POSCAR Strain Application Script Documentation

Rajesh Prashanth A
rajeshprasanth@rediffmail.com

Version 1.0.0, Thu Apr 4, 2024

1 Overview

This Python script applies controlled strain to a crystal lattice represented in a VASP POSCAR file. Supported strain types include uniaxial, biaxial, shear, and hydrostatic. It is useful for first-principles simulations of strained materials.

2 Installation Requirements

- Python 3.8 or higher
- ASE (`pip install ase`)
- NumPy (`pip install numpy`)

3 Strain Types Supported

Type	Direction(s)	Description
Uniaxial	x, y, z	Stretch/compress along one axis
Shear	xy, yz, xz	Distort lattice shape without uniform scaling
Biaxial	xy-biaxial, yz-biaxial, xz-biaxial	Expand/compress two axes simultaneously
Hydrostatic	xyz	Uniform expansion/compression along all axes

4 Mathematical Formalism

4.1 Strain Tensor

Strain is represented by a second-order tensor:

$$\boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{pmatrix}$$

Uniaxial strain: Only the diagonal element along the strained axis is non-zero. Example, x-direction:

$$\epsilon_{xx} = \text{strain\_value}, \quad \epsilon_{yy} = 0, \quad \epsilon_{zz} = 0$$

Biaxial strain: Two diagonal elements non-zero. Example, xy-plane:

$$\epsilon_{xx} = \epsilon_{yy} = \text{strain\_value}, \quad \epsilon_{zz} = 0$$

Shear strain: Off-diagonal elements non-zero. Example, xy-plane:

$$\epsilon_{xy} = \epsilon_{yx} = \text{strain\_value}$$

Hydrostatic strain: All diagonal elements equal:

$$\epsilon_{xx} = \epsilon_{yy} = \epsilon_{zz} = \text{strain\_value}$$

4.2 Deformation Matrix

The deformation gradient matrix $\mathbf{F}$ is:

$$\mathbf{F} = \mathbf{I} + \boldsymbol{\epsilon}$$

where $\mathbf{I}$ is the 3×3 identity matrix.

4.3 Applying Strain to the Lattice

The new lattice vectors $\mathbf{a}_i'$ are computed as:

$$\mathbf{a}_i' = \mathbf{F} \cdot \mathbf{a}_i$$

where $\mathbf{a}_i$ are the original lattice vectors.

Atomic positions are scaled automatically:

$$\mathbf{r}_j' = \mathbf{F} \cdot \mathbf{r}_j$$

where $\mathbf{r}_j$ are original atomic positions and $\mathbf{r}_j'$ are strained positions.

5 Script Functions

- **apply_strain(atoms, strain_direction, strain_percentage)**: Applies strain tensor to ASE Atoms object.
- **read_poscar(poscar_file)**: Reads POSCAR file into ASE Atoms object.
- **write_poscar(atoms, output_file)**: Writes ASE Atoms to POSCAR.
- **main()**: Command-line interface for batch usage.

6 Usage Examples

```
# Uniaxial strain along x-axis by 2%
python apply_strain.py -i POSCAR -d x -s 2.0 -o POSCAR_strained

# Biaxial strain in xy-plane by -1.5%
python apply_strain.py -i POSCAR -d xy-biaxial -s -1.5 -o POSCAR_biaxial

# Hydrostatic strain (uniform expansion) by 0.5%
python apply_strain.py -i POSCAR -d xyz -s 0.5 -o POSCAR_hydro
```

7 Notes and Best Practices

- ASE uses double precision for atomic positions.
- For fixed-cell relaxation in VASP, set `ISIF=2`.
- Positive strain: expansion; negative strain: compression.
- Invalid strain directions raise `ValueError`.

8 Optional Improvements

- Loop over multiple strain percentages.
- Logging and automated file naming.
- Integration with high-throughput DFT pipelines.

9 References

- ASE Documentation: `https://wiki.fysik.dtu.dk/ase/`
- VASP POSCAR Format: `https://www.vasp.at/`