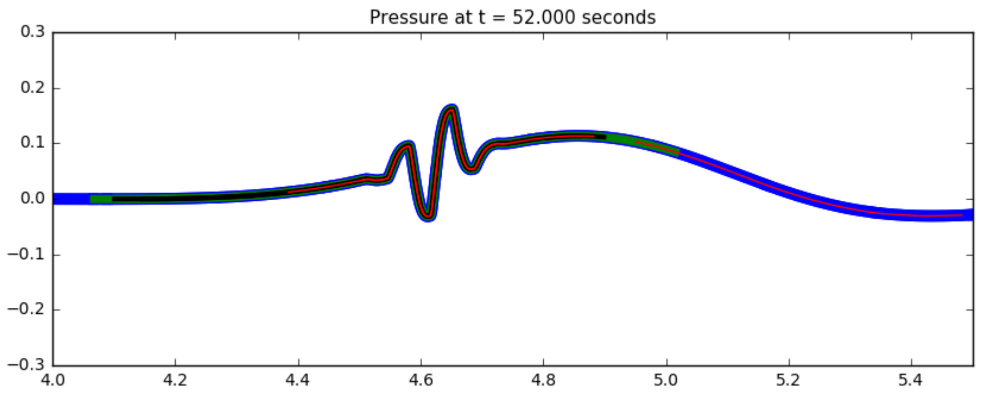


Figure 7.6: The final solution computed in example 2 for all four refinement strategies, in a zoomed in region of the domain. Difference-flagging shown in blue, error-flagging shown in red, adjoint-magnitude flagging shown in green, and adjoint-error flagging shown in black. Note that the lines are so overlapped that they are almost impossible to tell apart.

### *7.1.3 Computational performance*

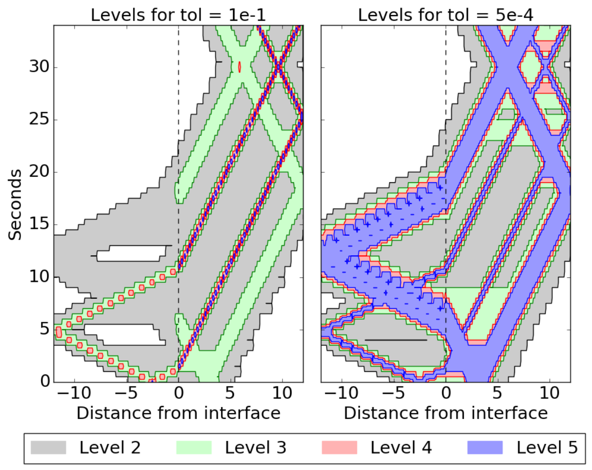
Recall that we are considering four different flagging methodologies: difference-flagging, error-flagging, adjoint-magnitude flagging, and adjoint-error flagging. To compare the results from these methods we will take into account

- the placement of AMR patches throughout the domain,
- the amount of CPU time required,
- the number of cell updates required, and
- the accuracy of the computed solution

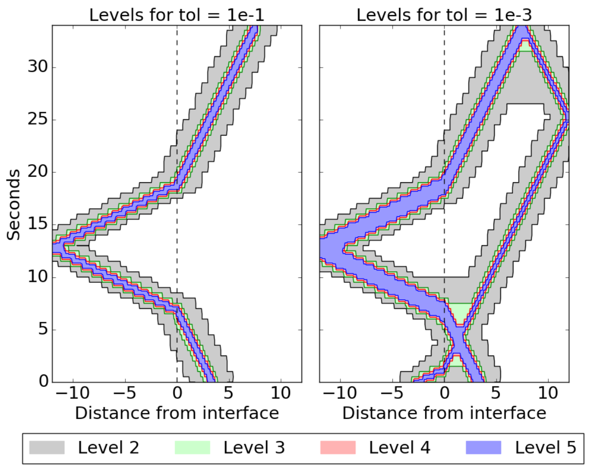
for each flagging method.

Let us begin by considering the placement of AMR levels throughout the domain. Since the adjoint-flagging methods take into account only the portions of the wave that will impact the region of interest during our time range of interest we expect that using adjoint-flagging will result in less of the domain being covered by fine resolution grids — where we are making the assumption that some of the waves will not ultimately have an effect on our region of interest.

In both of the examples we have presented this is in fact the case. Consider the contour plots shown in fig. 7.2 and fig. 7.3. If we are using difference-flagging then anywhere the forward solution is large (all of the areas in blue) would be flagged. In contrast, if we are using adjoint-magnitude flagging all of the areas where the inner product $\hat{q}^T(x,t)q(x,t)$ is large (all of the areas in green) would be flagged. Therefore, more of the domain would be flagged when using difference-flagging, resulting in more of the domain being covered by finer levels of refinement. For both of these one-dimensional examples a total of five levels of refinement are used for the forward problem, starting with 40 cells on the coarsest level and with a refinement ratio of 6 from each level to the next. So, the finest level in the forward problem corresponds to a fine grid with 51,840 cells. Where these finer levels of refinement are placed, of course, varies based on the flagging method being used. The adjoint problem was solved on a relatively coarse grid with 3000 cells, and no adaptive mesh refinement.



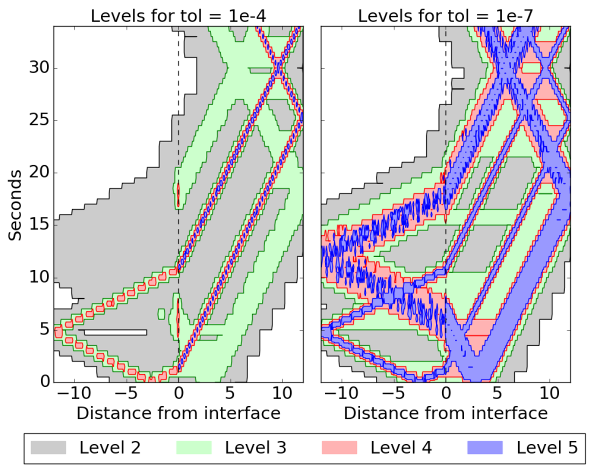
(a) Levels generated by using difference-flagging.



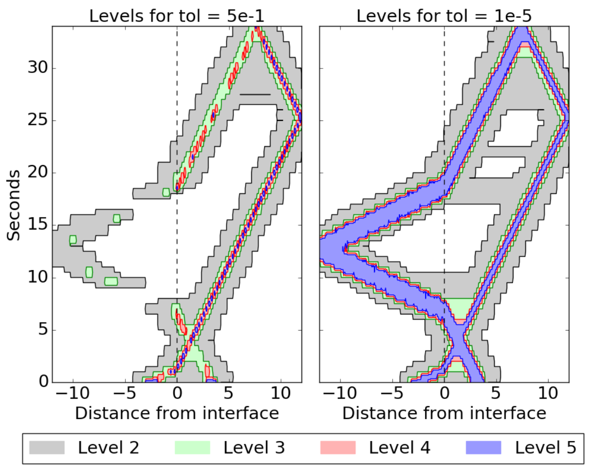
(b) Levels generated by using adjoint-magnitude flagging.

Figure 7.7: Levels used for the difference-flagging and adjoint-magnitude flagging methods for example 1. Tolerances used for the two difference-flagging runs shown are $10^{-1}$ and $5 \times 10^{-4}$. Tolerances used for the two adjoint-magnitude flagging runs shown are $10^{-1}$ and $10^{-3}$. A larger level number means greater resolution: so level 5 for example has a finer resolution than level 4. Recall that level 1 covers the entire domain, and is therefore not shown.

Figure 7.7 shows the levels of refinement for difference-flagging (on the left) and adjoint-magnitude flagging (on the right) for two different tolerances in example 1. The larger tolerance used for both of these flagging methods is $10^{-1}$ which yields a final error in the functional of interest of about $10^{-2}$ for both methods. The smaller tolerance used for difference flagging is $5 \times 10^{-4}$ and for adjoint-magnitude flagging is $10^{-3}$, both of which yield an error of about $10^{-6}$ in the functional of interest. Note that these two flagging methods both consider the value of the pressure in the cell when determining whether or not the cell should be flagged — difference-flagging flags the cell based on the relationship between the pressure in the current cell and the pressure in the



(a) Levels generated by using error-flagging.



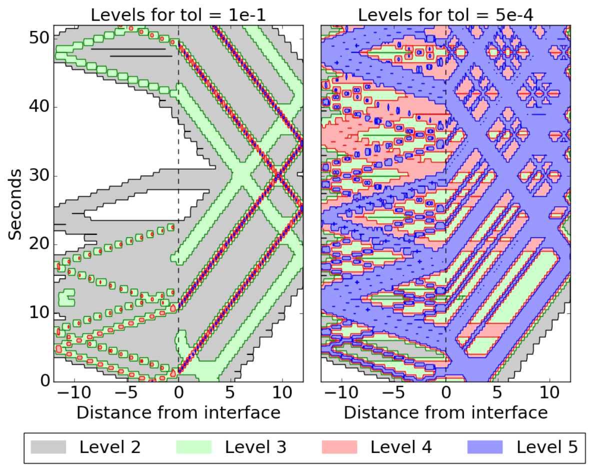
(b) Levels generated by using adjoint-error flagging.

Figure 7.8: Levels used for the error-flagging and adjoint-error flagging methods for example 1. Tolerances used for the two error-flagging runs shown are $10^{-4}$ and $10^{-7}$. Tolerances used for the two adjoint-error flagging runs shown are $5 \times 10^{-1}$ and $10^{-5}$. Recall that level 1 covers the entire domain, and is therefore not shown.

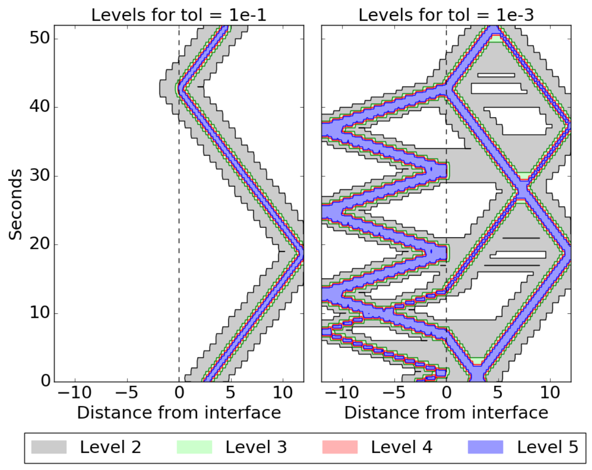
adjacent cells, and adjoint-magnitude flagging flags the cell based on the value of the inner product between $q(x,t)$ and $\hat{q}(x,t)$ in the cell. As expected, when the adjoint problem is taken into account the result is a smaller region of the domain begin refined to achieve a comparable level of error.

Similarly, fig. 7.8 shows the levels of refinement for error-flagging (on the left) and adjoint-error flagging (on the right) for two different tolerances in example 1. The larger tolerance used for error-flagging is $10^{-4}$ and for adjoint-error flagging is $5 \times 10^{-1}$, both of which yield a final error in the functional of interest of about $10^{-2}$. The smaller tolerance used for error-flagging is $10^{-7}$ and for adjoint-error flagging is $10^{-5}$, both of which yield an error of about $10^{-6}$ in the functional of interest. These two flagging methods consider the estimated error in the solution in each cell when determining whether or not the cell should be flagged. However, the adjoint-error flagging method takes the additional step of considering the inner product of this error estimate with the adjoint solution. Again, when the adjoint problem is taken into account the result is a smaller region of the domain being refined to achieve a comparable final error.

The tolerances shown were selected so that these figures would allow for easy comparison between the four methods. All of the larger tolerances chosen yield an error of about $10^{-2}$ in our functional of interest for each of the methods, and all of the smaller tolerances chosen yield an error of about $10^{-6}$ in our functional of interest. Therefore, it is appropriate to compare the amount of refinement levels present in these figures. Note that even though the same accuracy is achieved, the placement of regions of greater refinement varies between these four methods.



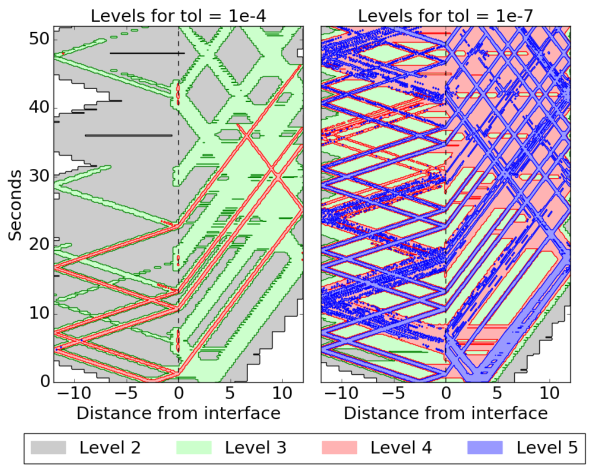
(a) Levels generated by using difference-flagging for example 2.



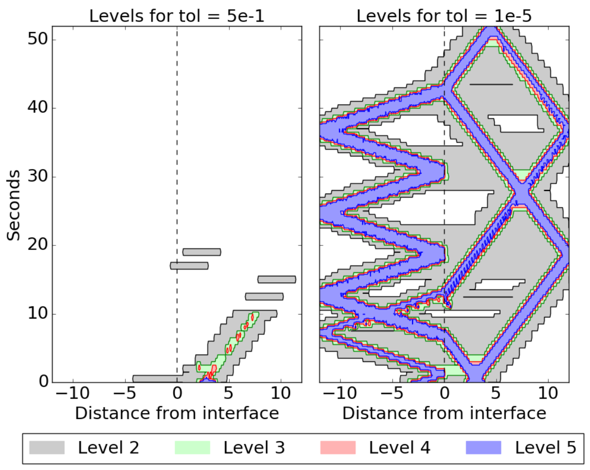
(b) Levels generated by using adjoint-magnitude flagging for example 2.

Figure 7.9: Levels used for the difference-flagging and adjoint-magnitude flagging methods for example 2. Tolerances used for the two difference-flagging runs shown are $10^{-1}$ and $5 \times 10^{-4}$. Tolerances used for the two adjoint-magnitude flagging runs shown are $10^{-1}$ and $10^{-3}$. Recall that level 1 covers the entire domain, and is therefore not shown.

For example 2, fig. 7.9 shows the levels of refinement for difference-flagging (on the left) and adjoint-magnitude flagging (on the right) using the same tolerances that were shown for example 1. Figure 7.10 shows the levels of refinement for error-flagging (on the left) and adjoint-error flagging (on the right) for example 2, where we have chosen the same tolerances that were shown



(a) Levels generated by using error-flagging for example 2.



(b) Levels generated by using adjoint-error flagging for example 2.

Figure 7.10: Levels used for the error-flagging and adjoint-error flagging methods for example 2. Tolerances used for the two error-flagging runs shown are $10^{-4}$ and $10^{-7}$. Tolerances used for the two adjoint-error flagging runs shown are $5 \times 10^{-1}$ and $10^{-5}$. Recall that level 1 covers the entire domain, and is therefore not shown.

for example 1. For this example we again see that when the adjoint problem is taken into account the result is a smaller region of the domain begin refined.

Also of interest is the amount of work that is required for each of the flagging methods we are considering, and the level of accuracy that resulted from that work. Figure 7.11 shows the error in our functional of interest for example 1 as the tolerance is varied for the four flagging methods we are considering. Difference-flagging was used with a tolerance of $10^{-12}$ to compute a very fine grid solution to our forward problem for this example. This solution was used to compute a fine grid value of our functional of interest, $J_{\text{fine}}$. The error between the calculated value for $J$ and $J_{\text{fine}}$ is shown.

Note that the magnitude of the error is nearly the same for a given tolerance when using either difference-flagging or adjoint-magnitude flagging although, as we will see, the amount of work

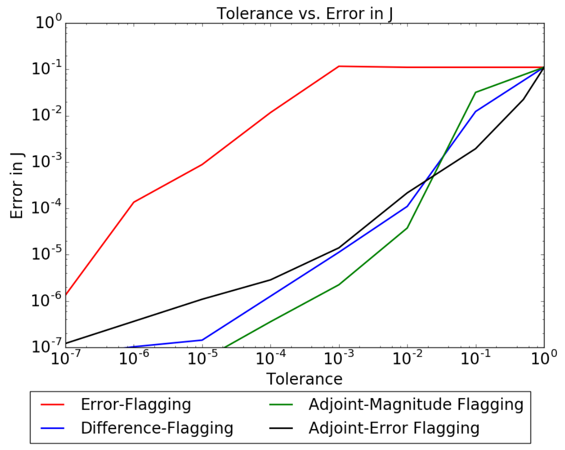


Figure 7.11: Accuracy for the different flagging methods on example 1. Shown is the error in the functional J for each tolerance value for the various flagging methods.

required is significantly less when using adjoint-flagging. Also note that the magnitude of the error when using adjoint-error flagging is consistently less than the magnitude of the tolerance being used, as we expected based on section 4.3. This means that adjoint-error flagging allows the user to enforce a certain level of accuracy on the final functional of interest by selecting a tolerance of the desired order. This capability has not existed in Clawpack previous to this work. Recall that the tolerance set by the user is being used to evaluate whether or not some given quantity is above that set tolerance. However, the quantity that is being evaluated is different for each of the flagging methods we are considering. Therefore, other than noting the general trends of each line in this figure, one cannot reach conclusions on relative merits by comparing one line with the others in this plot.

We are better able to compare the methods to one another if we consider the amount of computational time that is required to achieve a certain level of accuracy with each flagging method. The above examples were run on a quad-core laptop, and the OpenMP option of AMRClaw was enabled, which allowed all four cores to be utilized. Figure 7.12 shows two measures of the amount of CPU time that was required for each method vs. the accuracy achieved. On the left, we have

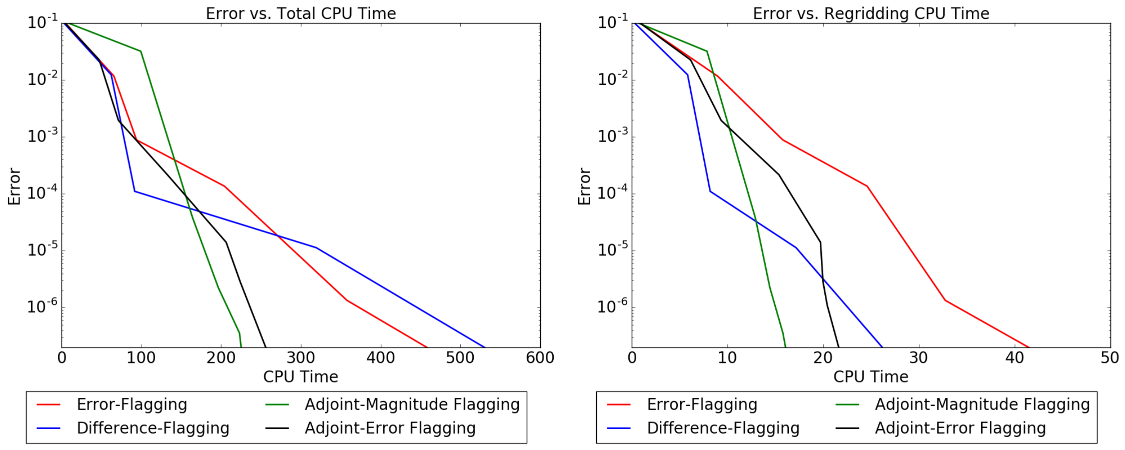


Figure 7.12: Performance measures for the different flagging methods on example 1. On the left: the total CPU time (in seconds) required vs. the accuracy achieved. On the right: the regridding CPU time (in seconds) required vs. the accuracy achieved. CPU times were found by averaging the CPU time over fifteen runs.

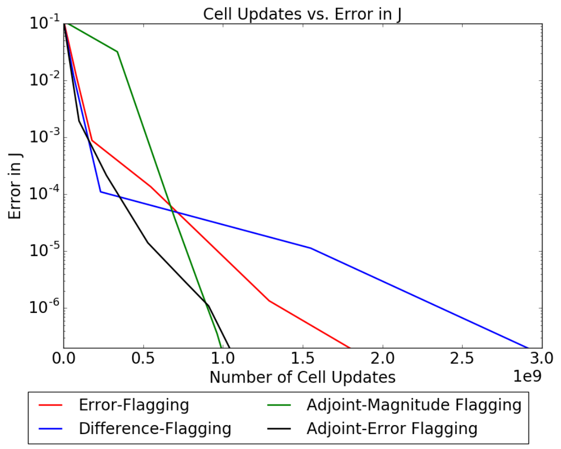


Figure 7.13: Number of cell updates calculated vs. the accuracy achieved for the four different flagging methods begin considered. Note that the number of cell updates axis is multiplied by 1e9.

shown the total amount of CPU time used by the computation. This includes stepping the solution forward in time by updating cell values, regridding at appropriate time intervals, outputting the results, and other various overhead requirements. For each flagging method and tolerance this example was run fifteen times, and the average of the CPU times for those runs was used in this plot. Note that while the adjoint-error and adjoint-difference flagging have higher CPU time requirements for lower accuracy, these two methods quickly show their strength by maintaining a low CPU time requirement while increasing the accuracy of the solution. In contrast, the CPU time requirements for difference-flagging and error-flagging quickly increase when increased accuracy is required. Also remember that the adjoint-flagging methods do have the additional time requirement of solving the adjoint problem, which is not shown in these figures. For this example, solving the adjoint required about 10 seconds of CPU time, which is small compared to the time spent on the forward problem. This CPU time was found by taking the average time required over ten simulations.

The right of fig. 7.12 shows the amount of CPU time spent on the regridding process vs. the accuracy of the functional $J$. Note that this is really where the differences between the four methods lie: the adjoint-flagging methods goal is to reduce the number of cells that are flagged while maintaining the accuracy of the solution. While we have seen from the left side of fig. 7.12 that this has certainly been successful in reducing the overall time required for computing the solution, we might expect that the time spent in the actual regridding process might be longer for the adjoint-flagging methods. This is due to the fact that adjoint-flagging requires various extra steps when determining whether or not to flag a cell (recall, for example, that we need to calculate the inner product between the adjoint and either the forward solution or the estimated error in the forward solution). However, as we can see from this figure, the amount of time spent in the regridding process is not significantly greater for the adjoint-flagging methods. In fact, for higher levels of accuracy the adjoint-flagging methods required less time for the regridding process than their non-adjoint counterparts because there are fewer fine grids.

Another advantage of the adjoint-flagging methods is the fact that they have less memory requirements than the alternative flagging methods. As we have already stated, since the adjoint method allows us to safely ignore regions of the domain, there are fewer fine grids as observed from the fact that many fewer cell updates were required. Figure 7.13 shows the number of cell updates required vs. the accuracy achieved for each of the flagging methods being considered. Note that the number of cell updates is significantly less for the adjoint-flagging methods than for their non-adjoint counterparts. This reduces the memory requirements for the computation. For this relatively small example memory usage when utilizing the non-adjoint flagging method did not become an issue, but it can quickly become a constraint for larger computations (in particular for three dimensional problems).

## *7.2 Two-dimensional linear variable coefficient acoustics*

The linear acoustics equations in two dimension in a piecewise constant medium are given by eq. (6.7), and they can be written in the form

$$q_t(x,y,t) + A(x,y)q_x(x,y,t) + B(x,y)q_y(x,y,t) = 0$$

with $q(x,y,t)$, $A(x,y)$, and $B(x,y)$ given by eq. (6.8).

As an example, consider $t_0 = 0$, $t_f = 21$, $a = -8$, $b = 8$, $\alpha = -1, \beta = 11$,

$$\rho(x,y) \equiv 1, \qquad \text{and} \qquad K(x,y) = \begin{cases} 4 & \text{if } x < 0, \\ 1 & \text{if } x > 0, \end{cases}$$

so that

$$c(x,y) = \begin{cases} 2 & \text{if } x < 0, \\ 1 & \text{if } x > 0, \end{cases} \qquad \text{and} \qquad Z(x,y) = \begin{cases} 2 & \text{if } x < 0, \\ 1 & \text{if } x > 0. \end{cases}$$

We use wall boundary conditions on the top and both sides,

$$u(a,y,t) = 0, \quad u(b,y,t) = 0 \qquad\qquad t \geq 0, \tag{7.6}$$

$$v(x,\beta,t) = 0 \qquad\qquad t \geq 0, \tag{7.7}$$

and extrapolation boundary conditions on the bottom at $y = -1$. The impedance jump at $x = 0$ will result in reflected and transmitted waves emanating from this interface. Also, we will have reflected waves from the walls along both sides and the top of our domain.

As initial data for $q(x, y, t)$ we take a smooth radially symmetric hump in pressure, and a zero velocity in both $x$ and $y$ directions. The initial hump in pressure is given by

$$p(x,y,0) = \begin{cases} 3 + \cos\left(\pi\left(r - 0.5\right)/w\right) & \text{if } |r - 0.3| \leq w \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

with $w = 0.15$ and $r = \sqrt{(x - 0.5)^2 + (y - 1)^2}$. As time progresses, this hump in pressure will radiate outward symmetrically.

We will consider two different examples, one where there are just a few waves that intersect at our region of interest during our time range of interest, and a second where a larger number of waves contribute to our region and time range of interest.

### *7.2.1 Example 3: capturing a few intersecting waves*

Suppose that we are interested in the accurate estimation of the pressure in the area defined by a rectangle centered about $(x, y) = (1.0, 5.5)$ at the final time $t_f = 21$. The small rectangle in Figure 7.14 and later figures shows this region of interest. Then we can use the functional

$$J = \int_{0.68}^{1.32}\int_{5.26}^{5.74} p(x,y,t_f)dy\,dx,$$

which corresponds to eq. (4.2) with

$$\varphi(x,y) = \begin{bmatrix} I(x,y) \\ 0 \\ 0 \end{bmatrix}, \tag{7.9}$$

where

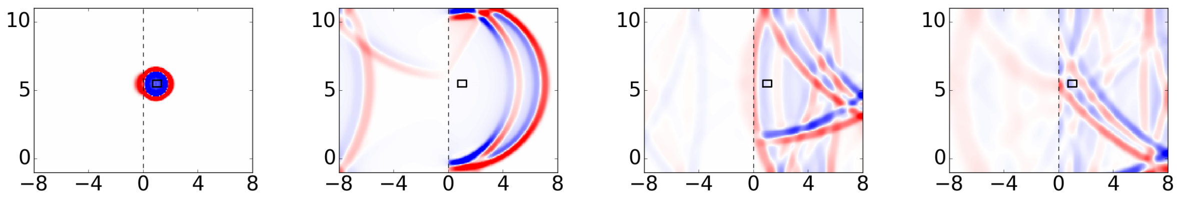
$$I(x,y) = \begin{cases} 1 & \text{if } 0.68 \leq x \leq 1.32 \text{ and } 5.26 \leq y \leq 5.74, \\ 0 & \text{otherwise.} \end{cases} \tag{7.10}$$

If we define the adjoint solution by solving

$$\hat{q}_t + \left(A^T(x,y)\hat{q}\right)_x + \left(B^T(x,y)\hat{q}\right)_y = 0 \qquad x \in [a,b], y \in [\alpha,\beta], \quad t_f \geq t \geq t_0 \tag{7.11}$$

with the same boundary conditions as the forward problem then the second and third terms in eq. (4.4) vanish and we are left with eq. (4.5), which is the expression that allows us to use the inner product of the adjoint and forward problems at each time step to determine what regions will influence the point of interest at the final time.

As the initial data for the adjoint $\hat{q}(x, y, t_f) = \varphi(x, y)$ we have a square pulse in pressure, which was described in eq. (7.9) and eq. (7.10). As time progresses backwards, waves radiate outward and reflect off the walls as well as transmitting and reflecting off of the interface at $x = 0$. The adjoint problem was solved on a $200 \times 200$ grid without mesh refinement, and several snapshots of the solution are shown in fig. 7.14. The location of interest is outlined with a black box in the plots.



(a) $t_f-$ 1 second (b) $t_f-$ 6 seconds (c) $t_f-$ 15 seconds (d) $t_f-$ 18.5 seconds

Figure 7.14: Computed results for two-dimensional acoustics adjoint problem for example 3. Times shown are the number of seconds before the final time, since the "initial" conditions are given at the final time. The small rectangle in this and later figures shows the region of interest defined by the functional $J$, and is the region where $I(x, y) = 1$. The color scale goes from blue to red, and ranges between $-0.005$ and $0.005$.

Although we have focused on the functional $J$, and are considering the accuracy and error in our estimates of this functional, in actuality we are typically interested in the accurate estimation of the pressure in the forward solution at a particular spatial and temporal area. The selection of the initial data for the adjoint plays a role in determining the functional $J$, and dictates the relationship between the solution of the forward problem and our functional of interest. Therefore, the initial data we select for the adjoint problem is undoubtedly significant. For this example we have selected initial conditions for the adjoint problem such that our functional $J$ is simply the integral of the forward solution over our region of interest. For the one-dimensional examples shown earlier, a Gaussian initial condition for the adjoint was used. An exploration of how these initial conditions affect the accuracy of the forward solution is not conducted in this work, but is an interesting area for future work.

Figure 7.15 compares the refinement levels used by each of the four different flagging methods on the forward problem for this example. Difference-flagging was used with a tolerance of $10^{-12}$ to compute a fine grid solution to our forward problem. The top row of the figure shows this fine grid solution, for the sake of reference. Each of the following rows shows the refinement patches (colored by level) being used in the simulation, from which we can quickly note that the adjoint-flagging methods have smaller regions of refinement than the other two flagging methods. The location of interest is outlined with a black box on all of the plots. For each of these flagging methods five levels of refinement are allowed for the forward problem, where the coarsest level is a $50 \times 50$ grid and a refinement ratio of 2 in both $x$ and $y$ is used from each grid to the next. So, the finest level in the forward problem corresponds to a fine grid of $1600 \times 1600$ cells. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. Where these finer levels of refinement are placed, of course, varies based on the flagging method being used. The adjoint problem was solved on a relatively coarse grid of $200 \times 200$ cells, and no adaptive mesh refinement. The tolerance used for the difference-flagging plots shown was $3 \times 10^{-2}$, for the adjoint-magnitude flagging plots the tolerance used was $3 \times 10^{-4}$, for the error-flagging plots the tolerance used was $6 \times 10^{-5}$, and for the adjoint-error flagging plots the tolerance used was $3 \times 10^{-3}$. These tolerances were chosen because they all resulted in an error of about $5 \times 10^{-4}$ in our functional $J$. Therefore, it is appropriate to compare the placement of the refined patches in the plots shown for all four flagging methods.

Note that, as in the one-dimensional examples, the new adjoint-based approaches allow refinement to occur only around the waves that eventually coalesce on the location of interest, with a region very close to the rectangle defined by the functional $J$ being refined at the final time $t_f = 21$.

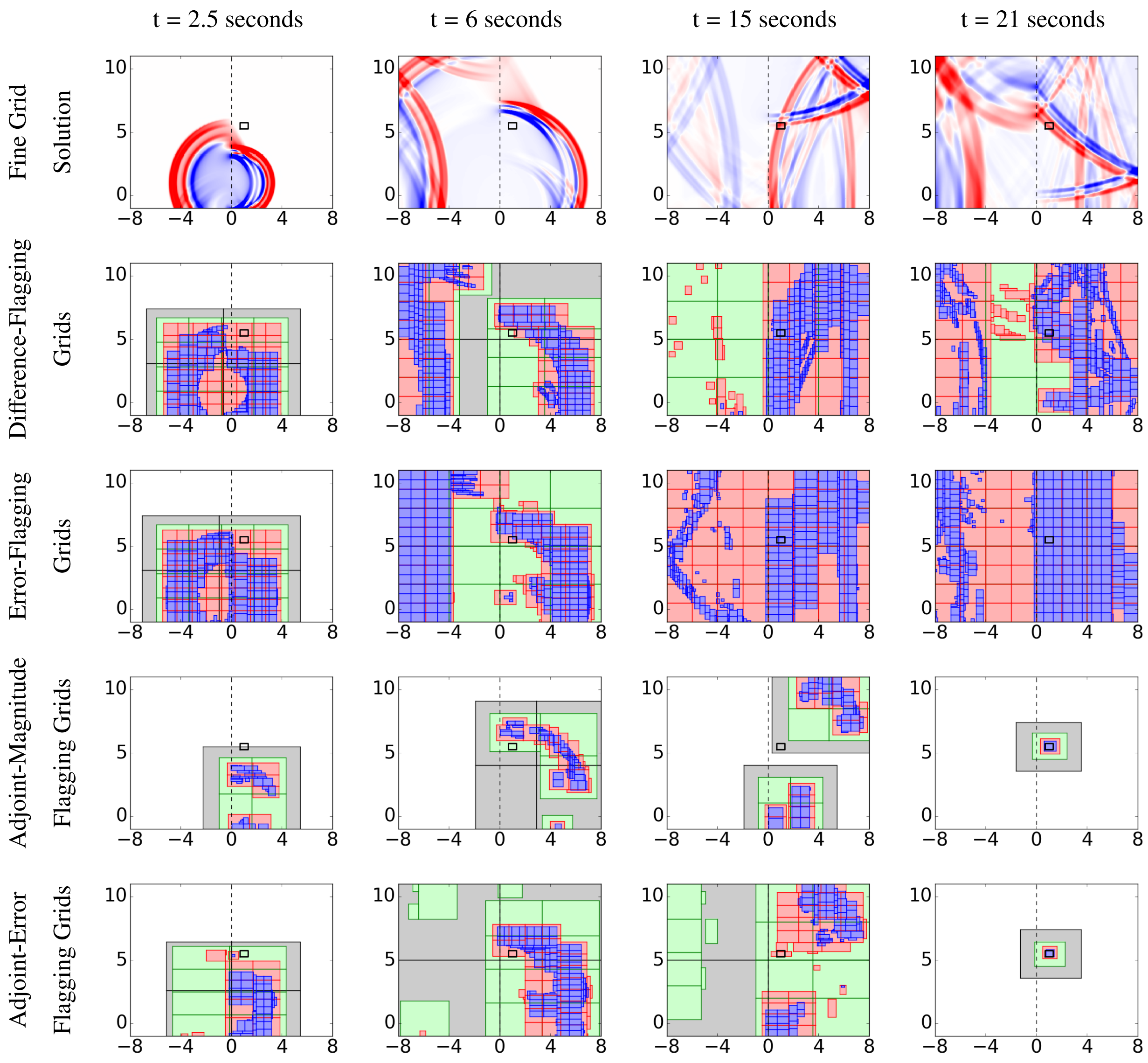


Figure 7.15: Plots for the two dimensional forward problem for example 3. The top row shows the fine grid solution, and the bottom four rows each show the grids for the different refinement levels being used for this problem when it is solved using one of the four flagging methods being presented. The color scale for the fine grid solution figures goes from blue to red, and ranges between −0.3 and 0.3. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively.

By contrast, the original difference-flagging and error-flagging approaches lead to refinement of many waves and portions of the domain that have no benefit in terms of determining the functional $J$ accurately.

### 7.2.2 *Example 4: capturing many intersecting waves*

How much of the domain must be refined with the adjoint approach depends on the specification of $J$. In this final example we simply move the location of interest to a place where more waves are converging at $t_f$ in order to illustrate that the adjoint approach will adapt to this situation. Suppose that we are now interested in the accurate estimation of the pressure in the area defined by a rectangle centered about $(x, y) = (3.5, 0.5)$, and define

$$J = \int_{3.18}^{3.82} \int_{0.26}^{0.74} p(x, y, t_f) dy\, dx.$$

For this example, the definition of the adjoint problem is the same as in example 3 except for the "initial data", for which we now take $\hat{q}(x, y, t_f) = \varphi(x, y)$ where $\varphi$ is as in eq. (7.9) with $I(x, y)$ now defined by

$$I(x, y) = \begin{cases} 1 & \text{if } 3.18 \le x \le 3.82 \text{ and } 0.26 \le y \le 0.74, \\ 0 & \text{otherwise.} \end{cases} \tag{7.12}$$

As time progresses backwards, waves radiate outward and reflect off the walls as well as transmitting and reflecting off of the interface at $x = 0$.

Figure 7.16 compares the refinement levels used by each of the two adjoint-flagging methods on the forward problem for this example. The top row of the figure shows the fine grid solution once again, for the sake of reference and to show the new location of interest. The new location of interest is outlined with a black box on all of the plots. As with fig. 7.15, each of the following rows shows the refinement patches (colored by level) being used in the simulation. For each of these flagging methods five levels of refinement are allowed for the forward problem, where the coarsest level is a $50 \times 50$ grid and a refinement ratio of 2 in both $x$ and $y$ is used from each grid to the next. So, as with example 3, the finest level in the forward problem corresponds to a fine grid

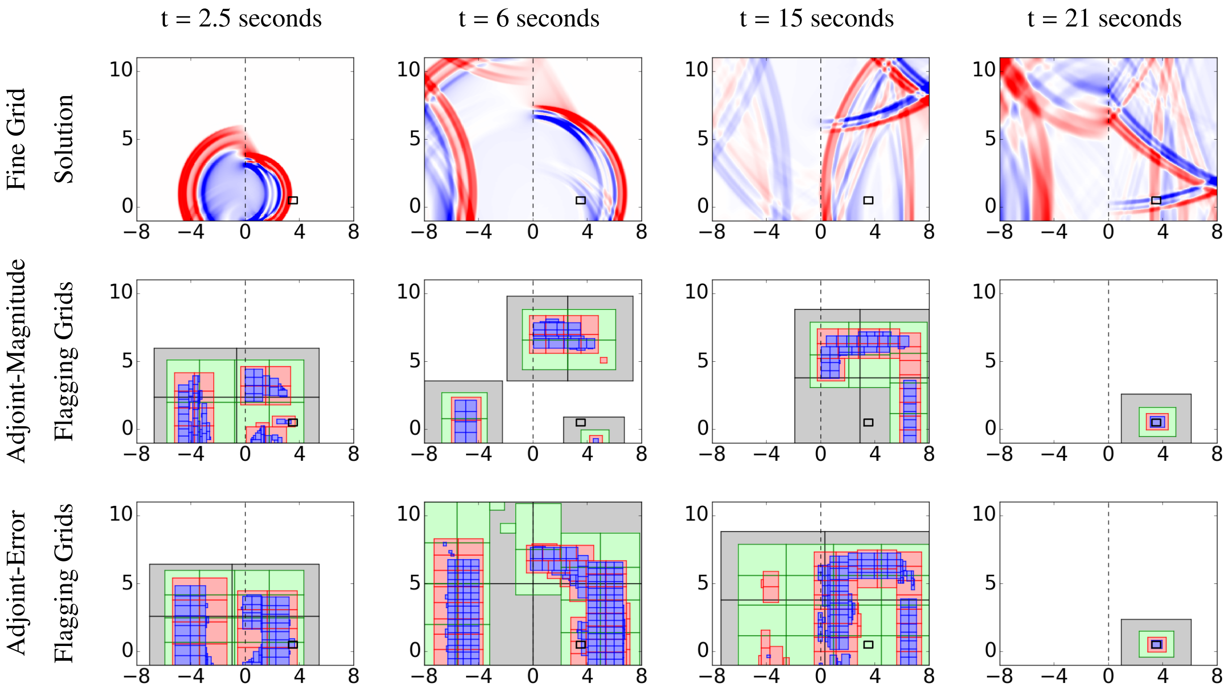


Figure 7.16: Plots for the two dimensional forward problem for example 4. The top row shows the fine grid solution, and the bottom two rows each show the grids for the different refinement levels being used for this problem when it is solved using one of the two adjoint-flagging methods. The color scale for the fine grid solution figures goes from blue to red, and ranges between −0.3 and 0.3. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively.

of $1600 \times 1600$ cells. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. Where these finer levels of refinement are placed, of course, varies based on the flagging method being used. The adjoint problem was again solved on a relatively coarse grid of $200 \times 200$ cells without adaptive mesh refinement.

The tolerance used for these plots are the same as the ones shown for example 3. Note that for the same tolerance the fine grid, difference flagging, and error flagging solutions are the same as in example 3, except for the placement of the gauge. Since none of these computations depended on the adjoint method, changing the area of interest (which changes the initial conditions for the adjoint problem) does not affect the computation. Since they would be the same as fig. 7.15, the figures for difference-flagging and error-flagging grids are not repeated in fig. 7.16.

### *7.2.3 Computational performance*

The advantage of using the adjoint method can be seen in the amount of work that is required for each of the flagging methods we are considering, and the level of accuracy that results from that work. Figure 7.17 shows the error in our functional of interest for the two-dimensional acoustics example 3 as the tolerance is varied for the four flagging methods we are considering. Our fine grid solution was used to calculate a fine grid value of our functional of interest, $J_{\text{fine}}$. The error between the calculated value of $J$, from the various tolerances used for each of the flagging methods, and $J_{\text{fine}}$ is shown.

Of particular interest, note that the magnitude of the error when using adjoint-error flagging is once again consistently less than the magnitude of the tolerance being used, as we expected based on section 4.3. This means that, as with example 1, adjoint-error flagging allows the user to enforce a certain level of accuracy on the final functional of interest by selecting a tolerance of the desired order. Recall that the tolerance set by the user is being used to evaluate whether or not some given quantity is above that set tolerance. However, the quantity that is being evaluated is different for each of the flagging methods we are considering. Therefore, other than noting the general trends in this figure, not much benefit comes from comparing each of the lines corresponding to a flagging method to the others in this plot.

We are better able to compare the methods to one another if we consider the amount of computational time that is required to achieve a certain level of accuracy with each flagging method. As with the one-dimensional examples, both of the two-dimensional examples was run on a quad-core laptop, and the OpenMP option of AMRClaw was enabled which allowed all four cores to be utilized. Figure 7.18 shows two measures of the amount of CPU time that was required for each method vs. the accuracy achieved for example 3. On the left, we have shown the total amount of CPU time used by the computation. This includes stepping the solution forward in time by

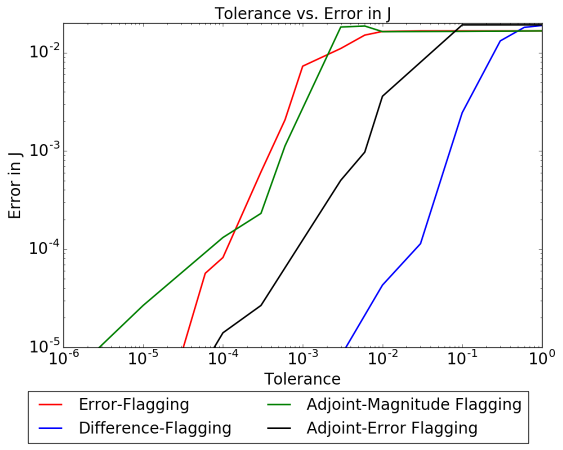


Figure 7.17: Accuracy for the different flagging methods on the two-dimensional acoustics example 3. Shown is the error in the functional $J$ for each tolerance value for the various flagging methods.

updating cell values, regridding at appropriate time intervals, outputting the results, and other various overhead requirements. For each flagging method and tolerance this example was run ten times, and the average of the CPU times for those runs was used to generate this plot. As with the one-dimensional example, while the adjoint-magnitude and adjoint-error flagging methods have a higher CPU time requirement for lower accuracy, these methods quickly showed their strength by maintaining a low CPU time requirement while increasing the accuracy of the solution. In contrast, the CPU time requirements for difference-flagging and error-flagging are larger than their adjoint-flagging counterparts when increased accuracy is required. Note that the adjoint-flagging methods do have the additional time requirement of solving the adjoint problem, which is not shown in these figures. For this example, solving the adjoint required about 37 seconds of CPU time, which is once again small compared to the time spent on the forward problem. As before, this CPU time was found by taking the average time required over ten simulations.

The right of fig. 7.18 shows the amount of CPU time spent on the regridding process vs. the accuracy of the functional $J$. Note that this is really where the differences between the four methods

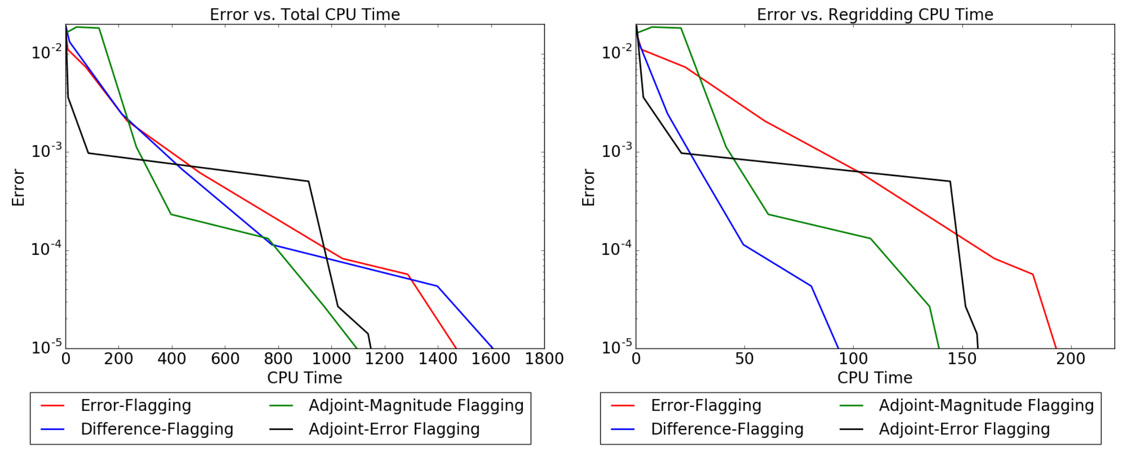


Figure 7.18: Performance measures for the different flagging methods on the two-dimensional acoustics example 3. On the left: the total CPU time (in seconds) required vs. the accuracy achieved. On the right: the regridding CPU time (in seconds) required vs. the accuracy achieved. CPU times were found by averaging the CPU time over ten runs.

lie: the adjoint-flagging methods goal is to reduce the number of cells that are flagged while maintaining the accuracy of the solution. While we have seen from the left side of fig. 7.18 that this has certainly been successful for adjoint-flagging in reducing the overall time required for computing the solution, we might expect that the time spent in the actual regridding process might be longer for the adjoint-flagging methods. However the amount of time spent in the regridding process is not significantly greater for the adjoint-flagging methods.

As with the one-dimensional example, here we see that the adjoint-flagging methods have less memory requirements than the alternative flagging methods. Figure 7.19 shows the number of cell updates required vs. the accuracy achieved for each of the flagging methods being considered. While the results are not as drastic as what we saw for example 1, note that the number of cell updates for example 3 is consistently less for the adjoint-flagging methods than for their non-adjoint counterparts, since there are fewer fine grids throughout the domain. This reduces the memory requirements for the computation. This example is also relatively small, and memory usage when

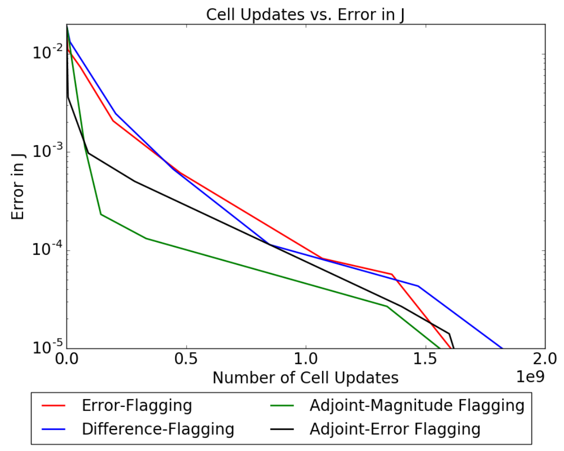


Figure 7.19: Number of cell updates calculated vs. the accuracy achieved for the four different flagging methods begin considered for example 3. Note that the number of cell updates axis is multiplied by 1e9.

utilizing the non-adjoint flagging method did not become an issue. However, in larger simulations (for instance, ocean-wide problems when considering tsunami modeling) memory constraints can become a significant consideration. It can also become a significant consideration in three dimensional problems.

# Chapter 8

# RESULTS FROM SHALLOW WATER EQUATIONS

We present several tsunami propagation examples utilizing the adjoint method to guide adaptive mesh refinement, as implemented in the GeoClaw software package. Verification of the results is performed by comparing the results from the adjoint method approach to the default *surface-flagging* approach already present in the GeoClaw software. In principle the adjoint flagging methodology could be used in conjunction with other adaptive mesh refinement tsunami models. Various other tsunami modeling codes exist, such as COMCOT [31], MOST [106, 108, 107], and ANUGA [8], but they offer constant nested refinement grids rather than using adaptive mesh refinement. A newer tsunami modeling package developed at the University of Bremen [97] does use adaptive mesh refinement, and uses triangular grids in contrast to Clawpack's logically rectangular grids.

In tsunami modeling we may wish to compute the sensitivity of the tsunami observed at our target location to changes in the data, e.g. to changes in the seafloor deformation if we are using a gradient-based optimization algorithm to solve an inverse problem to match observations (see [19] for one such application to tsunami source inversion). Or we may want to determine what potential source regions around the Pacific Rim give the largest tsunami response at a particular target location (such as Pearl Harbor, as considered in a study by Tang et al. [105]). Rather than solving many forward problems, this can be determined with a single adjoint solution. The adjoint method can also be used to refine only the waves that impact a target location during a pre-determined interval of time. This time interval of interest may be guided by tide gauge data (as is the case in the Japan 2011 tsunami discussed below).

The algorithms used in GeoClaw for tsunami modeling are described in detail in [71], and only a brief introduction will be given here. In general, GeoClaw solves the two-dimensional nonlinear shallow water equations in the form of a nonlinear system of hyperbolic conservation laws for depth and momentum. The numerical methods used are high-resolution Godunov-type finite volume methods, in which the discrete solution is given by cell averages of depth and momentum over the grid cells and the solution is updated in each time step based on fluxes computed at the cell edges. In Godunov-type methods, these fluxes are determined by solving a “Riemann problem” at each cell edge, which consists of the hyperbolic problem with piecewise constant initial data given by the adjacent cell averages. The general theory of these methods is presented in [69] and chapter 6, and details of the Riemann solver in GeoClaw can be found in [50]. Away from coastlines, this solver reduces to a Roe solver for the shallow water equations plus bathymetry, which means that the eigenstructure of a locally linearized Riemann problem is solved at each cell interface, making it no more expensive in the deep ocean than simply solving the linearized equations, but also capable of robustly handling nonlinearity near shore and inundation. Since the shallow water equations away from coastlines reduce to the linearized equations, a single adjoint equation (found by using the shallow water equations linearized about the ocean at rest) can be used. The algorithms and Riemann solvers for solving the adjoint problem for the shallow water equations are given in chapters 5 and 6. We consider only a few tsunami modeling examples in this thesis, but the adjoint method has already been used with GeoClaw in other publications. For examples, see [35, 21, 1]. It should be noted that adjoint-error flagging has not yet been implemented fully in GeoClaw. Therefore, in this chapter when we refer to adjoint-flagging we are speaking specifically about adjoint-difference flagging.

## *8.1 Hypothetical Alaska-Aleutian tsunami*

As an example, we consider a tsunami generated by a hypothetical earthquake on the Alaska-Aleutian subduction zone, the event denoted AASZe04 in the probabilistic hazard study of Crescent City, CA performed by [53]. Crescent City is of particular interest because it has been impacted by more damaging tsunamis in historic times than any other location on the west coast of the USA [36]. The major waves impinging on Crescent City from this hypothetical tsunami all occurred within 11 hours after the earthquake, so simulations will be run to this time. To simulate the effects of this tsunami on Crescent City a coarse grid is used over the entire Pacific (1 degree resolution) where the ocean is at rest. In addition to AMR being used to track propagating waves on finer grids, higher levels of refinement are allowed or enforced around Crescent City when the tsunami arrives. A total of 4 levels of refinement are used, starting with 1-degree resolution on the coarsest level, and with refinement ratios of 5, 6, and 6 from one level to the next. Only 3 levels were allowed over most of the Pacific, and the remaining level was used over the region around Crescent City. Level 4, with 20-second resolution, is still too coarse to provide any real detail on the effect of the tsunami on the harbor. It does, however, allow for a comparison of flagging cells for refinement using the adjoint method and using the default method implemented in Geoclaw. In this simulation we used 1 arc-minute bathymetry from the ETOPO1 Global Relief Model of [24] for the entire simulation area, as well as 1 arc-second and 1/3 arc-second bathymetry over the region about Crescent City from [54]. Internally, GeoClaw constructs a piecewise-bilinear function from the union of any provided topography files. This function is then integrated over computational grid cells to obtain a single cell-averaged topography value in each grid cell in a manner that is consistent between refinement levels.

Recall that the default Geoclaw refinement technique flags cells for refinement when the elevation of the sea surface relative to sea level is above some set tolerance, as described in [71], where the adaptive refinement and time stepping algorithms are described in more detail. We are referring to this flagging method as *surface-flagging*. The value selected for this tolerance has a significant impact in the results calculated by the simulation, since a smaller tolerance will result in more cells being flagged for refinement. Consequently, a smaller tolerance both increases the simulation time required and theoretically increases the accuracy of the results.

Two Geoclaw simulations were performed using surface-flagging, one with a tolerance of 0.14 and another with a tolerance of 0.09. Figure 8.1 shows the results of these two simulations, along with the grids at different levels of refinement used, for the sake of comparison. Note that each grid outlined in the figure, colored based on its level of refinement, is a collection of cells at a particular refinement. The grids and solution along the left of the figure correspond to the simulation with a tolerance of 0.14, and the grids and solution along the right of the figure correspond to the

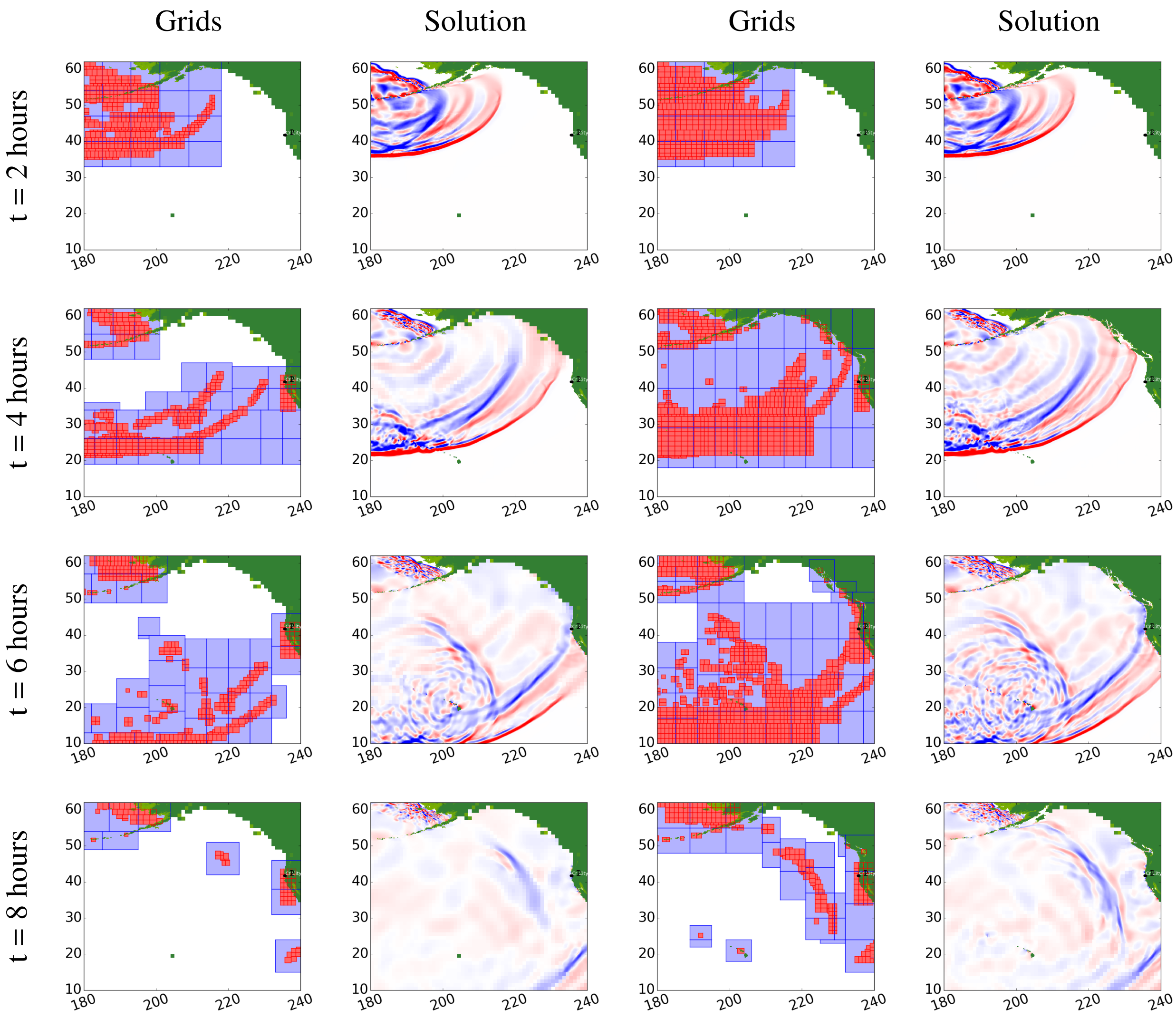


Figure 8.1: Computed results for tsunami propagation problem on two different runs utilizing the surface-flagging technique. The *x*-axis and *y*-axis are latitude and longitude, respectively. The grids and solution along the left correspond to the simulation with a tolerance of 0.14, and the grids and solution along the right correspond to the simulation with a tolerance of 0.09. In the grid figures each color corresponds to a different level of refinement: white for the coarsest level, blue for level two, and red for level three. The color scale for the solution figures goes from blue to red, and ranges between −0.3 and 0.3 meters (surface elevation relative to sea level). Times are in hours after the earthquake.

simulation with a tolerance of 0.09. Note that the simulation with a surface-flagging tolerance of 0.14 continues to refine the first wave until it arrives at Crescent City about 5 hours after the earthquake, but after about 6 hours stops refining the main secondary wave which reflects off the Northwestern Hawaiian (Leeward) Island chain before heading towards Crescent City. The second simulation, with a surface-flagging tolerance of 0.09, continues to refine this secondary wave until it arrives at Crescent City.

These two tolerances were selected because they are illustrative of two constraints that typically drive a Geoclaw simulation. The larger surface-flagging tolerance, of 0.14 was found to be approximately the largest tolerance that will refine the initial wave until it reaches Crescent City. Therefore, it essentially corresponds to a lower limit on the time required by the standard surface-flagging approach: any Geoclaw simulation with a larger surface-flagging tolerance would run more quickly but would fail to give accurate results for even the first wave. Note that for this particular example, even when the simulation will only give accurate results for the first wave to reach Crescent City, a large area of the wave front that is not headed directly towards Crescent City is being refined with the AMR. The smaller surface-flagging tolerance, of 0.09, refines all of the waves of interest that impinge on Crescent City thereby giving more accurate results at the expense of longer computational time, particularly since it also refines waves that will never reach Crescent City.

Now we consider the adjoint approach, which will allow us to refine only those sections of the wave that will affect our target region. For this example, we are interested in the accurate calculation of the water surface height ($\eta(x, y, t)$, defined in eq. (6.20)) in the area about Crescent City, California during the time range that is specified below. To focus on this area, we define a circle of radius $1^\circ$ centered about $(x_c, y_c) = (235.80917, 41.74111)$ where $x$ and $y$ are being measured in degrees. Setting

$$J = \int_{x_{min}}^{x_{max}} \int_{y_{min}(x)}^{y_{max}(x)} \eta(x, y, t_f) dy\, dx,$$

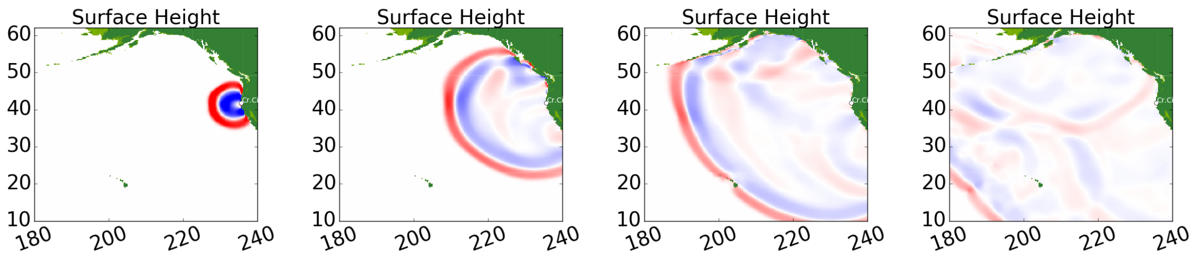
where the limits of integration define the appropriate circle, the problem then requires that

$$\varphi(x, y) = \begin{bmatrix} I(x, y) \\ 0 \\ 0 \end{bmatrix}, \tag{8.1}$$

where

$$I(x, y) = \begin{cases} 1 & \text{if } \sqrt{(x - x_c)^2 + (y - y_c)^2} \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{8.2}$$

This function $\varphi(x, y)$ defines the "initial data" $\hat{q}(x, y, t_f)$ for the adjoint problem. Figure 8.2 shows the results for the simulation of this adjoint problem. For this simulation a grid with 15 arcminute = 0.25° resolution was used over the entire Pacific and no grid refinement was allowed. The simulation was run out to 11 hours.



(a) $t_f - 1$ hour (b) $t_f - 3$ hours (c) $t_f - 5$ hours (d) $t_f - 7$ hours

Figure 8.2: Computed results for tsunami propagation adjoint problem. Times shown are the number of hours before the final time, since the "initial" conditions are given at the final time. The color scale goes from blue to red, and ranges between −0.05 and 0.05.

The topography files used for the adjoint problem are the same as those used for the forward problem. However, given that the adjoint problem is being solved on a coarser grid than the forward problem, the coastline between the two simulations varies. Since the coastline varies between the two simulations, when computing the inner product it is possible to find grid cells that are wet in the forward solution and dry in the adjoint solution. In this case, the inner product in those grid cells is set to zero.

The simulation of this tsunami using adjoint-flagging for the AMR was run using the same initial grid over the Pacific, the same refinement ratios, and the same initial water displacement as our previous surface-flagging simulations. The only difference between this simulation and the previous one is the flagging technique utilized. The first waves arrive at Crescent City around 4 hours after the earthquake, so we set $t_s = 3.5$ hours and $t_f = 11$ hours.

Figure 8.3 shows the Geoclaw results for the surface height at various different times, along with the grids at different levels of refinement that were used and the maximum inner product in the appropriate time range. Compare this figure to fig. 8.1, noting the extent of the grids at each refinement level for each of the three simulations.

### *8.1.1 Computational Performance*

The above example was run on a quad-core laptop, for both the surface-flagging and adjoint-flagging methods, and the OpenMP option of GeoClaw was enabled which allowed all four cores to be utilized. The timing results for the tsunami simulations are shown in Table 8.1. Recall that two simulations were run using surface-flagging, one with a tolerance of 0.14 (“Large Tolerance” in the table) and another with a tolerance of 0.09 (“Small Tolerance” in the table). Finally, a GeoClaw example using adjoint-flagging was run with a tolerance of 0.004 (“Forward” in the table), which of course required a simulation of the adjoint problem the timing for which is also shown in the table.

As expected, between the two GeoClaw simulations which utilized surface-flagging the one with the larger tolerance took significantly less time. Note that although solving the problem using adjoint-flagging did require two different simulations, the adjoint problem and the forward problem, the computational time required is only slightly more than the timing required for the large tolerance surface-flagging case. Another consideration when comparing the adjoint-flagging method with the surface-flagging method already in place in GeoClaw is the accuracy of the results. To test this, gauges were placed in the example and the output at the gauges compared across the

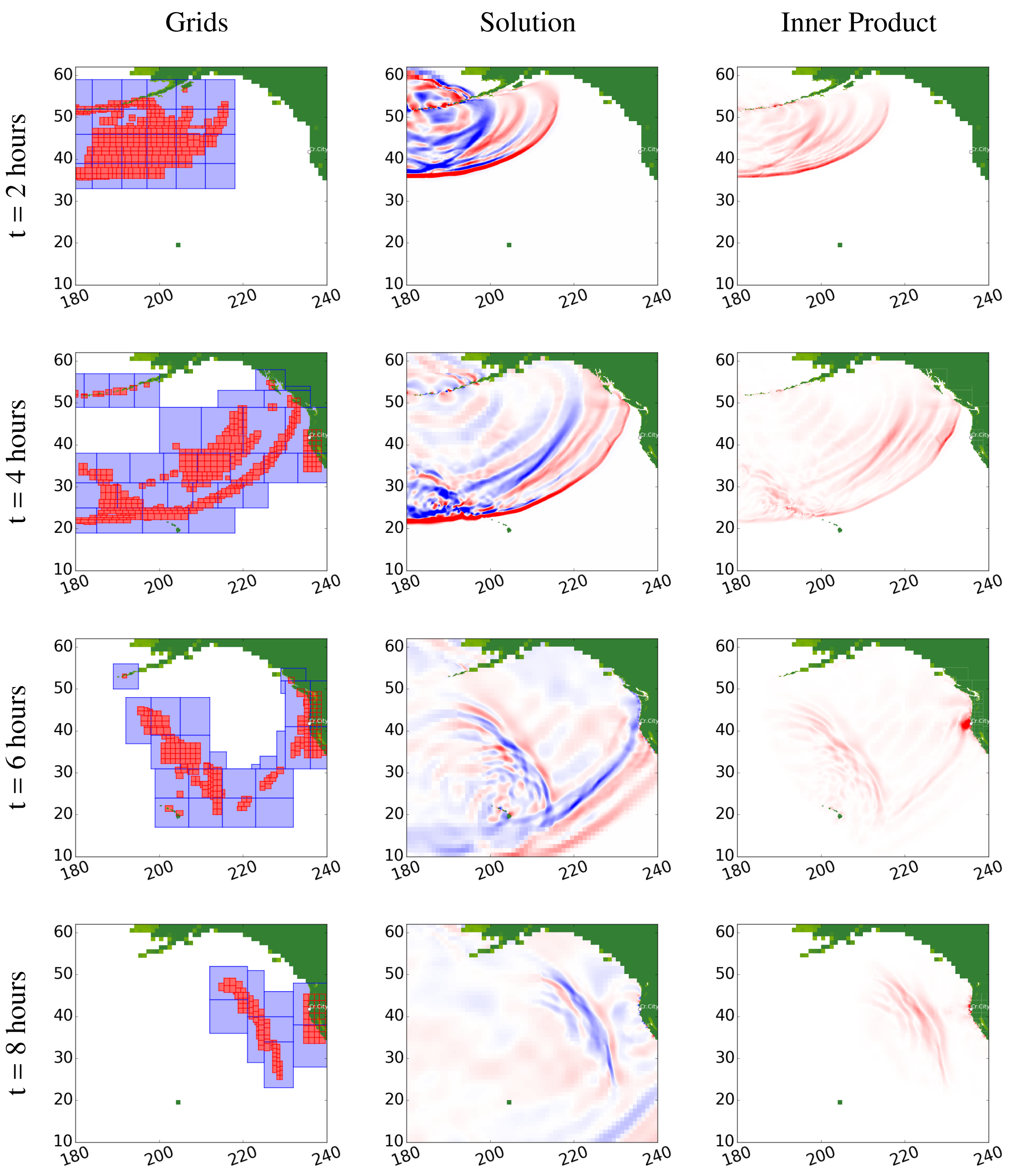


Figure 8.3: Computed results for tsunami propagation problem when adjoint-flagging is used. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: white for the coarsest level, blue for level two, and red for level three. The color scale for the surface height figures goes from blue to red, and ranges between $-0.3$ and 0.3. The color scale for the inner product figures goes from white to red, and ranges between 0 and 0.04. Times are in hours since the earthquake.

Table 8.1: Timing comparison for the example in Section 8.1.1 given in seconds.

| Surface-Flagging | | Adjoint-Flagging | |
|---|---|---|---|
| **Small Tolerance** | **Large Tolerance** | **Forward** | **Adjoint** |
| 8310.285 | 5724.461 | 5984.48 | 26.901 |

two different methods.

For the tsunami example two gauges are used: gauge 1 is placed at $(x, y) = (235.536, 41.67)$ which is on the continental shelf to the west of Crescent City, and gauge 2 is placed at $(x, y) = (235.80917, 41.74111)$ which is in the harbor of Crescent City. In fig. 8.4 the gauge results from the adjoint method are shown in blue, the results from the surface-flagging technique with a tolerance of 0.14 are shown in red, and the results from the surface-flagging technique with a tolerance of 0.09 are shown in green. Note that the blue and green lines are in fairly good agreement, indicating that the use of the adjoint method achieved a comparable accuracy with the smaller tolerance run using the surface-flagging method, although the time required was significantly less. While the larger tolerance run using the surface-flagging method had a similar time requirement to the adjoint method simulation, it agrees fairly well only for the first wave but then rapidly loses accuracy.

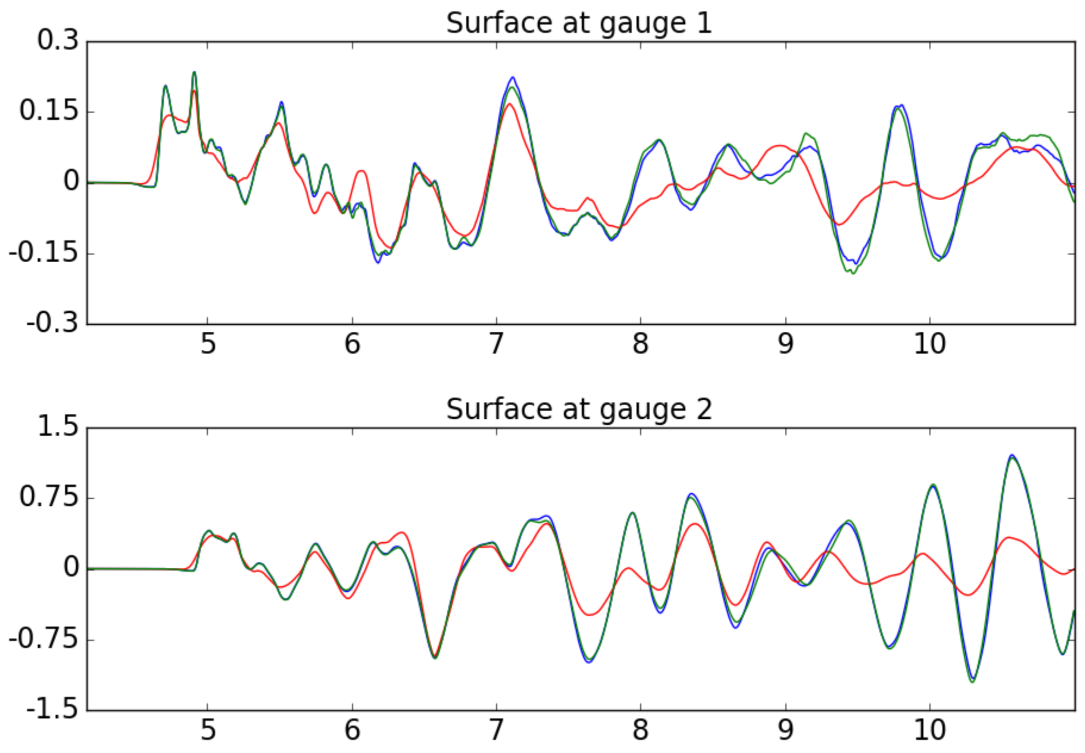


Figure 8.4: Computed results at gauges for tsunami propagation problem. The results from the simulation using the adjoint method are shown in blue, the results from the simulation using the surface-flagging method with a tolerance of 0.14 are shown in red, and the results from the simulation using the surface-flagging method with a tolerance of 0.09 are shown in green. Along the $x$-axis, the time since the occurrence of the earthquake is shown in hours.

## *8.2 Chile 2010 tsunami*

As another example illustrating the power of the adjoint method, consider the Chile 2010 tsunami. The impact that this (and other) tsunamis have had on Crescent City, CA has been the subject of a lot of research. For some examples on the studies focusing on the impact to Crescent City see [20, 113, 53]. Studies have also been done on the effects of harbor modification on mitigating Crescent City's vulnerability to tsunamis, for example see [36].

Recently, Adams and LeVeque [1] conducted a study comparing GeoClaw tsunami model results to detided tide gauge results at multiple destinations for each of several tsunamis. These results were also compared to MOST tsunami results, which is a depth-averaged long wave tsunami inundation model adapted by the National Oceanic and Atmospheric Administration (NOAA) for tsunami forecasting operations [106, 108, 107]. One of the tsunami and destination pairs considered was the Chile 2010 tsunami and its impact on Crescent City, CA. This particular example presented a great deal of difficulty for GeoClaw, due to the facts that

- the tsunami waves propagated across a large portion of the ocean, and
- significant waves were seen at Crescent City up to 26 hours after the earthquake.

In the preparation of [1], Adams spend a great deal of time and effort comparing the results from using adjoint-flagging and surface-flagging for this particular tsunami on a computer with 512 GB of memory. Six cores were used for the simulations discussed below.

*Advantage 1: Memory usage vs. user time.* After setting up a working adjoint-difference flagging model of this tsunami, the flagging method was switched to use the surface-flagging method which is currently the default in GeoClaw. Initially, no extra user effort was put into making sure that the regions allowing additional refinement were as small as possible to sufficiently resolve the wave (since this is not necessary when using adjoint-flagging). When this model was run, the simulation was only able to compute about 12 hours of tsunami propagation rather than the requested 26 hours because the simulation ran out of memory, due to the large portion of the ocean that was being refined. This first issue was resolved by a great deal of user time being spent on defining appropriately sized regions where greater refinement is allowed, and determining at what times those regions could be removed. With this extra user time and effort, the surface-flagging technique was able to produce results for the requested 26 hours of tsunami propagation time for this particular example. For details on the regions used for the surface-flagging case please see [1]. Anecdotally, Adams relied on visual inspection of the adjoint solution when determining where to place the regions where greater refinement is allowed — which speaks to the usefulness of the adjoint solution even when adjoint-flagging is not being used.

*Advantage 2: Computational time vs. accuracy of results.* In addition to the extra user time put into the surface-flagging method, the computational time required to run the simulation was about 90 hours of CPU time. In contrast, the adjoint method took 88 hours of CPU time to run. So, not only did the adjoint method require less user time but it also required less computational time. Because of the large number of grids that were enforcing refinement along the coast and near the target area, the computational time required for the adjoint does not offer as significant savings as we have seen for previous examples.

Moreover, even though the adjoint-flagging method took less time the accuracy of the computed results was retained. Figure 8.5 shows the results from surface-flagging and adjoint-difference flagging using GeoClaw for a gauge in Crescent City. Also shown are the observed de-tided tide gauge results, and the MOST computational results. Note that the results from the two GeoClaw runs are really on top of each other, showing that the results from the two flagging methods are nearly identical.

For the adjoint-flagging run shown in [1] a coarse grid was used over the entire Pacific (2 degree resolution) where the ocean is at rest. Up to three levels of refinement were allowed across the ocean, with refinement ratios of 5 and 6 from one level to the next. A fourth level of refinement was used, with a refinement ratio of 4 (so this level had 1 minute resolution), around Hawaii and some of Alaska. Other AMR regions were used to require certain levels of refinement around the source in Chile and around the target area in Crescent City. Please see [1] for the specifics of the regions used. The waves impinging on Crescent City between 14 hours and 26 hours after the earthquake were considered. When generating the surface-flagging simulation results that are

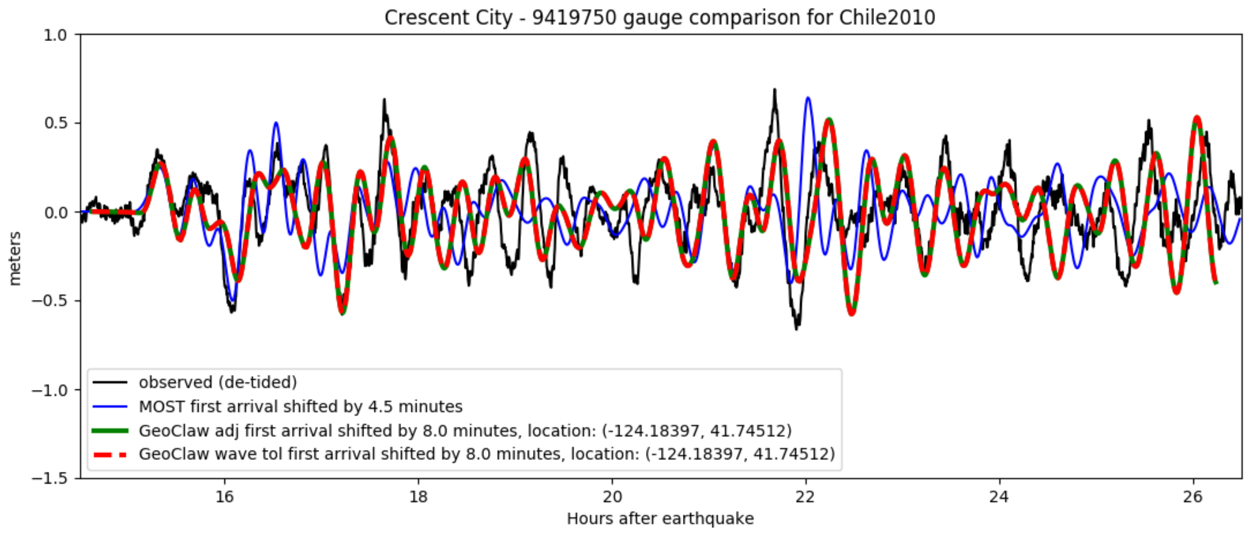


Figure 8.5: Compared results for surface-flagging and adjoint-flagging for the impact of the Chile 2010 tsunami on Crescent City, CA. Also show are the observed de-tided tide gauge results, and the MOST computational results. Please note the green and dashed red curves, which are really on top of each other, showing that the adjoint-flagging and surface-flaggings methods generate nearly identical results.

shown in fig. 8.5 only a single level of refinement (2 degree resolution) was allowed over the entire Pacific, and many regions were defined to allow refinement up to four levels of refinement in certain areas of the ocean. Again, other regions were used to require certain levels of refinement around the source in Chile and around the target area in Crescent City. A total of 8 levels of refinement are used, starting with 2-degree resolution on the coarsest level, and with refinement ratios of 5, 6, 4, 5, 2, 6, and 3 from one level to the next.

To better showcase the ability of the adjoint method to save user time, we ran this simulation again with even less restrictions on refinement regions. We allowed up to 4 levels of refinement across the entire ocean (so, up to 1 minute resolution), and up to 5 levels of refinement (12-second resolution) in water shallower than 100 feet. We used the same regions around Crescent City as used in [1], but modified them to allow (rather than require) refinement to occur. The only region of enforced refinement was around the Chile source, which forced the model to use 1-minute resolution around the source for the first four hours of tsunami propagation. In this simulation we used 1 arc-minute bathymetry from the ETOPO1 Global Relief Model of [24] for the entire simulation area, as well as 1 arc-second bathymetry and 1/3 arc-second bathymetry over the region about Crescent City from [54].

For this example, as mentioned above, we are interested in the accurate calculation of the water surface height in the area about Crescent City, California. To focus on this area, we define a Gaussian of width $\beta = 20$ meters centered about $(x_c, y_c) = (235.5661, 41.74512)$, where $x$ and $y$ are being measured in degrees. Setting

$$J = \int_{x_{min}}^{x_{max}} \int_{y_{min}(x)}^{y_{max}(x)} \varphi(x,y)\eta(x,y,t_f)dy\,dx, \tag{8.3}$$

where the limits of integration define the appropriate circle, we set

$$\varphi(x,y) = \begin{bmatrix} 0.1\exp(-(r/\beta)^2) \\ 0 \\ 0 \end{bmatrix}, \tag{8.4}$$

where $r = \text{haversine}(x, y, x_c, y_c)$ is the Haversine formula giving the distance in meters between the points $(x, y)$ and $(x_c, y_c)$. This function $\varphi(x, y)$ defines the "initial data" $\hat{q}(x, y, t_f)$ for the adjoint problem.

A figure for the adjoint problem results is not shown, because it appears nearly identical to the results shown in fig. 8.2. For this simulation a grid with 15 arcminute = 0.25° resolution was used over the entire Pacific, and a second region of refinement with a refinement ratio of 4 was required along the West Coast. The topography files used for the adjoint problem were the 1 arc-minute bathymetry from the ETOPO1 Global Relief Model of [24] for the entire simulation area.

For this example, we are concerned with the entire wave train that arrives at Crescent City. The first waves arrive around 14 hours after the earthquake, so we set $t_s = 14$ hours and $t_f = 26$ hours. Figures 8.6 and 8.7 show the GeoClaw results from running this problem using the adjoint-magnitude flagging method with a tolerance of $8 \times 10^{-6}$. The first column shows the surface

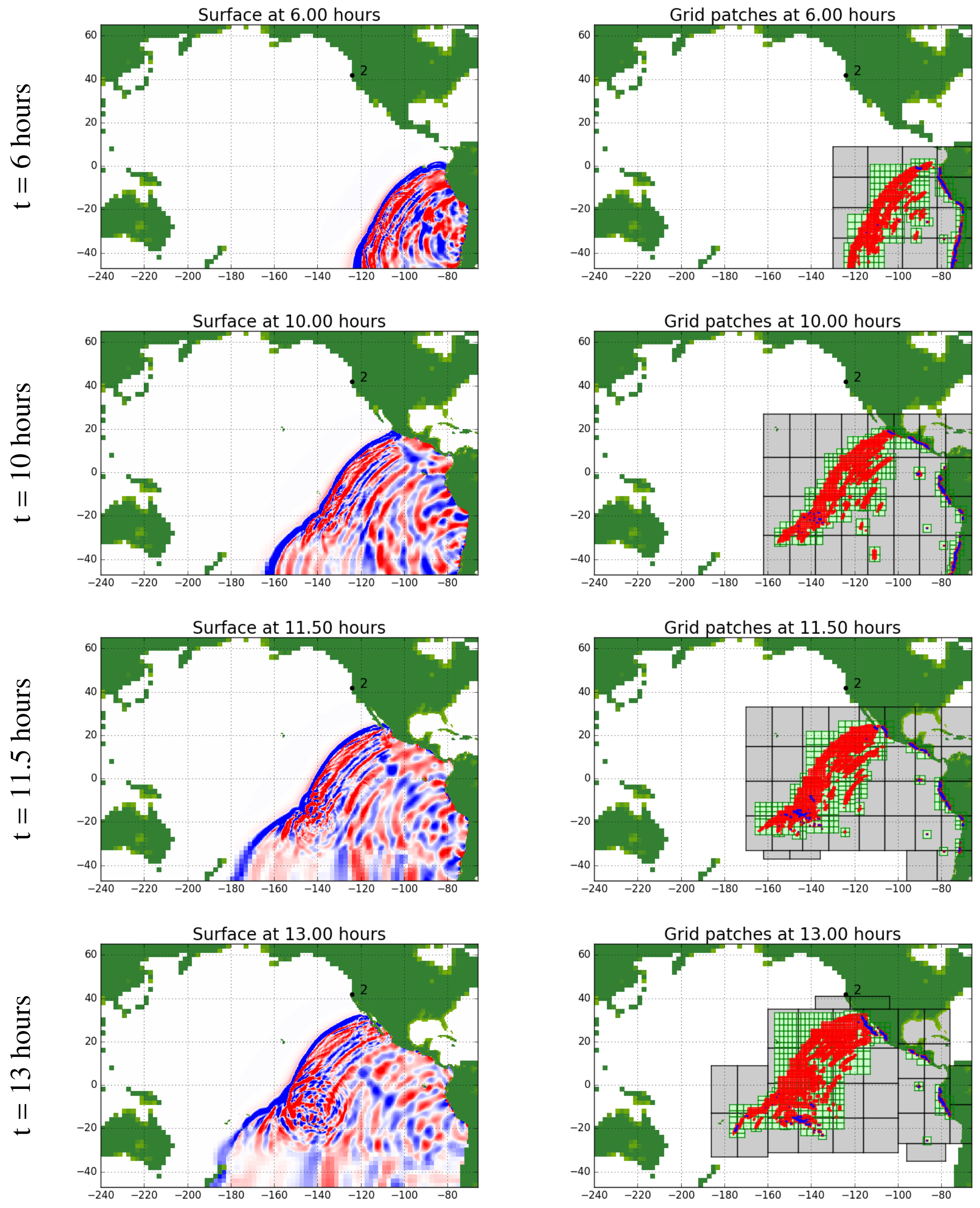


Figure 8.6: Computed results for Chile 2010 tsunami propagation problem when adjoint-flagging is used. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between −0.03 and 0.03. Times are in hours since the earthquake.

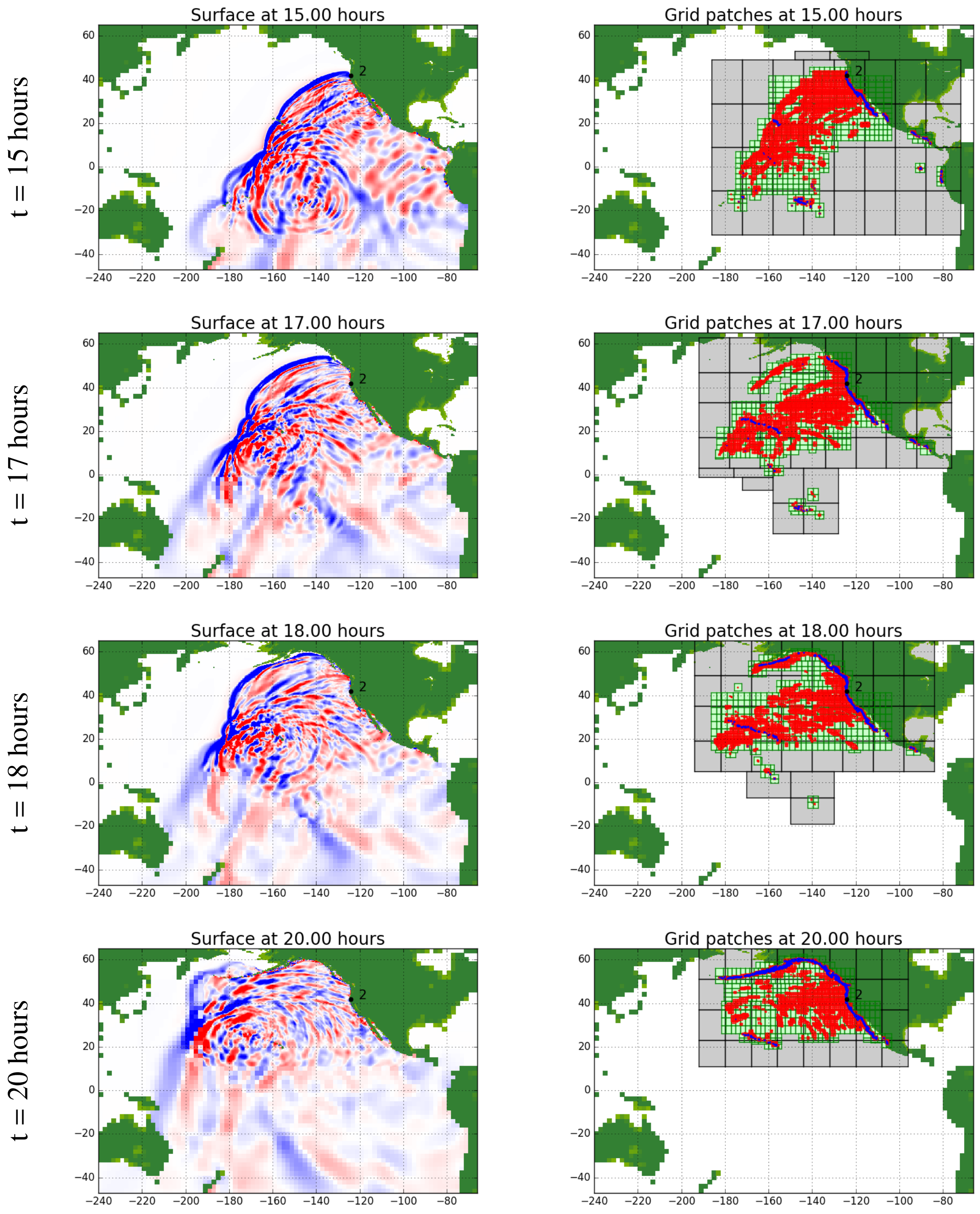


Figure 8.7: Computed results for Chile 2010 tsunami propagation problem when adjoint-flagging is used. The *x*-axis and *y*-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between −0.03 and 0.03. Times are in hours since the earthquake.

height of the water at various times and the second column shows the grids at different levels of refinement. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The sixth, seventh, and eighth levels of refinement, which are only allowed around Crescent City, are not shown but are nested within the fifth level of refinement.

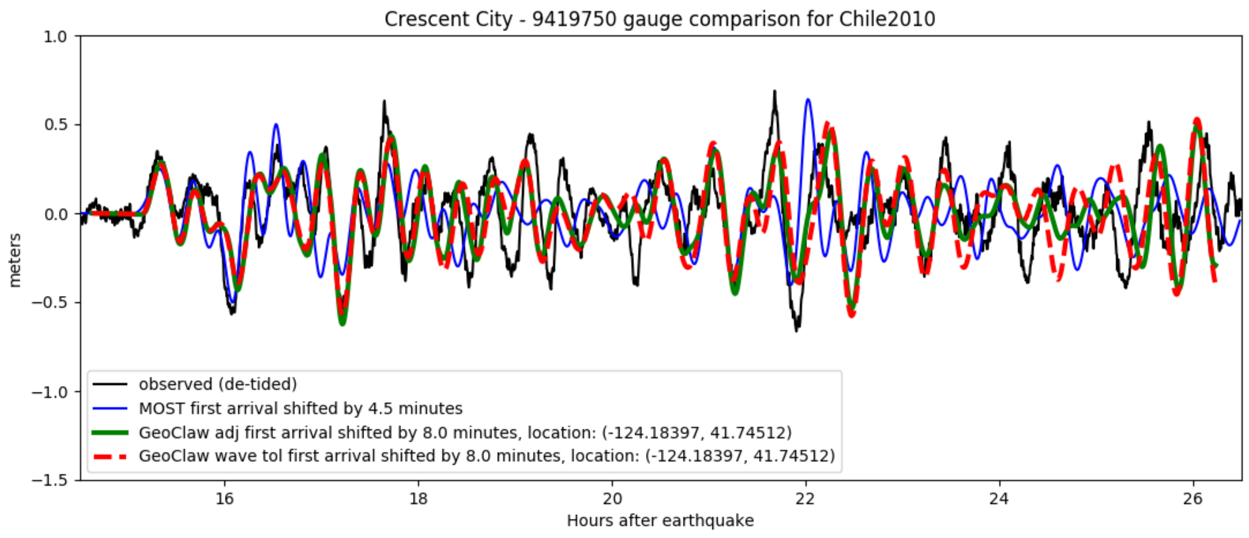


Figure 8.8: Compared results for surface-flagging and adjoint-flagging with limited numbers of regions of required refinement for the impact of the Chile 2010 tsunami on Crescent City, CA. Also show are the observed de-tided tide gauge results, and the MOST simulation results. Please note the green and dashed red curves, showing that the adjoint-flagging and surface-flaggings methods generate very similar results.

Figure 8.8 shows the gauge results from running this problem. In this figure the observed waves from a tide gauge are shown in black, the waves calculated by MOST are shown in blue, and the waves calculated by GeoClaw using surface-flagging are shown in red, and the waves calculated by GeoClaw using adjoint-difference flagging are shown in green. Note that the arriving waves are well captured by GeoClaw when we are using adjoint-flagging, even though there are very few regions allowing or forbidding increased refinement — the adjoint method is able to do all of that work for us and produce accurate results. There is some difference between the surface-flagging and adjoint-flagging results here, unlike the basically identical result we saw in fig. 8.5. This may indicate that regions requiring refinement are more important when trying to resolve later waves — since these waves are likely caused by edge waves propagating in the continental shelf, where the adjoint method is not as well equipped to guide AMR.

Given these results, we can see that the saving on memory restrictions and user time requirements that the adjoint method allows are substantial.

## *8.3 Japan 2011 tsunami*

Another tsunami that had a significant impact on Crescent City, CA was the March 11, 2011 tsunami that originated from Japan. This tsunami has been the subject of a lot of research, for both the near field [100, 60, 29, 28] and far field [56, 20, 113] effects. This tsunami is of particular interest in the far field because at many coastal time gauge stations the largest waves arrived several hours after the initial arrival of the tsunami waves [56]. As we will see from the tide gauge data presented below, this was true of tide gauges at Crescent City. With this example we will highlight another great strength of the adjoint method. Since we are able to define a time range of interest, we are able to selectively refine around only the waves that will influence our target location at particular points in time. Therefore, we are able to isolate and refine only the parts of the waves that contribute to the largest waves that arrive several hours after the initial tsunami waves. Isolating these parts of the waves allows us to study in more detail the behavior of the waves as they propagate across the ocean, and develop a greater understanding of the effect of the varied bathymetry and topography that waves encounter as they propagate across the ocean.

For this example we used the source deformation file used by [1] for the Japan 2011 event. We consider the waves impinging on Crescent City until about 13 hours after the earthquake, although the simulations are run out for a total of 15 hours. To simulate the effects of this tsunami on Crescent City a coarse grid is used over the entire Pacific (2 degree resolution) where the ocean is at rest. A total of 7 levels of refinement are used, starting with 2-degree resolution on the coarsest level, and with refinement ratios of 5, 6, 4, 5, 12, and 3 from one level to the next.

Generally, GeoClaw simulations use AMR to track propagating waves on finer grids and also enforce higher levels of refinement around the point of interest and in locations that the user believes need a higher level of refinement (through the use of user-defined *regions*). One of the main advantages of the adjoint method is that the method can determine what parts of the waves are significant for our area and time of interest. Therefore, a minimal number of regions was used for this example (which saved significant user time). A total of 4 levels of refinement (up to 1-minute of resolution) were allowed over most of the Pacific, and up to 5 levels of refinement (12-second resolution) were allowed in water shallower than 100 feet. Two regions were used around Crescent City, to allow up to 7 levels of refinement (with 1/3-arc second of refinement on the finest level). The only region of enforced refinement was around the Japan source, which forced the model to use 1-minute resolution around the source for the first hour of tsunami propagation. In this simulation we used 1 arc-minute bathymetry from the ETOPO1 Global Relief Model of [24] for the entire simulation area, as well as 1 arc-second bathymetry and 1/3 arc-second bathymetry over the region about Crescent City from [54].

For this example, as mentioned above, we are interested in the accurate calculation of the water surface height in the area about Crescent City, California. To focus on this area, we used the same adjoint solution as was used in section 8.2. Specifically, our functional of interest is given by eq. (8.3) and the initial conditions for the adjoint problem is a Gaussian defined by eq. (8.4).

To begin, let us consider the entire wave train that arrives at Crescent City. The first waves arrive around 9 hours after the earthquake, so we set $t_s = 8.5$ hours and $t_f = 15$ hours. Figures 8.9 and 8.10 shows the GeoClaw results from running this problem using the adjoint-magnitude flagging method with a tolerance of $1 \times 10^{-4}$. The first column shows the surface height of the water at various time and the second column shows the grids at different levels of refinement. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The sixth and seventh levels of refinement, which are only allowed around Crescent City, are not shown but are nested within the fifth level of refinement.

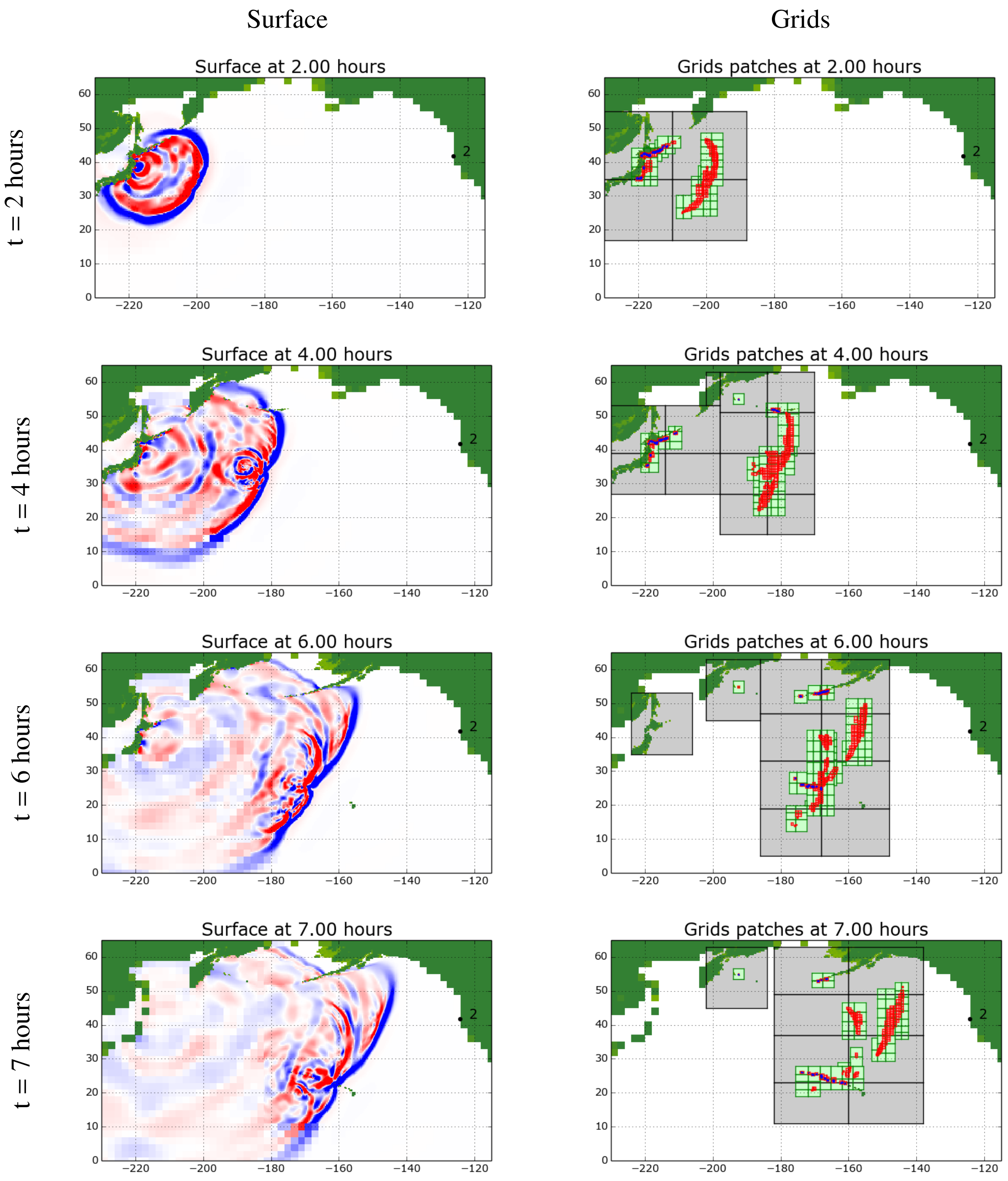


Figure 8.9: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 8.5$ and $t_f = 15$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between −0.1 and 0.1. Times are in hours since the earthquake.

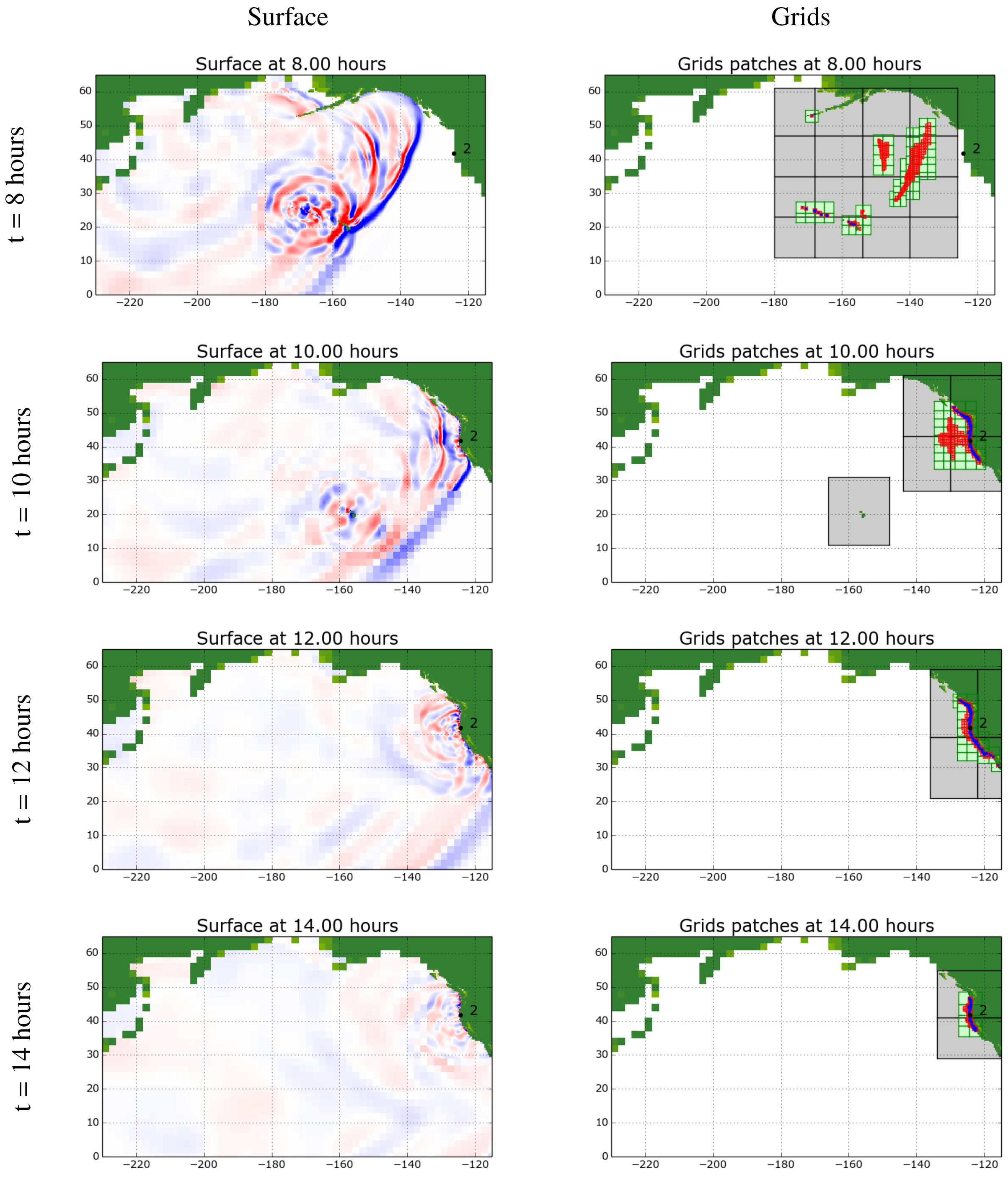


Figure 8.10: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 8.5$ and $t_f = 15$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between −0.1 and 0.1. Times are in hours since the earthquake.

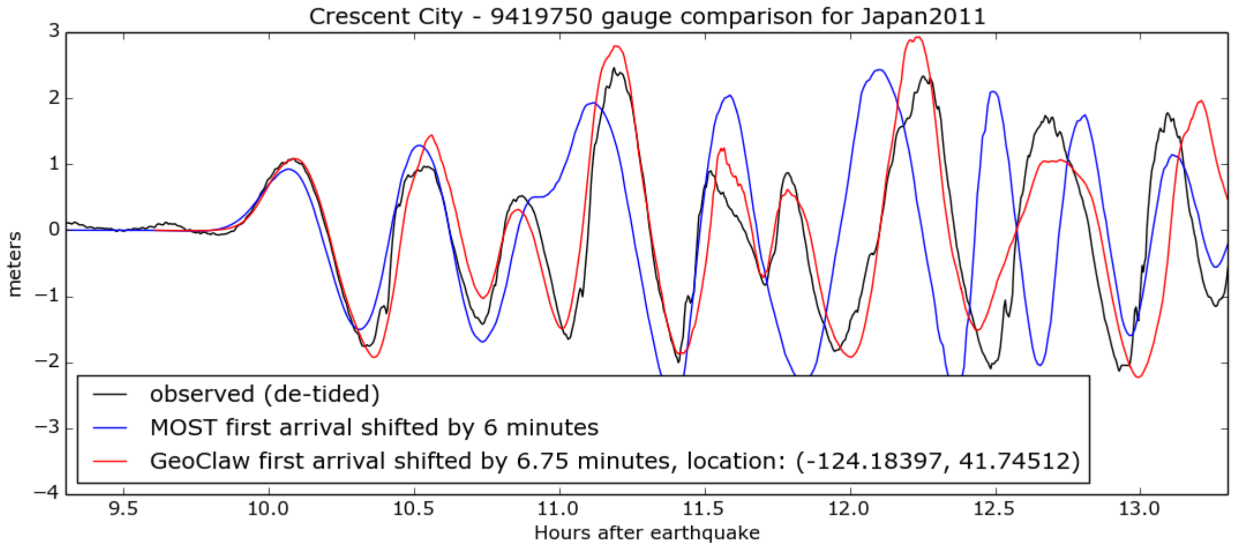


Figure 8.11: Results at the gauge for the Japan 2011 tsunami propagation problem with $t_s = 8.5$ hours and $t_f = 15$ hours. Along the $x$-axis, the time since the occurrence of the earthquake is shown in hours.

As can be noted from fig. 8.11, the wave that arrives at Crescent City a little after 11 hours after the earthquake and the wave that arrives between 12 and 12.5 hours after the earthquake are larger than the initially arriving waves. We are interested in isolating the parts of the wave that interact (e.g., with bathymetry as the waves travel across the ocean, or with the continental shelf in the form of edge waves) in such a way as to generate the large later waves.

First, we will focus on the wave that arrives a little after 11 hours post-quake. To focus on this wave we set $t_s = 10.75$ hours and $t_f = 11.3$ hours. Figures 8.12 and 8.13 shows the GeoClaw results from running this problem using the adjoint-magnitude flagging method with a tolerance of $3 \times 10^{-5}$. The first column shows the surface height of the water at various time and the second column shows the grids at different levels of refinement. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The sixth and seventh levels of refinement, which are only allowed around Crescent City, are not shown but are nested within the fifth level of refinement.

It is of interest to note from figs. 8.12 and 8.13 that although the Hawaii islands are resolved with the second level of refinement, the finer levels of refinement are not used to resolve the island chain. Therefore, the waves we saw reflecting off of Hawaii in figs. 8.9 and 8.10 do not play a significant role in the large wave that arrives at Crescent City a little after 11 hours post-quake.

Figure 8.14 shows the gauge results from running this problem with $t_s = 10.75$ hours and $t_f = 11.3$ hours. In this figure the observed waves from a tide gauge are shown in black, the waves calculated by MOST are shown in blue, and the waves calculated by GeoClaw are shown in red. Note that the waves that arrive during our region of interest are well captured by GeoClaw. In fact, the whole wave train that arrives before our region of interest is also captured very well. This is due to the fact that this early wave train becomes edge waves, which propagate within the continental shelf and combine to form the large wave that arrives at Crescent City a little after 11 hours post-quake. Note that after our time range of interest the calculated gauge results from GeoClaw are not accurate at all. This is because the simulation is no longer refining the area around Crescent City, as should be expected given that by selecting $t_f = 11.3$ we have stated that we are not interested in the behavior of the waves after this time.

We now focus on the wave that arrives a little after 12 hours post-quake. To focus on this wave we set $t_s = 11.75$ hours and $t_f = 12.5$ hours. Figures 8.15 and 8.16 shows the GeoClaw results from running this problem using the adjoint-magnitude flagging method with a tolerance of $3 \times 10^{-5}$. The first column shows the surface height of the water at various time and the second column shows the grids at different levels of refinement. For all of the refinement level plots the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The sixth and seventh levels of refinement, which are only allowed around Crescent City, are not shown but are nested within the fifth level of refinement.

It is of interest to note from figs. 8.15 and 8.16 that, in contrast to what we saw in figs. 8.12 and 8.13, here we have the Hawaiian islands being resolved with much finer levels of refinement. Therefore, in contrast to the large wave that arrives at Crescent City a little after 11 hours post-

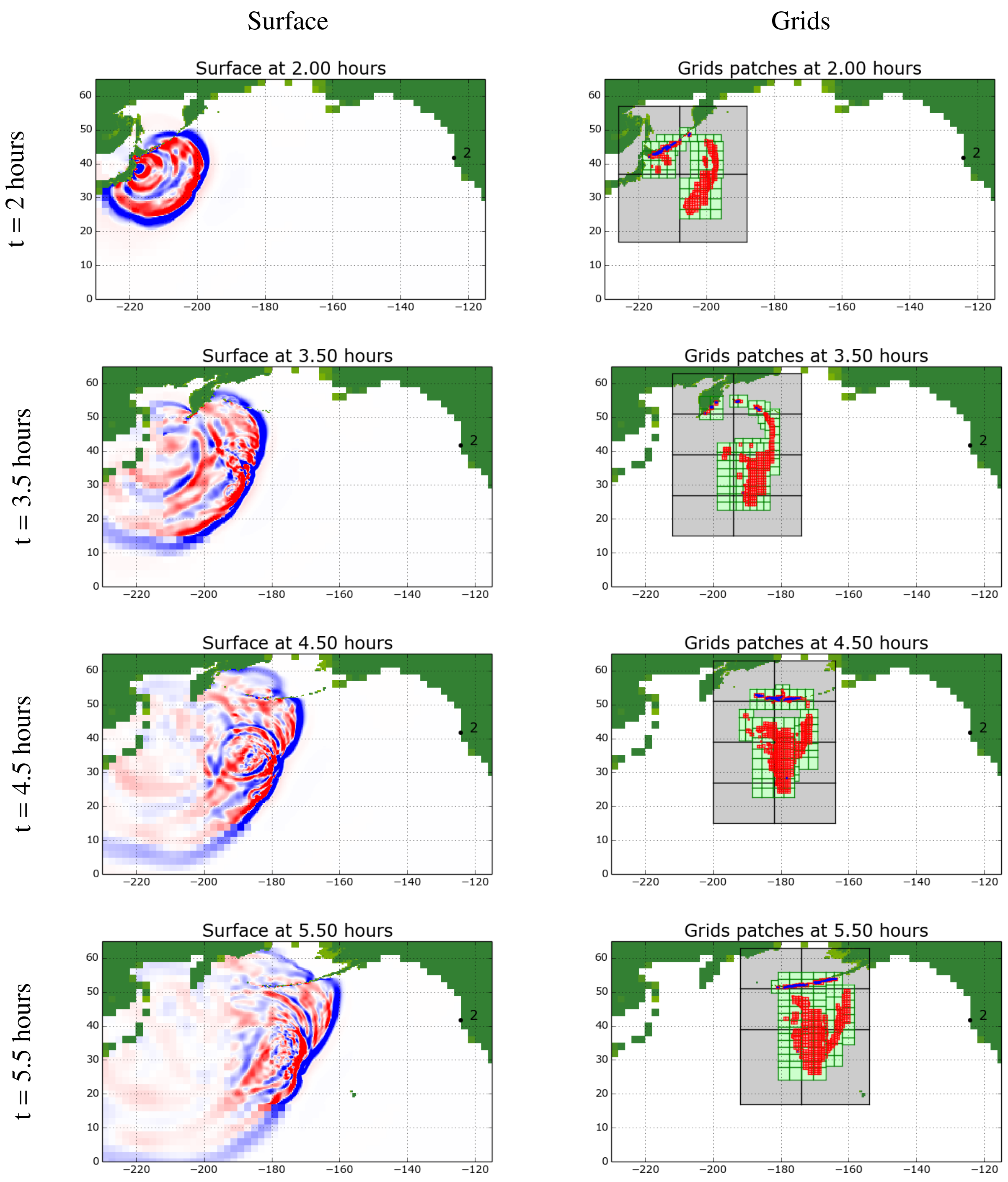


Figure 8.12: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 10.75$ and $t_f = 11.3$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between $-0.1$ and $0.1$. Times are in hours since the earthquake.

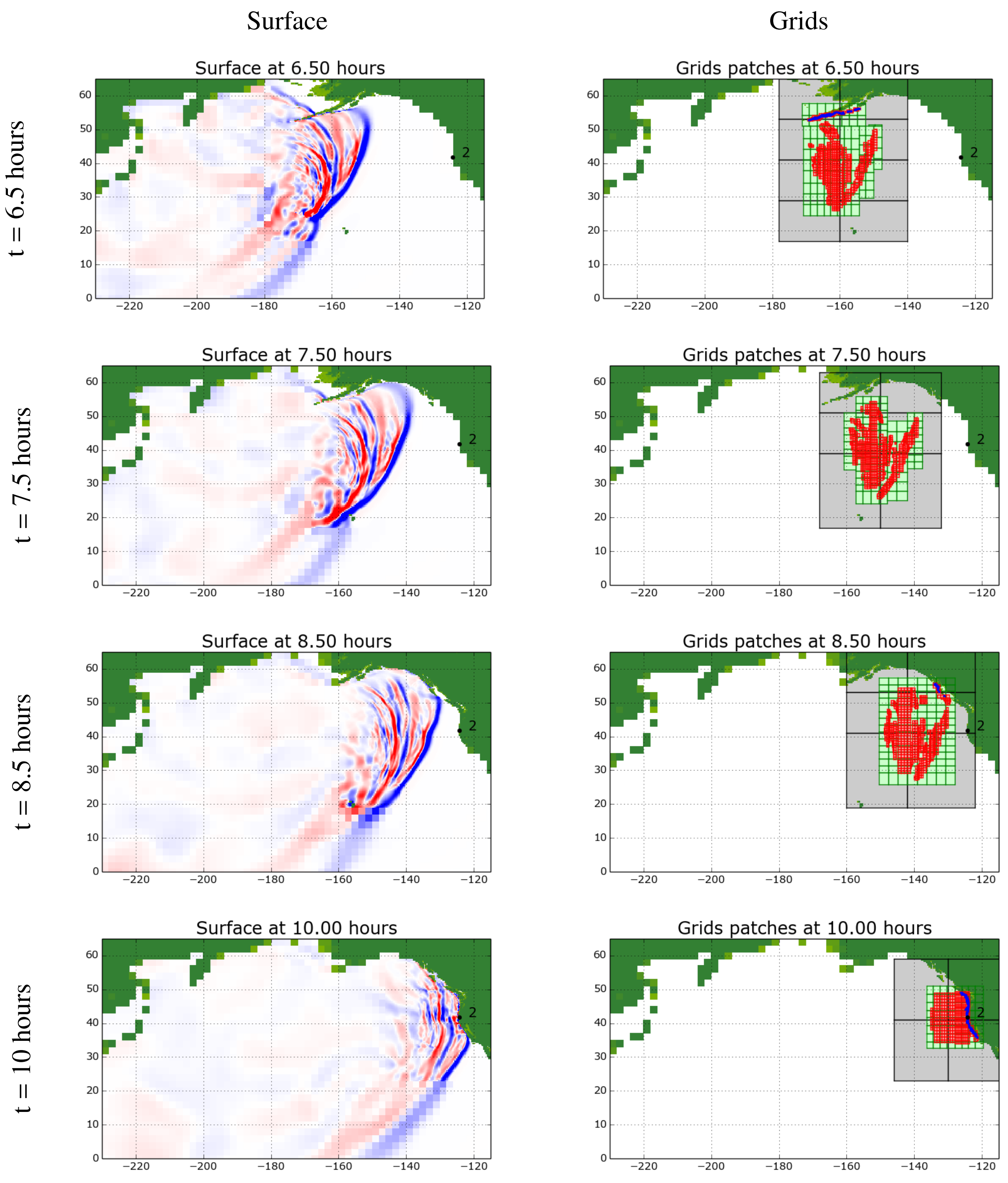


Figure 8.13: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 10.75$ and $t_f = 11.3$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between −0.1 and 0.1. Times are in hours since the earthquake.

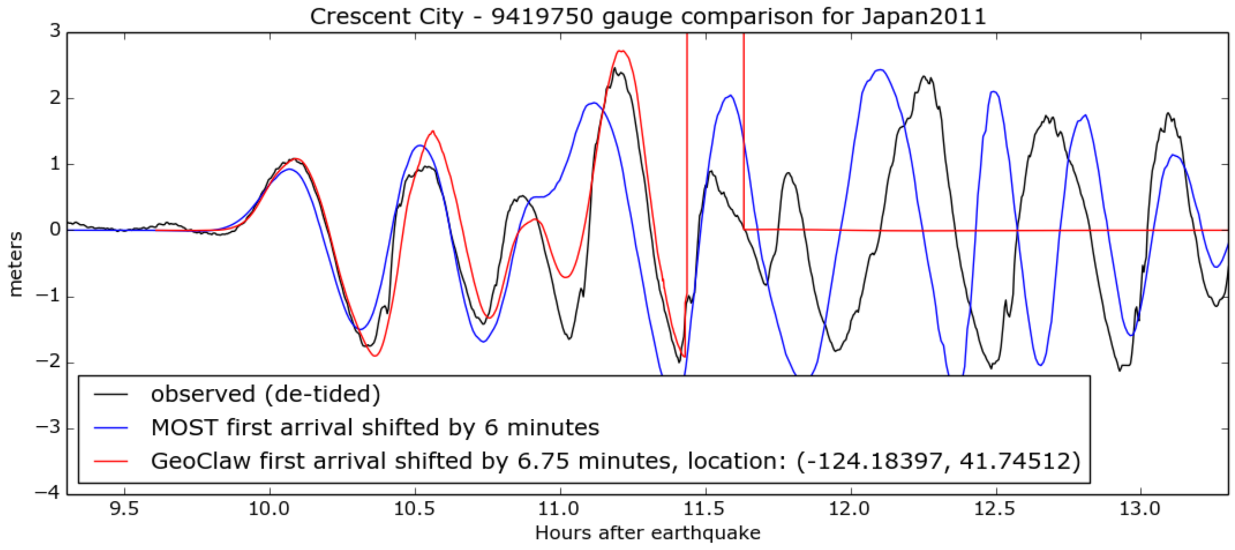


Figure 8.14: Results at the gauge for the Japan 2011 tsunami propagation problem with $t_s = 10.75$ and $t_f = 11.3$ hours. Along the $x$-axis, the time since the occurrence of the earthquake is shown in hours.

quake, the waves that reflect off of Hawaii play a significant role in the large wave that arrives a little after 12 hours post-quake.

Figure 8.17 shows the gauge results from running this problem with $t_s = 11.75$ hours and $t_f = 12.5$ hours. In this figure the observed waves from a tide gauge are shown in black, the waves calculated by MOST are shown in blue, and the waves calculated by GeoClaw are shown in red. Note that the waves that arrive during our region of interest are well captured by GeoClaw. In fact, the whole wave train that arrives before our region of interest is also fairly well captured. This is due to the fact that the early wave train becomes edge waves, which propagate within the Continental shelf and combine with the wave reflecting off to Hawaii to form the large wave that arrives at Crescent City a little after 12 hours post-quake. Note that after our time range of interest the calculated gauge results from GeoClaw are not accurate at all. This is because the simulation is no longer refining the area around Crescent City, as should be expected given that by selecting

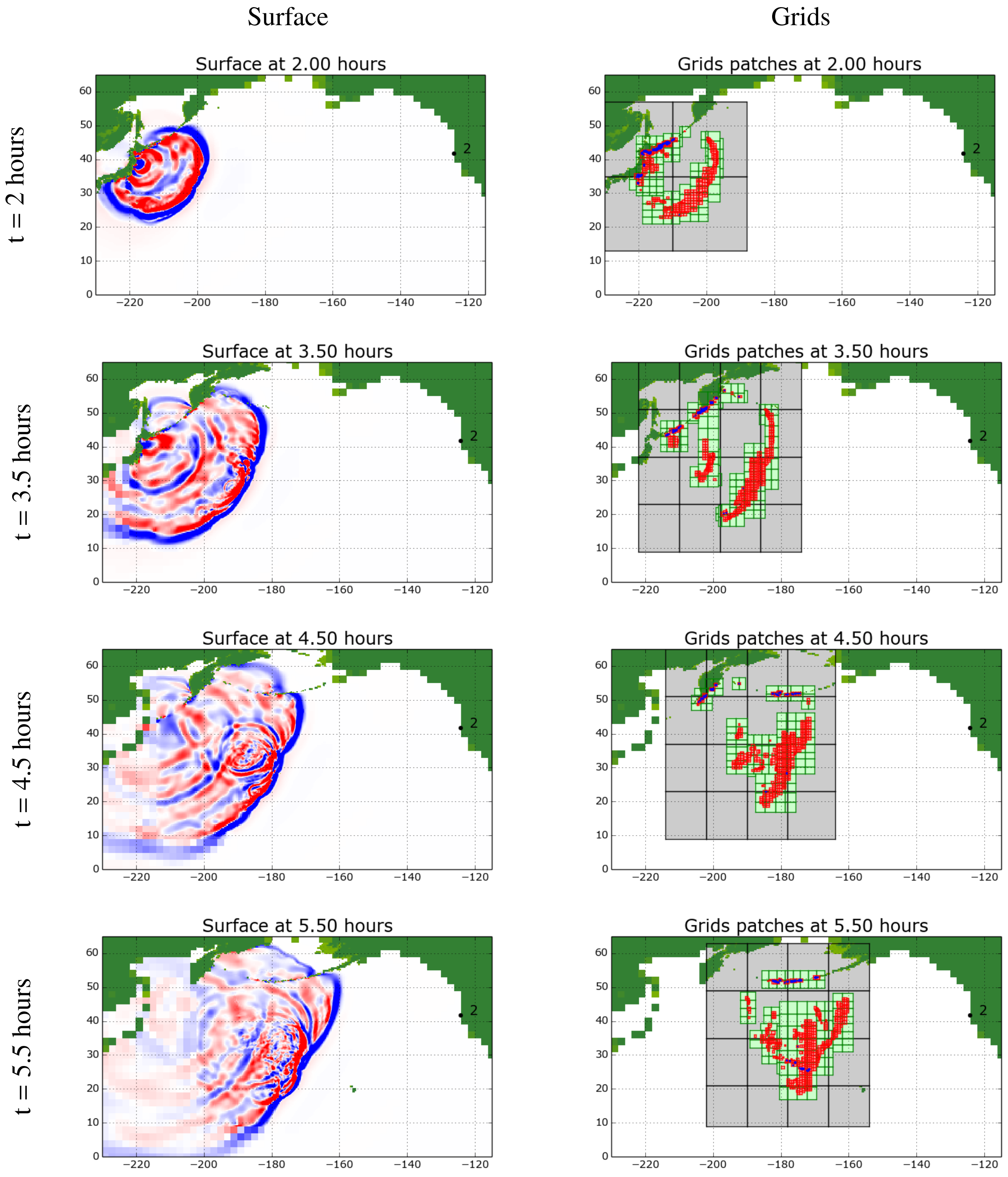


Figure 8.15: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 11.75$ and $t_f = 12.5$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between $-0.1$ and $0.1$. Times are in hours since the earthquake.

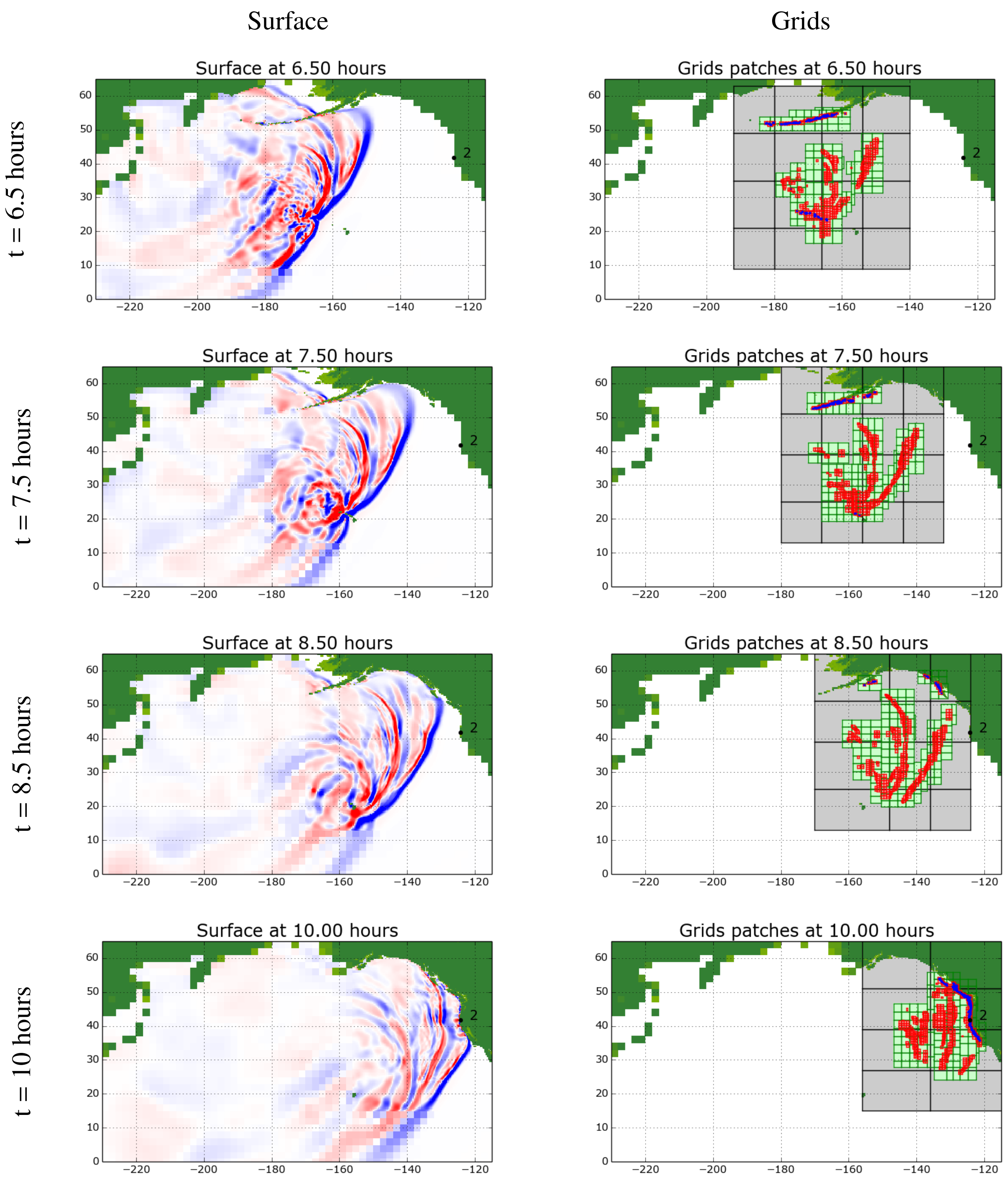


Figure 8.16: Computed results for Japan 2011 tsunami propagation problem when adjoint-flagging is used with $t_s = 11.75$ and $t_f = 12.5$ hours. The $x$-axis and $y$-axis are latitude and longitude, respectively. In the grid figures each color corresponds to a different level of refinement: the coarsest refinement level is shown in white, and the second, third, fourth and fifth levels of refinement are shown in grey, green, red, and blue, respectively. The color scale for the surface height figures goes from blue to red, and ranges between $-0.1$ and $0.1$. Times are in hours since the earthquake.

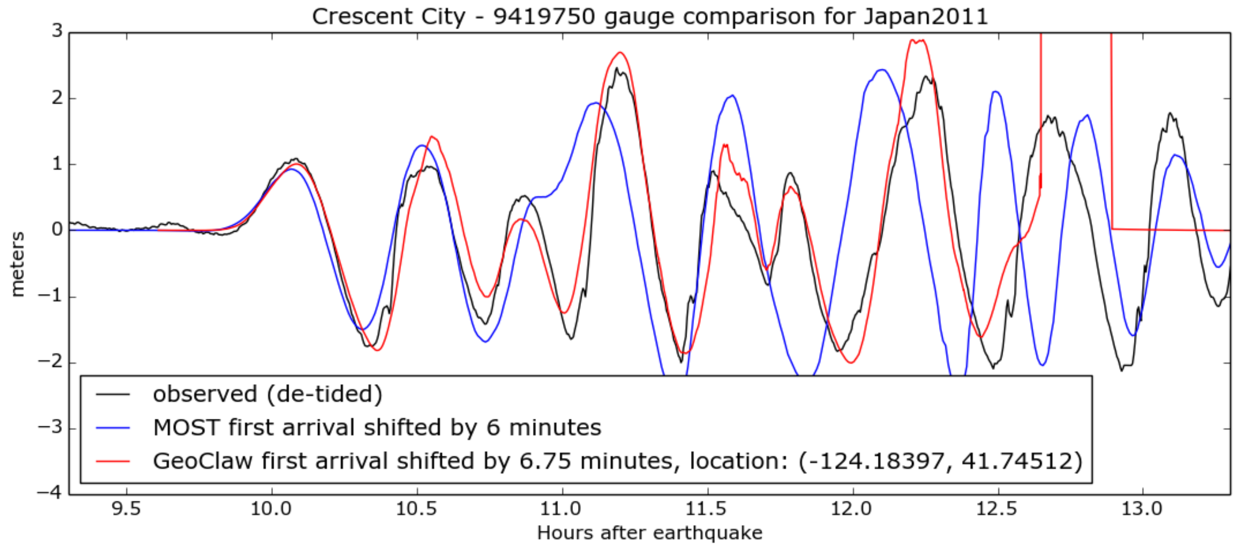


Figure 8.17: Results at the gauge for the Japan 2011 tsunami propagation problem with $t_s = 11.75$ and $t_f = 12.5$ hours. Along the $x$-axis, the time since the occurrence of the earthquake is shown in hours.

$t_f = 12.5$ we have stated that we are not interested in the behavior of the waves after this time.

Given these results, we can see that the time range of interest in the adjoint method can be used to study the parts of a tsunami wave that combine to generate particular waves at our target location. This capability is useful for gaining a greater understanding of the general behavior of waves as they propagate across the ocean, and analyzing interesting phenomena such as was seen at Crescent City from the Japan 2011 tsunami — where later waves were larger than the initially arriving tsunami generated waves.

## *8.4 Alaskan Source Impact on Puget Sound*

As a final example showcasing the capabilities of the adjoint method, we consider the problem of determining what source regions will have the greatest tsunami response at a particular target location. For instance, this kind of analysis has been done to determine what potential source regions around the Pacific Rim give the largest tsunami response at Pearl Harbor [105] and at particular target locations of interest along the U.S. west coast [98]. For these publications the authors calculated propagation time series of wave heights and velocities from many different tsunami sources using the MOST model. This required that a tsunami simulation be run for each of the sources being considered. The use of the adjoint method allows us to do a similar analysis with a single full scale simulation — we solve the adjoint problem with a GeoClaw simulation, and then by computing the appropriate inner products are able to determine the sensitivity of our location of interest to the tsunami sources being considered.

For this example we will focus on the impact that earthquake sources along the Alaskan-Aleutian subduction zone could have on the Puget Sound in Washington, U.S.A. Since we are interested in the impact to the Puget Sound, we will define an adjoint problem that will enable us to focus on this target location. As an aid to understanding the geography of the region a bit, and to give context to the initial conditions for the adjoint problem that are described below, fig. 8.18 shows the location of the adjoint initial condition. The adjoint initial condition is placed in the Strait of Juan de Fuca, which lies between the Olympic Peninsula, U.S.A., and Vancouver Island, Canada.

Recall that we are interested in the accurate calculation of the water surface height ($\eta(x, y, t)$, defined in eq. (6.20)). To focus on the Puget Sound, we define a Gaussian centered about $(x_c, y_c) = (236, 48.294501)$ where $x$ and $y$ are being measured in degrees. Since the waves from any forward problem being considered will be flowing into the Puget Sound, a leftward (out of the Sound) flowing adjoint initial condition was selected (which can be achieved by making sure that the $\hat{q}$ vector is a multiple of the left-going fluctuation given by eq. (6.24)). The idea behind this selection was to place the emphasis on the forward problem waves that are flowing into the Sound, since if simply a Gaussian initial condition for the adjoint was selected it would split into equal left going and right going waves and any right-going waves would be reflected off of the coastlines in the Puget Sound for a while before finding their way out to the open ocean. Therefore, any initially right-going waves would add computational complexity without adding much if anything to the actual question at hand: the effects of tsunami waves coming from outside of Puget Sound. This

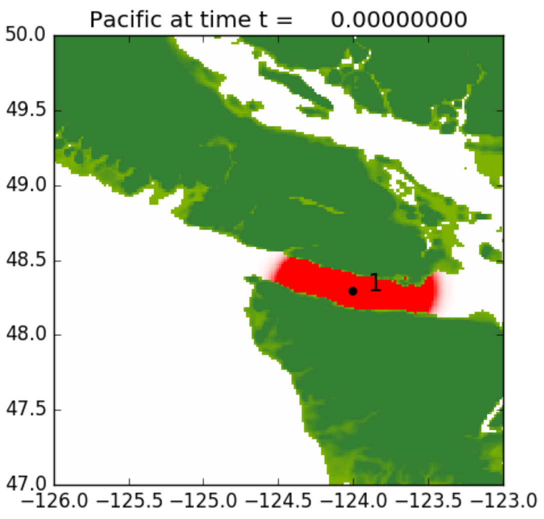


Figure 8.18: Initial conditions for the Puget Sound, Washington, adjoint problem. The initial condition is a Gaussian in the Strait of Juan de Fuca, centered about the location marked with a black dot and the label "1." South of the Strait of Juan de Fuca is the Olympic Peninsula, U.S.A. North of the strait is Vancouver Island, Canada.

does, however, have the effect of adding a momentum term to our functional of interest ($J$). Setting

$$J = \int_{x_{min}}^{x_{max}} \int_{y_{min}(x)}^{y_{max}(x)} \varphi(x, y) q(x, y, t_f) dy\, dx, \tag{8.5}$$

where the limits of integration define the appropriate circle, we set

$$\varphi(x, y) = \begin{bmatrix} \hat{\eta}(x, y) \\ \hat{\eta}(x, y)/c(x, y) \\ 0 \end{bmatrix}, \tag{8.6}$$

where $\hat{\eta} = 0.1 \exp(-(r/\beta)^2)$, $c = \sqrt{gh}$ is the speed of the gravity waves as we saw in section 6.3, $r = \text{haversine}(x, y, x_c, y_c)$ is the Haversine formula giving the distance in meters between the points $(x, y)$, and $(x_c, y_c)$, and $\beta = 15$ meters is the Gaussian width parameter. This function $\varphi(x, y)$ defines the "initial data" $\hat{q}(x, y, t_f)$ for the adjoint problem. The implications of the decision to add a leftward velocity to the adjoint initial conditions, and what effect this had on our final results, is an interesting area of future work.

Figure 8.19 shows the results for the simulation of this adjoint problem. For this problem, a coarse grid is used over the entire Pacific (1 degree resolution) where the ocean is at rest. In addition to AMR being used (with surface-flagging) to following propagating waves across the ocean, higher levels of refinement were enforced around the west coast of the U.S.A, around Hawaii, and around Alaska. A total of 3 levels of refinement are used, starting with 1-degree resolution on the coarsest level, and with refinement ratios of 5 and 3 from one level to the next. In this simulation we used 1 arc-minute bathymetry from the ETOPO1 Global Relief Model of [24] for the entire simulation area. The simulation was run out to 10 hours, to allow time for the wave train that is reflected off of Hawaii to reach Alaska, and we saved snapshots of the solution every 3 minutes.

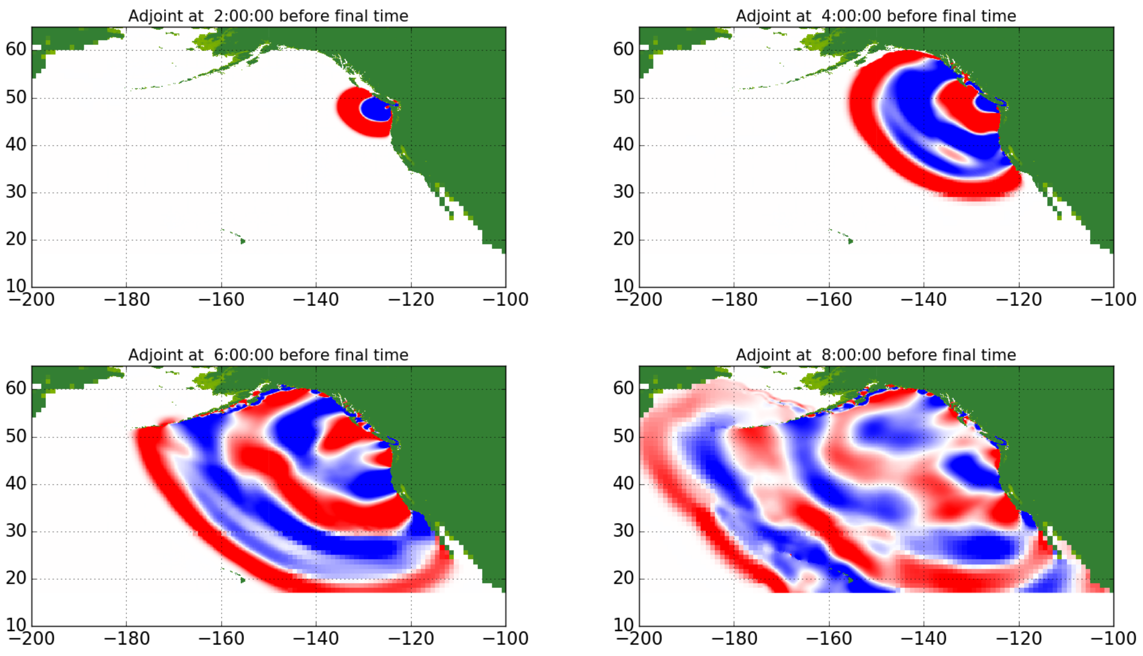

Adjoint at 2:00:00 before final time
Adjoint at 4:00:00 before final time
Adjoint at 6:00:00 before final time
Adjoint at 8:00:00 before final time
60
50
40
30
20
10
−200
−180
−160
−140
−120
−100

Figure 8.19: Computed results for adjoint beginning in the Strait of Juan de Fuca, with a total simulation time of 10 hours. The *x*-axis and *y*-axis are latitude and longitude, respectively. The color scale goes from blue to red, and ranges between −0.0001 and 0.0001. Times are in hours before the final time.

The NOAA center for tsunami research has developed a database of *unit sources* that can be used to generate initial conditions for tsunami propagation problems [45]. For this example we were interested in the effects to the Puget Sound of a slip occurring on any one of the unit sources found along the Alaskan-Aleutian subduction zone, which are shown plotted in fig. 8.20.

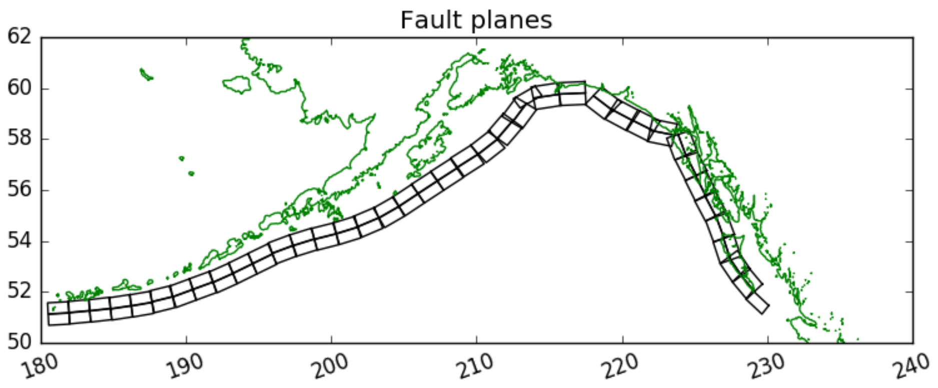


Figure 8.20: NOAA unit sources used to generate the initial conditions for the source impact analysis on the Puget Sound, WA.

Perturbation files were generated by giving each of these seventy-eight unit sources a slip of 10 meters, and the results from the perturbation of each unit source was saved as a separate `dtopo` file. Without the adjoint method, if we wanted to do a source impact analysis on Puget Sound we would need to run full GeoClaw simulations for each of these deformation files. Since we are using the adjoint method, we are able to take advantage of the fact that eq. (4.7) can give us our functional of interest at any time $t$. Therefore, by taking the inner product between the adjoint solution (which we have already computed) and the forward solution at any time, we can get the sensitivity of our location of interest to the source we are considering. To see if the method is robust with respect to time, we ran a short (1 hour of tsunami propagation time) GeoClaw simulation for each of the deformation files. This will allow us to consider the forward solution at various times for each of these sources. Figure 8.21 shows the calculated output from one of these short GeoClaw simulations, for the unit source `acsza21`, as an aid to visualizing what we are working with.

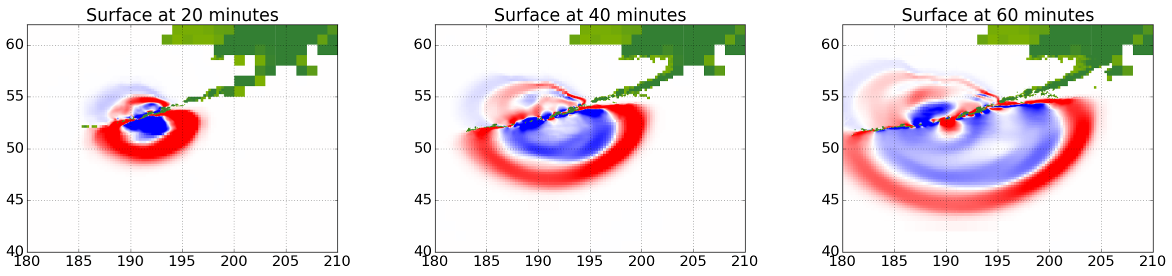


Figure 8.21: Computed results for one of the short GeoClaw runs conducted for the unit sources along the Alaskan fault. The color scale goes from blue to red, and ranges between −0.03 and 0.03. Times are in hours since the initial perturbation of the unit source.

For each of the unit sources we are considering we took the inner product between:

- the initial deformation given by the slip on the unit source and the adjoint solution snapshots,
- the forward solution after 30 minutes of wave propagation and the adjoint solution snapshots, and
- the forward solution after 60 minutes of wave propagation and the adjoint solution snapshots.

Figure 8.22 shows the results from this analysis. Note that figs. 8.22(b) and 8.22(c) show very similar results. However, fig. 8.22(a) varies from the other two figures. In theory, eq. (4.7) tells us that our functional of interest should be constant regardless of the time that we are considering. Therefore, the maximum value of our functional of interest should be constant as well. Once the GeoClaw simulation begins, the value of our functional of interest does seem to be robust with respect to time as can be seen in figs. 8.22(b) and 8.22(c). The functional of interest when we are considering the initial deformation does not, however, give the same results. Computing the functional of interest at time $t = 0$ is interesting because no forward wave propagation has occurred, which means that the directionality of the forward waves is not being taken into account. How this can and should be reconciled with the fact that our functional of interest should be constant in time is not clear, and needs further exploratory work. For now, our results appear to be robust in time for any time after the forward wave propagation begins.

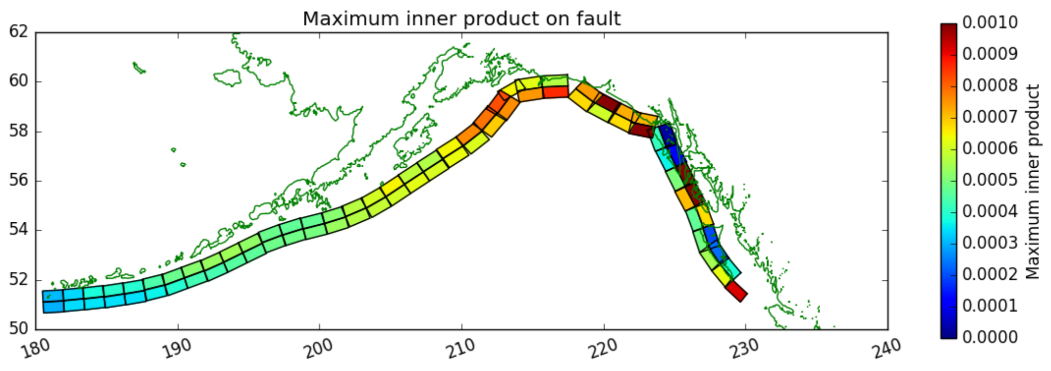
As another way to visualize the results, and to evaluate whether our results are robust with respect to time, fig. 8.23 shows the times corresponding to the maximum inner product between the forward solution corresponding to each unit source and the adjoint problem. The time, shown in hours, is the sum of the total time the forward problem has propagated (either 0, 30, or 60 minutes) and the total time the adjoint problem has propagated (backwards in time since the final time). This total time corresponds to the time $t$ (since the initial perturbation) at which the functional of interest

$$J = \int\int \hat{q}(x, y, 0)q(x, y, t)dy\,dx$$

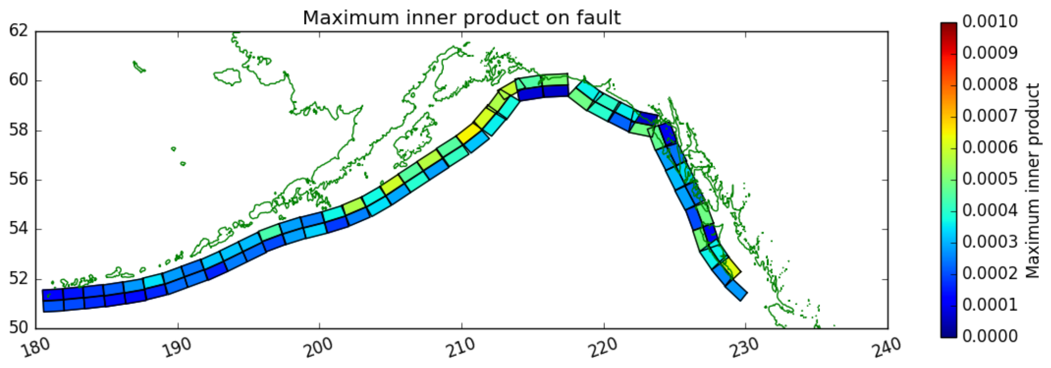
will be maximal. Recall that we have selected the initial conditions for the adjoint problem, $\hat{q}(x, y, 0)$, to be a Gaussian flowing out of the Strait of Juan de Fuca. Therefore, this total time corresponds to when the maximum wave height would be experienced flowing into the Strait for the forward problem generated from each unit source. Again, since our functional of interest is in theory constant in time, these results should be robust to changes in the forward problem time we are considering. However, similar to what we saw in fig. 8.22, note that while figs. 8.23(b) and 8.23(c) show very similar results, fig. 8.23(a) varies fairly significantly. Again, our results appear to be robust in time for any time after the forward wave propagation begins.

In summary, with the adjoint method we have been able to identify the unit sources that will have a maximal impact on our region of interest by

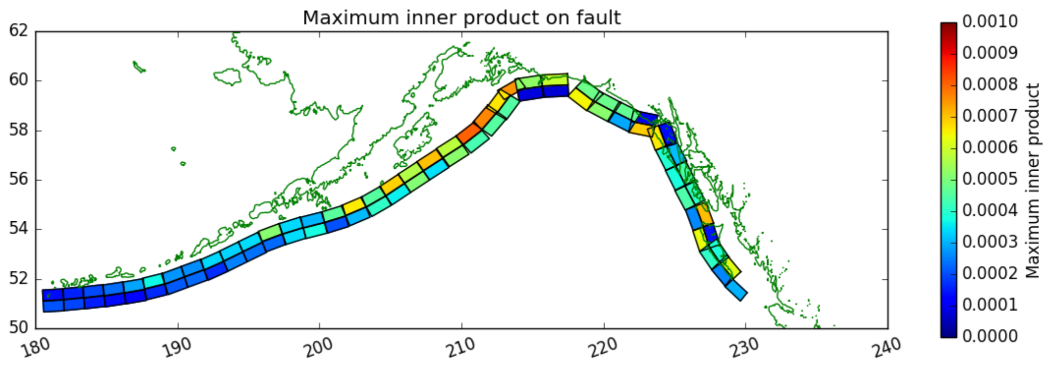
- solving the adjoint problem backward in time,
- calculating the solution for the forward problem for a short time (in this case 1 hour) for each of the unit sources we were considering, and
- calculating the inner product between the adjoint solution snapshots and the forward solution at a short time after the initial deformation.



(a) Maximum inner product between the adjoint solution and the initial deformation

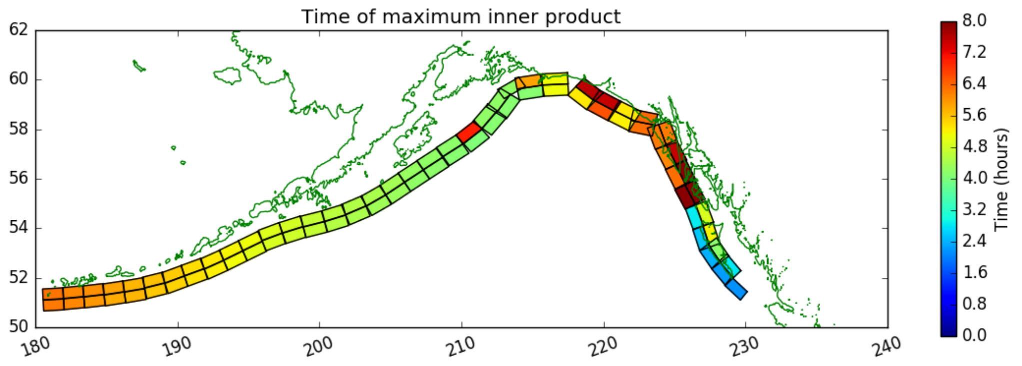


(b) Maximum inner product between the adjoint and forward solution 30 minutes after the initial deformation

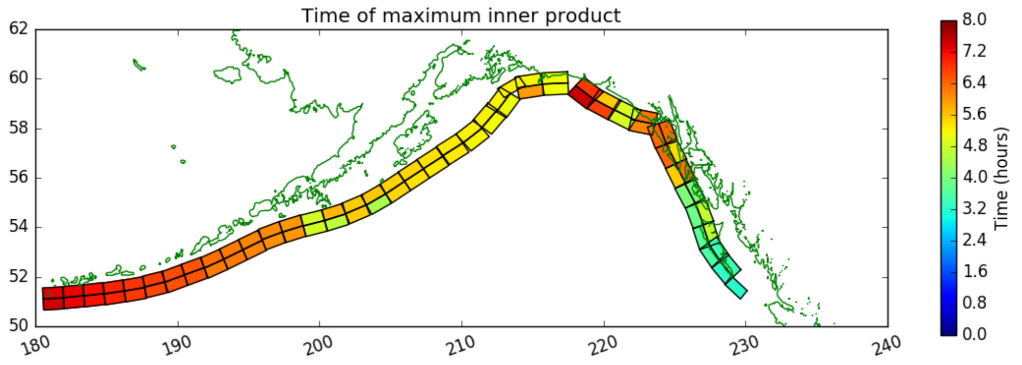


(c) Maximum inner product between the adjoint and forward solution 60 minutes after the initial deformation

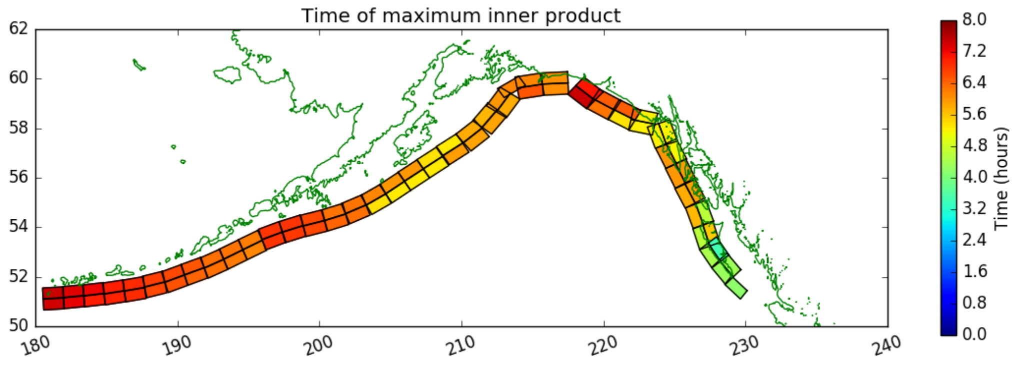
Figure 8.22: Maximum inner product for each unit source between the adjoint solution and the forward solution at various different times. Note that maximum inner product could occur for different adjoint solution snapshots (corresponding to different adjoint times) for each unit source.



(a) Time corresponding to maximum inner product between the adjoint solution and the initial deformation



(b) Time of max inner product between the adjoint and forward solution 30 minutes after initial deformation



(c) Time of max inner product between the adjoint and forward solution 60 minutes after initial deformation

Figure 8.23: Time of maximal inner product for each unit source. The time (in hours) corresponds to the sum of the time that the forward problem has propagated and the time the adjoint problem has propagated (backwards in time). This total time corresponds to the time at which the maximum wave height at Puget Sound would be experienced for the forward problem from each unit source.

We are then able to visualize, as we have in fig. 8.22, the maximum inner product for each unit source. The unit sources that resulted in larger maximum inner products will have a larger impact on our target location (in this case the Strait of Juan de Fuca). This in turn allows us to do source impact analysis or to design a source that will generate a tsunami of maximal impact.

Verifying that the results of this analysis are accurate (by running various forward problems to 10 or 11 hours) and preforming a study on the effects of the adjoint initial conditions on the final result are interesting areas of future work.

# Chapter 9
# SUMMARY, CONCLUSIONS, AND FUTURE WORK

## *9.1 Summary of results*

The focus of this work has been to develop a method that allows us to perform guided adaptive mesh refinement when working on a problem with a particular area of interest, and to implement this method in the context of Clawpack. This thesis has focused on the adjoint method, which has been widely used in a variety of different fields both for error correction and for adaptive mesh refinement. However, the continuous adjoint method which is presented in this work has had limited use in the literature. Furthermore, the application of the continuous adjoint method for guiding adaptive mesh refinement in the context of geosciences is novel to this work. This work extends the ideas of the continuous adjoint method, expands them to be used for guiding AMR, and develops various algorithms that are necessary for implementing this method in the context of Clawpack. A one-dimensional version of AMRClaw was also developed as part of this work, which allowed us to preform more in-depth analysis of the computational performance of the adjoint method being presented.

After providing some motivation in chapter 1, this work moved on to introducing some of the basic building blocks that were used. Chapter 2 introduced the adjoint method by providing some of the background for the method that can be found in the literature, and various examples of how the method has been used in a wide range of fields. The differences between the continuous and discrete adjoint methods were presented, with a short discussion of the drawbacks and benefits of each method. Finally, the mathematical development of the adjoint equation was also presented, both in the context of an algebraic system of equations and in the context of time-dependent linear equations. Chapter 3 presented a brief introduction to the finite volume method used in Clawpack, which set up the framework which this work used to solve the adjoint equation. Finally, chapter 4 introduced the methods currently in use in AMRClaw and GeoClaw for adaptive mesh refinement. This chapter transitions from speaking generally about solving a system of equations to focusing on how to *flag* grid cells for refinement, which is a fundamental part of the adaptive mesh refinement algorithms. Two approaches to using the adjoint method to guide adaptive mesh refinement are presented: *adjoint-difference flagging* and *adjoint-error flagging*. This chapter also transitioned from setting up a backdrop for the development of the adjoint method to discussing how the adjoint method will be used to guide adaptive mesh refinement.

Chapter 5 presented details on the implementation of the adjoint method in the context of AMRClaw and GeoClaw. This involved the development of several new algorithms, so that Clawpack could properly utilize the adjoint solution when flagging cells for refinement while solving the forward problem. Chapter 6 presented several systems of linear equations that were considered in this work, and developed the adjoint equation for each. Since Clawpack solves systems of linear equations using Riemann solvers, the equations used in the Riemann solvers for the adjoint problems being considered in this work are also presented. Finally, chapters 7 and 8 present the simulation results for using the adjoint method to guide adaptive mesh refinement in the context of linear variable coefficient acoustics and tsunami modeling using the shallow water equations. For the linear variable coefficient acoustics examples, the results from using adjoint-difference and adjoint-error flagging are compared to results using the flagging methods previously available in Clawpack. For the tsunami modeling case, various benefits of using the adjoint method to guide adaptive mesh refinement are presented in the context of several tsunami examples.

## 9.2 Conclusions

In this thesis I have developed the use of the continuous adjoint method to guide adaptive mesh refinement when the problem being considered has a particular location of interest, implemented this method in AMRClaw and GeoClaw, and developed a one-dimensional version of AMRClaw. The one-dimensional version of AMRClaw allowed for more extensive comparison between the theory presented in this thesis and the computational results found. The adjoint method was shown to provide significant improvements in

- the computational time required to run a simulation, particularly when the area of the domain that needs to be calculated to a fine resolution is small relative to the whole domain,
- the user time required, particularly when solving a problem that would historically require the user to define regions where refinement was either required or forbidden, and
- computer memory requirements, since smaller portions of the ocean are being refined and therefore less cell updates are being calculated.

The adjoint method also provided several new capabilities like

- allowing the user-set tolerance in AMRClaw to be related to the final desired error in the solution through the use of adjoint-error flagging,
- focusing in on certain time ranges of interest, which allows for identifying portions of a wave train that are relevant for particular waves of interest at the target location,
- preforming sensitivity analysis, and
- preforming analysis on source design and sources of potential maximal amplitude waves.

## *9.3 Future work*

In this work we have focused on examples involving the linear acoustics equations and the linearized shallow water equations. However, the adjoint method is not limited to working on only these equations. Using the adjoint method when solving non-linear equations requires that the equations be linearized. In the context of using the shallow water equations for tsunami propagation across the ocean this is fairly straightforward, since they can be linearized about the ocean at rest. An area of future work is studying the use of the adjoint method in the context of other nonlinear hyperbolic equations, where the adjoint equation would be derived by linearizing about a particular forward solution. This would require the development of some kind of automated process to shift between solving the forward problem, linearizing about that forward problem, solving the corresponding adjoint problem, and using that adjoint solution in guiding the adaptive mesh refinement for the forward problem.

We have focused in this work on the accurate calculation of the functional $J$, where the definition for this functional varied based on the target area of interest. However, we are typically concerned with the accurate estimation of the solution to the forward problem rather than the functional $J$. For the examples presented in this work we used various different initial conditions for the adjoint problem, but did not consider the effect this had on the accuracy of the forward solution. Examining the implications of the initial conditions used for the adjoint problem, and their effect on the accuracy of the foward solution, is another area for future work.

# BIBLIOGRAPHY

[1] L. M. Adams and R. J. LeVeque. GeoClaw Model Tsunamis Compared to Tide Gauge Results – Final Report. http://hdl.handle.net/1773/41886, 2017.

[2] C. Airiau, A. Bottaro, S. Walther, and D. Legendre. A methodology for optimal laminar flow control: Application to the damping of Tollmien-Schlichting waves in a boundary layer. *Physics of Fluids*, 15(5):1131–1145, 2003.

[3] V. Akcelik, G. Biros, and O. Ghattas. Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, SC '02, pages 1–15, 2002.

[4] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4):443 – 480, 1999.

[5] F. Angrand. Optimum design for potential flows. *International Journal for Numerical Methods in Fluids*, 3(3):265–282, 1983.

[6] L. Asner, S. Tavener, and D. Kay. Adjoint-based a posteriori error estimation for coupled time-dependent systems. *SIAM Journal of Scientific Computing*, 34:A2394A2419, 2012.

[7] J. P. Aubin. Behavior of the error of the approximate solutions of boundary value problems for linear elliptic operators by Galerkin's and finite difference methods. *Annali della Scuola Normale Superiore di Pisa - Classe di Scienze*, 21(4):599–637, 1967.

[8] Australian National University (ANU) and Geoscience Australia (GA. Anuga software, Accessed 2018.

[9] A. Bagchi and P. T. Brummelhuis. Parameter identification in tidal models with uncertain boundaries. *Automatica*, 30(5):745–759, 1994.

[10] D. Bale, R. J. LeVeque, S. Mitran, and J. A. Rossmanith. A wave-propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal of Scientific Computing*, 24:955–978, 2002.

[11] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[12] M. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1278–1286, Sep 1991.

[13] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[14] M. J. Berger, D. L. George, R. J. LeVeque, and K. T. Mandli. The GeoClaw software for depth-averaged flows with adaptive refinement. *Advances in Water Resources*, 34:1195–1206, 2011.

[15] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal of Numerical Analysis*, 35(6):2298–2316, 1998.

[16] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[17] F. Beux and A. Dervieux. Exact-gradient shape optimization of a 2-d Euler flow. *Finite Elements in Analysis and Design*, 12(3):281 – 302, 1992.

[18] T. R. Bewley. Flow control: New challenges for a new renaissance. In *Progress in Aerospace Sciences 37*, pages 21–58. American Elsevier Publishing Company, Inc, 2001.

[19] S. Blaise, A. St-Cyr, D. Mavriplis, and B. Lockwood. Discontinuous Galerkin unsteady discrete adjoint method for real-time efficient tsunami simulations. *Journal of Computational Physics*, 232, 2013.

[20] J. C. Borrero and S. D. Greer. Comparison of the 2010 Chile and 2011 Japan tsunamis in the far field. *Pure and Applied Geophysics*, 170:1249–1274, 2013.

[21] J. C. Borrero, R. J. LeVeque, D. Greer, S. O'Neill, and B. N. Davis. Observations and modelling of tsunami currents at the Port of Tauranga, New Zealand. In *Australasian Coasts & Ports Conference 2015*, pages 90–95. Engineers Australia and IPENZ, Auckland, New Zealand, 2015.

[22] G. Buffoni and E. Cupini. The adjoint advection-diffusion equation in stationary and time dependent problems: a reciprocity relation. *Rivista di Matematica della Universita di Parma*, 4:9–19, 2001.

[23] H.-P. Bunge, C. R. Hagelberg, and B. J. Travis. Mantle circulation models with variational data assimilation: inferring past mantle flow and structure from plate motion histories and seismic tomography. *Geophysical Journal International*, 152:280–301, 2003.

[24] C. Amante, and B. W. Eakins. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24, National Geophysical Data Center, NOAA, 2009.

[25] V. Carey, D. Estep, A. Johansson, M. Larson, and S. Tavener. Blockwise adaptivity for time dependent problems based on coarse scale adjoint solutions. *SIAM Journal of Scientific Computing*, 32:2121–2145, 2010.

[26] G. R. Carmichael, D. N. Daescu, A. Sandu, and T. Chai. Computational aspects of chemical data assimilation into atmospheric models. In *Computational Science — ICCS 2003*, pages 269–278, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[27] X. Chen and I. Navon. Optimal control of a finite-element limited-area shallow-water equations model. *Studies in Informatics and Control*, 18:41–62, 03 2009.

[28] M. Chini, A. Piscini, F. R. Cinti, S. Amici, R. Nappi, and P. M. DeMartini. The 2011 Tohoku (Japan) tsunami inundation and liquefaction investigated through optical, thermal, and SAR data. *IEEE Geoscience and Remote Sensing Letters*, 10(2):347–351, March 2013.

[29] M. Chini, L. Pulvirenti, and N. Pierdicca. Analysis and interpretation of the COSMO-SkyMed observations of the 2011 Japan tsunami. *IEEE Geoscience and Remote Sensing Letters*, 9(3):467–471, May 2012.

[30] Clawpack Development Team. Clawpack software, Accessed 2018. Version 5.3.

[31] COMCOT Development Team. Comcot: A tsunami modeling package, Accessed 2018.

[32] B. N. Davis. Adjoint code repository, Accessed 2018. Available at `https://github.com/BrisaDavis/adjoint`.

[33] B. N. Davis. Riemann code repository, Accessed 2018. Available at `https://github.com/BrisaDavis/riemann/tree/adjoint`.

[34] B. N. Davis and R. J. LeVeque. Author's title: Analysis and performance evaluation of adjoint-guided adaptive mesh refinement for linear hyperbolic PDE's using Clawpack. In Preparation, 2018.

[35] B. N. Davis and R. J. LeVeque. Adjoint methods for guiding adaptive mesh refinement in tsunami modeling. *Pure and Applied Geophysics*, 173:4055–4074, 2016.

[36] L. Dengler and B. Uslu. Effects of harbor modification on Crescent City, California's tsunami vulnerability. *Pure and Applied Geophysics*, 168:1175–1185, 2011.

[37] Y. Ding and S. S. Y. Wang. Identification of Manning's roughness coefficients in channel network using adjoint analysis. *International Journal of Computational Fluid Dynamics*, 19(1):3–13, 2005.

[38] Y. Ding and S. S. Y. Wang. Optimal control of open-channel flow using adjoint sensitivity analysis. *Journal of Hydraulic Engineering*, 132(11):1215–1228, 2006.

[39] J. Elliott and J. Peraire. *Aerodynamic optimization of unstructured meshes with viscous effects*, pages 542–559. American Institute of Aeronautics and Astronautics, 2018/04/16 1997.

[40] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4:105–158, 1995.

[41] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems I: A linear model problem. *SIAM Journal on Numerical Analysis*, 28(1):43–77, 1991.

[42] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems IV: Nonlinear problems. *SIAM Journal on Numerical Analysis*, 32(6):1729–1749, 1995.

[43] K. J. Fidkowski and D. L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49(4):673–694, 2011.

[44] B. T. Flynt and D. J. Mavriplis. Discrete adjoint based adaptive error control in unsteady flow problems. *AIAA Paper 2012-0078*, 2012.

[45] NOAA Center for Tsunami Research Events. Recent and historical tsunami events and relevant data.

[46] P. D. Frank and G. R. Shubin. A comparison of optimization-based approaches for a model computational aerodynamics design problem. *Journal of Computational Physics*, 98(1):74–89, 1992.

[47] S. W. Funke, P. E. Farrell, and M. D. Piggott. Reconstructing wave profiles from inundation data. *Computer Methods in Applied Mechanics and Engineering*, 322:167–186, 2017.

[48] I. Y. Gejadze and G. J. M. Copeland. Adjoint sensitivity analysis for fluid flow with free surface. *International Journal for Numerical Methods in Fluids*, 47:1027–1034, 2005.

[49] GeoClaw Development Team. Geoclaw software, Accessed 2018.

[50] D. L. George. Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation. *Journal of Computational Physics*, 227:3089–3113, 2008.

[51] M. B. Giles, M. C. Duta, J.-D. Müller, and N. A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA Journal*, 41(2):198–205, 2003.

[52] M. B. Giles and E. Suli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, pages 145–236, 2002.

[53] F. I. González, R. J. LeVeque, L. M. Adams, C. Goldfinger, G. R. Priest, and K. Wang. Probabilistic Tsunami Hazard Assessment (PTHA) for Crescent City, CA, September 2014.

[54] P. R. Grothe, L. A. Taylor, B. W. Eakins, K. S. Carignan, R. J. Caldwell, E. Lim, and D. Z. Friday. Digital Elevation Models of Crescent City, California: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-51, U.S. Dept. of Commerce, Boulder, CO, 2011.

[55] M. C. G. Hall. Application of adjoint sensitivity theory to an atmospheric general circulation model. *Journal of Atmospheric Sciences*, 43:2644–2652, 1986.

[56] M. Heidarzadeh and K. Satake. Waveform and spectral analyses of the 2011 japan tsunami records on tide gauge and DART stations across the pacific ocean. *Pure and Applied Geophysics*, 170:1275–1293, 2013.

[57] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[58] A. Jameson, L. Martinelli, and N.A. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1):213–237, Jan 1998.

[59] A. Jameson, N. Pierce, and L. Martinelli. Optimum aerodynamic design using the Navier-Stokes equations. In *35th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 1997.

[60] H. Jing, H. Zhang, Z. Wu, D. A. Yuen, and Y. Shi. Numerical simulation of march 11, 2011 honshu, japan tsunami. *Chinese Science Bulletin*, 57(27):3617–3622, Sep 2012.

[61] S. Kabanikhin, A. Hasanov, I. Marinin, O. Krivorotko, and D. Khidasheli. A variational approach to reconstruction of an initial tsunami source perturbation. *Applied Numerical Mathematics*, 83:22–37, 2014.

[62] S. M. Kast and K. J. Fidkowski. Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains. *Journal of Computational Physics*, pages 468–494, 2013.

[63] G. J. Kennedy and J. R. R. A. Martins. An adjoint-based derivative evaluation method for time-dependent aeroelastic optimization of flexible aircraft. In *Proceedings of the 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Boston, MA, April 2013. AIAA-2013-1530.

[64] M. Kouhi, E. Oñate, and D. Mavriplis. Adjoint-based adaptive finite element method for the compressible euler equations using finite calculus. *Aerospace Science and Technology*, 46:422 – 435, 2015.

[65] A. Lacasta, M. Morales-Hernández, P. Brufau, and P. García-Navarro. Application of an adjoint-based optimization procedure for the optimal control of internal boundary conditions in the shallow water equations. *Journal of Hydraulic Research*, 56(1):111–123, 2018.

[66] J. O. Langseth and R. J. LeVeque. A wave-propagation method for three-dimensional hyperbolic conservation laws. *Journal of Computational Physics*, 165:126–166, 2000.

[67] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Basel, 1992.

[68] R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics*, 131(CP965603):327–353, 1997.

[69] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2004.

[70] R. J. LeVeque. A well-balanced path-integral f-wave method for hyperbolic problems with source terms. *Journal of Scientific Computing*, 48(1-3):209–226, 07 2011.

[71] R. J. LeVeque, D. L. George, and M. J. Berger. Tsunami modeling with adaptively refined finite volume methods. *Acta Numerica*, pages 211–289, 2011.

[72] L. Y. Li, Y. Allaneau, and A. Jameson. Continuous adjoint approach for adaptive mesh refinement. *20th AIAA Computational Fluid Dynamics Conference*, 2011.

[73] S. Lia and L. Petzold. Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement. *Journal of Computational Physics*, 198:310–325, 2004.

[74] J. L. Lions. *Optimal control of systems governed by partial differential equations.* Berlin: Springer-Verlag., 1971. Translated by S. K. Mitter.

[75] R. B. Long and W. C. Thacker. Data assimilation into a numerical equatorial ocean model. I. the model and the assimilation algorithm. *Dynamics of Atmospheres and Oceans*, 13(3):379–412, 1989.

[76] Y. Luo and K. J. Fidkowski. Output-based space-time mesh adaptation for unsteady aerodynamics. *AIAA Paper 2011-491*, 2011.

[77] K. T. Mandli, A. J. Ahmadia, M. Berger, D. Calhoun, D. L. George, Y. Hadjimichael, D. I. Ketcheson, G. I. Lemoine, and R. J. LeVeque. Clawpack: building an open source ecosystem for solving hyperbolic pdes. *PeerJ Computer Science*, 2:e68, 2016.

[78] K. T. Mandli and C. N. Dawson. Adaptive mesh refinement for storm surge. *Ocean Modelling*, 75:36–50, March 2014.

[79] K. Mani and D. J. Mavriplis. Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems. *AIAA Paper 2007-3944*, 2007.

[80] J. Marburger. Adjoint-based optimal control of time-dependent free boundary problems, 2012.

[81] A. Mishra, K. Mani, D. Mavriplis, and J. Sitaraman. Time-dependent adjoint-based optimization for coupled aeroelastic problems. In *31st AIAA Applied Aerodynamic Conference*, San Diego, CA, 2013. AIAA 2013-2906.

[82] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2000.

[83] S. K. Nadarajah. *The discrete adjoint approach to aerodynamic shape optimization*. PhD thesis, Stanford University, 2003.

[84] M. Nemec and M. J. Aftosmis. Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes. In *18th AIAA Computational Fluid Dynamics Conference*, 2007.

[85] M. Nemec, M. J. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. In *46th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2008.

[86] J. C. Newman III, A. C. Taylor III, R. W. Barnwell, P. A. Newman, and G. J. W. Hou. Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *Journal of Aircraft*, 36(1):87–96, 1999.

[87] E. J. Nielsen and W. K. Anderson. Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. *AIAA Journal*, 37(11):1411–1419, 2018/04/16 1999.

[88] E. J. Nielsen, B. Diskin, and N. K. Yamaleev. Discrete adjoint-based design optimization of unsteady turbulent flows on dynamic unstructured grids. *AIAA Journal*, 48(6):1195–1206, 2010.

[89] C. Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4(1), 2014.

[90] M. A. Park. Adjoint-based, three-dimensional error prediction and grid adaptation. *AIAA Journal*, 42:1854–1862, 2004.

[91] M. A. Park. *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows*. PhD thesis, Massachusetts Inst. of Technology, 2008.

[92] N. A. Pierce and M. B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42:247–264, 2000.

[93] C. Pires and P. M. A. Miranda. Tsunami waveform inversion by adjoint methods. *Journal of Geophysical Research: Oceans*, 106(C9):19773–19796, 2001.

[94] O. Pironneau. On optimum profiles in Stokes flow. *Journal of Fluid Mechanics*, 59(1):117–128, 1973.

[95] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110, 1974.

[96] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer-Verlag Berlin Heidelberg, 1982.

[97] W. S. Pranowo. *Adaptive Mesh Refinement Applied to Tsunami Modeling: TsunaFLASH*. PhD thesis, Universität Bremen, 2010.

[98] L. Rasmussen, P. D. Bromirski, A. J. Miller, D. Arcas, R. E. Flick, and M. C. Hendershott. Source location impact on relative tsunami strength along the u.s. west coast. *Journal of Geophysical Research: Oceans*, 120(7):4945–4961, 7 2015.

[99] F. Rauser, J. Riehme, K. Leppkes, P. Korn, and U. Naumann. On the use of discrete adjoints in goal error estimation for shallow water equations. *Procedia Computer Science*, 1(1):107–115, 2010. ICCS 2010.

[100] Z. REN, B. WANG, T. FAN, and H. LIU. Numerical analysis of impacts of 2011 japan tohoku tsunami on china coast. *Journal of Hydrodynamics, Ser. B*, 25(4):580–590, 2013.

[101] M. Rumpfkeil and D. Zingg. The optimal control of unsteady flows with a discrete adjoint method. *Optimization and Engineering*, 11:5–22, 03 2010.

[102] Y. Samizo. Optimal control of shallow water flows using adjoint equation method. *Advanced Materials Research*, 403-408:466–469, 11 2011.

[103] B. F. Sanders and N. D. Katopodes. Optimal control of sudden water release from a reservoir. In *Managing Water: Coping with Scarcity and Abundance*, volume A, pages 314–319. ASCE, 1997.

[104] B. F. Sanders and N. D. Katopodes. Adjoint sensitivity analysis for shallow-water wave control. *Journal of Engineering Mechanics*, 126(9):909–919, 2000.

[105] L. Tang, C. Chamberlin, E. Tolkova, M. Spillane, V. V. Titov, E. N. Bernard, and H. O. Mofjeld. Assessment of potential tsunami impact for Pearl Harbor, Hawaii. *NOAA Technical Memorandum OAR PMEL-131*, 2006.

[106] V. V. Titov and F. I. González. Implementation and testing of the method of splitting tsunami (MOST) model. Technical report, NOAA Technical Memorandum ERL PMEL-112, 11 pp UNIDATA, 1997.

[107] V. V. Titov, F. I. González, E. N. Bernard, M. C. Eble, H. O. Mofjeld, J. C. Newman, and A. J. Venturato. Real-time tsunami forecasting: Challenges and solutions. *Natural Hazards*, 35:35–41, 2005.

[108] V. V. Titov and C. E. Synolakis. Numerical modeling of tidal wave runup. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 124(4):157–171, 1998.

[109] J. Tromp, C. Tape, and Q. Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophysical Journal International*, 160:195–216, 2005.

[110] D. A. Venditti and D. L. Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164:204–227, 2000.

[111] D. A. Venditti and D. L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 2002.

[112] D. A. Venditti and D. L. Darmofal. Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows. *Journal of Computational Physics*, 2003.

[113] R. I. Wilson, A. R. Admire, J. C. Borrero, L. A. Dengler, M. R. Legg, P. Lynett, T. P. McCrink, K. M. Miller, A. Ritchie, K. Sterling, and P. M. Whitmore. Observations and impacts from the 2010 Chilean and 2011 Japanese tsunamis in California (USA). *Pure and Applied Geophysics*, 170:1127–1147, 2013.

# Appendix A
# CODE LOCATION FOR THE EXAMPLES PRESENTED

The code for all the examples presented in this work is available online at [32]. This code can be easily modified to solve other systems of hyperbolic equations. This repository also contains other examples illustrating how adjoint-flagging can be used with AMRClaw and GeoClaw.