PySCES

Manual

Version 1.0.0

June 7, 2024

Contents

1 Introduction

PySCES is a highly parallelized PYTHON program to perform nonadiabatic molecular dynamics simulation for large–scale open quantum systems through the Linearized Semiclassical Initial Value Representation (LSC–IVR). Every dynamical variable in a simulation is propagated in phase space as prescribed by LSC-IVR. The electronic structure variables necessary to integrate the equations of motion are provided "on-the-fly" through the interface established between the code and external electronic structure softwares, specifically TeraChem[5, 4] and GAMESS.[2] The current implementation is capable of computing an electronic population correlation function through three different population estimators. For more details, we refer readers to earlier studies:

LSC–IVR

- W. H. Miller, *J. Phys. Chem. A* 105, 2942– 2955 (2001)
- Q. Shi and E. Geva, *J. Chem. Phys.* 118, 8173 (2003)

On-the-fly implementation of LSC–IVR and different population estimators

- K. Miyazaki and N. Ananth, *J. Chem. Phys.*, 159, 124110 (2023)

1.1 Acknowledgements

PySCES was initially written by Ken Miyazaki while at Cornell University. Christopher Myers extended the code to call TeraChem as the electronic structure driver at the University of California, Merced. The authors would also like to thank Thomas Trepl at University of Bayreuth, Germany, for his contributions to the code, including procedures for correcting nonadiabatic coupling sign-flips. The PySCES logo was designed by Fernanda Giongo Fernandes.

1.2 Citing The Code

If you find our code useful, we politely ask that you cite the following papers in any and all publications that utilize any results generated:

- C. A. Myers, K. Miyazaki, T. Trepl, C. M. Isborn, and N Ananth, *J. Chem. Phys.*, XXXX, XXXXXX, (2024)
- K. Miyazaki and N. Ananth, *J. Chem. Phys.*, 159, 124110 (2023)

2 Getting Started

2.1 Installation

The easiest way to get PySCES is to clone the repository from github. As the code is still in active development, this makes it convenient to get the latest changes as they are uploaded. PySCES can be installed with pip, and we recommend you make a new python environment with Conda prior to the installation:

```
conda create -n pysces python
conda activate pysces
cd /dir/where/you/git/cloned/
pip install -e .
```

The `-e` will tell pip not to copy over the code itself into your python environment. To get the latest updates, just go back to the location you cloned the repository, run a `git pull`, and your installation will also be updated. You do not need to run a `pip install` again.

2.2 Running PySCES

After installation, the command `pysces` will be registered with your conda environment and is used to initialize and run the simulation. After running this command, PySCES will look for an input file `input_simulation_local.py` in the current directory. This file is formatted in python and sets the primary variables that control the simulation.

Each input will usually contain the following information:

- Number of atoms in the molecule system.
- Number of electronic states to be simulated.
- Temperature to sample nuclear Wigner distributions from.
- The electronic state initially photoexcited.
- Either TeraChem or GAMESS runtime options.

Examples of this input file can be found in the `examples` directory of the main repository and can be used as basic templates for running simulations. The variable names used to control the simulation, including the settings listed above, are described in later sections of this manual.

2.3 Simulation Output

Upon starting a simulation, PySCES will create a directory called `logs` to store the relevant simulation information, including molecular positions, momenta, and electronic structure information.

`nuc_geo.xyz`

Contains the series of nuclear coordinates (in Angstroms) of the molecular system being simulated in `.xyz` format. Each comment line lists the simulation time (in a.u.) at which the geometry is recorded.

$$\begin{array}{llll} \text{No. atoms} & & & \\ \quad t_0 & & & \\ \text{<elm}_1\text{>} & X_1 & Y_1 & Z_1 \\ \text{<elm}_2\text{>} & X_2 & Y_2 & Z_2 \\ \quad \vdots & & & \end{array}$$

nuclear_P.txt

Contains the time propagation of nuclear momenta (a.u.) of the molecular system in `.xyz` format. The same format of `nuc_geo.xyz` is also used here

$$\begin{array}{llll} \text{No. atoms} & & & \\ \quad t_0 & & & \\ \text{<elm}_1\text{>} & P_{x1} & P_{y1} & P_{z1} \\ \text{<elm}_2\text{>} & P_{x2} & P_{y2} & P_{z2} \end{array}$$

The following data files are designed to be loaded into pandas as space delimited files, while remaining easily readable in their native text form:

```
data = pandas.read_csv('file.txt', delim_whitespace=True, comment='#')
```

The first line contains names of each datum with the following lines containing the data itself.

corr.txt

Contains the correlation function (occupations) of each electronic state.

Time	Total	S0	S1	S2	···
t_0	$\sum_i C_i(t_0)$	$C_0(t_0)$	$C_1(t_0)$	$C_2(t_0)$	$\cdots$
t_1	$\sum_i C_i(t_1)$	$C_0(t_1)$	$C_1(t_1)$	$C_2(t_1)$	$\cdots$
t_2	$\sum_i C_i(t_2)$	$C_0(t_2)$	$C_1(t_2)$	$C_2(t_2)$	$\cdots$
		$\vdots$			

electric_pq.txt

Contains the time propagation of electronic position and momenta variables (x_i, p_i).

Time	p0	q0	p1	q1	···
t_0	$p_0(t_0)$	$q_0(t_0)$	$p_1(t_0)$	$q_1(t_0)$	$\cdots$
t_1	$p_0(t_1)$	$q_0(t_1)$	$p_1(t_1)$	$q_1(t_1)$	$\cdots$
t_2	$p_0(t_2)$	$q_0(t_2)$	$p_1(t_2)$	$q_1(t_2)$	$\cdots$
		$\vdots$			

energy.txt

Contains the time propagation of the adiabatic energies of each state, computed with either TeraChem or GAMESS.

Time	Total	S0	S1	S2	⋯
t_0	$E_T(t_0)$	$E_0(t_0)$	$E_1(t_0)$	$E_2(t_0)$	⋯
t_1	$E_T(t_1)$	$E_0(t_1)$	$E_1(t_1)$	$E_2(t_1)$	⋯
t_2	$E_T(t_2)$	$E_0(t_2)$	$E_1(t_2)$	$E_2(t_2)$	⋯
		⋮			

grad.txt

Contains the adiabatic gradients (in a.u.) of the energies of each state, computed with either TeraChem or GAMESS.

S0	S1	S2	⋯
# time_step 0			
# time t_0			
$g_0(x_1)$	$g_1(x_1)$	$g_1(x_1)$	⋯
$g_0(y_1)$	$g_1(y_1)$	$g_1(y_1)$	⋯
$g_0(z_1)$	$g_1(z_1)$	$g_1(z_1)$	⋯
$g_0(x_2)$	$g_1(x_2)$	$g_1(x_2)$	⋯
$g_0(y_2)$	$g_1(y_2)$	$g_1(y_2)$	⋯
$g_0(z_2)$	$g_1(z_2)$	$g_1(z_2)$	⋯
	⋮		

Here, the $g_i(\alpha_k)$ is the derivative of adiabatic energy E_i with respect to the k-th nuclei in the α Cartesian direction.

nac.txt

Contains the adiabatic coupling vectors (in a.u.) between each adiabatic states, computed with either TeraChem or GAMESS.

S0_S1	S0_S2	⋯	S1_S2	⋯
# time_step 0				
# time t_0				
$d_{01}(x_1)$	$d_{02}(x_1)$	⋯	$d_{12}(x_1)$	⋯
$d_{01}(y_1)$	$d_{02}(y_1)$	⋯	$d_{12}(y_1)$	⋯
$d_{01}(z_1)$	$d_{02}(z_1)$	⋯	$d_{12}(z_1)$	⋯
$d_{01}(x_2)$	$d_{02}(x_2)$	⋯	$d_{12}(x_2)$	⋯
$d_{01}(y_2)$	$d_{02}(y_2)$	⋯	$d_{12}(y_2)$	⋯
$d_{01}(z_2)$	$d_{02}(z_2)$	⋯	$d_{12}(z_2)$	⋯
	⋮			

Here, the $d_{ij}(\alpha_k)$ is the nonadiabatic coupling between the adiabatic states i and j with respect to the change in k-th nuclei in the α Cartesian direction. More precisely, $d_{ij}(x_k) = \langle \Psi_i | \partial / \partial X_k | \Psi_j \rangle$.

timings.txt

Contains wall time for each electronic structure job. Because GAMESS computes all components (energies, gradients, and couplings) in a single job, only the total time is relevant for a simulation driven by GAMESS.

Total_QC	gradient_0	gradient_1	$\cdots$	nac_0_1	nac_0_2	$\cdots$	Wall_Time
$\sum_i T_i$	$T[\mathbf{g}_0(t_0)]$	$T[\mathbf{g}_1(t_0)]$	$\cdots$	$T[\mathbf{d}_{01}(t_0)]$	$T[\mathbf{d}_{02}(t_0)]$	$\cdots$	$\mathtt{W}(t_0)$
$\sum_i T_i$	$T[\mathbf{g}_0(t_1)]$	$T[\mathbf{g}_1(t_1)]$	$\cdots$	$T[\mathbf{d}_{01}(t_1)]$	$T[\mathbf{d}_{02}(t_1)]$	$\cdots$	$\mathtt{W}(t_1)$
$\sum_i T_i$	$T[\mathbf{g}_0(t_2)]$	$T[\mathbf{g}_1(t_2)]$	$\cdots$	$T[\mathbf{d}_{01}(t_2)]$	$T[\mathbf{d}_{02}(t_2)]$	$\cdots$	$\mathtt{W}(t_2)$
$\sum_i T_i$	$T[\mathbf{g}_0(t_3)]$	$T[\mathbf{g}_1(t_3)]$	$\cdots$	$T[\mathbf{d}_{01}(t_3)]$	$T[\mathbf{d}_{02}(t_3)]$	$\cdots$	$\mathtt{W}(t_3)$
		$\vdots$					

The last column labeled `Wall_Time` is the total wall time for the entire time step, where as the first column labeled `Total_QC` is the sum of all individial wall times measured by each TeraChem server for each job. We use the notation $T[\mathbf{g}_0(t_3)]$ to denote the time measured by the TeraChem server to compute the gradient $\mathbf{g}_0$ at time t_3.

3 General Input Options

nel

Type: `int`
Default: `3`

Description: Number of electronic states to simulate. The identities of electronic states to be included in the simulation are separately specified through `elab` for GAMESS and `tcr_state_options` for TeraChem. For example, `nel=3` and specifying `elab` = [0, 2, 3] or `tcr_state_options=`{'grads=[0, 2, 3]} will use the S_0, S_2, and S_3 states in the simulation.

natom

Type: `int`
Default: `N/A`

Description: Number of atoms in the molecular system

nnuc

Type: `int`
Default: `3*natom`

Description: Number of nuclear degrees of freedom

temp

Type: `float`
Default: `300`

Description: Temperature of the system in degrees kelvin that define the Wigner distributions in nuclear space.

sampling

Type: `string`
Default: `'conventional'`
Description: Type of a population estimator for electronic variables in LSC–IVR. Can be either 'conventional', 'modified', or 'spin'. See reference [3] for more details.
Note 1: The 'spin' option is implemented only for `nel` = 3.
Note 2: In the case of `nel` = 1, the 'wigner' option results in an unphysical radius of sampling, hence only the 'sc' option will be available.

q0

Type: `List[float]`
Default: `[0.0]*nel`
Description: centers of initial electronic position coherent state.

p0

Type: `List[float]`
Default: `[0.0]*nel`
Description: centers of initial electronic momentum coherent state.

pN0

Type: `float`
Default: `0.0`
Description: The center of initial momentum distribution of nuclear normal modes. A non–zero value means all nuclear modes will have a finite initial momentum.

frq_scale

Type: `float`
Default: `0.967`
Description: Scaling factor to multiply all normal mode frequencies by. This value will vary depending on electronic structure methods used to compute normal coordinates and frequencies. A collection of scaling factors for various electronic structure levels can be found at CCCBDB website[1]

integrator

Type: `string`
Default: `'RK4'`
Description: Type of integrator for the equations of motion. Can be either 'RK4' (4th–order Runge Kutta), 'ABM' (4th–order Adams–Bashforth–Moulton predictor–corrector), or 'BSH' (Bulirsch–Stoer).

QC_RUNNER

Type: `string`
Default: `'gamess'`
Description: The type of driver to use for performing electronic structure calculations. This can be set to either 'gamess' or 'terachem'.

restart

Type: `bool`
Default: `False`
Description: Determines if the simulation is restarting from a previous trajectory (True), or if the simulation is starting from new samples of initial conditions (False).

restart_file_in

Type: `string`
Default: `'restart.json'`
Description: The name of a file to save the simulation status from every time step. This file will be used with `restart` option when one desires to restart a simulation from the end of a previous run.

elab

Type: `list[int]`
Default: `[1]`
Description: The list of electronic state labels given by GAMESS that are included in the dynamics simulation. For example, "1" indicates "STATE 1" in GAMESS output file, "3" indicates "STATE 3", and so on. The length of this list must be equal to `nel`.

init_state

Type: `int`
Default: 1
Description: The index of initially occupied electronic state. An initially populated state will be chosen from `elab` according to this index. For example, `init_state = 3` and `elab = [1,2,4,5]`, "STATE 4" in a GAMESS output file will be initially populated.

3.1 RK4 Propagation

tmax

Type: `float`
Default: `20671.0`
Description: Number of time steps to take during the simulation. The default value results in approximately 500fs trajectory when using a time step of 1 a.u..

Hrk4

Type: `float`
Default: `1.0`
Description: RK4 integrator time step in atomic units. 1 a.u. is approximately 41.3 fs

3.2 ABM Propagation

timestep

Type: `float`
Default: `1.0`
Description: ABM integrator time step in atomic units. 1 a.u. is approximately 41.3 fs

nstep

Type: `int`
Default: `16700`
Description: Number of time steps to take during the simulation. The default value results in approximately 400fs trajectory when using a time step of 1 a.u.

3.3 BSH Propagation

NOTE: BSH integrator is not rigorously tested for the performance of a long time simulation. In general, RK4 integrator is recommended over ABM for computational efficiency, while BSH is not recommended for use.

tmax_bsh

Type: `float`
Default: `10`
Description: Maximum propagation time in a.u.

Hbsh

Type: `float`
Default: `3.0`
Description: BSH time step to be tried in atomic units.

tol

Type: `float`
Default: `0.01`
Description: Error tolerance ratio

4 TeraChem Specific Input Options

tcr_port

Type: `str | List[str]`
Default: `not set`
Description: The IP address of the TeraChem server. This can be either a single string, or a list of strings

tcr_host

Type: `int | List[int]`
Default: `9876`
Description: Port number for the TCPB server. Each server must have a unique port number. This can be either a single string, or a list of integers

tcr_server_root

Type: `str`
Default: `current working directory`
Description: Directory where the TeraChem server is located. This is typically the same directory that you started the server in. If not specified, ground and excited states must be computed all over again for every job.

tcr_job_options

Type: `dict`
Default: `empty dict`
Description: Dictionary of options to be passed to TeraChem through the protocol buffers . See the TCPB documentation for more information

tcr_state_opts

Type: `dict[str, int | list]`
Default: `{'max_state': nel-1, 'grads': None}`

Description:

`max_state`

Type: `int`
Default: `nel-1`

Description Maximum excited state to use. If specified, then S_0 -S_{max_state} electronic states will be computed by TeraChem and used in the dynamics.

`grads`

Type: `list[int] | str`
Default: `nel-1`

Description List of integers specifying the (molecular) state indices to use in the simulation. For example, if set to `'grads': [1, 2, 4]`, then the S_1, S_2, and S_4 will be used in the dynamics, inclusing all nonadibatic couplings between the states. However, states S_0 and S_3, although computed by TeraChem, will *not* be used in the dynamics.

There are times where one may want TeraChem to run with specific job options for only a handful of jobs during a single timestep. For example, one may want to compute electrostatic potential charges (ESP or RESP charges) for the ground state, but do not want this this analysis to run for every excited gradient and nonadiabatic coupling job. Or perhaps, one may need derivatives of excited state dipoles, and only want these to be solved for during one of the excited state jobs. In either case, the user can take advantage of the `tcr_spec_job_opts`, which is a dictionary with keys pertaining to each of the step-jobs ("gradient_0", "gradient_1", "nac_1_2", etc.) and values set to a dictionary of additional options that will update `tcr_job_options` for the keyed jobs only.

For instance, if both of the two examples above were desired, then one would set in the input file

```
tcr_spec_job_opts = {
    'gradient_0': {
        'resp': 'yes'
    },
    'gradient_1': {
        'cisdipolederiv': 'yes'
    }
}.
```

which would compute the ESP and RESP charges on the S_0 state only, as well as CIS dipole derivatives on the S_1 state only.

tcr_spec_job_opts

Type: `dict`
Default: `{}`
Description: Sets job specific jobs options. Each key corresponds to the job during a time step (e.g. "gradient_0") and contains another dictionary of options to pass to TeraChem during this job

The first few time steps of a simulation can also be ran additional different job options on top of those already specified for the rest of the simulation. For example, if one is starting at a geometry near a transition state or far from the molecules equilibrium geometry, extra SCF steps or a different solver may be needed in order to converge the ground state orbital equations. For example,

```
tcr_initial_frame_opts = {
    'n_frames': 3,
    'scf': 'diis+a',
    'maxit': 100
}
```

would add the 'scf' and 'maxit' options to all TeraChem jobs performed for the first 3 simulation time steps. On the forth time step, these options will no longer be used and only the options specified in `tcr_job_options` will be passed to TeraChem.

tcr_initial_frame_opts

Type: `dict`
Default: `{'n_frames': 0}`
Description: Sets job options for the first `n_frames` frames. The remaining keys in the dictionary are used as options passed to Terachem.

4.1 Normal-Mode Inputs

If generating nuclear initial conditions from a Wigner distribution over nuclear coordinates, a frequency calculation in TeraChem must be performed by the user prior to starting this simulation. The locations of the relevant files from the frequency calculations, which should be found the scrathc directory of the TeraChem job, are specified with these options:

fname_tc_xyz

Type: `str`
Default: `None`
Description: Locations of the `Geometry.xyz` coordinate file from a TeraChem frequency calculation. These coordinates are then used as the geometry optimized coordinates within the program.

fname_tc_feo_freq

Type: `str`
Default: `None`
Description: Locations of the `Geometry.frequencies.dat` file from a TeraChem frequency calculation.

fname_tc_redmas

Type: `str`
Default: `None`
Description: Locations of the `Reduced.mass.dat` file from a TeraChem frequency calculation containing the reduced masses of the normal modes.

fname_tc_freq

Type: `str`
Default: `None`
Description: Locations of the `Frequencies.dat` coordinate file from a TeraChem frequency calculation.

4.2 Additional TeraChem Options

tcr_log_jobs

Type: `bool`
Default: `True`
Description: Whether or not to save the TeraChem job results in a self contained `.yaml` file. This includes energies, gradients, NACs, and each line printed to the TeraChem output file for every job.

5 GAMESS Specific Input Options

When GAMESS is selected as the electronic structure driver, jobs can be run with one of two methods. The first of which will create a SLURM submission file and submit an individual job to run GAMESS. The following options can be used to help facilitate the the job submission:

ncpu

Type: `int`
Default: 1
Description: Number of CPUs used to run GAMESS program with.

nnode

Type: `int`
Default: 1
Description: Number of CPU compute nodes to run the GAMESS calculations on.

`partition`

Type: `string`
Default: `None`
Description: Partition name of the compute cluster to run all GAMESS simulations on.

The other method requires the user to specify the locations of a bash script that will be called to run GAMESS. With this method, it is left to the user to specify how computational resources are allocated or what system GAMESS will run on. The script will be called with two arguments, the input and output files, both of which are controllec by the LSC code itself.

`sub_script`

Type: `string`
Default: `None`
Description: Location of the GAMESS submission script to call for every MD time step. The script will be called with two arguments to use with GAMESS, and input file and output file.

5.1 Other GAMESS options

`elab`

Type: `list[int]`
Default: `[1]`
Description: The list of electronic state labels given by GAMESS that are included in the dynamics simulation. For example, "1" indicates "STATE 1" in GAMESS output file, "3" indicates "STATE 3", and so on. The length of this list must be equal to `nel`.

6 References

References

[1] Nist computational chemistry comparison and benchmark database, nist standard reference database number 101, August 2020.

[2] Giuseppe M. J. Barca, Colleen Bertoni, Laura Carrington, Dipayan Datta, Nuwan De Silva, J. Emiliano Deustua, Dmitri G. Fedorov, Jeffrey R. Gour, Anastasia O. Gunina, Emilie Guidez, Taylor Harville, Stephan Irle, Joe Ivanic, Karol Kowalski, Sarom S. Leang, Hui Li, Wei Li, Jesse J. Lutz, Ilias Magoulas, Joani Mato, Vladimir Mironov, Hiroya Nakata, Buu Q. Pham, Piotr Piecuch, David Poole, Spencer R. Pruitt, Alistair P. Rendell, Luke B. Roskop, Klaus Ruedenberg, Tosaporn Sattasathuchana, Michael W. Schmidt, Jun Shen, Lyudmila Slipchenko, Masha Sosonkina, Vaibhav Sundriyal, Ananta Tiwari, Jorge L. Galvez Vallejo, Bryce Westheimer, Marta Wloch, Peng Xu, Federico Zahariev, and Mark S. Gordon. Recent developments in the general atomic and molecular electronic structure system. *The Journal of Chemical Physics*, 152(15):154102, April 2020.

[3] K. Miyazaki and N. Ananth. Nonadiabatic simulations of photoisomerization and dissociation in ethylene using ab initio classical trajectories. *The Journal of Chemical Physics*, 159(12):124110, 09 2023.

[4] Stefan Seritan, Christoph Bannwarth, Bryan S. Fales, Edward G. Hohenstein, Christine M. Isborn, Sara I. L. Kokkila-Schumacher, Xin Li, Fang Liu, Nathan Luehr, James W. Snyder Jr., and et al. Terachem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. *WIREs Computational Molecular Science*, 11(2):e1494, 2021.

[5] Ivan S. Ufimtsev and Todd J. Martínez. Quantum chemistry on graphical processing units. 3. analytical energy gradients, geometry optimization, and first principles molecular dynamics. *Journal of Chemical Theory and Computation*, 5(10):2619–2628, 2009. PMID: 26631777.