ascii-phonons Documentation

Release 1.0.0

Adam J. Jackson

April 03, 2016

Contents

Contents:

CHAPTER 1

Introduction

ascii-phonons is a package to produce attractive images and animations of phonon modes in crystals.

Visualisation is a powerful tool for the study of vibrations in the solid state. "Semi-automatic" animations have been generated for scientific publications, where they provide insight to spectroscopic observations. [1], [2] In order to make this type of imagery more accessible, and add some visual interest to ajjackson's PhD thesis, a more convenient and automatic toolchain has been developed.

Images are rendered using the open-source 3D animation package Blender. As Blender has a notoriously steep learning curve, a *Command-line interface* is provided which may be used to generate images without interacting with the Blender interface. It is intended that a simple *Graphical user interface* will also be made available.

[1] http://dx.doi.org/10.1063/1.4917044 (see animation)
[2] http://dx.doi.org/10.1103/PhysRevB.92.144308 (see animation)

CHAPTER 2

Installation

Will be outlined in more detail. For now, see the README file.

2.1 Paths

Ascii-phonons relies on the script files **scripts/ascii-phonons** and **scripts/ascii-phonons-gui** finding core functionality in the module init file **ascii_phonons/__init__.py**. In previous versions of ascii-phonons, this required the top-level folder (i.e. the folder produced by *git clone* or by unzipping a downloaded file) to be included in the user's PYTHONPATH. In the most recent versions, this is not necessary; as long as the folder structure is left intact, the scripts should be able to find what they need.

2.2 Blender

A recent version of Blender is required; development is currently based on Blender 2.76 and later. At least version 2.70 is needed, which provides the wireframe modifier used to draw the bounding box.

2.2.1 Linux

Note that the versions of Blender available in package managers such as apt-get are often quite dated. Installing the latest version for Linux is easy, however; just download the .tar.gz file, untar it and add the directory to your PATH.:

```
mv /some/blender/download.tar.bz2 /some/directory && cd /some/directory
tar -xf /my-blender-download.tar.bz2
echo "export PATH="${PWD}/my-blender-folder:${PATH}" >> ${HOME}/.bashrc
source ${HOME}/.bashrc
```

2.3 Documentation dependencies

- Sphinx
- Mock

Command-line interface

A command-line utility, **ascii-phonons** is provided in the *scripts* folder. This program generates a temporary Python 3 script, making use of the `vsim2blender` module, and calls Blender. The only mandatory argument is a `.ascii` file containing the crystal structure and phonon mode data.

```
./scripts/ascii-phonons [OPTARGS] my_crystal.ascii [OPTARGS]
```

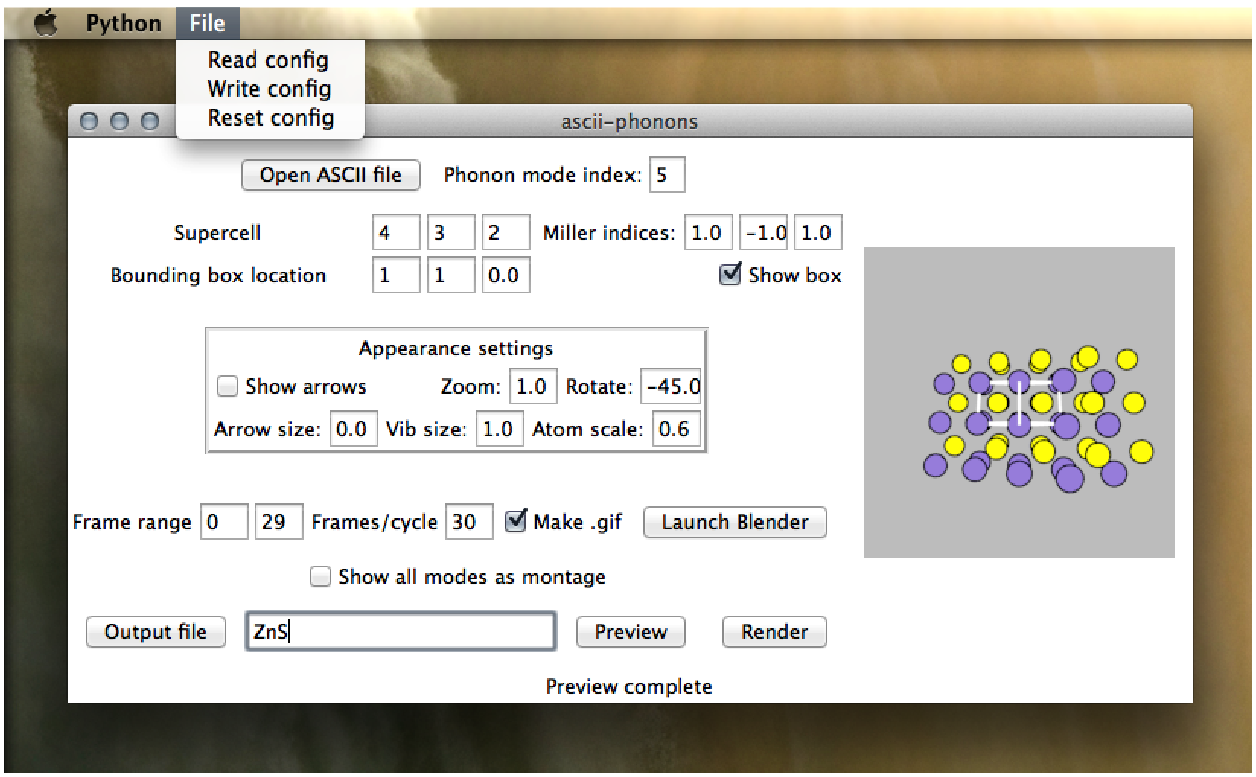
The detailed output is controlled with optional arguments, as outlined below. The list of accepted arguments may also be viewed by calling with the "help" argument `-h`

```
./scripts/ascii-phonons -h
```

Syntax	Description
`-B PATH, --blender_bin PATH`	The script will make an educated guess if this is not provided.
`-m I, --mode_index`	The index of the phonon mode to use (counting from 0).
`-d X Y Z, --supercell_dimensions X Y Z`	Make a supercell; three integers X Y Z specify multiples of lattice vectors
`-s, --static`	Output a single static image
`-f, --n_frames`	Number of frames in animation (default 30)
`-o PATH/NAME, --output_file PATH/NAME`	Filename for output. Format specifier (.png, .gif) is appended automatically. If -o is specified, the Blender GUI is not launched unless -g is also specified.
`-g, --gui`	Open full Blender GUI session, even if rendering output.
`--gif`	Create a .gif file using Imagemagick convert. This flag is ignored if no output file is specified.
`-v, --vectors`	Show eigenvectors with static arrows.
`--scale_factor X.Y`	Floating-point scale factor for atom size. 1.0 = covalent radius. It is recommended to reduce this value when visualising with arrows, in order to prevent arrows from being hidden inside atoms.
`--vib_magnitude X.Y`	Floating-point scale factor applied to displacements. The default value of 3 was selected for Cu2ZnSnS4; this may need to be adjusted for different systems.
`--arrow_magnitude X.Y`	Floating-point scale factor applied to arrows created with the -v flag.
`--normalise_vectors`	Rescale arrows such that the maximum for each mode is a fixed length.
`--no_box`	Hide the unit cell bounding box.
`--box_position X Y Z`	Adjust the bounding pox position in the supercell with floating-point multiples X Y Z of the lattice vectors.
`--miller X Y Z`	Miller indices determining camera direction. Negative and fractional values are permitted.
`--montage`	Use Imagemagick to create a tiled array output of all modes. The *-m* flag is ignored. An animated .gif will be output unless `--static` is specified.
`--montage_args`	Additional args for montage command e.g. `--montage_args='-tile {cols}x{rows}'` (Note the use of = and single quotes)
`--camera_rot ROT`	Rotated camera position in degrees
`--zoom X.Y`	Floating-point zoom adjustment. A sensible starting point is calculated automatically and corresponds to the value 1.0, but this factor can be used for further adjustment.
`--config PATH/FILE.conf`	Path to user configuration file.
`--do_mass_weighting`	Apply mass weighting to atom movements. This has usually already been done in the construction of the .ascii file, and should not be repeated.
`--orthographic`	Use orthographic projection (i.e. no perspective effect)

Graphical user interface

A simple graphical user interface (GUI) is available, including a "preview" window. At this time a useful subset of features is implemented. User configuration files can be saved and loaded using the File menu. As shown below, on Mac OSX this is located at the top of the screen; on other Unix-like systems and Windows the menu is part of the floating GUI window. When a config file is loaded with "Read config", it is combined with the existing configuration so, for example, the user can a new colour scheme over an existing set of colour parameters. If this behaviour is not desired, use "Reset config" before "Read config".



To launch the GUI, which is compatible with Python 3 and Python 2.7, simply run **scripts/ascii-phonons-gui**.

4.1 Dependencies

- Tkinter is used to draw the GUI. This is included in standard Python distributions.

- The "PIL" module is loaded to handle the preview rendering. This dependency is best satisfied by installing "Pillow". On Linux, the tk imaging part of this is often packaged separately, with names like *python-imaging-tk*.

CHAPTER 5

Python Interface

The generation of temporary files and calls to Blender are handled by a python module *ascii_phonons*.

ascii_phonons.**call_blender**(***options*)

Generate a temporary script file and call Blender

Typically Blender is called in batch mode to render one or a series of .png image files.

ascii_phonons.**montage_anim**(***options*)

Render animations for all phonon modes and present as array

ascii_phonons.**montage_static**(***options*)

Render images for all phonon modes and present as array

ascii_phonons.**parse_tuple**(*tuple_string*, *value_type=<type 'float'>*)

Get a tuple back from string representation

Three representations are recognised: '[1,2,3]' : JSON-style '1 2 3' : Simple space-separated '1,2,3': Simple comma-separated

Parameters

- **tuple_string** (*str*) – Serialised tuple
- **value_type** (*type*) – Type to cast values to

CHAPTER 6

Blender Interface

The interface with Blender is managed as a Python add-on module `vsim2blender`. See the module index for the documentation of these modules. The *Command-line interface* works by generating a temporary script file and executing the script with Blender. Advanced Blender users may prefer to directly import the `vsim2blender` module and use it with Blender's scripting tools. The key plotting tools are all in `vsim2blender.plotter`, with supporting functions in the other modules.

6.1 Top-level functions

6.2 List of modules

6.2.1 Arrows

Arrow graphics are used to indicate the phonon eigenvectors. The arrow is a blender file `arrow_cylinder.blend` of unit length lying along the x axis. This file can be modified if a different arrow shape is desired.

6.2.2 Ascii file importer

Functions relating to the import of v_sim ascii files

class `ascii_importer.`**`Mode`**

Collection of vibrational mode data imported from a v_sim ascii file

Parameters

- **`freq`** (*`float`*) – Vibrational frequency
- **`qpt`** (*`3-list of reciprocal space coordinates`*) – **q**-point of mode
- **`vectors`** (*`Nested list; 3-lists of complex numbers corresponding to atoms`*) – Eigenvectors

`ascii_importer.`**`cell_vsim_to_vectors`**(*cell_vsim*)

Convert between v_sim 6-value lattice vector format (ref) and set of three Cartesian vectors

Parameters **`cell_vsim`** (*`2x3 nested lists`*) – Lattice vectors in v_sim format

Returns Cartesian lattice vectors

Return type 3-list of 3-Vectors

`ascii_importer.import_vsim(filename)`

Import data from v_sim ascii file, including lattice vectors, atomic positions and phonon modes

Parameters **filename** – Path to .ascii file

Returns cell_vsim, positions, symbols, vibs

Return cell_vsim Lattice vectors in v_sim format

Return type 2x3 nested lists of floats

Return positions Atomic positions

Return type list of 3-Vectors

Return symbols Symbols corresponding to atomic positions

Return type list of strings

Return vibs Vibrations

Return type list of "Mode" namedtuples

6.2.3 Plotter

Commands for adding atoms to the scene and animating them.

Mathematics

The key equation is: [1]

$$\mathbf{u}(jl,t) = \sum_{\mathbf{q},\nu} \mathbf{U}(j,\mathbf{q},\nu) \exp(i[\mathbf{q}\mathbf{r}(jl) - \omega(\mathbf{q},\nu)t])$$

Where ν is the mode identity, ω is frequency, $\mathbf{U}$ is the displacement vector, and $\mathbf{u}$ is the displacement of atom j in unit cell l. We can break this down to a per-mode displacement and so the up-to-date position of atom j in cell l in a given mode visualisation

$$\mathbf{r}'(jl,t,\nu) = \mathbf{r}(jl) + \mathbf{U}(j,\mathbf{q},\nu) \exp(i[\mathbf{q}\mathbf{r}(jl) - \omega(\mathbf{k},\nu)t])$$

Our unit of time should be such that a full cycle elapses over the desired number of frames.

A full cycle usually lasts $2\pi/\omega$, so let $t = \frac{2\pi f}{\omega N}$; $-\omega t$ becomes $-\omega \frac{2\pi f}{\omega N} = 2\pi f/N$ where f is the frame number.

$$\mathbf{r}'(jl,t,\nu) = \mathbf{r}(jl) + \mathbf{U}(j,\mathbf{q},\nu) \exp(i[\mathbf{q}\mathbf{r}(jl) - 2\pi f/N])$$

The arrows for static images are defined as the vectors from the initial (average) positions to one quarter of the vibrational period (i.e. max displacement)

Module contents

6.2.4 Camera

Camera placement is an interesting problem. The current method `vsim2blender.camera.setup_camera()` looks along the y axis and estimates a sensible distance based on the lattice parameters, but is occasionally thrown off.

The successor in development allows the camera position to be specified by giving the Miller indices of a plane to view.

[1]

13. (a) Dove, Introduction to Lattice Dynamics (1993) Eqn 6.18

Configuration files

Plain-text configuration files are used to provide supporting data and allow per-user tweaking. In addition to the provided files **settings.conf** and **elements.conf**, which are found in the `addons/vsim2blender` folder of the project, the user can maintain their own configuration file in this format. When this file is provided via the `--config` flag of the *Command-line interface* or using the `vsim2blender.read_config()` function of the Python library, user settings will take precedent over the defaults. Configuration files can also be loaded in the GUI; this allows multiple config files to be "layered", and the parameters discovered through a GUI session may be exported for re-use by the GUI, CLI or Python interface.

The format of these files is a typical plain text .ini-style format and is implemented with configparser. Conventionally the file extension is .conf, but this is not enforced. Parameters are grouped into *sections* with a header in square brackets; the parameters themselves are separated from their values with = or `:` markers.

```
[header]

Like = this
Or: this

# And comments are indicated with a '#'
; or a ';'
```

Section headers are *case-sensitive*, and are all lower-case. Options are not case-sensitive.

7.1 Settings

settings.conf, which lies inside the **vsim2blender** package, contains default settings that are not related to specific elements.

```
[general]
box_thickness = 5
outline_thickness = 3

[colours]
background = 0.5 0.5 0.5
box = 1. 1. 1.
outline = 0. 0. 0.
```

7.2 Elements

Data for each element is included in the `elements.conf` configuration file. Relative atomic masses are drawn from standard reference data. [1] Where this reference gives a range and/or the relative abundance of isotopes is unknown, a simple mean was taken.

```
[masses]
C = 12.0106    # The mass of carbon is a floating-point number in a.m.u.
```

Atomic radii are drawn from a recent study encompassing elements with atomic numbers up to 96. [2]

```
[radii]
Ac = 2.15   # The covalent radius of Ac is a floating-point number in angstroms
```

Colours are assigned somewhat arbitrarily for a handful of elements which have been used in WMD Group publications. Suggestions are welcome for a more mainstream pallette. The values are RGB tuples, with values ranging from 0 to 1.

```
[colours]
Cu = 0.8 0.3 0.1
```

7.3 User configuration

An example user configuration file, with an alternative colour scheme, is included in the main project directory as **example.conf**. Note that the colour information for elements and for other parts of the image may be mixed freely.

[1] http://www.nist.gov/pml/data/comp J. S. Coursey, D. J. Schwab, J. J. Tsai, and R. A. Dragoset, NIST Physical Measurement Laboratory
[2] http://dx.doi.org/10.1039/B801115J B. Cordero *et al.* (2008) *Dalton Trans.* **2008** (21) 2832-2838

CHAPTER 8

Development

Development is in progress and hosted on Github. Please use the issue tracker for feature requests, bug reports and more general questions.

The target for input files is the ASCII format used by v_sim, a useful program and currently one of the only tools available for visualising phonons. These files contain all the information needed to define a crystal sructure and its vibrations. It is presumed that these are generated by Phonopy from *ab initio* electronic structure calculations, and initially this code will only target the features used by Phonopy. Extension to the full ASCII format is of course welcome.

Animation and rendering is done in Blender. The preferred approach to scripting Blender is to write an "addon" which carries out importing duties. However, the target user for ascii-phonons is not familiar with Blender's interface and should not need to learn it. A traditional Blender addon, accessed throught the Blender GUI, is therefore inappropriate. In theory Blender can be built as a Python library but this appears to be quite difficult and would create a high barrier to use. Instead, ascii-phonons works by creating an addon library for Blender, creating a temporary script file which uses this library and calling Blender as a subprocess.

The initial target platforms are modern GNU/Linux distributions and Mac OS X. Operation under Windows is not actively being tested, but is desirable.

CHAPTER 9

License

This software is made available under the GNU Public License, version 3. The license is available online, and a copy should always be included with the code.

CHAPTER 10

Acknowledgements

Work on this package began while ajjackon was a PhD student funded by EPSRC through the Center for Sustainable Chemical Technologies (grant no. EP/G03768X/1) at the University of Bath. Further work to fix bugs and improve the documentation and useability has taken place as a Research Assistant the same research group, while funded by the ERC (project 277757).

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

A

C

I

M

P