## NAME

rpbqff - push-button quartic force fields

## SYNOPSIS

**rpbqff** [OPTIONS] [INFILE]

## DESCRIPTION

**rpbqff** runs quartic force fields at the push of a button. It currently handles three types of coordinates: symmetry-internal coordinates (SICs), specified in the format expected by **intder**(1); Cartesian coordinates, and normal coordinates. The latter two are determined automatically from the input Cartesian geometry. The normal coordinates are determined by first running a harmonic force field in Cartesian coordinates, and then the full QFF is evaluated at displacements along the resulting normal coordinates.

## OPTIONS

This section lists the command line options supported by **rpbqff**. In addition to these options,**r pbqff** expects an input file. If omitted, the name of this input file is taken to be *pbqff.toml*. See the INPUT section for details about its contents.

**−c**, **−−checkpoint**
: Resume from the checkpoint files in the current directory (*chk.json* and *res.chk*).

**−n**, **−−no-del**
: Don't delete any files when running the single-point energies.

**−o**, **−−overwrite**
: Overwrite a previous run.

**−v**, **−−version**
: Print the git version information and exit.

**−t**, **−−threads** *THREADS*
: Set the maximum number of threads to use. Defaults to 0, which means to use as many threads as there are CPUs.

**−h**, **−−help**
: Print help information and exit.

## INPUT

This section describes the contents of the input file. Unless otherwise noted, every option is required and has no default value. See the EXAMPLES section for some example inputs with sensible values for these options. The input format is TOML, Tom's Obvious Minimal Language. For help constructing your input file, see **qffbuddy**(1) which should have been included with **rpbqff**.

**geometry** *String*
: The initial geometry to use for the computation. Both XYZ and Z-matrix geometries are accepted.

**optimize** *bool*
: Whether or not to perform a geometry optimization on the input

**charge** *isize*
: The molecular charge. This value can be spliced into the template using the {{.charge}} directive.

**step_size** *f64*
: The size of the displacement to take in the QFF coordinates.

**sleep_int** *usize*
: The interval in seconds to wait between loops checking if any jobs have finished.

**job_limit** *usize*
The maximum number of jobs to submit at once, as determined by the number of individual input files. This distinction is important when chunk_size is greater than 1 because the maximum number of jobs submitted to the queue will be job_limit / chunk_size .

**chunk_size** *usize*
The number of individual calculations to bundle into a single queue submission.

**coord_type** *CoordType*
The type of coordinate to use in the QFF. Currently-supported values are "sic", "cart", and "normal". Note that SIC QFFs require an additional input file called intder.in to define the internal coordinates.

**template** *TemplateSrc*
The template input file for the quantum chemistry program. Supported formatting directives depend on the program in question. Molpro supports {{.geom}} for the geometry and {{.charge}} for the molecular charge, while Mopac expects a static template.

**hybrid_template** *Option<TemplateSrc>*
An optional template file for the cubic and quartic portion of the QFF. If this is provided, the regular template is used only for the harmonic portion of the QFF, and this is used for the rest of the points.

**queue_template** *Option<TemplateSrc>*
The template input file for the queuing system. Supported formatting directives include {{.basename}} for the base name of the submit script, which is useful for naming the job in the queue, and {{.filename}} for the name of the quantum chemistry program input file. Not all program and queue combinations expand these, however.

**program** *Program*
The quantum chemistry program to use in running the QFF. Currently-supported values are "cfour", "dftb+", "molpro", "mopac".

**queue** *Queue*
The queuing system to use in running the QFF. Currently-supported values are "local", which uses bash to run computations directly, "pbs", and "slurm".

**findiff** *Option<bool>*
Whether to use finite differences or least-squares fitting for the potential energy surface. Currently normal coordinates are the only coord_type to use this option, so it has a default value of false, meaning use the fitted version of normal coordinates. Setting this option to true forces the use of finite differences for the normal coordinate QFF.

**check_int** *usize*
The interval at which to write checkpoint files. Every coordinate type will write an initial checkpoint (res.chk), but this interval determines whether or not checkpoints are written while the single-point energies are being run. A value of 0 will disable checkpoints entirely. This interval refers to the number of polling iterations that have occurred, not the number of jobs that have completed. The iteration count is shown in the log file when the number of jobs remaining is printed.

**weights** *Option<Vec<f64>>*
An optional vector of atomic masses to use for normal coordinate generation and Spectro.

**dummy_atoms** *Option<usize>*
An optional number of atoms to hold constant in the QFF displacements. These must come at the end of the geometry. Experimental

**#[serde(default)]**
Resume a normal coordinate QFF from the initial HFF phase.

**norm_resume_hff** *bool*

## EXAMPLE

The following is an example input file for c-C3H2 in SICs in Mopac on the Slurm queuing system:

```
geometry = """
C
C 1 CC
C 1 CC 2 CCC
H 2 CH 1 HCC 3 180.0
H 3 CH 1 HCC 2 180.0

CC =        1.42101898
CCC =       55.60133141
CH =        1.07692776
HCC =       147.81488230
"""
optimize = true
charge = 0
step_size = 0.005
coord_type = "sic"
program = "mopac"
queue = "slurm"
sleep_int = 2
job_limit = 2048
chunk_size = 1
template = "scfcrt=1.D-21 aux(precision=14 comp xp xs xw) PM6 THREADS=1"
check_int = 100
```

## PROGRAM-SPECIFIC DETAILS

The following sections contain program-specific details about the config file.

### DFTB+

Thanks to DFTB+'s strange input format, **pbqff** has a relatively harder time automatically removing geometry optimization directives from the input template. As such, it expects a template like the one shown below:

```
Driver = GeometryOptimization {
  Optimizer = Rational {}
  MovedAtoms = 1:-1
  MaxSteps = 100
  OutputPrefix = "geom.out"
  Convergence {
    Energy = 1e-8
    GradElem = 1e-8
    GradNorm = 1e-7
    DispElem = 1e-7
    DispNorm = 1e-7
  }
}
```

In particular, **pbqff** will only recognize the exact pattern **Driver = GeometryOptimization** for identifying optimization input (though it ignores case). For single-point energy calculations, it will strip this out, and for geometry optimizations, it will refrain from adding the default optimization commands.

## SEE ALSO

**qffbuddy**(1)