# Practical exercises: REKS calculations using GAMESS-US code

# Purpose of the exercises

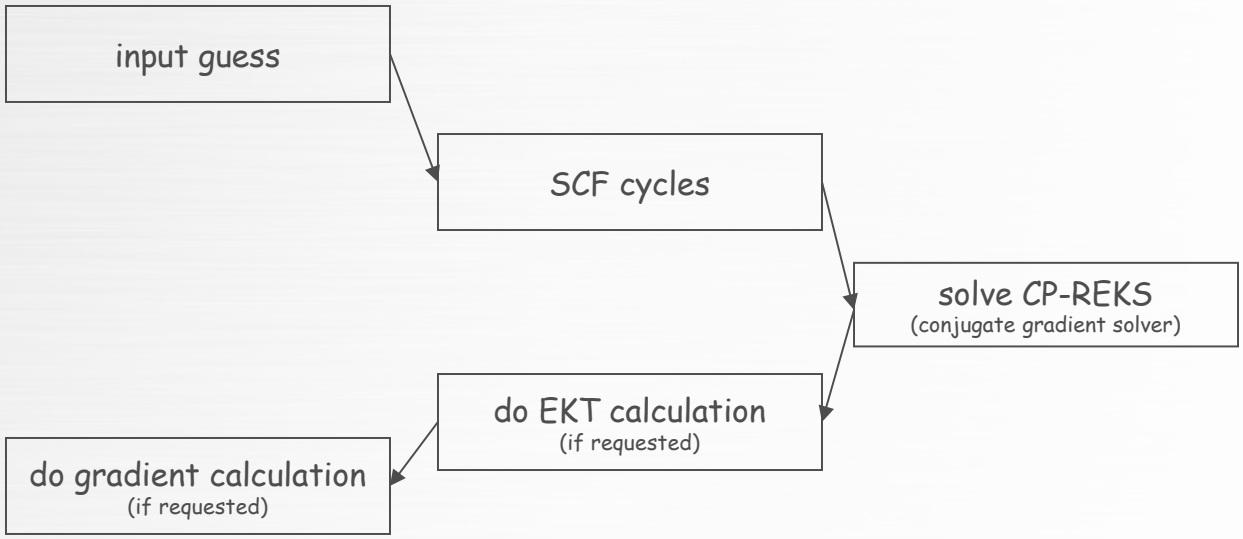
1. What is available in GAMESS-US?
   - REKS/SSR variants implemented in GAMESS
   - Flowchart of the calculation
   - Properties and analysis
2. Input file and keywords
   - Types of calculations in GAMESS
   - Geometry optimization
     - Internal geometry optimizer
     - External geometry optimizer, DL-FIND
     - External NAMD package, pyUNI-xMD
3. Practical calculations for the ground electronic states; TME diradical
   - Geometry optimization of the (meta-)stable conformations
   - Optimization of the minimum energy path (NEB with DL-FIND)
4. Non-adiabatic dynamics simulations of excited state decay in PSB3 cation
   - Optimization of the ground state species; E/Z conformations
   - Optimization of excited state species; S1 minimum
   - Setting up the initial conditions for NAMD; NX init. cond. generator
   - Running the dynamics with pyUNI-xMD

# REKS/SSR availability in GAMESS-US

| | |
|---|---|
| REKS(2,2)/SA-REKS(2,2): | energy and gradient |
| SSR(2,2): | energy and gradient |
| SSR(3,2): | energy and gradient |
| SSR(4,4): | not available yet; a rudimentary implementation in TeraChem |

SA-REKS/SSR implement computation of IPs and EAs via Extended Koopmans' Theorem (EKT)

Typical SA-REKS/SSR computation workflow:



# GAMESS-US Input file

REKS/SSR computation needs an extra block in the input file

```
$contrl scftyp=reks runtyp=gradient dfttyp=bhhlyp icharg=0 maxit=200 mult=1 $end
$dft sg1=.true. $end
$scf nconv=6 npunch=2 $end
$reks
 rexType=2
 rexTarget=2
wpps=0.50
rexShift=0.4
rexDIIS=no
rexEKT=yes
EKTEA=yes
$end
 $basis gbasis=n31 ngauss=6 ndfunc=1 $end
 $guess guess=moread norb=<your number of orbitals> $end
 $system timlim=999999100 mwords=<your memory> $end
 $data
.
.
.
```

- requests REKS/SSR calculation
- new block for REKS/SSR
- 0/1/2 – SA-REKS / SSR(2,2) /SSR(3,2)
- 1/2 – $S_0$ / $S_1$
- SA weighting factor; if = 1, then single state REKS calculation
- level shift; used to stabilize SCF convergence; good values ~0.2-0.5
- yes/no – use /don't use DIIS acceleration
- yes/no – calculate/don't calculate the IP's from EKT
- yes/no – calculate/don't calculate the EA's from EKT
- REKS needs initial guess from previous RKS calculation; the default guess is very bad!

Resources:

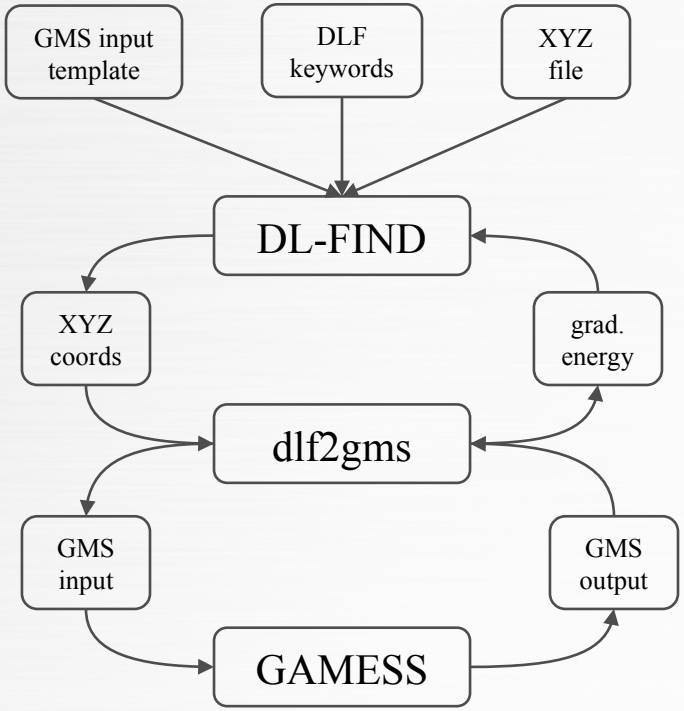
reks_gamess_input.txt, reks_gamess_manual, TopCurrChem_368_97.pdf, WIREs_5_146.pdf

# Standalone DL-FIND flowchart

Standalone DL-FIND code; not integrated in GAMESS-US

Three input files are needed:

- GAMESS input file; used as a template for creating input files at optimization steps
- XYZ file with the starting geometry
- DL-FIND input file; collects all the keywords




Example of use: ./find.x input.dlf > output.dlf

# DL-FIND keywords

| NAME | VALUE | COMMENT |
|---|---|---|
| job | minimize | # job type: "minimize", "neb_frozen", "neb_free“ |
| coordinates | tme-00.xyz | # coordinates in XYZ format; for NEB, may contain more than one image |
| coord_type | dlc | # type of coordinates; can be "cart" or "dlc"; other coordinates are NYI |
| interface | gamess | # what QC program to use; it's GAMESS-US only, for the moment |
| inputfile | tme-00-ssr-bhhlyp-6-31gs.inp | # name of the QC input file; contain all GAMESS keywords |
| tolerance_g | 4.5e-4 | # tolerance for the gradient (default value) |
| tolerance_e | 1.0e-6 | # tolerance for the energy (default value) |
| trust_radius | 0.5 | # trust radius for the optimization (default value) |
| neb_constant | 0.02 | # neb force constant (default value) |
| neb_images | 17 | # the number of neb images; N+2, where N is the desired number of images |
| maxcycle | 100 | # max number of optimization cycles |
| restart | no | # (yes/no) whether it's restart job (yes) or not (no); default "no" |
| lbfgs_mem | 100 | # how many steps keep in the L-BFGS memory (default value = 3N-6) |
| printlevel | 4 | # print level: 0-nothing, 1-verbose, 4-very verbose |
| climb_img | no | # (yes/no) if "no", do not spawn the climbing image |
| tmp_input | tme_00_gms.inp | # temporary GAMESS input file that will be created from "inputfile"; default = temp_gms.inp |
| vec_update | yes | # (yes/no) whether to update/not update the QC eigenvectors (in the tmp_input file) during the geometry search |
| exe_script | gmsrun | # set the name of the QC execution script |
| parameters_script | | # QC execution script parameters. Default: no parameters |

# Environment setup

Set up the environment; add in your .bashrc

```
#python env
module use /projects/academic/cyberwksp21/Modules
module load jupyter
eval "$(/projects/academic/cyberwksp21/SOFTWARE/Conda/bin/conda shell.bash hook)"
conda activate libra2
unset LD_LIBRARY_PATH

#Intel compilers
module load intel/20.2
module load intel-mpi/2020.2
module load mkl/2020.2

#set scratch directory
export SCRATCH=$HOME/work-dir

# GAMESS-US env
export GMSSCR=$SCRATCH/gam-scr
export GMSPATH=/projects/academic/cyberwksp21/Software/gamess-2018/qmmm-reks-2018-6.3
export GMSVER=01

# DL-FIND env
export DLFPATH=/projects/academic/cyberwksp21/Software/dl-find-standalone

#visualization tools
module load molden/5.9
module load cuda/5.5.22
module load vmd/v1.9.2

#pyUNI-xMD env
export PYUNIXMDHOME=/projects/academic/cyberwksp21/Software/pyUNI-xMD/unixmd-gamess
export PYTHONPATH=$PYUNIXMDHOME/src:$PYUNIXMDHOME/util:$PYTHONPATH
export UXMD2GMS=/projects/academic/cyberwksp21/Software/pyUNI-xMD/uxmd2gms
```

# Environment setup

In your $HOME/bin, add two scripts:

gmsrun:

```
#!/bin/bash
inpfile="$1"
datfile=$(basename -- "$inpfile")
datfile="${datfile%.*}"
datfile="$datfile.dat"
if [ -e $GMSSCR/$datfile ]
then
        rm -f $GMSSCR/$datfile
fi
$GMSPATH/rungms $inpfile 01 16 1
```

gmsrun calls GAMESS-US

it uses 16 cores

"-n 16" should be set, when submitting the jobs

example:

gmsrun your_input.inp > your_output.out

sbatchwrap.sh:

```
#!/bin/bash
$@
```

sbatchwrap.sh is used to pass

arguments to sbatch; e.g.,

sbatch -N1 -n 16 -p <queue> -o output.dlf sbatchwrap.sh $DLFPATH/find.x input.dlf

Make the scripts executable.

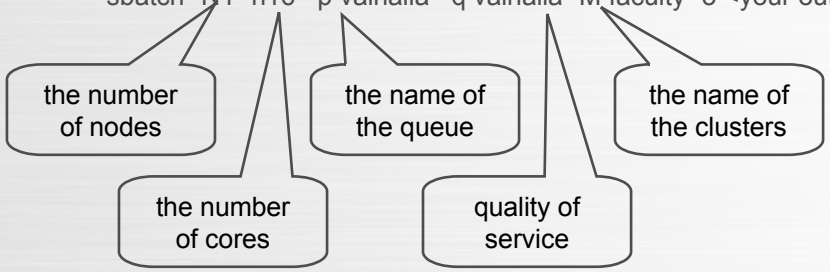
Create a softlink:

ln –s /projects/academic/cyberwksp21/Software/gamess-2018/qmmm-reks-2018-6.3/rungms ~/bin/rungms

# How to submit jobs to SLURM

Submit from the command line:

sbatch -N1 -n16 –p valhalla –q valhalla -M faculty -o <your output> sbatchwrap.sh <your command>




Examples:

- submit a DL-FIND optimization

  sbatch -N1 -n16 –p valhalla –q valhalla -M faculty -o **output.dlf** sbatchwrap.sh **$DLFPATH/find.x input.dlf**

- submit a GAMESS calculation

  sbatch -N1 -n16 –p valhalla –q valhalla -M faculty -o **psb3.out** sbatchwrap.sh **gmsrun psb3.inp**

- submit a pyUNI-xMD run

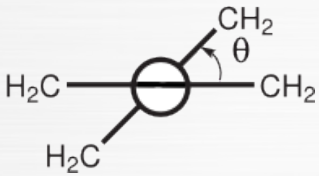
  sbatch -N1 -n16 –p valhalla –q valhalla -M faculty -o **log_psb3** sbatchwrap.sh **python3 run_psb3.py**

# Applications of REKS: Tetramethyleneethane diradical

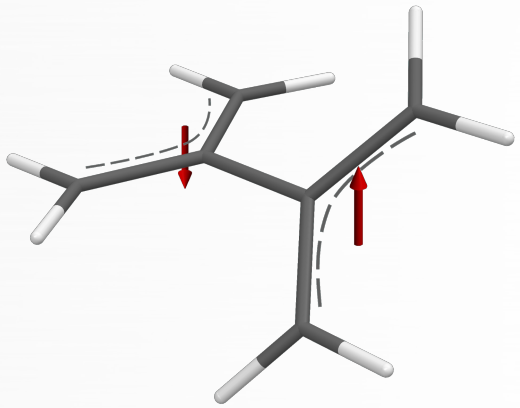
used in organic synthesis; as a ligand in metal complexes…

resonance structures

frontier orbitals

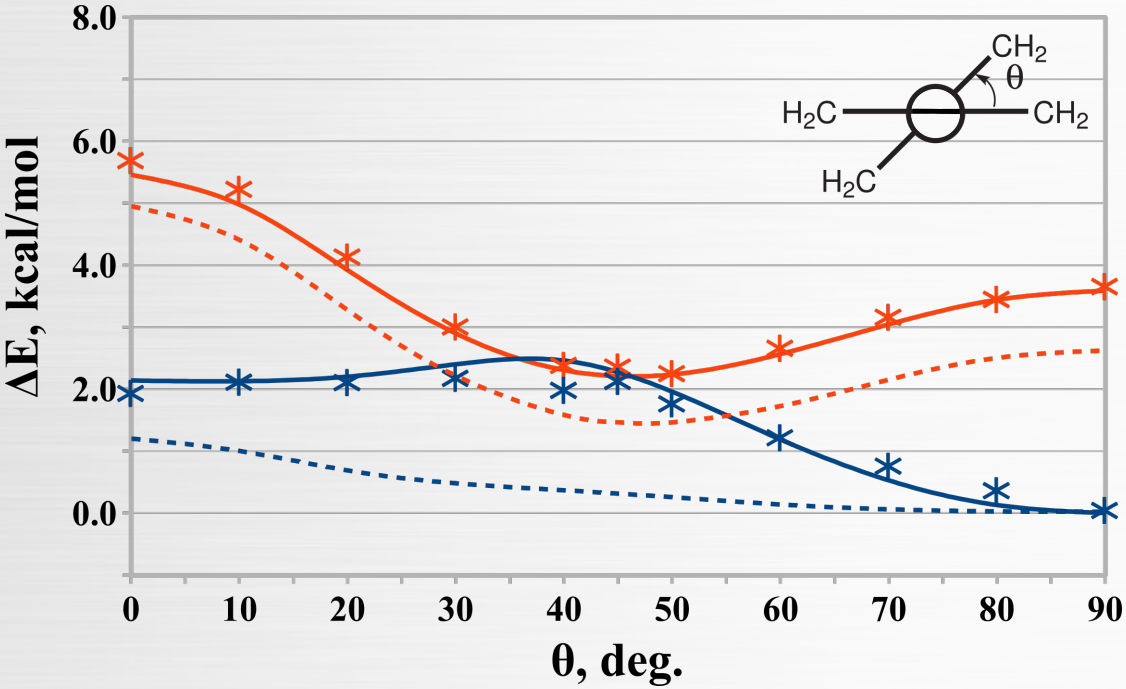



TME: singlet or triplet?



| | | |
|---|---|---|
| Matrix isolated TME: | triplet EPR signal; linear CW plot | (Dowd, 1970, 1986) |
| NIPE spectroscopy ($TME^{\cdot-}$): | singlet below (ca. 3 kcal/mol) triplet | (Clifford et al., 1998) |
| Theory (CASSCF,CI, etc..): | singlet always below triplet | (Borden et al., 1987) |

# Applications of REKS: Tetramethyleneethane diradical




REKS: Filatov&Shaik, 1999

CASPT2: Caballol et al., 2000

QMC: Jordan et al., 2013
Barborini&Coccia, 2015

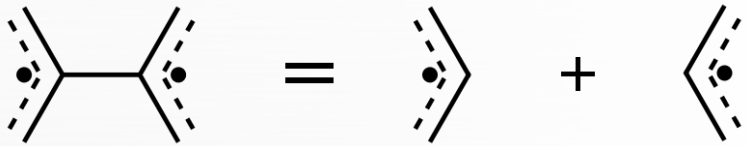
Singlet is a global energy minimum

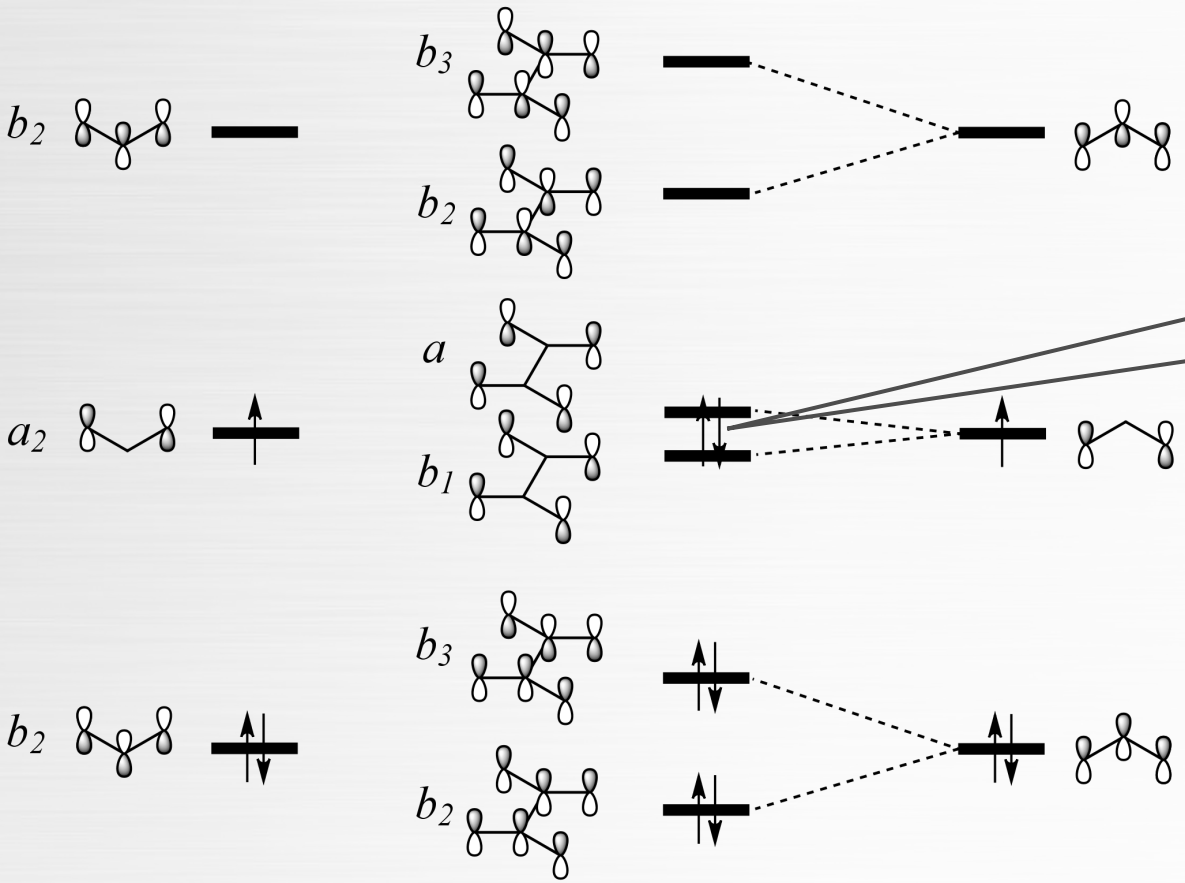
Triplet is *meta-stable* at intermediate $\theta$ (trapping, slow relaxation)

Reconciles theory and experiment (Lineberger&Borden, 2013)

# TME: electronic structure

TME = 2 allyl radicals





# TME: REKS calculations

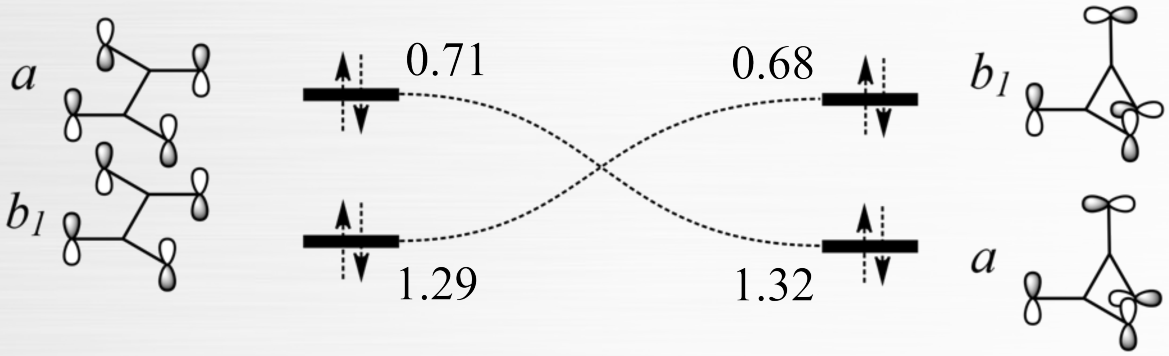
1. Set up single point REKS calculation
    - set up the geometry of planar TME
    - run an RKS calculation
    - use the RKS eigenvectors for REKS guess
    - run a REKS calculation; use SSR(2,2) with $\mathrm{WPPS} = 0.95$
2. Optimize the geometries; use DL-FIND
    - set up a DL-FIND geometry search
    - use the REKS eigenvectors from the single point calc.
    - optimize the geometry of planar TME
    - set up the geometry of 90° twisted conformation
    - optimize the geometry; use eigenvectors from the previous calc.
3. Minimum energy path; use the NEB optimization in DL-FIND
    - set up the NEB search
        - use planar and 90° twisted geometries
        - run geodesic_interpolate to set intermediate geometries
        - use the interpolated geometries to start the NEB search
    - run the NEB optimization in DL-FIND
    - use the available visualization tools to visualize the NEB path

# TME: REKS calculations (continued…)

Repeat the calculations for 90° twisted TME

Is this conformation higher or lower in energy than the planar?

What is the ordering of populations of $b_1$ and $a$ fractionally occupied orbitals?




Orbital populations swap. At an intermediate angle they may equalize.
Inspect the GAMESS output files in ./rundir. Check the orbital populations.
Use your lovely molecular visualization tool to visualize the orbitals.
From output.dlf, copy the final energies and visualize the NEB path.

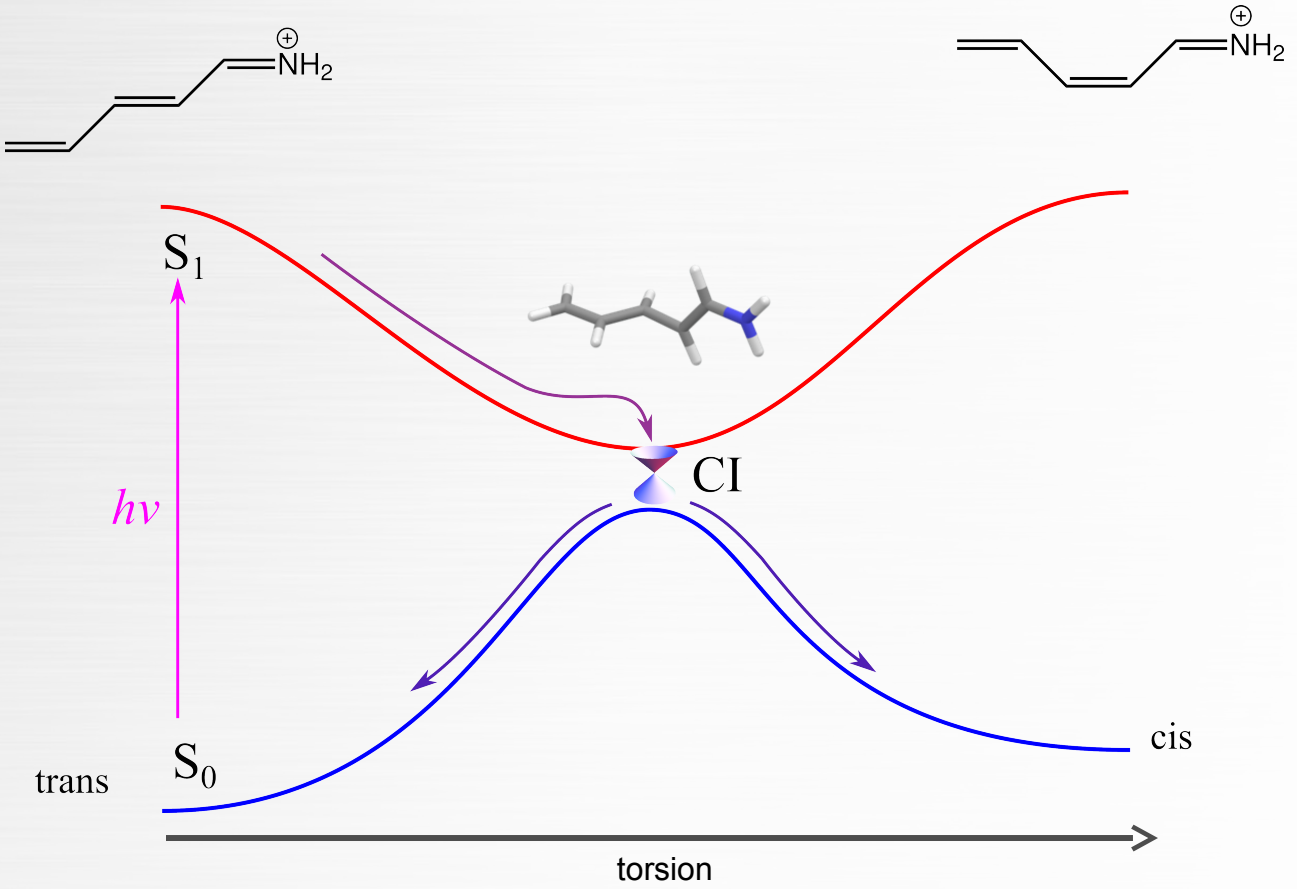
# PSB3: Dynamics of excited state decay

JCTC Journal of Chemical Theory and Computation — Article
Cite This: J. Chem. Theory Comput. 2018, 14, 4499−4512 — pubs.acs.org/JCTC

**Direct Nonadiabatic Dynamics by Mixed Quantum-Classical Formalism Connected with Ensemble Density Functional Theory Method: Application to *trans*-Penta-2,4-dieniminium Cation**

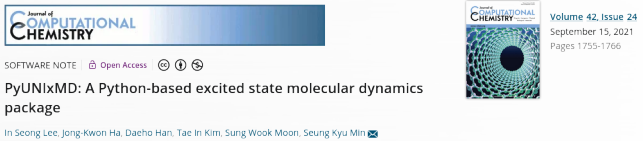
Michael Filatov,* Seung Kyu Min,* and Kwang S. Kim

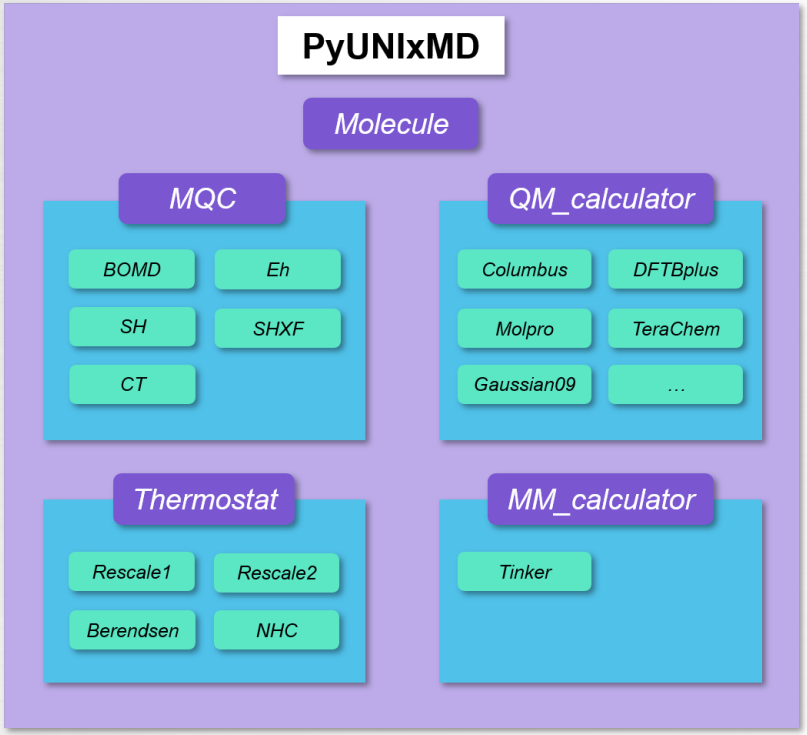
| | SSR | MS-CASPT2 | CASSCF |
|---|---|---|---|
| cis:trans | 63:37 | 79:21 | 54:46 |



# pyUNI-xMD program

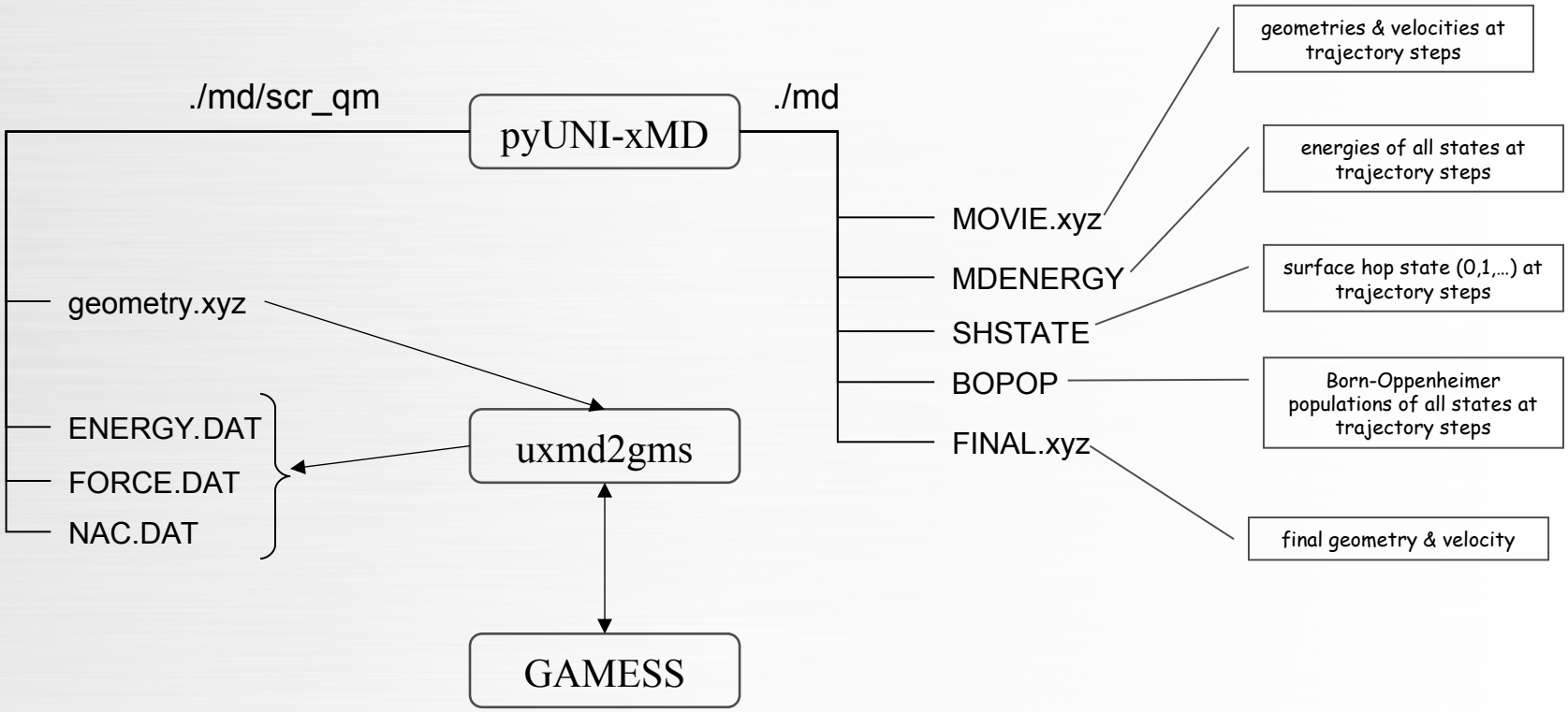
URL: https://jkha-rtd-test.readthedocs.io/en/latest/overview.html




+ GAMESS-US / SSR

# pyUNI-xMD / SSR interface




All log files of GAMESS calculations at the trajectory points can be saved in ./qm_log

# pyUNI-xMD / SSR interface

How to start pyUNI-xMD dynamics run: python3 run.py >& log

Pre-requisites:

- GAMESS input file. This file will be used as a template. It should contain all relevant keywords
- init.xyz file. This file should contain the coordinates (Å) and velocities (a.u.) of all atoms
- run.py script. This script specifies all the parameters of the simulation.

Sample run.py script:

```
from molecule import Molecule
import qm, mqc
from thermostat import *
from misc import data

with open ("./init.xyz", "r") as f:
    geom = f.read()

mol = Molecule(geometry=geom, nstates=2, charge=1., unit_pos="angs")

qm = qm.gamess.SSR(molecule=mol, \
    Baeck_An = "yes", \
    gam_scr = "$HOME/work-dir/gam-scr", \
    template_file = "GMS_input_psb3.inp", \
    tmp_file_name = "GMS_run_psb3_010.tmp", \
    qm_path="$UXMD2GMS/", \
    nthreads=16, \
    version="01")

md = mqc.SHXF(molecule=mol, nsteps=1250, dt=0.25, istate=1, elec_object="density", rho_threshold=0.02, \
    sigma=0.2, hop_rescale="momentum", hop_reject="keep", l_xf1d=False, l_econs_state=True, \
    unit_dt="fs", verbosity=2)

md.run(qm=qm, output_dir="./", l_save_scr=True, l_save_qm_log=True, l_save_mm_log=False)
```

import pyUNI-xMD modules

import initial geometry & velocities

define molecule

parameters of the SSR calculation and temporary files; more in $PYUNIXMDHOME/src/qm/gamess/ssr.py

parameters of MD simulation

parameters of MD run

# PSB3: The plan

1. Optimize the geometries of the main species
    - the ground state geometry of trans-PSB3
    - the ground state geometry of cis-PSB3
    - the excited ($S_1$) state geometry
    - use these species to set reference for the dynamics runs
2. Set up the initial conditions
    - compute the vibrational frequencies in the $S_0$ trans-PSB geometry
    - use Newton-X to generate ~500 initial conditions
    - select the initial conditions to your liking
    - set the init.xyz file (geom. & velocities)
    - set appropriate keywords in the run.py script
3. Run the simulations
    - start python3 run.py in the queuing system
    - relax and enjoy your time
    - from time to time, monitor the progress of your simulation
        - use Avogadro, or VMD, or Molden to visualize the trajectory
    - when all is finished, analyze the results
        - hop time
        - final conformation

# That was all!