Oceanic Eddy & Front Simulation

June 29, 2025

1 Functionalities

The `OceanicEddyAndFrontWindow` and `EddySimulationWidget` classes provide core functionalities for simulating and visualizing oceanic eddies and fronts:

- **Initialization**: Sets up a simulation with a coarse grid (size 10–200) and synthetic eddy data (1–10 eddies, strength 0.5–2.0, radius 0.1–0.3), supporting non-hydrostatic dynamics ($< 1\,\mathrm{km}$ grid spacing), baroclinic/barotropic instabilities, mixed-layer instability (MLI, $< 10\,\mathrm{km}$), and potential vorticity (PV) frontogenesis.

- **Simulation Control**: Manages start, pause, reset, and export operations, with input validation for parameters like grid spacing (0.1–10.0 km), Coriolis parameter ($10^{-5}10^{-3}\,\mathrm{s}^{-1}$), and Rossby number (0.05–0.5).

- **Visualization**: Displays a 400x400 pixel image of vorticity, colored by vorticity magnitude and gradient, with optional overlays for vertical velocity, density, velocity shear, buoyancy, and PV. Fine grid patches (refinement factor 2–8) are shown for high-vorticity regions, and streamlines visualize flow.

- **Parameter Input**: Allows configuration of grid size, eddy properties, vorticity diffusion (0.005–0.02), rotation rate (0.05–0.2 $\mathrm{rad/s}$), background flow (0.02–0.1 $\mathrm{m/s}$), and instability parameters (e.g., baroclinic shear, mixed-layer depth).

- **Data Export**: Exports simulation data (vorticity, vertical velocity, density, shear, buoyancy, PV statistics) to a timestamped text file.

2 Simulation Logic

The simulation logic in `OceanicEddyAndFrontWindow` and `EddySimulationWidget` manages eddy and front dynamics and visualization.

2.1 Initialization

- **Purpose**: Configures the simulation with synthetic eddy data and fields.
- **Process**:
  - Initializes coarse grid ($N \times N$, $N = 10200$) over $[-1, 1] \times [-1, 1]$ with spacing $\Delta x = 2/N$.
  - Sets up 1–10 eddies with random centers ($x_0, y_0 \in [-0.8, 0.8]$), strengths ($0.81.2 \times$ input strength), and radii ($0.81.2 \times$ input radius).
  - Computes vorticity: $\zeta = \sum \zeta_i \exp(-r^2/(2R_i^2)) \cdot (1 - Ro \tanh(r/R_i))$, where $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, R_i is eddy radius, and Ro is Rossby number.
  - Initializes non-hydrostatic fields (w, p') for grid spacing $< 1\,\mathrm{km}$, density (ρ) for baroclinic instability, velocity shear for barotropic instability, buoyancy for MLI ($< 10\,\mathrm{km}$), and PV for frontogenesis.
  - Creates fine grid patches (size floor$(N/5) \times$ refinement factor) for eddies with strength above threshold.
  - Sets up visualization with a 400x400 QImage, logging actions to `debug.log`.

2.2 Simulation Control

- **Purpose**: Manages simulation operations (start, pause, reset, export).
- **Process (Start)**:
  - Validates inputs: grid size (10–200), grid spacing (0.1–10.0 km), eddy strength (0.5–2.0), etc.
  - Initializes fields if not set, starts a 100 ms timer to call `update_simulation`.
  - Disables start button, enables pause/reset/export buttons, logs to `debug.log`.
- **Process (Pause)**:
  - Stops timer, enables start/reset/export buttons, disables pause button, logs to `debug.log`.
- **Process (Reset)**:
  - Stops timer, clears fields (vorticity, velocity, density, etc.), resets time and eddy data.
  - Enables reset/initialize buttons, disables start/pause/export, logs to `debug.log`.
- **Process (Export)**:

  - Collects simulation data (time, grid size, eddy info, statistics) and writes to a timestamped file.
  - Logs export action, displays success message via `QMessageBox`.

2.3 Visualization Update

- **Purpose**: Updates the visualization of eddy and front dynamics.
- **Process**:
  - Fills 400x400 QImage with white, normalizes vorticity (ζ) and its gradient magnitude ($|\nabla\zeta|$).
  - Colors pixels: red ($255\times$grad if grad > 0.7), green ($255\times(1-$vorticity$)(1-$grad$)$), blue ($255 \times$ vorticity$(1-$ grad$)$).
  - Adjusts colors for non-hydrostatic (vertical velocity, pressure), baroclinic (density), barotropic (shear), MLI (buoyancy), and PV frontogenesis effects.
  - Visualizes fine grids with higher resolution, draws red rectangles around patches.
  - Draws 10 streamlines starting at random points, using induced velocities.
  - Updates widget via `paintEvent`.

3 Physics and Mathematical Models

The simulation implements models for oceanic eddy and front dynamics.

3.1 Vorticity Field

- **Purpose**: Models eddy vorticity with ageostrophic effects.
- **Equations**:

$$\begin{aligned}
r &= \sqrt{(x-x_0)^2 + (y-y_0)^2}, \\
\zeta_i &= \begin{cases} \zeta_0(r/R_i), & r < R_i, \\ \zeta_0(R_i/r), & r \geq R_i, \end{cases} \\
\zeta &= \sum_i \zeta_i \exp\left(-\frac{r^2}{2R_i^2}\right)\left(1 - Ro\tanh\left(\frac{r}{R_i}\right)\right),
\end{aligned}$$

  where ζ_0 is eddy strength, R_i is radius, Ro is Rossby number, clipped to $[-10, 10]$.

- **Implementation**: `initialize_field` and `update_simulation` compute total vorticity.

3.2 Induced Velocity

- **Purpose**: Computes velocity field with Coriolis effects.
- **Equations**:

$$\begin{aligned} u_{\text{geo}} &= -\frac{\zeta_0(y-y_0)}{2\pi r^2}, \\ v_{\text{geo}} &= \frac{\zeta_0(x-x_0)}{2\pi r^2}, \\ u_{\text{ageo}} &= -Ro\, v_{\text{geo}}, \\ v_{\text{ageo}} &= Ro\, u_{\text{geo}}, \\ u &= \sum_i (u_{\text{geo}} + f u_{\text{ageo}}), \\ v &= \sum_i (v_{\text{geo}} + f v_{\text{ageo}}), \end{aligned}$$

  where f is the Coriolis parameter, $r^2 > 10^{-6}$.

- **Implementation**: `compute_induced_velocity` calculates velocities for streamlines and eddy motion.

3.3 Non-Hydrostatic Effects

- **Purpose**: Models vertical velocity and pressure perturbation for scales $< 1\,\text{km}$.
- **Equations**:

$$\begin{aligned} w &= -0.1\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right), \\ p' &= -0.05\zeta, \end{aligned}$$

  where $\frac{\partial u}{\partial x}$, $\frac{\partial v}{\partial y}$ are computed via finite differences ($\Delta x = 2/N$).

- **Implementation**: `compute_nonhydrostatic_effects` updates fields.

3.4 Baroclinic Instability

- **Purpose**: Models vorticity tendency from density gradients.
- **Equations**:

$$\begin{aligned} \frac{\partial \zeta}{\partial t} &= -0.1\left(\frac{\partial \rho}{\partial x} + \frac{\partial \rho}{\partial y}\right), \\ \frac{\partial \rho}{\partial t} &= -\left(\frac{\partial \zeta}{\partial x}\frac{\partial \rho}{\partial x} + \frac{\partial \zeta}{\partial y}\frac{\partial \rho}{\partial y}\right), \end{aligned}$$

where $\rho \in [1020, 1030]\,\mathrm{kg/m^3}$, gradients use finite differences.

- **Implementation**: `compute_baroclinic_effects` updates vorticity and density.

3.5 Barotropic Instability

- **Purpose**: Models vorticity tendency from velocity shear.
- **Equations**:

$$\frac{\partial \zeta}{\partial t} = 0.05\left(\frac{\partial s}{\partial x} - \frac{\partial s}{\partial y}\right),$$
$$\frac{\partial s}{\partial t} = -\left(\frac{\partial \zeta}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial \zeta}{\partial y}\frac{\partial s}{\partial y}\right),$$

where $s \in [-0.1, 0.1]$ is shear, clipped after updates.

- **Implementation**: `compute_barotropic_effects` updates vorticity and shear.

3.6 Mixed-Layer Instability

- **Purpose**: Models vorticity tendency from buoyancy gradients ($< 10\,\mathrm{km}$).
- **Equations**:

$$\frac{\partial \zeta}{\partial t} = 0.1h\left(\frac{\partial b}{\partial x} - \frac{\partial b}{\partial y}\right),$$
$$\frac{\partial b}{\partial t} = -\left(\frac{\partial \zeta}{\partial x}\frac{\partial b}{\partial x} + \frac{\partial \zeta}{\partial y}\frac{\partial b}{\partial y}\right),$$

where $b \in [-0.1, 0.1]$ is buoyancy, h is mixed-layer depth.

- **Implementation**: `compute_mli_effects` updates vorticity and buoyancy.

3.7 Potential Vorticity Frontogenesis

- **Purpose**: Models PV and vorticity tendencies from strain.
- **Equations**:

$$\frac{\partial q}{\partial t} = -\sigma\left(\left(\frac{\partial q}{\partial x}\right)^2 + \left(\frac{\partial q}{\partial y}\right)^2\right),$$
$$\frac{\partial \zeta}{\partial t} = 0.05\frac{\partial q}{\partial t},$$
$$\frac{\partial q}{\partial t} = -\left(\frac{\partial \zeta}{\partial x}\frac{\partial q}{\partial x} + \frac{\partial \zeta}{\partial y}\frac{\partial q}{\partial y}\right),$$

where $q \in [-1, 1]$ is PV, σ is strain rate.

- **Implementation**: `compute_pv_frontogenesis_effects` updates PV and vorticity.

4 Algorithms

4.1 Initialization Algorithm

- **Input**: Grid size (N), eddy strength, radius, number of eddies, refinement threshold, factor, Coriolis parameter, Rossby number, grid spacing, instability parameters.
- **Steps**:
  1. Log initialization to `debug.log`.
  2. Validate inputs: $N \in [10, 200]$, grid spacing $\in [0.1, 10]\,\mathrm{km}$, etc.
  3. Create grid: $x, y \in [-1, 1]$, $\Delta x = 2/N$, $X, Y =$ meshgrid(x, y).
  4. Initialize vorticity: $\zeta = 0$, size $N \times N$.
  5. For each eddy (1–10):
     - Set random center $(x_0, y_0) \in [-0.8, 0.8]$, strength ($0.81.2 \times$ input), radius ($0.81.2 \times$ input).
     - Compute $r = \sqrt{(X - x_0)^2 + (Y - y_0)^2}$, $\zeta_i = \zeta_0(r/R_i$ if $r < R_i,$ else $R_i/r)$.
     - Add $\zeta_i \exp(-r^2/(2R_i^2))(1 - Ro \tanh(r/R_i))$ to ζ.
  6. If non-hydrostatic (grid spacing $< 1\,\mathrm{km}$):
     - Initialize $w, p' = 0$, add $w = \zeta_0 0.1 \exp(-r^2/R_i^2) \sin(\arctan 2(Y - y_0, X - x_0))$, $p' = \zeta_0 0.05 \exp(-r^2/R_i^2)$.
  7. If baroclinic: Initialize $\rho = 1025 + sYN_s$, clip to $[1020, 1030]\,\mathrm{kg/m^3}$, add $\zeta_0 0.01 \exp(-r^2/(2R_i^2))$.
  8. If barotropic: Initialize shear $s = sY$, clip to $[-0.1, 0.1]$, add $\zeta_0 0.05 \exp(-r^2/(2R_i^2))$.
  9. If MLI (grid spacing $< 10\,\mathrm{km}$): Initialize $b = b_g Y + 0.01 \sin(2\pi X/0.2)$, clip to $[-0.1, 0.1]$, add $\zeta_0 0.02 \exp(-r^2/(2R_i^2))$.
  10. If PV frontogenesis: Initialize $q = \zeta + f(b/h)/N_s$ (if MLI), add $\zeta_0 f_{\mathrm{pv}} \exp(-r^2/(2R_i^2))$, clip to $[-1, 1]$.
  11. For eddies with $|\zeta_0| >$ threshold: Create fine grid (floor$(N/5) \times$refinement factor), compute fine vorticity and fields.
  12. Call `update_visualization`, log completion or errors via `QMessageBox`.

4.2 Start Simulation Algorithm

- **Input**: Simulation parameters (vorticity diffusion, rotation rate, etc.).
- **Steps**:
  1. Log start to `debug.log`.
  2. If $\zeta =$ None, call `initialize_simulation`.
  3. Validate inputs: vorticity diffusion (0.005–0.02), rotation rate (0.05–0.2 $\mathrm{rad/s}$), etc.
  4. Connect timer (100 ms) to `update_simulation`.
  5. Disable start button, enable pause/reset/export, log completion or errors.

4.3 Pause Simulation Algorithm

- **Input**: None.
- **Steps**:
  1. Log pause to `debug.log`.
  2. Stop timer, enable start/reset/export buttons, disable pause button.
  3. Log completion or errors via `QMessageBox`.

4.4 Reset Simulation Algorithm

- **Input**: None.
- **Steps**:
  1. Log reset to `debug.log`.
  2. Stop timer, clear fields ($\zeta, w, p', \rho, s, b, q$), reset time, eddy data, fine grids.
  3. Enable reset/initialize buttons, disable start/pause/export, clear QImage.
  4. Call `update`, log completion or errors via `QMessageBox`.

4.5 Update Simulation Algorithm

- **Input**: Simulation parameters.
- **Steps**:

1. Log update to `debug.log`.
2. Compute $\Delta t = \min(0.1, 0.5\Delta x / \max(|\text{background flow}|, 0.1))$, increment time.
3. Update eddy centers: $x_0 += (\text{background flow} + u_{\text{ind}})\Delta t$, $y_0 += v_{\text{ind}}\Delta t$, wrap if $x_0 > 1$.
4. Decay vorticity: $\zeta \to \zeta \exp(-\text{decay rate}\Delta t)$.
5. Compute new vorticity: $\zeta_{\text{new}} = \sum \zeta_i \exp(-r^2/(2R_i^2))(1 - Ro \tanh(r/R_i))$.
6. Update vorticity: $\zeta = \zeta_{\text{new}} + \nu\Delta t\nabla^2\zeta - \sigma\Delta t(x\frac{\partial\zeta}{\partial x} + y\frac{\partial\zeta}{\partial y}) + \text{background flow}(0.1Y)$, clip to $[-10, 10]$.
7. Apply instabilities:
   - Baroclinic: Add $\Delta t \cdot (-0.1(\frac{\partial\rho}{\partial x} + \frac{\partial\rho}{\partial y}))$, update ρ.
   - Barotropic: Add $\Delta t \cdot 0.05(\frac{\partial s}{\partial x} - \frac{\partial s}{\partial y})$, update s.
   - MLI: Add $\Delta t \cdot 0.1h(\frac{\partial b}{\partial x} - \frac{\partial b}{\partial y})$, update b.
   - PV frontogenesis: Add $\Delta t \cdot 0.05(-\sigma((\frac{\partial q}{\partial x})^2 + (\frac{\partial q}{\partial y})^2))$, update q.
8. If non-hydrostatic: Update w, p'.
9. Update fine grids, interpolate to coarse grid.
10. Call `update_visualization`, log completion or errors.

4.6 Export Simulation Algorithm

- **Input**: None.
- **Steps**:
  1. Log export to `debug.log`.
  2. Validate: If $\zeta = \text{None}$, raise error.
  3. Collect data: time, grid size, eddy info, statistics ($\min, \max, \text{mean}, \text{std}$) for vorticity, vertical velocity, density, shear, buoyancy, PV.
  4. Write to `oceanic_eddies_output_YYYYMMDD_HHMM.txt`.
  5. Log completion, show success via `QMessageBox`, or errors if failed.