rdfpy

Release 1.0.0

Feb 17, 2023

Contents:

1 Installation **3**

2 Introduction and Examples **5**
- 2.1 What are Radial Distribution Functions? 5
- 2.2 Example: RDF of a Crystal Structure 5

3 Licence **7**

rdfpy

rdfpy is a Python library for fast computation of 2D and 3D radial distribution functions.

CHAPTER 1

Installation

rdfpy can be installed using `pip`:

```
pip install rdfpy
```

CHAPTER 2

Introduction and Examples

2.1 What are Radial Distribution Functions?

The radial distribution function (RDF) (or pair correlation function) characterises the structure of a system of particles. If we select an arbitrary particle as the origin, the RDF describes the number of particles we would observe relative to the bulk density of the system, as a function of distance. This is calculated and averaged over every particle in the structure being considered. The formal definition of the RDF is

$$g_i(r) = \frac{n_i(r)}{4\pi r^2 \delta r \rho}$$

where $n_i(r)$ is the number of particles between distances r and δr, and $\rho = \frac{N}{V}$ is the number density. Dividing by ρ ensures that the RDF is centred around 1 when the density of particles observed at some distance does not deviate from the bulk density.

2.2 Example: RDF of a Crystal Structure

In this example, we use **rdfpy** to compute the RDF of crystalline Ti. The Crystallographic Information File (CIF) can be obtained from here, and is provided by the Materials Project. First we import the `rdf` function from **rdfpy**, as well as pymatgen for obtaining atom coordinates from the CIF file, and numpy.

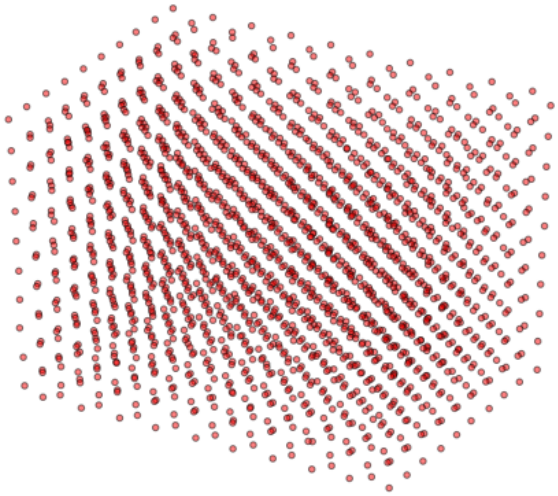
We then load the structure and create a supercell so that there are enough atoms in the crystal to compute an RDF from. Also, to make the resulting function smoother, we add some noise to the coordinates.

```
import numpy as np
from rdfpy import rdf
from pymatgen import Structure

structure = Structure.from_file('/path/to/Ti_mp-46_computed.cif')
structure.make_supercell(10)

coords = structure.cart_coords
```

(continues on next page)

(continued from previous page)

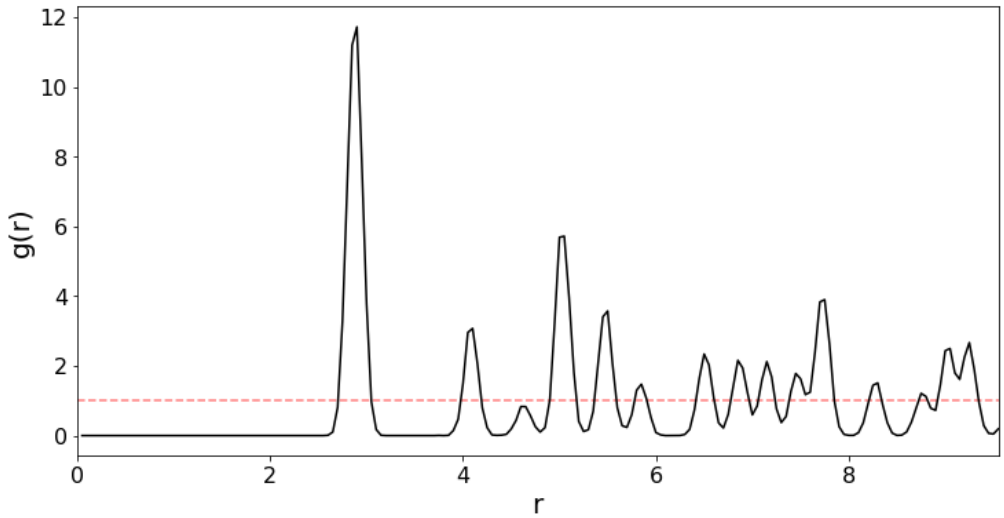
```
noise = np.random.normal(loc=0.0, scale=0.05, size=(coords.shape))
coords = coords + noise
```



Finally we use **rdfpy** to compute the RDF from the coordinates, specifying the step size parameter `dr`:

```
g_r, radii = rdf(coords, dr=0.05)
```

Plotting `g_r` against `radii` results in the following function.



CHAPTER 3

Licence