ESIgen Documentation

Release 0.0.5+0.g470f10a.dirty

Jaime RGP, InsiliChem

Jan 23, 2019

For users

Automatically generate supporting information documents for your Chemistry publications online.

CHAPTER 1

Quick usage

1.1 Online server

Note: This is only a demo server, so performance won't be stellar... All files will be deleted within 1h and we won't collect any data from you. Refer to the *Installation* docs if you want to setup your own (local) server.

1. Visit esi.insilichem.com and upload your Computational Chemistry outputs there. Any of the examples in cclib data should work.
2. Choose one of the *Builtin templates* or create your own (read on templating *Syntax*).
3. Profit! You can now inspect the report and use the interactive viewer to rotate, move and *make measurements* in your molecule(s).
4. If you want to download the results, you have several options in the footer:
   (a) Download all the processed data in a ZIP file which will include all the files listed below.
   (b) Download only the rendered images in the browser.
   (c) Markdown report (plain-text)
   (d) XYZ coordinates
   (e) CML coordinates (Chemical Markup Language, XML-like)
   (f) Raw JSON (Programmatic representation of all the parsed data in your logfiles)
   (g) Chemical JSON (Simplified summary of the JSON dump, with more meaningful names)
5. You can also export to online services that generate citable DOI identifiers for your data.
   (a) Figshare
   (b) Github Gist (must be converted to repository and then sync'ed with Zenodo integration)
   (c) Zenodo

1.2 Command-line (batch processing)

1. Install ESIgen in your computer (read *Installation*).
2. Run `esigen filename.log`. That's it! You can even provide several files at once with `esigen file1.log file2.log` or `esigen my_files_*.log`.
3. If you want to one of the *Builtin templates* or use your own (read on templating *Syntax*), specify it with `esigen -t mytemplate.md filename.log`. Ideal for quick reports on your daily routine. The template `checks.md` has been designed for this specific purpose.

The ESIgen suite also includes several other executables:

- `esigenweb`. Creates a local webserver like the one in esi.insilichem.com, but running only in your computer.
- `esixyz`. Extracts XYZ coordinates from a computational chemistry logfile. Can be used with optimization jobs to request a particular step. Use `esixyz -h` for more help.

Futhermore, since ESIgen uses `cclib` under the hood, all its executables are available as well. Namely, `ccget` and `ccwrite`. Again use `-h` for more help.

CHAPTER 2

Installation

If you need to process a lot of files or are worried about your privacy, we recommend to install ESIgen locally. It also makes for a good day-to-day tool if you have to check a lot of output files routinely!

2.1 Recommended steps

Download the latest release and execute the installer. The binaries `esigen` and `esigenweb` will be available under `$INSTALL_PATH/bin`. In Windows, a shortcut will also be available in Start menu.

If the installer is not available for your platform, use one of the methods below.

2.2 With conda

Conda is package manager for Python that greatly simplifies the installation of Python libraries.

1. Download Miniconda 3 for your platform. 2a. Install `esigen` in a Python 3.6 environment (default for Miniconda 3). 2b. Install `esigen` in a Python 2.7 environment if you don't have PyMol already (Linux and MacOS, only).
2. Run `esigenweb` to launch the web server, or `esigen` for the CLI tool.

For Linux & Mac, this roughly translates to:

```
## Install Conda
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh && bash␣
↪Miniconda*.sh

## Install esigen
conda install -c insilichem esigen

## Or, if you also need PyMol
# conda create -n esigen -c omnia -c egilliesix -c insilichem python=2.7 esigen
```

(continues on next page)

(continued from previous page)

```
# conda activate esigen

## Run!
esigen -h  # CLI
esigenweb  # WEB
```

For Windows, it's almost the same. Just install Miniconda with the *.exe, and use the Anaconda Prompt (in Start Menu) to install Numpy. Then, use `pip` for `esigen`. The commands are:

```
conda install numpy esigen
esigen -h  # CLI
esigenweb  # WEB
```

2.3 With pip

If you don't want to install Conda, or the package is not available in your platform, you will need to provide the additional dependencies yourself. Assuming you already have Python installed, run:

```
pip install numpy
pip install esigen
# or, for latest dev version
pip install https://github.com/insilichem/esigen/archive/master.zip
```

CHAPTER 3

Templating in ESIgen

3.1 Builtin templates

Some templates are included with ESIgen for your convenience:

- `default.md`. Uses an interactive 3D viewer in the web interface and static images on the command line. A table of magnitudes is provided, together with the coordinates and, if available, first 10 frequencies.
- `TD.md`. Same as default, but listing the excitation energies of TD calculations.
- `simple.md`. Dummy example to help illustrate easy templating.
- `chemshell.md`. Shows experimental support for some ChemShell QM/MM jobs.
- `checks.md`. Day-to-day analysis tasks for Gaussian jobs.

However, you might want to modify them or create your own from scratch. Keep reading for further details.

3.2 Syntax

ESIgen templates are written using **Markdown** (a subset of HTML designed to be highly readable) syntax on top of the wonderful **Jinja2** engine, which allows easy keyword replacing. For example:

```
# {{ name }}

- Stoichiometry: {{ stoichiometry }}
- Electronic Energy (eV): {{ electronic_energy }}
```

, which results in something like this:

3.2.1 heme_fe2singlet_fe3doublet

- Stoichiometry: C34H30FeN4O4(1-,2)
- Electronic Energy (eV): -1957.86428957

As you see, you just write normal Markdown (syntax is described here) and then insert the desired variables between double curly braces, like this `{{ energy }}`. Jinja2 also supports `for` loops, `if` conditionals and so on, but you won't probably need it. However, one of the advanced features might be helpful: filters, which allow you to post-process some values (ie, centering with respect to a fixed width with `{{ value|center(20) }}`).

3.3 Data Fields

All fields provided by cclib parsers are available. Simply refer to the official documentation on Parsed Data and Parsed Data Notes to check the full list and the compatibility matrix across different programs.

Additionally, ESIgen provides some more fields and methods you can use during the templating:

- Features
  - `{{ name }}`: Extracted from the filename, without the extension.
  - `{{ filename }}`: Filename, with extension.
  - `{{ filepath }}`: Full path to the file.
  - `{{ stoichiometry }}`. Self-explanatory.
  - `{{ imaginary_freqs }}`: Number of negative frequencies.
  - `{{ mean_of_electrons }}`: Mean of `alphaelectrons` and `betaelectrons`.
  - `{{ metadata['route'] }}`: First of Gaussian route sections. For other programs, check `metadata`.
  - `{{ nsteps }}`: Number of optimization steps. Extracted from `scfenergies` size.
  - `{{ solvent }}`. Solvent chosen, if applicable. (Gaussian only).
  - ModRedundant scans (Gaussian only, might change anytime; see `checks.md` for an example):
    - `{{ modredvars }}`: List of variables being scanned (R83, A21, D125...).
    - `{{ modreddefs }}`: Atomic definition of those variables (2 atoms for R, 3 for A, 4 for D)
    - `{{ modredvalues }}`: (m, n) array for variable values (distance in A, angle in degrees), where m is the number of cycles and n the number of variables.
    - `{{ modredenergies }}`: (j, k, l) array of -dE/dx values, where j is the number of cycles, k the number of iteration at each cycle and l the number of variables being scanned.
- Magnitudes (Gaussian only)
  - `{{ electronic_energy }}`. Last of `scfenergies`, Eh.
  - `{{ thermalenergy }}`: Sum of electronic and thermal energies, Eh.
  - `{{ zeropointenergy }}`: Sum of electronic and zero-point energies, Eh.
- Structural info
  - `{{ viewer3d }}`: Insert interactive 3D depiction of the structure. Only available in web UI.

  - `{{ image }}`: Static depiction of the structure. Requires PyMol (not available on public demo!)
  - `{{ cartesians }}`: Molecule structure exported in XYZ format.

- Functions
  - `{{ convertor(value, from_unit, to_unit) }}` can be used to change units in most cases. Check here for supported constants. If not, you can always use normal math inside the curly braces (`{{ (10+value)**2 }}`).

- Experimental support for QM/MM jobs in ChemShell (might change anytime; see `chemshell.md` template for an example)
  - `{{ scfenergies }}`: Lists "QM/MM energy" entries.
  - `{{ mmenergies }}`: List of dicts which detail MM energy decomposition for each cycle.
  - `{{ energycontributions }}`: List of dicts which detail QM/MM energy decomposition for each cycle.
  - `{{ optdone }}`: `True` if optimization converged (only available with Turbomole backend).

Note: Depending on the software used to create the output file, some fields might not be available. When in doubt, you can check the JSON dump of the files by appending `/json` to the report URL (a link is also available in the bottom of the page). This will list all the attributes available for each file. (JSON dumps are best viewed in Firefox 57+).

3.4 Missing Values

An additional flag `missing` is passed to the template to control what to report when the requested field is missing in the file. By default, in the WebUI, maximum length is restricted to 10 chars.

In the `default.md` template, `missing` is checked in every row to control if the row should be written or not. If `missing` is set to the empty string `''` (default value) and the value is not present in the file, the row won't be written; if missing is set to any other string (for example, `N/A`) and the value is not present, the row will be written but the value will be reported as `N/A`.

For example, if `stoichiometry` is not available in the file:

Case A: `missing = ''`

```
# TDNI_GGCMPW

__Requested operations__

`p td=(nstates=30) MPW1PW91 scrf=(solvent=water) geom=connectivity def2tzvp`

__Relevant magnitudes__

| Datum                                          | Value                      |
|:-----------------------------------------------|---------------------------:|
| Charge                                         |                         -2 |
| Multiplicity                                   |                          1 |
| Stoichiometry                                  |              C7H9N3NiO4S(2-) |
| Number of Basis Functions                      |                        570 |
| Electronic Energy (eV)                         |            -2644.5302088499993 |
| Mean of alpha and beta Electrons               |                         75 |
```

Case B: `missing = 'N/A'`, default

```
# TDNI_GGCMPW

__Requested operations__

`p td=(nstates=30) MPW1PW91 scrf=(solvent=water) geom=connectivity def2tzvp`

__Relevant magnitudes__

| Datum                                              | Value                      |
|:---------------------------------------------------|---------------------------:|
| Charge                                             |              -2            |
| Multiplicity                                       |               1            |
| Stoichiometry                                      |      C7H9N3NiO4S(2-)       |
| Number of Basis Functions                          |             570            |
| Electronic Energy (Eh)                             |    -2644.5302088499993     |
| Sum of electronic and zero-point Energies (Eh)     |             N/A            |
| Sum of electronic and thermal Energies (Eh)        |             N/A            |
| Sum of electronic and thermal Enthalpies (Eh)      |             N/A            |
| Sum of electronic and thermal Free Energies (Eh)   |             N/A            |
| Number of Imaginary Frequencies                    |             N/A            |
| Mean of alpha and beta Electrons                   |              75            |
```

CHAPTER 4

Using ESIgen programmatically

ESIgen is only a wrapper around the excellent `cclib` project, which handles the tedious job of parsing computational chemistry logfiles. It then provides all the parsed fields to a `Jinja2` templating engine via the main class, `esigen.core.ESIgenReport`. This class also hooks to several molecular viewers implemented in `esigen.render` (PyMol, NGLView, ChemView). The API is straight-forward:

```
from esigen import ESIgenReport
reporter = ESIgenReport(logfile_path)
all_fields = reporter.data_as_dict()
subyacent_ccdata = reporter.data
# Fill template with parsed data ()
print(reporter.report(template='default.md'))
# If inside Jupyter Notebook, get an interactive preview
reporter.view_with_nglview()
```

Check the docstrings for `ESIgenReport` and `ESIgenReport.report()` for more information on available options.

4.1 How to extend ESIgen

TLDR: Check the source for the `esigen.io` module.

Extending ESIgen to add support for more fields should not be necessary. Instead, contributing to `cclib` is preferred and, that way, other users will benefit from the new additions. That said, we do add some fields in ESIgen for the sake of easy templating. This is controlled in the `esigen.io.ccDataExtended` class, which is a subclass of the original `cclib.parser.data.ccData_optdone_bool` class, responsible of holding the data fields (as listed here). Our subclass creates a few property methods (functions that can be called as attributes, without the `()` syntax) that serve as aliases to compute some properties on the fly based on the existing (static) fields. If you want to add more, subclass `esigen.io.ccDataExtended`, add the properties and overwrite the `_properties` class attribute to reflect the new additions:

```
from esigen.io import ccDataExtended
class ccDataExtendedExtra(ccDataExtended):
    _properties = ccDataExtended._properties[:] + ['my_new_property']

    @property
    def my_new_property(self):
        return 'Custom value computed out of static fields'
```

This is needed because `esigen.io.ccDataExtended` defines a new method `as_dict`, that acts as the original `ccData.getattributes`, but also collecting the values from all the elements defined in `_properties`.

Temporarily, we have also added some more static attributes and subclassed the original `GaussianParser` (based on `cclib.parser.Gaussian`) to fill them in by adding more rules in `esigen.io.GaussianParser.extract`. Eventually, these new fields will be available in `cclib`, but in the meantime you can use these.

All the code in this module (`esigen.io`) can serve as an example on how to extend ESIgen to fill your own needs. You only have to change the default parameters during the creation of your ESIgenReport objects:

```
from esigen import ESIgenReport
# default
report = ESIgenReport(logfile_path)
# custom parser, default datatype
report = ESIgenR
```

CHAPTER 5

Get help

If you have any questions, please feel free to submit an issue in our Github repository.

CHAPTER 6

Citation

ESIgen is scientific software, funded by public research grants: Spanish MINECO (project CTQ2017-87889-P), Generalitat de Catalunya (project 2014SGR989), J.R.G.P.: Generalitat de Catalunya (grant 2017FI_B2_00168), P.G.O.: Spanish MINECO (grant FPI BES-2015-074190). If you make use of ESIgen in scientific publications, please cite our article in JCIM. It will help measure the impact of our research and future funding!

```
@article{doi:10.1021/acs.jcim.7b00714,
    author = {Rodríguez-Guerra Pedregal, Jaime and Gómez-Orellana, Pablo and Maréchal,
↪ Jean-Didier Pierre},
    title = {ESIgen: Electronic Supporting Information Generator for Computational␣
↪Chemistry Publications},
    journal = {Journal of Chemical Information and Modeling},
    year = {2018},
    doi = {10.1021/acs.jcim.7b00714},
        note ={PMID: 29506387},
    URL = {
            https://doi.org/10.1021/acs.jcim.7b00714
    },
    eprint = {
            https://doi.org/10.1021/acs.jcim.7b00714
    }
}
```