

GEOCLAW User's Guide

David L George
Department of Applied Mathematics
University of Washington

March 18, 2008

1 Introduction

GEOCLAW is a subset of the CLAWPACK software [4], a set of fortran routines for solving hyperbolic systems of PDE. This document contains information specific to GEOCLAW. More general documentation for CLAWPACK is available at the CLAWPACK website, and should be consulted if you are new to CLAWPACK. GEOCLAW is available for download at www.geoclaw.org, and is included with the more general CLAWPACK software after version 4.3, available at www.clawpack.org. GEOCLAW uses many of the standard routines from AMRCLAW—the adaptive mesh refinement libraries included with CLAWPACK—as well as many specialized routines for solving geophysical flow problems, such as the nonlinear shallow water equations over varying topography with dry regions. Future development of GEOCLAW will include extensions to other geophysical applications involving hyperbolic problems or wave propagation.

2 History

GEOCLAW supersedes the TSUNAMICLAW package and provides the original plus additional functionality. Additionally, we believe that the user control is simpler, more flexible and more powerful. Where TSUNAMICLAW was a stand-alone package modified from CLAWPACK, GEOCLAW is an integrated subset of CLAWPACK and will be included in the standard CLAWPACK distributions after version 4.3.

3 General Features of GEOCLAW

GEOCLAW allows one to model various flooding problems with the shallow water equations, particularly global-scale tsunamis and inundation on a latitude-longitude grid (or smaller local flooding on a Cartesian grid) with a diverse range of spatial and temporal scales. This is accomplished by using a single coarse *level 1* grid for the entire domain, and evolving rectangular Cartesian sub-grids of higher refinement, *level 2*, \dots , *level n*, that track moving waves and inundation at the shoreline. Up to six levels may be used (though typically four or less is recommended). At any given time in the calculation, a particular level of refinement may have numerous disjoint grids associated with it. User specified integers determine the refinement ratios between particular levels, which can lead to a large degree of refinement even for a small number of levels.

The following example illustrates a typical usage of grid-level refinement for calculating the Indian Ocean Tsunami. Consider four levels with refinement ratios of 10. The entire Bay of Bengal can be covered with a *level 1* grid with 50×50 grid cells, each $100\text{km} \times 100\text{km}$ ($1^\circ \approx 100\text{km}$). A second level might be dedicated to tracking the deep ocean tsunami with grid cells that are $(10\text{km})^2$ (this resolution is sufficient given the long

wavelength of deep ocean tsunamis). Third-level grids near the shore could resolve the compressing tsunami onto $(1\text{km})^2$ grid cells, resolution sufficient to identify vulnerable coastlines. Fourth-level grids would then allow resolution onto $(100\text{m})^2$ grid cells. Alternatively, if a specific region was of interest, one could easily resolve a specific community or shoreline onto $(10\text{m})^2$ grid cells using four levels and the right refinement ratios, allowing detailed inundation studies. Calculations of this sort can easily be done on personal desktop or laptop computers using GEOCLAW. Computing the same domain at this single highest resolution would require 10^{10} grid cells and prohibitively small time-steps, and would not be feasible given current computing technology.

GEOCLAW, as part of CLAWPACK, is based on a finite volume numerical method, which means that the solution is represented as a piecewise constant, with numerical values approximating the average solution value in each discrete computational grid cell. There is no specific reference to the shoreline—grid cells may simply be wet or dry depending on their location, and may fill-up or drain-out of water as waves inundate or draw-down at the shoreline. This means that dry land is part of the computational domain, and a single grid is a simple rectangle that may overlap the shoreline. GEOCLAW solves the shallow water equations in their physically relevant conservative form, therefore, the solution is represented as water depth and momentum. This is one feature that allows convergence to discontinuous bores. See [2, 3, 5, 1] for a mathematical descriptions of these methods.

4 Obtaining and Preparing to Run GEOCLAW

CLAWPACK is written in fortran 77, and can be run on any system with a proper fortran compiler, however, this documentation is written with the assumption that you are using a Unix type of operating system. GEOCLAW consists of a directory of specialized library routines that are used along with the CLAWPACK superset, as well as an example application directory that will be used to specify problem specific parameters. Routines in the library should typically not require any modification, and should only be modified at your own risk. (One possible exception are the `.i` files, for increasing memory allocation if needed). Users will typically run their own applications by copying the application directory and modifying parameters in the input files in that directory, as explained further in Section 4.

A stripped-down version of CLAWPACK, containing only the routines necessary for running GEOCLAW, is available at www.geoclaw.org. The specialized GEOCLAW routines, including library routines and some example application directories are in `$CLAW/geoclaw`, where `$CLAW` refers to the top level CLAWPACK directory on your system. That is, on your system, the unzipped directory should be in a path specified by an environment variable that you should name `CLAW`.

Before using GEOCLAW, the source code in the libraries must be made into object files by running `make` in the library directories `$CLAW/amrclaw/2d/lib` and `$CLAW/geoclaw/2dxy/lib`. The makefiles are written under the assumption that you have set an environment variable `FC` to the compiler of your choice, eg. `gfortan`, `f77`, etc. It is recommended to set these environment variables in one of your login scripts (such as `.cshrc` or `.bashrc` depending on your shell).

5 Application Directory

GEOCLAW is run from an application directory that may reside anywhere in your directory structure. An application directory contains files where parameters will be set for each specific problem. GEOCLAW uses several input `.data` files. Standard CLAWPACK computational parameters are set in `amr2ez.data`, such as the physical domain and refinement parameters. Parameters specific to GEOCLAW are in several other `.data` files. These files contain documentation explaining the parameters that they contain. The application directory also contains one required fortran routine `setprob.f`. For standard applications it should only be

necessary to set parameters in the input `.data` files. To make the executable that runs GEOCLAW, type `make ./xgeoclaw` from the application directory. The executable file `xgeoclaw` can then be run from this directory. An example application directory that comes with GEOCLAW is in

```
$CLAW/geoclaw/applications/tsunami/sumatra26dec04/BengalBay.
```

6 The Computational Domain

GEOCLAW can compute on either a latitude-longitude grid or a Cartesian grid. In the former case, the latitude-longitude computational units are mapped internally to spherical physical space using a capacity function that takes into account the true physical geometry of each computational grid cell. Use of this capacity function requires that `mcapa = 2` in `amr2ez.data`, and `maux ≥ 3`. When computing on a latitude-longitude domain the bounds of the domain are specified by longitude (x), and latitude (y) in `amr2ez.data`. Latitude should be specified in the standard -90° to 90° convention, where the southern latitudes are negative. Although it is theoretically possible to compute all the way to the poles, this code has not been tested at extreme latitudes (because of vanishing grid cells at the poles, some numerical issues may arise if computing at extreme latitudes). Typically it is possible to use either the -180° to 180° convention for longitude, or the 0° to 360° convention, where in both cases, the prime meridian corresponds to 0° and longitude increases toward the east. If the computational domain crosses the dateline however, and does not wrap around the entire globe, then the 0° to 360° convention must be used, so that the longitude of the computational domain increases logically. If the computational domain wraps around the entire globe in longitude, then periodic boundary conditions in x should be specified in `amr2ez.data`. In this case, -180° to 180° , degree units could be used for longitude as well, with the computational domain being -180° to 180° , however, it is preferable to place the computational boundary at 0° at the prime meridian rather than at 180° in the middle of the Pacific Ocean which is much more likely to be of interest. When computing on a latitude-longitude grid, output units are in meters and seconds. This cannot be adjusted because of internal calculations.

If you wish to compute on a Cartesian domain, then the physical space and computational space are the same, and domain bounds are set in meters or some other desired unit. It is important that `mcapa = 0` in this case. When computing on a Cartesian domain, units are selected by adjusting the gravitational constant in `setprob.data` appropriately. (See the standard CLAWPACK documentation for more details about the `mcapa` and `naux` variables).

7 Bathymetry

The user must provide a file or files that contain the bathymetric and topographic data. The names of these numeric ascii text files are specified in `settopo.data`. The files must conform to the standard GIS format, where data begins in the northwest corner and advances in longitude first toward the northeast corner before moving to the next latitude south. Several file formats are allowed, as described in `settopo.data`. If your bathymetry data does not conform to one of these standards, you must preprocess your data appropriately before using the files in GEOCLAW.

At the start of the computation, GEOCLAW will read in all of the bathymetry files indicated in `settopo.data` and put the data into arrays. Separate bathymetry files can be nested or overlap in any manner desired, as long as the union of all of the regions in the files is at least as large as the computational domain specified in `amr2ez.data`. Typically there will be at least one file that has bathymetry covering the entire computational region, however, this isn't strictly necessary. GEOCLAW will determine the bathymetry in computational cells by integrating the bathymetry arrays, assigning a value to the computational cell depending on the

region it covers. When two or more bathymetry arrays overlap the same area, GEOCLAW will use the array with the highest resolution.

It is possible to specify idealized bathymetry (such as a simple function) rather than using a data file. (See `settopo.data`).

8 Initialization

GEOCLAW is capable of using dynamic fault models to initialize a tsunami problem. The dynamic model should describe the time-dependent seafloor displacement in some region. To use a dynamic model of this form, see `setdtopo.data`. It is also possible to use a static initialization. In this case the bathymetry will not be altered dynamically, and the problem will be initiated by an initial condition that describes the water surface at the start of the computation. To use this option see `setqinit.data`.

It is possible to instantaneously displace the bathymetry at some time after the start of the computation, in order to generate the tsunami or flood. If this option is desired, use the dynamic fault model option in `setdtopo.data`. The dynamic fault file would then just have a single time in the first column corresponding to the time of displacement.

A NOTE ABOUT RESTARTS: CLAWPACK has a restart feature that allows a computation to be resumed from a previous computation. This is very useful in GEOCLAW since, often, a single real tsunami may be computed many times for studying different regions. When a restart is used in conjunction with a dynamic fault model, it is important to change the initial time in `amr2ez.data` to the computational time at the beginning of the restart. This is so that the state of the bathymetry at the end of the previous computation can be resumed correctly.

9 Time-Stepping and Anisotropic Refinement

The number of grid levels and refinement ratios between grid levels are set in `amr2ez.data`. The variable `mxnest` sets the maximum number of grid levels, and the variable `inrat` is a list of (`mxnest - 1`) integers specifying the refinement ratios between these levels. The default behavior of AMRCLAW is to refine isotropically in space and time. For general hyperbolic conservation laws, maintaining stability (and accuracy) typically requires this equal refinement in time. That is, a grid at *level* n must typically take `inrat(n-1)` time-steps for each time-step of *level* $(n - 1)$ grids. This restriction is due to the CFL condition which, in the case of hyperbolic problems, depends on the speed at which characteristic information propagates. For the shallow water equations, characteristic wave speeds are related to the depth of the water. In *shallow* water, wave speeds are slower, which means that grids in water that is much more shallow than the rest of the domain could take much larger time-steps while still maintaining stability and accuracy. Therefore, for modeling tsunamis, if grids at a particular level are restricted to shoreline regions, or regions with relatively shallow water compared to grids at the next lower level, those grids would not typically require refinement in time as large as refinement in space. For this reason, it is possible to select anisotropic refinement in GEOCLAW. To select this option, `mxnest` should be preceded by a negative sign in `amr2ez.data`. In this case, three lists of integers, `inratx`, `inraty` and `inratt` are required in `amr2ez.data`. The ratios for `inratx` and `inraty` would typically be identical, but `inratt` may have some integers that are smaller for levels that are restricted to shallow regions. Note that the variables `maxleveldeep` and `depthdeep` in `settsunami.data` allow control of grid level restrictions to shallow regions. By carefully tuning `maxleveldeep`, `depthdeep`, `inratx`, `inraty` and `inratt`, one can greatly speed calculations and still maintain stability. An example is shown below.

Suppose that the following values are set in `settsunami.data`:

```

:
:
1.0d2      depthdeep
:
:
2          maxleveldeep
:
:

```

Then it might be feasible to reduce time-steps taken on *level 3* grids since they are restricted to shallow regions, by setting the following variables in `amr2ez.data`:

```

:
:
-3         mxnest
6 6       inratx
6 6       inraty
6 2       inratt
:
:

```

In this case, *level 3* grids will be refined spatially in x and y by a factor of 6, but will only take 2 time-steps for every time-step taken on *level 2* grids. Since the ocean may have depths of up to 4000 meters, it might be possible to get away with even fewer time-steps, possibly 1. (Since wave speeds are related to $\sqrt{\text{depth}}$, a rough calculation suggests that wave speeds will be faster by a factor of $\sqrt{4000/100} \approx 6$ in the deep ocean, implying that time-steps on *level 3* might approach that of time-steps on *level 2*. However, wave speeds are also related to water velocity, which can be quite large near the shoreline, so generally one should be more conservative. When reducing time-step refinement, careful attention should be paid to the Courant numbers reported for each level (which can be output by selecting the `verbose` options in `amr2ez.data`).)

10 Data Output

GEOCLAW uses the same output format as the standard CLAWPACK software. At user specified times (set in `amr2ez.data`) the entire solution on every grid is output into an ascii text file `fort.qnnnn`, where the `nnnn` ranges from 0000 for the initial time to the number of output times. Information about this file, such as the time and the number of grids, is output into `fort.tnnnn` with a corresponding number. These files conform to a standard format that can be read by Matlab graphics routines for visualization (see section 10). The solution in `fort.qnnnn` is grouped by grids—the solution on a particular grid is written in several columns following some header information about that grid’s size and location. Each column of the solution data represents a variable, and each row represents a particular grid cell. The solution data for a grid starts at the lower left corner and advances in the x -direction first (indexed by i). In GEOCLAW, the solution is output as four variables in four columns: $q(i, j, 1)$ is the water depth, $q(i, j, 2)$ and $q(i, j, 3)$ are the momenta (depth times water velocity) in the x and y directions respectively, and $q(i, j, 4)$ is the surface elevation, where 0 is mean sea level. Note that having the surface elevation and the water depth allows determination of the bathymetry in a grid cell, since depth subtracted from surface elevation gives the elevation of the bathymetry or topography. Also, since GEOCLAW solves the shallow water equations in their physically relevant conservative form in order to allow convergence to bores, momenta are output rather than the more

common variable of water velocity. Velocity can be recovered by dividing the momentum by depth. Some care should be exercised when doing this, since velocities at vanishing depths may not be accurate, and division by zero for dry cells should obviously be avoided.

GeoClaw also allows the user to output onto fixed (non-evolving) grids with a user-specified spatial and temporal resolution. This is done by interpolating the actual computational solution on evolving grids to the fixed grids. To use this feature see `setfixedgrids.data`. Some MATLAB routines for plotting the solution on these grids are available in `$CLAW/geoclaw/matlab`. If you want to output the solution at very small time intervals, it is highly recommended to use fixed grids, rather than specifying a large number of output times in `amr2ez.data`, since those output times require the entire computational solution, on coarse and fine grids. This may reduce the size of the computational timesteps, and lead to expensive and highly diffused calculations. It is ideal when **GeoClaw** can choose as large a timestep as possible.

GeoClaw also allows the user to output the solution as a time series at specified locations or “gauges.” To use this feature see `setgauges.data`.

When using the latitude-longitude option to specify the computational domain, all output quantities are in units of meters and seconds. Therefore, in a particular grid cell, the momenta output in the solution are in the locally orthogonal longitude and latitude directions, and given in meters squared per second.

11 Matlab Graphics and Output Visualization

GEOCLAW comes with a set of Matlab plotting routines, `.m` files, in the directory `$CLAW/matlab`, as well as a few routines in `$CLAW/geoclaw/matlab`. Matlab uses the variable `MATLABPATH` when run under Unix, which can also be set in your login scripts to include the paths to these directories.

GEOCLAW uses the **CLAWPACK** Matlab script `plotclaw2.m` as the main plotting routine. Run in Matlab `plotclaw2` from the application directory. A number of plotting options are specified in `setplot2.m`, and `setprob.m` which reside in the application directory (see **CLAWPACK** documentation for descriptions and examples). `plotclaw2` calls `setplot2` if you select `yes` when queried from the Matlab prompt. For some specialized **GEOCLAW** plotting features, set `PlotType=11` or `PlotType=12` in `setplot2.m`. This creates a colored surface plot of the water surface and land, using some colormaps provided. Waves are shaded from red at the peaks to dark blue at the troughs. The default view is directly overhead, but note that these surface plots can be rotated and zoomed, either manually or by adjusting Matlab’s `view` properties. Specific user issued plotting features, such as axis labeling, axis limits, color axis properties for the wave shading, or other options, can be specified in `afterframe.m` in the application directory. The standard **CLAWPACK** documentation provides a full explanation of how to use the Matlab plotting features.

12 Acknowledgements

GEOCLAW is a subset of the existing **CLAWPACK** software. The **CLAWPACK** and **AMRCLAW** routines were written by

Randall J. LeVeque
University of Washington.

The adaptive mesh refinement algorithms were written by

Marsha Berger
Courant Institute, NYU.

Many of the Matlab plotting routines were written by

Donna Calhoun
CEA, Saclay France.

Numerous other people have contributed to CLAWPACK and AMRCLAW in a variety of ways. Development of this software has been supported by grants from the NSF and the DOE.

13 Usage Restrictions

This software is made available for research and instructional use only. You may copy and use this software for these non-commercial purposes at no charge. For all other uses (including distribution of modified versions), please contact the authors. Since TSUNAMICLAW is an extension of CLAWPACK, all of the usage restrictions and copyright notices for CLAWPACK apply to TSUNAMICLAW.

This software is made available “as is,” without any assurance that it will work for your purposes. The software may have defects and is to be used at your own risk.

References

- [1] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35:346–365, 1998.
- [2] D. L. George. *Finite Volume Methods and Adaptive Refinement for Tsunami Propagation and Inundation*. PhD thesis, University of Washington, 2006.
- [3] D. L. George. Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation. *J. Comput. Phys.*, 227(6):3089–3113, March 2008.
- [4] R. J. LeVeque. CLAWPACK software. <http://www.clawpack.org>.
- [5] R. J. LeVeque. *Finite Volume Methods For Hyperbolic Problems*. Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2002.