The Journal of Open Source Software

pyCoastal: a Python package for numerical modeling in coastal engineering

Stefano Biondi [1]

1 University of Florida, United States

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 23 July 2025
Published: unpublished

Summary

pyCoastal is an open-source Python framework for the numerical simulation of coastal hydrodynamics and scalar transport processes. It provides a modular and extensible platform for solving linear and nonlinear partial differential equations relevant to coastal systems, including wave propagation, pollutant dispersion, and viscous fluid dynamics. The framework supports structured 1D and 2D Cartesian grids, configurable through lightweight YAML files, and includes reusable components for grid generation, boundary condition management, numerical operators, and time integration schemes. Physical modules are implemented as standalone classes and can be easily composed or extended for prototyping and research. pyCoastal emphasizes clarity and reproducibility, with a strong focus on animated visualization, clean code structure, and pedagogical transparency. It is particularly suited for research prototyping, model intercomparison studies, and educational applications in coastal engineering, fluid mechanics, and numerical modeling.

Statement of need

Numerical modeling of coastal processes, such as wave propagation, shallow water flow, and pollutant transport, typically relies on specialized software frameworks that are often complex to configure, extend, or adapt to new applications. Tools like SWAN, ADCIRC, and OpenFOAM, although powerful, present significant barriers due to their steep learning curves and rigid internal structures. Studies have shown that more computationally demanding models do not necessarily yield higher accuracy, especially when simpler models are tuned effectively (Lashley et al., 2020). Furthermore, successful calibration of numerical models hinges on the ability to accurately represent key physical processes and structural features (Simmons et al., 2017). This complexity poses challenges both for researchers aiming to prototype models rapidly and for instructors seeking clear, demonstrable tools for teaching.

pyCoastal addresses this issue by offering a lightweight and modular coastal modeling framework fully in Python. It is designed to prioritize clarity and reproducibility, allowing users to define simulations through human-readable YAML configuration files and execute them with minimal setup. The codebase provides reusable components for grid generation, numerical operators, time integration schemes, and boundary condition handling, supporting both classical and custom physical models with ease. Its structure is designed to support both research applications and instructional use in topics such as coastal hydrodynamics, numerical modeling, and environmental fluid mechanics. Moreover, the framework integrates numerous established coastal-engineering formulations, such as wave run-up, sediment transport, and boundary layer calculations, enabling users to compute essential coastal parameters with ease. As a result, this module functions as a versatile library suitable for both academic research and industrial applications. In conclusion, pyCoastal offers a balance of flexibility and structure suitable for a range of academic and applied contexts.

Functionality

pyCoastal comprises a collection of interchangeable modules that may be employed independently or combined to construct comprehensive simulation workflows. The Grid module supplies the UniformGrid class for defining one dimensional and two dimensional structured meshes. The Operator module provides finite difference stencils for computing spatial derivatives such as gradient, divergence and Laplacian. The Physics modules implement models for shallow water dynamics, wave advection, pollutant transport and viscous flow. The Boundary module supports Dirichlet, Neumann, sponge layer and wall conditions to represent inflow, outflow and reflective behaviours. Simulation results may be visualised in real time through integration with Matplotlib animation capabilities. All aspects of a simulation—including grid geometry, physical parameters, solver settings and boundary definitions—are specified via a human readable YAML configuration file to ensure full parameterisation and reproducibility.

Examples

The main purpose of this package is to provide a fully python based tool that allows to build, test and play with fluid dynamics numerical modeling. In the following section, some of the pre-built cases are explained :

The `generate_irregular_wave` function builds a band-limited random wave time series based on standard oceanographic spectra:

```python
# -------------------------------------------------------------------
# 3) Generate wave boundary forcing
# -------------------------------------------------------------------
from pyCoastal.tools.wave import generate_irregular_wave

t_vec, eta_bc = generate_irregular_wave(
    Hs=Hs, Tp=Tp,
    duration=duration,
    dt=dt,
    spectrum=spectrum_type,
    gamma=gamma
)
```

Where the inputs for domain and wave generation are taken from a YAML file:

```yaml
grid:
  nx: 200
  ny: 200
  dx: 1.0
  dy: 1.0

physics:
  gravity: 9.81
  depth: 5.0

forcing:
  type: jonswap   # options: pm or jonswap
  gamma: 3.3
  Hs: 0.5         # significant wave height (m)
  Tp: 3.0         # peak period (s)

solver:
  dt: 0.1
  duration: 60.0

output:
  gauge: [100, 100]  # grid indices (i, j)
```

3.1.1. Pierson–Moskowitz (PM) spectrum

This spectrum for a fully-developed sea (Alves et al., 2003) is defined as:

$$S_{PM}(f) = \frac{5}{16} H_s^2 f_p^4 f^{-5} \ \exp\left[-\frac{5}{4} \left(\frac{f_p}{f} \right)^4 \right],$$

where: - $S_{PM}(f)$ is the spectral energy density [m²/Hz]
- H_s is the significant wave height [m]
- f_p is the peak frequency [Hz], with $f_p = 1/T_p$
- T_p is the peak wave period [s]
- f is the frequency [Hz]

3.1.2 JONSWAP spectrum

This spectrum modifies the PM with a peaked enhancement factor (Hasselmann et al., 1973) as:

$$S_J(f) = S_{PM}(f)\ \gamma^{\exp\left[-\frac{1}{2\sigma^2} \left(\frac{f}{f_p} - 1 \right)^2 \right]},$$

where: - $S_J(f)$ is the JONSWAP spectral energy density [m²/Hz]
- γ is the peak enhancement factor (typically 3.3)
- σ is the spectral width parameter, with $\sigma = 0.07$ for $f < f_p$ and $\sigma = 0.09$ for $f > f_p$

Once $S(f)$ is defined, the surface elevation time series is:

$$\eta(t) = \sum_i A_i \cos(2\pi f_i + \phi_i),$$

where: - $\eta(t)$ is the free-surface elevation at time t [m]
- A_i is the wave amplitude for frequency f_i [m]
- ϕ_i is a random phase shift in $[0, 2\pi]$
- Δf is the frequency resolution

The amplitudes A_i are given by:

$$A_i = \sqrt{2\, S(f_i)\, \Delta f}.$$

To test it, run this example:

```
python examples/wave2D_irregular.py
```

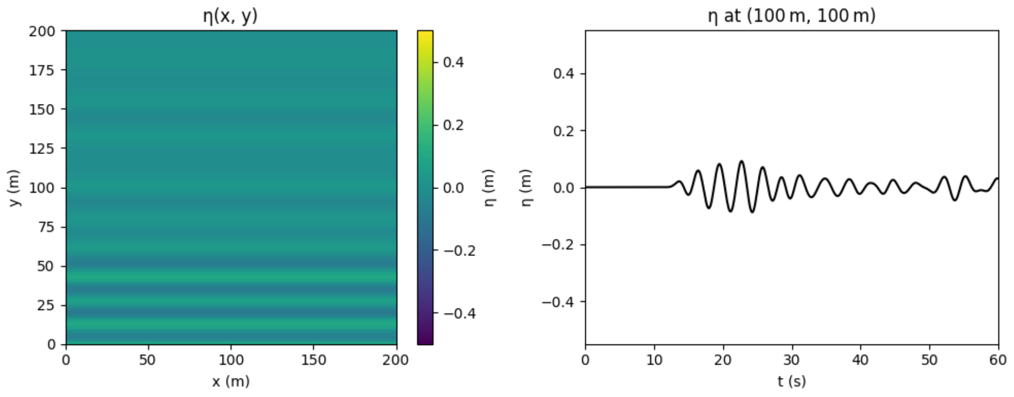

Figure 1: Irregular wave field output of the example. The left subplot shows the upper view of the wave field (incoming from the south boundary). The side boundaries are set as free-slip conditions, while the northern boundary has a wave absorption condition. The subplot on the right shows the surface elevation over time at observation points.

3.2 2D Water Drop (Circular Wave Propagation)

This example demonstrates the classic 2D linear wave equation for the surface elevation η in time with celerity c:

$$\frac{\partial^2 \eta}{\partial t^2} = c^2 \nabla^2 \eta,$$

and solves in the domain with zero-Dirichlet boundary conditions on all edges and an initial Gaussian hump at the center. The time-stepping update is a second-order explicit scheme:

$$\eta^{n+1} = 2\eta^n - \eta^{n-1} + (c\Delta t)^2 \nabla^2 \eta^n.$$

To test, run:

```
python examples/water_drop.py
```

The output provides real-time animation, allowing users to visually observe expanding circular wavefronts and their reflections. Additionally, it is fully configurable via YAML, enabling easy adjustment of domain size, resolution, wave speed, CFL number, and simulation duration without modifying the code.

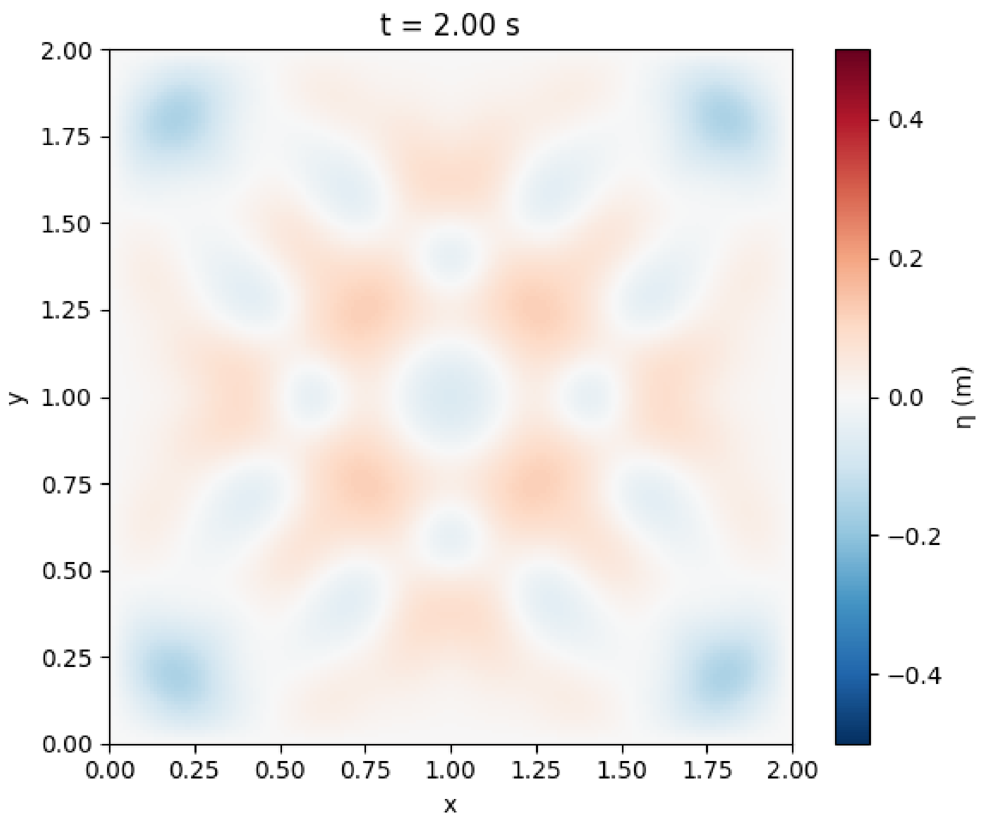

Figure 2: Solution of the classic 2D linear wave equation. The colormap represents the water surface elevation.

Run the Example

```
python examples/water_drop.py
```

Conclusion Future directions

The examples shown here illustrate only a subset of the capabilities of pyCoastal. The existing tools already cover a broad range of coastal-hydrodynamics problems and are designed for future expansion into new application areas. In the next development phase, pyCoastal will extend support to spatially varying bathymetry, wave energy dissipation and breaking models, and fully coupled nearshore hydrodynamics. To date, the framework provides an extensive library of coastal-engineering formulations for calculating transport-related coefficients and wave parameters, such as sediment transport rates, dispersion relations, and wave setup coefficients, which will form the foundation for these forthcoming enhancements.

Acknowledgements

I thank professors Alberto Canestrelli (UF) and Donald Slinn (UF) for the teachings in the numerical modeling and Hydrodynamics field.

References

see paper.bib

Alves, J. H. G. M., Banner, M. L., & Young, I. R. (2003). Revisiting the pierson–moskowitz asymptotic limits for fully developed wind waves. *Journal of Physical Oceanography*, *33*(7), 1301–1323. https://doi.org/10.1175/1520-0485(2003)033%3C1301:RTPALF%3E2.0.CO;2

Hasselmann, K., Barnett, T., Bouws, E., Carlson, H., Cartwright, D., Enke, K., Ewing, J., Gienapp, H., Hasselmann, D., Kruseman, P., Meerburg, A., Müller, P., Olbers, D., Richter, K., Sell, W., & Walden, H. (1973). Measurements of wind-wave growth and swell decay during the joint north sea wave project (JONSWAP). *Deutsche Hydrographische Zeitschrift*, *8*, 1–95.

Lashley, C. H., Zanuttigh, B., Bricker, J. D., Meer, J. van der, Altomare, C., Suzuki, T., Roeber, V., & Oosterlo, P. (2020). Benchmarking of numerical models for wave overtopping at dikes with shallow mildly sloping foreshores: Accuracy versus speed. *Environmental Modelling & Software*, *130*, 104740. https://doi.org/10.1016/j.envsoft.2020.104740

Simmons, J. A., Harley, M. D., Marshall, L. A., Turner, I. L., Splinter, K. D., & Cox, R. J. (2017). Calibrating and assessing uncertainty in coastal numerical models. *Coastal Engineering*, *125*, 28–41. https://doi.org/10.1016/j.coastaleng.2017.04.005