RMG Study Group

Session I: Git, Sphinx, webRMG

Connie Gao

9/20/2013

Why the RMG Study Group?

- Devoted to RMG-**Py**
- Help new users learn how to set up RMG
- Existing users and developers can discuss specifics in code
- RMG-Py documentation gets written before each session! (except today's session)

Today's Topics

- Git: the version control tool for RMG
- Sphinx: tool for creating documentation
- webRMG: online tools for working with RMG

Key URLs

- http://www.github.com/GreenGroup
  - Git repository of all RMG-Py code
- http://greengroup.github.io/RMG-Py/
  - Online version of the current RMG-Py documentation
- http://rmg.mit.edu
  - Official RMG-Py documentation, thermodynamics and kinetics database browser, and web tools
- http://dev.rmg.mit.edu
  - Developmental version of rmg.mit.edu with latest features and potential bugs
  - To use, add 18.172.0.124 dev.rmg.mit.edu to hosts file in your operating system

Git

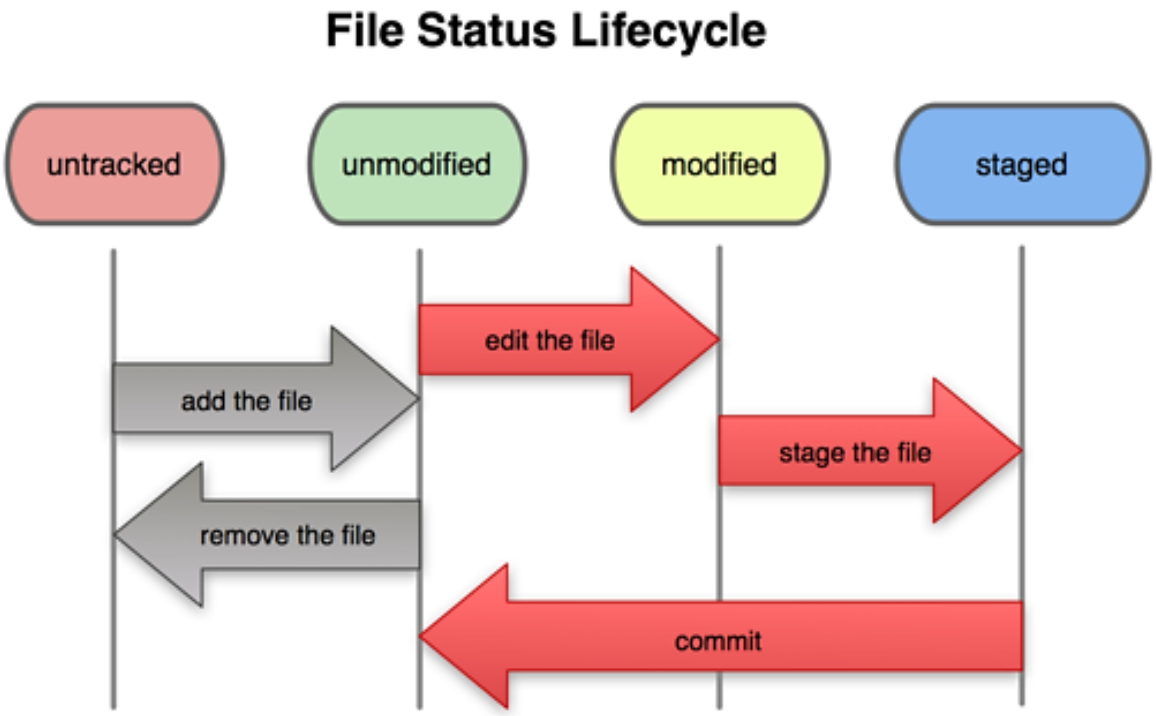
Git

- Git is a version control tool
  - Multiple users can edit multiple copies of code
  - Single user can create multiple branches for a single respository
- Online detailed tutorial:
  - http://git-scm.com/book
- Where to find programs to help you use git:
  - http://git-scm.com/downloads

Basic Git Workflow

1. **Modify** files in your working directory.
2. **Stage** files, adding snapshots of them to your staging area.
3. Make a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

Git file lifecycle



Notes:

If a particular version of a file is in the **git directory**, it's considered **committed**.
If it's modified but has been added to the **staging area**, it is **staged**.
If it was **changed** since it was checked out but has not been staged, it is **modified**.

Getting started: create a local repo

Two common scenarios: (only do one of these)

a) To **clone an already existing repo** to your current directory:

```
$ git clone <url> [local dir name]
```

This will create a directory named *local dir name*, containing a working copy of the files from the repo, and a **`.git`** directory (used to hold the staging area and your actual repo)

b) To **create a Git repo** in your current directory:

```
$ git init
```

This will create a **`.git`** directory in your current directory.

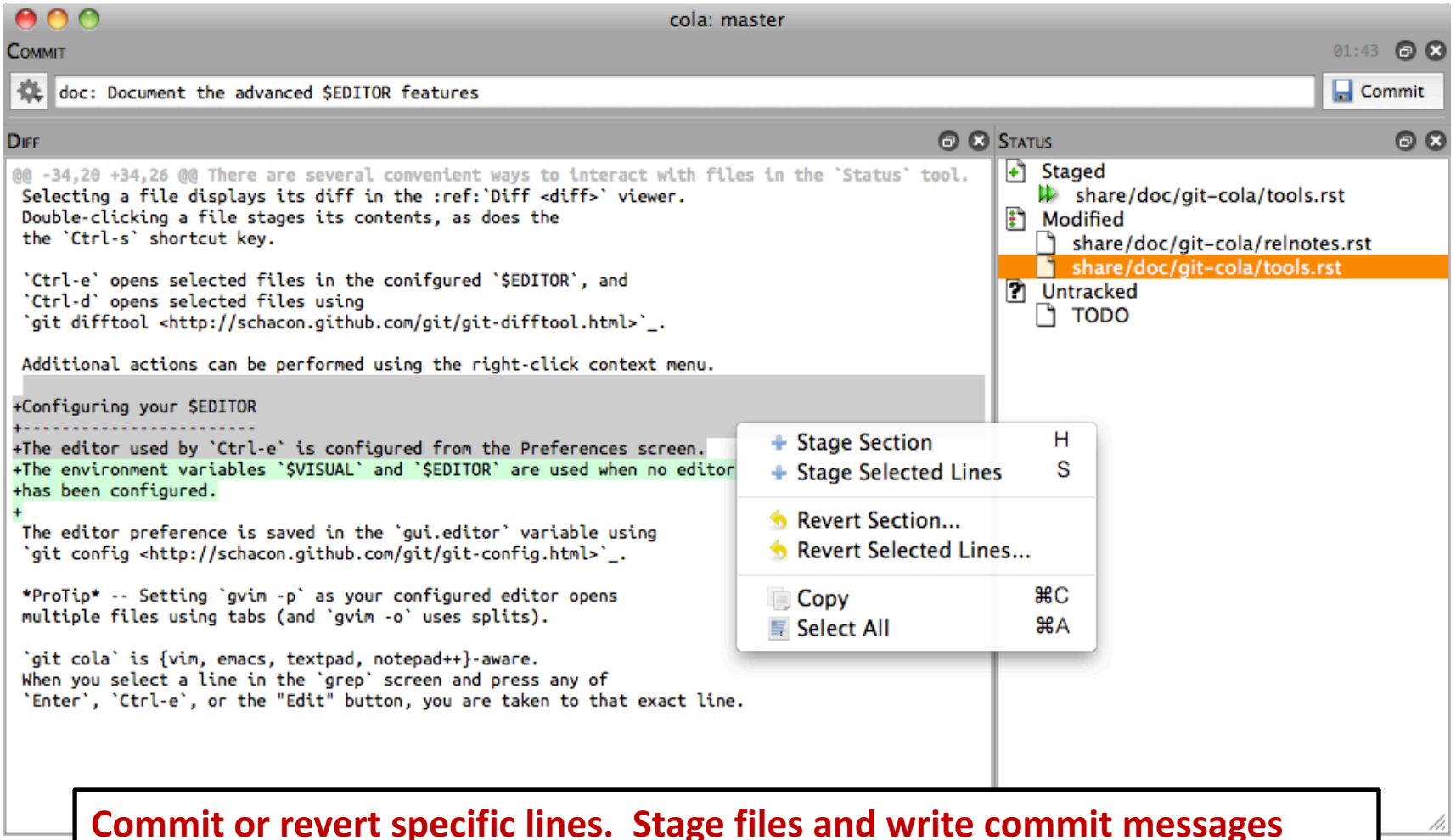
Then you can commit files in that directory into the repo:

```
$ git add file1.java
$ git commit –m “initial project version”
```

Common Git commands

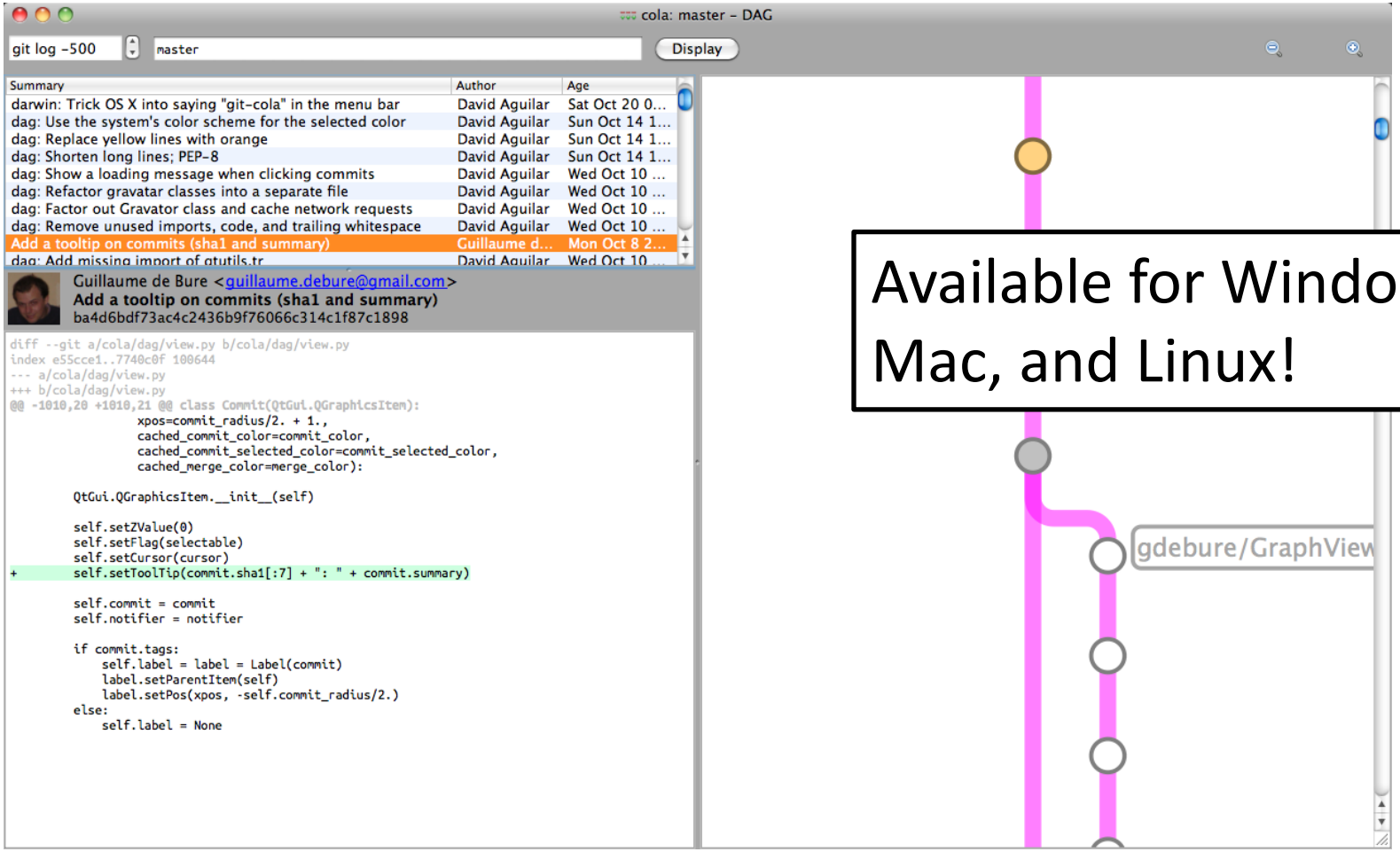
command	description
`git clone` ***`url [dir]`***	copy a git repository so you can add to it
`git add` ***`files`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init`, `reset`, `branch`, `checkout`, `merge`, `log`, `tag`	

You can do all of this using Git-cola: a powerful GUI interface



Commit or revert specific lines. Stage files and write commit messages graphically. Amend commits.

Git-cola: a powerful GUI nterface



Visualize past commit history and repository branches. (Great for tracking specific code changes.)

Steps for getting started with git and RMG-Py

1. **Clone** the RMG-Py git repository onto your local directory (or onto a server such as Pharos which has all the RMG-Py dependencies as well as git preinstalled)
2. After making changes to code, test to ensure nothing is broken
3. Run **git pull official master** to pull in latest official commits
4. **Stage** your modified files and make a **commit** of your changes, with a useful commit message
5. **Push** the commit to your github repository

Note: Post issues and bugs in the code to the RMG-Py issues page on github so others can help you address your problem
https://github.com/GreenGroup/RMG-Py/issues


SPHINX
Python Documentation Generator

Sphinx

- Tool for creating *intelligent* and *beautiful* documentation
- Output formats in both HTML and LaTeX PDF
- Uses reStructuredText as its markup language

RMG-Py documentation

- Located in RMG-Py/documentation folder
- How to build:
  - Run ‘make documentation’ in parent RMG-Py folder will build the HTML pages
  - Or, go to documentation folder and run ‘make html’ to make HTML pages or ‘make latexpdf’ to create the pdf version of the documentation
- Edit the files inside the documentation/source folder

reStructuredText Basics

- Primer for reS:
  - http://sphinx-doc.org/rest.html#rst-primer
- Example of Sphinx markup:

```
This is a Title
===============
That has a paragraph about a main subject and is set when the '='
is at least the same length of the title itself.

Subject Subtitle
----------------
Subtitles are set with '-' and are required to have the same length
of the subtitle itself, just like titles.

Lists can be unnumbered like:

 * Item Foo
 * Item Bar

Or automatically numbered:

 #. Item 1
 #. Item 2

Inline Markup
-------------
Words can have *emphasis in italics* or be **bold** and you can define
code samples with back quotes, like when you talk about a command: ``sudo``
gives you super user powers!
```

Example markup converted to HTML:

This is a Title

That has a paragraph about a main subject and is set when the '=' is at least the same length of the title itself.

Subject Subtitle

Subtitles are set with '-' and are required to have the same length of the subttitle itself, just like titles.

Lists can be unnumbered like:

- Item Foo
- Item Bar

Or automatically numbered:

1. Item 1
2. Item 2

Inline Markup

Words can have *emphasis in italics* or be **bold** and you can define code samples with back quotes, like when you talk about a command: `sudo` gives you super user powers!

Using python function references in documentation

- Function comments can be imported to Sphinx documentation automatically

ORIGINAL CODE
RMG-Py/rmgpy/statmech/vibration.pyx

```
cdef class HarmonicOscillator(Vibration):
    """
    A statistical mechanical model of a set
    harmonic oscillators. The attributes ar

    =============== ============================================================
    Attribute       Description
    =============== ============================================================
    `frequencies`   The vibrational frequencies of the oscillators
    `quantum`       ``True`` to use the quantum mechanical model, ``False`` to use the classical model
    =============== ============================================================

    In the majority of chemical applications, the energy levels of the
    harmonic oscillator are of similar magnitude to :math:`k_\\mathrm{B} T`,
    requiring a quantum mechanical treatment. Fortunately, the harmonic
    oscillator has an analytical quantum mechanical solution.
    """

    def __init__(self, frequencies=None, quantum=True):
        Vibration.__init__(self, quantum)
        self.frequencies = quantity.Frequency(frequencies)

    def __repr__(self):
        """
        Return a string representation that can be used to reconstruct the
        HarmonicOscillator object.
        """
        result = 'HarmonicOscillator(frequencies={0!r}'.format(self.frequencies)
```

DOCUMENTATION SOURCE

RMG-Py/documentation/source/reference/statmech/harmonicoscillator.rst

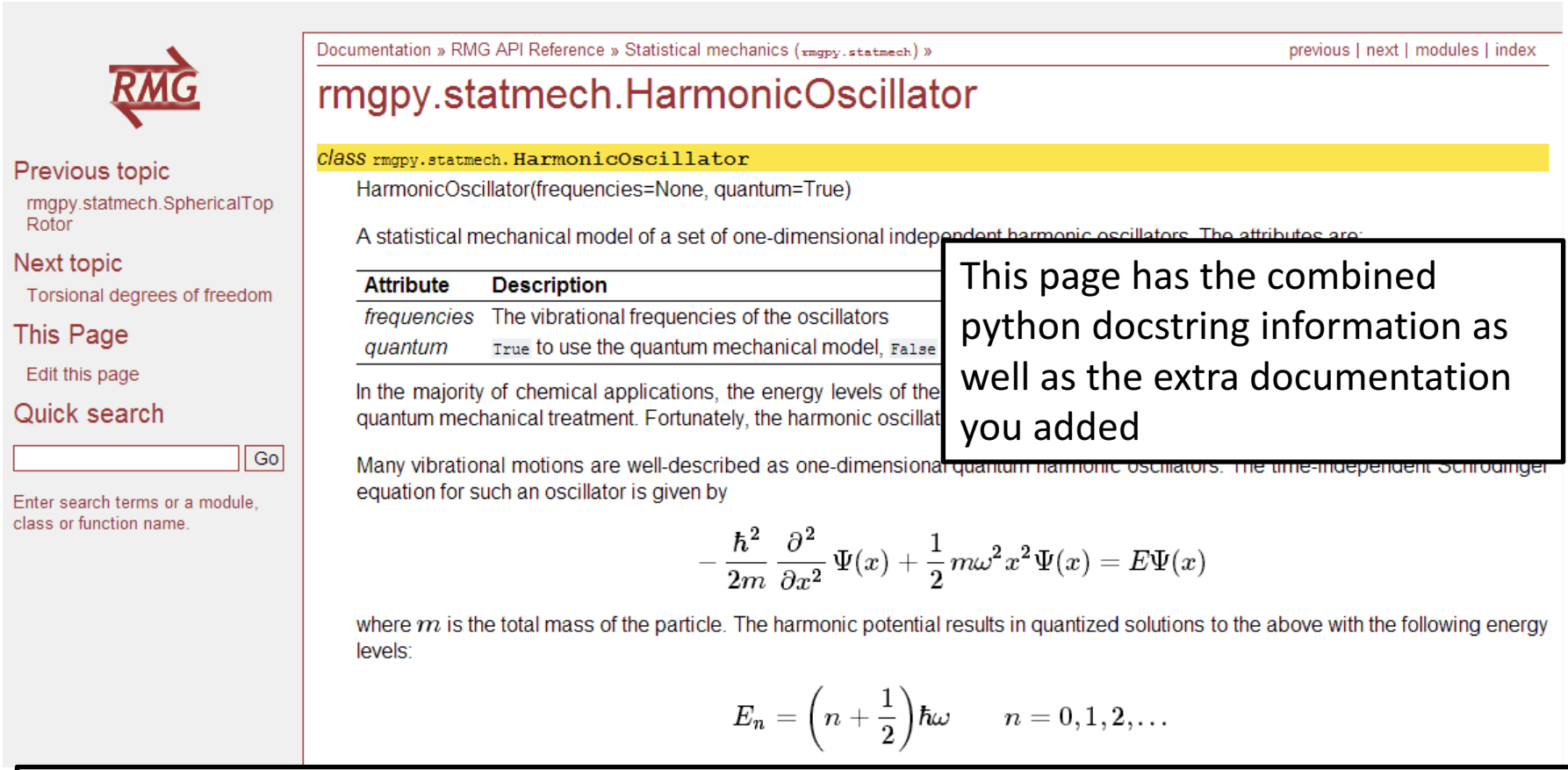
```
1  **********************************
2  rmgpy.statmech.HarmonicOscillator
3  **********************************
4
5  .. autoclass:: rmgpy.statmech.HarmonicOscillator
6
7      Many vibrational motions are well-described as one
8      oscillators. The time-independent Schrodinger equa
9      is given by
10
11     .. math:: -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi(x) + \frac{1}{2} m \omega^2 x^2
       \Psi(x) = E \Psi(x)
12
13     where :math:`m` is the total mass of the particle. The harmonic potential
14     results in quantized solutions to the above with the following energy levels:
15
16     .. math
17
18     Above w
19     the gro
20     not zero; this energy is called the *zero-point energy*.
21
22     The harmonic oscillator partition function is obtained by summing over
23     the above energy levels:
24
25     .. math:: Q_\mathrm{vib}(T) = \sum_{n=0}^\infty \exp \left( -\frac{\left( n + \frac{1}{2} \right) \hbar
       \omega}{k_\mathrm{B} T} \right)
```

This line pulls original function docstring into documentation automatically

… but you can append on more information or equations than the docstring by writing additional documentation here

DOCUMENTATION HTML PAGE

RMG-Py/documentation/build/html/
reference/statmech/harmonicoscillator.rst



Documentation » RMG API Reference » Statistical mechanics (rmgpy.statmech) » previous | next | modules | index

Previous topic
rmgpy.statmech.SphericalTop Rotor

Next topic
Torsional degrees of freedom

This Page
Edit this page

Quick search
Go
Enter search terms or a module, class or function name.

rmgpy.statmech.HarmonicOscillator

class rmgpy.statmech.**HarmonicOscillator**

HarmonicOscillator(frequencies=None, quantum=True)

A statistical mechanical model of a set of one-dimensional independent harmonic oscillators. The attributes are:

Attribute	Description
frequencies	The vibrational frequencies of the oscillators
quantum	`True` to use the quantum mechanical model, `False`

In the majority of chemical applications, the energy levels of the
quantum mechanical treatment. Fortunately, the harmonic oscillat

Many vibrational motions are well-described as one-dimensional quantum harmonic oscillators. The time-independent Schrödinger equation for such an oscillator is given by

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\Psi(x) + \frac{1}{2}m\omega^2 x^2 \Psi(x) = E\Psi(x)$$

where m is the total mass of the particle. The harmonic potential results in quantized solutions to the above with the following energy levels:

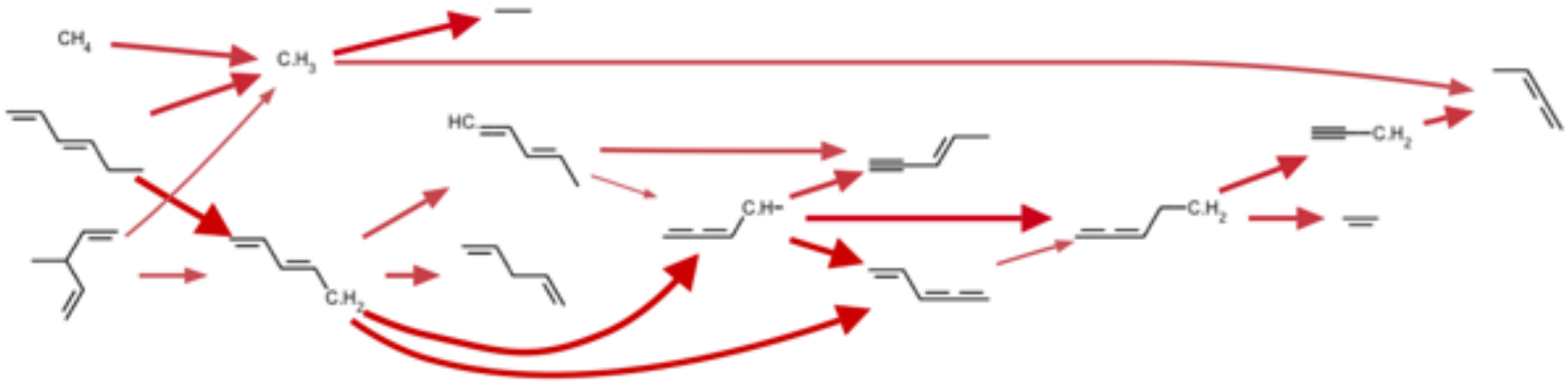
$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega \qquad n = 0, 1, 2, \ldots$$

This page has the combined python docstring information as well as the extra documentation you added

NOTE: The official github documentation pages for RMG-Py automatically update when changes are made to the code docstrings, so you don't have to do any extra work

Steps for writing RMG-Py documentation

1. Create and edit new documentation pages (.rst files) locally. Use code references where appropriate to save time.
2. Build html pages locally to preview that the pages are satisfactory in layout.
3. Use **git** to **commit** the source .rst files that you modified to the RMG-Py github repository.
4. You are done! Official RMG-Py documentation pages will be updated automatically.



webRMG

webRMG

- URLs:
  - Official version: http://rmg.mit.edu
  - Development (newer) version: http://dev.rmg.mit.edu
- Make RMG's databases more transparent, accessible, and modifiable
- Web tools for visualizing kinetics and reactions more easily
- Built on a Django python framework which queries the RMG-Py code itself!

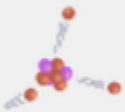
Features

Database

Browse the RMG database of chemical parameters

Molecule Search

Generate an image of a molecule and search for its thermochemistry



Kinetics Search

Search for the kinetics of a chemical reaction

Simulation & Tools

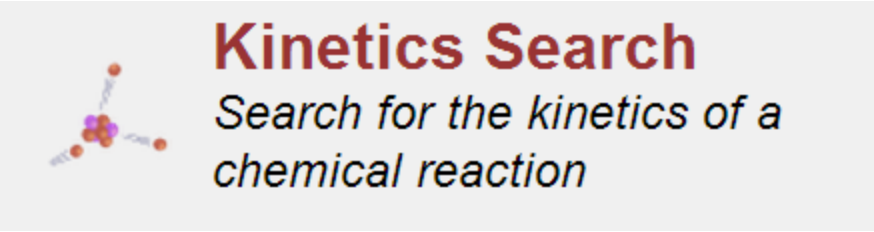
Additional tools to supplement RMG

- Editing or adding new rates or thermo data to the database
  - Edit source online with account
  - Changes make an automatic commit on github
    - Commit gets reviewed by Green Group members before being published to official
- Export database for RMG-Java
  - Converts RMG-Py database to RMG-Java database format
  - Exports current version of database on website in .zip or .tar.gz format

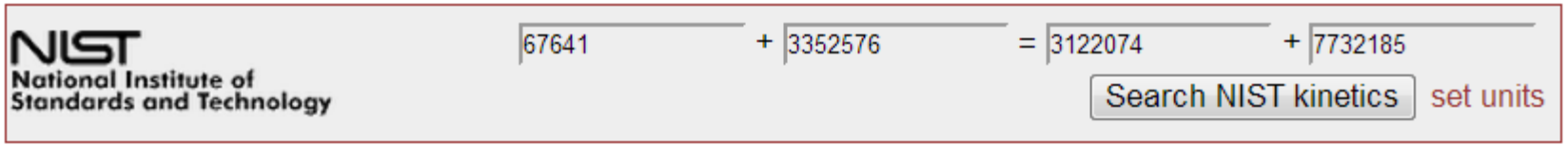
Molecule Search

Generate an image of a molecule and search for its thermochemistry

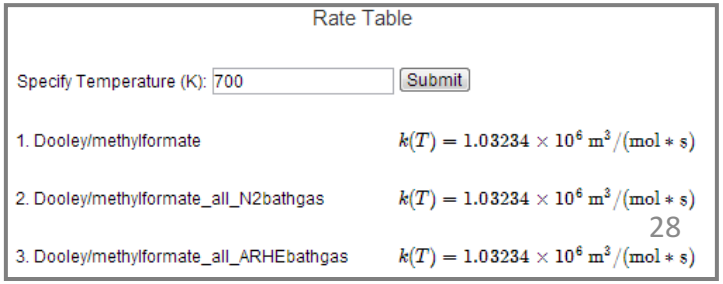
- Use any identifier and convert it to an adjacency list used in RMG
  - SMILES
  - InChI
  - Common chemical names
- Preview molecule while editing adjacency list
- View molecule data:
  - Molecular weight, identifiers, thermo



- Compare kinetics in model with kinetics found in RMG's databases
  - Displays matching reaction library results, rate estimates from RMG-Java and RMG-Py, identifies rate rule contributions, links to sources
  - Toolbar for searching NIST database automatically fills in reactant and product fields



  - Compare rates at a given T



Simulation & Tools

Additional tools to supplement RMG

- Model visualization (RMG-Py and Java friendly)
  - Visualize a model generated from RMG using its chemkin file and RMG dictionary on HTML page
  - Clicking any reaction link searches RMG database for reaction kinetics
  - Clicking on any molecule's image gives additional info and link to thermochemistry
- Model comparison (RMG-Py and Java friendly)
  - Compare two models against each other
  - Identify common vs. unique species and reactions
  - Identify similarities or differences in thermo and kinetics

Simulation & Tools

Additional tools to supplement RMG

- Flux diagram generation (RMG-Py and Java friendly)
  - Create video of a reaction network using RMG condition file, chemkin file, and RMG dictionary
  - Can optionally use chemkin output file for a non-isothermal simulation (*may currently be broken*)
- Populate Reactions (RMG-Py only)
  - Generate a list of all reactions occurring given a set of initial species listed in a condition file (produces the set of edge reactions in a simulation)

Simulation & Tools

Additional tools to supplement RMG

- Plot forward and reverse kinetics (RMG-Py and Java friendly)
  - Use chemkin file and RMG dictionary (optional) to generate plots of forward and reverse reaction kinetics
- Create RMG-Java Kinetics Library
  - Convert a chemkin file and RMG dictionary into a RMG-Java kinetics library
- Merge models
  - Generate merged chemkin file and RMG dictionary for 2 models

Simulation & Tools

Additional tools to supplement RMG

- Create Input File (RMG-Py only) (*may currently be broken*)
  - Use web form to generate input file for a RMG job or modify an existing input file
  - More user friendly and less error prone than working with source code
  - Provides filled-in values as recommendations for advanced options in the input file