contactpp Documentation

Release 1.1

Pascal Bugnion

August 18, 2014

Contents

CHAPTER 1

Installation

1.1 Requirements

python2.7, numpy, scipy. These can be installed on a Debian system using:

```
$ sudo apt-get install python-numpy python-scipy
```

1.2 The easy way

The easiest way to download and install `contactpp` is from the Python package index. Run:

```
$ easy_install contactpp
```

This requires root access (unless you are running in a virtual environment). To install without root access, run:

```
$ easy_install --user contactpp
```

1.3 From github

Clone the git repository using:

```
$ git clone https://github.com/pbugnion/contactpp.git
```

Navigate to the source's root directory (`contactpp`) and run:

```
$ python setup.py install
```

1.4 From source

If you have a *.zip or *.tar.gz archive with the source, unpack the archive in a directory, navigate into this directory and run:

```
$ python setup.py install
```

CHAPTER 2

Introduction

contactpp generates pseudopotentials for interparticle interactions which accurately reproduce the contact potential. Potentials can be generated to reproduce the bound state, or scattering states with either positive or negative scattering lengths. This program currently only offers norm-conserving pseudopotentials in the form detailed by Troullier and Martins.

The formalism is detailed in [BLNC].

CHAPTER 3

Command line usage

Current command line usage is geared towards producing output suitable for the Casino Quantum Monte Carlo program. It can readily be extended to output pseudopotentials in a different format.

The command structure is:

```
$ gen_pseudo <type> [<args> ... ]
```

where <type> is one of `troullier`, `utp` or `swell`. A description of the arguments for each pseudopotential can be obtained using:

```
$ gen_pseudo help <type>
```

The command will produce a `manual_interactions` block suitable for inclusion in the Casino input file. For instance:

```
$ gen_pseudo troullier repulsive 1.0 1.0

%block manual_interaction
polynomial
order : 13
cutoff : 1.4690133838
c_0 : -6.68710073203
c_1 : 0.0
c_2 : 37.5550272665
c_3 : 0.0
c_4 : -76.8391296021
c_5 : 11.4986279905
c_6 : 86.9227576451
c_7 : -14.7318519405
c_8 : -74.7316666352
c_9 : 28.0837647928
c_10 : 29.2577382704
c_11 : -21.989806382
c_12 : 4.13182642699
%endblock manual_interaction
```

We look at the arguments for each type of potential in more detail.

3.1 Troullier-Martins pseudopotentials

To generate a pseudopotential following the Troullier-Martins form of pseudopotentials, use one of the following commands:

```
$ gen_pseudo troullier [--cutoff=<cutoff>] repulsive <a> <E>
$ gen_pseudo troullier attractive <a> <E> <cutoff>
$ gen_pseudo troullier bound <a> <cutoff>
```

We look at the arguments for each branch in detail.

3.1.1 Repulsive branch

```
$ gen_pseudo troullier repulsive <a> <E>
```

`<a>` is the scattering length, `<E>` is the calibration energy. The cutoff is calculated by default as the value of the first antinode of the true wavefunction at incident energy `E`. This can be altered by passing the command line argument `--cutoff=XX`:

```
$ gen_pseudo --cutoff=XX repulsive <a> <E>
```

3.1.2 Attractive branch

```
$ gen_pseudo troullier attractive <a> <E> <cutoff>
```

`<a>` is the scattering length, `<E>` is the Fermi energy and `<cutoff>` is the cutoff, which must be smaller than the first node of the wavefunction.

3.1.3 Bound state

```
$ gen_pseudo troullier bound <a> <cutoff>
```

`<a>` is the scattering length and `<cutoff>` is the cutoff, which must be smaller than the first node of the wavefunction.

3.2 UTP pseudopotentials

To generate a UTP, use one of:

```
$ gen_pseudo utp [options] [--cutoff=<cutoff>] repulsive <a> <E>
$ gen_pseudo utp [options] attractive <a> <E> <cutoff>
```

The arguments are identical to those for the relevant branch of the Troullier-Martins potential. The optimizer will print convergence information as it runs. This can be suppressed by passing `--quiet` or `-q` as an option.

3.3 Square well pseudopotential

To generate a square well or top hat pseudopotential, use:

```
$ gen_pseudo swell [--radius=<radius>] [--] <a>
```

where `<a>` is the scattering length, and `--radius` is the potential radius.

For repulsive potentials, if the radius is unspecified, the code will create a pseudopotential with zero effective range.

For attractive potential, the radius must be specified. To indicate a negative scattering length, add a double dash `--` before the scattering length. For instance, to create a square well with scattering length -0.5, use:

```
$ gen_pseudo swell --radius=0.3 -- -0.5
```

CHAPTER 4

Python API

We expose a python API to make it easy to customise the output of `contactpp` or include it as part of other scripts.

Importing `contactpp` exposes three functions:

- `contactpp.make_troullier_potential`
- `contactpp.make_utp_potential`
- `contactpp.make_square_well_potential`

See the API description below for details on the arguments to each of these functions.

Each function returns a pseudopotential object. As an example of how to use this object, we generate a Troullier-Martins pseudopotential and plot it using matplotlib.

```
import contactpp
import matplotlib.pyplot as plt
import numpy as np

pseudopotential = contactpp.make_troullier_potential(
        "repulsive",scattering_length=1.0,calibration_energy=1.0)

print pseudopotential.cutoff
# 1.8637284049928018

print pseudopotential.coefficients
# numpy array of coefficients.

rs = np.linspace(0.,2*pseudopotential.cutoff)
plt.plot(rs, pseudopotential.V(rs))
plt.show()
```

4.1 Generating the pseudopotentials

`contactpp.`**`make_troullier_potential`**(*branch*, *scattering_length*, *calibration_energy=None*, *cutoff=None*)

Create a TroullierPotential object.

Parameters **branch: {"repulsive", "scattering", "bound"}** :

scattering_length: float :

Float indicating the scattering length. Must be positive if branch is "repulsive" or "bound", and negative otherwise.

calibration_energy: float :

Energy at which to calibrate the Troullier potential. The Fermi energy is normally a good choice. Must be larger than 0 if branch is "repulsive" or "attractive". This parameter is ignored if branch is "bound". In that case, the calibration energy is taken to be the true bound state energy -1/2a^2.

cutoff: float :

Choice of cutoff radius. Compulsory if branch is "bound" or "attractive". If branch == "repulsive", this is chosen as the first antinode of the true wavefunction at the calibration energy by default, but can be overriden.

Returns **TroullierBoundPotential or TroullierScatteringPotential** :

contactpp.**make_utp_potential**(*scattering_length*, *fermi_energy*, *cutoff=None*, *init_coeffs=None*, *integrator='vode'*, *npoints=1000*, *nks=10*, *objective_function='rms'*, *dos=None*, *verbose=True*)

Create a UTPPotential object.

Parameters **scattering_length** : float

Scattering length. If positive, the pseudopotential is built for the repulsive branch. If negative, it is built for the attractive branch. Note that UTP potentials do not exist for the bound state.

fermi_energy : float

The UTP constructor will minimize the errors in phase shifts over the range `k = 0` to `kf`, where `kf = sqrt(Ef)` is the Fermi wavevector.

cutoff : float

Choice of cutoff radius. Compulsory for negative scattering lengths. If the branch is "repulsive", the cutoff is chosen as the first antinode of the true wavefunction at `3/5 * fermi_energy`. Specifying the cutoff explicitly will override this.

init_coeffs : iterable, optional

Initial coefficients for the optimizer. If not specified, a Troullier pseudopotential is constructed and those coefficients are used. If specified, the coefficients will be interpreted as: `[ v1, v2, v4, v5 ... v11 ]`, with the potential defined as, `V(x)/Ef = (1-x^2) * (v1*(0.5+x) + v2*x^2 + (2v2-v1)*x^3 + v4*x^4 + sum_{i=5}^13 v_i x^i` where `x = r/cutoff`.

integrator : string, optional

Which integrator to use when computing the phase shift. See the documentation for `scipy.integrate.ode` for details. If the cutoff chosen is very small, it might be worthwhile changing this to `"dopri5"`. `"vode"` by default.

npoints : int, optional

Number of points to use in the integration. If the cutoff chosen is small, convergence of the results with respect to `npoints` should be checked. 1000 by default.

nks : int, optional

Number of k-points to use when integrating the error in phase shifts. Increasing this may lead to somewhat better results. 10 by default.

objective_function : string, optional

Type of objective function to minimize: either "rms" to minimize `int_0^kF { [delta_PP(k) - delta_true(k)]^2 dos(k) dk`, where dos is another optional argument, or "max" to minimize `max(|delta_PP(k) - delta_true(k)|)`. "rms" by default.

dos : function, dos(k) -> density of states, optional

When using the RMS objective function, this argument sets the cost function that is optimized when constructing the potential. The objective function is: `int_0^kF { [delta_PP(k) - delta_true(k)]^2 dos(k) dk`. By default, `dos(k) = lambda k: k**2`. Must accept a numpy array (of k-values) as input. If the objective_function is not RMS, specifying this argument raises a ValueError.

verbose : boolean, optional

Display convergence information. True by default.

`contactpp.`**`make_square_well_potential`**(*scattering_length*, *radius*)

Create a SquareWellPotential object. Can be either a top-hat potential (for positive scattering lengths) or a well (for negative scattering lengths.

Parameters **scattering_length** : float

Float indicating the scattering length.

radius : float

Radius of the potential.

Returns **SquareWellPotential** :

4.2 Pseudopotential objects

class `contactpp.troullier.`**`TroullierScatteringPotential`**(*ppsi_polynomial*, *cutoff*, *calibration_energy*, *scattering_length*)

Object representing a scattering potential.

Represents a scattering pseudopotential built using the Troullier-Martins formalism.

Attributes

cutoff: float	
calibration_energy: float	
scattering_length: float	
coefficients: numpy array	array of coefficients for the pseudopotential, V(r) = sum(coefficients[i]*r^i) for r < cutoff.
psi: function	psi(r) returns the value of the pseudowavefunction at the calibration energy at r.
true_psi: function	true_psi(r) returns the value of the contact wavefunction at the calibration at r.
V: function	V(r) returns the value of the pseudopotential at r.

class `contactpp.troullier.`**`TroullierBoundPotential`**(*ppsi_polynomial*, *cutoff*, *calibration_energy*, *scattering_length*)

Object representing a scattering potential.

Represents a scattering pseudopotential built using the Troullier-Martins formalism.

Attributes

cutoff: float	
calibration_energy: float	
scattering_length: float	
coefficients: numpy array	array of coefficients for the pseudopotential, V(r) = sum(coefficients[i]*r^i) for r < cutoff.
psi: function	psi(r) returns the value of the pseudowavefunction at the calibration energy at r.
true_psi: function	true_psi(r) returns the value of the contact wavefunction at the calibration at r.
V: function	V(r) returns the value of the pseudopotential at r.

class `contactpp.utp.`**`UTPPotential`**(*potential_polynomial*, *cutoff*, *scattering_length*)
Object representing a scattering potential.

class `contactpp.swell.`**`SquareWellPotential`**(*height*, *radius*)
Object representing a Square well potential.

Attributes

height	float	Potential height (negative for a well).
radius	float	Potential radius

CHAPTER 5

Citing

If you use contactpp to generate a pseudopotential, please cite as [BLNC].

CHAPTER 6

Release notes

6.1 1.1

When making UTP potentials, the Python API allows the user to choose an objective function that minimizes the maximum error in phase shift, rather than minimizing the RMS error.

6.2 1.0.3

Initial public tested release. *contactpp* can generate:

- Square well and top hat pseudopotentials,
- Troullier-Martins pseudopotentials,
- UTP pseudopotentials.

This program provides a command line script that produces input suitable for inclusion in a Casino input file, and a Python API.

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*

[BLNC] P. O. Bugnion, P. López Ríos, R. J. Needs and G. J. Conduit, High-fidelity pseudopotentials for the contact interaction.

[BLNC] P.O. Bugnion, P. López Ríos, R.J. Needs, and G.J. Conduit, High-fidelity pseudopotentials for the contact interaction.

C

C