The SchNarc software for machine learning based surface hopping molecular dynamics simulations

Julia Westermayr, Maximilian Tiefenbacher, Brigitta Bachmair, Johann Dorn, Philipp Marquetand

September 20, 2021

University of Vienna
Institute of Theoretical Chemistry
Währinger Straße 17, 1090 Vienna
Austria

Contents

1 Case Study: Fulvene

In this section, you will learn how to use SchNarc on the example of the excited state dynamics of fulvene using reference data computed with MOLPRO[1] and SA(2)-CASSCF(6,6)/6-31G* – in accordance to Ref. [2]. The whole tutorial is closely related to the SHARC tutorial. First, follow the instructions at https://github.com/schnarc/SchNarc and install SchNarc, [3] SHARC [4] and SchNetPack. [5, 6]

All data will be provided and you will not have to do any reference calcluations.

1.1 Initial training set generation

In this section, it is explained how to generate a training set in a data base format provided within the atomic simulation environment (ASE) [7] for later use in SchNarc.

As an example, we will start with 100 geometries saved in xyz-format for which you will be provided with reference calculations. Go into the folder "1_TrainingSet/". You can find several folders starting with "ICOND" in the "InitialConditions" that have been created with SHARC using Wigner sampling. [8, 9] In addition, the gradient, nonadiabatic coupling, and phase/overlap keywords have been added to the QM.in files so that SHARC requests these properties for computation with MOLPRO. [1] Note that overlaps are only needed for phasecorrection. A reference wavefunction (wf.1) has to be provided in each folder. Therefore, we copy the SAVE folder into each directory. This is specified in the QM.in file using the keyword "phases" and "savedir ./SAVE/".

We have provided a jupyter notebook named "GenerateInitialTrainingset.ipynb". Go through this script and see how to convert the QM.out files to an initial training set. In addition it is shown how to convert trajectory data into a data base.

Format of data

Below, you can see examples of how the data are parsed considering 2 singlet and 2 triplet states. Although we only consider 2 singlets in fulvene, we extend this section to 2 triplets as triplet states can also be described with SchNarc. The molecular Coulomb Hamiltonian (MCH) [10, 11, 12] can be seen in Fig. 1. The total number of states is $N_{states} = N_{singlets} + 3 * N_{triplets}$ with N_{states}, $N_{singlets}$, and $N_{triplets}$ denoting the number of states in total,

the number of singlets and the number of triplets, respectively. As it is visible, the triplet states (red) are parsed only once for the energies. In case the data contain spin-orbit couplings (SOCs), two values (dependent on the quantum chemistry program SOCs can be complex numbers) are saved in the data base with the indices in the image indicating the position of the value in the array learned with SchNarc.

H	S0	S1	T1	T2	T1	T2	T1	T2
S0	E1	1,2	3,4	5,6	7,8	9,10	11,12	13,14
S1		E2	15,16	17,18	19,20	21,22	23,24	25,26
T1			E3	27,28	29,30	31,32	33,34	35,36
T2				E4	37,38	39,40	41,42	43,44
T1					E5	45,46	47,48	49,50
T2						E6	51,52	53,54
T1							E7	55,56
T2								E8

Figure 1: Hamiltonian matrix in the Molecular Coulomb Hamiltonian (MCH) [12, 10, 11] basis. The matrix is shown as outputted by SHARC and most other quantum chemistry programs. 2 singlet and 2 triplet states are considered in this example. Note that triplet states are shown three times as they are triple degenerated in the MCH basis.

Figure 2 shows a matrix with each entry containing a vector of nonadiabatic coupling vectors (NACs). The blue entries show which vectors are saved in the data base and are trained. As you can see, NACs between the same states are zero. The same is true for NACs between states of different spin-multiplicity. We further make use of the following relation:

$$\mathbf{NAC}_{ij} = -\mathbf{NAC}_{ji}, \tag{1}$$

and thus only learn one vector. The dipole moment vectors are shown in Fig. 3. This format is in accordance with SHARC. One matrix exists for each direction, i.e., x, y, and z. The permanent dipole moments are on the diagonal, while the transition dipole moments are on the off-diagonal. As you can see, dipoles only exist between states of same spin-multiplicity or of same magnetic moment. In the case of two singlet and two triplet states, 6 dipole vectors are stored, i.e., 3 permanent dipole moment vectors and 3 transition dipole moment vectors. The entry in the matrix in Fig. 3 indicates the position of each value in the vector in the data base. All trained values in this way will be correctly parsed to SHARC. Note that the data set also requires metadata with the entries *n_ singlets* and *n_ triplets*, see the jupyter notebook for an example.

NACs	S0	S1	T1	T2	T1	T2	T1	T2
S0	0	NAC01	0	0	0	0	0	0
S1		0	0	0	0	0	0	0
T1			0	NAC12	0	NAC12	0	NAC12
T2				0	-NAC12	0	-NAC12	0
T1					0	NAC12	0	NAC12
T2						0	-NAC12	0
T1							0	NAC12
T2								0

Figure 2: Nonadiabatic coupling vectors (NACs) indicated in matrix notation between different singlet and triplet states.

Dipoles	S0	S1	T1	T2	T1	T2	T1	T2
S0	1	2	0	0	0	0	0	0
S1		3	0	0	0	0	0	0
T1			4	5	0	0	0	0
T2				6	0	0	0	0
T1					4	5	0	0
T2						6	0	0
T1							4	5
T2								6

Figure 3: An example of how dipole values are written in SHARC format in one direction. This matrix is needed in all three dimensions.

Phase correction

The jupyter notebook further shows an example how to carry out phase correction. For more details on phase correction, we refer to Refs. [13, 14]. In addition to or instead of phase correction, a phase free training algorithm can be applied, for details see Ref. [3] and the next section. Note that phases will always be saved in the data base if overlap calculations are carried out and are provided in the QM.out file. For finding the phase, we use a threshold of $|0.5|$. If the absolute value of an overlap is smaller than 0.5, a proper phase vector cannot be found and phases cannot be corrected. In this case, interpolation between the geometry that should be added to the training set and the initial geometry is required. You can see an example of how to interpolate geometries in the jupyter notebook used above. The functions to interpolate between structures are implemented in SchNarc.

The steps that are needed for interpolation may vary with the systems

size and deviation to the reference structure. The closer the new geometry is to the reference geometry, the less steps are required for interpolation. In the example provided, we use 30 interpolation steps to ensure proper phase correction for every step. Note that the phases of the final structure can only be corrected when a phasevector for every structure in between can be found. The phasevectors have to be multiplied in a consecutive fashion. Consequently, calculations cannot be carried out in parallel, but have to be done in serial and the SAVE folder from the previous calculation has to be copied into the folder of the current calculation. The file "submit_qmcalc.sh" shows an example of how a script can look like to execute this task.

1.2 Generating a data base from trajectory-files

In addition, reference trajectories carried out with SHARC can be useful to generate an initial training set for SchNarc. In the end of the notebook, you can also find an example of how to use SchNarc to extract data from trajectory-files of SHARC.

1.3 Training of SchNarc

SchNarc, i.e., SchNet for excited states, [3] is trained very similarly to SchNet [5, 6]. The molecular representation of a molecule in its chemical and structural environment stays the same. Thus the properties are represented in an atom-wise fashion and atoms are described in their chemical and structural environment within a specified cutoff region. There are, however, a few differences related to the number of excited states and properties involved. The necessary input files and additional parameters to set during training will be discussed in the following. In case you do not specify anything, the default values will be taken for training.

Find out how to train a SchNarc model by going to the folder **2_Training** and opening the jupyter notebook. Alternatively, you can execute the scripts provided in the folder and read the README.md for more information. With the help function (`--help option`), you can find out about all the possible parameters and arguments to set for training SchNarc.

Below you will find more information on the input files needed to train a model and on the arguments you can set.

Tradeoffs for different properties

In addition to the ground state potential, also the excited state potentials are treated within one SchNet model. Additionally, it is possible to train

the nonadiabatic couplings, spin-orbit couplings, and (transition) dipole moments, if they are included in the database. In order to specify the properties and numbers of states that need to be trained, you need a separate file. This includes the properties to be trained, the corresponding trade-offs, i.e. the weights of each property within the loss function, and the number of singlet, doublet, triplet, and quartet states included. An example is given below in Fig. 4.

```
energy:   1.0   2 0 2 0
forces:   1.0   2 0 2 0
dipoles:  0.001 2 0 2 0
nacs:     0.001 2 0 2 0
socs:     100   2 0 2 0
```

Figure 4: An example of a tradeoff file that specifies the properties to be trained (first column) along with the corresponding tradeoffs (second column) and the number of singlet, doublet, triplet, and quartet states (third - sixth columns, respectively).

Practical hint: It is advisable to check the size of the mean of each property or the mean average error (MAE) / root mean squared error (RMSE) for each property after a few steps of training. The trade-offs can then be chosen to make the properties equally large to weight them equally within the loss function. This tradeoff-file can be given to the model via setting the argument "−−tradeoff tradeoffs.yaml". If you don't want to train a certain property which resides in your database just don't include the keyword and the corresponding line.

Additional training arguments

Since the excited-state properties, i.e. nonadiabatic couplings, spin-orbit couplings, and transition dipole moments, suffer from effects of the arbitrary phase of the wave function (see Refs [15, 13] for more information). Therefore, for training, they either need to be phase corrected in advance (see previous section in the tutorial) or trained in a phase-free manner. [3] The latter possibility can be defined by the argument $--min\_loss$ and is the most important argument that differs from the usual way of training with SchNet. A summary of the arguments to specify that are important are given in Table 1. Note, that a comprehensive and complete parameter selection, such as the number of nodes, hidden layers, the learning rate, etc. can be found in the Tutorial of SchNet.

Argument	Key	Example/Explanation
$--split$	int int	splits the data set into training (first number), validation (second number), and test (rest of data points) set
$--cutoff$	int	cutoff-region for the description of an atom (default: 5 Bohr)
$--batch\_size$	int	batch size; in case you train on a GPU a too large batch size can lead to memory issues
$--tradeoff$	str	tradeoffs.yaml, see Fig. 4
$--min\_loss$	/	Trains the excited-state properties in a phase-free manner
$--real\_socs$	/	Puts a mask onto the imaginary values of socs holds only for some quantum chemistry programs
$--cuda$	/	train on a GPU
$--logger$	str	set to tensorboard if you want to check your models with tensorflow; otherwise don't set anything
$--inverse\_nacs$	int	Trains "smooth" NACs by multiplication with corresponding ΔE_{ij} for training and by division for prediction *Hint:* more stable when training with reference energies (int=1) (int=2 means training with NN energies)
$--order$	/	trains properties by the states according to energy this only works for the adiabatic basis Attention: only implemented for singlet states

Table 1: Arguments and options to specify for training that differ to or are not available in SchNet.

1.4 Validation of trained models

After training, SchNet writes the best model (termed "best_model") into the folder you have specified for training. You can validate your model by computing the MAE or RMSE on your training, validation, and test set. For this, carry out "eval_cpu.sh" if you use a CPU or "eval_gpu.sh" if you have a GPU. The split option can carry the strings "train", "validation", "test", and "all". The error will be computed on the training set, validation set, test set or on all data points, respectively. The error is reported in the Training folder in the file "evaluation.csv". The error is not the phase-free error! So if you train phase-free, you should recompute the error on phase-free properties.

1.5 Using models for prediction

The models can also be used to predict the properties of new geometries. The predictions are saved in the training folder and are named "predictions.npz". See how to generate a database with SHARC from Wigner sampling using the file "get_initial_conditions.sh". Execute the files "pred_cpu.sh" and "pred_gpu.sh" to predict the properties of the geometries. Also see the README.md file or the jupyter notebook.

1.6 Adaptive Sampling

SchNarc is a combination of SchNet for excited states and the molecular dynamics program SHARC (Surface hopping including ARbitrary Couplings). In order to run surface hopping dynamics with SchNarc, you need an initial conditions file, e.g. from a Wigner distribution [9], that you can generate with the tools of SHARC (see Ref. [4]). This file can be generated with the quantum chemistry reference method, or alternatively, with your SchNet models if the potentials are converged and accurate. Further, different to SHARC, you need to specify the data base used for training and the "best_model", that contains the final SchNet model for excited states. This is saved in the folder, you specified for training.

To carry out adaptive sampling, you need at least 2 trained ML models. You need to specify both paths and additionally the thresholds you want to use for terminating the trajectories.

Go into the folder "3_AdaptiveSampling" and check the jupyter notebook or read the README.md to find an example how to do adaptive sampling with SchNarc. Below, you can find additional explanations on the arguments that are used to run SchNarc-MD.

Arguments to set when running SchNarc-MD

The model paths are added by setting: "–modelpaths MODEL2 MODEL3 ...". MODEL2 and MODEL3 should point to the training directory. You can use as many models as you wish to use. For propagation, the mean of the predictions of the models is used.

Thresholds for energies, forces, dipoles, nacs, and socs, respectively, are indicated with "–thresholds xx xx xx xx xx" adding five arguments. The values are given in atomic units as used by SHARC. Set "−−adaptive" to enable adaptive sampling.

The argument "print_uncertainty" can be added to print the uncertainty between the models each time step in the file "NN.log". In this way, you can

Argument	Key	Example/Explanation
$--adaptive$	/	needs to be set for adaptive sampling
$--thresholds$	float [x5]	threshold for each property to terminate trajectories the numbers are for energies [a.u.], forces [a.u.], dipoles [a.u.], nacs [a.u.], and socs [a.u.]
$--print\_uncertainty$	/	can be set to print the model errors
$--modelpaths$	str	should point to the directories of the models used for adaptive sampling
$--cuda$	/	run on a GPU
$--logger$	str	set to tensorboard if you want to check your models with tensorflow; otherwise don't set anything
$--order$	/	use this when you trained with this argument

Table 2: Arguments and options to specify for running adaptive sampling with SchNarc.

get a feeling of optimal thresholds for each property you wish to predict.

In order to add meaningful/useful geometries to the training set/database, the script in the appendix can be used to interpolate geometries between a start and end point geometry (scans). For non-linear interpolation use the script "interpolation.py" in the $SCHNARC folder. See the initial training set generation for more information. Usage:

1.7 Running surface hopping dynamics with SchNarc

For final production runs, several choices exist. You can stay in the adaptive sampling run, but can also use other arguments to make simulations more efficient. The arguments will be explained below. Further, you have the choice to approximate nonadiabatic couplings. Therefore you have to set "$--hessian\ --nac\_approx\ 1\ X\ Y$". This means you will use the hessian approximation and NAC-method number 1 (the only one approved in SchNarc) and thresholds for the energy gaps between singlet-singlet (X) and triplet-triplet states (Y) to compute approximated NACs, respectively. A.u. are used.

Go into the Dynamics folder and follow the instructions in the README.md to find out how to carry out dynamics with SchNarc.

Argument	Key	Example/Explanation
$--hessian$	/	needs to be set for NAC approximation
$--nac\_approx$	1 float float	floats specify the energy gaps to compute approximated NACs first value is for singlets, second for triplets units: a.u.
$--emodel2$	str	path for a second model to predict and compare energies during dynamics
$--nacmodel$	str	path to a separate model trained on NACs
$--socmodel$	str	spath to a separate model trained on SOCs
$--order$	/	use this when you trained with this argument

Table 3: Arguments and options to specify for running surface hopping dynamics with SchNarc.

References

[1] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, Molpro: a general-purpose quantum chemistry program package, *WIREs Comput. Mol. Sci.*, **2**, 242–253 (2012).

[2] L. M. Ibele, B. F. E. Curchod, A molecular perspective on Tully models for nonadiabatic dynamics, *Phys. Chem. Chem. Phys.*, **22**, 15183–15196 (2020).

[3] J. Westermayr, M. Gastegger, P. Marquetand, Combining SchNet and SHARC: The SchNarc Machine Learning Approach for Excited-State Dynamics, *J. Phys. Chem. Lett.*, **11**, 3828–3834 (2020).

[4] S. Mai, M. Richter, M. Ruckenbauer, M. Oppel, P. Marquetand, L. González, SHARC2.0: Surface Hopping Including ARbitrary Couplings – Program Package for Non-Adiabatic Dynamics, sharc-md.org (2018).

[5] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, SchNet – A Deep Learning Architecture for Molecules and Materials, *J. Chem. Phys.*, **148**, 241722 (2018).

[6] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, K.-R. Müller, SchNetPack: A Deep Learning Toolbox For Atomistic Systems, *J. Chem. Theory Comput.*, **15**, 448–455 (2019).

[7] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K. W. Jacobsen, The atomic simulation environment—a Python library for working with atoms, *J. Phys. Condens. Matter*, **29**, 273002 (2017).

[8] E. Wigner, E. Guth, Gruppentheorie und ihre Anwendung auf die Quantenmechanik der Atomspektren, *Monatsh. Math.*, **39**, A51–A51 (1932).

[9] E. Wigner, On The Quantum Correction for Thermodynamic Equilibrium, *Phys. Rev.*, **40**, 749–750 (1932).

[10] S. Mai, P. Marquetand, L. González, A General Method to Describe Intersystem Crossing Dynamics in Trajectory Surface Hopping, *Int. J. Quantum Chem.*, **115**, 1215–1231 (2015).

[11] S. Mai, P. Marquetand, L. González, Nonadiabatic Dynamics: The SHARC Approach, *WIREs Comput. Mol. Sci.*, **8**, e1370 (2018).

[12] M. Richter, P. Marquetand, J. González-Vázquez, I. Sola, L. González, SHARC: Ab Initio Molecular Dynamics with Surface Hopping in the Adiabatic Representation Including Arbitrary couplings, *J. Chem. Theory Comput.*, **7**, 1253–1258 (2011).

[13] J. Westermayr, M. Gastegger, M. F. S. J. Menger, S. Mai, L. González, P. Marquetand, Machine Learning Enables Long Time Scale Molecular Photodynamics Simulations, *Chem. Sci.*, **10**, 8100–8107 (2019).

[14] F. Plasser, S. Gómez, M. F. S. J. Menger, S. Mai, L. González, Highly Efficient Surface Hopping Dynamics using a Linear Vibronic Coupling Model, *Phys. Chem. Chem. Phys.*, **21**, 57–69 (2019).

[15] A. V. Akimov, A Simple Phase correction Makes a Big Difference in Nonadiabatic Molecular Dynamics, *J. Phys. Chem. Lett.*, **9**, 6096–6102 (2018).