OpenEdge Documentation

1 Overview

OpenEdge is an extension of the SPARTA code that introduces additional functionalities for modeling plasma interactions and magnetic fields. This document provides an overview of the new features, commands, and parameters added to OpenEdge.

2 Commands and Parameters

2.1 `read_plasma_state`

The `read_plasma_state` command sets the plasma background and magnetic fields data. It supports two types of plasma state: `constant` and `file`. The `constant` option sets uniform values across the simulation domain, while the `file` option reads spatially varying data from an input file.

2.1.1 Syntax

```
read_plasma_state plasma_state <type> [options]
```

2.1.2 Parameters

- `<type>`: Specifies the type of plasma state. It can be either `constant` or `file`.
- `[options]`: Additional parameters depending on the chosen type.

2.1.3 Example

```
read_plasma_state plasma_state constant efield 0 0 0 bfield 1 0 0 flow 0 0 0 dens_i 1e19 den
```

This command sets a constant plasma state with the following properties:

- **Electric Field (`efield`)**: (0, 0, 0)
- **Magnetic Field (`bfield`)**: (1, 0, 0)
- **Flow Velocity (`flow`)**: (0, 0, 0)

- **Ion Density** (`dens_i`): 1×10^{19} m^{-3}
- **Electron Density** (`dens_e`): 1×10^{19} m^{-3}
- **Electron Temperature** (`temp_e`): 11600 K
- **Ion Temperature** (`temp_i`): 11600 K

2.2 `global lorentz_force`

The `global lorentz_force` command enables or disables the Lorentz force calculation in the simulation. When enabled, the Lorentz force is calculated for each particle using the electric and magnetic fields specified in the plasma state.

2.2.1 Syntax

```
global lorentz_force <value>
```

2.2.2 Parameters

- `<value>`: Specifies whether to enable (`yes`) or disable (`no`) the Lorentz force calculation.

2.2.3 Example

```
global lorentz_force yes
```

This command enables the Lorentz force calculation in the simulation.

2.3 `global plasma_background_material`

The `global plasma_background_material` command sets the species, mass, and charge of the plasma background material.

2.3.1 Syntax

```
global plasma_background_material <species> mass <mass> charge <charge>
```

2.3.2 Parameters

- `<species>`: Specifies the species of the plasma background material (e.g., `D` for Deuterium, `O` for Oxygen).
- `<mass>`: The mass of the species (in atomic mass units).
- `<charge>`: The charge of the species (in elementary charge units).

2.3.3 Example

```
global plasma_background_material D mass 2.0 charge 1.0
```

This command sets the plasma background material to Deuterium with a mass of 2.0 atomic mass units and a charge of +1 elementary charge unit.

3 Code Implementation

3.1 Setting the Plasma State

The plasma state is initialized using the `read_plasma_state` command. The implementation involves reading the specified parameters and storing them in appropriate data structures. For a constant plasma state, the values are uniform across all grid cells.

3.2 Enabling the Lorentz Force

The Lorentz force is enabled using the `global lorentz_force` command. The implementation involves checking the flag during particle motion calculations and applying the Lorentz force if the flag is set.

3.3 Particle Motion with Lorentz Force

When the Lorentz force is enabled, the particle motion is calculated using the Boris algorithm, which is efficient and accurate for integrating the equations of motion in the presence of electromagnetic fields.

4 Example Usage

```
# Set a constant plasma state
read_plasma_state plasma_state constant efield 0 0 0 bfield 1 0 0 flow 0 0 0 dens_i 1e19 der

# Enable Lorentz force calculation
global lorentz_force yes

# Set plasma background material
global plasma_background_material D mass 2.0 charge 1.0
```

In this example, the simulation is configured with a constant plasma state, the Lorentz force calculation is enabled, and the plasma background material is set to Deuterium with a mass of 2.0 atomic mass units and a charge of +1 elementary charge unit.

5 Plasma Data File Structure

OpenEdge supports loading and interpolating plasma background data for use in plasma simulations. Plasma data can be loaded from an HDF5 file structured with 1D and 2D datasets to represent parameters such as density, temperature, and gradients for electrons and ions. The HDF5 file should follow the naming convention presented below.

5.1 HDF5 File Structure

Each plasma data HDF5 file contains plasma background data organized as follows:

```
struct PlasmaData {
    std::vector<std::vector<double>> dens_e;       // Electron density grid (200, 200)
    std::vector<std::vector<double>> dens_i;       // Ion density grid (200, 200)
    std::vector<std::vector<double>> grad_temp_e;  // Electron temperature gradient grid (20
    std::vector<std::vector<double>> grad_temp_i;  // Ion temperature gradient grid (200, 20
    std::vector<std::vector<double>> parr_flow;    // Parallel flow grid (200, 200)
    std::vector<double> r;                         // Radial positions (200,)
    std::vector<std::vector<double>> temp_e;       // Electron temperature grid (200, 200)
    std::vector<std::vector<double>> temp_i;       // Ion temperature grid (200, 200)
    std::vector<double> z;                         // Axial positions (200,)
};
```

This structure allows for efficient interpolation over the plasma parameters in both radial (r) and axial (z) directions, making it suitable for simulation needs. The expected datasets within each file are organized by the following names:

- **dens_e**: 2D array representing electron density over the (r, z) grid.
- **dens_i**: 2D array representing ion density over the (r, z) grid.
- **grad_temp_e**: 2D array containing the gradient of electron temperature over the (r, z) grid.
- **grad_temp_i**: 2D array containing the gradient of ion temperature over the (r, z) grid.
- **parr_flow**: 2D array containing the parallel flow values over the (r, z) grid.
- **r**: 1D array representing radial positions.
- **temp_e**: 2D array containing electron temperature over the (r, z) grid.
- **temp_i**: 2D array containing ion temperature over the (r, z) grid.
- **z**: 1D array representing axial positions.

5.2 Example Usage

To load a plasma data file in OpenEdge, use the following command:

```
read_plasma_data plasma_data file plasma_data_openedge.h5
```

For cases where a specific plasma parameter needs to be constant, the command can specify the constant values as follows:

```
read_plasma_data plasma_data constant parameter {value}
```

In this case, the specified parameter will be assumed to be uniform and constant across the domain with the given value.

6 Magnetic Field Data File Structure

OpenEdge supports loading and interpolating magnetic field data for use in plasma simulations. Magnetic field data can be loaded from an HDF5 file structured with 1D and 2D datasets to represent radial (r), axial (z), and field components B_r, B_z, and B_t. The HDF5 file should follow a naming convention presented below.

6.1 HDF5 File Structure

Each magnetic field data HDF5 file contains field data organized as follows:

```
struct MagneticFieldData {
    std::vector<double> r;                    // Radial positions
    std::vector<double> z;                    // Axial positions
    std::vector<std::vector<double>> br;      // Radial magnetic field component (B_r) grid
    std::vector<std::vector<double>> bz;      // Axial magnetic field component (B_z) grid
    std::vector<std::vector<double>> bt;      // Tangential magnetic field component (B_t) g
};
```

This structure allows for efficient interpolation over the magnetic field in both radial (r) and axial (z) directions, making it suitable for both 1D and 2D representations. The expected datasets within each file are organized by the following names:

- **r**: 1D array representing radial positions.
- **z**: 1D array representing axial positions.
- **br**: 2D array containing values of the radial magnetic field component B_r over the (r, z) grid.
- **bz**: 2D array containing values of the axial magnetic field component B_z over the (r, z) grid.
- **bt**: 2D array containing values of the tangential magnetic field component B_t over the (r, z) grid.

6.2 Example Usage

To load a magnetic field data file in OpenEdge, use the following command:

```
read_magnetic_fields magnetic_fields file bfield_openedge.h5
```

For cases where the magnetic field is constant, the command can specify the constant field values for each component B_r, B_t, and B_z as follows:

```
read_magnetic_fields magnetic_fields constant bfield {B_r} {B_t} {B_z}
```

In this case, the magnetic field will be assumed to be uniform and constant across the domain with the specified vector values.

7 Surface Data File Structure

OpenEdge supports loading and interpolating surface data from Ftridyn or other Binary Collision Approximation (BCA) codes. The expected file format is an HDF5 file, named following the convention: incident_species_atomic_number_on_target_species_atomic_number.h
A separate file is required for each incident species. For example, in the case of an oxygen plasma interacting with a tungsten wall, the required files would be 8_on_74.h5 (for oxygen on tungsten) and 74_on_74.h5 (for tungsten on tungsten).

Each HDF5 file contains surface interaction data in the following structure:

```
 struct SurfaceData
{
std::vector<double> E;
 Energies std::vector<double> A;
 Angles std::vector<std::vector<double>> rpyld;
 Reflection yield std::vector<std::vector<double>> spyld;
 Sputtering yield
 };
```

Description of Fields:

- **E**: A 1D array of incident particle energies (in eV).
- **A**: A 1D array of incident angles (in degrees), measured relative to the surface normal.
- **rpyld**: A 2D array of reflection yields, where each element corresponds to the reflection coefficient for a given combination of energy (E) and angle (A).
- **spyld**: A 2D array of sputtering yields, where each element corresponds to the sputtering coefficient for the same energy and angle combination. The dimensions of the **rpyld** and **spyld** arrays are (E.size(), A.size()), meaning that for each energy value in E, there is a corresponding array of sputtering and reflection coefficients for each angle in A.

Example: For an oxygen species (atomic number 8) impinging on a tungsten target (atomic number 74), the file 8_on_74.h5 will contain:

- A 1D array of energies (E), e.g., [10, 20, 30, ...] eV.
- A 1D array of angles (A), e.g., [0, 30, 60, 90] degrees.
- Corresponding 2D arrays of reflection yields (rpyld) and sputtering yields (spyld) for every combination of energy and angle.