Milo 1.0.2

User Manual

1. Introduction

1.1. About Milo

Milo is a quasiclassical direct dynamics suite that interfaces with the electronic structure program Gaussian. While other direct dynamics are available, they are often difficult to use and have little documentation. Most importantly, it is also difficult to quickly add new features to these programs. Milo is a complete rework in Python of our previous C++ based program, DynSuite, and was designed to overcome these shortcomings. While Python is not as computationally efficient as C++, the ease of distribution and the time needed to add new features make it the best option, especially seeing as most computation time is spent inside the electronic structure program. The overarching goal of Milo is to allow computational chemistry groups to focus more on chemistry than software.

1.2. Contact Information

For help using Milo, to report bugs, or to request features, please contact Daniel Ess at dhe@chem.byu.edu.

1.3. Citing Milo

Please cite Milo 1.0.2 as:
Milo, Revision 1.0.2, M. S. Teynor, N. Wohlgemuth, L. Carlson, J. Huang, S. L. Pugh, B. O. Grant, R. S. Hamilton, R. Carlsen, D. H. Ess, Brigham Young University, Provo UT, 2021.

1.4 Version History

Milo 1.0.0 (2020 Oct 31)
- Add basic features for gas-phase direct dynamics.

Milo 1.0.1 (2021 May 5)
- Minor bug fixes.

Milo 1.0.2 (2021 Aug 23)
- Fixed bug that prevented Gaussian 09 and the Velocity Verlet algorithm from being called.

2. Installation & Usage

2.1. Requirements

Python:
Milo has only been tested against Python 3.8. It is expected to work with Python 3.6+, but will not work with Python 3.5 or older.

Gaussian:
Milo can interface with both Gaussian 16 and Gaussian 09 to perform force calculation. Milo expects Gaussian to be loaded as a module, as either “g16” for Gaussian 16 or “g09” for Gaussian 09.

Operating System:
Milo is only intended to work on a Linux based operating system through the command line.

2.2. Installation Guide

1. Unzip `milo-1.0.2.zip` in your home directory
2. Add the following to your `.bashrc`:
   ```
   export PYTHONPATH=$PYTHONPATH:$HOME/milo-1.0.2
   ```
3. (Optional) Also add the following to your `.bashrc`, to make calling scripts easier:
   ```
   export PATH=$PATH:$HOME/milo-1.0.2/milo_1_0_2/tools
   module load python/3.8
   ```
4. Source your `.bashrc` or restart your session.

You are now ready to run your first Milo job.

2.3. Using Milo

To run a Milo job with Gaussian 16:

```
module load python/3.8
module load g16
python -m milo_1_0_2 < job.in > job.out
```

To run a Milo job with Gaussian 09:

```
module load python/3.8
module load g09
python -m milo_1_0_2 < job.in > job.out
```

2.4. Typical Milo Workflow

Single gas-phase dynamics trajectory:

1. Run a high-precision frequency calculation on the optimized starting geometry in Gaussian using 'freq=hpmodes'.
2. Use the parse_frequencies.py script to create a Milo input file using the output file from the high-precision frequency calculation. For help using make_milo_input.py, run '`python parse_frequencies.py --help`'.
3. Edit the input file created by parse_frequencies.py with details for your job. (See Section 3 of this manual for details about input.)
4. Edit a copy of the submission script.
5. Submit the job to be run.

Ensemble of gas-phase, independent trajectories:

1. Run a high-precision frequency calculation on the optimized starting geometry in Gaussian using 'freq=hpmodes'.
2. Use the parse_frequencies.py script to create a Milo input file using the output file from the high-precision frequency calculation. For help using make_milo_input.py, run '`python parse_frequencies.py --help`'.
3. Edit the input file created by parse_frequencies.py with details for your job. (See Section 3 of this manual for details about input.)
4. (Optional - for running trajectories both forwards and backwards.) Make a copy of the newly edited input file, and edit this copy to contain the details needed for running trajectories in the other direction. Usually you will only change the phase (switching bring_together and push_apart) and possibly the number of steps.
5. Use setup_ensemble.py to create the forward input files, backward input files, and submission scripts for all the jobs. For help using setup_trajectories.py, run '`python setup_ensemble.py --help`'.
6. Submit the jobs to be run.

2.5. SLURM Submission Script Example

The following is intended to be saved as a .sh file to allow users to submit jobs to BYU's Office of Research Computing. While it may not work perfectly as is for other HPC systems, it will hopefully be a helpful starting place for other SLURM users.

The first few lines of this script should be edited for each job. JOB_NAME should be set as the basename (with no extension) of the .in input file. To run Gaussian 09 jobs, uncomment the `module load g09` line and remove the `module load g16` line.

```
#!/bin/bash
#SBATCH --nodes=1 --ntasks=1 --cpus-per-task=24
#SBATCH --mem=24G
#SBATCH -t 16:00:00
#SBATCH -C 'avx2'

export JOB_NAME=

export TEMPORARY_DIR=/tmp/$SLURM_JOB_ID
export JOB_SOURCE_DIR="$SLURM_SUBMIT_DIR"

function cleanup
{
  echo "---"
  echo "Starting clean up"

  for file in "$TEMPORARY_DIR"/*.{out,xyz,txt}; do
    [ -e $file ] || continue
    cp -vp $file "$JOB_SOURCE_DIR"
  done

  mkdir -p "$JOB_SOURCE_DIR/com_log_files/"
  echo "Archiving .com and .log files"
  tar -cf ${JOB_NAME}_${SLURM_JOB_ID}_com_log_files.tar *.{com,log}
  cp -v ${JOB_NAME}_${SLURM_JOB_ID}_com_log_files.tar
"${JOB_SOURCE_DIR}/com_log_files/"

  cd "$JOB_SOURCE_DIR"
  rm -fr "$TEMPORARY_DIR"

  echo "Clean up finished at `date`"
}

echo "---"
echo "Milo SLURM Diagnostic Information"
echo "---"
echo "Start date and time: `date`"
echo "---"
echo "Nodes assigned:"
cat `/fslapps/fslutils/generate_pbs_nodefile`
echo "---"
echo "Job Source Directory: $JOB_SOURCE_DIR"
```

```
echo "Temporary Directory: $TEMPORARY_DIR"
mkdir $TEMPORARY_DIR
cp -v "$JOB_SOURCE_DIR/$JOB_NAME.in" "$TEMPORARY_DIR"
cd "$TEMPORARY_DIR"
echo "---"
echo "Starting Milo run"

module load python/3.8
#module load g09
module load g16
python -m milo_1_0_2 < "$JOB_NAME.in" >
"${JOB_NAME}_${SLURM_JOB_ID}.out" &
pid=$!
# Associate the function "cleanup" with the TERM signal, which is
usually
# how jobs get killed
trap "kill $pid; cleanup; exit 1" TERM SIGTERM KILL SIGKILL
wait $pid

cleanup
```

3. Input

3.1. Overview

Milo uses text input files, which are separated into sections following this general format:

```
$section_name
    parameter_name option
    parameter_name option option
    ...
$end
```

Note: Input files are case insensitive and extra whitespace is ignored.

Required Section:

Section Name	Description
`$job`	General job parameters
`$molecule`	Spin, charge and cartesian coordinates of the molecule
`$frequency_data *`	Frequency data needed for initial velocity sampling
`$velocities *`	Initial cartesian velocities for each atom

`*` Either `$frequency_data` or `$velocities`, but not both, must be given.

Optional Sections:

Section Name	Description
`$isotope`	Isotopic numbers or masses in amu
`$comment`	Ignored by Milo

3.2. Job Section

The `$job` section is used to specify parameters for how the job should run. Following is a list of all the parameters that can be specified in the `$job` section of an input file.

step_size

Specifies the length of each time step in femtoseconds.

Default:

1.00

Options:

n > 0 Step size of *n* fs.

max_steps

Specifies how many steps should be completed before stopping.

Default:

no_limit

Options:

no_limit	No max steps (run until externally forced to quit).
n	Stop after *n* steps.

Note:

When `no_limit` is selected, Milo will not generate the final .xyz file.

If *n* is 0, Milo will do the initial sampling, and calculate initial velocities and geometry displacements, but will not do any force calculations.

temperature

Specifies the temperature in kelvin used for initial velocity sampling.

Default:

298.15

Options:

n	Sample at *n* kelvin.

oscillator_type

Specifies whether the harmonic oscillator approximation used to sample initial velocities and positions should be treated classically or quasiclassically.

Default:

quasiclassical.

Options:

classical	Do not account for zero point energy.
quasiclassical	Account for zero point energy.

geometry_displacement

Specifies whether the initial geometry at time step 0 should be modified using the harmonic oscillator model specified by `oscailltor_type`.

Default:

off.

Options:

edge_weighted	Modify initial geometry using an edge weighted distribution.
gaussian	Modify initial geometry using a gaussian distribution.
uniform	Modify initial geometry using a uniform distribution.
off	Do not modify initial geometry.

rotational_energy

Specifies whether rotational energy should be added to the molecule.

Default:

off.

Options:

on	Add rotational energy.
off	Do not add rotational energy.

integration_algorithm

Specifies which integration algorithm should be used to propagate forces and update atom positions at each time step.

Default:

verlet.

Options:

verlet	Use the Verlet algorithm.
velocity_verlet	Use the Velocity Verlet algorithm..

energy_boost

Specifies if the energy of the system should be boosted. This is only possible when `$frequency_data` is used to sample initial velocities. When energy boost is used, it iteratively raises the temperature and resamples until the vibrational energy falls between the values given.

Default:

off.

Options:

off	Do not boost energy.
on *min max*	Boost energy until the kinetic energy is between *min* and *max*, where *min* and *max* are given in kcal/mol.

Note:

This is useful for raising the energy of an intermediate to mirror the energy of a transition state. This allows a researcher to postulate whether a dynamic effect originates from the reaction coordinate motion of the transition state.

fixed_mode_direction

Specifies if the given vibrational mode should be followed in the positive or negative direction. Only works if initial velocities are being sampled from frequency data.

Default:

None.

Options:

n 1	Follow the *n*th vibrational mode in the positive direction.
n -1	Follow the *n*th vibrational mode in the negative direction.

Notes:

This command can be specified multiple times to control multiple vibrational modes. For example:

```
fixed_mode_direction    2 -1
fixed_mode_direction    4 1
```

phase

Specifies the direction the imaginary, transition state mode (i.e. the first negative frequency listed) should be followed for initial velocity sampling.

Default:

random.

Options:

random	Randomly choose the direction.
bring_together *n m*	Follow the imaginary mode in the direction that leads to a decreased distance between the *n*th and *m*th atoms.
push_apart *n m*	Follow the imaginary mode in the direction that leads to an increased distance between the *n*th and *m*th atoms.

Note:

If the initial geometry has no imaginary mode the phase parameter will be ignored.

random_seed

Specifies the random seed to use for classical and quasiclassical sampling.

Default:

generate.

Options:

generate	Use an internally generated random seed. Will be different every time the job is run.
n	Use *n* as the random seed.

Note:

When used with the phase parameter, the random seed can be used to ensure that forward and backward trajectories (two different jobs) have connected initial velocities.

program

Specifies the electronic structure program to use for force calculations.

Default:

gaussian16.

Options:

gaussian09	Gaussian 09
gaussian16	Gaussian 16

gaussian_header

Specifies the method, basis set, implicit solvent model, etc. to be copied verbatim into the Gaussian 09 or Gaussian 16 route section when submitting calculations.

Default:

No default.

Options:

method Pastes *method* in the appropriate place when submitting calculations.

Note:

Do not include the 'force' keyword. Only specify the model chemistry. Example:

```
$job
    gaussian_header  m06 6-31g(d,p)    # do not put '# force'
$end
```

gaussian_footer

Specifies any details to go after the molecule structure at the end of the file. Usually used for specifying basis sets and pseudo potentials not listed in the header.

Default:

Blank. If not specified, no footer is used.

Options:

footer Paste *footer* in the appropriate place when submitting calculations.

Note:

Often *footer* will need to represent multiple lines. This can be done by using "\n" where the new line should go.

Example:

```
    gaussian_footer    line_1\nline_2\nline_3
```

It will often be easier to use the `$gaussian_footer` section for multiline footers.

memory

Specifies the memory, in GB, the electronic structure program will request when performing force calculations.

Default:

None. When this option is not given, no specific amount of memory will be requested.

Options:

(memory) '%mem=*(memory)*gb' will be put at the top of the Gaussian .com file.

processors

Specifies the number of processors the electronic structure program will request when performing force calculations.

Default:

None. When this option is not given, no specific number of processors will be requested.

Options:

(processors) '%nprocshared=*(processors)*' will be put at the top of the Gaussian com file.

3.3. Gaussian Footer Section

The `$gaussian_footer` section is used to specify the details to go after the molecule structure at the end of the file. Usually used for specifying basis sets and pseudo potentials not listed in the header. If this section is included, the gaussian_footer parameter in the `$job` section will be ignored. Comments will not be removed from this section.

Example:

```
$gaussian_footer
H C P 0
6-31g(d,p)
****
Ir 0
lanl2dz
****

Ir 0
lanl2dz

$end
```

3.4. Molecule Section

The `$molecule` section is used to specify the charge, spin, atoms, and coordinates in angstroms of the molecule.

The charge and spin are listed, in that order, on the first line under the `$molecule` heading, separated by only whitespace.

After the charge and spin line, the atom and the corresponding x-, y- and z-coordinates in angstroms are listed, with every atom on its own line. The atomic symbol is given first, followed by the x-, then y-, then z-coordinates, all separated by only whitespace.

The order that atoms are listed determines the indexes used for the `$isotope` and `$bond_tracker` sections, with the first atom being one, and so on.

Extra spaces and tabs can be useful to make the input file more readable, but are not needed.

Example:

```
$molecule
    0 1
     O    0.00000000    0.00000000    0.11903700   # index 1
     H    0.00000000    0.75704900   -0.47614700
     H    0.00000000   -0.75704900   -0.47614700
$end
```

3.5. Isotope Section

Note: Care should be taken when using the `$isotope` section. Frequencies, displacements, reduced mass, and force constants (the frequency calculation data used in initial velocity sampling) change when different isotopes are used. Do not use this section to change the isotopes without first running a new frequency calculation.

The `$isotope` section is used to specify the isotope masses. The index of the atom is given first, followed by either the exact isotopic mass in amu or the mass number. For example, for ^{18}O either 18 or 17.99915961286 could be given, with the same result. If a decimal point is in the number, it is assumed to be the exact isotopic mass, otherwise it is assumed to be the mass number.

When the isotope is not given for an atom, the most abundant isotope is used. Isotope data was collected from the National Institute of Standards and Technology online database.

Example:

```
$molecule
    0 1
     O    0.00000000    0.00000000    0.11903700  # index 1
     H    0.00000000    0.75704900   -0.47614700
     H    0.00000000   -0.75704900   -0.47614700
$end

$isotope
    1    16.0  # Use exactly 16.0 amu as the mass of oxygen
    3    2     # Use deuterium for the second hydrogen (3rd atom)
$end
```

3.6. Frequency Data Section

Note: The `$velocities` and `$frequency_data` sections are mutually exclusive. If both sections are given, the job will return an error.

The `$frequency_data` section is used to input the frequencies (in cm^{-1}), the reduced masses (in amu), the force constants (in millidyne/angstrom), and the cartesian displacements (in angstroms) from a high-precision frequency calculator (in Gaussian 09 and 16, this is requested by using force=hpmodes in the route line), which used the same model chemistry that will be used for the dynamics.

All the data for a single mode is put on one line, in the following order: frequency, reduced mass, force constants, cartesian displacements. All numbers are separated by only whitespace (no commas). The cartesian displacements are listed x, y, z by atom in the same order they appear in the `$molecule` section.

This section would be very tedious to type out by hand. It is recommended that you use the make_milo_input.py script to parse frequencies from a Gaussian frequency calculation.

Partial example (end of lines are truncated):

```
$frequency_data
  # frequency  r.mass  f.con.  atom1_x  atom1_y  atom1_z  atom2_x ...
    1682.1354  1.0895  1.8163  0.00000  0.00000 -0.07382  0.00000 ...
    3524.4296  1.0389  7.6032  0.00000  0.00000  0.04553  0.00000 ...
    3668.7401  1.0827  8.5864  0.00000 -0.07070  0.00000  0.00000 ...
$end
```

3.7. Velocities Section

Note: The `$velocities` and `$frequency_data` sections are mutually exclusive. If both sections are given, the job will return an error.

The `$velocities` section is used to specify initial cartesian velocities (in meters/second). They can be specified in scientific or standard notation. The x-, y- and z-components of the velocity for each atom are listed, with every atom on its own line following the same order that was used in the `$molecule` section, all separated by only whitespace.

Example:

```
$velocities
    0.000000e+00   -4.501297e+02   -6.072997e+02
    0.000000e+00    1.392428e+03    2.097487e+03
    0.000000e+00    5.751837e+03    7.540427e+03
$end
```

3.8. Comments

Comments can be useful but do not affect how Milo runs.

Block comments are set off with `$comment`.

```
$comment
    This is a comment and will be ignored by Milo.
$end
```

In-line comments are started with a number sign. Everything after the number sign is ignored.

```
$job
    temperature    298.15  # This is also a comment.
$end
```

3.9. Examples

Example input file for simple quasiclassical trajectory of water:

```
$comment
    Example input file for a simple quasiclassical trajectory of a
    water molecule.
$end

$job
    step_size              1.00
    max_steps              100
    temperature            298
    gaussian_header        m062x 3-21g
$end

$molecule
    0 1
     O    0.00000000    0.00000000    0.11903700
     H    0.00000000    0.75704900   -0.47614700
     H    0.00000000   -0.75704900   -0.47614700
$end

$frequency_data
    # All the data for a mode needs to be on a single line. Before
    # submitting, make sure there are only three lines of data.
    # Mode 1
    1682.1354 1.0895 1.8163  -0.00000  0.00000 -0.07382   0.00000
0.39258  0.58580   0.00000 -0.39258  0.58580
    # Mode 2
    3524.4296 1.0389 7.6032   0.00000 -0.00000  0.04553   0.00000
0.60700 -0.36126  -0.00000 -0.60700 -0.36126
    # Mode 3
    3668.7401 1.0827 8.5864  -0.00000 -0.07070 -0.00000   0.00000
0.56106 -0.42745   0.00000  0.56106  0.42745
$end
```

Example input file for quasiclassical trajectory initiated from a transition state between two conformational isomers of vinyl alcohol.

```
$job
    gaussian_header         m06 6-31g(d,p)
    step_size               1.00
    max_steps               100
    temperature             298.15
    phase                   bring_together 3 7
    memory                  24
    processors              24
    random_seed             generate
$end

$molecule
    0 1
    C     1.200474    -0.217166     0.005312
    H     2.140373     0.322623     0.057025
    H     1.229644    -1.301649    -0.058870
    C     0.047579     0.438089     0.007341
    H     0.006491     1.530036     0.049096
    O    -1.172883    -0.178611    -0.110414
    H    -1.481767    -0.447660     0.760148
$end

$isotope
     1    12.00000
     2     1.00783
     3     1.00783
     4    12.00000
     5     1.00783
     6    15.99491
     7     1.00783
$end

$frequency_data
    -429.7212  1.1336  0.1233  -0.00292  0.00889 -0.03109   0.00077
0.00556 -0.05316  -0.02203  0.01414 -0.10613  -0.00366  0.00475
0.04786  -0.00607  0.00297  0.18807   0.03616 -0.06491 -0.01976
-0.46819  0.84501  0.08511
     482.6444  2.1944  0.3012   0.15280  0.02762 -0.00072  -0.10925
0.49529 -0.00808   0.72285  0.04312  0.04838   0.00402 -0.20033
```

```
-0.00083   0.11275 -0.19094 -0.00473  -0.14914  0.09901 -0.00017
-0.22657  0.13759 -0.01446
     695.5585  1.3254  0.3778   0.00107 -0.00235  0.03220  -0.03099
-0.01134  0.72199   0.04196  0.02776 -0.44882   0.00331  0.00052
-0.15641   0.00328 -0.01524  0.30098  -0.02132 -0.01067  0.04367
0.27193  0.18995  0.21170
     891.2537  1.3611  0.6370  -0.00076  0.00699 -0.16945  -0.02583
-0.03426  0.71555  -0.00514 -0.04272  0.66491  -0.00354  0.00018
0.05685   0.00468  0.00553 -0.06946   0.00907 -0.00035  0.00355
-0.06652 -0.00829 -0.02660
     952.7036  1.9153  1.0242  -0.18213 -0.09106 -0.00438  -0.55407
0.56242 -0.02924   0.49500 -0.05956  0.01882  -0.05486  0.00069
-0.01172   0.20515  0.01308  0.04767   0.16262  0.03519  0.01168
0.09476  0.00148 -0.03096
     980.3828  1.0914  0.6181   0.00317  0.00235  0.00543   0.02508
-0.00777 -0.28773  -0.02140 -0.02932  0.51127   0.00461  0.00460
-0.08513  -0.01986 -0.03000  0.79321  -0.01227 -0.00069 -0.00695
0.11819 -0.00482  0.04248
    1149.9397  2.0261  1.5786  -0.00374 -0.07273 -0.01432  -0.08971
0.11002 -0.09390   0.35685 -0.06467  0.07319   0.08597  0.15162
0.10428   0.27266  0.16442 -0.10035  -0.13837 -0.08878 -0.08134
0.67719  0.25981  0.34089
    1235.6207  1.6208  1.4579  -0.01873  0.07168 -0.00653  -0.00836
0.03099 -0.07646  -0.38386  0.05709  0.05802  -0.06357 -0.17120
0.06609  -0.18116 -0.17310 -0.09807   0.05266  0.05813 -0.05501
0.71758  0.34749  0.28045
    1327.1159  1.1186  1.1608  -0.02036  0.06769  0.00307  -0.11530
0.21117  0.01657  -0.44062  0.06366 -0.00853   0.03377 -0.04183
-0.00776   0.84930 -0.01157  0.02424  -0.02365 -0.03141  0.00317
-0.07760 -0.07267 -0.02678
    1403.6059  1.2317  1.4297   0.04510 -0.06215 -0.00202  -0.35029
0.60774  0.03103  -0.51781 -0.06472 -0.01154   0.11240  0.02391
-0.00193  -0.45595 -0.00516  0.00098  -0.03007 -0.00352  0.00292
-0.07410 -0.02666 -0.01981
    1738.0235  5.1967  9.2488  -0.35203  0.19150  0.00084  -0.08250
-0.41701 -0.01100   0.35876  0.28926  0.01289   0.44331 -0.14033
-0.00242  -0.36738 -0.26904 -0.01922  -0.05461 -0.01108  0.00424
-0.12907 -0.03659 -0.03122
    3113.2425  1.0866  6.2048   0.00700  0.00097  0.00022  -0.13882
-0.08635 -0.00870  -0.00086  0.10245  0.00692   0.01042 -0.08363
-0.00287  -0.04714  0.97566  0.03583  -0.00121 -0.00015 -0.00060
-0.00138 -0.00519  0.00705
```

```
    3160.0842  1.0687  6.2876   0.05715 -0.04183 -0.00124  -0.54176
-0.31896 -0.03028  -0.01660  0.75352  0.04508  -0.01195  0.01933
0.00057   0.01313 -0.16728 -0.00579   0.00028 -0.00005  0.00003
0.00269  0.00144 -0.00156
    3260.9026  1.1169  6.9975  -0.05390 -0.08325 -0.00613   0.66065
0.37139  0.03577  -0.02726  0.63939  0.03837   0.00022 -0.00701
-0.00027  -0.00410  0.05790  0.00200   0.00053  0.00044 -0.00002
0.00143 -0.00095  0.00035
    3904.2252  1.0668  9.5804  -0.00058 -0.00012  0.00008  -0.00144
-0.00018  0.00195   0.00051  0.00095 -0.00171  -0.00092  0.00008
-0.00110  -0.00223 -0.00864 -0.00222   0.02220  0.01854 -0.05562
-0.33141 -0.28593  0.89688
$end
```