API Reference

This document describes the complete API for the SM_2D simulation package.

Core Simulation Classes

Simulation

The primary class for running simulations.

Usage Example

```python
from sm_2d import Simulation

sim = Simulation(
    beam_type="proton",
    energy=150.0,  # MeV
    grid_size=(100, 100),
    spacing=(0.1, 0.1)  # cm
)

results = sim.run()
```

Constructor Parameters

Parameter	Description	Default
`beam_type`	Particle type (“proton”, “carbon”, etc.)	`"proton"`
`energy`	Initial beam energy in MeV	`150.0`
`grid_size`	Tuple (nx, ny) for grid dimensions	`(100, 100)`
`spacing`	Tuple (dx, dy) in cm	`(0.1, 0.1)`
`mcs_model`	Multiple Coulomb scattering model	`"null"`
`em_model`	Energy loss model	`"bethe"`

Methods

Method	Description
`run()`	Execute the full simulation and return results
`reset()`	Clear all simulation state
`get_dose()`	Return 2D dose distribution array
`save_state(path)`	Save current simulation state to file
`load_state(path)`	Load simulation state from file

BeamConfiguration
Defines beam parameters and initial conditions.

Attributes

Attribute	Description	Type
`particle_type`	PDG particle code	`int`
`energy`	Initial kinetic energy (MeV)	`float`
`position`	Initial (x, y) coordinates (cm)	`tuple`
`direction`	Initial direction vector	`numpy.array`
`sigma_x`	Initial beam width in x (cm)	`float`
`sigma_y`	Initial beam width in y (cm)	`float`

DetectorConfiguration
Defines detector geometry and scoring.

Attributes

Attribute	Description	Type
`nx`, `ny`	Number of voxels in each direction	`int`
`dx`, `dy`	Voxel size in each direction (cm)	`float`
`score_dose`	Enable dose scoring	`bool`
`score_fluence`	Enable particle fluence scoring	`bool`
`score LET`	Enable linear energy transfer scoring	`bool`

Physics Models

Energy Loss Models

Bethe-Bloch Model

The Bethe-Bloch equation describes energy loss for charged particles:

$$\frac{\mathrm{dE}}{\mathrm{dx}} = K z^2 \left(\frac{Z}{A}\right) \left(\frac{1}{\beta^2}\right) \left[\ln\left(2 m_e c^2 \beta^2 \frac{\gamma^2}{I}\right) - \beta^2\right]$$

Where:
- $K = 4\pi N_A r_e^2 m_e c^2$
- z is the particle charge
- Z, A are the atomic number and weight of the medium
- I is the mean excitation potential

Usage:

```
sim = Simulation(em_model="bethe")
```

Bohr Straggling Model

Energy loss fluctuations modeled using Bohr's Gaussian approximation:

$$\Omega_B^2 = 4\pi e^4 z^2 N_e x$$

Where:

- N_e is the electron density of the medium
- x is the path length

Usage:

```
sim = Simulation(em_model="bethe_bohr")
```

Multiple Coulomb Scattering

Available MCS models:

Model	Description
`null`	No scattering (straight-line transport)
`hansen`	Hansen et al. analytic model
`fermi_eyges`	Fermi-Eyges theory with Molière theory
`li`	Li's analytical model

Usage:

```
sim = Simulation(mcs_model="hansen")
```

Lateral Spread Theory

The Fermi-Eyges theory computes lateral spread using:

$$\Sigma^2(x) = \int_0^x \mathrm{d}x' T^2(x')(x - x')^2$$

Where $T(x')$ is the scattering power.

Analysis Tools

DoseAnalysis

Provides dose distribution analysis and statistics.

Usage Example

```
from sm_2d.analysis import DoseAnalysis

analysis = DoseAnalysis(dose_array)
stats = analysis.compute_statistics()
```

Methods

Method	Description
`compute_statistics()`	Return dose statistics (max, mean, D95, etc.)
`find_cax_index()`	Find central axis position
`get_profile(axis)`	Extract dose profile along axis
`compute_gamma(dose_ref)`	Perform gamma analysis
`plot_depth_dose()`	Plot depth dose curve
`plot_lateral_profiles()`	Plot lateral dose profiles

Visualization

Plotting and visualization utilities.

Usage Example

```
from sm_2d.vis import plot_dose_heatmap

plot_dose_heatmap(dose, spacing=(0.1, 0.1))
```

Functions

Function	Description
`plot_dose_heatmap()`	Create 2D dose distribution heatmap
`plot_depth_dose()`	Plot dose vs depth curve
`plot_lateral_profiles()`	Plot lateral dose profiles at multiple depths
`compare_doses()`	Overlay multiple dose distributions
`plot_gamma()`	Visualize gamma analysis results

Batch Simulation

BatchSimulation

Run multiple simulations with parameter sweeps.

Usage Example

```
from sm_2d import BatchSimulation

batch = BatchSimulation(
    base_config={
        "beam_type": "proton",
        "grid_size": (100, 100)
    },
    parameter_sweep={
        "energy": [100, 120, 140, 160, 180],
        "mcs_model": ["null", "hansen"]
    }
)

results = batch.run()
```

Utility Functions

File I/O

Function	Description
`load_dose(path)`	Load dose distribution from file
`save_dose(dose, path)`	Save dose distribution to file
`export_results(results, path)`	Export simulation results to JSON
`import_results(path)`	Import results from JSON file

Conversion Utilities

Function	Description
`mev_to_joules(E)`	Convert MeV to joules
`joules_to_mev(E)`	Convert joules to MeV
`gy_to_mev_per_g(dose)`	Convert Gy to MeV/g
`mev_per_g_to_gy(dose)`	Convert MeV/g to Gy

Error Handling

Exception Hierarchy

Exception	Description
`SM2DError`	Base exception for all SM_2D errors
`ConfigurationError`	Invalid simulation configuration
`PhysicsModelError`	Physics model computation error
`FileFormatError`	File I/O or parsing error

Exception	Description
`ConvergenceError`	Simulation failed to converge

Error Handling Example

Best Practice

```python
from sm_2d import Simulation, SM2DError

try:
    sim = Simulation(energy=150.0)
    results = sim.run()
except SM2DError as e:
    logger.error(f"Simulation failed: {e}")
    # Handle error appropriately
```

Performance Optimization

GPU Acceleration

The simulation supports GPU acceleration via CuPy for compatible operations.

Enabling GPU

```python
import sm_2d
sm_2d.set_backend("gpu")  # Use CuPy backend

sim = Simulation(energy=150.0)
results = sim.run()
```

Parallel Processing

Batch simulations automatically use multiprocessing for parallel execution.

```python
batch = BatchSimulation(
    base_config={...},
    parameter_sweep={...},
    n_jobs=4  # Use 4 CPU cores
)
```

Configuration Files

YAML Configuration

Simulations can be configured via YAML files:

```yaml
# config.yaml
beam:
  type: proton
  energy: 150.0

grid:
  size: [100, 100]
  spacing: [0.1, 0.1]

physics:
  mcs_model: hansen
  em_model: bethe_bohr

detector:
  score_dose: true
  score_fluence: true
```

Load configuration:

```python
from sm_2d import load_config

config = load_config("config.yaml")
sim = Simulation(**config)
```

Bibliography

1. B. Gottschalk, **On the scattering power of radiotherapy protons**, Med. Phys. 37 (2010)
2. P. Andreo, **On the clinical spatial resolution achievable with protons and heavier charged particle radiotherapy**, Phys. Med. Biol. 54 (2009)
3. D. J. Brenner, **The impact of the scattering power on the lateral spread of proton beams**, Med. Phys. 37 (2010)
4. J. B. Hansen, et al., **A simple model for the lateral scattering of protons in water**, Phys. Med. Biol. 57 (2012)
5. I. Kawrakow, **Accurate condensed history Monte Carlo simulation of electron transport**, Med. Phys. 27 (2000)