Plasma Instabilities with a 1D Particle-In-Cell

Marty-Bazan Jeanne, Lolivier Antonin

May 25, 2024

CONTENTS:

CHAPTER ONE

SRC

1.1 FileHelper module

class FileHelper.**FileHelper**

Bases: `object`

Class for handling file operations.

# Methods:

- ***create_test_directory()*: Creates a directory to save the simulation results based on the parameters in the**
  'Parameters.json' file. (Static Method)
- ***get_test_directory_name()*: Get the name of the directory where the simulation**
  results are saved. (Static Method)

static create_test_directory()

Creates a filename for the plot based on the parameters in the 'Parameters.json' file.

# Returns:
str: The filename for the plot.

# Raises:
ValueError: If the directory already exists.

static get_test_directory_name()

Get the name of the directory where the simulation results are saved.

# Returns:
str: the name of the directory where the simulation results are saved.

1.2 MainProgram module

class MainProgram.**MainProgram**

Bases: `object`

Main class to run the Particle-in-Cell simulation of a 1D plasma and visualize the results.

# Attributes:

- *parameters (dict)*: Dictionary containing simulation parameters loaded from 'Parameters.json'.
- *directory_name (str)*: Name of the directory where the simulation results are saved.

# Methods:

- *set_parameters()*: Loads simulation parameters from 'Parameters.json'.
- *execute()*: Execute the simulation and visualizes the results using the Simulation and Visualizer classes.

`execute()`

Execute the simulation and visualizes the results using the Simulation and Visualizer classes.

`set_parameters()`

Loads simulation parameters from 'Parameters.json'.

`simulate()`

Runs the simulation using the Simulation class.

`visualize()`

Visualizes the results of the simulation using the Visualizer class.

1.3 Simulation module

`class Simulation.Simulation`(*parameters*, *directory_name*)

Bases: `object`

Class for simulating a 1D plasma using the Particle-in-Cell method.

# Args:

- *parameters (dict)*: Dictionary containing simulation parameters.
- *directory_name (str)*: Name of the directory where the simulation results will be saved.

# Attributes:

- *dx (float)*: Spatial resolution.
- *EPSILON_0 (float)*: Permittivity of free space.
- *FACTOR (float)*: Factor used in the calculation of potential.
- *particles_number (int)*: Number of particles in the simulation.
- *particle_charge (float)*: Charge of each particle.
- *particle_mass (float)*: Mass of each particle.
- *domain_size (float)*: Size of the simulation domain.
- *cells_number (int)*: Number of cells in the simulation domain.
- *max_initial_velocity_deviation (float)*: Maximum deviation in initial velocities.
- *iterations_number (int)*: Number of simulation iterations.
- *potential_matrix (numpy.ndarray)*: Matrix used in the calculation of potential.
- *positions (numpy.ndarray)*: Array containing particle positions for each iteration.
- *velocities (numpy.ndarray)*: Array containing particle velocities for each iteration.
- *dt (numpy.ndarray)*: Array containing time steps for each iteration.

# Methods:

- *set_initial_conditions()*: Set initial conditions for the simulation.
- *init_arrays()*: Initialize arrays and matrices for the simulation.
- *compute_charge_density(iteration)*: Compute charge density at a given iteration.
- *compute_potential(density)*: Compute potential based on charge density.
- *compute_electric_field(potential)*: Compute electric field based on potential.
- *compute_particle_electric_field(electric_field, iteration)*: Compute particle electric field.
- *compute_force(particle_electric_field)*: Compute force on particles.
- *compute_time_step(iteration)*: Compute time step based on particle velocities.
- *update_positions_velocities(force, iteration)*: Update particle positions and velocities using Euler method.
- *save_results()*: Save simulation results to 'results.npz'.
- *run()*: Run the simulation.

`compute_charge_density()`

Compute charge density for the current iteration.

# Returns:
numpy.ndarray: Charge density array.

`compute_electric_field(`*potential*`)`

Compute electric field based on potential.

# Args:

- *potential (numpy.ndarray)*: Potential array.

# Returns:
None

`compute_force(`*particle_electric_field*`)`

Compute force on particles.

# Args:

- *particle_electric_field (numpy.ndarray)*: Array containing electric field for each particle.

# Returns:
numpy.ndarray: Force array.

`compute_particle_electric_field()`

Compute particle electric field using interpolation.

# Returns:
numpy.ndarray: Array containing electric field for each particle.

`compute_potential(`*density*`)`

Update potential based on charge density using relaxation method.

# Args:
density (numpy.ndarray): Charge density array.

# Returns:
None

`compute_time_step()`

Compute time step based on particle velocities.

# Args:

- *iteration (int)*: Iteration index.

# Returns:
None

`init_arrays()`

Initialize arrays for the simulation.

# Returns:
None

`run()`

Run the simulation.

# Returns:
None

`save_global_results()`

Save global results to output files.

# Remarks:

- **The results are saved to the following files:**
  - 'positions.csv': Particle positions.
  - 'velocities.csv': Particle velocities.
  - 'electric_field.csv': Electric field.
  - 'potential.csv': Potential.
- These datas are saved every 'iteration_save_rate' iterations.

# Returns:
None

`save_unit_results()`

Save unit results to output files.

# Remarks:

- **The results are saved to the following files:**
  - 'followed_particle.csv': Followed particle position and velocity.
  - 'followed_cell.csv': Followed cell potential and electric field.
- These datas are saved every iteration.

# Returns:
None

`set_initial_conditions()`

Set initial conditions for the simulation.

# Returns:
None

`update_positions_velocities`(*force*)

Update particle positions and velocities using Euler method.

# Args:

- *force (numpy.ndarray)*: Force array.

# Returns:
None

`write_output_files_headers`()

Write headers to output files.

# Returns:
None

1.4 Visualizer module

class `Visualizer.Visualizer`(*iteration_save_rate*, *position_filename='positions.csv'*, *velocity_filename='velocities.csv'*)

Bases: `object`

Class for visualizing the results of a particle-in-cell simulation.

# Attributes:

- *directory_name (str)*: Name of the directory where the simulation results are saved.
- *position_filepath (str)*: Path to the file containing the particle positions.
- *velocity_filepath (str)*: Path to the file containing the particle velocities.
- *electric_field_filepath (str)*: Path to the file containing the electric field values.
- *potential_filepath (str)*: Path to the file containing the potential values.
- *followed_particle_filepath (str)*: Path to the file containing the followed particle data.
- *followed_cell_filepath (str)*: Path to the file containing the followed cell data.
- *time (numpy.ndarray)*: Array containing the time values.
- *iteration_save_rate (int)*: Number of iterations between each saved step.

# Methods:

- *plot_phase_space(iteration_number=-1)*: Plot the phase space of the particles at a specific iteration.
- *plot_phase_space_foreach_saved_step()*: Plot the phase space of the particles for each saved step.
- *plot_electric_field(iteration_number=-1)*: Plot the electric field at a specific iteration.
- *plot_electric_field_foreach_saved_step()*: Plot the electric field for each saved step.
- *plot_potential(iteration_number=-1)*: Plot the potential at a specific iteration.
- *plot_potential_foreach_saved_step()*: Plot the potential for each saved step.
- *plot_single_particle_position()*: Plot the position of a followed particle over time.
- *plot_single_particle_phase_space()*: Plot the phase space of a single followed particle.
- *plot_potential_for_single_cell()*: Plot the potential for a single cell over time.
- *plot_electric_field_fot_single_cell()*: Plot the electric field for a single cell over time.

`plot_electric_field`(*iteration_number=-1*)

Plot the electric field at a specific iteration.

# Args:

-*iteration_number* (int, optional): The iteration number at which the electric field
should be plotted. Defaults to -1.

# Remarks:
If *iteration_number* is -1, the electric field for the last iteration is plotted.

`plot_electric_field_for_single_cell`()

Plot the electric field for a single cell over time.

`plot_electric_field_foreach_saved_step`()

Plot the electric field for each saved step according to the *iteration_save_rate*.

`plot_phase_space`(*iteration_number=-1*)

Plot the phase space of the particles at a specific iteration.

# Args:

- *iteration_number* (int, optional): The iteration number at which the phase space should be plotted. Defaults to -1.

# Remarks:
If *iteration_number* is -1, the phase space for the last iteration is plotted.

`plot_phase_space_foreach_saved_step`()

Plot the phase space of the particles for each saved step according to the *iteration_save_rate*.

`plot_potential`(*iteration_number=-1*)

Plot the potential at a specific iteration.

# Args:

- ***iteration_number* (int, optional): The iteration number at which the potential**
  should be plotted. Defaults to -1.

`plot_potential_for_single_cell`()

Plot the potential for a single cell over time.

`plot_potential_foreach_saved_step`()

Plot the potential for each saved step according to the *iteration_save_rate*.

`plot_single_particle_phase_space`()

Plot the phase space of a single followed particle.

`plot_single_particle_position`()

Plot the position of a followed particle over time.

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX