Adaptive Mesh Refinement

June 29, 2025

1 Functionalities

The `AdaptiveMeshRefinement` class provides the following functionalities:

- **Initialization**: Sets up the adaptive mesh refinement system with a specified grid size and refinement threshold, maintaining a list of cells targeted for refinement.
- **Refinement Computation**: Identifies grid cells requiring refinement based on temperature gradients and vorticity of ocean and atmosphere temperature fields.
- **Grid Refinement**: Refines the resolution of specified grid cells by interpolating values from neighboring cells, applied to temperature or salinity fields.
- **Logging and Error Handling**: Logs initialization, refinement computation, and grid refinement operations using the `logging` module, with exception handling to catch and log errors.

2 Simulation Logic

The simulation logic in `AdaptiveMeshRefinement.py` centers around the `AdaptiveMeshRef` class, which manages dynamic grid refinement for the simulator.

2.1 Initialization

- **Purpose**: Initializes the adaptive mesh refinement system with grid parameters and a threshold for refinement criteria.
- **Process**:
  - Initializes parameters: `grid_size` (N) and `threshold`, which determines when refinement is triggered based on gradients or vorticity.
  - Creates an empty list (`refined_cells`) to store coordinates of cells

targeted for refinement.

  – Logs initialization details and handles exceptions.

2.2 Refinement Computation

- **Purpose**: Identifies grid cells for refinement based on physical criteria derived from ocean and atmosphere temperature fields.
- **Process**:
  – Clips input temperature fields (T_o, T_a) to $[250, 350]$ K to prevent extreme values.
  – Computes gradients of T_o and T_a (∇T_o, ∇T_a) using numerical differentiation.
  – Calculates a combined gradient magnitude across both fields.
  – Computes simplified 2D vorticity from ∇T_o.
  – Creates a refinement mask based on cells where gradient magnitude or vorticity exceeds the threshold.
  – Stores coordinates of cells to refine in `refined_cells`.
  – Returns the refinement mask for use in `Model.py`.

2.3 Grid Refinement

- **Purpose**: Refines specified grid cells by interpolating field values from neighboring cells.
- **Process**:
  – Copies the input field (T) to avoid in-place modification.
  – For each cell in `refined_cells`, creates a 2×2 subgrid with interpolated values based on the current and neighboring grid points.
  – Assigns the mean of the subgrid to the cell, clipped to $[250, 350]$ K.
  – Returns the refined field.

3 Physics and Mathematical Models

The `AdaptiveMeshRefinement` class implements models to identify and refine grid cells based on physical criteria, using numerical approximations for gradients and vorticity.

3.1 Gradient Magnitude

- **Purpose**: Computes the combined gradient magnitude of ocean and atmosphere temperature fields to identify regions with steep changes.
- **Equation**:

$$\nabla T_o = \left(\frac{\partial T_o}{\partial x}, \frac{\partial T_o}{\partial y}\right),$$
$$\nabla T_a = \left(\frac{\partial T_a}{\partial x}, \frac{\partial T_a}{\partial y}\right),$$
$$|\nabla| = \sqrt{\left(\frac{\partial T_o}{\partial x}\right)^2 + \left(\frac{\partial T_o}{\partial y}\right)^2 + \left(\frac{\partial T_a}{\partial x}\right)^2 + \left(\frac{\partial T_a}{\partial y}\right)^2},$$

  where gradients are approximated using central differences, and each term is clipped to $[-10^{10}, 10^{10}]$ before computing the magnitude, which is clipped to $[0, 10^5]$.

- **Implementation**: `compute_refinement` uses `np.gradient` to compute ∇T_o and ∇T_a, ensuring numerical stability through clipping.

3.2 Vorticity

- **Purpose**: Computes a simplified 2D vorticity from ocean temperature gradients to identify regions of rotational flow.
- **Equation**:

$$\omega_x = \frac{\partial}{\partial y}\left(\frac{\partial T_o}{\partial x}\right),$$
$$\omega_y = \frac{\partial}{\partial x}\left(\frac{\partial T_o}{\partial y}\right),$$
$$\omega = |\omega_x - \omega_y|,$$

  where gradients are approximated using central differences, and ω is clipped to $[0, 10^5]$.

- **Implementation**: `compute_refinement` applies `np.gradient` to ∇T_o components to compute vorticity, used as a refinement criterion.

3.3 Refinement Criteria

- **Purpose**: Determines which grid cells require refinement.
- **Equation**:

$$\text{mask}_{i,j} = (|\nabla|_{i,j} > \theta) \text{ or } (\omega_{i,j} > \theta/10),$$

  where θ is the `threshold` parameter.

- **Implementation**: `compute_refinement` generates a boolean mask for cells where either condition is met, storing coordinates in `refined_cells`.

3.4 Grid Refinement

- **Purpose**: Interpolates field values in targeted cells to increase resolution.
- **Equation**:

$$\begin{aligned}
T_{\text{subgrid},0,0} &= 0.5T_{i,j} + 0.125(T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1}), \\
T_{\text{subgrid},0,1} &= 0.5T_{i,j} + 0.125(T_{i-1,j} + T_{i+1,j} + T_{i,j+1} + T_{i,j-1}), \\
T_{\text{subgrid},1,0} &= 0.5T_{i,j} + 0.125(T_{i+1,j} + T_{i-1,j} + T_{i,j-1} + T_{i,j+1}), \\
T_{\text{subgrid},1,1} &= 0.5T_{i,j} + 0.125(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}), \\
T_{\text{refined},i,j} &= \text{mean}(T_{\text{subgrid}}),
\end{aligned}$$

  where $T_{\text{refined},i,j}$ is clipped to $[250, 350]$ K.

- **Implementation**: `refine` applies this interpolation to each cell in `refined_cells`, averaging the subgrid to update the field value.

4 Algorithms

4.1 Initialization Algorithm

- **Input**: `grid_size` (N), `threshold`.
- **Steps**:
  1. Log initialization parameters: `grid_size`, `threshold`.
  2. Store `grid_size` and `threshold` as instance variables.
  3. Initialize an empty list `refined_cells`.
  4. Log completion of initialization.
  5. Handle exceptions and log errors if initialization fails.

4.2 Refinement Computation Algorithm (`compute_refinement`)

- **Input**: Ocean temperature T_o, atmosphere temperature T_a.
- **Steps**:
  1. Log start of computation.
  2. Clip T_o and T_a to $[250, 350]$ K.

3. Compute gradients: $\nabla T_o = (\frac{\partial T_o}{\partial x}, \frac{\partial T_o}{\partial y})$, $\nabla T_a = (\frac{\partial T_a}{\partial x}, \frac{\partial T_a}{\partial y})$ using `np.gradient`.

4. Compute gradient magnitude: $|\nabla| = \sqrt{\left(\frac{\partial T_o}{\partial x}\right)^2 + \left(\frac{\partial T_o}{\partial y}\right)^2 + \left(\frac{\partial T_a}{\partial x}\right)^2 + \left(\frac{\partial T_a}{\partial y}\right)^2}$, with terms clipped to $[-10^{10}, 10^{10}]$ and result to $[0, 10^5]$.

5. Compute vorticity: $\omega_x = \frac{\partial}{\partial y}\left(\frac{\partial T_o}{\partial x}\right)$, $\omega_y = \frac{\partial}{\partial x}\left(\frac{\partial T_o}{\partial y}\right)$, $\omega = |\omega_x - \omega_y|$, clipped to $[0, 10^5]$.

6. Create refinement mask: $\text{mask}_{i,j} = (|\nabla|_{i,j} > \theta)$ or $(\omega_{i,j} > \theta/10)$.

7. Store coordinates (i, j) where $\text{mask}_{i,j} =$ True in `refined_cells` for $i, j \in [1, N-2]$.

8. Return `mask`.

9. Log completion or errors if computation fails.

4.3 Grid Refinement Algorithm (`refine`)

- **Input**: Field T (e.g., temperature or salinity).
- **Steps**:
  1. Log start of refinement.
  2. Copy T to `T_refined` to avoid in-place modification.
  3. For each (i, j) in `refined_cells`:
     - Create a 2×2 subgrid:
       * $T_{\text{subgrid},0,0} = 0.5T_{i,j} + 0.125(T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1})$.
       * $T_{\text{subgrid},0,1} = 0.5T_{i,j} + 0.125(T_{i-1,j} + T_{i+1,j} + T_{i,j+1} + T_{i,j-1})$.
       * $T_{\text{subgrid},1,0} = 0.5T_{i,j} + 0.125(T_{i+1,j} + T_{i-1,j} + T_{i,j-1} + T_{i,j+1})$.
       * $T_{\text{subgrid},1,1} = 0.5T_{i,j} + 0.125(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1})$.
     - Set $T_{\text{refined},i,j} = \text{mean}(T_{\text{subgrid}})$, clipped to $[250, 350]$ K.
  4. Return `T_refined`.
  5. Log completion or errors if refinement fails.