The Journal of Open Source Software

Mantaray: A Rust Package for Ray Tracing Ocean Surface Gravity Waves

Bryce Irving [1], **Guilherme P. Castelao** [2], **Colin Beyers** [1], **James Clemson** [1], **Jackson Krieger** [1], **Gwendal Marechal** [1], **Nicholas Pizzo** [3], **and Bia Villas Bôas** [1]

1 Colorado School of Mines, Golden, CO, USA **2** National Renewable Energy Laboratory, Golden, CO, USA **3** Graduate School of Oceanography, University of Rhode Island, Narragansett, RI, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Gabriele Bozzola

Reviewers:

- @milancurcic
- @gauteh

Submitted: 08 May 2025
Published: unpublished

Summary

Ocean surface gravity waves are an important component of air-sea interaction, influencing energy, momentum, and gas exchanges across the ocean-atmosphere interface. In specific applications such as refraction by ocean currents or bathymetry, ray tracing provides a computationally efficient way to gain insight into wave propagation. In this paper, we introduce `Mantaray`, an open-source software package implemented in Rust, with a Python interface, that solves the ray equations for ocean surface gravity waves. Mantaray is designed for performance, robustness, and ease of use. The package is modular to facilitate further development and can currently be applied to both idealized and realistic wave propagation problems (Fig. 1).

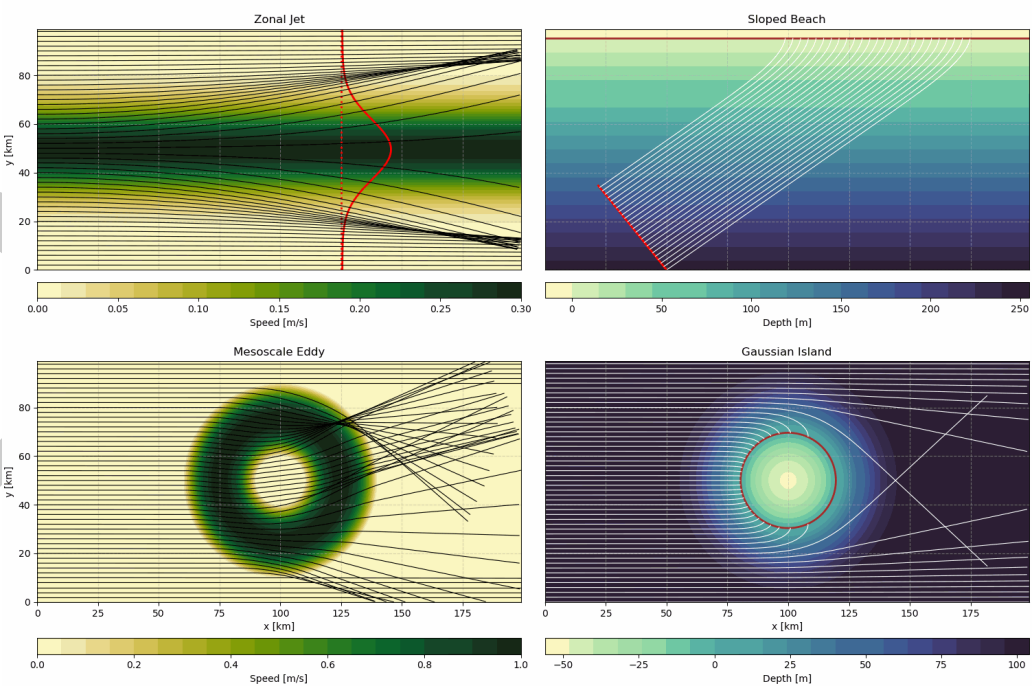

Figure 1: Examples of ray tracing performed using `Mantaray`. Top left: waves in deep water interacting with a zonal jet. Bottom left: waves in deep water interacting with a mesoscale eddy. Top right: waves encountaring varyring bathymethy approaching a linear beach. Bottom right: waves approaching a Gaussian island.

The Journal of Open Source Software

Statement of need

Ray tracing is a long-standing method for investigating wave propagation across a wide range of disciplines, including optics, seismology, and oceanography, providing a simple framework for studying the evolution of waves in spatially varying media. For ocean surface gravity waves, ray-based approaches have been used to study refraction by mesoscale currents (e.g., Mapp et al., 1985; Marechal & Marez, 2022; Romero et al., 2017), changes in bathymetry (Kukulka et al., 2017; e.g., Munk & Traylor, 1947), and statistical effects such as directional diffusion of wave action (e.g., Smit & Janssen, 2019; Villas Bôas & Young, 2020).

Although ray tracing has been widely used in surface wave studies, the software implementations are often written in Fortran or C/C++ (e.g., O'Reilly et al., 2016) and are not always openly available or actively maintained. More recently, open-source Python tools—such as the one by Halsne et al. (2023)—have improved accessibility and reproducibility. Mantaray complements these efforts by providing a ray tracing solution built in Rust, a modern programming language that combines memory safety and execution speed with tools for seamless Python integration. This choice balances the ease-of-use associated with Python and the computational efficiency of compiled languages, filling a gap for users who need robust, high-performance ray tracing within a user-friendly environment.

While Rust is still relatively new in the scientific software ecosystem, especially in oceanography, the development of Mantaray illustrates its potential for broader adoption in geoscientific computing and demonstrates the maturity and capability of Rust for physics solvers. Our package aims to contribute to language diversity in Earth sciences and establish Rust as a top-of-mind language for developing efficient, modern scientific software.

Key Features

Mantaray is composed of two primary layers:

1. Core Engine (Rust): Implements the numerical integration of the ray equations considering stationary (no time dependence) currents $\mathbf{U}(x, y)$ in a Cartesian domain.

   The dispersion relationship for linear surface gravity waves is given by:

$$\sigma = \left[gk \tanh\left(kH(x, y)\right)\right]^{1/2},$$

   where σ is the intrinsic frequency of the waves, g is the gravitational acceleration, k is the wavenumber magnitude, and H is the water depth. The ray equations describing wave propagation can be written as (Phillips, 1966):

$$\mathbf{c_g} = \frac{\partial \sigma}{\partial \mathbf{k}},$$

$$\dot{\mathbf{x}} = \mathbf{c_g} + \mathbf{U}(x, y),$$

$$\dot{\mathbf{k}} = -\boldsymbol{\nabla}\sigma - \boldsymbol{\nabla}\left(\mathbf{k} \cdot \mathbf{U}\right),$$

   where $\mathbf{c}_g$ is the group velocity, $\mathbf{k} = (k_x, k_y)$ is the wavenumber vector, $\mathbf{x} = (x, y)$ is the wave position vector, and the dot notation represents the total time derivative following the wave.

   `Mantaray` integrates the ray equations using a 4th-order Runge-Kutta scheme from the `ode_solvers` crate, with bilinear interpolation for spatial fields such as bathymetry and surface currents.

2. Python Interface: Provides a high-level API for initializing simulations, supplying input fields, and running ray integrations. The current version of the package supports:
    - Cartesian domains with arbitrary bathymetry and current fields input as NetCDF3 files
    - Configurable integration parameters (step size, duration of the integration)
    - Output of ray paths as Xarray Datasets for easy visualization and diagnostics

Note that input is curently limited to NetCDF-3 classic format, but work is ongoing to enable full NetCDF4 compatibility in future releases. `Mantaray` has two main functionalities: `single_ray`, for tracing an individual ray, and `ray_tracing`, for tracing a collection of rays.

Example: The following example illustrates the use of the `single_ray` functionality for tracing a wave that is initially propagating from left to right with a wavelength of 100 m. Note that `bathymetry` and `current` are strings with the path to the respective forcing fields.

```
import numpy as np
import mantaray

# Define initial conditions
k0 = 2*np.pi/100 # initial wavenumber magnitude
theta0 = 0 # initial direction
# Calculates wavenumber components
kx0 = k0*np.cos(phi0)
ky0 = k0*np.sin(phi0)

# Define initial position
x0 = 0
y0 = 500

# Define integration parameters
duration = 1000 # duration in seconds
timestep = 0.1 #timestep in seconds

# Performs integration
ray_path = mantaray.single_ray(x0, y0, kx0, ky0,
                               duration, timestep, bathymetry, current)
```

Example: The `ray_tracing` functionality works similarly, but it takes a collection of initial conditions as `numpy` arrays. In the case below, we are propagating four identical rays, with different initial positions.

```
import numpy as np
import mantaray

# Define initial conditions
k0 = 2*np.pi/100 # initial wavenumber magnitude
theta0 = 0 # initial direction
# Calculates wavenumber components
kx0 = k0*np.cos(phi0)*np.ones(4)
ky0 = k0*np.sin(phi0)*np.ones(4)

# Define initial position
x0 = np.array([0, 0, 0, 0])
y0 = np.array([100, 300, 500, 700])

# Define integration parameters
```

```
duration = 1000 # duration in seconds
timestep = 0.1 #timestep in seconds

# Performs integration
ray_path = mantaray.ray_tracing(x0, y0, kx0, ky0,
                                duration, timestep, bathymetry, current)
```

Acknowledgements

This work was authored in part by NREL for the U.S. Department of Energy (DOE), operated under Contract No. DE-AC36-08GO28308. We thank the Colorado School of Mines Summer Undergraduate Research Fellowship (SURF) and the Mines Undergraduate Research Fellowship (MURF) for partially supporting undergraduate students BI, JC, and JK. BVB was supported by the ONR MURI award N00014-24-1-2554, and NASA award 80NSSC24K1640 through the SWOT Science Team. CB was supported by NASA awards 80NSSC23K0979 through the International Ocean Vector Winds Science Team and 80GSFC24CA067 through the ODYSEA science team as part of the Earth System Explorer program. JK and GM were supported by NASA award 80NSSC24K0411 through the S-MODE Science Team. All co-authors are thankful to Luiz Irber for helpful recommendations throughout the development of this package. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government.

References

Halsne, T., Christensen, K. H., Hope, G., & Breivik, Ø. (2023). Ocean wave tracing v. 1: A numerical solver of the wave ray equations for ocean waves on variable currents at arbitrary depths. *Geoscientific Model Development*, *16*(22), 6515–6530.

Kukulka, T., Jenkins III, R. L., Kirby, J. T., Shi, F., & Scarborough, R. W. (2017). Surface wave dynamics in delaware bay and its adjacent coastal shelf. *Journal of Geophysical Research: Oceans*, *122*(11), 8683–8706.

Mapp, G. R., Welch, C. S., & Munday, J. C. (1985). Wave refraction by warm core rings. *Journal of Geophysical Research: Oceans*, *90*(C4), 7153–7162.

Marechal, G., & Marez, C. de. (2022). Variability of surface gravity wave field over a realistic cyclonic eddy. *Ocean Science*, *18*(5), 1275–1292.

Munk, W. H., & Traylor, M. A. (1947). Refraction of ocean waves: A process linking underwater topography to beach erosion. *The Journal of Geology*, *55*(1), 1–26.

O'Reilly, W. C., Olfe, C. B., Thomas, J., Seymour, R., & Guza, R. (2016). The california coastal wave monitoring and prediction system. *Coastal Engineering*, *116*, 118–132.

Phillips, Owen. M. (1966). *The Dynamics of the Upper Ocean*. Cambridge University Press.

Romero, L., Lenain, L., & Melville, W. K. (2017). Observations of surface wave–current interaction. *Journal of Physical Oceanography*, *47*(3), 615–632.

Smit, P. B., & Janssen, T. T. (2019). Swell propagation through submesoscale turbulence. *Journal of Physical Oceanography*, *49*(10), 2615–2630.

Villas Bôas, A. B., & Young, W. R. (2020). Directional diffusion of surface gravity wave action by ocean macroturbulence. *Journal of Fluid Mechanics*, *890*, R3.