

Newton-X User's Guide

Light and Molecules Group

February 27, 2023

Contents

1	Presentation	3
1.1	Newton-X	3
1.2	Installation	4
1.3	Bug reports	4
1.4	Contributions	4
2	Quickstart	6
2.1	Preparation	6
2.2	Running the program	6
2.2.1	Analytical model: Spin-Boson Hamiltonian	6
2.2.2	TDDFT: Gaussian	7
2.3	Outputs	8
2.3.1	HDF5 output	9
3	Main program	10
3.1	Overview	10
3.2	Input overview	10
3.2.1	Mandatory input files	10
3.2.2	Optional inputs	11
3.2.3	Input generation	11
3.3	Running dynamics	11
3.3.1	Some tips	11
3.3.2	Following the dynamics	12
3.4	Restarting dynamics	12
3.5	<code>nxconfig</code> options	13
4	Electronic structure interfaces	17
4.1	Available methods	17
4.1.1	Built-in analytical models	17
4.1.2	Interface to external programs	17

4.2	Spin-Boson Hamiltonian	18
4.2.1	Model description	18
4.2.2	Parameters specification	19
4.2.3	<code>sbh</code> options	19
4.3	Collection of 1D-models	20
4.3.1	Model description	20
4.3.2	Parameters specification	21
4.3.3	<code>onedim_model</code> options	21
4.4	Columbus	22
4.4.1	Setup	22
4.4.2	Running dynamics	22
4.4.3	<code>columbus</code> options	23
4.5	Turbomole	24
4.5.1	Setup	24
4.5.2	Running computations	24
4.5.3	<code>turbomole</code> options	25
4.6	Gaussian 16	26
4.6.1	Setup	26
4.6.2	Running dynamics	26
4.6.3	<code>gaussian</code> options	26
4.7	Orca	27
4.7.1	Setup	27
4.7.2	Running dynamics	27
4.7.3	<code>orca</code> options	28
4.8	Complex Surface Fewest-Switch Surface Hopping (CS-FSSH)	28
4.8.1	Analytical models	28
4.8.2	Integration with Columbus	29
5	Surface hopping	29
5.1	Some definitions	29
5.2	Integration	30
5.3	Local diabaticization	30
5.4	Fewest switch methods	31
5.5	Rescaling the velocities	31
5.5.1	Rescaling velocities in the (h, g) plane	32
5.5.2	Rescaling the velocities in the momentum direction	32
5.5.3	After hopping	33
5.6	Decoherence correction	33
5.7	Classical velocity rescale	33
5.8	<code>sh</code> options	33
6	Time overlap computation	36

7	Non-adiabatic and time-derivative couplings	37
7.1	Formalism	37
7.2	Usage	37
7.2.1	Setup	38
7.2.2	<code>cioverlap.od</code> interface	38
7.2.3	<code>cioverlap</code> interface	38
7.2.4	Generation of CIS-like wavefunctions	39
7.2.5	<code>cioverlap</code> options	39
7.3	Couplings computed	41
7.3.1	<code>nad_setup</code> options	41
7.3.2	Example	42
8	Outputs description	43
8.1	Minimal output	43
8.2	Output for adiabatic dynamics with more than one state	43
8.3	Output for non-adiabatic dynamics	44
8.4	Output with complex surfaces (CS-FSSH)	44
8.5	Method-specific outputs	44
8.5.1	Exciton models	44
8.6	Supplementary data	44
8.7	Text files	45
8.7.1	Note about parsing	45
8.7.2	<code>geometries.xyz</code>	45
8.7.3	<code>velocities.dat</code>	45
8.7.4	<code>nad.dat</code>	45
8.7.5	<code>wf.dat</code>	46
8.7.6	<code>energies.dat</code>	46
8.7.7	<code>populations.dat</code>	47
8.7.8	<code>seed.dat</code>	47
8.7.9	<code>sh_prob.dat</code>	47
8.7.10	<code>hopping_veloc.dat</code>	47
8.7.11	<code>hopping_veloc_micro.dat</code>	47
8.7.12	<code>gradients.dat</code> (<code>lvprt >=4</code>)	47
8.7.13	<code>osc_str.dat</code> (<code>lvprt >=4</code>)	48
8.7.14	<code>linmom.dat</code> (<code>lvprt >=5</code>)	48
8.7.15	<code>angmom.dat</code> (<code>lvprt >=5</code>)	48
8.8	Summary	48

1 Presentation

1.1 Newton-X

Newton-X is a set of programs designed to propagate semi-classical trajectories with surface hopping. It is composed of the following programs:

- `nx_moldyn`: propagation of a semi-classical trajectory ;
- `nx_restart`: generation of new inputs from a trajectory ;
- `nx_test`: interface to run examples provided in the package.

The program is designed to do one thing only: from a set of starting conditions, propagate the dynamics until the last step. It will not perform any statistical analysis of the results, nor will it generate initial conditions. The preferred way to handle these steps are:

- initial conditions: `initcond.pl`, from Newton-X classical series ;
- statistical analysis: `ULamDyn`.

1.2 Installation

The install process is described in the `INSTALL.md` file. Please refer to this file for installation instructions.

1.3 Bug reports

Bug should be reported as issues in our Gitlab repo <https://gitlab.com/light-and-molecules/newtonx>.

1.4 Contributions

This program has been written as an evolution of the “classical” Newton-X [1, 3], originally written by Mario Barbatti.

The code is developed in the Light and Molecules group from the Theretical Chemistry group in the “Institut de Chimie Radicalaire” (ICR, Aix-Marseille Université), with contributions from external collaborators. In particular, the code has been written by:

- Mattia Bondanza (University of Pisa): Tinker/MNDO and Tinker/G16 MMPol interfaces ;
- Saikat Mukherjee (ICR): ZPE corrections, diabatic populations ;
- Mauricio Persico and Giovanni Granucci (University of Pisa): overlap-based decoherence corrections ;
- Max Pinheiro Jr. (ICR): adaptive time-step ;
- Felix Plasser (Loughborough University): Local diabatization ;
- Eduarda Sangiogo Gil (University of Pisa): exciton models with Mopac and Gaussian, Mopac interface ;
- Baptiste Demoulin (ICR): H5MD, main developer and maintainer.

- Mario Barbatti (ICR): original author of the classical version Newton-X, supervision.

2 Quickstart

In this part we describe how to run basic Newton-X dynamics using two different electronic structure methods:

- Analytical models (spin-boson hamiltonian model)
- TDDFT (with Gaussian 16).

The goal is to provide a quick overview of how Newton-X runs.

Before starting, please ensure you have properly compiled / installed Newton-X following the instructions provided in the **README** of the distribution.

2.1 Preparation

Any Newton-X computation requires at least four components to run:

- a starting geometry (`geom.orig`)
- a set of starting velocities (`veloc.orig`)
- a list of options in namelist format (`user_config.nml`)
- some input for the electronic structure computation, in a folder `JOB_NAD` (the program used can compute non-adiabatic couplings) or `JOB_AD` (the program cannot compute the couplings).

The examples for this tutorial are taken from the `$NXHOME/examples` directory, so no further step will be required.

Typically, the starting geometry and velocity will come from an initial conditions generation program. `user_config.nml` can be generated by the Newton-X input generator `$NXHOME/utils/nxinp`. The input for the electronic structure computation will, of course, depend on the type of computation you wish to perform.

2.2 Running the program

2.2.1 Analytical model: Spin-Boson Hamiltonian

Dynamics with analytical models do not require any supplementary setup, and we can just use the files from `$NXHOME/examples/SBH/01-MD/inputs/`. We will run the dynamics in the folder `nx_tutorials/sbh/`:

```
mkdir sbh/ && cd sbh/
cp -r $NXHOME/examples/SBH/01-MD/inputs/* .
```

The dynamics can then be started with the `$NXHOME/utils/moldyn` script. All log information goes on `STDOUT`, so the output has to be redirected to, say, `md.out`:

```
$NXHOME/bin/nx_moldyn > md.out 2>&1
```

The dynamics should be very fast and complete under a few seconds. At the end of the computation, you should have the following elements in your directory:

```
$ ls -l  
  
nx_exported_config.nml  DEBUG/  geom.orig  JOB_AD/  md.out  RESULTS/  TEMP/  
user_config.nml  veloc.orig
```

- `md.out` contains the log for the computation.
- `TEMP` is the directory where the dynamics was computed ;
- `RESULTS` contains the data produced during the computation (geometries, energies, ...) ;
- `DEBUG` contains the full content of `TEMP` folder every 10 steps ;
- `nx_exported_config.nml` is the full configuration generated, that will be used by other programs, such as `nx_restart` (see below).

You can take a look at `nx_exported_config.nml` to check the value of all parameters that are used during the dynamics. Please refer to this documentation, or to the FORD generated code documentation, for more details about each parameter.

You can inspect `md.out` to get a basic understanding of the different parts of the dynamics. In this example it is very verbose, as we used the highest level of printing (`lvprt = 5`). By default the log is printed for every step. This can be controlled by setting the `kt` parameter, to only print log (except for `ERRORS` and `WARNINGS`) only every `kt` step.

In `user_config.nml` the option `use_txt_outputs` is set to `.true.`, meaning that text files will be produced even if the code was compiled with HDF5 support (by default, compiling the code with HDF5 support will prevent the production of text files). You can inspect these files now, and we will come back to this later (see [2.3](#)).

2.2.2 TDDFT: Gaussian

Running a dynamics with Gaussian 16 requires setting up the `g16root` environment, as Newton-X will look for the `g16` executable as `$g16root/g16/g16`. Gaussian 16 also requires the `GAUSS_EXEDIR` and `GAUSS_SCRDIR` to be setup. It is also possible to use the `g16.profile` script to setup all these environments before running Newton-X:

```
source $g16root/g16/bsd/g16.profile
```

To run this computation you will also need a program to compute the state-overlap matrix. In the example provided, we make use of the `cioverlap` suite of programs from J. Pittner. Please ensure that the `cioverlap`, `cis_casida` and `cis_slatergen` executables, as well as the LA library, are available in the folder located at `$CIOVERLAP`:

```
export CIOVERLAP="/path/to/cioverlap/executables/"
```

We will run the computation in `tddft`:

```
mkdir tddft/  
cp -r $NXHOME/examples/GAUSSIAN/01-MD-TDDFT-NAD-CIO/inputs/* tddft/  
cd tddft/
```

The `JOB_AD` folder contains the Gaussian input file, that must be `gaussian.com` (hard-coded). This is a standard Gaussian input file. The `basis` file contains the name of the basis set, and will be used to generate outputs for computing the atomic orbitals overlap between successive time-steps.

Running the computation is done exactly as before:

```
$NXHOME/bin/nx_moldyn > md.out 2>&1 &
```

This one is slightly longer to run, and you can follow the computation with:

```
tail -f md.out
```

In this case, the only difference with respect to the first example is that Newton-X will obtain the time-derivatives through the state-overlap matrix. The same set of directories and files as in the previous example will be produced.

2.3 Outputs

As highlighted above, the examples are setup so that text files will be produced even if the code has been compiled with HDF5 support. This is done by setting up the `use_txt_outputs` variable in `user_config.nml`.

All output files have a straightforward array structure, and can be processed right away by tools like `awk`, `gnuplot` or `numpy`. All data are provided in a.u., **except for** `geometry.xyz` that contains atomic positions in angströms. Some particular files:

- `cur_step_and_time`: contains the current step and current time (used for extracting restart files) ;
- `hopping_veloc*`: in case a surface hopping occurs, these files will contain the velocity before and after the hop, both in terms of micro-iterations (`hopping_veloc_micro.dat`) and macro-iterations (`hopping_veloc.dat`) ;
- `seed.dat`: contains the random seed (used to restart dynamics).

`geometry.xyz` is a standard `xyz` file and can be directly loaded in any visualization program.

2.3.1 HDF5 output

The HDF5 output (default: `dyn.h5`) is compliant with the H5MD format. It contains three main top level groups:

- `/h5md`: general information about the dynamics ;
- `/particles`: particles-dependent data (`position`, `velocity`, `force`, ...) ;
- `/observables`: average values (`kinetic_energy`, `potential_energy`, `wavefunction`, ...).

It should be possible to directly load this file in VMD, provided the [H5MD VMD plugin](#) has been installed.

The best way to interact with this kind of outputs is through the Python module `h5py`.

3 Main program

3.1 Overview

Apart from the `nx_moldyn` program used to run the dynamics, this part of Newton-X also comprises `nx_test` to perform test trajectories, and `nx_restart` to generate restart points from a trajectory.

We also provide a Perl script to generate inputs for Newton-X (`nxinp`). The other scripts present in the `utils` folder are designed to be used only as helpers by the main `nx_moldyn` main program.

3.2 Input overview

3.2.1 Mandatory input files

Input parameters By default, Newton-X uses a file named `user_config.nml` as input. This file has the structure of a Fortran namelist, that can contain any number of groups that corresponds to the different parts of the computation. The following general groups are defined:

- **nxconfig**: the main configuration group, and the **only one that is mandatory** for any computation. It contains immutable information, such as the number of atoms considered, the program used, the number of states... (see 3.5)
- **sh**: configuration for the integration of the time-dependent Schrödinger equation, and for the surface hopping algorithm (see 5.8) ;
- **nad_setup**: configuration for the computation of non-adiabatic couplings, with information about which couplings to take into account (see 7.3.1);
- **cioverlap**: configuration for the generation of time-derivative couplings from the state overlap matrix (see 7.2.5);
- **auxnac**: configuration for the generation of time-derivatives couplings from approximate analytic methods ;
- **adapt_dt**: configuration for the adaptive time-step algorithm ;
- **zpe_correct**: configuration for the zero-point energy correction schemes ;
- **h5md**: configuration for the generation of an H5MD output.

In addition, groups for configuring the electronic structure methods are also available: **sbh** for spin-boson Hamiltonian models, **columbus** for the Columbus program, etc. The options available are described in the subsections dealing with electronic structure methods, in section 4.

The input file is parsed with a home-made namelist parser, and follows the general namelist syntax. The only exception is that only 1D arrays are supported, with the following declaration:

```
my_array = 1, 2, 4, 5
```

This will declare an array with 4 elements, with values 1, 2, 4 and 5.

In addition, for each dynamics, Newton-X will also output a file `nx_exported_config.nml`, that contains the full set of parameters used for the computation.

System parameters The following additional input files are also mandatory for any computation:

- `geom.orig`: starting geometry, in Columbus input geometry format.
- `veloc.orig`: starting velocity, with one line per atom ;
- A set of input files (or parameters) for the electronic structure computation (in either `JOB_AD` or `JOB_NAD`).

3.2.2 Optional inputs

Optionnally, the following files can also be provided:

- `wf.inp`: starting wavefunction, given as a set of `nstat` lines, each containing the real and imaginary part of the wavefunction for each state.
- `rndseed`: initial seed for the random number generated. The format of this file depends on the compiler, and this option is mainly aimed at restarting computation.

3.2.3 Input generation

Inputs for Newton-X can be generated by the Perl `$NXHOME/Utils/nxinp` helper script. The script is interactive, and will propose several options for the different sections of the computation.

For each parameter, the user can either keep the proposed value (press `ENTER`), or enter the requested value. If anything is wrong with the selected value, the script won't go further and ask to correct the answer. At the end of each section, pressing `ENTER` will get the user back to the title screen. An input is only saved on disk by selecting option 7 on the title screen ("Save current configuration and exit").

The generated file is called `user_config.nml` and is intended to work directly with the `nx_moldyn` program. It will only contain the parameters that have been entered during input generation (including the ones where the default value has been kept).

3.3 Running dynamics

3.3.1 Some tips

- The program `nx_moldyn` has an option to stop just after the initialization process (dry-run). This can be useful to check the full set of parameters that will be used by the program before a production run, and can be done with the following command:

```
nx_moldyn --dry-run > md_dry.out 2>&1
```

- All programs have a `--help` command line option (or `-h`) documenting the command line interface.
- The `thres` parameter (from `nxconfig` section) controls the execution of the surface hopping procedures. If `thres = 100`, non-adiabatic dynamics is assumed, while when `thres = 0`, fully adiabatic dynamics will be performed (no change of surface will ever occur). The values in between are not implemented yet !

3.3.2 Following the dynamics

- All log information will be printed at `STDOUT`.
- The most important messages will have the `WARNING` or `ERROR` header, so grepping for those is a good way to see if anything goes unexpected.
- Surface hopping will be indicated as a `WARNING`. Furthermore, at the end of each step, the section “Hopping events” is printed as a 3-components vector corresponding to the number of hoppings, rejected hoppings (kinetic energy) and rejected hoppings (velocity) (see section 5).
- The surface hopping process is logged at `TEMP/sh.out`, with a format similar to the the main log at `STDOUT`.

3.4 Restarting dynamics

Newton-X can extract a new set of input files from any step in a trajectory with the program `nx_restart`. This program has to be called from the directory containing the `RESULTS` folder, and will do one of the following:

- Either create a folder `INFO_RESTART` with the data from the last step in the current trajectory (this is the default behaviour) ;
- Or create a folder `inputs_from_step_n` with the data from the selected step `n`, if run with the command:

```
nx_restart --step n
```

In the first case, `nx_moldyn` can be called directly in the main working directory, without any further manipulation: all input files will be taken from `INFO_RESTART`, and the original outputs will be updated with new data. In the second case, the set of inputs are to be used to start a brand new trajectory, with a new set of outputs. In both cases, `INFO_RESTART` and `inputs_from_step_n` will contain an updated `nx-restart-config.nml` file that sets `init_step`, `init_time` and `nxrestart` to the required value.

The complete set of commands available for `nx_restart` can be obtained with:

`nx_restart --help`

3.5 nxconfig options

Key	Description	Options	Default
<code>check_mo_ovl</code>	Perform a check on the conditioning of the MO overlap matrix between times <code>t</code> and <code>t-dt</code> .	<ul style="list-style-type: none">• 0 - Do not check the MO overlap.• 1 - Perform the overlap check.	0
<code>dc_method</code>	Define how to obtain the derivative couplings for integrating the TDSE	<ul style="list-style-type: none">• 0 - Do not compute (or read) the couplings (for adiabatic dynamics for instance)• 1 - Read non-adiabatic coupling vectors directly from the QM code• 2 - Read a state-overlap matrix (and compute it if required)• 3 - Auxiliary method (Baek An for instance)	AUTO
<code>debug_path</code>	Where to store debug files.		"DEBUG"
<code>dt</code>	Time step for the classical equations (in fs)		0.5
<code>etot_drift</code>	Kill trajectory if total energy deviate more than <code>Etot_drift</code> in comparison to the value in <code>t = 0</code> (in eV)		0.5
<code>etot_jump</code>	Kill trajectory if total energy deviate more than <code>Etot_jump</code> in one time step (in eV)		0.5
<code>gamma_model</code>	CS-FSSH - Model to compute the resonances.	<ul style="list-style-type: none">• 0 - Resonances provided by the QM computation (analytical models only) ;• 1 - Resonances obtained by an external script, with path "<code>path_gamma_model</code>" ;• 2 - Resonances obtained through an internal model.	AUTO
<code>init_geom</code>	File containing the initial geometries for the simulation.		"geom.orig"
<code>init_input</code>	Location of the QM input files to use (for the first step)		"JOB_AD"
<code>init_step</code>	Initial step number of the simulation		0
<code>init_time</code>	Initial time of the simulation		0.0

init_veloc	File containing the initial velocities for the simulation.	" veloc.orig "
killstat	Finish dynamics if after hopping the system remains more than timekill (see timekill description) fs on state killstat (1 - ground state)	1
kt	Print output at each kt steps	1
lvprt	Debug level: amount of output to print and output files to keep <ul style="list-style-type: none"> • 1 - Only error messages • 2 - Warning and error messages • 3 - (DEFAULT) Info, warning and error messages • 4 - Same as 3, with supplementary messages • 5 - Full debug output (warning, huge amount of text can be created !) 	3
methodname	(INTERNAL) Name of the method used	AUTO
nat	Number of atoms in the system	10
nstat	Number of states included in the dynamics	2
nstatdyn	Initial state of the dynamics (1 - ground state)	2
nxrestart	Indicates if the trajectory is a restart from a previous one. <ul style="list-style-type: none"> • 0 - This <i>IS NOT</i> a restart • 1 - This <i>IS</i> a restart 	0
output_path	Where to store the results of the computation	" RESULTS "
path_gamma_model	CS-FSSH - For "gamma_model = 1", specifies the path of the script used to compute the resonances.	AUTO

prog	Quantum chemistry program and method	<ul style="list-style-type: none"> • 0.1 - (Built-in) Spin-Boson Hamiltonian • 0.2 - (Built-in) Recoherence Model Hamiltonian • 0.3 - (Built-in) 1D models (including models from Tully et al.) • 1.1 - Columbus with MCSCF gradients • 1.2 - Columbus with MRCI gradients • 2.1 - Turbomole TDDFT • 2.2 - Turbomole RICC2 • 2.3 - Turbomole ADC2 • 3.1 - Gaussian (G16) • 10.1 - Mopac FOMO-CI • 11.1 - Exciton model with Mopac (FOMO-CI) • 11.2 - Exciton model with Gaussian (TDDFT) • 12.1 - Tinker-Gaussian MMPol • 12.2 - Tinker-MNDO QM/MM 	1.1
progrname	(INTERNAL) Name of the external QM program used		AUTO
ress_shift	CS-FSSH - For “gamma_model = 2”, specifies the shift in resonance energy.		AUTO
run_complex	Toggle CS-FSSH dynamics		.false.
same_mo	CS-FSSH - For “gamma_model = 2”, specify if the normal QM run should use the MO from the reference or not.		.true.
save_cwd	Save the working directory every “save_cwd” step	<ul style="list-style-type: none"> • -1 - Never save the working directory (DEFAULT for <code>lvprt > 3</code>) • N - Save the working directory every “N” step. 	10
thres	Threshold to initiate nonadiabatic coupling calculation (in eV)	<ul style="list-style-type: none"> • 0 - Only adiabatic dynamics • 100 - Only nonadiabatic dynamics • N - Initiate NAC computation when $DE < N$ 	100
thrs_cos	Threshold for the cosine similarity check in MO overlap between times t and $t + dt$.		0.98
thrs_norm	Threshold for the F-norm similarity check in MO overlap between times t and $t + dt$.		2.5

timekill	Finish dynamics if after hopping the system remains more than timekill fs on state killstat (see killstat description). 0 means deactivate killstat (in fs)	0
tmax	Total duration of the trajectory (in fs)	100
use_txt_outputs	Use text outputs (instead or with HDF5 files).	.true.
with_adt	Toggle adaptive time-step algorithms. <ul style="list-style-type: none"> • 0 - Do not trigger adaptive time-step at any time. • 1 - Allow the execution of adaptive time-step when required. 	0

4 Electronic structure interfaces

4.1 Available methods

4.1.1 Built-in analytical models

Newton-X implements the following analytical models:

- Spin-Boson Hamiltonian models ;
- Model potential for recoherence study ;
- Collection of 1D models ;
- 2D conical intesection models.
- CS-FSSH models: combination of 2 harmonic and / or exponential surfaces.

No further setup is required to use any of those models.

4.1.2 Interface to external programs

Newton-X is currently officially interfaced with the following programs:

- Columbus 7 (requires a version compiled with `-byterecl` to run with `cioverlap` time-derivative couplings) ;
- Turbomole 7.3 (version 7.5 do not output atomic orbitals coefficient, preventing the computation of state-overlap matrices) ;
- Gaussian 16 (not tested with Gaussian 09) ;
- ORCA 5 and higher (tested with ORCA 5.0.3).

The following interfaces have been integrated by external developers:

- Tinker / MNDO (Mattia Bondanza, University of Pisa) ;
- MOPAC (Eduarda Sangiogo Gil, University of Pisa) ;
- Exciton models with MOPAC and Gaussian (Eduarda Sangiogo Gil, University of Pisa).

For all interfaces, Newton-X tries to assume a standard installation, following the recommendation from each software !

4.2 Spin-Boson Hamiltonian

4.2.1 Model description

The model is discussed in details in references [12, 14, 5, 15]. In Newton-X, the model is applied in adiabatic representation, as presented in [6].

In this 2-state model the energies E_i , gradients $\frac{\partial E_i}{\partial Q_k}$ and non-adiabatic couplings F_{12}^k are computed with the following expressions, with i denoting a state and k denoting an oscillator in a system with N oscillators:

$$\begin{aligned}
E_i &= \frac{1}{2} \sum_j^N M_j \omega_j^2 R_j^2 + (-1)^i f \quad (i = 1, 2) \\
\frac{\partial E_i}{\partial Q_k} &= M_k \omega_k^2 R_k + (-1)^i g_k \left[\frac{\eta}{f} \right] \quad (k = 1, \dots, N), (i = 1, 2) \\
F_{12}^k &= -\frac{1}{2} g_k \left[\frac{\nu_0}{f^2} \right] \quad (k = 1, \dots, N), (i = 1, 2) \\
f &= (\eta^2 + \nu_0^2)^{1/2} \\
\eta &= \left(\sum_N g_j R_j \right) + \epsilon_0,
\end{aligned} \tag{1}$$

where R_j is the coordinate of oscillator j .

In the above expressions the terms ω_j and g_j come from the spectral density used, with expression:

$$J(\omega) = \frac{\pi}{2} \sum_{j=1}^N \frac{4g_j^2}{M_j \omega_j q_0^2} \delta(\omega - \omega_j). \tag{2}$$

The spectral density is typically derived in two different models, with their discretization:

- in the Debye model [20], the density is given by:

$$J_D(\omega) = \frac{E_r}{2} \frac{\omega \omega_c}{\omega^2 + \omega_c^2} \tag{3}$$

and discretized following:

$$\begin{aligned}
\omega_j &= \tan \left(\frac{j}{N} \tan^{-1} \left(\frac{\omega_{max}}{\omega_c} \right) \right) \omega_c \\
g_j &= \left[\frac{M_j E_r}{\pi N} \tan^{-1} \left(\frac{\omega_{max}}{\omega_c} \right) \right]^{1/2} \omega_j;
\end{aligned} \tag{4}$$

E_r is then the bath reorganization energy and ω_c is the characteristic frequency ;

- in the ohmic model [16], the density is given by:

$$J(\omega) = \frac{1}{2} \pi \hbar \xi \omega e^{-\frac{\omega}{\omega_c}} \tag{5}$$

and dicretized following:

$$\begin{aligned}
\omega_0 &= \frac{\omega_c}{N} \left(1 - e^{-\omega_{max}/\omega_c} \right) \\
\omega_j &= -\omega_c \ln \left(1 - j \frac{\omega_0}{\omega_c} \right) \\
g_j &= (\xi \hbar \omega_0 M_j)^{1/2} \omega_c;
\end{aligned} \tag{6}$$

here ξ is the Kondo parameter.

4.2.2 Parameters specification

In all cases the following parameters have to be specified:

- ω_{max} (in cm-1)
- ω_c (in cm-1)
- ϵ_0 (in cm-1)
- ν_0 (in cm-1)

Depending on the spectral density, the following parameters will also be needed:

- Debye density: E_r (in cm-1)
- Ohmic density: ξ (dimensionless)

Newton-X also offers the possibility of user's defined spectral density. In that case, values g_k (hartree / a_0) and ω_k (cm-1) for each oscillator must be provided as a separate file, located in `JOB_NAD/usersd_sbh.inp`, with the following format:

```
138.2378    0.012
203.879     0.000
```

This file specifies for instance two oscillators, with:

- $\omega_1 = 138.2378 \text{cm}^{-1}$, $g_1 = 0.012 \text{hartree}/a_0$;
- $\omega_2 = 203.879 \text{cm}^{-1}$, $g_2 = 0.000 \text{hartree}/a_0$;

For built-in densities, no other input will be required.

For Spin-Boson Hamiltonian model, only the column corresponding to the x coordinate of `geom.orig` will be taken into account !

4.2.3 sbh options

List of options for the SBH model:

Key	Description	Options	Default
e0	epsilon_0 parameter for the SBH model (in cm-1)		12000
er	Er parameter for Debye model (in cm-1)		0
jw	Type of spectral density	<ul style="list-style-type: none"> • user - User-provided file (named user_sd_sbh.inp) • ohmic - Ohmic model • debye - Debye model 	user
n	Number of oscillators in the system		2
v0	nu_0 parameter for the SBH model (in cm-1)		800
wc	Omega_c parameter for Debye and Ohmic models (in cm-1)		0
wmax	Omega_max parameter for Debye and Ohmic models (in cm-1)		0
xi	Xi parameter for Ohmic model		0

4.3 Collection of 1D-models

4.3.1 Model description

Newton-X implements the following one-dimensional models:

- the simple avoided crossing (SAC), dual avoided crossing (DAC) and extended coupling with reflection (ECR) models from Tully[24] ;
- the double arch (DA) model from Subotnik and Shenvi[23] ;
- the Nikitin Hamiltonian (NH)[17] (equation 7 from reference).

All these models define a potential V and its derivative dV , which are used to compute the energies, gradients and non-adiabatic couplings with the following expressions.

$$\begin{aligned}
E_{1,2} &= \frac{1}{2}(V_{11} + V_{22}) \mp \left(\frac{1}{4}(V_{22} - V_{11})^2 + V_{12}^2 \right)^{1/2} \\
G_{1,2}(x) &= \frac{1}{2} \left(\frac{dV_{11}}{dx} + \frac{dV_{22}}{dx} \right) \mp \left(\frac{1}{4}(V_{22} - V_{11}) \left(\frac{dV_{22}}{dx} - \frac{dV_{11}}{dx} \right) + V_{12} \frac{dV_{12}}{dx} \right) \left(\frac{1}{4}(V_{22} - V_{11})^2 + V_{12}^2 \right)^{-1/2} \\
F_{12} &= \frac{1}{1 + \left(\frac{2V_{12}}{V_{22} - V_{11}} \right)^2} \left(\frac{1}{(V_{22} - V_{11})} \frac{dV_{12}}{dx} - \frac{V_{12}}{(V_{22} - V_{11})^2} \left(\frac{dV_{22}}{dx} - \frac{dV_{11}}{dx} \right) \right)
\end{aligned} \tag{7}$$

The reader is referred to the references given above for the exact expressions of V and dV .

4.3.2 Parameters specification

To remain as general as possible, the parameters for each model will be specified as a single array of real values. The model is specified with the `onedim_mod` key from the `onedim_model` section of the configuration (see next section).

The models currently implemented have the following parameters defined:

Table 3: Parameters for all implemented 1D models in Newton-X

Model	onedim_mod	Parameters (in order)
SAC	1	$A = 0.01$; $B = 1.6$; $C = 0.005$; $D = 1.0$
DAC	2	$A = 0.10$; $B = 0.28$; $C = 0.015$; $D = 0.06$; $E0 = 0.05$
ECR	3	$A = 6E-4$; $B = 0.1$; $C = 0.9$
DA	4	$A = 6E-4$; $B = 0.1$; $C = 0.9$; $Z = 0.4$
NH	5	$A = 0.05$; $B = 0.1$; $\theta(\pi/n) = 12$; $alpha = 2$; $DE = 0.01$

To specify the individual parameters for each model, a file named `onedim_parameters.inp` has to be present in `JOB_NAD`, with the following content, with parameters indicated in the same order as in table 3:

```
&onedim_parameters
parm(1) = 0.01
parm(2) = 1.6
parm(3) = 0.005
parm(4) = 1.0
/
```

So, this example file would define the model “SAC”.

4.3.3 onedim_model options

Key	Description	Options	Default
-----	-------------	---------	---------

- 1 - Simple avoided crossing [Tully, JCP 93, 1061 (1990)]
- 2 - Dual avoided crossing [Tully, JCP 93, 1061 (1990)]
- 3 - Extended coupling with reflection [Tully, JCP 93, 1061 (1990)]
- 4 - Double arch [Subotniki and Shenvi, JCP 134, 024105 (2011)]
- 5 - Nikitin Hamiltonian [Nikitin, Adv Chem Phys 5, 135 (2011)]

4.4 Columbus

4.4.1 Setup

- To use the Columbus interface, you should define the environment `COLUMBUS` pointing to the directory containing the Columbus executables.
- All Columbus executables should be in the `PATH`:

```
export PATH=$PATH:$COLUMBUS
```

4.4.2 Running dynamics

Two types of calculations are currently supported:

- MCSCF dynamics (`prog = 1.1`)
- MRCI dynamics (`prog = 1.2`) (serial only)

In both cases the `runc` master script will be called by Newton-X as:

```
runc -m memory
```

where `memory` is the value of the `mem` parameter from `&nxcconfig`. If the program terminates with a non-zero status, the dynamics abort.

For both types of jobs, the MCSCF energies will first be read from `WORK/mcscfsm`. If the MCSCF did not converge, an warning is issued. Then, for MCSCF dynamics, the contributions to the wavefunction are read either from `WORK/mcpcls`, or from the set of files `LISTINGS/mcpcls.drt1.state...` if the former does not exist. For MRCI dynamics, the MRCI energies are read from `WORK/ciudgsm`, and the contributions are read from `WORK/cipcls`, while the weight of reference function is checked from `WORK/ciudgls`.

If non-adiabatic couplings are requested from Columbus, the gradients are read from the set of files `GRADIENTS/cartgrd.drt1.state...`, and from `GRADIENTS/cartgrd.all` if the couplings are not computed.

If non-adiabatic couplings are to be computed with Columbus, and subsequently used (typically if `cioverlap` will not be used, with local diabatization for instance), they are read from the files `GRADIENTS/cartgrd.nad.drt1.state...`

Finally, if the files `LISTINGS/trncils.FROMdrt1.state...` exist, the oscillator strengths will be extracted from and reported in the log file. **Important:** the values reported in the `RESULTS` folder by Newton-X will always be for the transitions from the ground state !

4.4.3 columbus options

Key	Description	Options	Default
<code>all_grads</code>	Which gradients to compute.	<ul style="list-style-type: none"> • 0 - Compute only the gradient for the current state. • 1 - Compute the gradient for all states. 	1
<code>ci_conv</code>	What to do if CI calculation do not converge.	<ul style="list-style-type: none"> • 0 - warn and continue • 1 - kill trajectory 	0
<code>ci_iter</code>	Maximum number of iteration in the CI procedure (<code>niter = nstat * ci_iter</code> in <code>ciudgin</code>).		30
<code>ci_type</code>	Type of the CI job.	<ul style="list-style-type: none"> • 0 - No CI (MCSCF) • 1 - MRCI 	AUTO
<code>cirestart</code>	CI restart options.	<ul style="list-style-type: none"> • 0 - do not use previous CI vector • 1 - use previous CI vector 	0
<code>citol</code>	Convergence criterion for CI.		1E-4
<code>grad_lvl</code>	Level at which the gradients should be computed	<ul style="list-style-type: none"> • 1 - Compute the gradients at the MCSCF level. • 2 - Compute the gradients at the MRCI level. 	AUTO
<code>ivmode</code>	Initial CI-vector generation mode. See COLUMBUS docs (<code>ciudg</code> program) for a list of options. <code>Cirestart</code> keyword has priority over <code>ivmode</code> keyword. The default is generate vectors by iterative diagonalization of the reference space hamiltonian matrix.		8
<code>mc_conv</code>	What to do if MCSCF calculation do not converge.	<ul style="list-style-type: none"> • 0 - warn and continue • 1 - kill trajectory 	0
<code>mel</code>	(INTERNAL) Maximum excitation level		AUTO
<code>mem</code>	Core memory (takes precedence over <code>nxconfig</code> section) (in MWords)		200

<code>mocoef</code>	Molecular orbitals usage.	<ul style="list-style-type: none"> • 0 - Use the same mocoef file for all time steps • 1 - Use the mocoef from the previous time step 	1
<code>mode</code>	(INTERNAL) General type of job.	<ul style="list-style-type: none"> • "ms" - Multistate • "ss" - singlestate • "mcs" - Multiconfigurational gradients • "mc" - Multiconfigurational 	AUTO
<code>prt_mo</code>	Save molecular orbitals file every 'prt_mo' step	<ul style="list-style-type: none"> • -1 - Deactivate this option (orbitals will never be saved) • N - Save every N steps 	20
<code>quad_conv</code>	Set value of NCOUPL in <code>mcsconfin</code> .		60
<code>reduce_tol</code>	Tolerance in CI calculations.	<ul style="list-style-type: none"> • 0 - keep <code>rtolci</code> and <code>rtolbk</code> in <code>ciudgin</code> as 1E-4 for all states • 1 - use <code>citol</code> only for <code>nstatdyn</code>. For all other states use <code>10 * citol</code> • 2 - use <code>citol</code> for <code>nstatdyn</code>, <code>10 * citol</code> for states coupled to <code>nstatdyn</code> (as defined in <code>transmomin</code>), and <code>1000 * citol</code> for other states 	1

4.5 Turbomole

4.5.1 Setup

To run dynamics with Turbomole, the `TURBODIR` environment has to be set accordingly to the Turbomole documentation, and the `scripts` and the corresponding `bin` directories should be in the `PATH`:

```
export TURBODIR="/path/to/TURBOMOLE/"
export PATH="$TURBODIR/scripts:$PATH"
export PATH=$TURBODIR/bin/`sysname`: $PATH
```

4.5.2 Running computations

Newton-X supports three methods for running dynamics with Turbomole:

- DFT / TDDFT (`prog` = 2.0)
- MP2 / CC2 (`prog` = 2.1)
- MP2 / AC2 (`prog` = 2.2)

The programs used will of course depend on the type of job requested. Any Turbomole job will require at least the following files:

- `control`
- `basis`
- `mos`

Optionnally, the `auxbasis` file can be included for RI jobs.

In (TD)DFT dynamics, we have three different cases:

- Excited state dynamics (`typedft = 1`);
- Ground state dynamics, with excited states computed (`typedft = 2`);
- Ground state only (`typedft = 3`).

Newton-X will first check if `auxbasis` is present. If so, the SCF program will be `ridft`, followed by `rdgrad` for the gradients. Please note that these program do not work for excited states. If no RI-type computation is requested, then Newton-X will call `dscf` and `grad` (or `dscf_smp` and `grad_smp` if `nnodes > 1`) for `typedft = 3` (ground state only). If `typedft = 1` (excited state dynamics), then the gradient program called will be `egrad`. Finally, for `typedft = 2` (ground state dynamics in the presence of excited states), Newton-X will call `grad` for the gradients, and compute the energies with `escf`.

The energies, oscillator strengths and contribution to the wavefunction are read from the file `scf.out` (`typedft = 3`) or `grad.out`. The gradients will be extracted from the file `gradient` in all cases.

For CC2 and ADC2 dynamics, Newton-X will call `dscf` then `ricc2` (or `dscf_smp` and `ricc2_smp` if `nnodes > 1`). For ADC2, two `grad` computations have to be carried out. This is handled automatically.

The energies, oscillator strengths and contributions to the wavefunction will be read from the file `grad.out`. If the D1 diagnostics are higher than 0.04 (for MP2) or 0.05 (for CC2), a warning will be issued. Gradients are extracted from the `gradient` file.

4.5.3 turbomole options

Key	Description	Options	Default
<code>multiplicity</code>	Multiplicity of the excited states given as one of the irrep options in control file of Turbomole.		1
<code>nnodes</code>	Number of cores to use for parallel Turbomole (smp, no mpi!)		1
<code>npre</code>	Number of roots used in preoptimization steps given as one of the irrep options in control file of Turbomole.		1

nstart	Number of start vectors generated or read from file given as one of the ir-rep options in control file of Turbomole.	1
---------------	--	---

4.6 Gaussian 16

4.6.1 Setup

To run dynamics with Gaussian 16, the environment **g16root** has to be set and points to the location of the **g16** folder. Newton-X will call Gaussian executables (**g16**, **rwdump** mostly) as:

```
$g16root/g16/g16 gaussian.com
```

for instance.

4.6.2 Running dynamics

The Gaussian interface can be used to run restricted TDDFT dynamics. The unrestricted method is not tested. Newton-X will require a **gaussian.com** input, as well as a **basis** file containing the name of the basis set required (this is used for computing time-derivatives with external overlap programs). If generated basis sets are to be used, all information about the generated basis have to be written in a **basis2** file. All those file should be put in the **JOB_AD** directory.

The energies, oscillator strengths and contributions to the wavefunction will be extracted from the output file **gaussian.log**. If the Gaussian job did not converge properly, an error is issued, and the dynamics is immediately terminated.

4.6.3 gaussian options

Key	Description	Options	Default
kind_g09	Defining the wavefunction occupation, restricted or unrestricted approaches	<ul style="list-style-type: none"> • 0 - closed shell RHF wavefunction. (closed shell in GS, Singlets in ES) • 1 - Open shell, restricted or unrestricted (radicals) 	0
ld_thr	Linear dependence threshold.		14
mocoef	Molecular orbitals usage.	<ul style="list-style-type: none"> • 0 - Compute the initial guess at every time step • 1 - Use the mocoef from the previous time step • 2 - Use the same checkpoint file for all time steps 	1

<code>prt_mo</code>	Save mocoef to DEBUG directory every <code>prt_mo</code> timesteps.	20
<code>td_st</code>	Reading of the excited states from previous steps. <ul style="list-style-type: none"> • 0 - Compute the excited states without previous reference • 1 - Read states from the checkpoint file, what file is controlled by mocoef parameter 	0

4.7 Orca

4.7.1 Setup

Newton-X will look for the `orca` executable as `$ORCA/orca`, so the `$ORCA` environment has to be defined. This environment is also required to run the test suite for the NX/ORCA interface.

4.7.2 Running dynamics

ORCA can be used to run dynamics with the TD-DFT or its TDA variant. A file `orca.inp` should be present in the `JOB_AD` directory, and that input should be able to produce energies and forces. The only mandatory keyword is `ENGRAD`.

A line specifying the input coordinate is mandatory (the name `orca.xyz` is also hard-coded):

The ORCA output is named `orca.out`.

A section `%tddft` is always added by Newton-X to setup the `nroots` and `iroot` parameters. Other TD-DFT parameters can be added to the input without NX caring for them. NX will also never parse nor touch other blocks, or even the route line (starting with `!`).

By default ORCA performs TDA computation, and not full TD-DFT. Full TD-DFT can be switched on by setting `TDA false` in the `%tddft` block of `orca.inp`. By default NX will also try to parse the TDA results. Therefore, if full TD-DFT is performed in ORCA, the **is_tda must be switched to .false.** in the Newton-X input, in the section `%orca`.

The files `orca.gbw` and `orca.out` will be backed up every `prt_mo` step (see below), and will be compressed to save disk space.

The interface offers several options for the initial guess with the `mocoef` option:

- `mocoef = 0`: always compute the initial guess ;
- `mocoef = 1`: use a guess from the previous time-step (*i.e* keep the `orca.gbw`) ;
- `mocoef = 2`: use the same MO guess for all steps. This option requires the presence of `orca.gbw` file in `JOB_AD`, along with `orca.inp`.

4.7.3 orca options

Key	Description	Options	Default
<code>is_tda</code>	Flag to decide if the computation is full TD-DFT ('.false.') or TDA ('.true.').		.true.
<code>mocoef</code>	Molecular orbitals usage.	<ul style="list-style-type: none">• 0 - Compute the initial guess at every time step• 1 - Use the mocoef from the previous time step• 2 - Use the same checkpoint file for all time steps	1
<code>prt_mo</code>	Save molecular orbitals (GBW file) every 'prt_mo' step.		20

4.8 Complex Surface Fewest-Switch Surface Hopping (CS-FSSH)

CS-FSSH is implemented in NX following the paper by Kossoski and Barbatti [13], and is available for a selection of analytical models, and in combination with Columbus.

A CS-FSSH dynamics can be performed with the following addition in `user_config.nml`:

```
&nxcconfig
  run_complex = .true.
/
```

When doing so, Newton-X will restrict `progrname` and `methodname` as:

- `progrname = columbus`
- or `progrname = analytical AND methodname = cs_fssh`.

4.8.1 Analytical models

For analytical models no further configuration is required from `nxcconfig`. The model has to be selected by the `cs_mod` keyword in the `&cs_fssh` section of `user_config.nml`, and can have the following values:

- `cs_mod = 1`: 2RHE (first potential is exponential, second is harmonic) ;
- `cs_mod = 2`: 2REH (first potential is harmonic, second is exponential) ;
- `cs_mod = 3`: 2REE (both potential are exponential) ;
- `cs_mod = 4`: 2REE (both potential are harmonic).

An example is available for each model in the `examples/CS-FSSH-ANALYTICAL` folder. In particular, the file `JOB_NAD/constants.dat` documents how to specify the parameters for each potentials. Please refer to the original paper [13] for details about the interaction potentials and the coupling to the reference harmonic potential.

4.8.2 Integration with Columbus

For using CS-FSSH in combination with Columbus some configuration is needed. First, the resonance energies cannot be extracted from any QM computation, so a model has to be chosen to obtain them. This is specified by the variable `nxconfig.gamma_model`:

- `gamma_model = 1`: resonance energies are obtained by an external script, that is found at `nxconfig.path_gamma_model`; the script or program `nxconfig.path_gamma_model` will be copied into the `TEMP` directory, and executed from there at each step.
- `gamma_model = 2`: resonance energies are obtained through interpolation within Newton-X. In that case, a file `gamma_stateN` (where `N` is the state number) has to be provided for each state. These files contain pre-computed pairs of energies and widths. Newton-X will then carry out a third order polynomial interpolation to find the widths at the instantaneous energy of each state.

For `gamma_model = 2`, a reference computation is also required. This computation is a ground-state only computation, where the only interest lies in the energy of the system. Inputs for this computation have to be provided in a folder `REF/`, similar to `JOB_NAD`. The reference computation will be carried out **before** the normal QM calculation, and it is possible to reuse the molecular orbitals from the reference by setting `nxconfig.same_mo` to 1.

Examples are provided in the folders `examples/COLUMBUS/06-MD-CS-FSSH-MODEL-1` and `examples/COLUMBUS/06-MD-CS-FSSH-MODEL-2` for both models.

5 Surface hopping

In this part we break down the different steps to determine the evolution of the electronic population of the different states, and to determine if surface hoppings can occur at a given step. We then describe the different decoherence correction schemes, and the methods used to ensure total energy conservation after hopping from one state to another.

5.1 Some definitions

The following notations will be used throughout this part:

- $A_K(t)$ is the population of state K at time t ;
- $\gamma_K(t)$ is the phase of state K at time t , and we define $\gamma_{KL}(t) = \gamma_K(t) - \gamma_L(t)$;

- $\sigma_{KL}^{NAD}(t)$ is the time-derivative coupling between states K and L at time t (see 7) defined by:

$$\sigma_{KL}^{NAD}(t) \quad (8)$$

- $E_K(t)$ is the electronic energy of state K at time t ;
- T is the time of the classical trajectory ;
- ΔT is the classical time-step ;
- t is the time during surface hopping trajectory (microiteration time)
- δT is the microiteration time-step ;
- $t_0 = T - \Delta T$ is the starting time for the microiterations ;
- $t_1 = T$ is the ending time for the microiterations.

All quantities are computed only at T_0 and T_1 . Thus, the corresponding data for any time $T_0 < t < T_1$ is obtained by linear interpolation (or quadratic interpolation for the energies).

5.2 Integration

The first problem handled is the one of the evolution of the electronic population. In Newton-X we follow the description from reference [8]. The following equations have to be solved:

$$\begin{aligned} \dot{A}_K(t) &= - \sum_{K \neq L} A_L(t) e^{i\gamma_{KL}} \sigma_{KL}^{NAD}, \\ \gamma_K(t) &= \int_0^t E_K(t') dt'. \end{aligned} \quad (9)$$

Newton-X implements different algorithms for integrating these equations:

- Second-order method (Euler's method) (only used for the first integration step);
- Finite elements methods from [8] (`integrator` = 0) ;
- Butcher's method [4] (`integrator` = 1) ;
- Local diabaticization [19] (`integrator` = 2, see section 5.3) ;
- **Experimental** Unitary propagators (`integrator` = 3).

The algorithm starts by integrating the phase, then injects the new phase to integrate the electronic population.

5.3 Local diabaticization

To do.

5.4 Fewest switch methods

Once phases and populations have been integrated, the hopping probabilities are computed, either with the standard Tully expression [24]:

$$b_{KL}(t) = -2\text{Re}(A_j(t)A_j^*(t)e^{i\gamma_{KL}}\sigma_{KL}^{NAD}(t)), \quad (10)$$

or with the more advanced Tully - Hammes-Schiffer expression [11]:

$$g_{KL} = \frac{\int_t^{t+dt} dt b_{LK}(t)}{|A_K(t)|^2}. \quad (11)$$

In the second case, the finite elements scheme is used to perform the integration.

When the probabilities have been obtained, the program implements the fewest switch algorithm from Tully [24].

5.5 Rescaling the velocities

If the fewest switch algorithm finds that the system can hop to another state, the program will decide if the hop is possible or not based on total energy conservation:

$$\begin{aligned} E_{tot}^{new} - E_{tot}^{old} &= 0 \\ E_{pot}^{new} + E_{kin}^{new} - E_{pot}^{old} + E_{kin}^{old} &= 0 \\ \underbrace{E_{pot}^{old} - E_{pot}^{new}}_{\Delta E} + E_{kin}^{old} &= E_{kin}^{new} \end{aligned} \quad (12)$$

Thus, the surface hopping is possible only when:

$$\Delta E + E_{kin}^{old} \geq 0. \quad (13)$$

Ensuring total energy conservation means that the atomic velocities will need to be rescaled according to equation 12, and this can be done in two different ways in Newton-X. Let:

- J and L be the new and old states, respectively (*i.e.* the system hops from L to J) ;
- v_α^n and v_α^o be the new (rescaled) and old velocities of atom α , respectively ;
- M_α be the atomic mass of atom α ;
- h_α^{LJ} be the normalized non-adiabatic coupling vector between states J and L for atom α ;
- g_α^{LJ} be the normalized gradient difference between between states J and L for atom α :

$$g_\alpha^{LJ} = (\text{grad}_\alpha^J - \text{grad}_\alpha^L)/2;$$

- u_α be the direction of rescale for the atom α ;

5.5.1 Rescaling velocities in the (h, g) plane

This is usually the best choice [7], when non-adiabatic coupling vectors are available. This corresponds to `adjmom` = 2, and requires the additional angle parameter θ to define the angle in the (h, g) plane. In Newton-X θ is defined by the `adjtheta` variable, set to 0 by default (corresponding to a rescaling in the direction of h).

The rescaling direction can be rewritten as:

$$u_\alpha = \cos(\theta)h_\alpha^{LJ} + \sin(\theta)g_\alpha^{LJ}. \quad (14)$$

Here it is more sound to only consider the kinetic energy in the u direction when determining if the system can hop or not. This amounts to replacing equation 13 by:

$$\Delta E + \frac{1}{2 \sum_\alpha \frac{u_\alpha^2}{M_\alpha}} (v_\alpha^o \cdot u_\alpha)^2 \geq 0. \quad (15)$$

If this condition holds, we can attempt to hop from surface L to surface J . The new velocities can be written as:

$$v_\alpha^n = v_\alpha^o + \gamma_{LJ} \frac{u_\alpha}{M_\alpha} \quad (16)$$

(the division by M_α is useful when expressing the kinetic energy).

Injecting this expression into equation 12, we obtain the second order equation for γ_{LJ} :

$$\underbrace{\frac{1}{2} \sum_\alpha \frac{u_\alpha^2}{M_\alpha} \gamma^2}_{A/2} + \underbrace{\sum_\alpha v_\alpha^o u_\alpha \gamma}_{B} - \Delta E = 0 \quad (17)$$

Defining $\Delta = B^2 + 2A\Delta E$, the equation may have solutions only when $\Delta > 0$. If this is not the case, the hopping is **frustrated (velocity condition)**, and Newton-X will report it as a warning. The velocities are then multiplied by the `mom` parameter.

If $\Delta > 0$, then we take the solution leading to the lowest change in magnitude:

$$\gamma = \begin{cases} \frac{-B+\sqrt{\Delta}}{A} & \text{if } |-B + \sqrt{\Delta}| < |-B - \sqrt{\Delta}| \\ \frac{-B-\sqrt{\Delta}}{A} & \text{if } |-B + \sqrt{\Delta}| \geq |-B - \sqrt{\Delta}|. \end{cases} \quad (18)$$

5.5.2 Rescaling the velocities in the momentum direction

In cases where non-adiabatic coupling vectors are not available, a possibility is to rescale the velocities in the momentum direction. However, if the system is too large, the condition from equation 13 may always hold, and back hoppings will happen all the time. A solution to overcome this problem is to rescale the kinetic energy in this expression according to the number of degrees of freedom in the system N_{DF} . We thus have two cases:

- `adjmom` = 1: (default) use the original $\Delta E + E_{kin}^{old} \geq 0$ (equation 13).
- `adjmom` = 0: replace the condition with $\Delta E + E_{kin}^{old}/N_{DF} \geq 0$.

If the condition doesn't hold, then we have a **frustrated hopping (kinetic energy)**. Else, and contrary to the case when rescaling in the (h, g) plane, it is always possible to hop. Indeed, we can write:

$$v_{\alpha}^n = \gamma v_{\alpha}^o. \quad (19)$$

Reporting this expression into equation 12 gives:

$$\begin{aligned} \gamma^2 E_{kin}^{old} &= \Delta E + E_{kin}^{old} \\ \gamma &= \pm \sqrt{\frac{\Delta E}{E_{kin}^{old}} + 1} \end{aligned} \quad (20)$$

The sign chosen is given by the `mom` parameter.

5.5.3 After hopping

If the velocity has been rescaled at step t , it is necessary to also adapt the starting and ending velocities as well, according to:

$$\begin{aligned} v(T_0) &= v(T_0) - a^j * (t - T_0), \\ v(T_1) &= v(T_0) + a^j * (\Delta T - (t - T_0)), \end{aligned} \quad (21)$$

where a^j is the acceleration on surface j . This modification just affects the interpolation of velocities, and not the *classical* velocities at time T_0 and T_1 .

5.6 Decoherence correction

Optionnally, it is possible to apply a decoherence correction to the electronic populations:

- with `decohmod = 0`, no correction is applied ;
- `decohmod = 1` corresponds to energy-based decoherence scheme [9];
- `decohmod = 2` corresponds to overlap-based decoherence scheme [10].

All populations are **renormalized** after this step.

5.7 Classical velocity rescale

Finally, if surface hopping occurred, the classical velocities need to be rescaled as well in a direction given either by the momentum (`adjmom < 0`) or by some direction in the (g, h) plane. The process is exactly the same as for rescaling the velocities in during the determination of surface hopping possibility (see above, 5.5).

5.8 sh options

Key	Description	Options	Default
<code>adjmom</code>	Condition for accepting a hop, and direction of velocity rescaling (see manual for more details).	<ul style="list-style-type: none"> • 0 - (Large systems, NAC not available) Rescale in the direction of velocity, and scale kinetic energy in the hopping condition with the number of degrees of freedom. • 1 - (NAC not available) Rescale in the direction of velocity, and don't rescale the kinetic energy in the hopping condition. • 2 - (NAC available) Rescale in a direction given by the angle "adjtheta" in the plane (h, g). 	0
<code>adjtheta</code>	Direction (in degrees) in which the velocity is rescaled after a hopping (for "adjmom = 2" only).	<ul style="list-style-type: none"> • 0 - adjust along non adiabatic vector h • 90 - adjust along gradient vector g • n - adjust along the given "n" angle in the plane (h, g) 	0
<code>decay</code>	(EDC model) Decay time.		0.1
<code>decohmod</code>	Decoherence correction model to used	<ul style="list-style-type: none"> • 0 - No decoherence correction • 1 - EDC energy based decoherence (Granucci and Persico, <i>JCP</i>, 126 (2007)) • 2 - ODC Overlap based decoherence (Granucci, Persico and Zocante, <i>JCP</i>, 133 (2010)) 	1
<code>decovlp</code>	(ODC model) Gaussian width		0.05
<code>forcesurf</code>	Force the hopping to this surface (GS is 1) if <code>nohop</code> is set.		1
<code>getphase</code>	Phase to use	<ul style="list-style-type: none"> • 0 - use phase provided by the overlap of CI vectors (NOT IMPLEMENTED). • 1 - use phase provided by the scalar product between $h(t)$ and $h(t-dt)$. 	1
<code>iatau</code>	(ODC model) Propagate ancillary gaussian wavepacket every <code>iatau</code> steps.		1

integrator	Integrator type for the time-dependent Schrödinger equation	<ul style="list-style-type: none"> • 0 - Finite element method • 1 - Butcher, 5th order • 2 - Unitary propagators for local-diabatization method • 3 - Unitary propagators (Experimental) 	1
mom	What to do after a frustrated hopping	<ul style="list-style-type: none"> • -1 - Invert momentum direction • 1 - Keep momentum direction 	1
ms	Number of sub-timesteps for integration of the time-dependent Schrödinger equation. For local-diabatization (integrator = 2), this value will be used as a dummy value (the number of integration step is determined automatically).		20
nohop	Force the hopping at a certain time step	<ul style="list-style-type: none"> • 0 - Normal surface hopping • -1 - Hopping is not allowed at any time • n - Hopping is forced at (and only at) step n (n = positive integer) 	0
nrelax	Number of substeps after hopping, in which other hopping is forbidden		0
phase	Integrate the phase along the trajectory (only for debug purposes)	<ul style="list-style-type: none"> • 0 - No, phase is always 0 • 1 - Yes 	1
popdev	Kill trajectory if total adiabatic population deviate more than popdev from the unity		0.05
probmin	Do not hop if probability is smaller than probmin		0
seed	Random number generation	<ul style="list-style-type: none"> • -2 - (FOR TESTS ONLY) Use a home-made PRNG • -1 - A randomized seed is used • 0 - A default random-number seed is used • n - Use this number as a random seed 	-1
thrup	(ODC model) Wavefunction overlap threshold (S_{min})		0.005
tully	Fewest-switch algorithm	<ul style="list-style-type: none"> • 0 - Tully • 1 - Hammes-Schiffer and Tully 	1
vdoth	(INTERNAL) Indicate the content of the nad object.	<ul style="list-style-type: none"> • 0 - 'nad' contains non-adiabatic coupling vectors. • 1 - 'nad' contains time-derivatives. 	AUTO

6 Time overlap computation

Newton-X will need a “double-molecule”, *i.e* a computation on a virtual super-system composed of the superposition of the system at times t and $t - dt$, in two cases:

- time-derivative couplings cannot be derived from non-adiabatic coupling vectors, and should be obtained from the state-overlap matrix (see [7](#)) ;
- the conditionning of the molecular orbitals overlap matrix should be checked to prevent any trouble coming from orbital rotations (for instance, in CASSCF computations).

To this end, Newton-X will create at the first step of the dynamics the inputs required to run a computation on a system with size doubled, and put those inputs inside a folder `double_molecule_input`. Then, at each step of the dynamics, it will use these inputs in coordination with the current and previous geometries to perform the overlap calculation in the folder `overlap`. The content of this folder is backed up as `overlap.old`.

The computation is very simple and only requires printing the atomic orbitals of the super-system. Of course such a system can trigger crashes of the electronic structure program, but it doesn’t matter as long as the AOs are properly printed.

The conversion from AOs to MOs is then carried out either by a state-overlap program if needed, or else by internal Newton-X routines.

7 Non-adiabatic and time-derivative couplings

The integration of the time-dependent Schrödinger equation requires the evaluation of time-derivative couplings, that are derived from atomic velocities v_i and non-adiabatic coupling vectors h_i^{KL} with the expression:

$$\sigma_{KL}^{NAD}(t) = \sum_i v_i h_i^{KL} \quad (22)$$

For electronic structure methods allowing the computation of non-adiabatic coupling vectors the derivation is straightforward.

7.1 Formalism

It is possible, however, to estimate the time-derivative couplings by means of the time evolution of the state-overlap matrix, following Hammes-Schiffer and Tully:[11]

$$\sigma_{KL}^{NAD}(t) \approx \frac{1}{2\Delta t} \left[\left\langle \phi_K \left(t - \frac{\Delta t}{2} \right) \middle| \phi_L \left(t + \frac{\Delta t}{2} \right) \right\rangle - \left\langle \phi_K \left(t + \frac{\Delta t}{2} \right) \middle| \phi_L \left(t - \frac{\Delta t}{2} \right) \right\rangle \right] \quad (23)$$

This expression can be recast in terms of the actually computed wavefunctions at multiples of the time-step Δt as:[18]

$$\begin{aligned} \sigma_{KL} \left(t - \frac{3\Delta t}{2} \right) &\approx \frac{1}{2\Delta t} [\langle \phi_K(t - 2\Delta t) | \phi_L(t - \Delta t) \rangle - \langle \phi_K(t - \Delta t) | \phi_L(t + 2\Delta t) \rangle] \\ \sigma_{KL} \left(t - \frac{\Delta t}{2} \right) &\approx \frac{1}{2\Delta t} [\langle \phi_K(t - \Delta t) | \phi_L(t) \rangle - \langle \phi_K(t) | \phi_L(t - \Delta t) \rangle] \\ \sigma_{KL}(t) &\approx \frac{1}{2} \left[3\sigma_{KL} \left(t - \frac{\Delta t}{2} \right) - \sigma_{KL} \left(t - \frac{3\Delta t}{2} \right) \right], \end{aligned} \quad (24)$$

the last expression being obtained by linear interpolation. The problem is then reduced to computing the overlap of wavefunctions at different time-steps.

Newton-X is currently interfaced with two programs, based on different approaches to compute the state-overlap matrices:

- `cioverlap` implements a determinants derivative (DD) methods, and is available for all external programs, for either multi-reference[18] and single-reference wavefunctions[2] ;
- `cioverlap.od` implements an orbital derivative (OD) method[21] and is available for linear response and single-reference methods (namely, with Turbomole and Gaussian). It should be noticed that the current implementation[22] only allows computation of couplings between excited states.

7.2 Usage

In Newton-X the state-overlap computation programs are run in the `cioverlap` directory. The content of this folder is backed up at each step as `cioverlap.old`.

7.2.1 Setup

Newton-X will look for the programs in the location given by the environment `$CIOVERLAP`, which should point to the directory containing the programs.

A set of statically compiled binaries is available with each release of Newton-X through the Gitlab interface, as an archive `cioverlap_exe.tar.gz`:

```
export CIOVERLAP=/path/where/cioverlap/executables/will/be/put
tar xf cioverlap_exe.tar.gz
cp cioverlap_exe/* $CIOVERLAP
```

The binaries are compiled on Debian 11, and linked against the ATLAS library as provided in the standard repository. All sources are provided under `external/cioverlap` and `external/od-cioverlap`. Please refer to the instructions in these packages if you wish to compile the programs by yourself.

The provided version of `cioverlap.od` provided is not parallel. The sources for an OpenMP-enabled version are also provided in `external/od-cioverlap/` (the `serial` directory contains the sources for the statically compiled binary). If needed, you can compile this version, and copy the relevant executable:

```
cd external/od-cioverlap
make
cp cioverlap $CIOVERLAP/cioverlap.od_omp
```

Please note that the name `cioverlap.od_omp` is hardcoded in Newton-X !

The OpenMP version will be called if `blasthread` is superior to 1.

7.2.2 cioverlap.od interface

- Synopsis:

```
cioverlap.od > cioverlap.out
cioverlap.od_smp > cioverlap.out
```

7.2.3 cioverlap interface

- Synopsis:

```
cioverlap $CIO_OPTIONS < cioverlap.input > cioverlap.out
```

where `$CIO_OPTIONS` corresponds to the option `cio_options` from the Newton-X configuration (see below).

- Command line option description (`cio_options`):

Command option	line	Description
-s mask_file		File with transmomin format, tells cioverlap which overlap matrix elements (Slater det. prescreening)
-a		Use file phases.old to patch phases of bras wavefunctions from the previous step
-b		Use file phases.old to patch phases of kets wavefunctions from the previous step
-o		Use file phases.old to patch phases of wavefunctions from the previous step
-t screeningthr		Double; threshold for the Slater determinant prescreening procedure
-e excitrnk		Integer; excitation rank for the Slater determinant prescreening procedure determinant pairs which are mutually more than excitrnk -excited will be omitted. For TDDFT the value -1 suppresses this screening and unnecessary generation of excitlistfile
-i inactive		Integer; number of orbitals which are never excited from (generally higher than ncore). Influences efficiency if omitted (or lower than correct value), while incorrectly high value will cause erroneous results.
-A from to		Integer; 0 (default): calculates overlap in the basis of Slater det. ; 1 (NOT SUPPORTED): calculates overlap in the basis of FULL CI GUGA wave functions.

7.2.4 Generation of CIS-like wavefunctions

For methods based on linear response or single-reference determinants, it is necessary to generate a wavefunction that will be used by the state-overlap programs. The preferred method in Newton-X is to use the **cis_casida** program, that will generate a Casida ansatz from the set of singles amplitudes.

In Newton-X **cis_casida** is called with the following command inside the **cioverlap** folder:

```
$CIOVERLAP/cis_casida $CISC_OPTIONS < cis_casida.inp > cis_casida.out
```

where **\$CISC_OPTIONS** corresponds to the content of the **cisc_options** setup in the configuration file. The command line switches are the following:

The command outputs a single file, **casidawf**, a binary file containing the resulting wavefunction. Usually, it is renamed as **eivectors1** to be used as the current wavefunction in **cioverlap**.

7.2.5 cioverlap options

Command option	line	Description
-o		orthonormalize the wave functions
-c		In TDDFT, use \mathbf{F} instead of $(\mathbf{X}+\mathbf{Y})$ to build the wave function. Note that the resulting wavefunction will not necessarily be orthonormal unless -o is used simultaneously.
-i <code>inactive_occ</code>		integer; neglect all excitations from the number of lowest orbitals in the TDDFT response function
-I <code>inactive_virtual</code>		integer; neglect all excitations to the number of highest orbitals in the TDDFT response function

Key	Description	Options	Default
blasthread	Number of CPU cores used by BLAS in CIOVERLAP programs. superior to 1 - Parallel execution	<ul style="list-style-type: none"> • 1 - Serial execution 	1
ci_cons	COLUMBUS: Computation of determinant overlap in the time-derivative coupling.	<ul style="list-style-type: none"> • 0 - Compute all determinant overlap terms • 1 - Neglect determinant overlap terms when the hank is too high or when the determinants are orthogonal 	1
cio_options	Options to be passed to cioverlap program		None
cisc_options	Options to be passed to cis_casida program		None
coptda	GAUSSIAN: Linear response vectors used in the evaluation of NAD.	<ul style="list-style-type: none"> • 0 - $X\rangle$ • 1 - $X+Y\rangle$ 	1
generate_wf	Generate a CIS-like wavefunction with "cis_casida" program	<ul style="list-style-type: none"> • 0 - Don't generate the CIS-like WF • 1 - Generate the CIS-like WF 	1
ncore	CIS Casida: Number of core orbitals that should be ignored.		0
ndisc	CIS Casida: Number of virtual orbitals that should be ignored.		0
ovl_prog	Program to use for computing the state overlap matrix	<ul style="list-style-type: none"> • 1 - cioverlap (based on Tully and Hammes-Schiffer) • 2 - cioverlap-od (orbital derivative method) • 3 - wfoverlap (from Plasser et al.) (NOT IMPLEMENTED) 	1

read_ovl_matrix File where the state overlap matrix is
printed (RELATIVE TO 'TEMP').

AUTO

Most of those options are set automatically with respect to the electronic structure program or method chosen. The easiest way to see the default values is to do a dry run with the `nx_moldyn` program with a basic configuration:

```
nx_moldyn -d > md_dry.out 2>&1
```

7.3 Couplings computed

Newton-X implements screening capabilities over to decide which couplings should be computed and taken into account. This is handled by control parameters that independently apply to the full list of $N_c = N_{stat}(N_{stat} - 1)/2$ couplings. The list of couplings to be computed is dynamically updated during the computation.

The keywords involved all belong to the `&nad_setup` section of the Newton-X configuration.

After creating a list of couplings to be computed, Newton-X will write a file `transmomin` that can be parsed by `cioverlap` and Columbus. It also stores internally a screening matrix, used to screen the final state-overlap matrix obtained with `cioverlap.od`.

7.3.1 nad_setup options

Key	Description	Options	Default
cascade	Control the computation of NAC between state of interest and states above / below	<ul style="list-style-type: none">• 0 - Compute NAC above AND below• 1 - Compute NAC below only	0
current	Compute NAC only for pairs of states including the current state.	<ul style="list-style-type: none">• 0 - Deactivate this screening• 1 - Activate	1
include_pair	Pairs of state for which the NACs should always be computed if current state is part of this pair. Should be a comma separated array 1,2,1,3 will always compute NAC S0-S1 and S0-S2 (when the system is in S0)		0
kross	Control the computation of NAC between non-consecutive states	<ul style="list-style-type: none">• 0 - Do not calculate nonadiabatic couplings between non-consecutive states• 1 - Calculate nonadiabatic couplings between non-consecutive states	1

never_state	Array of states for which the nonadiabatic coupling vectors should never be computed.		0
--------------------	---	--	---

Key	Description	Options	Default
cascade	Control the computation of NAC between state of interest and states above / below	<ul style="list-style-type: none"> • 0 - Compute NAC above AND below • 1 - Compute NAC below only 	0
current	Compute NAC only for pairs of states including the current state.	<ul style="list-style-type: none"> • 0 - Deactivate this screening • 1 - Activate 	1
include_pair	Pairs of state for which the NACs should always be computed if current state is part of this pair. Should be a comma separated array 1,2,1,3 will always compute NAC S0-S1 and S0-S2 (when the system is in S0)		0
kross	Control the computation of NAC between non-consecutive states	<ul style="list-style-type: none"> • 0 - Do not calculate nonadiabatic couplings between non-consecutive states • 1 - Calculate nonadiabatic couplings between non-consecutive states 	1
never_state	Array of states for which the nonadiabatic coupling vectors should never be computed.		0

7.3.2 Example

The combination of the three flags **kross**, **cascade** and **current** leads to eight different setups, namely: two-state model (TS), three-state model (3S), horizontal coupling (HC), lower diagonal (LD), partial coupling (PC), neighbour coupling (NC) and complete coupling (CC).

In table 14 we provide an example of the resulting couplings with each model (with (**kross**, **cascade**, **current**) ordering), when 5 states are considered ($N_{stat} = 5$), with a dynamics currently running on state 3 ($N_{dyn} = 3$).

Table 13: Couplings computed with each model (with `(kross, cascade, current)` ordering), when 5 states are considered ($N_{stat} = 5$), with a dynamics currently running on state 3 ($N_{dyn} = 3$).

Parameters	model	N_c	21	31	32	41	42	43	51	52	53	54
(0, 1, 1)	TS	1			x							
(0, 0, 1)	3S	2			x			x				
(1, 1, 1)	HC	$N_{dyn} - 1$		x	x							
(0, 1, 0)	LD	$N_{dyn} - 1$	x		x							
(1, 0, 1)	PC	$N_{stat} - 1$		x	x			x			x	
(0, 0, 0)	NC	$N_{stat} - 1$	x		x			x				x
(1, 1, 0)	LT	$(N_{dyn}^2 - N_{dyn})/2$	x	x	x							
(1, 0, 0)	CC	$(N_{stat}^2 - N_{stat})/2$	x	x	x	x	x	x	x	x	x	x

8 Outputs description

The outputs of Newton-X can be produced in two forms, depending on the compilation options:

- if compiled with the `USE_HDF5` flag, the output will consist in an HDF5 file in the H5MD format ;
- if not compiled with HDF5 support, or if the parameter `use_txt_outputs` is set to `.true.` (in the main configuration section `nxconfig` of `user_config.nml`), then text files will be produced.

All output is always written every `kt` step (see section 3.5).

8.1 Minimal output

By default, Newton-X will **always** write the following quantities (in brackets, the corresponding flag from the input, see below):

- the geometries (`prt_geom`);
- the velocities (`prt_veloc`);
- the gradients, for all states if available (`prt_grad`);
- the total, kinetic and potential energies (`prt_etot`, `prt_ekin` and `prt_epot` respectively).

8.2 Output for adiabatic dynamics with more than one state

When `nstat > 1`, Newton-X will also save:

- the oscillator strengths (`prt_osc`).

8.3 Output for non-adiabatic dynamics

In addition, Newton-X saves the following for non-adiabatic dynamics:

- the current dynamics state `nstatdyn` (`prt_nstatdyn`) ;
- the non-adiabatic couplings (`prt_nad`) ;
- the wavefunction (and electronic population) for each state (`prt_wf`) ;
- the surface hopping probabilities (`prt_shprob`) ;
- the surface hopping generated random numbers (`prt_shrx`) ;
- the surface hopping random seed (`prt_shseed`) ;

8.4 Output with complex surfaces (CS-FSSH)

In addition, Newton-X writes:

- the imaginary part of the non-adiabatic couplings, **only if** `prt_nad` is on, and if the `gamma_model` used is equal to 0 (see above, CS-FSSH) ;
- the resonance energy `gamma` (`prt_gamma`).

8.5 Method-specific outputs

8.5.1 Exciton models

For exciton models Newton-X will save:

- the diabatic Hamiltonian (`prt_diab_ham`) ;
- the diabatic energies (`prt_diab_en`) ;
- the diabatic population (`prt_diab_pop`) ;
- (only with MOPAC) the MOPAC information file (`prt_mopdyn`).

8.6 Supplementary data

Optionnally, Newton-X can also save:

- the linear momentum (`prt_linmom`) ;
- the angular momentum (`prt_angmom`).

All of these elements will be updated according to the value of `kt` (save the data only when the step is a multiple of `kt`), or when surface hopping occurs.

It is important to note, also, that the update of outputs is the last operation in every step of the dynamics !

8.7 Text files

In this section we describe the files that will be produced if text files are requested. We start by describing the default files, and the optional files will be indicated by the `lvprt` value required in the title of the corresponding section.

8.7.1 Note about parsing

Unless otherwise stated, all floating point numbers are printed with the Fortran format `F20.12`, the time is printed with `F10.3`, and step with `I20`. No space is reserved between fields. For instance, a line containing a time, a step, and `x`, `y` and `z` component of some observable, it will be printed with the Fortran format `'(F10.3,I20,3F20.12)'`.

8.7.2 geometries.xyz

This file will contain the evolution of the atomic coordinates during the dynamics (*i.e* the trajectory). It follows the `XYZ` file format specification, and can be directly loaded into any program that can read this format (VMD, Molden, ...). For each frame, the time and step number are indicated.

The file has the following organization:

```
Nr of atoms
time      step
At1       X(At1)   Y(at1)   Z(at1)
At2       X(At2)   Y(at2)   Z(at2)
At3       X(At3)   Y(at3)   Z(at3)
...
Nr of atoms
time      step
At1       X(At1)   Y(at1)   Z(at1)
At2       X(At2)   Y(at2)   Z(at2)
At3       X(At3)   Y(at3)   Z(at3)
...
```

8.7.3 velocities.dat

This file contains the list of the velocities of atoms along the trajectory. The format is the same as for `geometries.xyz`, except that the atom names are not indicated (we just have the `X`, `Y` and `Z` components of each velocities).

8.7.4 nad.dat

This file contains the non-adiabatic vectors. For each time step we save the following information:

- the current time and step ;

- the number of couplings and the number of atoms ;
- for each coupling and each atom, the x , y , z components of the vectors.

The couplings are saved in the following order:

1. state 2 / state 1 ;
2. state 3 / state 1 ;
3. state 3 / state 2 ;
4. ...

Overall the file has the following structure, for N atoms:

```

Time      Step
NCOUPLINGS  NATOMS
X(AT1, COUPL1)  Y(AT1, COUPL1)  Z(AT1, COUPL1)
X(AT2, COUPL1)  Y(AT2, COUPL1)  Z(AT2, COUPL1)
...
X(ATN, COUPL1)  Y(ATN, COUPL1)  Z(ATN, COUPL1)
X(AT1, COUPL2)  Y(AT1, COUPL2)  Z(AT1, COUPL2)
X(AT2, COUPL2)  Y(AT2, COUPL2)  Z(AT2, COUPL2)
...
Time      Step
NCOUPLINGS  NATOMS
X(AT1, COUPL1)  Y(AT1, COUPL1)  Z(AT1, COUPL1)
X(AT2, COUPL1)  Y(AT2, COUPL1)  Z(AT2, COUPL1)
...
X(ATN, COUPL1)  Y(ATN, COUPL1)  Z(ATN, COUPL1)
X(AT1, COUPL2)  Y(AT1, COUPL2)  Z(AT1, COUPL2)
X(AT2, COUPL2)  Y(AT2, COUPL2)  Z(AT2, COUPL2)
...
```

8.7.5 wf.dat

This file describes the wavefunction for each state, with the real and imaginary parts printed separately. It has the following structure:

```

Time      Step      Re(WF1)      Im(WF1)      Re(WF2)      Im(WF2) ...
```

8.7.6 energies.dat

This file contains the energy decomposition of the system, as well as the current state the trajectory runs on. If we are currently on state i , and we jump to state k , we will have the following structure:

```

Time_1  Step_1  i  Etot      Ekin      Epot(1)  Epot(2)  Epot(3) ...
Time_2  Step_2  k  Etot      Ekin      Epot(1)  Epot(2)  Epot(3) ...
```

8.7.7 populations.dat

This file contains the electronic populations for each state, and has the following structure:

```
Time Step Population(1) Population(2) Population(3) ...
```

8.7.8 seed.dat

This file contains the initial seed used for the random number generator initialization. Each element of the seed is written on a different line. The number of elements will be compiler and platform dependent.

8.7.9 sh_prob.dat

This file contains the transition probabilities (and optionnally, for `lvprt > 1`, the generated random numbers). The current time corresponds to the time of the classical trajectory, and the steps indicated refer to the micro-iteration steps (ranging from 1 to ms). It has the following structure (RNG is the generated random number at this step).

```
Time Step (RNG) Prob(1) Prob(2) Prob(3) ...
```

8.7.10 hopping_veloc.dat

This file contains the classical velocity (macro-iteration) before and after surface hopping, when a true hopping event occurs. We include the time of hopping, the step, as well as both velocities in the following way:

```
SURFACE HOPPING AT t= curr_time STEP step
  Velocity before hopping
  ...
```

```
  Velocity after hopping
  ...
```

```
=====
```

Such a section will be present for each hopping event.

8.7.11 hopping_veloc_micro.dat

This file contains the interpolated velocity (micro-iteration) before and after surface hopping. It has the same structure as `hopping_veloc.dat`.

8.7.12 gradients.dat (lvprt >=4)

This file saves the gradients along the trajectory, and has exactly the same structure as `nad.dat`.

8.7.13 osc_str.dat (lvprt >=4)

This file records the computed oscillator strenghts along the trajectory, and haas the following structure:

```
Time    Step    f(1->2)    f(1->3)    ...
```

8.7.14 linmom.dat (lvprt >=5)

This file contains the linear momentum along the trajectory, with the following shape:

```
Time      Total x    Total y    Total z    Norm
```

8.7.15 angmom.dat (lvprt >=5)

This file contains the angular momentum along the trajectory, with the following shape:

```
Time      Total x    Total y    Total z    Norm
```

8.8 Summary

Table 14: Summary of the outputs produced by Newton-X

Data	flag	Txt file	H5MD path
Geometries	prt_geom	geometries.xyz	particles/position
Velocities	prt_veloc	velocities.dat	particles/velocities
Gradients	prt_grad	gradients.dat	particles/gradients
Total energy	prt_etot	energies.dat	observables/total_energy
Kinetic energy	prt_ekin	energies.dat	observables/kinetic_energy
Potential energies	prt_epot	energies.dat	observables/potential_energy
Oscillator str.	prt_osc	osc_str.dat	observables/oscillator_strengths
Dynamic state	prt_nstatdyn	energies.dat	observables/nstatdyn
Non-adiabatic couplings	prt_nad	nad.dat	particles/nad
Wavefunctions	prt_wf	wf.dat	observables/wavefunctions
Electronic populations	-	populations.dat	observables/populations
SH hopping probabilities	prt_shprob	shprob.dat	observables/sh_probabilities
SH random number	prt_shrx	shprob.dat	observables/generated_random_number
SH random seed	prt_shseed	seed.dat	observables/random_seed
Imaginary NAD	prt_inad	inad.dat	particles/inad
Resonance energy	prt_gamma	gamma.dat	observables/gamma
Diabatic Hamiltonian	prt_diab_ham	diahham.dat	observables/diahham
Diabatic populations	prt_diab_pop	diapop.dat	observables/diapop
Diabatic energies	prt_diab_en	diapop.dat	observables/diaen
MOPAC info file	prt_mopdyn	exc_mop.dyn	-
Linear momentum	prt_linmom	linmom.dat	observables/linear_momentum
Angular momentum	prt_angmom	angmom.dat	observables/angular_momentum

References

- [1] BARBATTI, M., GRANUCCI, G., PERSICO, M., RUCKENBAUER, M., VAZDAR, M., ECKERT-MAKSIĆ, M., AND LISCHKA, H. The on-the-fly surface-hopping program system newton-x: Application to ab initio simulation of the nonadiabatic photo-dynamics of benchmark systems. *Journal of Photochemistry and Photobiology A: Chemistry* 190, 2-3 (2007), 228–240.
- [2] BARBATTI, M., PITTNER, J., PEDERZOLI, M., WERNER, U., MITRIĆ, R., BONAČIĆ-KOUTECKÝ, V., AND LISCHKA, H. Non-adiabatic dynamics of pyrrole: Dependence of deactivation mechanisms on the excitation energy. *Chemical Physics* 375, 1 (2010), 26–34.
- [3] BARBATTI, M., RUCKENBAUER, M., PLASSER, F., PITTNER, J., GRANUCCI, G., PERSICO, M., AND LISCHKA, H. Newton- <scp>x</scp> : a surface-hopping program for nonadiabatic molecular dynamics. *WIREs Computational Molecular Science* 4, 1 (2013), 26–33.
- [4] BUTCHER, J. C. A modified multistep method for the numerical integration of ordinary differential equations. *Journal of the ACM* 12, 1 (1965), 124–135.
- [5] CHEN, H.-T., AND REICHMAN, D. R. On the accuracy of surface hopping dynamics in condensed phase non-adiabatic problems. *The Journal of Chemical Physics* 144, 9 (2016), 094104.
- [6] DRAL, P. O., BARBATTI, M., AND THIEL, W. Nonadiabatic excited-state dynamics with machine learning. *The Journal of Physical Chemistry Letters* 9, 19 (2018), 5660–5663.
- [7] FABIANO, E., KEAL, T., AND THIEL, W. Implementation of surface hopping molecular dynamics using semiempirical methods. *Chemical Physics* 349, 1-3 (2008), 334–347.
- [8] FERRETTI, A., GRANUCCI, G., LAMI, A., PERSICO, M., AND VILLANI, G. Quantum mechanical and semiclassical dynamics at a conical intersection. *The Journal of Chemical Physics* 104, 14 (1996), 5517–5527.
- [9] GRANUCCI, G., AND PERSICO, M. Critical appraisal of the fewest switches algorithm for surface hopping. *The Journal of Chemical Physics* 126, 13 (2007), 134114.
- [10] GRANUCCI, G., PERSICO, M., AND ZOCCANTE, A. Including quantum decoherence in surface hopping. *The Journal of Chemical Physics* 133, 13 (2010), 134111.
- [11] HAMMES-SCHIFFER, S., AND TULLY, J. C. Proton transfer in solution: Molecular dynamics with quantum transitions. *The Journal of Chemical Physics* 101, 6 (1994), 4657–4667.

- [12] KELLY, A., AND MARKLAND, T. E. Efficient and accurate surface hopping for long time nonadiabatic quantum dynamics. *The Journal of Chemical Physics* 139, 1 (2013), 014104.
- [13] KOSSOSKI, F., AND BARBATTI, M. Nonadiabatic dynamics in multidimensional complex potential energy surfaces. *Chemical Science* 11, 36 (2020), 9827–9835.
- [14] LANDRY, B. R., FALK, M. J., AND SUBOTNIK, J. E. Communication: the correct interpretation of surface hopping trajectories: How to calculate electronic properties. *The Journal of Chemical Physics* 139, 21 (2013), 211101.
- [15] LANDRY, B. R., AND SUBOTNIK, J. E. How to recover marcus theory with fewest switches surface hopping: Add just a touch of decoherence. *The Journal of Chemical Physics* 137, 22 (2012), 22A513.
- [16] MAKRI, N. The linear response approximation and its lowest order corrections: an influence functional approach. *The Journal of Physical Chemistry B* 103, 15 (1999), 2823–2829.
- [17] NIKITIN, E. *The Theory of Nonadiabatic Transitions: Recent Development with Exponential Models*. Advances in Quantum Chemistry Volume 5. Elsevier, 1970, ch. 4, pp. 135–184.
- [18] PITTNER, J., LISCHKA, H., AND BARBATTI, M. Optimization of mixed quantum-classical dynamics: Time-derivative coupling terms and selected couplings. *Chemical Physics* 356, 1-3 (2009), 147–152.
- [19] PLASSER, F., GRANUCCI, G., PITTNER, J., BARBATTI, M., PERSICO, M., AND LISCHKA, H. Surface hopping dynamics using a locally diabatic formalism: Charge transfer in the ethylene dimer cation and excited state dynamics in the 2-pyridone dimer. *The Journal of Chemical Physics* 137, 22 (2012), 22A514.
- [20] REKIK, N., HSIEH, C.-Y., FREEDMAN, H., AND HANNA, G. A mixed quantum-classical liouville study of the population dynamics in a model photo-induced condensed phase electron transfer reaction. *The Journal of Chemical Physics* 138, 14 (2013), 144106.
- [21] RYABINKIN, I. G., NAGESH, J., AND IZMAYLOV, A. F. Fast numerical evaluation of time-derivative nonadiabatic couplings for mixed quantum-classical methods. *arXiv preprint arXiv:1505.04811* (2015), nil.
- [22] STOJANOVIĆ, L., BAI, S., NAGESH, J., IZMAYLOV, A., CRESPO-OTERO, R., LISCHKA, H., AND BARBATTI, M. New insights into the state trapping of uv-excited thymine. *Molecules* 21, 11 (2016), 1603.
- [23] SUBOTNIK, J. E., AND SHENVI, N. A new approach to decoherence and momentum rescaling in the surface hopping algorithm. *The Journal of Chemical Physics* 134, 2 (2011), 024105.

- [24] TULLY, J. C. Molecular dynamics with electronic transitions. *The Journal of Chemical Physics* 93, 2 (1990), 1061–1071.