MCTDH-X v2.22

Numerically Exact Many-Body Dynamics of Indistinguishable Particles

– User Manual –

Contact: mctdhx@ultracold.org

Download and Support: http://ultracold.org/forum

by Axel U. J. Lode and Marios C. Tsatsos

Abstract

This document is intended to guide any potential user to the installation and use of the MCTDH-X program package. MCTDH-X is a highly efficient numerical algorithm to solve the many-body interacting Schrödinger equation. The present manual describes the basic philosophy and structure of the program and the workflow of the MCTDH-X package. In particular, it explains how to operate and control the main program and the analysis routines from the respective input files.The output of the main program's numerical computations is visualized with bash scripts as images and video files with any desired plotting or data processing program. A description of the usage of predefined scripts is included in the present manual. Multiple computations, such as parameter scans can be automated with the provided "Monster" script. Finally, *Mercurial*, the version management system, is described and development and good programming guidelines are suggested.

Contents

List of Figures

List of Tables

1 Installation

1.1 Prerequisites for the main program

The prerequisites to install the MCTDH-X program are Fortran compilers (Intel, PGI or GNU). Fast Fourier transforms need to be provided by either Intel MKL or FFTW libraries. An FFTW source is included in the package and will be compiled during the installation if FFTW is selected. LAPACK routines also have to provided as a library – either within the Intel MKL, or, if selected an included OpenBLAS source will be compiled and linked. A version of MPI, which provides an MPI Fortran compiler wrapper, such as `mpif90`, is also a prerequisite. For the successful compilation of the included Mercurial version management, python development headers have to be installed on the system. In Debian-based Linux distributions just install the packages `gfortran`, `libopenmpi-dev`, `openmpi-common`, `openmpi-bin` with your package/software management. In Ubuntu, for instance, you may do the following in terminal:

```
sudo apt-get install gfortran libopenmpi-dev openmpi-common openmpi-bin
```

1.2 Prerequisites for full functionality

To make the MCTDH-X software fully functional, the following packages are needed: `python`, `libpython-dev`, `scons`, `gnuplot`, `mercurial`, `mplayer2`, `mencoder`. With these packages, the integrated software management of the MCTDH-X package, and its visualization (movie) scripts should be fully functional. In Ubuntu, for instance, you may type the following in terminal to install the prerequisites:

```
sudo apt-get install python libpython-dev scons gnuplot mercurial mplayer2 mencoder
```

1.3 Running the installation

After the prerequisites are fulfilled, to install MCTDH-X one has to first unpack the program package (this unpacking can be skipped when you download from the repository with `hg clone`):

```
tar -xvf mctdhx.tgz
```

Subsequently, the program can be installed by running the interactive installation script:

```
./Install_MCTDHX.sh
```

The script offers several configuration options, like different platforms (`Generic, mpif90, CrayXC40`) and compilers (`Intel, GNU, PGI,ftn`). On a new platform, a modification of the Makefile or scons script might nevertheless be necessary. If the `Generic` build option is selected, the build is performed with gfortran and the included OpenBlas and FFTW sources. The `mpif90` option will use either `mpif90` or `mpif90.openmpi` for the build and the `CrayXC40` option selects the `Makefile.hornet` for the Cray computer Hornet and uses the Cray Fortran compilers `ftn`. To get further information (if you are not at all familiar with GNU make and the build on your platform fails), contact and seek help at mctdhx@ultracold.org or in the forum http://ultracold.org/forum. The installation script creates a few aliases and appends to the `$PATH` environment variable. It creates a file `.mctdhxrc` and `.guantumrc` and appends to the `.bashrc` to source the `.mctdhxrc`, such that the new commands become available. Available commands/aliases are collected in the following table 2.

Alias	What it does
`cdm`	takes you to the MCTDH-X installation directory
`mcg.sh <X>`	MCTDH-X code grep. Script which parses the MCTDH-X code and scripts for the argument `<X>`
`data_miner.sh`	run an interactively configured analysis on computations in all subdirectories
`MCTDHX`	executes the the program (for interactive use, only. Most queueing systems prevent aliases from working and the executables then have to be copied and run in the current working directory)
`MCTDHX_analysis`	executes the the analysis program (for interactive use, only. Most queueing systems prevent aliases from working and the executables have to be copied and run in the current working directory)
`libcp`	copy the dynamic library `libmctdhx.so` to the current working directory
`inpcp`	copy the example inputs `MCTDHX.inp,analysis.inp` to the current directory
`bincp`	copies the executables of the main and analysis programs to the current directory.
`libmake`	will recompile the dynamic library `libmctdhx.so`
`x-make`	will recompile the whole program
`mctdhx_hg`	alias to the supplied mercurial distribution (can be used for version management and to update the program)*
`mctdhx_gnuplot`	supplied gnuplot program to visualize data*.
`mctdhx_mencoder`	supplied mencoder program to make movies from gnuplot images*.
`mctdhx_mplayer`	supplied mplayer program to view the movies*.

Table 2: Aliases and scripts provided with the MCTDH-X installation.
(aliases marked by a * are only created if respective included software is not present in the system)

2 An MCTDH-X Tutorial

A straightforward way of familiarizing youself with the usage of MCTDH-X, without further reading the remainder of this user manual, is to follow this step-by-step tutorial. The tutorial includes running the main program to compute a many-body eigenstate as well as to compute a time-evolution for bosons after changeing the potential. Subsequently, the analysis program is run to extract some quantities of interest that then are visualized with the included bash scripts to obtain videos of the computed dynamics.

2.1 Computing an Eigenstate by Relaxation

After the installation script has finished, you should make sure that the aliases and links are working. To do so, first type

```
source ~/.mctdhxrc
ls $MCTDHXDIR
```

. Now you should see the executables `MCTDHX_<compiler>`, `MCTDHX_analysis_<compiler>` and the library `libmctdhx.so`. If you don't, the installation did not terminate correctly and you should

check what went wrong (see the log files created in `./log/`) and contact the developers in case you cannot find out or fix the error.

To get started, it's best to create a directory for the tutorial computations:

```
mkdir ~/MCTDH-X-Tutorial
cd ~/MCTDH-X-Tutorial
```

. To copy all necessary files to run the computation in this directory, we make use of the alias `inpcp` and `libcp` which copy the example inputs `MCTDHX.inp`, `analysis.inp` and the dynamic library `libmctdhx.so` to the current directory.

```
libcp
inpcp
ls
```

. The `ls` command should show a list including `MCTDHX.inp`, `analysis.inp`, and `libmctdh.so`. With these three files, we're able to run the main and analysis programs. The default `MCTDHX.inp` is configured to compute the eigenstate of $N = 2$ bosons in a one-dimensional harmonic oscillator potential with unit frequency and with $M = 4$ orbitals. To make things a little more interesting, let's change the the number of bosons to $N = 50$ by editing `MCTDHX.inp` and setting `Npar = 50` in the `System_Parameters` namelist, i.e.,

```
gedit ./MCTDHX.inp
```

. Since we're going to compute some dynamics later, we use the opportunity to also enlarge the number of grid points and the grid extension in the `DVR_Parameters` namelist (a bit further down in the `MCTDHX.inp` file). We set:

```
Npar = 50
NDVR_X = 256
x_initial = -12.d0
x_final = 12.d0
```

. With these adjustments made, we can run the relaxation to the groundstate of the harmonic oscillator potential by typing

```
MCTDHX
```

. This should show an output similar to the following figure 1. This computation is going to take a minute so sit back and relax or just get a coffee, until it finished. The energy you should see on screen in the final propagation step should be identical to `259.14031953`. To visualize the output, let's use gnuplot:

```
mctdhx_gnuplot
plot "20.0000000orbs.dat" u 1:8, "" u 1:(sqrt($24**2)), "" u 1:(sqrt($22**2))
```

. The `plot` command visualizes the density $\rho(x)$ (column `8`) and the first two natural orbitals $\phi_1^{(NO)}(x)$ and $\phi_2^{(NO)}(x)$ (column `24` and `22`, respectively) in the last ASCII output file of the relaxation `20.0000000orbs.dat`. For a full explanation of the structure of this file, see table 14. In figure 2, you can see a screenshot of what your gnuplot output should look alike. To find out the occupations of the natural orbitals, let's have a look at the last line of the `NO_PR.out` file:

```
 Initialization::: M over N and N+M-1 over N   0.0000000000000000        23426.000000000698
 allocation ok for DVR_Parameters
 allocation ok for Matrix_Elements
 ####################################
 ####   INPUT READ AND DISTRIBUTED  ####
 ####################################
 ####################################
 ####    COEFFICIENTS INITIALIZED   ####
 ####################################
 ####################################
 ####        DVR INITIALIZED        ####
 ####################################
 ####################################
 # INTERPARTICLE POTENTIAL INITIALIZED #
 ######################################
 ######################################
 # PROCESS           0  IS THE MASTER PROCESS #
 ######################################
 ################################################################################

@@@@@@@@@@    @@@@@@@  @@@@@@@  @@@@@@@     @@@  @@@             @@@  @@@
@@@@@@@@@@@  @@@@@@@@  @@@@@@@  @@@@@@@@    @@@  @@@             @@@  @@@
@@! @@! @@!  !@@         @@!    @@!  @@@    @@!  @@@             @@!  !@@
!@! !@! !@!  !@!         !@!    !@!  @!@    !@!  @!@             !@!  @!!
@!! !!@ @!@  !@!         @!!    @!@  !@!    @!@!@!@!@  @!@!@!@!@  !@@!@!
!@!   ! !@!  !!!         !!!    !@!  !!!    !!!@!!!!  !!!@!@!!!    @!!!
!!:     !!:  :!!         !!:    !!:  !!!    !!:  !!!              !: :!!
:!:     :!:  :!:         :!:    :!:  !:!    :!:  !:!             :!:  !:!
:::      ::    ::: :::     ::     :::: ::    ::   :::              ::  :::
 :      :     :: :: :     :     :: :  :      :   : :              :   ::

 ################################################################################
 ##############################################
 ###### I USE THE FFTW FFT LIBRARY ##############
 ##############################################
 Memory allocation is ok
  Binary files properly opened
 Headers written to Binary file OLD_MCTDHB_COEF.inp
 Headers written to Binary file OLD_MCTDHB_ORBS.inp
================================================================================================================================================
 ==============================Propagation started
================================================================================================================================================
 Time   0.0000000000000000      Step  0.10000000000000001      Energy   298.83559308652258
 Time   5.0000000000000003E-002 Step  1.0000000000000000E-002 Energy   285.38512055116814
```

Figure 1: Output of an MCTDH-X computation in the shell.

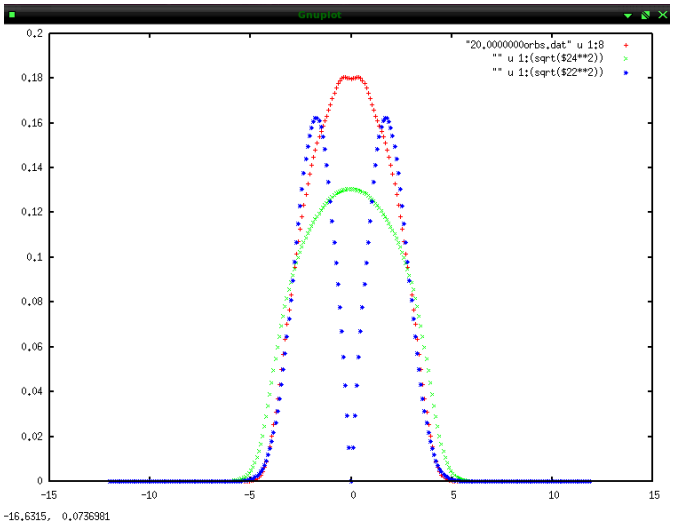

Figure 2: Visualization of MCTDH-X relaxation using gnuplot. The density $\rho(x)$ and the first two natural orbitals $\phi_1^{(NO)}(x)$ and $\phi_2^{(NO)}(x)$ are shown as red, green and blue line, respectively.

```
tail -n 1 NO_PR.out
```

. We get

```
20.00000000000031 0.5453108642221355E-02 0.1164565131824607E-01 0.2265201507726221E-01 0.9602492249622707 259.1403195319616
```

which is the time t, the natural occupations $\rho_M^{(NO)}(t), \rho_{M-1}^{(NO)}(t), ..., \rho_1^{(NO)}(t)$ and the energy of the system $E(t)$, in the final column (see also table 11 for the structure of the `NO_PR.out` file). We conclude that despite the strong interactions of $\lambda_0 = 1.0$, our eigenstate of the $N = 50$ bosons in the

harmonic confinement is close to condensed, since 96% of the bosons sit in the lowest single-particle state. In the single-well, this absence of fragmentation in the groundstate is anticipated. There is, however, numerous examples on the emergence of fragmentation in the dynamics of ultracold bosonic systems. To see the occurence of fragmentation, it's therefore instructive to change the potential in which our eigenstate was computed and trigger some dynamics.

2.2 Computing the Time-Evolution of a System

To propagate a given initial state in time, one has to change only a few parameters in the `MCTDHX.inp` file. To start, it's best to create a subdirectory for the propagation inside the `MCTDH-X-Tutorial` directory which contains the relaxation:

```
cd ~/MCTDH-X-Tutorial
mkdir propagation-double-well
```

. As indicated by the name, we're going to propagate the groundstate of the harmonic potential, which we computed in the previous subsection, in a double well potential. First, we copy the necessary files to the newly created directory:

```
cp MCTDHX.inp libmctdhx.so analysis.inp PSI_bin CIc_bin ./propagation-double-well/
cd propagation-double-well
```

. Subsequently, the input file needs to be adapted with a text-editor, i.e.,

```
gedit MCTDHX.inp
```

. To make the program propagate the initial state in the binary data files `PSI_bin` and `CIc_bin`, the following parameters have to be adapted:

```
Job_Prefactor=(0.d0,-1.d0)
GUESS='BINR'
Binary_Start_Time=20.0d0
```

as also explained in the inlined documentation in the `MCTDHX.inp` file and in table 3, `Job_Prefactor = (0.d0, -1.d0)` triggers the program to do a forward time propagation, `GUESS = 'BINR'` will make it read the initial state from the binary files `CIc_bin` and `PSI_bin`, and `Binary_Start_Time = 20.d0` chooses the time at which the binary files `CIc_bin` and `PSI_bin` are read for the initial state. Now, in order to define the parameters specifying the propagation in the double well, we change the following variables in the `MCTDHX.inp` :

```
Integration_Stepsize=0.0001d0
whichpot="h+d"
parameter1=2.d0
parameter2=8.d0
parameter3=1.d0
```

. `Integration_Stepsize=0.0001d0` specifies the initial time-step (compare table 3). The `which_pot = ''h+d''` variable selects a potential which is the sum of a displaced harmonic potential and a discplaced Gaussian barrier in its center. To define the potential `parameter1=2.d0` is the displacement, `parameter2=8.d0` specifies the height of the barrier in the center of the parabola and `parameter3=1.d0` gives the width of this barrier. Finally, we select the appropriate integrator to propagate the coeffcients' equations of motion:

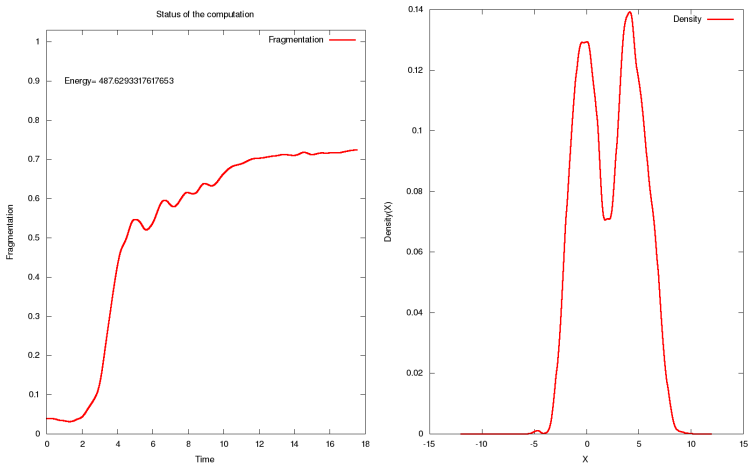

Figure 3: Checking the status of an MCTDH-X computation with `mctdhx_status.sh`. The left panel displays the time-evolution of fragmentation, i.e., $1 - \sum_{i=2}^{M} \rho_i^{(NO)}(t)$, in the system and the right panel shows a snapshot of the current density.

```
Coefficients_Integrator='MCS'
```

. Now the input is adjusted to start the propagation and we type

```
MCTDHX
```

to run it and wait until it finished. After it has finished, we can get a quick visualization of the final state by typing

```
mctdhx_status.sh
gnome-open status.png
```

. The script `mctdhx_status.sh` plots the time-evolution of the fragmentation in a computation as well as a snapshot of the current density. The picture you should see is displayed in Figure 3. Finally, we could invoke several different bash-scripts to visualize the computed time-evolution as movies (cf. table 9) or simply run all available scripts with the visualization master script, like so:

```
vms.sh all $PWD -7.0 9.0
```

(cf. also section 4.3). `vms.sh` automatically runs the analysis program on the data in the present directory and creates videos of the density as well as the correlation functions of the system. For reference, these are also available on the website at http://ultracold.org/documentation.

3 Program Structure

The MCTDH-X package contains a main program to perform the actual numerical task and an analysis program that is used to compute the desired quantities of analysis from the many-body wavefunction. For the purpose of visualization, bash scripts that generate .mpg or .avi video files by running gnuplot and mencoder are provided. A source tarball of gnuplot and mencoder are provided with MCTDH-X in the `External_Software` subdirectory and installed with the installation script,

as mentioned above. To automate and simplify the use of the main program, the analysis program as well as the visualization bash scripts, the program package also contains a graphical user interface (Guantum), as well as scripts that automate series of computations (`MonsterScript.sh`) and a script for the automation of data processing (`data_miner.sh`).

3.1 Main Program

The main program can be run by typing `MCTDHX` (in wrappers or in runscripts the `MCTDHX` alias might not work, and one needs to use `MCTDHX_intel`, `MCTDHX_pgf` or `MCTDHX_gcc`). Although, in the case of a computationally intense task, it is a lot faster to use the shared memory and distributed memory parallelization of the program and run it with one the various instances of MPI launchers (such as `mpirun,mpiexec, mpiexec.hydra,aprun,...`). Examples for running MCTDHX in parallel can be found in the example PBS scripts directory `PBS_Scripts`. Since the configuration of such an hybridly parallel job is complicated, it is easier to just use the `MonsterScript.sh` script that will automate whole series of hybridly parallel computations. They can be found in the `./Computation_Scripts` directory. If a manual configuration of a task is needed or desired, this is done by adapting one of the PBS runscript examples. Depending on the hardware architecture, the most efficient way is usually to run MCTDH-X with at least as many MPI processes as there are orbitals. The OpenMP shared memory parallelization takes care of efficiently performing the computational task inside of each MPI process. The program's structure is entirely modular, i.e., all subroutines are collected in Fortran modules. To inspect the program structure, please consult the html documentation by opening the `index.html` file in the `documentation/html` subdirectory. In this html documentation all important variables are explained and for each routine call- and caller-graphs are given.

3.2 Analysis Program

The analysis program can be run by typing `MCTDHX_analysis`. The analysis program is serial, i.e., no shared or distributed memory parallelization is used so far. The program is also entirely modular and the structure, call- and caller graphs, as well as a documentation of important variables is available in the same html documentation as for the main program (see file `documentation/html/index.html`).

3.3 Scripts

In the subdirectory `./bin/Scripts` there is the `MonsterScript.sh` script that can configure whole series of computations. The input file `ParameterScan.inp` is in the `./Input_Examples` directory and contains an in-line documentation. `MonsterScript.sh` allows a scan of 5 user-defined parameters for the relaxation, and 5 user-defined parameters for the subsequent propagations. A parameter scan for potential, DVR, particle or orbital number, and magnitude and/or width of the interaction can be configured, for instance. The configuration of `MonsterScript.sh` is detailed in table 10. The directory `./Computation_Scripts` contains the visualization master script, `vms.sh`, that automatically runs the MCTDHX analysis program and visualizes a given computation. The possible different visualization videos and plots are realized as bash scripts which can be found in the `./Visualization_Scripts` directory.

4 Input File driven Usage MCTDH-X

The basic workflow of a numerical solution of the time-dependent many-boson Schrödinger equation, $\hat{H}|\Psi\rangle = i\partial_t|\Psi\rangle$ (TDSE), with MCTDH-X is the following:

1. Define the Hamiltonian $\hat{H}$.
   Modify `Get_InterParticle_Potential.F` and `Get_1bodyPotential.F`.

2. Define the initial state $|\Psi\rangle$.
   Modify `Get_Initial_Coefficients.F` and `Get_Initial_Orbitals.F`.

3. Solve the TDSE by propagating in real or imaginary time.
   `libmake`, then `libcp`, (maybe `inpcp`, modify `MCTDHX.inp`),finally `MCTDHX`.

4. Analyze and visualize the solution $|\Psi\rangle$.
   (maybe `inpcp`, modify `analysis.inp`), run `MCTDHX_analysis`, process ASCII data with `gnuplot`.

Steps (1) through to (3) are done with the main program and step (4) is done with the analysis program and movie bash scripts. There is several ways of automatization of the above 4-step scheme. As mentioned above, there is a scripts called `MonsterScript.sh` in the `Computation_Scripts` directory that allows a fully integrated and automated processing of whole series of computations. Finally, the data processing to videos is automated in the visualization master script `vms.sh`.

4.1 Defining the Hamiltonian

The Hamiltonians that can be treated by the MCTDH-X package in its current implementation are those with maximally two-body operators. In general such a Hamiltonian hence contains the kinetic energy $\hat{T}$ and the external potential $\hat{V}$ and an interparticle interaction $\hat{W}$:

$$\hat{H} = \sum_{i=1}^{N}(\hat{T}_i + \hat{V}_i) + \sum_{i<j=1}^{N} \hat{W}_{ij}. \tag{1}$$

To fully define this Hamiltonian in dimensionless units [see Phys. Rev. A 77, 033613 (2008)] one has to define the one-body potential $\hat{V}$, and the two-body interaction $\hat{W}$. It has to be stressed, that both operators can be in principle time-dependent.

Specifying the one-body potential $\hat{V}$:

The one-body potential $\hat{V}$ is specified in the file `source/ini_guess_pot/Get_1bodyPotential.F`. This file contains a Fortran subroutine with a case selection for the dimensionality of the treated problem. The potentials defined in this routine can be selected and modified with the input parameters `whichpot, parameter1, parameter2, ..., parameter30` in `MCTDHX.inp`. Appendix A collects the predefined potentials and parameters. If a custom potential is desired, it has to be implementated in the file `source/ini_guess_pot/Get_1bodyPotential.F` (look there for the loops which are enclosed by the `if`- exceptions for `whichpot .eq. custom1D/custom2D/custom3D`, respectively).

Specifying the two-body potential $\hat{W}$:

The two-body interaction $\hat{W}$ is specified in the file `Get_InterParticle_Potential.F` in the directory `source/ini_guess_pot/`. This file contains a single Fortran subroutine with a case selection on the type of interparticle interaction and the evaluation of its action (`Interaction_Type` is the corresponding input variable). The standard types are a contact interaction potential (`Interaction_Type=0,6`) or short range Gaussian interaction (`Interaction_Type=1,2,3,4,5`). For these cases of a short-range interparticle interaction, the width of it can be adjusted with the input variable `Interaction_Width` in `MCTDHX.inp`. Generally, a non-contact interaction requires `Interaction_Type > 0`. `Interaction_type=5` is a time-dependent non-contact interaction and `Interaction_type=6` allows the simulation of contact interactions with a time-dependent interaction strength. The different evaluation types that `Interaction_Type` selects and the necessary properties of the interaction potentials are also specified in the html documentation and the example input in the directory `Input_Examples`. If a custom interaction potential is desired, this has to be implemented in the `Get_InterParticle_Potential.F` routine. The table in Appendix B shows the predefined interparticle interaction potentials as well as how to configure them with the input file's parameters.

Defining the initial state $|\Psi\rangle$:

The MCTDH-X wavefunction is a multiconfigurational expansion with time-dependent coefficients $C_{\vec{n}}(t)$ and time-dependent configurations $|\vec{n};t\rangle$. In order to fully define the wavefunction,

$$|\Psi(t)\rangle = \sum_{\{\vec{n}\}} C_{\vec{n}}(t)|\vec{n};t\rangle, \tag{2}$$

one has to define all coefficients $C_{\vec{n}}$ and all configurations $|\vec{n};t\rangle$. In the case of a fully user-defined guess, i.e., `GUESS='HAND'` in the input file, the wavefunction is supplied via Fortran routines. The coefficients are defined in the file `source/ini_guess_pot/Get_Initial_Coefficients.F` and the orbitals from which the configurations are built are defined in the file `Get_Initial_Orbitals.F` in the directory `source/ini_guess_pot/`. Both contain a single Fortran subroutine, that assigns the corresponding array (for a documentation of the routines, please see the html documentation). For a relaxation, i.e., the calculation of a certain eigenstate of the many-body Hamiltonian $\hat{H}$, usually it is a good choice to start from a Gross-Pitaevskii, i.e., single-configurational state with $|\Psi\rangle = |N,0,0,...\rangle$. For a propagation there is also the possibility to restart the computation from a previous one by specifying `GUESS='BINR'` in the input file.

4.2 Running the computation and analysis

In this subsection, it is specified how to run the MCTDH-X program and analysis manually, i.e., by configuring the `MCTDHX.inp` and `analysis.inp` input files and running the respective programs. To read how to automate this process using the `MonsterScript.sh` and `vms.sh` scripts, please consult the next subsection 4.3.

The input of the main program

To run the MCTDH-X program, it is best to have each calculation done in a separate directory. The program needs two files to perform the computation. First, the dynamic library, `libmctdhx.so`, and second, the input file with computational/numerical parameters `MCTDHX.inp`. The library contains the above four subroutines that define the Hamiltonian and the initial state (`Get_Initial_Coefficients.F`, `Get_Initial_Orbitals.F`, `Get_InterParticle_Potential.F`, `Get_1bodyPotential.F`). **If any of these files was modified, the library has to be recompiled and copied to the computation's directory!** Recompiling the library `libmctdhx.so` is conveniently achieved by issueing the command:

```
libmake
```

After compilation, the library can be copied to the working directory by typing

```
libcp
```

Finally, the input files should be copied to the working directory. This can be done by typing

```
inpcp .
```

The two files `MCTDHX.inp` and `analysis.inp` are now inside the working directory. The details of the input variables in those files can be taken from the in-line documentation in the input file or the html code documentation. The `MCTDHX.inp` file basically contains variables like the particle number, the number of orbitals, the (dimensionless) interaction strength, number of primitive basis functions, their type, details of the integration (integrator,stepsize,accuracy,order,...), and many more. See table 3 for all currently available parameters in the main program and their meaning.

System Parameters Namelist		
Parameter	*Meaning*	*Options*
JOB_TYPE	Select problem, i.e. MCTDH for bosons, MCTDH for fermions or TDCI for bosons.	Character, 'BOS', 'FER', 'FCI'; Default 'BOS'
Morb	Select number of time-dependent, variationally optimized basis functions	Integer, no default.
Npar	Select number of structureless particles	Integer, no default.
xlambda_0	Adjust prefactor of two-body potential in the Hamiltonian	Real, no default.
mass	Mass of the particles	Real, default 1.d0
Job_Prefactor	Select which direction to propagate the equations of motion in time.	Complex, (0.0d0,-1.0d0) Forward propagation; (0.0d0,+1.0d0) Backward propagation, (-1.0d0,0.0d0) (Improved) Relaxation; Default (-1.d0,0.d0).
`NProjections`	Number of times that the projection operator $\hat{\mathcal{P}}$ is applied to the right-hand side of the orbital equations of motion	Integer, default 2.

GUESS	Specifying if the initial guess is defined in .dat files, the routines `Get_Initial_Orbitals.F`, `Get_Initial_Coefficients.F`, or in the binary files `CiC_bin` and `Psi_bin`	Character, 'HAND' means the `Get_Initial...` routines are used, 'DATA' means .dat files will be read in, 'BINR' means `*_bin` files will be read in, default 'HAND' .
Diagonalize_OneBodyh	Select to use eigenfunctions of the one-body potential in `Get_1bodyPotential.F`	Logical, .T. or .F., .T. only non-FFT[1]-DVRs[2], i.e., DVR_X/Y/Z≠4, default is .F.
Binary_Start_Time	Define point in time at which the wavefunction is read from binary files in the case of `GUESS='BINR'`	Real, no default.
Restart_State	If restarting from a Block Davidson computation, this selects which state in the block is used as the initial value.	Integer, default 1
Restart_Orbital_FileName	Define filename of orbitals to read if `GUESS='DATA'`	Character, no default.
Restart_Coefficients_FileName	Define filename of coefficients to read if `GUESS='DATA'`	Character, no default.
Vortex_Seeding	In propagations: Multiply initial orbitals with phase/density profile?	Logical, default .F.
Vortex_Imprint	In relaxations: Apply a projection operator to a certain orbital density profile?	Logical, default .F.
Profile	If Vortex_Imprint is true, this will select the respective shape of the imprinted (select 'tanh','poly','phase', or 'phase-x' or define custom profile in Get_Vortex_Profile.F in /source/ini_guess_pot/)	Character, default 'tanh'
Fixed_LZ OrbLz	Multiply fixed phase profile on the orbital in relaxations How many times 2π the phase is going to jump if Fixed_LZ is true. If any value is -666, the respective orbital is unchanged.	Logical, default .F. Integer, default 0,0,0,0,0,0,0,0,0,0
DVR Namelist		
DIM_MCTDH	Specifying the dimensionality of the problem	Integer, 1,2 or 3, default 1.
NDVR_X	Specifying the number of DVR functions in X dimension	Integer, default 256.

[1]Fast Fourier Transform
[2]Discrete Variable Representation

NDVR_Y	Specifying the number of DVR functions in Y dimension	Integer, default 1.
NDVR_Z	Specifying the number of DVR functions in Z dimension	Integer, default 1.
DVR_X	Which DVR will be used in X direction	Integer, 1 means harmonic oscillator DVR, 3 means sine DVR, 4 means FFT DVR and 5 exponential DVR, default 4.
DVR_Y	Which DVR will be used in Y direction	Integer, 1 means harmonic oscillator DVR, 3 means sine DVR, 4 means FFT DVR and 5 exponential DVR, default 4.
DVR_Z	Which DVR will be used in Z direction	Integer, 1 means harmonic oscillator DVR, 3 means sine DVR, 4 means FFT DVR and 5 exponential DVR, default 4.
x_initial	Where the spatial grid in X dimension starts	Real, default -8.0.
x_final	Where the spatial grid in x dimension stops	Real, default 8.0.
y_initial	Where the spatial grid in Y dimension starts	Real, default -8.0.
y_final	Where the spatial grid in Y dimension stops	Real, default 8.0.
z_initial	Where the spatial grid in Z dimension starts	Real, default -8.0.
z_final	Where the spatial grid in Z dimension stops	Real, default 8.0.
Integration Namelist		
Time_Begin	Time at which the simulation shall start	Real, default 0.0.
Time_Final	Time at which the simulation shall stop	Real, default 20.0.
Time_Max	Maximal time	Real, default $1.d99$.
Output_TimeStep	Times at which orbital output shall be written.	Real, default 0.1.
Output_Coefficients	Multiple of Output_TimeStep at which coefficient output shall be written	Real, default 1
Integration_Stepsize	Stepsize of the integration scheme (for relaxation this is fixed, for propagation this is adaptive)	Real, default 0.01
Error_Tolerance	Error tolerance for the integration scheme	Real, default $1.d-9$

Minimal_Occupation	Minimal occupation for not considering an eigenvalue as 0 in the inversion of the density matrix elements	Real, default $1.d-12$
Minimal_Krylov	Minimal size of Krylov basis for coefficients' integration	Integer, default 4
Maximal_Krylov	Maximal size of Krylov basis for coefficients' integration	Integer, default 20
Orbital_Integrator	Integrator for the orbital equations of motion	Character, 'ABM' or 'OMPABM' means that (OpenMP parallelized) Adams Bashforth Moulton predictor corrector integrator is used, 'BS' means Bullirsch-Stör, 'RK' means Runge-Kutta, and 'STIFF' means ZVODE specialized to stiff equations is used; default 'OMPABM'.
Orbital_Integrator_Order	Order of the integration of the orbitals' integrator, choice depends on the Integrator	Integer, for 'RK' it is 5 or 8, for 'ABM'/'OMPABM' it is 2 to 8, for 'BS' its 2 to 16, and for 'STIFF' it is 1 or 2, default 7.
Orbital_Integrator_MaximalStep	Restricts the maximum stepsize in the orbital equations' integration	Real, default 0.01.
Write_ASCII	Specifying wether ASCII files are output during the computation	Logical, default .T..
Error_Rescale_TD	Specifying a scale for the Error_Tolerance parameter for time-dependent one-body potentials.	Real, default 1.0.
LZ	Trigger one-body angular momentum operator in the Hamiltonian	Logical, .T. or .F., default .F. .
OMEGAZ	Prefactor of one-body angular momentum operator.	Real, default 0.0.
STATE	Which eigenstate of the Hamiltonian to compute.	Integer, 1 means the groundstate, default 1. Stable for Coefficients_Integrator='DSL' or 'DAV' .

Coefficients_Integrator	Which integrator to use for the coefficients equations of motion.	Character, 'MCS' means MCTDH SIL routine, 'DSL' means MCTDH SIL diagonalization routine, 'DAV' means Davidson diagonalization, 'BDV' means block Davidson diagonalization; no default.
BlockSize	If Block Davidson (BDV) is used as integrator for the coefficients then this selects the number of coefficient vectors in the block	Integer, default 4
Olsen	If Block Davidson is used as integrator for the coefficients, this flag toggles the Olsen correction (Jacobi-Davidson)	Logical, default .F.
RLX_Emin	If Block Davidson is used as integrator for the coefficients then this selects the lower energy bound for the block	Real, default -1.d90
RLX_Emax	If Block Davidson is used as integrator for the coefficients then this selects the upper energy bound for the block	Real, default 1.d90
Potential Namelist		
whichpot	Select from predefined list of potentials, see table 20.	Character, no default.
parameter1	Parameters to tune predefined potentials, see table 20.	Real, default 1.0.
parameter2-30	Parameters to tune predefined potentials, see table 20.	Real, default 0.0.
Interaction Namelist		
Which_Interaction	Select predefined interparticle interaction potentials	Character, default 'gauss'; For time-dependent interactions, 'TDHIM', 'TDgauss1' a Gaussian sinusoidially modulated amplitude, and 'TDgauss2', a Gaussian with sinusoidially modulated width, are currently defined.
Interaction_Width	Modify parameter in the interparticle interaction.	Real, default 0.15.
interaction_parameter1-10	Parameters to tune predefined interaction potentials, see table B.	Real, default 0.0.

Interaction_Type	How the Local interaction potentials $\hat{W}_{sl}$ and their action is evaluated	Integer, 0 means δ-like contact interaction potential, 1 to 4 will use the routine `Get_InterParticle_Potential.F` to generate the interaction potential. 1 means the potential is separable and hence one potential vector is allocated for each spatial dimension; 2 means the potential depends on the distance of the particles only, hence, a tridiagonal representation is used (only for aequidistant DVRs 3,4 and 5); 3 means full interaction matrix will be stored (very large array!!!), 4 interaction matrix evaluated with successive FFT (IMEST); 5 means time-dependent interaction with IMEST; 6 means time-dependent contact interaction; 7 means both contact and non-zero ranged interactions with IMEST; default 0.

Table 3: MCTDH-X main program input file parameters.

Special parameters to treat multi-level atoms and spinors

In order to treat particles that have internal structure, like multileveled atoms or spinor particles, a set of special input variables has been introduced to the above `System Parameters Namelist`. These define the number of levels, the presence of a conical intersection and whether the interparticle interaction contains a spin-dependent part or not. In principle, a conical intersection amounts to off-diagonal terms in the one-body Hamiltonian $\hat{h}_i^{(CI)}$ which couple the different levels whereas a spin-dependent interparticle interaction means that there is two-particle interactions $\hat{W}_{\rm spin}$ that can change the spin of particles. These terms read as follows:

$$\hat{h}_i^{(CI),kk} = \left(\hat{T}_i + V_{kk}(\hat{\vec{r}}_i)\right) \otimes \mathbf{1}_{\vec{r}}; \qquad \hat{h}_i^{(CI),kj} = V_{kj}(\vec{r}_i)\hat{\pi}_{jk} \tag{3}$$

$$\hat{W}_{\rm spin} = \sum_{\nu=x,y,z} \left(\mathbf{S}^\nu \otimes \mathbf{1}_{\vec{r}}\right) \hat{W}(\vec{r},\vec{r}';t) \left(\mathbf{S}^\nu \otimes \mathbf{1}_{\vec{r}'}\right). \tag{4}$$

Here, the operator $\hat{\pi}_{jk}$ was defined, which makes level j appear in the coordinate space of level k for the spinor orbital on its right. Furthermore, the representation $\mathbf{S}^\nu$ was introduced for the general spin operators in $\nu = x, y, z$ direction. For problems including spin-orbit interactions the folloing Hamiltonian $\hat{h}_{SO}$ is added to the one-body Hamiltonian:

$$\hat{h}_{SO} = \gamma\left[\alpha\left(\hat{p}_x\mathbf{S}^y - \hat{p}_y\mathbf{S}^x\right) + \beta\left(\hat{p}_x\mathbf{S}^y + \hat{p}_y\mathbf{S}^x\right)\right]. \tag{5}$$

The parameters α, β, γ are the prefactor of the Rashba spin-orbit term, Dresselhaus spin-orbit term and the overall strength of the spin-orbit coupling (see also input parameters below). Generally, when the above terms are present in the Hamiltonian, a transfer of population between the different levels of the treated particles is allowed. The input variables necessary to control the program through the `MCTDHX.inp` file in the case of multileveled or spinor particles are specified in the following table 4.2.

System Parameters Namelist		
Parameter	*Meaning*	*Options*
`NLevel`	How many levels do the considered particles have?	Integer, default 1.
`Multi_level`	Do the atoms have internal structure?	Logical, default `.F.`
`Conical_Intersection`	Does the one-body Hamiltonian contain terms $V_{jk}(\vec{r})$ that couple different internal states?	Logical, default `.F.`. If set to `.T.`, the potential `VTRAP_EXT` which is defined in `Get_1bodyPotential.F` contains one additional vector that stores V_{jk}
`InterLevel_InterParticle`	Does the interparticle interaction couple different internal states directly (attention: this is for multileveled atoms which are *NOT* spinors	Logical, default `.F.`
`xlambda<X>`	Interparticle-intra-level interaction strength for non-spinors; `<X>=1,2,3`	Real, default `0.d0` .

`xlambda12`	Interparticle-inter-level interaction strength for non-spinors;	Real, default `0.d0` .
`Spinor`	Are the treated atoms spinors with a spin-dependent interparticle interaction	Logical, default `.F.`
`Lambda<X>`	spin-independent (`<X>=1`) and spin-dependent (`<X>=2`) interparticle interaction strength, respectively, for each level.	Real array, dimension 10, default `0.d0` .
`SpinOrbit`	Toggles inclusion of spin-orbit-interaction in the Hamiltonian	Logical, default .F.
`Rashba_Prefactor`	Magnitude of Rashba-spin-orbit-interaction	Real, default `0.d0`
`Dresselhaus_Prefactor`	Magnitude of Dresselhaus-spin-orbit-interaction	Real, default `0.d0`
`SpinOrbit_Prefactor`	Magnitude of total spin-orbit (Rashba + Dresselhaus) term	Real, default `0.d0`

It is important to note that it is necessary to specify at least as many one-body potentials as there are levels or spinor components in the treated particles. In the case of atoms featuring a conical intersection, the number of potentials is `NLevel + 1`. This is done in the routine `Get_NLevelPotentials` in the `Get_1bodyPotential.F` source file. The available predefined multilevel potentials are collected in the following table 4.2.

`whichpot`	Description	Potential	Parameters
`HO1D`	Parabolic potentials with different frequencies and offset for different spin components or levels. This potential is defined for two-level or spin-$\frac{1}{2}$ atoms, only.	$V_\uparrow(x) = \frac{1}{2}p_1^2x^2$; $V_\downarrow(x) = \frac{1}{2}p_2^2(x-p_3)^2 + p_4$	p_1, p_2 are the frequencies of the $\uparrow,\downarrow$ components/levels, respectively. p_3 is the horizontal displacement of the two parabolas and p_4 their relative offset.
`linearZ1D`	Parabolic optical confinement with linear Zeeman shift and a spatially homogeneous magnetic field in one dimension.	$V_{m_F}(x) = \frac{1}{2}p_1^2x^2 + m_F p_2\|x\|$	p_1 is the frequency of the optical confinement and p_2 defines the magnetic field strength.

Special parameters to treat ultracold atoms in an optical cavity

To deal with a system of bosons in interaction with an optical cavity, i.e., a field of photons which in turn generates a one-body potential for the atoms through the dipole force, special input parameters have been defined. These are listed in the following table 4.2.

System Parameters Namelist		
Parameter	*Meaning*	*Options*
`Cavity_BEC`	Toggle cavity treatment	Logical, default `.F.`
`NCavity_Modes`	How many modes to take into account treating the cavity	Integer, default `1`
`Cavity_PumpRate`	Rate, at which the laser is pumping the cavity (through the atoms)	Real, default `0.d0`
`Cavity_LossRate`	Rate, at which photons are lost from the cavity	Real, default `0.d0`
`Cavity_K0`	Magnitude of the wave-vector defining the resonance frequency of the cavity	Real, default `0.d0`
`Cavity_AtomCoupling`	Coupling strength of the atomic resonance to the pumping laser frequency	Real, default `0.d0`
`Pump_Switch`	Is the pumping laser intensity ramped up exponentially, kept constant and then ramped down exponentially?	Logical, default `.F.`
`RampupTime`	Time over which the pump laser intensity is increased linearly up to `Cavity_PumpRate`.	Real, default `0.d0`
`RampdownTime`	Time over which the pump laser intensity is kept constant at `Cavity_PumpRate`.	Real, default `0.d0`
`PlateauTime`	Time over which the pump laser intensity is decreased linearly to `0.d0`.	Real, default `0.d0`
`Pump_Oscillate`	Does the pump power oscillate [as $\epsilon \sin(\omega_p t)$] when it reached the plateau in the exponential ramping procedure?	Logical, default `.F.`
`Pump_Amplitude`	Amplitude ϵ of the oscillation of the pump power in units of `Cavity_PumpRate`.	Real, default `0.d0`
`Pump_Period`	Period ω_p of the oscillation of the pump power.	Real, default `0.d0`.
`X_Cavity_Pump_Waist`	Gaussian envelope's width for pump laser for two-dimensional systems	Real, default `0.d0`
`Cavity_Mode_Waist`	Gaussian envelope's width for cavity mode in two-dimensional systems	Real, default `0.d0`
`Error_Rescale_Cavity`	Parameter in the integration namelist to rescale the error tolerance of the integrator of the cavity equation of motion	Real, default `1.d0`

Special parameters to treat atoms in optical lattices

MCTDH-X offers two ways of dealing with atoms in optical lattices. Lattice Hamiltonians can be treated using an exact diagonalization treatment for one-, two-, and three-dimensional lattices. Especially, in the two- and three-dimensional cases, the dimensionality of the Hilbert space and the matrix that has to be diagonalized in the exact diagonalization approach is exploding rapidly and the problem size can no longer be handled. For the larger systems, one has to use the so-called MCTDBH approach, where the Bose-Hubbard Hamiltonian is expanded in a multi-configurational basis with a number of effective single-particle states than there is lattice sites. With this approach, a systematic improvement beyond mean-field theories such as the discrete non-linear Schrödinger equation is possible. Contrary to the approaches like time-evolved block decimation, matrix product states or time-dependent density-matrix renormalization group, MCTDBH can provide accurate predictions in time and for higher dimension than $D = 1$, because the partitioning of the Hilbert space is done by the variational principle and *not artificially introduced*. The parameters to select the former exact diagonalization or the latter MCTDBH approach are collected in the following table 4.2.

System Parameters Namelist		
Parameter	*Meaning*	*Options*
`Bose_Hubbard`	Do an exact diagonalization of a Bose Hubbard Hamiltonian? Attention: Has to be used together with `JOB_TYPE = 'FCI' !`	Logical, default `.F.`
`Periodic_BH`	Has the treated lattice system periodic boundary conditions?	Logical, default `.F.`
DVR Parameters Namelist		
`DVR_<I>`	If `DVR_<I> = 6`, MCTDHB is applied to a lattice of that many sites in `I = X/Y/Z` direction. Attention: has to be used together with `JOB_TYPE = 'BOS' !`	Integer, default `4`.

It is important to note that in the case that an exact diagonalization of a Bose-Hubbard Hamiltonian is done, the one-body potential offset is obtained from the routine `Get_BH_Offset`. If MCTDHB applied to the Bose-Hubbard Hamiltonian the default routine `Get_1bodyPotential` is used to compute the on-site potential energy offset. Both routines can be found in the file `./source/ini_guess_pot/Get_1bodyPotential.F` .

The input of the analysis program

The `analysis.inp` file contains the desired quantities of analysis and specifies, for which points in time these are needed. and table 8 for all currently available parameters in the analysis program and their meaning.

Parameter	Meaning	Options
ZERO_body		
Time_From	Time from which to start analysis.	Real, no default.
Time_to	Time at which to stop analysis.	Real, no default.
Time_Points	Time points in analysis.	Integer, no default.
Total_Energy	Comupte the total, kinetic, potential and interaction energies	Logical, default .F.
Orbitals_Output	Create ASCII orbital output.	Logical, default .T.
FTOrbitals_Output	Create ASCII Fourier-transformed orbital output.	Logical, default .F.
Coefficients_Output	Create ASCII coefficients output.	Logical, default .T.
MatrixElements_Output	Output of reduced one-body and two-body density matrix elements.	Logical, default .F.
HamiltonianElements_Output	Output of one-body and two-body Hamiltonian matrix elements.	Logical, default .F.
GetA	Partial sums on the two-body Hamiltonians' matrix elements	Logical, default .F.
Dilation	Factor to dilate real space in order to obtain better k-space resolution (Only touched if AutoDilation is false)	Integer, default 1
AutoDilation	Toggle if threshold Kdip is used to evaluate optimal dilation for FFTs (only for one-dimensional computations!)	Logical, default .F.
Kdip	Threshold to compute optimal dilation for FFTs if AutoDilation is true	Real, default 0.0001
ONE_body		
Density_x	Output of diagonal of one-body density in space.	Logical, default .F. .
Density_k	Output of diagonal of one-body density in momentum space.	Logical, default .F. .
Pnot	Computation of nonescape probability P_{not}?	Logical, default .F.
xstart	Where does the integration on the density for P_{not} start?	Real, no default.
xend	Where does the integration on the density for P_{not} stop?	Real, no default.
ystart	Where does the integration on the density for P_{not} start?	Real, no default.
yend	Where does the integration on the density for P_{not} stop?	Real, no default.
zstart	Where does the integration on the density for P_{not} start?	Real, no default.

zend	Where does the integration on the density for P_{not} stop?	Real, no default.
Phase	Computation of the phase?	Logical, default .F. .
Gradient	Computation of the phase gradient?	Logical, default .F. .
Cavity_Order	Computation of cavity order parameter	Logical, default .F. .
TWO_body		
Correlations_X	Computation of spatial correlation functions on the full grid?	Logical, default .F. .
Correlations_K	Computation of momentum correlation functions on the full grid?	Logical, default .F. .
StructureFactor	Output of dynamic structure factor and local correlation functions	Logical, default .F. .
xref	x value of reference point of dynamic structure factor	Real, default 0.d0
yref	y value of reference point of dynamic structure factor	Real, default 0.d0
zref	z value of reference point of dynamic structure factor	Real, default 0.d0
Correlation_Coefficient	Output of correlation coefficient $\tau = \frac{\langle \vec{r}_1 \vec{r}_2 \rangle - \langle \vec{r} \rangle^2}{\langle \vec{r}^2 \rangle - \langle \vec{r} \rangle^2}$	Logical, default .F. .
Geminals	Toggle output of natural geminal occupations in GO_PR.out	Logical, default .F. .
FullGeminals	Toggle output of natural geminals in `<time>NzMy-x-geminals.dat` files	Logical, default .F. .
corr1restr	Computation of spatial first order correlation functions on a restricted grid?	Logical, default .F. .
xini1	Restricted grid start	Real, no default.
xfin1	Restricted grid stop	Real, no default.
xpts1	Number of grid points for restricted grid.	Integer, no default.
corr1restrmom	Computation of spatial correlation functions on a restricted momentum grid?	Logical, default .F. .
kxini1	Restricted grid start	Real, no default.
kxfin1	Restricted grid stop	Real, no default.
kpts1	Number of grid points for restricted grid.	Integer, no default.
corr2restr	Computation of spatial second order correlation functions on a restricted grid?	Logical, default .F. .
xini2	Restricted grid start	Real, no default.
xfin2	Restricted grid stop	Real, no default.

xpts2	Number of grid points for restricted grid.	Integer, no default.
corr2restrmom	Computation of momentum second order correlation functions on a restricted grid?	Logical, default .F. .
kxini2	Restricted grid start	Real, no default.
kxfin2	Restricted grid stop	Real, no default.
kpts2	Number of grid points for restricted grid.	Integer, no default.
MANY_body		
lossops	Loss operators, i.e., projectors on $N = 2$ Hilbert space.	Logical, default .F. .
border	Border partitioning $N = 2$ Hilbert space for evaluation of the loss operators.	Real, no default.
Entropy	Computation of diverse entropy measures	Logical, default .F. .
NBody_C_Entropy	Computation of entropy of matrix elements of full N-body density matrix	Logical, default .F. .
TwoBody_Entropy	Computation of entropy of 2-body density matrix	Logical, default .F. .
SingleShot_Analysis	Toggle the output of random deviates of the N-body density matrix.	Logical, default .F.
SingleShot_FTAnalysis	Toggle the output of random deviates of the N-body momentum density matrix.	Logical, default .F.
NShots	Number of single shots (random deviates of the N-body density) to compute.	Integer, default 10
CentreOfMass	Toggle sampling of centre-of-mass operator	Logical, default .F.
CentreOfMomentum	Toggle sampling of centre-of-momentum operator	Logical, default .F.
ShotVariance	Toggle computation of integrated variance of single shots	Logical, default .F.
NSamples	Number of samples to make from the centre-of-mass operator	Integer, default 10000
anyordercorrelations_X	toggle output of higher order correlations $\rho^{(p)}(\vec{r}_{ref}, ..., \vec{r}_{ref}, \vec{r}_{order-1}, \vec{r}_{order-2})$	Logical, default .F.
anyordercorrelations_X	toggle output of higher order correlations $\rho^{(p)}(\vec{r}_{ref}, ..., \vec{r}_{ref}, \vec{r}_{order-1}, \vec{r}_{order-2})$	Logical, default .F.

order	specify order up to which correlation functions are to be output	Integer, default 10
oneD	toggle output of correlations with one free variable $\vec{r}_{order}$	Logical, default .T.
twoD	toggle output of correlations with two free variables $\vec{r}_{order-1}$ and $\vec{r}_{order}$	Logical, default .F.
c_ref_x	x value of reference point for higher order correlations	Real, default 0.d0
c_ref_y	y value of reference point for higher order correlations	Real, default 0.d0
c_ref_z	z value of reference point for higher order correlations	Real, default 0.d0
TWO_D		
MOMSPACE2D	Output of 2D correlation functions' $g^{(1)}(\vec{r}_1'\|\vec{r}_1)$ and $g^{(2)}(\vec{r}_1', \vec{r}_1)$ in slices?	Logical, default .F. .
REALSPACE2D	Output of 2D momentum correlation functions' slices?	Logical, default .F. .
REALSKEW2D	Output of 2D skew correlation function $g^{(1)}(\vec{r}\|-\vec{r};t)$ and $g^{(2)}(\vec{r}_1, -\vec{r}_1;t)$?	Logical, default .F. .
MOMSKEW2D	Output of 2D skew momentum correlation function $g^{(1)}(\vec{k}\| - \vec{k};t)$ and $g^{(2)}(\vec{k}_1, -\vec{k}_1;t)$?	Logical, default .F. .
x1const	Keep X-coordinate of first position (momentum) $\vec{r}_1$ ($\vec{k}_1$)) constant?	Logical, default .T. .
x1slice	At which value to keep the X-coordinate of $\vec{r}_1$ ($\vec{k}_1$))?	Real, default 0.0.
y1const	Keep Y-coordinate of first position (momentum) $\vec{r}_1$ ($\vec{k}_1$)) constant?	Logical, default .T. .
y1slice	At which value to keep the Y-coordinate of $\vec{r}_1$ ($\vec{k}_1$))?	Real, default 0.0.
x2const	Keep X-coordinate of second position (momentum) $\vec{r}_1'$ ($\vec{k}_1'$)) constant?	Logical, default .F. .
x2slice	At which value to keep the X-coordinate of $\vec{r}_1'$ ($\vec{k}_1'$))?	Real, default 0.0.
y2const	Keep Y-coordinate of second position (momentum) $\vec{r}_1$ ($\vec{k}_1$)) constant?	Logical, default .F. .
y2slice	At which value to keep the Y-coordinate of $\vec{r}_1'$ ($\vec{k}_1'$))?	Real, default 0.0.
PROJ_X	Calculate effective density one-dimensional potential $V_{eff} = \int d\xi \sum_{ij} \rho_{ij}\phi_i^*(x,y,t)\phi_j(x,y,t)$, where $\xi = x$ and/or y	Logical, default .F. .

DIR	Specifying the direction for the effective one-dimensional potential.	Character, 'X' means $\xi = x$, 'Y' means $\xi = y$ and 'B' means both $\xi = x$ and $\xi = y$ are computed.
L_z	Computation of angular momentum eigenvalue and matrix elements.	Logical, default .F. .

Table 8: MCTDH-X analysis parameters

This documentation of the variables in the `analysis.inp` is also available in the html code documentation and the in-line documentation of the example file. After an appropriate modification, the program can be run (in an interactive shell) by typing

```
MCTDHX
```

or by using/adapting/submitting one of the example PBS scripts in the `PBS_Scripts` directory and submitting the computation into the queue of a job scheduling system.

The analysis can be run after a computation has finished and the modification of the `analysis.inp` file with the following command:

```
MCTDHX_analysis
```

After the analysis program terminated successfully, ASCII files (structured as specified in the file `documentation/Analysis_output_documentation`) are in the working directory of the program. These files can be visualized using e.g. `gnuplot` or any other visualization software for data.

4.3 Available Visualization Scripts

The directory `Visualization_Scripts` contains several bash scripts to process the ASCII output of an `MCTDHX_analysis` run into `.mpg` or `.avi` movies. For a list of the currently available scripts and their function, see Table 9.

The way in which this is achieved, is by first processing the output files of an an `MCTDHX_analysis` run with `mctdhx_gnuplot` and produce a time-series of images. Subsequently, this time-series of images is encoded as a movie file using `mctdhx_mencoder`. `mctdhx_gnuplot` and `mctdhx_mencoder` are installed on your platform from the source tarball in the subdirectory `External_Software` by the MCTDH-X installation script. The automated way of using the visualization scripts is through the visualization master script. The Visualisation Master Script (`vms.sh`) is a short bash script that can be used to conveniently facilitate the creation of videos from a computation's data. To use vms.sh, just run it with the command

```
vms.sh <Movie Type> <Computation directory> <Lower plot range>
       <Upper plot range> <#Gridpoints> <2D-Slice 1> <2D-Slice 2>
```

where `Movie type` is one of the scripts in the `Visualisation_Scripts` directory. If `Movie type=''all''` is specified, all available movie scripts will be run for the particular computation. The computation directory must contain the input file `MCTDHX.inp` of the computation. The other arguments are optional and need not to be given. They can be used to fine-tune the output of the

Visualization Script	Function
`1D-CORR1-K`	Movie of the coherence $\|g^{(1)}\|^2$
`1D-CORR2-RESTR-X`	Movie of the two-body correlations $\|g^{(2)}\|$ on a restricted grid.
`1D-DENSITY_X-TIME-EVOLUTION-PM3D`	"Movie" of the density $\rho(x,t)$, i.e., 2D plot where the y-axis is time.
`1D-CORR1-RESTR-K`	Movie of the coherence $\|g^{(1)}\|^2$ in momentum space on a restricted grid.
`1D-CORR2-X`	Movie of the two-body correlations $\|g^{(2)}$.
`1D-CORR1-RESTR-X`	Movie of the coherence $\|g^{(1)}\|^2$ on a restricted grid.
`1D-DENSITY_K`	Movie of the momentum density $\rho(k,t)$.
`1D-CORR1-X`	Movie of the coherence $\|g^{(1)}\|^2$.
`1D-DENSITY_X`	Movie of the density $\rho(x,t)$.
`2D-DENSITY_M2`	Movie of the density and the first two orbitals in two-dimensional computations.
`2D-DENSITY_M3`	Movie of the density and the first three orbitals in two-dimensional computations.
`2D-DENSITY_M4`	Movie of the density and the first four orbitals in two-dimensional computations.
`2D-ORB_AVG_PHASE_DENSITY_Lz`	Movie of Orbital average phase, density and angular momentum in two-dimensional computations.
`2D-AVG-PHASE`	Movie of the average phase in two-dimensional computations.
`2D-DENSITY_K`	Movie of the momentum density in two-dimensional computations.
`2D-AVG-PHASE-PHASEGRADIENT-DENSITY-ENERGY-LZ`	Movie of the average phase, phase gradient, density and angular momentum in two-dimensional computations.
`2D-DENSITY_X`	Movie of the density in two-dimensional computations.
`CI`	Movie of the CI coefficients.
`nat_occ_loop`	Plotting the natural occupations of a computation.

Table 9: List of available visualization scripts.

All the above scripts take 5 command line arguments: `start time, stop time, time increment, number of orbitals, number of particles`. To generate a movie of the coherence on a restricted grid in momentum space one could do `1D-CORR1-RESTR-K 0 100 0.1 4 101` – this command would generate a movie for the first 100 time units in steps of 0.1 for an $M = 4$ computation with $N = 101$ particles. The visualization master script, `vms.sh`, automates the usage of the above movie scripts.

analysis program, i.e., the plotting range, the gridpoints and the slices for the output of the correlation functions of two-dimensional computations. If you need to find out the valid strings to use, then run the command `./vms.sh` and list of the available strings will be output to the screen. If the command `./vms.sh <Movie Type> <Computation directory> ...` is entered successfully, it will generate all the ASCII data from the binary MCTDH-X output and build the movie(s). All the visualization scripts in the `Visualization_Scripts` subdirectory have five command line arguments: `timeInitial`, the time at which the movie will begin; `timeIncrement`, the time difference between each frame; `timeFinal`, the time at which the movie will end; `MOrbs`, the total number of orbitals in the simulation and `Npar`, the number of particles in the simulation. `vms.sh` basically calls the respective desired visualization script(s) with the appropriate arguments. Eventually, it prints a short message '`And we are happy.`' once the video has been successfully created. Of course, one can also call the scripts in the `Visualization_Scripts` directory manually by typing

```
<Visualization_Script> <timeInitial> <timeFinal> <timeIncrement> <MOrbs> <Npar>
```

where `<Visualization_Script>` can be chosen from the list in Table 9.

4.4 Configuring the Monster Script

This script basically acts as a secretary, creating subdirectories, copying files, editing text, and submitting jobs in a coordinated fashion so that after the user configures just 2 text files, a single function call can result in thousands of computers working for days to weeks to calculate up to a combined 10-dimensional paramter scan. There are two steps to configuring `MonsterScript.sh`: configuring the input file and preparing the run files. An example input file can be found at `Input_Examples/ParameterScan.inp` which contains in-line documentation (see 10). Inside the working directory, one must place a properly configured `ParameterScan.inp`, `libmctdhx.so`, a binary executable file consistent with `$binary` defined inside `ParameterScan.inp`, and an input template consistent with `$Input_Template` defined inside `ParameterScan.inp`. Once these 4 files are properly configured and placed, the script is called by running
`$MCTDHXDIR/MonsterScript.sh`. The script scans up to 5 user defined relaxation parameters, runs them until convergence is detected and, if desired, automatically scans *each* relaxation with up to 5 user-defined propagation parameters. When running on a cluster, the script will automatically restart jobs if they finish, due to time constraints, before the calculation is complete. To circumvent job number restrictions on clusters, the script will build a series of runscripts that each simultaneously run many calculations in a single job, rather than a single computation per job.

A little under-the-hood knowledge is useful to to effectively use and debug `MonsterScript.sh`. The script initially calls `$MCTDHXDIR/Computation_Scripts/ParameterScan_Propagation.sh` or `$MCTDHXDIR/Comptuation_Scripts/ParameterScan_Relaxation.sh` depending on the input file configuration, which then iterates through the corresponding parameter set. As the script loops through each parameter set, it calls either
`$MCTDHXDIR/Scripts/IterateParameters.sh` or `$MCTDHXDIR/Scripts/IterateParameters_relax.sh` which then perform the actual secretary functionality using about a dozen smaller scripts inside `$MCTDHXDIR/Scripts`. If it is determined that a new calculation must be run when using a cluster, a few lines are added to a runscript in the working directory called `run#.sh` for some number `#`. To avoid duplicate computations, a file in the computation directory is created called `RunFlag`, which is automatically deleted when the job runs out of time or the computation is complete, via some code within the runscript. The runscript accumulates calculation jobs until the total number of nodes

requested reaches a threshold defined by `$MPMDNodes` in `ParameterScan.inp`, and then it is submitted using (in most cases) a `qsub` command within either `$MCTDHXDIR/Scripts/MPMDrun_relax.sh` or `$MCTDHXDIR/Scripts/MPMDrun_prop.sh` on line 106. It is often useful to comment this line for testing purposes, as it will prevent the script from submitting any jobs, leaving it to only copy, move, and edit files. Directories with incomplete calculations are stored in files `Relax_Array` and `Prop_Array*`. Directories are removed from these files when their corresponding calculations are complete, triggering an end of the corresponding `ParameterScan_*.sh` function call when the array is empty.

When `MonsterScript.sh` is run without any arguments, all lingering runscripts, `RunFlag`'s, and arrays are cleared. If the user wishes to keep these files, they must run `MonsterScript.sh save`.

Parameter Name	Description	Values
`Do_Relax`	Specifies whether to do or skip relaxations.	"T", "F"
`Do_Propagations`	Specifies whether to do or skip propagations	"T", "F"
`Propagation_Start`	Specifies how to start propagations when relaxation is skipped	"BINR", "HAND"
`Do_Analysis`	Specifies whether to do or skip analysis	"T", "F"
`MonsterName`	Suffix for job names "MONSTER_$MonsterName"	any string
`runhost`	Specifies which cluster is used, or if no cluster is used	"hermit", "hornet", "maia", "bwgrid", "PC"
`numnodes`	Specifies number of nodes used for relaxation jobs	Integer
`MPMD`	Specifies whether to run in Multiple Program Multiple Data mode, i.e. multiple computations per submitted job. This only applies to scans on a cluster, and is recommended if more than 20 computations are desired	"T", "F"
`MPMDjobs`	If using MPMD mode, this specifies how many nodes are requested for each job	Integer
`maxjobs`	Maximum number of jobs allowed on the queue (20 for hornet and hermit)	Integer
`binary`	Name of MCTDHX executable in working directory	usually "MCTDHX_intel"
`Relaxation_Template`	Name of input template for relaxations in working directory	usually "MCTDHX.inp"

`Propagation_Template`	Name of input template for propagations in working directory (if identical to `Relaxation_Template`, the parameters adjusted for the relaxation will be copied, if not then the paramters adjusted for the relaxation will *not* be copied).	usually "MCTDHX.inp"
`Relaxtime`	Time to run relaxations for	Positive number
`NParameters`	Number of parameters to scan for relaxations	[1-5]
`Parameter#`	#=1-5, name of relaxation parameter #	Any parameter found in MCTDHX.inp
`List#`	#=1-5, specifies if a list of values is used for relaxation parameter #	"T", "F"
`Parameter#List`	#=1-5, if `$List#`="T", specifies parameter # values to scan over	Array
`Scan#_Start`	#=1-5, if if `$List#`="F", speficies beginning of range of relaxation parameter # to be scanned	number
`Scan#_Stop`	#=1-5, if if `$List#`="F", speficies end of range of relaxation parameter # to be scanned	number
`Scan#_Step`	#=1-5, if if `$List#`="F", speficies step of range of relaxation parameter # to be scanned	number
`MaxNodes`	Maximum number of nodes allocated for a single propagation computation	Positive integer
`Prop_Time_Final`	End time of propagation computations	Positive number
`Prop_NParameters`	Number of propagation parameters to scan over	[1-5]
`Prop_Parameter#`	#=1-5, name of propagation parameter #.	Any parameter in "MCTDHX.inp"
`Prop_List#`	#=1-5, specifies if a list of values is used for propagation paramter #	"T","F"
`Prop_Parameter#List`	#=1-5, if `$Prop_List#`="T", specifies propagation parameter # values to scan over	list of appropriate values
`Prop_Scan#_Start`	#=1-5, if `$Prop_List#`="F", specifies beginning of scan range for propagation parameter #	Number
`Prop_Scan#_Stop`	#=1-5, if `$Prop_List#`="F", specifies end of scan range for propagation parameter #	Number
`Prop_Scan#_Step`	#=1-5, if `$Prop_List#`="F", specifies step of scan range for propagation parameter #	Number

Table 10: Parameters inside `ParameterScan.inp`.

5 MCTDH-X main and analysis program output documentation

5.1 Main program output

MCTDH-X has several standard output files, and it will generate additional ASCII output if the toggle `Write_ASCII` is set to true in the input of a computation. Specifically, standard output is comprised by the files `NO_PR.out`, `Initialization.dat`, `Error.dat`, and `Timing.dat`. The additional ASCII output is comprised of `<time>orbs.dat` and `<time>coefs.dat` files – these can also be generated after a computation has finished by running the analysis program which processes the binary data files `PSI_bin` and `CIc_bin`. Set aside the binary `PSI_bin` and `CIc_bin` and the ASCII `Initialization.dat` files, all the above output files are column-formatted ASCII files.

5.1.1 Output structure in standard MCTDHX computations

For the case that atoms without internal structure are treated (`Multi_Level = .F.`), the column structure of the ASCII output files is described in the following tables 11,12,13,14,15.

Column 1	Column 2 to $(1+M)$	Column $2+M$
Time t	Natural orbital occupations $\rho_M^{(NO)}(t)$ to $\rho_1^{(NO)}(t)$	Energy $E(t)$

Table 11: `NO_PR.out` file structure.
This table explains the column structure of the `NO_PR.out` file. M stands for the number of orbitals in the computation.

5.1.2 Output structure in block Davidson relaxations

In the case of block Davidson computations, several vectors of coefficients are relaxed simultaneously using the same set of orbitals. Consequently, the structure of the output changes as specified in the following tables 16, and 17.

For every vector in the block, a separate file with natural occupations is generated whose structure is identical to table 11. The naming convention for these files is `NO_PR.BL<state>` where `<state>` stands for the index of the vector in the block.

5.1.3 Output structure in multilevel MCTDHX computations

For the case that atoms with internal structure are treated (`Multi_Level = .T.`), the column structure of the ASCII output files is described in the following tables 18,19.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
N^o of integration step	Time t	$E_{orb}(t)$	$E_{orb,rel}(t)$	$E_{CI}(t)$	$E_{CI,rel}(t)$	$E_{tot}(t)$	$E_{V_t}(t)$

Table 12: Structure of the `Error.dat` file.

This table explains what is saved in the different columns of the `Error.dat` file. $E_{orb}(t)$ and $E_{orb,rel}(t)$ are the absolute and relative integration errors from the orbitals' equations of motion, respectively. $E_{CI}(t)$ and $E_{CI,rel}(t)$ are the integration errors from the coefficients' equations of motion, respectively. $E_{tot}(t) = E_{orb}(t) + E_{CI}(t)$ is the sum of the orbital and coefficients integration errors and $E_{V_t}(t)$ it an (experimental) error measure for the error due to a time-dependency of the external one-body potential.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
N^o of integration step	Time t	Execution time	T_{step}	T_{CI}	T_{ρ}	$T_{CI,Func}$	T_{Orb}

Table 13: `Timing.dat` file structure.

this table displays the column structure of the `Timing.dat` output file. Here,T_{step} is the execution time for the present integration step, $T_{CI} = T_{\rho} + T_{CI,Func}$ is the overall execution time in this step spent on the configuration interaction, i.e., the coefficients part of the program. T_{ρ} is the runtime consumed to invert the matrix elements of the reduced one-body density, $T_{CI,Func}$ is the execution time consumed in applying the Hamiltonian to the coeffcients vector. T_{Orb}, finally, is the execution time spent for evaluating the right hand side of the orbitals' equations.

Column 1 to 3	Column 4	Column 5	Column 6 & 7	Column 8 & 9	Column 10 & 11 to (10+2M) & (11+2M)
x, y, z	DVR weight	$V(x, y, z, t)$	$\rho_w(x, y, z; t)$	$\rho_{(NO)}(x, y, z; t)$	$\phi_1(x, y, z; t)$ to $\phi_M(x, y, z; t)$

Column (11+2M+1) & (11 + 2M + 2) to (11+4M+1) & (11+4M+2)
$\phi_M^{(NO)}(x, y, z; t)$ to $\phi_1^{(NO)}(x, y, z; t)$

Table 14: `<time>orbs.dat` file structure.

This table explains the column structure of the `<time>orbs.dat` output files of the main or analysis program. x, y, z are the spatial coordinates,$V(x, y, z, t)$ is the one-body potential, $\rho_w(x, y, z; t)$ is the density in working orbitals, $\rho_{(NO)}(x, y, z; t)$ is the density in natural orbitals, $\phi_1(x, y, z; t)$ to $\phi_M(x, y, z; t)$ are the working orbitals, and $\phi_M^{(NO)}(x, y, z; t)$ to $\phi_1^{(NO)}(x, y, z; t)$ are the natural orbitals. Please note, that some of the quantities are complex numbers which then are output decomposed in their real and imaginary parts in two columns (as specified by the column numbers).

Column 1	Column 2 & Column 3
N^o of Coefficient	Real and imaginary part of the coefficient

Table 15: Structure of the `<time>coef.dat` files

Column 1	Column 2 to Column 2 $\cdot$ blocksize + 1
N^o of Coefficient	Real and imaginary part of the coefficient for each vector in the block

Table 16: Structure of the `<time>coef.dat` files for block Davidson computations

Column 1 to 3	Column 4	Column 5	Column 6 & 7	Column 8 & 9 to $(8+2M)$ & $(9+2M)$	$\cdots$
x, y, z	DVR weight	$V(x,y,z,t)$	$\rho_{(NO)}(x,y,z;t)$	$\phi_M^{(NO)}(x,y,z;t)$ to $\phi_1^{(NO)}(x,y,z;t)$	Repeat columns $6 \cdots 9+2M$ for all vectors in the block

Table 17: `<time>orbs.dat` file structure for block Davidson computations.
This table explains the column structure of the `<time>orbs.dat` output files of the main or analysis program in the case of Block-Davidson relaxations. x, y, z are the spatial coordinates,$V(x,y,z,t)$ is the one-body potential, $\rho_{(NO)}(x,y,z;t)$ is the density in natural orbitals, and $\phi_M^{(NO)}(x,y,z;t)$ to $\phi_1^{(NO)}(x,y,z;t)$ are the natural orbitals. Please note, that some of the quantities are complex numbers which then are output decomposed in their real and imaginary parts in two columns (as specified by the column numbers). Importantly, the dots $\cdots$ imply that the columns 6 to $9+2M$ are repeated for all the wavefunctions in the block Davidson computation.

Column 1	Column 2 to $(1+M)$	Column $(2+M)$	Column $(3+M)$ to $3+M+N_l$
Time t	Natural orbital occupations $\rho_M^{(NO)}(t)$ to $\rho_1^{(NO)}(t)$	Energy $E(t)$	State populations (density) for all levels

Table 18: `NO_PR.out` file structure for computations with `Multi_Level=.T.`.
This table explains the column structure of the `NO_PR.out` file. M stands for the number of orbitals in the computation and N_l is the number of internal states considered.

Column 1 to 3	Column 4	Column 5 to $4+N_l+N_{CI}$	Column $5+N_l+N_{CI}$ & $6+N_l+N_{CI}$ to $3+3N_l+N_{CI}$ & $4+3N_l+N_{CI}$	Column $5+3N_l+N_{CI}$ & $6+3N_l+N_{CI}$ to $3+5N_l+N_{CI}$ & $4+5N_l+N_{CI}$
x, y, z	DVR weight	$V^i(x,y,z,t)$ and $V^i_{CI}(x,y,z,t)$	$\rho^i_w(x,y,z;t)$	$\rho^i_{(NO)}(x,y,z;t)$

Column $5+5N_l+N_{CI}$ & $6+5N_l+N_{CI}$ to $3+5N_l+N_{CI}+2MN_l$ & $4+5N_l+N_{CI}+2MN_l$	Column $5+5N_l+N_{CI}+2MN_l$ & $6+5N_l+N_{CI}+2MN_l$ to $3+5N_l+N_{CI}+4MN_l$ & $4+5N_l+N_{CI}+4MN_l$
$\phi^i_k(x,y,z;t)$ for $i=1,..,N_l$ and $k=1,..,M$	$\phi^{(NO),i}_k(x,y,z;t)$ for $i=1,..,N_l$ and $k=M,..,1$

Table 19: `<time>orbs.dat` file structure for multilevel computations.
This table explains the column structure of the `<time>orbs.dat` output files of the main or analysis program for the case that `Multi_Level=.T.` was set. N_l is the number of internal states and N_{CI} is the number of conical intersections ($N_{CI}=0$ if `Conical_Intersection=.F.`). Please note, that the index of the internal state is always running first before the orbitals' index. x, y, z are the spatial coordinates,$V^i(x,y,z,t)$ is the one-body potential of internal state i, $V^i_{CI}(x,y,z,t)$ is the coupling of the $i-$th conical interaction, $\rho^i_w(x,y,z;t)$ is the density in working orbitals for internal stat i, $\rho^i_{(NO)}(x,y,z;t)$ is the density in natural orbitals for state i, $\phi^i_1(x,y,z;t)$ to $\phi^M_1(x,y,z;t)$ are the working orbitals in internal state i, and $\phi^{(NO),i}_M(x,y,z;t)$ to $\phi^{(NO),i}_1(x,y,z;t)$ are the natural orbitals in state i. Please note, that some of the quantities are complex numbers which then are output decomposed in their real and imaginary parts in two columns (as specified by the column numbers).

The structure of the output files described in tables 12,13,15, is identical for multilevel computations.

5.2 Analysis program output

The output of the MCTDH-X analysis software is toggled with the input file `analysis.inp` as described in the table 8. Generally there are two kinds of different output file structures. Some analysis quantities are scalar and their time series will be saved in a single file, like e.g., the nonescape probability and some other quantities need one file per point in time, like e.g., the density or momentum density. `Orbitals_Output` and `Coefficients_Output` toggle the output of

the files `<time>orbs.dat` and `<time>coefs.dat` – their structure is the same as when one sets the variable `Write_ASCII` in the main programs input to `.T.`, see tables 14 and 15.

The structure of the output files generated by setting the respective analysis variables is collected in Appendix **??** in tables 22,23,24,25,26,27,28,29,30,31,32,33,34,35,36, and 37.

6 Developer Guidelines

The MCTDH-X program package uses Mercurial as a version management and provides a documentation in `.pdf` format as well as a `.html` code documentation generated by Doxygen. Generally, if you are planning to implement something useful it is recommended to browse the code documentation in `documentation/html/index.html` to find out in which module to start. The package developers will also be happy to help you with this.

In writing code, please stick to the following principles:

1. develop code in different branches of the repository (see section 7 below for how to create/manage branches)
2. write code that is readable (use indentation and obvious variable and subroutine names)
3. write code that is modular (group written subroutines or variable declarations into modules)
4. use CamelCase naming conventions, where appropriate. Every new word in declarations (of subroutines, variables or modules) should start with a capital letter. For subroutines and modules their *action(s)* are prefixed with an underscore, like for instance `Get_KineticEnergyAction_AllOrbitals`.
5. update this manual with new functionality and user guidance
6. add code documentation that doxygen can process, i.e., start commented lines with `!>` or `!<`. This is especially crucial for new subroutines, such that other users are able to understand how your routine works and may use or further develop it.
7. share your version by commiting it to your working copy and letting the developers know, such that your development branch can be merged and tested to enter the next release of the package (see paragraph on the version management below).
8. use the test-script `testMCTDHX.sh` to test your developments against reference values.

7 Version Management

The tool that is used to manage the development of the MCTDH-X software is Mercurial, in terminal, `hg`. It is available on most Unix-based systems and facilitates the contribution of multiple developers to a project. Mercurial is a distributed version management system, i.e., each user of it has the the full version history available locally. The basic operations are collected in the following list:

1. making a full copy of a repository with `hg clone <repository location>`
2. creating branches with `hg branch <branch name>`.

3. showing a summary of the changes with respect to the last revision by `hg status`
4. adding and removing files by `hg add <files>` and `hg remove <files>`
5. add all new files and remove all missing files with `hg addremove`
6. showing the history of commit messages for all revisions by `hg log`
7. add a version number to your revision with `hg tag <version label>`. The structure of the version label is explained below.
8. commiting changes made to a revision of the repository as a new revision with `hg commit -u <username>`
9. *pushing* or *pulling* the changes made in the present repository to another repository by `hg push <repository location>` or `hg pull <repository location>`.
10. updating the repository after pulling/pushing a changeset by `hg update`

It is important to note here that the version management of the central repository is not open for all users. If you are interested in contributing, write an email to the developers at mctdhx@ultracold.org to obtain an account for the software repository.

Version labels: The version labels for MCTDH-X follow the following syntax:

```
<major>.<minor><small><fix>
```

Here, `<major>` is an integer that is incremented by 1 when a new major feature is commited (for instance the capability of the program to treat mixtures of indistinguishable particles). The `<minor>` label is an integer, that is incremented by 1 when a new minor feature like, for instance, a new operator or a new analysis routine is ready. The `small>` label is an integer which is incremented by 1, whenever changes to the repository are made which do not correspond to a feature like, for instance, an improvement of the efficiency or an update to the manual.

Contribution how-to:

The strategy taken for the development of MCTDH-X is that every contributor should create her/his own branch(es) for the software development. Branches may be created with a name describing the respective contribution or just the contributor's name. When a new version is due to be released, all the different branches will be merged into the default branch to form a release.

In the following, the basic steps to create branches and to push your current revision into the main repository are described. These include to first *clean up* your current version, second make sure if the changes you want to share as a contribution to the main repository are there and which files are affected, third commit the changes as a new revision to your local repository and write a thorough description of the implemented changes and fourth, push the changeset to the main repository.

Create branch: To create a branch, run `hg branch <name>`. Typically `<name>` should be your user name in the forum or descriptive for the features that are planned to be developed in this branch.

Clean up: To remove the compiled included software, run `make -f <Makefile.your-configuration> purge` and then, to remove the objects and libraries generated in the compilation of the main program, run `make -f <Makefile.your-configuration> clean` .

Inspect the changeset: Check if indeed only the files that you intended to modify show up when you run `hg status`. If unsure, look through the individual files and in case of doubt, recompile/reinstall the program and test it again (for instance by running `testMCTDHX.sh`) and start over with the first step.

Commit revision: Once you verified and tested your changes, commit them to your local repository by `hg commit -u <username>` . After issueing this command you will be prompted to write a so-called commit message. The text should include a brief but concise description of the changes entering the revision and a list of open tasks or known problems. The changelog of the program is the timeline of these commit messages and they are therefore crucial to let the other users and programmers know about changes.

Share the changes: After your local revision is prepared, you can share the changeset by contributing it to the main repository by running `hg push`, which will prompt you for username and password. If you get a message that notifies you of the creation of a *remote head*, please contact the developers at mctdhx@ultracold.org since it's likely that there is a conflict between your changeset and the latest revision in the repository. In that case, it is necessary to merge the heads and resolve the conflicts.

Feedback and suggestions as well as bug-reports are welcome anytime at mctdhx@ultracold.org or http://ultracold.org/forum.

A Predefined one-body potentials

whichpot	Description	Potential	Parameters
HO1D	1D harmonic oscillator	$V(x) = \frac{1}{2}p_1^2x^2$	$p_1 \equiv$trap frequency.
HO2D	2D harmonic oscillator	$V(x,y) = \frac{1}{2}(p_1^2x^2 + p_2^2y^2)$	$p_{1/2} \equiv$trap frequency in x/y
HO3D	3D harmonic oscillator	$V(x,y,z) = \frac{1}{2}(p_1^2x^2 + p_2^2y^2 + p_3^2x^2)$	$p_{1/2/3} \equiv$trap frequency in $x/y/z$
h+d	1D harmonic oscillator plus Gaussian central barrier	$V(x) = \frac{1}{2}(x-p_1)^2 + p_2 exp(-\frac{(x-p_1)^2}{2p_3^2})$	$p_1 \equiv$ diplacement, $p_2 \equiv$ height of Gaussian, $p_3 \equiv$ width of Gaussian
h2D+d	2D harmonic oscillator plus Gaussian central barrier	$V(x) = \frac{1}{2}(p_1^2x^2 + p_2^2y^2) + p_3 exp(-(\frac{(x)^2}{2p_4^2} + \frac{(y)^2}{2p_5^2}))$	$p_{1/2} \equiv$ trap frequency in x-/y-direction, $p_3 \equiv$ height of Gaussian, $p_{4/5} \equiv$ width of Gaussian in direction x/y
HO<X>D+td_gauss	<X>D harmonic oscillator plus time-dependent [height $A(t)$] Gaussian central barrier in x-direction	$V(x) = \frac{1}{2}p_1^2((x-p_2)^2 + y^2) + A(t) exp(-\frac{(x-p_5)^2}{2p_6^2})$	$p_1 \equiv$ trap frequency in x- and y-direction, $p_2 \equiv$ displacement of the minimum of the trap w.r.t. the barrier, $p_5 \equiv$ displacement of Gaussian w.r.t. $x = 0$, $p_6 \equiv$ width of Gaussian barrier in direction x. Height $A(t) = \begin{cases} \frac{tp_3}{p_4} & t \leq p_4 \\ p_3 & t > p_4 \end{cases}$
Tilt	Tilted triple well potential	$V(x) = -p_1x + p_2 \sin(2x)^4 + (x/2.2)^{20}$	$p_1 \equiv$ tilt parameter, $p_2 \equiv$ depth of the wells.
Tiltinit	single well potential to initialize system for a propagation in the tilted triple well potential Tilt.	$V(x) = V(x) = -p_1x + p_2 \sin(2x)^4 + [(x - \frac{p_3\pi}{2})/0.7]^{20}$	$p_1 \equiv$ tilt parameter, $p_2 \equiv$ depth of the wells, p_3 is used to select displacement from the origin.
OL_HW_1D	lattice in one dimension with hard wall boundaries	$V(x) = \begin{cases} p_1 \sin(p_2x)^2 & p_3 < x < p_4 \\ 1000 & \text{else} \end{cases}$	$p_1 \equiv$ depth of lattice, $p_2 \equiv$ frequency of lattice, p_3, p_4 are the boundaries for the hard walls.

`TDHIM`	Harmonic potential with time-dependent frequency for benchmarks with the time-dependent harmonic interaction Hamiltonian.	$V(x) = \frac{1}{2}(1 + \sin(t)\cos(2t)\sin(\frac{1}{2}t)\sin(0.4t))x^2$	No parameters, use with `which_interaction ='TDHIM'`
`tun`	Potential for tunneling to open space dynamics	$V(x) = \begin{cases} \frac{1}{2}x^2 & x \leq 2 \\ 2.2662969 & \\ exp(-2(x-2.25)^2) & x > 2 \end{cases}$	no parameters, initial wavefunction should be localized at $x = 0$, use `whichpot='ini'` for relaxation of initial state.
`thr`	Tunneling to open space with a threshold	$V(x) = \begin{cases} \frac{1}{2}x^2 & x \leq 2 \\ Ax^3 + Bx^2 & \\ +Cx + D & 2 < x \leq 4 \\ p_1 & x > 4 \end{cases}$	Parameter p_1 defines the threshold and the polynomial coefficients $A = 1 - \frac{1}{4}p_1$, $B = 2.25p_1 - 9.5$, $C = 28 - 6p_1$, $D = 5T - 24$.
`DQD`	Double quantum dot potential	$V(x) = -p_1 exp(-p_2(x + \frac{1}{2}p_3)^2) - p_4 exp(-p_5(x - \frac{1}{2}p_3))$	$p_1 \equiv$ depth of first quantum dot, $p_2 \equiv$ width of first quantum dot, $p_3 \equiv$ distance of the two dots, $p_4 \equiv$ depth of second quantum dot, $p_5 \equiv$ width of second quantum dot.
`DQDLASER`	Double quantum dot potential illuminated by a laser with time-dependent amplitude $V_{\text{las}}(x,t)$.	$V(x) = -p_1 exp(-p_2(x + \frac{1}{2}p_3)^2) - p_4 exp(-p_5(x - \frac{1}{2}p_3)) + V_{\text{las}}(x,t) exp(-p_9(x - p_{10})^2)$	$p_{1-5} \equiv$ same as in `DQD`, $V_{\text{las}}(x,t) = p_7 \cos(p_8 t)\sin(\pi\frac{t}{p_6})x$ for $t \leq p_6$ and $V_{\text{las}}(x,t) = 0$ else.
`zero`	No potential	$V = 0$	No parameters. Boundary conditions determined by the discrete variable representation.
`MQT_ini`	Rectangular 1D box	$V(x) = \infty \quad \forall[x \notin (0, 20 - p_1)]$ and $V(x) = 0 \quad \forall[x \in)0, 20 - p_1(]$	$p_1 \equiv$ barrier width in propagation `MQT_prop`
`MQT_prop`	Rectangular 1D box, barrier and open space	$V(x) = \infty \quad \forall[x < 0]$ and $V(x) = 0 \quad \forall[x \in)0, 20 - p_1(]$ and $V(x) = 0 \quad \forall[x > 20]$ and $V(x) = 0.05\forall[x \in (20 - p_1, 20)]$	$p_1 \equiv$ barrier width

`qpl`	1D lattice with 2 frequencies/amplitudes	$V(x) = p_1 \cos(p_2 x) + p_3 \cos(p_4 x)$	$p_{1/3} \equiv$ amplitudes of the lattices, $p_{2/4} \equiv$ frequencies of the lattices.
`rot2D`	2D harmonic oscillator with rotating anisotropy	$V(x,t) = \frac{1}{2}((1 + p_a(t))(x\cos(p_1 t) + y\sin(p_1 t))^2 + (1 - p_a(t))(y\cos(p_1 t) - x\sin(p_1 t)))$	$p_1 \equiv$ the rotation frequency, $p_a \equiv$ the time-dependent anisotropy. $p_2 \equiv$ anisotropy maximum, $p_3 \equiv$ ramp-up and ramp-down time of the anisotropy, $p_4 \equiv$ plateau time at which $p_a = p_2$.
`stir2D`	2D harmonic trap with rotating stirring rod	$V(x,t) = \frac{1}{2}(x^2 + y^2) + p_3 exp[-\frac{1}{p_4}(x - p_2\cos(p_1 t))^2 + (y - p_2 \sin(p_1 t))^2]$	$p_1 \equiv$ stirring frequency, $p_2 \equiv$ stirring radius, $p_3 \equiv$ height of Gaussian rod, $p_4 \equiv$ width of Gaussian rod.
`ellipse2D`	Elliptic two-dimensional hard-walled potential well	$V(x) = \begin{cases} 1000 & \sqrt{(\frac{x}{p_1})^2 + (\frac{y}{p_2})^2} > p_3 \\ 0 & \text{else} \end{cases}$	$p_{1/2/3} \equiv$ parameters defining the ellipse.

Table 20: Predefined potentials and parameters.

B Predefined interaction potentials

`Which_Interaction` and `Interaction_Type`	Description	Potential
'gauss' and $1, 2, 3, 4$	Gaussian interparticle interaction of width `Interaction_Width`$= \sigma$	$W(\vec{r},\vec{r}') = \lambda_0 \frac{1}{(\sqrt{2\pi\sigma^2})^D} exp\left(-\frac{\lvert\vec{r}-\vec{r}\rvert^2}{2\sigma^2}\right)$
0	Contact interaction with constant strength λ_0	$W(\vec{r},\vec{r}') = \lambda_0 \delta(\vec{r}-\vec{r}')$
'cos' and 6	Contact interaction with time-dependent strength $\lambda(t)$	$W(\vec{r},\vec{r}') = \lambda_0 \cos(I_1 t)\delta(\vec{r}-\vec{r}')$
'sin' and 6	Contact interaction with time-dependent strength $\lambda(t)$	$W(\vec{r},\vec{r}') = \lambda_0 \sin(I_1 t)\delta(\vec{r}-\vec{r}')$
'TDHIM' and 5	Time-dependent version of the harmonic interaction model	$W(\vec{r},\vec{r}') = \lambda_0(1+0.4\sin^2(t))(\vec{r}-\vec{r}')^2$
'TDgauss1' and 5	Gaussian interaction with width `Interaction_Width`$= \sigma$ and time-dependent amplitude	$W(\vec{r},\vec{r}') = (\lambda_0 + I_1 \sin(I_2 t)) \frac{1}{(\sqrt{2\pi\sigma^2})^D} exp\left(-\frac{\lvert\vec{r}-\vec{r}\rvert^2}{2\sigma^2}\right)$
'TDgauss2' and 5	Gaussian interaction with width `Interaction_Width`$= \sigma$ at $t = 0$ that is time-dependently modulated	$W(\vec{r},\vec{r}') = \lambda_0 \frac{1}{(\sqrt{2\pi(\sigma^2+I_1\sin(I_2 t))})^D}$ $exp\left(-\frac{\lvert\vec{r}-\vec{r}\rvert^2}{2(\sigma^2} + I_1 \sin(I_2 t))\right)$
'lennart_j' and 4	I_2– Screened Lennart-Jones potential	$W(\vec{r},\vec{r}') = I_1\left(\frac{\sigma}{\lvert\vec{r}-\vec{r}\rvert^12} - \frac{\sigma}{\lvert\vec{r}-\vec{r}\rvert^6}\right)$ for $\lvert\vec{r}-\vec{r}\rvert > I_2$ and $W(\vec{r},\vec{r}') = I_1\left(\frac{\sigma}{I_2^12} - \frac{\sigma}{I_2^6}\right)$ for $\lvert\vec{r}-\vec{r}\rvert \leq I_2$
'HIM' and 4	Harmonic interaction model	$W(\vec{r},\vec{r}') = \lambda_0(\vec{r}-\vec{r}')^2$

Table 21: Predefined time-independent and time-dependent interaction potentials. I_X stands for `Interaction_ParameterX` from the input, σ for `Interaction_Width`, and D for the dimensionality of the problem.

C Structure of the output of the analysis program

Column 1 to 3	Column 4 to $3 + N_l$
x, y, z or k_x, k_y, k_z	$\rho^i(x, y, z; t)$ or $\rho^i(k_x, k_y, k_z; t)$

Table 22: Structure of the `<time>N<N>M<M>x-density.dat` and `<time>N<N>M<M>k-density.dat` files.
These files are generated if `Density_X/Density_K` it true. x, y, z and k_x, k_y, k_z are the spatial and momentum grid, respectively, N_l is the number of internal states, and $\rho^i(x, y, z; t)$ and $\rho^i(k_x, k_y, k_z; t)$ are the spatial and momentum densities, respectively. In the case of a computation treating atoms with internal structure, the index i runs through all internal states of the considered atoms and one density is output for every state. In the names of the files `<time>` is the time t `<N>` is the particle number, `<M>` is the orbital number.

Column 1	Column 2
Time t	Nonescape probability $P_{not}(t, x_s, x_e)$

Table 23: The nonescape probability output file `Nonescape`.
If the input variable `Pnot` and the borders x_s and x_e were defined with `xstart` and `xend` in the input of the analysis program, this file is created.

Column 1	Column 2	Column 3	Column 4
Time t	$S_{\rho-r}(t) = -\int d\vec{r}\rho(\vec{r};t)ln[\rho(\vec{r};t)]$	$S_{\rho-k}(t) = -\int d\vec{k}\rho(\vec{k};t)ln[\rho(\vec{k};t)\ ]$	$S_C(t) = \sum_{\vec{n}} -\lvert C_{\vec{n}}(t)\rvert^2 ln[\lvert C_{\vec{n}}(t)\rvert^2]$

Column 5	Column 6	Column 7
$S_n(t) = \sum_i -\frac{n_i(t)}{N} ln[\frac{n_i(t)}{N}]$	$I = \frac{1}{\sum_{\vec{n}} \lvert C_{\vec{n}}(t)\rvert^4}$	$S_C^N(t) = \sum_{\vec{n},\vec{n}'} -\lvert C_{\vec{n}}(t)\rvert^2 ln[\lvert C_{\vec{n}'}(t)\rvert^2]$

Table 24: Structure of the `Entropy.dat` file.
This file is generated when `Entropy` is set to true. The last column is only present if `NBody_C_Entropy=.T.` is set in `analysis.inp`.

Column 1	Column 2	Column 3	Column 4
Time t	$S_{\rho^{(2)}-r}(t) = -\int d\vec{r}_1 d\vec{r}_2 \rho^{(2)}(\vec{r}_1,\vec{r}_2;t)\, ln[\rho^{(2)}(\vec{r}_1,\vec{r}_2;t)]$	$S_{\rho-k}(t) = -\int d\vec{k}_1 d\vec{k}_2 \rho^{(2)}(\vec{k}_1,\vec{k}_2;t)\, ln[\rho^{(2)}(\vec{k}_1,\vec{k}_2;t)]$	$S_{\rho_k^{(GO)}}(t) = \sum_i -\frac{\rho_i^{(GO)}(t)}{N} ln[\frac{\rho_i^{(GO)}(t)}{N}]$

Table 25: Structure of the `TwoBody_Entropy.dat` file
. This file is generated when `TwoBody_Entropy` is true.

Column 1 to 3	Column 4 to 6	Column $7+5(i-1)$	Column $8+5(i-1)$ & $9+5(i-1)$	Column $10+5(i-1)$
x,y,z or k_x,k_y,k_z	x',y',z' or k'_x,k'_y,k'_z	$\rho_i(x,y,z;t)$ or $\rho_i(k_x,k_y,k_z;t)$	$\rho_i^{(1)}(x,y,z\|x',y',z';t)$ or $\rho_i^{(1)}(k_x,k_y,k_z\|k'_x,k'_y,k'_z;t)$	$\rho_i(x',y',z';t)$ or $\rho_i(k'_x,k'_y,k'_z;t)$

Column $11+5(i-1)$
$\rho_i^{(2)}(x,y,z\|x',y',z';t)$ or $\rho_i^{(2)}(k_x,k_y,k_z\|k'_x,k'_y,k'_z;t)$

Table 26: Structure of the `<time>N<N>M<M>x-correlations.dat` and `<time>N<N>M<M>k-correlations.dat` files for multilevel computations.
(For the explanation of the filenames, see table 22). These files are created, if the input variable `Correlations_X` and `Correlations_K`, respectively, are set to be true. The files contain all necessary quantities to compute the one-body as well as the diagonal of the two-body normalized (Glauber-) correlation function $g_i^{(1)}$ and $g_i^{(2)}$, respectively, for all internal states i. For instance, $|g_1^{(1)}|^2 = \left|\frac{\rho_1^{(1)}(x_1,x_1';t)}{\sqrt{\rho_1(x_1;t)\rho_1(x_1';t)}}\right|^2$ can be plotted as the value of `( (Column 8)^2 + (Column 9)^2 )` divided by `(Column 7)` $\times$ `(Column 10)`.

Column 1 to 3	Column 4 to 6	Column 7	Column 8 & 9	Column 10
x,y,z or k_x,k_y,k_z	x',y',z' or k'_x,k'_y,k'_z	$\rho(x,y,z;t)$ or $\rho(k_x,k_y,k_z;t)$	$\rho^{(1)}(x,y,z\|x',y',z';t)$ or $\rho^{(1)}(k_x,k_y,k_z\|k'_x,k'_y,k'_z;t)$	$\rho(x',y',z';t)$ or $\rho(k'_x,k'_y,k'_z;t)$

Column 11
$\rho^{(2)}(x,y,z\|x',y',z';t)$ or $\rho^{(2)}(k_x,k_y,k_z\|k'_x,k'_y,k'_z;t)$

Table 27: Structure of the `<time>N<N>M<M>x-correlations.dat` and `<time>N<N>M<M>k-correlations.dat` files.
(For the explanation of the filenames, see table 22). These files are created, if the input variable `Correlations_X` and `Correlations_K`, respectively, are set to be true. The files contain all necessary quantities to compute the one-body as well as the diagonal of the two-body normalized (Glauber-) correlation function $g^{(1)}$ and $g^{(2)}$, respectively. For instance, $|g^{(1)}|^2 = \left|\frac{\rho^{(1)}(x_1,x_1';t)}{\sqrt{\rho(x_1;t)\rho(x_1';t)}}\right|^2$ can be plotted as the value of `( (Column 8)^2 + (Column 9)^2 )` divided by `(Column 7)`$\times$`(Column 10)`.

Column 1 & 2	Column 3 & 4	Column 5	Column 6
x, x' or k, k'	$\rho^{(1/2)}(x\|x';t)$ or $\rho^{(1/2)}(k\|k';t)$	$\rho(x;t)$ or $\rho(k;t)$	$\rho(x';t)$ or $\rho(k';t)$

Table 28: Structure of the `<time>N<N>M<M><x/k>corr<1/2>restr.dat` files.
The generation of these files is triggered by the analysis input variables `corr1restr`,`corr2restr`,`corr1restrmom`,`corr2restrmom`. Similar to the above table 27, the normalized correlation functions $g^{(1)}$ and $g^{(2)}$ can be computed from the contents of these files, but for one-dimensional computations and on a restricted grid which is specified through the analysis input variables `<x/k>ini<1/2>`,`<x/k>fin<1/2>`,`<x/k>pts<1/2>`, respectively.

Column 1 to 3	Column 4	Column 5	Column 6& 7
x, y, z or k_x, k_y, k_z	$\rho(x_{ref}, y_{ref}, z_{ref}, x, y, z; t)$ or $\rho(k_x, k_y, k_z; t)$ or $\rho(k_{x,ref}, k_{y,ref}, k_{z,ref}; t)$	$\rho^{order}(\{k_{x,ref}, k_{y,ref}, k_{z,ref}\}, x, y, z; t)$ or $\rho^{order}(\{k_{x,ref}, k_{y,ref}, k_{z,ref}\}, k_x, k_y, k_z; t)$	

Table 29: Structure of the `<time>N<N>M<M>x/k-order-<order>-correlations1D.dat`
(For the explanation of the filenames, see table 22). These files are created, if the input variable `anyordercorrelations_X` and `anyordercorrelations_K` as well as `oneD`, respectively, are set to be true. The files contain all necessary quantities to compute the correlation functions up to order `<order>`. Here, $x_{ref}/y_{ref}/z_{ref}/k_{x,ref}/k_{y,ref}/k_{z,ref}$ correspond to the reference point specified in the input file by `c_ref_x`/`c_ref_y`/`c_ref_z`, respectively.

Column 1	Column 2 & 3
time t	Real & imaginary part of $\tau = \frac{\langle x_1 x_2\rangle - \langle x\rangle^2}{\langle x^2\rangle - \langle x\rangle^2}$

Table 30: Structure of the `CorrelationCoefficient.dat` file. This file is created when `Correlation_Coefficient` it true.

Column 1 to 3	Column 4 and 5	Column 6 and 7	Column 8 to 10	Column 11 & 12
x, y, z	Real and imaginary part of $\rho^{(2)}(\vec{r}_1 = \vec{r}'_1 = \vec{R}, \vec{r}_2 = \vec{r}'_2 = \vec{r})$	Real and imaginary part of $\rho^{(1)}(\vec{r}_1 = \vec{R}, \vec{r}'_1 = \vec{r})$	k_x, k_y, k_z	Real and imaginary part of dynamic structure factor $G = 1 + N\mathcal{F}(\rho^{(2)}(\vec{r}_1 = \vec{r}'_1 = \vec{R}, \vec{r}_2 = \vec{r}'_2 = \vec{r} - 1)$

Table 31: Structure of the `<time>N<N>M<M>x-StructureFactor.dat` files. These files are output if `StructureFactor` is true.

Column 1	Column 2	Column 3	Column 4
Time t	$P_0^2(t)$	$P_1^1(t)$	$P_2^0(t)$

Table 32: Structure of the `lossops_N2_<border>.dat` files.
The generation of such a file is triggered by the `lossops` input variable being set to `.T.`. `<border>` is controlled by the `border` input variable. For each point in time t, a line in this file contains the probability $P_0^2(t)$ to find 2 particles to the left of `border`, the probability $P_1^1(t)$ to find one particle to the left and one to the right of `border`, and the probability $P_2^0(t)$ to find two particles to the right of `border`.

Column 1 to 4	Column 5	Column 6	Column 7 & 8	Column 9
$r_{1x}, r_{1y}, r_{2x}, r_{2y}$	$\rho(\vec{r}_1;t)$	$\rho(\vec{r}_2;t)$	$\rho^{(1)}(\vec{r}_1\|\vec{r}_2;t)$	$\rho^{(2)}(\vec{r}_1\|\vec{r}_2;t)$

Table 33: The structure of the `<time>N<N>M<M><x/k><Slice 1>-<Slice 2>-correlations.dat` files
. These are output by the analysis program if the analysis input variable `MOMSPACE2D` or `REALSPACE2D` is set to `.T.`. `<Slice 1/2>` specify which cut through the real- or momentum-space density are in the file.

Column 1 and 2	Column 3	Column 4 and 5	Column 6	Column 7
r_{1x}, r_{1y}	$\rho(\vec{r}_1;t)$	$\rho^{(1)}(\vec{r}_1\|-\vec{r}_1;t)$	$\rho(-\vec{r}_1;t)$	$\rho^{(2)}(\vec{r}_1,-\vec{r}_1;t)$

Table 34: The structure of the `<time>N<N>M<M><x/k>-SkewCorrelations.dat` files
. These files are output of the analysis program if the analysis input variable `MOMSKEW2D` or `REALSKEW2D` is set to `.T.` .

Column 1, 2, 3	Column 4 to $3+N_{shots}$
x, y, z	N_{shots} samples of the N-body density

Table 35: The structure of the `<time>N<N>M<M><x/k>SingleShots.dat` files
. These files are output of the analysis program if the analysis input variable `SingleShot_Analysis` or `SingleShot_FTAnalysis` is set to `.T.` .

Column 1, 2, 3	Column 4
x, y, z	Histogram of $N_{samples}$ samples of the centre-of-mass-operator.

Table 36: The structure of the `<time>N<N>M<M><x/k>CentreOfMass.dat` files
. These files are output of the analysis program if the analysis input variable `CentreOfMass` or `CentreOfMomentum` is set to `.T.` .

Column 1 to 3	Column 4	Column 5 to $(5+M)$	Column $(6+M)$ & $(7+M)$	Column $(8+M)$ & $(9+M)$ to $(8+3M)$ & $(9+3M)$
x, y, z	$\xi_{avg}(x,y,z;t)$	$\xi_1(x,y,z;t)$ to $\xi_M(x,y,z;t)$	$\nabla_x\xi_{avg}(x,y,z;t)$ & $\nabla_y\xi_{avg}(x,y,z;t)$	$\nabla_x\xi_1(x,y,z;t)$ & $\nabla_y\xi_1(x,y,z;t)$ to $\nabla_x\xi_M(x,y,z;t)$ & $\nabla_y\xi_M(x,y,z;t)$

Table 37: The structure of the `<time>N<N>M<M>phase.dat` files.
The generation of the files is toggled by setting the analysis input variable `PHASE` to `.T.`. If additionally `GRADIENT` it set to `.T.` then, Columns $(8+M)$ to $(9+3M)$ containing the phase gradients will be generated. Here x, y, z are the coordinates, $\xi_{avg}(x,y,z;t)$ is the average phase, $\xi_1(x,y,z;t)$ to $\xi_M(x,y,z;t)$ are the M orbital phases, $\nabla_x\xi_{avg}(x,y,z;t)$ & $\nabla_y\xi_{avg}(x,y,z;t)$ is the x and y component of the average phase's gradient, and $\nabla_x\xi_1(x,y,z;t)$ & $\nabla_y\xi_1(x,y,z;t)$ to $\nabla_x\xi_M(x,y,z;t)$ & $\nabla_y\xi_M(x,y,z;t)$ are the x and y components of the gradients of the M orbital phases.