The QUANTICS Package: A set of programs for multi-dimensional quantum dynamics.

User's Guide

Version 1

Release 1

Authors:

G. A. Worth, M. H. Beck, A. Jäckle, H.–D. Meyer, F. Otto, M. Brill, and O. Vendrell

Address:

The School of Chemistry, University of Birmingham,
Edgbaston, B15 2TT, UK
Email: g.a.worth@bham.ac.uk

Address:

Theoretische Chemie, Physikalisch–Chemisches Institut,
Im Neuenheimer Feld 229, D–69120 Heidelberg, Germany
Email: Hans-Dieter.Meyer@pci.uni-heidelberg.de

July 9, 2017

Contents

List of Tables

List of Figures

List of Examples

Copyright

How to cite the software and important references are given in the on-line documentation included with the package.

IntroductionChapter 1

Introduction

The MCTDH method is an efficient algorithm for the solution of the time-dependent Schrödinger equation. For a full description of the theory see the review [1]. You may also wish to read the MCTDH book [?]. The MCTDH program has been developed to perform quantum mechanical wavepacket propagations employing this method. All the options and variants of the MCTDH method presented in the review are implemented. Furthermore, the MCTDH program can be used to propagate wavefunctions numerically exactly and to diagonalise a Hamiltonian by the Lanczos algorithm. A variety of programs included in the MCTDH package serve to analyse the results of a calculation and compute observable quantities, which can directly be plotted with the help of GNUPLOT scripts.

The installation of the MCTDH package is described in Appendix E.

This documentation is intended to help the user by explaining, with many examples, how to set up and run a calculation and analyse the results. For a calculation the Hamiltonian operator and the input parameters must be defined. This is done in two ASCII files, named operator and input file, which must have .op and .inp, respectively, as extension. The required data is put in as keywords. In both files, the keywords are grouped together into sections, each with a specific set of information. The sections start with a line containing the keyword *XXX*-SECTION, and end with END-*XXX*-SECTION, where *XXX* is the name of the section. Everything following a # is treated as comment.

How to set up the operator and input file will be detailed in the following chapters. Note, however, that this Guide does not claim to be complete. Although the majority of options of the MCTDH package — and in particular those being most important for your daily work — is described, there are probably still options useful for you that are not documented here. For the full list of options, see therefore the HTML manual. The HTML manual also describes the installation process.

Some parts of the User's guide are labelled as advanced topics, indicated by a "*" in the table of contents. These parts contain information on features of the MCTDH package that make the programs more convenient to use but do not extend their functionality. The advanced topics also deal with options of the MCTDH package which are needed in special cases only. You may skip these parts until you got more experienced with the MCTDH package.

Please keep in mind the following typographical conventions which are designed to help you reading the User's Guide:

`Typewriter` The typewriter font is used for literal characters, such as keywords and labels given in the input files, the names of routines and variables, and extracts of the source code.

Italics The italics font indicates arguments which are supposed to be substituted by the user.

Bold face Bold face emphasises the names of programs and scripts in the MCTDH Package, and their options.

Sans serif The sans serif font is employed for files, directories, and paths.

UPPERCASE The different sections that arrange the input and operator files are given in uppercase.

SMALL CAPS Small capital letters are used for the names of persons as well as programs that are not part of the MCTDH package.

Chapter 2

A QUANTICS tutorial

When you have successfully installed the QUANTICS package (see Appendix E), you have various programs in the field of multi-dimensional quantum dynamics at your disposal. Before we go into the details of how to use these programs, we would like to invite you to a short tour of the MCTDH package, by performing some exemplary calculations. On this trip you will get an overview of the opportunities the MCTDH package offers. The tour shall also demonstrate the ease of employing the program and give you an impression of the efficiency of the code.

First set up and move to a suitable directory in which to run the tutorial calculations (e. g. $MCTDH_DIR/tutorial or $HOME/tutorial), then follow the instructions below. The tutorial uses standard problems. Once a calculation has been made, try to understand the input files, they can be used as templates for other calculations. The expression $MCTDH_DIR occurring in the following examples stands for the path of the MCTDH-directory.

2.1 Determining the absorption spectrum for the photodissociation of NOCl

The photodissociation of NOCl is a simple photo-chemical reaction. After excitation from the ground to the first excited state, $S_0 \rightarrow S_1$, the chlorine atom dissociates on a femto-second time-scale. This results in a broad band for the absorption spectrum. This system was used for the first application of MCTDH to a realistic system [3].

The calculation consists of two stages. First, the ground state wavefunction is generated by energy relaxation of an initial guess wavefunction on the ground state surface, S_0. The second stage then places this wavepacket on the excited state surface, S_1, leading to photodissociation.

1. Copy the files $MCTDH_DIR/inputs/nocl0.inp and $MCTDH_DIR/inputs/nocl1.inp to your tutorial directory, and create there the directories nocl0 and nocl1.

2. To perform the ground state relaxation calculation, type

   ```
   quantics nocl0
   ```

   You will have to wait about 2 seconds. (The timings given in this manual are for a 3 GHz PC running under Linux).

3. To perform the photo-dissociation calculation, type

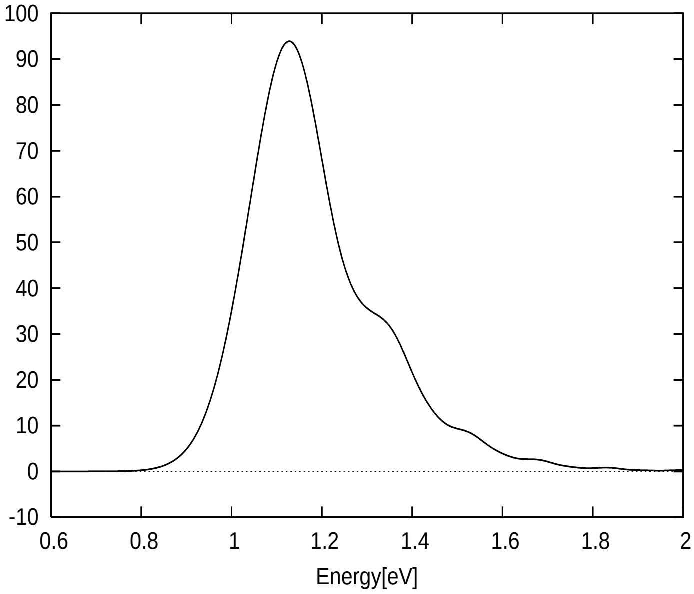

Figure 2.1: The absorption spectrum for the NOCl molecule on excitation to the S_1 state.

```
quantics nocl1
```

This will again take about 2 seconds.

NB There is now the option `-mnd` (make name directory) which allows you to skip the creation of the name directory. E. g.

```
quantics -mnd nocl1
```

will make the name directory before starting the calculation.

The calculation can now be analysed. Move to the directory nocl1 which contains all the data files from the propagation.

1. To watch the system dissociating, type

   ```
   showd1d -a -M -y 5 -sm f1
   ```

   In order to understand the options and parameters, type **showd1d -h** and see the HTML documentation. Try the other format options (-S, -T) and inspect the motion of the other degrees of freedom (f2, f3). The program showd1d also supports interactive plotting. Start the program with

   ```
   showd1d -inter
   ```

   and follow the menu options to select and alter the plot.

2. To plot the spectrum, type

   ```
   autospec -g 1 0.6  2.0 ev 0.0 1
   gnuplot -persist spectrum.pl
   ```

The first line produces a GNUPLOT file with data to plot the spectrum. This is done from 0.6 eV to 2.0 eV and using a simple cosine cutoff function to allow for the finite propagation time. The result is shown in Fig. 2.1. In order to understand the options and parameters, type **autospec -h** and see the HTML documentation. Note that the spectrum shown is the Fourier-transform of the autocorrelation function times the energy. Hence it is assumed that the ground state energy is at zero, such that energy equals excitation energy. If this is not the case, use option `-e` to shift the energy scale. The `-FT` option suppresses the multiplication with the energy, showing directly the Fourier-transform. (NB The option `-FT` is now default. Use option `-EP` to switch on the energy prefactor, or use `-Mb` <dipole-moment>, to plot the properly normalized absorption spectrum in mega barns.)

To make life easier, there exist a number of bash scripts (so called pl-scripts) which automatically call an analyse routine and plot the results. The above commands are equivalent to

```
plspec 0.6  2.0 ev
```

One may alternatively call **plspec** without arguments. The script will then prompt you for the missing input. Finally, the command **plauto** plots the autocorrelation function, the command **plnat** plots the natural populations, **plqdq** plots the expectation values of the coordinates, and the commands **plupdate**, **plupdate -e** , and **plspeed** show information on the performance of the integration. **plall** prints a list of all pl-scripts, but for more information see the HTML documentation. Note that all pl-scripts support the `-h` option. We do recomment the use of the pl-scripts!

The program **showsys** is a powerful tool for plotting 1D and in particular 2D views on wavepackets and potentials. To plot the potential one first has to generate a so called pes file. To do so, move up to the tutorial directory, where nocl1.inp is located, and type

```
quantics -pes nocl1
```

This will generate the files pes, log.pes, and op.log.pes in the nocl1 name-directory. The pes file is an operator file in which all terms containing derivative operators or CAPs are deleted. The WARNING message which appears can be ignored. It just tells you, that the mctdh program will not perform a propagation, although there is a keyword `propagation` in the input file. Now move back to the name-directory and type

```
showsys
```

A menu appears (see Example 11.6), which allows various options to be set. Go to menu point 10 (type 10), and change the plot task to *2 = plot pes* (type 2). Next input a 1 three times and a 2D cut through the surface (with theta fixed to 1.545 radians) will pop up. Now use menu point *20 = change coordinate section*, i.e. chose another cut. If one gives *x* and two numbers, a 1D plot will appear. After you have played around enough, go back to menu point 20 and input

```
 x  y  2.1
```

Then use menu point 5. You will be asked for a file name. Chose any convenient name, e. g. xyz. The plot data is then written to the file xyz for later use.

Next we want to inspect the wavefunction. Go to menu point 10 and chose *5 = plot reduced density*. The density, i.e $|\Psi|^2$ integrated over all coordinates, except those specified by *x* and *y* (that is integrated over all angles in the present case), will be shown. Input a 1

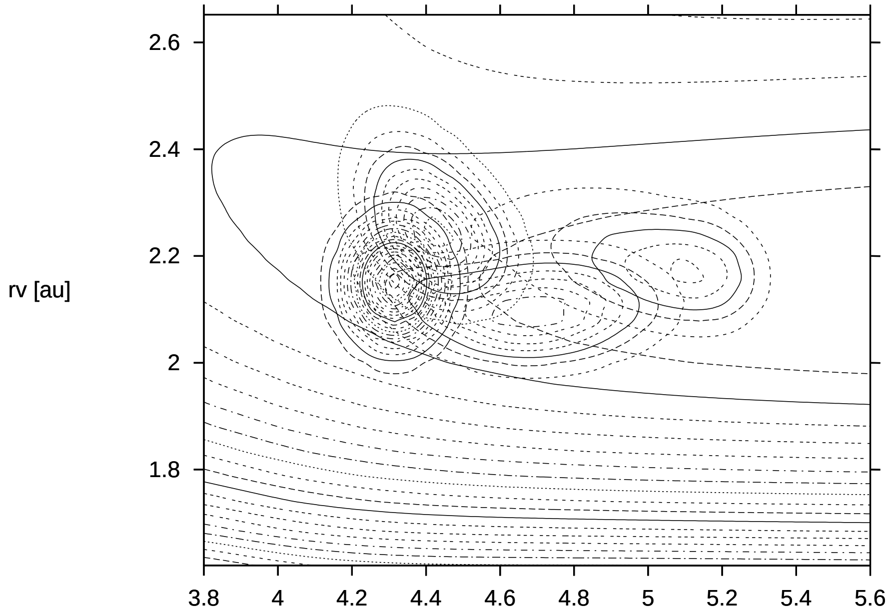

Figure 2.2: Overlay plot, wavepacket on potential. The Wavepacket density is shown for the times t = 0, 10, 20, 30 fs. The density is obtained by integrating $|\Psi|^2$ over all angles, whereas the potential contour lines are obtained by fixing the angle to 2.1 rad.

three times and you will see the initial density. Pressing RETURN will display the density propagated by one time step, and so on. After you have returned to the menu, chose point *400 = Overlay plots* and then *410 = File for overlay* and enter the file name (xyz). After inputting 1's you will now see an overlay plot, i. e. the wavepacket on top of the contour lines of the potential.

With menu points 240 and 245 one may switch off the legend (or keys) and the title. Menu point 285 allows to take larger time steps and with point 280 one may switch to different plot forms, e. g. to plot all time slices at once. Such a plot is shown in Fig. 2.2.

Inspect the ASCII files of the name directories, in particular output, log, and timing. The file input contains a copy of the input file, the options, and the operator file. Thus, it tells you exactly what you have been doing. Since an NOCl run is so fast, NOCl is ideally suited for testing. Just play around with it! You may e. g. change the numbers of single particle functions or alter the integrator accuracies. You also may try the options, e. g. to start a continuation run type:

```
quantics -c -tfinal 50 nocl1
```

Type **quantics -h** to obtain the list of options.

2.2 Determining state populations for the photo-excitation of pyrazine

The pyrazine molecule contains a classic example of vibronic coupling. Two states, which are close in energy, are coupled by motion along one vibrational mode, resulting in a broad spectrum for the upper state. This system can be described using the simple vibronic-coupling model Hamiltonian.

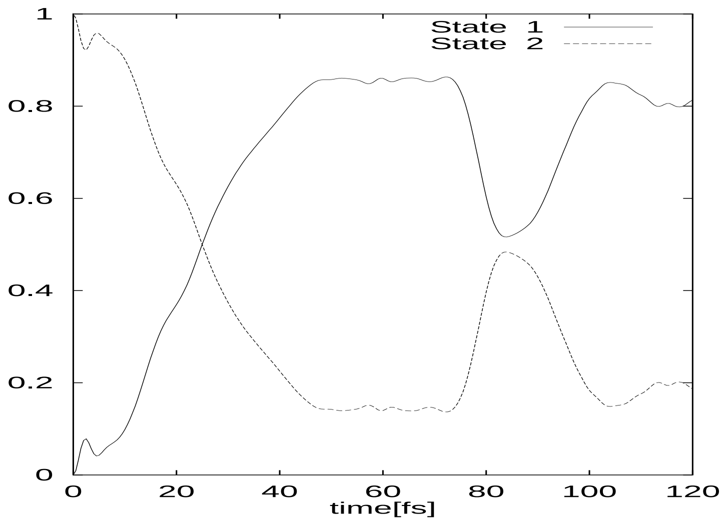

Figure 2.3: The diabatic state populations of the pyrazine molecule after excitation to the S_2 state, calculated using a 4-mode model.

The vibronic-coupling model Hamiltonian is well suited to the MCTDH method, being already in the product form required for maximum efficiency. For further details of this system, see Refs. [5–8], and the references therein.

In this tutorial, we use a simple 4-mode 2-state model. This qualitatively reproduces the experimental spectrum after the addition of phenomenological broadening. The calculation takes the ground state wavefunction (here a simple product of gaussians as the ground state surface is harmonic), and places it on the S_2 excited surface. Propagation then takes place, and rapid population transfer to the S_1 state is observed. Finally, the spectrum of the model system is calculated.

1. Copy the file $MCTDH_DIR/inputs/pyr4.inp, and create the directory pyr4.
2. To perform the photo-excitation calculation, type

   ```
   quantics pyr4
   ```

   This will take about 20 seconds.

The calculation can now be analysed. Move to the directory pyr4 which contains all the data files from the propagation.

1. To plot the diabatic state populations, type

   ```
   rdcheck -g 1 0
   gnuplot -persist chk.pl
   ```

   or, more simply, type

   ```
   plstate
   ```

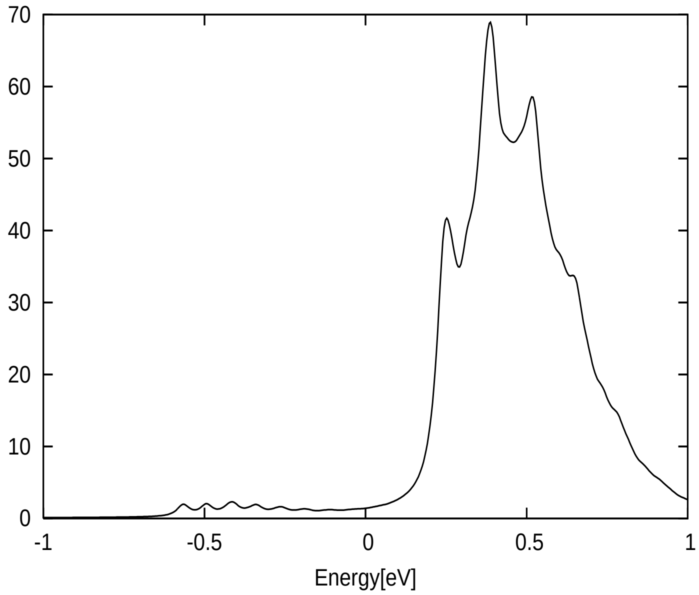

Figure 2.4: The absorption spectrum for the pyrazine molecule on excitation to the S_2 state, calculated using a 4-mode model with phenomenological broadening.

The result is shown in Fig. 2.3. Note that very fast transfer occurs to the S_1 state. At around 80 fs the system returns to the conical intersection connecting the two states, and a second transference of population occurs.

2. To plot the spectrum, type

```
autospec -g 1 -e -0.2258 eV  -1.0 1.0 eV 30 1
gnuplot -persist spectrum.pl
```

The first line produces a GNUPLOT file with data to plot the spectrum from -1.0 eV to 1.0 eV. An energy shift of 0.2258 eV has been added due to the zero point energy of the system. A phenomenological broadening with a relaxation time of 30 fs has also been added. The result is shown in Fig. 2.4. Again, the same figure is generated more simply by typing

```
plspec  -e -0.2258 eV  -1.0 1.0 eV 30 1
```

2.3 Determining reaction probabilities for the exchange reaction of H+H_2

The H+H_2 system is the smallest reactive molecular system, but it is the prototype of all three atom reactions. As interaction potential we will use the LSTH potential energy surface. This is a full 3D surface and as such must be first transformed to MCTDH product form. The Potfit program can accomplish this fast and reliably (at least as long as the full primitive product grid is not too large). After the wavepacket is propagated the reaction probability is determined by flux analysis. See the MCTDH review [1] or the original publication [9] for more details. Here we will perform a scattering calculation for vanishing total angular momentum $(J = 0)$ only. Thus the result is a initial-state selected reaction probability and not a cross section.

1. Copy the files $MCTDH_DIR/pinputs/lsth.inp and $MCTDH_DIR/inputs/hh2.inp to your tutorial directory, and create the directories lsthfit and hh2.

2. To perform the potential fit calculation, type

   ```
   potfit lsth
   ```

   This will take about 5 seconds.

3. To perform the scattering calculation, type

   ```
   quantics hh2
   ```

   This will take less than 5 minutes.

4. To perform the flux analysis, move to the directory hh2 and type

   ```
   flux -e lsth 0.4 2.0 ev rv
   ```

   This will take less than 10 seconds. The option `-e lsth` sets the zero point of the energy to the minimum of the H_2 potential curve. The other arguments set the energy interval to 0.4 – 2.0 eV and select the rv-CAP for analysis.

The results of the calculation can now be inspected. Type

```
plflux
```

and you will see the reactive flux, i. e. the quantum flux going into the rv–CAP, and the energy distribution of the initial wavepacket. The reaction probability is just the quotient of these two data sets. It can be seen by typing

```
plflux -r
```

The results are shown in Fig. 2.5. One may compare them with those of reference [10].

Inspect the ASCII files of both name directories, lsthfit and hh2. The Potfit program will be described in more detail later in this guide. The motion of the wavepacket can again be visualised with the aid of **showd1d**. In particular the θ degree of freedom is interesting. Type

```
showd1d -a -y 10 f3
```

and repeatedly press RETURN to step through the pictures. Initially the molecule is in the $j = 0$ rotational state and the density is evenly distributed over all angles. After about 20 fs the wavepacket reaches the saddle-point region and the system is in the transition state. The transition state is collinear and consequently the angular distribution is now strongly peaked at zero degrees. At later times a more evenly angular distribution is again assumed. You may also inspect the motion of the other two degrees of freedom.

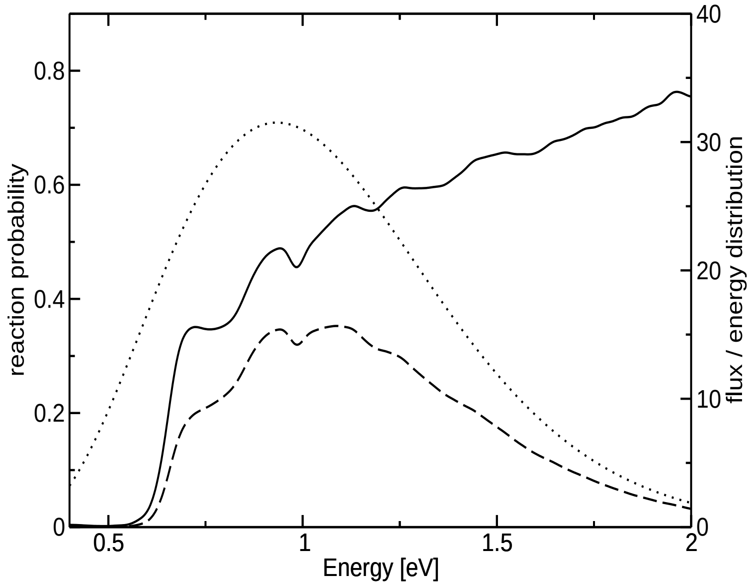

Figure 2.5: This picture shows the reaction probability of the system $H+H_2(\nu=0,j=0)$ for total angular momentum $J=0$ (solid line), which is the quotient of the quantum flux going into the rv-CAP (dashed line) and the energy distribution of the initial wavepacket (dotted line).

2.4 Determining the vibrational spectrum of LiCN

The MCTDH program is not only capable of propagating wavepackets but also of diagonalising a Hermitian Hamiltonian operator, by employing the Lanczos algorithm. The time-independent Schrödinger equation is then solved rather than the time-dependent one. This feature, and similarly the possibility of performing a numerically exact propagation, has been implemented into the mctdh package because then the very convenient operator generation is available for these tasks. Lanczos diagonalisation and exact propagation are, of course, possible only for comparatively small problems.

As a small example of this feature let us determine the vibrational spectrum of a two-dimensional model of the LiCN electronic ground state, with the CN bond length frozen at its equilibrium value. The initial wavefunction is chosen arbitrarily; the intensities thus have no physical meaning. To keep the CPU time short only a small number of Lanczos iterations will be made. The number of iterations is sufficient to converge the lowest 0.5 eV of the spectrum.

1. The LiCN surface is not linked by default. It must be first linked to the program by re-compiling MCTDH.

   ```
   compile -i licn mctdh
   ```

   It might be that you need to copy `licnsrf.f` from the addsurf directory to source/surfaces. Alternatively you may set a link (run the script **mklinks**). Type `quantics -ver` to inspect, which surfaces are included. See the HTML documentation *Installation and Compilation / Compiling the Programs* and *Hamiltonian Documentation / Available Surfaces* for more details.

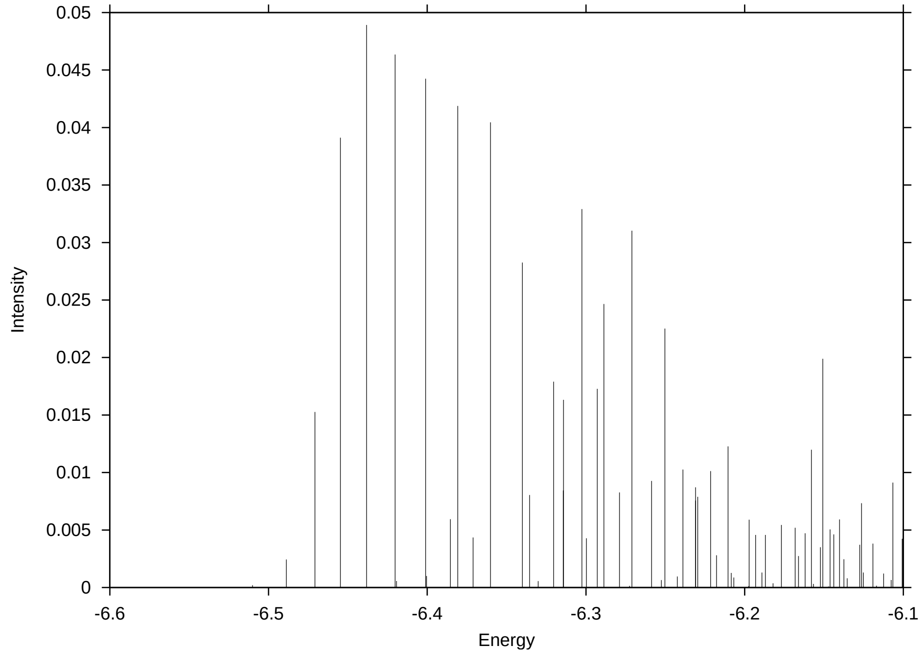

Figure 2.6: The vibrational spectrum of LiCN.

2. Copy the file $MCTDH_DIR/inputs/licn.inp, and create the directory licn.

3. To diagonalise the Hamiltonian, type

```
quantics licn
```

This will take less than 20 seconds.

The calculation can now be analysed. Move to the directory licn which contains all the data files from the diagonalisation. The eigenvalues, intensities and error estimates for the energies are stored in the ASCII file eigval.

1. To see the results, type e.g.

```
less eigval
```

The first line describes the entries of the eigval file.

2. To plot the spectrum, type

```
pleigval -a -6.6 -x -6.1
```

This displays the spectrum in the converged energy range. The result is shown in Fig. 2.6. Note that energies with very small intensities are not visible. To display all lines, add the option `-l` in order to use a logarithmic scale for the intensities.

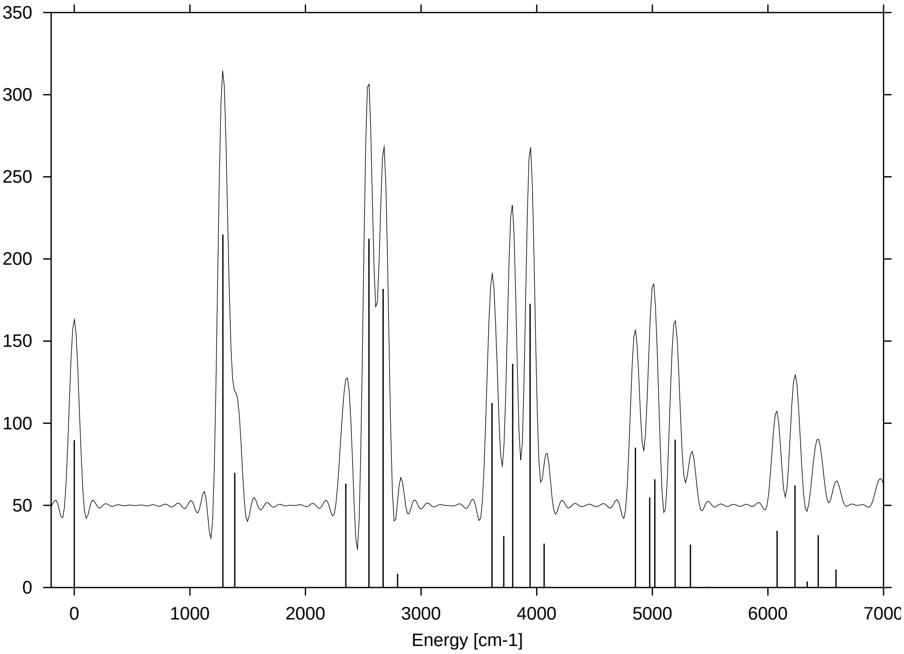

Figure 2.7: The vibrational spectrum of CO_2 as obtained by Fourier transform of the autocorrelation function and by FD using the same autocorrelation function. For better visibility, the Fourier spectrum is shifted upwards by 50 units.

2.5 Determining the vibrational spectrum of CO_2 by filter-diagonalisation

To Fourier transform the autocorrelation function is the straightforward procedure to extract eigen-energies from a time evolved wavepacket. This, however, requires a very long propagation time T as the resolution improves only like $\hbar/T$. This limit, set by the uncertainty relation, can be overcome when employing the filter-diagonalisation (FD) method introduced by Neuhauser. Our particular version of the FD method is discussed in Refs. [11, 12].

The following example shall show how filter-diagonalisation and MCTDH-propagation can be combined. The example is similar to the problem studied in Ref. [12], however, here we sacrifice some accuracy in order to gain speed.

1. Copy the file $MCTDH_DIR/inputs/co2t.inp, and create the directory co2.
2. Copy the file $MCTDH_DIR/finputs/co2ft.inp to the directory co2.
3. To perform the MCTDH propagation, type

   ```
   quantics co2t
   ```

   This will take less than 2 minutes.

As done in the previous examples, you should study the log, output, timing, etc files and investigate natural– and grid–populations. In particular it is useful to investigate the spectrum. Thus type

```
plspec -e -2534.52981 cm-1 -200 7000 cm-1
```

Table 2.1: Vibrational energies ($J = 0$) of CO_2. The MCTDH/FD energies , E_{FD}, are compared with experimental ones, E_{exp}. ΔE and ΔI denote internal error estimates of the eigen-energies and intensities, respectively. Missing entries specify states that have not been detected. In this case the intensity is taken from a larger calculation and is shown in brackets. The missed states are all of very low intensity, except for state 26. Here the computed state represents the two neighbouring states, 25 and 26. A calculation with a longer propagation time or with several wavepackets will detect more states. (See Ref. [12]). All energies are given in cm^{-1} with respect to the ground state energy.

No.	E_{exp}	E_{FD}	ΔE	Intensity	ΔI
0	0.000	-0.002	0.054	4.49d-2	3.32d-5
1	1285.414	1285.393	0.096	1.07E-1	3.05E-5
2	1388.188	1388.276	0.264	3.48E-2	7.77E-5
3	2349.148	2349.090	0.086	3.17E-2	2.26E-4
4	2548.374	2548.349	0.017	1.06E-1	3.65E-5
5	2671.113	2671.152	0.008	9.08E-2	3.00E-5
6	2797.154	2796.339	0.752	4.23E-3	2.10E-4
7	3612.845	3612.860	0.013	5.61E-2	7.65E-5
8	3714.789	3715.040	0.108	1.57E-2	9.32E-5
9	3792.679	3792.531	0.075	6.78E-2	2.16E-4
10	3942.480	3942.562	0.002	8.62E-2	3.35E-4
11	4064.101	4064.190	0.040	1.34E-2	2.31E-4
12	4225.043			(1.05E-4)	
13	4673.332			(6.17E-5)	
14	4853.622	4853.747	0.046	4.27E-2	1.62E-4
15	4977.828	4977.548	0.572	2.73E-2	3.41E-4
16	5022.273	5022.408	0.492	3.41E-2	1.61E-4
17	5099.668			(9.38E-4)	
18	5197.251	5197.442	0.055	4.47E-2	8.24E-6
19	5329.746	5329.986	0.029	1.31E-2	6.07E-5
20	5475.283	5480.947	0.839	2.19E-4	1.84E-5
21	5667.488			(1.80E-7)	
22	5915.216			(1.41E-4)	
23	6016.687			(4.45E-4)	
24	6075.984	6076.471	0.554	2.12E-2	1.50E-3
25	6227.915	6233.344	0.074	3.08E-2	8.89E-5
26	6239.852			(1.31E-2)	
27	6347.956	6333.856	2.964	2.15E-3	3.33E-4
28	6435.398	6434.692	0.261	1.59E-2	6.00E-4
29	6503.081			(3.09E-6)	
30	6588.730	6588.345	0.745	5.48E-3	2.46E-5

The option `-e -2534.52981 cm-1` shifts the zero point of the energy scale by -2534.52981 cm^{-1} which is the ground state energy. Thus, the ground state is now expected at zero. Try the options `-g 0`, `-g 1`, and `-g 2` and you will understand, why `-g 1` is the default. The plot depicts the spectral lines having a width of almost 100 cm^{-1}. This demonstrates that a precise determination of eigen-energies by Fourier transform of the autocorrelation function is difficult. (See Fig. 2.7)

To continue with the tutorial move to the co2 directory and type

```
filter co2ft
```

This runs the filter-diagonalisation and creates the files filter.eig, filter.inp, and filter.log. The file filter.inp repeats the input file, but additionally shows all default and computed parameter values. The file filter.log displays what **filter** has been doing. It also contains a list of all computed eigenvalues and intensities. The file filter.eig again contains the computed eigenvalues, but omits those which lie outside the energy window or which are detected as spurious according to an internal error measure.

Unfortunately, the file filter.eig may still contain spurious eigenvalues. These are detected by performing several filter-diagonalisations with slightly different parameters and keeping only those eigenvalues which are stable. To perform the additional filter-diagonalisation runs, type

```
filter "file_outputname=f1,window_energypoints=150"  co2ft
filter "file_outputname=f2,vp_principle = 1/H,filter_function=box" co2ft
```

The double quoted arguments of **filter** overwrite the values from the input file. We used a different number of energy points (150 rather than 125) and a different variational principle (1/H rather than H) together with a different damping function (`box` rather than `cos`) as compared to the first filter-diagonalisation. (See the HTML documentation for details). The output files are now `f1.*` and `f2.*`, respectively. A list of the stable eigenvalues together with an error estimate based on the spread of the eigenvalues is produced by the command

```
fdmatch filter.eig f1.eig f2.eig | sort -n | fdcheck 3 0 > results
```

You may inspect this data, `cat results`, and plot a stick spectrum

```
plfdspec -c -a -200  results
```

The content of the file results is compiled in Tab. 2.1 (columns 3–6) and compared with experimental data (column 2).

To obtain all eigenvalues in this energy range, one has to run the propagation with different initial states, either sequentially or (more efficient) in parallel by performing a multi-packet propagation. Also, increasing the accuracy (more SPFs, e.g. 16/16/14) and increasing the propagation time (e.g. 250 fs) will help to detect states of low intensity. See Ref. [12] for details.

2.6 Determining eigenstates by improved relaxation

Improved relaxation is a MCSCF variant where the SPFs are optimised by relaxation (propagation in negative imaginary time), but the A-vector is determined by diagonalisation of the Hamiltonian matrix evaluated in the set of the present SPFs. In contrast to filter diagonalisation, improved relaxation yields not only the eigenenergies but also the eigenstates. Improved relaxation is more accurate than filter diagonalisation, but also more elaborate, because one has to perform a separate calculation for each state.

1. Copy the file $MCTDH_DIR/inputs/co2_gs.inp to your tutorial directory and similarly the files co2_sym.inp, co2_asym.inp, and co2_excite.inp.

2. To perform the relaxation, execute the command:

   ```
   quantics -mnd co2_gs
   ```

and, after the job has finished, run the inputs co2_sym, co2_asym, and co2_excite. Move to the directory co2_gs and type **rdrlx -e** to read the rlx_info file. This produces the following output.

```
 time  order  q   beta*1000    Energy[cm-1]         ovl*1000     Delta-E
  -2.0    0   0   0.000E+00    2922.190 202 862     0.0000    0.000E+00
 0.000   15*  0   804.71296    2534.937 314 025     0.0000   -3.873E+02
 0.500   11   0   999.98713    2534.663 002 346     0.0000   -2.743E-01
 1.250   10   0   999.99715    2534.558 864 426     0.0000   -1.041E-01
 2.000    9   0   999.99971    2534.535 729 282     0.0000   -2.314E-02
 4.000    8   0   999.99990    2534.528 474 396     0.0000   -7.255E-03
 6.000    6   0   0.293E-08    2534.528 204 472     0.0000   -2.699E-04
 8.000    4   0   0.874E-10    2534.528 194 788     0.0000   -9.684E-06

WARNING: Davidson did not converge for 1 diagonalisations.
```

The star $\star$ at 15 indicates that that this Davidson diagonalisation did not converge. 16 Davidson iterations are needed for convergence, but in the input file the maximum number of iterations was limited to 15. The non-convergence is of no relevance here, because all later iterations converged. However, if non-convergence of the Davidson happens frequently, one cannot trust the results, but has to repeat the calculation with a different integrator setting.

The first data line with the negative time gives the energy expectation value of the initial wavefunction. The second line, $t = 0.0$, gives the energy obtained by diagonalising the Hamiltonian represented in the orbitals of the initial wavefunction. Then the orbitals (SPFs) are relaxed and the Hamiltonian matrix, built from the new orbitals, is diagonalised again. This procedure is repeated till convergence is reached.

For the present example the convergence of the improved relaxation scheme is fast. beta denotes the squared overlap of the current A-vector with the previous one. If beta is very close to one, the difference from 1 is printed. ovl denotes the squared overlap of the current wavefunction with the initial wavefunction. This data is evaluated only for relaxation=lock runs. More information on the performance of the improved relaxation run is obtained when dropping the option -e from **rdrlx**. (Try rdrlx -h). A graphical visualisation of the convergence is provided by **plrlx**. Try **plrlx**, **plrlx -a 3**, and **plrlx -E -l**.

Inspect the outputs of the other relaxation runs in a similar way. Note that considerably more Davidson iterations are needed for converging higher excited states. Note also that the energy scale is shifted via the keyword `rlxunit=cm-1,2534.528194` to display directly the excitation energies. Inspect the input files and try to understand every line.

The tiny 3D problem CO_2 is, of course, too simple to show the strengths of *improved relaxation*. If you wish to solve some 6D problems, run the input files hono.dav.inp and H2CS.$*$.inp

2.7 Determining eigenstates by block improved relaxation

The block variant of improved relaxation is very useful if several low-lying states are to be computed. It makes use of the single-set multi-packet feature of MCTDH, i. e. the different packages are formally put on different (single-set) electronic states. True electronic states, either in multi-set or single-set formalism, can be added as well. Because the packets are treated in single set, the SPFs are propagated (or relaxed) on a state-averaged mean field. As there is only one set of SPFs, the SPFs cannot be optimal for one eigenstate, they are optimized for the full block of eigenstates to be computed. Hence the block form will in general require more SPFs to achieve the same accuracy as a (single) improved relaxation. But because the block form generates several eigenstates at once, it is often more convenient and sometimes even numerically more efficient. However, a block-relaxation requires considerably more memory than a single relaxation.

The following example takes more computation time than the previous ones. It may be skipped if one is not particularly interested in block improved relaxation.

1. Copy the file $MCTDH_DIR/inputs/blkHONO.inp to your tutorial directory.
2. To perform the block-relaxation, execute the command:

   ```
   quantics -mnd blkHONO &
   ```

Edit the input file such that the numbers and keywords, which appear after "#" or "##", become valid. Then run the input again. The convergence of the eigenenergies is most conveniently visited by running the script **rdrlx**. The convergence can be inspected graphically by running **plbrlx**, which is very similar to **plrlx**, but requires as argument the number of the state to be plotted. The converged eigenenergies (in cm^{-1}) obtained from these runs read as follows:

```
        blk.1         sing.1         blk.2         blk.3

  0      0.006          0.000          0.000          0.000
  1     93.992         93.973         93.974         93.972
  2    600.920        600.872        600.873        600.871
  3    710.781        710.623        710.625        710.621
  4    796.056        795.999        796.000        795.997
  5    944.422 .      944.111        944.116        944.108
  6   1055.391       1055.384       1055.385       1055.384
  7   1188.605 :     1188.073       1188.079       1188.070
  8   1267.671       1267.600       1267.609       1267.598
  9   1306.671       1306.601       1306.604       1306.595
 10   1313.852 +     1312.748       1312.761       1312.736
 11   1386.094 :     1385.262       1385.263       1385.247
 12   1405.637       1405.519       1405.545       1405.510
 13   1549.283 +     1547.471       1547.458       1547.431
 14   1575.849 +     1574.831       1574.851       1574.821
 15   1640.938       1640.887       1640.887       1640.884
 16   1690.148       1690.009       1690.034       1690.006
 17   1726.574 :     1726.015       1726.050       1726.009
 18   1770.187 #     1761.572 u     1761.638       1761.581
 19   1783.998 *     1779.401       1779.466       1779.377
 20   1829.101       1829.017       1829.017       1829.013
 21   1858.538 .     1858.223       1858.242       1858.210
 22   1901.955 +     1902.884       1897.807 .     1897.580
 23   1909.695 #     1897.012 u     1902.886       1902.838
 24   1970.004 #     1966.496 *     1961.701       1961.558
 25   2002.997 :     2002.376       2002.404       2002.323
 26   2025.521       2025.382       2025.384       2025.381
 27   2049.494 :     2048.983       2049.045       2048.967
 28   2120.335 .     2120.044       2120.019       2120.002
 29   2147.592 #     2136.577 .     2136.567 .     2136.276
 30   2173.911 $     2153.985       2154.080       2153.897
 31   2211.437 :     2210.637       2210.633       2210.622
 32   2242.353 +     2240.892       2240.933       2240.825
 33   2292.107 +     2291.121       2291.196       2291.096
```

```
 34   2306.560       2306.468       2306.477       2306.460
 35   2341.711 $     2339.301 $     2322.512 :     2321.754
 36   2357.212 $     2340.799 u     2339.333       2339.225
 37   2376.415 $     2376.423 $     2339.687 .     2339.416
 38   2396.093 $     2370.751 .     2370.709 .     2370.415
 39   2402.523 $     2400.605 u     2376.419       2376.401

 . > 0.2, : > 0.5, + > 1.0, * > 2.0, # > 5.0, $ > 10., u unconverged
```

The first run, blk.1, used the SPF set 9/4/16/18 and took 46 min CPU time on a 3.2 GHz Pentium 4, and 165 MB RAM, the second run with the SPF set 10/5/30/20 took 3 h CPU time and 560 MB RAM, the third run with 12/5/42/28 SPFs took 11 h CPU time and 1340 MB RAM. The third calculation, blk.3, is fully converged and serves as reference. Deviations from these results are indicated by $. \;:\; + \;*\; \#$ and $ when they are larger than 0.2, 0.5. 1.0, 2.0 5.0, and 10 cm^{-1}, respectively.

As one notices, the eigenenergies obtained with the first run are not too accurate. In particular the last four states are quite bad, showing deviations of more than 10 cm^{-1}. The second calculation, blk.2, however, shows rather good results. All deviations are below 1 cm^{-1}, and in most cases the deviations are even below 0.1 cm^{-1}.

To compare block- with single-relaxation, separate single-relaxations were performed using the eigenstates of the block-relaxation as start vectors. The small set, 9/4/16/18, of SPFs was used and the results are shown in the column sing.1 . The results of the single relaxations are much improved as compared to the corresponding block relaxation, except for the last four states and for the excited states no 18 and 23. These states (as well as state 36) did not converge, i. e. the energy kept on oscillating. The energies displayed in this case are some mean value. In the block-relaxation the SPFs are optimized to represent all 40 states under consideration, whereas in the single-relaxation they are optimized for a particular eigenstate. Obviously more SPFs are needed in a block-relaxation to obtain results of similar quality. However, the single relaxations took between 30 s and 3 min each, depending on the state to be relaxed. In total they took 1 h CPU time, which is similar to the 3 h used by the second block-relaxation. Remembering that for the single-relaxations we took excellent starting vectors, namely the eigenstates obtained by the first block-relaxation, and that the second block-relaxation yield eigenenergies of better accuracy than the sing.1 calculations, one may conclude that single- and block-relaxation take similar amounts of CPU time for obtaining similar accuracy. But the memory consumption of the block-relaxation is considerably larger (20 MB compared to 560 MB). However, it requires much less human effort to run a block-relaxation as compared to run 40 single relaxations.

2.8 Using potfit and chnpot to fit a surface to *ab initio* data points

In this example it is shown how to use *ab initio* data points to generate a natural potential and how this natural potential can then be interpolated into a more suitable grid for a MCTDH simulation. To perform such tasks the programs **potfit** and **chnpot** will be used, respectively. A detailed description of how such tasks are accomplished is found in chapter 12.2 of this guide.

The data used in this example – taken from Ref. [13] – is a 2D cut corresponding to the PES of the CO_2^- anion in C_{2v} symmetry. The two coordinates are the length of the two CO bonds and the angle between them, i. e. symmetric stretch and bending.

2.8.1 Transforming the *ab initio* data to product form

1. Create a new directory, for example co2fit

2. From the directory $MCTDH_DIR/pinputs/ copy to the directory co2fit the files:

   - co2_potfit.inp
   - co2_pes
   - co2_r_grid
   - co2_theta_grid

   The file co2_pes contains the *ab initio* energies and the two following files the corresponding grid points.

3. Move to the co2fit directory and execute the command:

   ```
   potfit -mnd co2_potfit
   ```

A new subdirectory co2_potfit has been created, change to it and execute

```
showpot -vfit
```

In the interactive menu type 3 times `1, return` to see the plot of the natural potential fit. By definition it is identical to the original data on the grid points since the same number of natural potentials as grid points has been used (Inspect the co2_potfit.inp file).

2.8.2 Interpolating the natural potential to a new primitive grid

1. Copy the file $MCTDH_DIR/pinputs/co2_chnpot.inp to the co2fit directory.

2. Execute the command:

   ```
   chnpot -mnd co2_chnpot
   ```

A new subdirectory co2_chnpot has been created, which contains new dvr and natpot files. As before, the newly interpolated potential can be inspected using the utility **showpot** and following the interactive menu.

The process outlined in the present and previous subsections can be repeated using the alternative set of files co2_r_dense_grid, co2_dense_pes, co2_theta_dense_grid , co2_chnpot_dense.inp, co2_potfit_dense.inp. The initial grid is double as dense as the original one. After the potfit stage the data points are interpolated to the same grid as in the previous case. The rms of the difference of the two interpolations to the same grid is 9.55×10^{-6} au, i. e. 0.26 meV. This is a very good value considering that the potential spans an energy interval of 8 eV. Indeed, using **showpot** to compare the outcome of both **chnpot** runs shows that the obtained potential energy surfaces are virtually equal, which constitutes a nice example of the usefulness of the **chnpot** utility when preparing an MCTDH calculation from *ab initio* data points.

We have used this method quite succesfully to create multi-dimensional 3D-fits to *ab initio* data. The only restriction is that the data is to be supplied on a product grid. Equidistance of the grid points, however, is not required.

2.9 Optimizing an external field with Optimal Control Theory (OCT)

MCTDH can be used to perform coherent control calculations within the OCT scheme. The OCT algorithm was developed by Tannor and coworkers [14] and by Rabitz and coworkers [15]. For this purpose **mctdh** and the routine **efield** are called from the script **optcntrl**. OCT maximizes the expectation value $\langle\Psi(T)|\, O\, |\Psi(T)\rangle$ at the final time T, where O denotes some positive semidefinite hermitian operator.

The control target O can be defined in two different ways. If O is the projector onto a target quantum state, i.e., $O = |\Psi_{\rm tar}\rangle\, \langle\Psi_{\rm tar}|$, then it is sufficient to specify the target state $|\Psi_{\rm tar}\rangle$. If O is a general operator, e.g., a projector onto an electronic state $O = |S\rangle\, \langle S|$, it has to be specified in the operator file. At present only one target operator can be specified in the operator file, it is, however, possible to use it with multiple initial states.

Multi-target optimizations are possible by using the multi-packet algorithm for target states. Multi-packet wave functions are treated within a multi-target optimization procedure. For target states the control functional J can be chosen either as

$$J_1(E) = N_{\rm tar}^{-1} \sum_i^{N_{\rm tar}} \left|\left\langle \Psi_i(T) | \Psi_{({\rm tar},i)} \right\rangle\right|^2 - \alpha_0 \int_0^T dt \frac{E^2(t)}{S(t)}, \tag{2.1}$$

or as

$$J_2(E) = N_{\rm tar}^{-2} \left| \sum_i^{N_{\rm tar}} \left\langle \Psi_i(T) | \Psi_{({\rm tar},i)} \right\rangle \right|^2 - \alpha_0 \int_0^T dt \frac{E^2(t)}{S(t)}. \tag{2.2}$$

Here α_0 is the so-called penalty factor that penalizes for strong fields and $S(t)$ serves as a pulse envelope that can be defined in the operator file. The functional (2.1) leads to optimizations of the target state populations only while within (2.2) the phases are also aligned. [16] If $N_{tar} = 1$, both functionals are identical.

Example inputs are provided under $MCTDH_DIR/inputs/optcntrl. The Python script **optcntrl** parses the input file and invokes OCT related programs from the MCTDH package. Note: the script reqires **Python** 2.4 or 2.5 and relies on the **Python** executable being found in /usr/bin/env python. This path can be changed in the first line of the script $MCTDH_DIR/bin/python/optcntrl.py.

From the input file a number of temporary input files for the actual MCTDH calculations are created. The same applies for the operator file. The operator file must contain at least two operators, the system Hamiltonian including the dipole operator multiplied with the electric field and a second operator containing the dipole operator alone. The system Hamiltonian is used to perform the propagations while the dipole operator is used to evaluate the electric field.

To execute an example, create a new empty directory, change to it, and copy the example input files, e.g. “pyrazine.inp” and “pyrazine.op” (see Ref. [17]). Inspect the input file and run the command:

```
optcntrl -mnd pyrazine.inp
```

The command

```
optcntrl -h
```

will provide help about options.

Recently, the a number of new features have been implemented into OCT-MCTDH, such as optional filtering of the field and the use of different optimization schemes. [**?**, **?**, **?**] Please refer to the HTML documentation for details.

2.10 Concluding Remarks

This tutorial has shown you some typical applications of MCTDH. In order to ensure that within this tutorial all calculations can be done quickly – only the optimal control example takes somewhat longer, 60 minutes (on a 3 GHz PC) – we have chosen rather small example systems. This, however, should not mislead you, MCTDH is for treating *large* systems! The full power of MCTDH is uncovered when turning to problems which require such a large primitive product grid, that a standard (numerical exact) wavepacket propagation becomes impossible on a workstation. A good example for such a problem is the calculation of the absorption spectrum of pyrazine, as discussed in Ref. [6]. There the primitive product grid amounted to 6.6×10^{20} points whereas the MCTDH calculation required only 3.76×10^{6} configurations, 687 MByte RAM and $52h$ CPU time. (This was done in 1998 on an IBM RS/6000 power2 workstation, which is much slower than any modern PC.)

Chapter 3

Defining the type of calculation to be made

In this chapter we present how to define and start the calculation to be made. Possible types are propagation, relaxation or diagonalisation. Propagation and relaxation calculations can be performed either using the MCTDH method, or numerically exactly, i. e. using the full primitive product grid. We also give a brief overview of the output to be produced.

3.1 Specifying the task for QUANTICS

The QUANTICS program package can perform different tasks specified in the RUN-SECTION. The following tasks are possible:

Keyword	Level	Description
`gendvr`	1	A DVR file will be generated (see Sec. 4).
`genoper`	2	An operator file will be generated (see Sec. 6).
`genpes`	2	A special operator file, called pes, will be generated, that contains the potential energy surface (see below).
`gengmat`	2	A special operator file, called pes, will be generated, that contains an element of the G-matrix of the kinetic energy (see below).
`geninwf`	3	An initial wavefunction (restart file) will be generated (see Sec. 7).
`test`	4	All input files will be checked and all other files, necessary for a propagation, will be created, but no propagation step will be performed (see below).
`propagation`	4	Propagation in real time (see Sec. 3.3).
`relaxation`	4	A relaxation or an improved relaxation will be performed (see Sec. 3.4 or Sec. 3.5 respectively).
`continuation`	4	A continuation of the run will be performed (see Sec. 3.9).
`diagonalisation`	4	The Hamiltonian will be diagonalised (see Sec. 3.7).

When a high level task is to be performed, the necessary tasks of lower order are automatically included. E.g. by setting the keyword `propagation` one implicitly also sets `gendvr`, `genoper` and `geninwf`.

The pes-files, created with the keywords `genpes` or `gengmat` (or the options `-pes` or `-gmat`), are special operator files which include only potential like operators. Note that all non-diagonal (i.e. kinetic energy) terms are automatically removed from the Hamiltonian when setting the keyword `genpes`. The pes-file is usually used in the context of the analyse routines **vminmax** or **showsys** (see Sec. 11.10). The **showsys** program can plot 2D-cuts through potential energy surfaces (or other multidimensional functions) provided in the form of a pes-file. The pes-file, created with the `genpes`-keyword, contains the potential energy surface of the system. If the keyword `gengmat = I1,I2` (or the option `-gmat I1 I2`) is used then the pes-file contains the (I1,I2) matrix element of the G-matrix of the kinetic energy operator. A kinetic energy operator can always be written in the form:

$$T = \frac{1}{2} \sum_{i,j}^{f} p_i^\dagger G_{ij} p_j + \sum_{i=1}^{f} F_i p_i + V_{extra} \tag{3.1}$$

where G_{ij}, F_i and V_{extra} are potential like terms. This equation defines the G-matrix. Note that the kinetic energy, defined in the Hamiltonian-Section, does not need to be of this particular form.

A further useful task is performed if the keyword `test` is used. In this case all input files are checked and all output files are created (even a psi file for t=0) like in a propagation run, but no propagation step is done. A test run may hence be used to convert a restart-file into a psi-file. It is possible to start a continuation run (see Sec. 3.9) after a test run.

3.2 Specifying the desired output

The output produced by the QUANTICS program is sent to a directory called the name-directory. The (absolute or relative) path of this directory is specified by the `name` keyword in the RUN-SECTION of the input file. The name-directory should already exist when the QUANTICS program is started. Otherwise one may use the option `-mnd` (make name directory) to create a new directory.

During run-time, a number of files are generated in the name-directory. Some of these are always created while others have to be selected by the user. The most interesting file of the former category is the log file, which records what happens at the various stages of a run.

Frequently used files of the second category are the output, auto, and psi files. The output file contains some basic physical quantities of the wavefunction, such as norm, energy, state populations, and the position and momentum expectation value of each coordinate. The auto file contains the auto-correlation function as a function of time. Both files are in ASCII format. In the psi file the wavefunction as a function of time is stored in binary format. The three files are selected by placing the keywords `output`, `auto`, or `psi` in the RUN-SECTION. Examples for the RUN-SECTION are given in the following sections. A complete list of the available files and options can be found in the HTML documentation.

The files gridpop, check, steps, update, timing, and speed only serve to check the efficiency or accuracy of an MCTDH calculation. They are very useful during the test phase of your calculations. Since some of them might become rather large, however, you may turn them off for your production calculations. NB the files output, timing, speed, and (for CMF runs) update are opened by default. To turn them off, use the keywords `no-output` (or `screen`), `no-timing`, etc.

3.3 Propagating a wavepacket

For performing a wavepacket propagation using the MCTDH method you first have to set up the Hamiltonian in an operator file (see Sec. 6). This operator file must then be specified in the OPERATOR-SECTION of the input file (see Sec. 6.1). In the input file the primitive and single-particle basis (Secs. 4 and 5), as well as the initial wavefunction (Sec. 7), must also be defined. Finally, you may select an integration scheme different from the default (Sec. 8).

A wavepacket propagation is then initiated by placing the keyword `propagation` in the RUN-SECTION. A typical example is

```
RUN-SECTION
propagation
tfinal = 50.0    tout = 1.0
name = results
psi  auto  gridpop
end-run-section
```

The parameters `tfinal` and `tout` denote the time the propagation will run up to and the time interval after which the data is output (in femtoseconds). The name-directory is results in our example. The other parameters have been established in Sec. 3.2. A number of additional options may be selected in the RUN-SECTION. We refer the reader to the HTML documentation for details.

3.4 Relaxing a wavepacket to produce the lowest eigenstate

In a relaxation calculation a wavepacket is propagated in imaginary time to produce the lowest ro-vibrational eigenstate. Analogous to a (real-time) propagation, an operator and an input file are required to define the Hamiltonian, the primitive and single-particle basis, the initial wavefunction, and possibly the integration scheme.

A relaxation calculation is selected in the RUN-SECTION by the keyword `relaxation` instead of `propagation`. The RUN-SECTION may read

```
RUN-SECTION
relaxation
tout   =  10.0
tfinal = 100.0
name = results
end-run-section
```

The parameters have been introduced in Secs. 3.2 and 3.3.

After a successful run, the desired lowest eigenstate is stored in the restart file. This eigenstate can then be used as initial wavefunction in following calculations, by using the `file` keyword in the INIT_WF-SECTION.

To check the convergence of a relaxation calculation with respect to the propagation time `tfinal`, you may look at the total energy being displayed in the output file. If this has not changed significantly during the last outputs, the eigenstate is converged. Otherwise, the calculation should be continued to longer times. See Sec. 3.9 for continuing calculations.

3.5 *Advanced topic*: Improved relaxation. Generation of excited eigenstates

This section may be difficult to understand, if one is not familiar with the constant mean field (CMF) integration scheme. For a brief discussion on CMF see Sec. 8.2 and 8.4.2. A more comprehensive discussion on CMF can be found in the MCTDH review. The *Improved Relaxation* algorithm is discussed in the MCTDH feature article [4] and more recently (and more comprehensively) in refs. [**?**, **?**, 18].

Consider a relaxation run where the CMF integration scheme is adopted and the SIL integrator is used to propagate the A-vector. In this case it seems to be somewhat cumbersome to generate a relaxed (i.e. propagated in imaginary time) A-vector by taking a linear combination of the eigenvectors of the Lanczos matrix. It is more meaningful to replace the relaxed A-vector by the ground state of the Lanczos matrix, as one is interested in the ground-state but not in a proper propagation in imaginary time. This modification leads not only to a faster convergence but, more importantly, offers the possibility to converge to an excited state. One simply takes the n-th state of the Lanczos matrix as "relaxed" A-vector. However, the dimension of the Lanczos matrix is in general small compared to the length of the A-vector. The pseudo-spectrum of the Lanczos matrix is thus a poor simulation of the spectrum of the matrix representation (in the set of the single particle functions) of the Hamiltonian H. Although the n-th eigenstate of the Lanczos matrix may be a good approximation to an eigenstate of the H-matrix, it may not approximate the n-th state of H, but a higher one. Thus the algorithm of improved relaxation will converge to some eigenstate, but one is not sure, which one it is (except when the ground-state is sought).

To deal with this situation, the algorithm is set up such, that that eigenvector of the Lanczos matrix is taken, which has the largest overlap with the one of the previous CMF step. (A warning is written to the log file, if this overlap-squared is smaller than 0.66). There are two ways to select the initial Lanczos vector. The first way is to write `relaxation = ` *n* to the `RUN-SECTION`. The n-th eigenvector of the Lanczos matrix is then taken as starting point of the relaxation. The counting starts from zero, i.e. $n = 0, 1, 2, \ldots, 98$. Note that the Lanczos matrix depends on the starting vector, which in this case is the initial wavefunction defined in the `INIT_WF-SECTION`.

The second way is to specify `relaxation = follow`. In this case the starting vector of the improved relaxation is that eigenvector of the Lanczos matrix, that has the largest overlap with the initial wavefunction defined in the `INIT_WF-SECTION`.

The Krylov space and thus the dimension of the Lanczos matrix grows with each step of the Lanczos iteration.This process is stopped when the accuracy criterion is satisfied (the SIL accuracy parameter is now interpreted as the tolerated error in milli Hartree of the energy of the desired Lanczos eigenvalue) or when the specified maximal dimension of the Lanczos matrix is reached. If the keyword `full` is given as a second argument of the `relaxation` keyword, then during the very first build up of the Krylov space the iteration will be continued till the maximal dimension is reached. This feature is useful to ensure that the improved relaxation starts from the correct Lanczos eigenvector. There is also the keyword `ortho`, which may appear as an argument to the `relaxation` keyword. `ortho` forces the SIL integrator to perform a full re–orthogonalisation of the Krylov space. This often significantly improves the convergence, but for long A-vectors it may take some CPU–time. In short, the improved relaxation command may read `relaxation=follow,full,ortho`.

As the A-vector now changes discontinuously, the standard variant of the CMF step size control, `CMF/var`, does not work. One must either work with fixed CMF steps, `CMF/fix`, or let the step size control depend on the single particle functions only, `CMF/varphi`. The latter choice is usually to be preferred. For convenience `CMF` is set equivalent to `CMF/varphi` when the run-type is improved relaxation. In all other cases `CMF` is interpreted as `CMF/var`. Note that the integrator parameters, initial CMF step size, CMF accuracy (only for varphi), maximal Lanczos space and SIL accuracy, may have decisive effects on the convergence. The SIL accuracy and space should be chosen higher than for propagation. Use an SIL accuracy between 10^{-8} and 10^{-11} and a Lanczos space between 20 and 200 (the maximal size is 500). The higher the sought eigenstate lies, the larger must be the Lanczos (or Krylov) space. The convergence to higher lying states also requires more SPFs than needed for propagation (and as indicated by the natural weights).

The single-particle functions are propagated in imaginary time, i. e. they converge toward the lowest states of the mean-field operator, irrespectively whether they are needed for representing the wavefunction or not. If, for example, there is no node-less single-particle function approximating the ground-state of the mean-field operator, then the propagation in imaginary time will rapidly change the single-particle functions in order to generate such a function. Due to this rapid change, the method then fails to converge. One thus may have to select the initial single-particle functions somewhat more carefully. However, there is no implemented algorithm to do so and one may therefore be forced to use more single-particle functions for improved relaxation than finally needed to represent the converged wavefunction.

When the keyword `orben` is set in the RUN-SECTION, then the so called *orbital energies* are computed and output to the file orben. The orbital energies are obtained by diagonalising an appropriate mode-Hamiltonian in the set of the single-particle functions. The mode Hamiltonian is conveniently defined as the trace over the mean-fields of the particular mode under discussion. I. e.

$H_{av} = \sum_{j=1}^{n_\varkappa} \langle \boldsymbol{H} \rangle_{jj}^{(\varkappa)}$.

See the MCTDH-review for a definition of the mean-field operators $\langle \boldsymbol{H} \rangle^{(\varkappa)}$.

The eigen-functions of H_{av}, called *energy orbitals*, allow to define *energy weights* as the diagonal values of the density matrix expressed in the basis of the energy orbitals. The energy weights are very useful for assigning quantum numbers to a relaxed wavefunction. If for each mode there is one weight that is close to one (larger than 2/3, say), then the full wavefunction is characterised by the quantum numbers of the dominant energy orbitals. For further information see the HTML documentation under `Input-Documentation/Run-Section` and `Output-Documentation/Data-files`.

More recently (Aug 2003) a Davidson diagonaliser has been implemented to replace the SIL for improved relaxation. The Davidson algorithm is much superior to Lanczos when the generation of a single excited eigenstate is asked for. For the Davidson, there is a new keyword `relaxation=lock`. This is similar to `relaxation=follow` but more stable, because `lock` searches for the eigenvector with the largest overlap with the initial wavefunction, whereas `follow` takes the eigenvector with the largest overlap with previous vector. The keyword `ortho` is not allowed when using the Davidson, and `full` is useful in special cases only. See the HTML documentation for more details. A typical input for improved relaxation with Davidson is provided by inputs/hono.dav.inp . Note that in this example one is computing the about 120th eigenstate in A' symmetry.

There are several versions of the Davidson diagonaliser implemented. With the keyword `DAV` one calls a routine which works for hermitian Hamiltonians. With the aid of the keyword cDAV one may diagonalise non-hermitian (e. g. CAP augmented) Hamiltonians and compute complex resonance energies. However, in most applications the Hamiltonian will not only be hermitian, it will also be real. (A Hamiltonian is called real, if $H\Psi$ is real for every real Ψ). As QUANTICS is written to propagate wave packets, almost all variables are declared as complex. Hence one is wasting memory and CPU time by not making use of the reality of the wave function. To reduce this waste, one may use the keywords rDAV or rrDAV. rDAV is similar to DAV, but it stores the Davidson vectors as real. This substantially reduces the memory demand. rrDAV performs in addition the matrix-vector operation HA in real arithmetic, which speeds up this step. Note that rrDAV can be used only, if each single operator appearing in the Hamiltonian is real. This, in particular, excludes the use of the operator p. (One may replace it by the operator dq). Furthermore, rrDAV cannot be used, if there are so called muld-operators (i. e. multi-dimensional operators which act on more than one MCTDH particle simultaneously). The use of natural potentials, however, is allowed. Note that the relaxation of the SPFs is always done in complex arithmetic.

Turning more to the technical side we note that the integrator setting for improved relaxation is quite different from the one for propagation. For improved relaxation (with Davidson) the CMF accuracy is usually set to a rather low value, around 10^{-3}, whereas high accuracies, $10^{-7}\cdots 10^{-8}$ and $10^{-8}\cdots 10^{-10}$, are to be used when propagating the SFPs and the A-vector, respectively. (See Sec. 8.4.2). For propagating the SFPs we recommend the RK8 integrator. In contrast to a propagation run one now prefers to have an output after each update step, i. e. after each diagonalisation. To achieve this, one sets tout=all. In this case one should also give tpsi, even if the psi file is not written. The tpsi time is taken as the upper limit for the update interval.

When computing the ground state, the Davidson diagonaliser is usually quite fast, i. e. requires only few iterations. Turning to excited states, however, the number of Davidson-iterations may become quite large. In order to speed up the calculation in this case, a better pre-conditioner was implemented. If one sets precon=N then an $N \times N$ block of the Hamiltonian matrix is diagonalised and used to improve the pre-conditioner. One should be careful not to use a too large value for N, otherwise the build-up of the pre-conditioner takes more time as saved by performing a smaller number of Davidson iterations. It is usually not useful to use precon when the ground state is computed. When computing higher excited states, on the other hand, the pre-conditioner can be very helpful.

When excited states are to be computed one usually uses relaxation=lock. This, however, requires that an initial state is provided, which has a decent overlap with the state to be computed. There are two convenient ways to generate such an initial state. The first one is to operate with some excitation operator on the ground state, or on some converged excited state. (Compare with the co2_*.inp input files on MCTDH_DIR/inputs). The other way is to diagonalise appropriate 1D-operators with eigenf or mode-operators with meigenf and build the initial state as Hartree product from those low-dimensional eigenstates. (Compare with the hono.dav.inp and H2CS.r1r2.inp input files on MCTDH_DIR/inputs). Furthermore, one may use the orthogonalise keyword to purify an initial state from contributions of already converged eigenstates. (See HTML documentation under INIT_WF-SECTION).

When the CI-pace is too small (i. e. when the A-vector length is too short) the improved relaxation algorithm will not converge due to a “variational breakdown”. Not only the state requested but all states below this one must be representable by the SPF basis sets. For highly excited states this may require large CI-spaces. One must be careful when using mode combinations and avoid too strongly combined modes because over-combination will make the CI-space too small. Relaxation to the ground state is always unproblematic. If a relaxation to an excited state does not converge, one has to use more SPFs, although the natural populations may already be very low. The natural population indicates how important an orbital is for representing the desired state. But again, here the orbitals have to represent in addition all states below the desired one.

Rather than (single) relaxation one may use block-relaxation. See Sec. 2.7 for an example. The keyword `split-rst` splits the restart file of the block-relaxation into individual restart files for each eigenstate. Block-relaxation does not allow the use of the keywords-arguments `follow` or `lock`, only `relaxation=0` is allowed. Hence one computes the b lowest eigenstates, where b denote the block size. However, one may set the keyword `rlxemin` (see the HTML docu for a full description). This will force the code to compute the b lowest states *above* the argument of `rlxemin`.

An initial wavefunction for a block improved relaxation run can be read by using the keywords `block-spf`, `block-A`, or, if there is already a wavefunction in correct block form, simply by using the `file` keyword. The keyword `autoblock` is particularly useful. See the HTML docu for a comprehensive description of these keywords.

Finally we note that the rlx_info file is most conveniently read with the aid of the script **rdrlx**. Type **rdrlx -h** for obtaining more information. The script **plrlx** (or **plbrlx** for block-relaxation) plots the energy versus relaxation time.

3.6 Performing a numerically exact calculation

The QUANTICS program — although originally developed for wavefunction dynamics within the MCTDH scheme — also allows one to perform numerically exact wavefunction propagations. Employing the QUANTICS program for such calculations has the advantage that one can benefit from the easy way a Hamiltonian can be set up. A numerically exact calculation may also be useful for comparison with an MCTDH calculation. Of course, a numerically exact propagation is only feasible for rather small systems.

For a numerically exact wavepacket propagation an operator file and the same input sections as in an MCTDH calculation are required. A numerically exact calculation can be made by including the keyword `exact` in the RUN-SECTION in addition to the calculation type keyword (e.g. propagation or relaxation). Rather than using the low-dimensional MCTDH single-particle functions, this sets up a wavepacket (or for a non-adiabatic system one for each electronic state) on the full product primitive basis. The operator is also set up on this full grid.

To propagate this wavepacket any of the integrators listed in Tab. 8.1 can be used. Although the ABM integrator is the default, the best performance is typically obtained by the SIL integrator, because it exploits the fact that the equations of motion in a numerically exact calculation are linear. To select the SIL integrator, insert for instance

```
INTEGRATOR-SECTION
  SIL = 20, 1.0d-6
end-integrator-section
```

into your input file. The parameters are discussed in Sec. 8.3. The ABM, BS or RK5/8 integrator can be chosen as described in the examples in Sec. 8.

3.7 Diagonalising the Hamiltonian using the Lanczos algorithm

The QUANTICS program is also capable of diagonalising a Hamiltonian using the Lanczos scheme. In such a diagonalisation run the wavefunction is automatically represented on a (primitive) product grid, i.e. in the same way as in a numerically exact calculation.

A diagonalisation run therefore requires the same input sections as a numerically exact calculation, and of course an operator file. A possible RUN-SECTION reads

```
RUN-SECTION
diagonalisation = 10000
name = results
end-run-section
```

The number associated with the `diagonalisation` keyword indicates the number of Lanczos iterations to be made. The `name` keyword has been explained in Sec. 3.2. Other keywords that may be specified in a diagonalisation run can be found in the HTML documentation.

The computed eigenenergies and intensities, together with an error estimate of the eigenenergies, are compiled in the eigval file. If the error of the desired eigenvalues is to large, the calculation can be continued to increase the number of iterations. See Sec. 3.9 for continuing calculations.

Note that one-dimensional Hamiltonians can be numerically exactly diagonalised using a DVR basis. See Sec. 7.6 for more details of this.

3.8 Starting a calculation

The MCTDH program is started by typing

```
quantics myinput
```

on your console, where we assumed that your input file is named myinput.inp. If the input file is not stored in the current directory, add the correct path to the input file's name. A variety of options may be used on starting the QUANTICS program. Type

```
quantics -h
```

to get an overview.

To find out during run-time how far a propagation is proceeded, you may look for instance at the last line of the log file, by typing

```
tail -1 results/log
```

where *results* has to be substituted by the name-directory.

Any error messages that might be raised during run-time are sent directly to the screen and additionally written to the log file. See there in case of problems. If the calculation was successful, the results can be analysed by the Analyse programs and scripts. This is detailed in Chap. 11.

3.9 *Advanced topic*: Continuing or stopping a calculation

A calculation that has been finished may be continued again, so propagating to longer times or performing a larger number of Lanczos iterations. This is done by adding the keyword `continuation` to the RUN-SECTION. (Alternatively, one may start a continuation run using the `-c` option on the command line, see the HTML documentation for details.) In a propagation run the final time must also be increased. For instance, if the previous calculation finished after 50 fs, then you may set `tfinal = 75.0` to propagate over the next 25 fs. Similarly, if 10 000 iterations were made in a diagonalisation run, set `diagonalisation = 15000` for the next 5 000.

Again, it is often more convenient to use options. E. g. the command

```
quantics -c -tfinal 75.0 <inputfile>
```

will start a continuation run where the final time is set to 75 fs. The use of other options like `-I`, `-tcpu` or `-tstop` may be useful. Type **quantics -h** to see the list of options. Note that a continuation run reads only the RUN-SECTION and ignores all the other sections of the input file as well as the operator file. All the necessary information is read from the read-write files of the name directory.

Sometimes it is desirable that a calculation is continued with a different integrator setting. This can be accomplished by giving the keyword `continuation=integrator` or by the option `-ci`. In this case the INTEGRATOR-SECTION will be read.

Finally, the `continuation` keyword can be used in order to try to complete a crashed calculation. The program however does not check the output files for consistency. The continuation thus might fail if some relevant data was lost due to the crash.

Another option of the QUANTICS program is to stop a calculation during run-time in a controlled manner, such that it can be resumed later. To this end include the stop file by adding the keyword `stop` to the RUN-SECTION. To halt a calculation after the next output, edit the stop file and write `stop` to its first line. Alternatively one may just remove the stop file. The stop is automatically deleted when the run finishes. It thus may serve as a lock-file. As long as the stop file exists, the run is not finished.

Rather then simply writing `stop` to the stop file one may supply more specific commands which will halt the program after a certain real-time or CPU-time has passed. This allows e. g. to outwit CPU-time limits. Rather than editing the stop file ”by hand”, one may let the QUANTICS program do that. This can be accomplished by giving the keywords `tcpu` and/or `tstop` in the RUN-SECTION or by using the options `-tcpu` and/or `-tstop`. See the HTML documentation for details.

3.10 *Advanced topic*: Using parallel shared memory hardware

If shared memory hardware is available QUANTICS can take advantage of it. The parallel features of QUANTICS are used if the keyword `usepthreads = I` is set in the RUN-SECTION, where `I` stands for the number of processors that should be used. Further arguments can be added, which disable the parallelisation of the different QUANTICS routines. The parallelisation of the following routines can be disabled: phihphi (`no-phihphi`), calcha/funka2 (`no-funka`), summf (`no-summf`), mfields (`no-mfields`), hlochphi (`no-hlochphi`), hlochphi1m (`no-hlochphi1m`), funkphi (`no-funkphi`), getdavmat (`no-getdavmat`) and dsyev (`no-dsyev`). In the QUANTICS code the funkphi routine calls several subroutines: hlochphi1m, mfsumphi1m, dicht1phi1ms, hunphi1ms, addhunphi1ms and project1ms. (The parallelisation of the first routine is switched off by setting the `no-hlochphi1m` keyword. Setting the `no-funkphi` keyword switches off the parallelisation of the other routines all together.) The funka2 routine is only used in the case of a relaxation run with the `rrDAV` integrator.

There are more keywords. The `mem-calcha` and the `mem-mfields` keywords enable QUANTICS to use more memory for a more efficient parallelisation of the calcha and mfields routines. In default mode the parallelisation is optimized for low memory requirements. Furthermore there is the `summf2` keyword. If this keyword is set, a differently parallelised summf routine is used. This may increase the efficiency of the parallelisation if a very large combined mode is present, dominating the calculation of summf. A parallel version of the LAPACK dsyev-routine is used in QUANTICS. The `dsyev = I` keyword can be used to set the minimum size of the matrix that is diagonalized in parallel.

The results of a parallel calculation may slightly differ from those of a non-parallel one due to numerical reasons. Furthermore a parallel calculation needs more memory. This is one of the reasons why the parallel use of the routines can be disabled. Depending on the type and the parameters of the calculation some routines may only marginally improve the performance of the parallelisation. They can be turned off to save memory. Moreover, depending on the calculations made, some routines can produce overhead which overcompensates the gain of their parallelisation. These routines should also be turned off.
Example:

```
RUN-SECTION
usepthreads = 4, no-funkphi
...
end-run-section
```

In this example a parallel calculation with 4 processors will be performed but the parallelisation of the funkphi routine is disabled. Sometimes disabling the parallelisation of routines even increases the computational speed, as it is the case for propagation of C_2H_4. If the calculation is made with all routines running in parallel mode the parallel part of the program, according to Amdahl's law, is 61% whereas the parallel part is 75% if the parallelisation of the funkphi routine is disabled.

In general the parallelisation works better for larger systems, i.e. systems with many Hamiltonian terms and many single particle functions. This is due to the fact that either loops over the Hamiltonian terms or loops over the single particle functions are parallelised. We have observed that the performance of the parallelisation does not only depend on the system studied but also on the computer platform and compiler. The C_2H_4 system has been propagated on an Itanium cluster using the Intel compiler. This resulted in a speedup of 2.30 (4 processors) for which Amdahl's law states a parallel parallel part of 75%. The same system, propagated on a quad-opteron, also with the Intel compiler, showed a speedup of 2.58 (4 processors) and hence a parallel part of 81.7%.

One of our better examples is $H_2 + H_2$ inelastic scattering. Here the Hamiltonian consists of many terms because there is a large potfit. A speedup of 6.19 is observed when running this system on 8 processors in parallel. This implies that 95.5% of the work is done in parallel. The memory used increased form 56 MB (one processor) to 60 MB (8 processors).

The performance of the parallelisation can be monitored using the keyword `ptiming`. If this keyword is set in the RUN-SECTION an additional timing file, called ptiming, is created containing information about the time spent in each thread and routine. This keyword also can be used if `usepthreads` is not set. In this case no ptiming file is created, but the timing file contains information about the timing of the parallelised routines. This helps to decide what routine should be used in parallel mode if the parallelisation is turned on.

The ptiming file is structured in the following way ($H_5O_2^+$ propagation):

```
Subroutine          Calls        cpu          sum    thread: 1  thread: 2
 phihphi               52     822.72       822.83     411.44     411.39
 calcha               168   11297.70     11363.55    5680.55    5683.00
 mfields               51    8054.95      8055.37    4026.99    4028.38
 summf                 51    4991.97      9529.60    4764.80    4764.80
 hlochphi1m          5617   11071.72     11204.26    5667.22    5537.04
 funkphi             5237     716.71       726.63     329.93     396.70
```

The first column shows the name of the parallel subroutine, the next column gives the number of calls to this routine. The column “cpu” shows how much cpu time is spend for the computation. The columns “thread: p” give the real time spend in each thread (here 2) for the computations made, these values are summed up in column “sum”. In this example the parallelisation of the summf routine works badly. This can be seen, because the cpu time for the summf evaluation is much lower than the sum of the realtime spend for this task. Using the `summf2` keyword this problem can be fixed. The corresponding line in a `summf2` run indicates that the parallelisatioin now works better:

```
Subroutine          Calls       cpu          sum   thread: 1  thread: 2
   ...
 summf             188394    3809.27     3868.90    1917.64    1951.25
   ...
```

Ignore the number of calls. It is increased because now not summf is timed but an internal routine called by summf. More things can be seen with help of the ptiming files. Sometimes the parallelisation creates cpu time overhead. This cannot be discoverd by checking only one ptiming file. In the case of the C_2H_4 propagation the routines controlled with the `no-funkphi` keyword produce overhead. But the corresponding ptiming file is:

```
Subroutine          Calls        cpu          sum    thread: 1  thread: 2
   ...
 funkphi           121389     606.69       627.99     308.03     319.96
```

Here the columns “cpu” and “sum” compare very well. But the ptiming file for one thread, i.e. `usepthreads=1`, reads:

```
Subroutine          Calls        cpu          sum    thread: 1
   ...
 funkphi           120780     174.36       175.23     175.23
```

Comparing the “cpu” column of both files a strong increase of cpu time is discovered, 174.36s (1 proc) to 606.69s (2 proc). Hence the `no-funkphi` keyword should be used to avoid this overhead. Finally we give some overall timings:

	1 processor	4 processors		
		default	no-summf	summf2
$H_5O_2^+$	9h 52m 20s	3h 50 m 52s	3h 52m 06s	3h 09m 49s
		default	no-funkphi	
C_2H_4	22m 21s	12m 20s	8m 49s	

If the `ptiming` keyword is used but the lrt library is not linked the following error message appears.

```
###########################################
###  --- no clock_gettime command ---
###  lrt not linked!
###  If lrt is available,
###  modify the script compile.cnf
###########################################
```

If your compiler supports this library the script compile.cnf has to be modified.[1] The option `-lrt` must be added in the line `MCTDH_ADD_LIBS` in the section of the compiler that is used. Then QUANTICS must be compiled again.

A similar error message appears if the pthread library was not linked for compilation of QUANTICS.

```
###########################################
###  --- no xxx_yyyy command ---
###  pthread library not linked!
###  If your compiler supports pthread,
###  modify the script compile.cnf
###########################################
```

`xxx_yyyy` is replaced by the name of the routine the program tried to execute but could not be found. If your compiler supports pthreads, the script compile.cnf has to be modified (see above). The option `-lpthread` must be added in the line `MCTDH_ADD_LIBS` and the option `-pthread` must be added in the line `MCTDH_CFLAGS` in the section of the compiler that is used. Then QUANTICS must be compiled again.

For the standard compilers, e.g. GNU or Intel, all the necessary extensions of the compile scripts are already done.

To improve the efficiency of the shared memory parallel QUANTICS on NUMA (non uniform memory access) machines the numa.h library can be linked [2]. To do so the -u option must be given for compilation:

```
compile -u quantics
```

Doing so the POSIX-threads created during an QUANTICS-run are distributed and bound cyclicly to the available processors to prevent thread migration. Of course, this only works if the numa library is available on your computer.

On NUMA machines (e.g. typical Opteron or Xeon clusters) we observed that execution times of identical runs may vary by more than 20%. Using the numa.h library, however, these surprising variations vanish and all runs take almost identical execution times. These times agree with the shortest times observed without using numa.h.

3.11 *Advanced topic*: Using parallel distributed memory hardware

Beside parallel shared memory hardware, QUANTICS can make use of parallel distributed memory hardware. For the distributed memory parallelization of QUANTICS the Message Passing Interface (MPI) was used. To use the MPI-parallel QUANTICS, the keyword

[1] One should edit compile.cnf_be and compile.cnf_le as well because compile.cnf is overwritten by one of the latter files when **install_quantics** is executed.

[2] You may visit http://oss.sgi.com/projects/numa/ to find more information about NUMA API

`usempi` must be set in the RUN-SECTION and the QUANTICS program must be started with the `mpirun` command. Further the MPI compilation of QUANTICS has to be used (see below). To invoke an MPI-parallel run with 32 MPI-prozesses the command could look like this:

```
mpirun -np 32 quantics.mpi-g77 <inputfile>
```

Further arguments can be added to the `usempi` keyword, which disable the parallelisation of the MPI-parallel routines. These routines are: calcha (`no-calcha`), funka2 (`no-funka2`), mfields (`no-mfields`), getdavmat (`no-getdavmat`), dsyev (`no-dsyev`), phihphi (`no-phihphi`), hlochphi (`no-hlochphi`) and mpirdavstep/mpibdavstep (`no-dav`). These keywords are similar to those used in shared memory parallelisation with the `usepthreads` keyword but the last one, `no-dav`, is different. This keyword disables the usage of the MPI-parallel routines mpirdavstep/mpibdavstep. This means that the davidson vectors that are built during an improved relaxation step are stored on one node and are not distributed. Additionally there is the keyword `dav=I`, where `I` stands for the maximum number of davidson vectors that are stored on one node. If this keyword is not set the maximum allowed number of vectors (integration order) is equally distributed over the available nodes (intorder/# of MPI-prozesses). If `I` is set to a higher value this can lead to smaller communication costs because the vecors are held on a smaller number of prozessors.

As in the case of the shared memory parallelisation the results of a parallel calculation may slightly differ from those of a non-parallel one due to numerical reasons. An example for a RUN-SECTION in an MPI-parallel run is shown in the following Example:

```
RUN-SECTION
usempi, no-dsyev
...
end-run-section
```

Here the parallelisation of the diagonalisation routine `DSYEV` is turned off. The number of prozesses is not specified in the RUN-section as is must be done in the shared memory parallel case. This is done via the `mpirun`-command (see above).

In the case of the distributed memory parallelization not all parts of QUANTICS are parallelized, only the mean-field computation and the A-vector propagation are parallel but not the SPF-propagation. This is due to the fact that large calculations are A-vector dominated ones in general. Hence calculations with a considerable contribution from the SPF propagation to the cpu time are not suited for the MPI-parallel QUANTICS. The best tested case was the $H_5O_2^+$-propagation where up to 1024 prozessors were used. This calculation showed a maximum speedup of 118, this corresponds to a parallel part of over 99%.

If the keyword `ptiming` is set for an MPI-parallel run the file `mpitiming` is created. This file contains timing information for each MPI prozess. An `mpitiming`-file for a run with 8 prozesses is shown here:

```
 Process           cpu           %cpu
-------------------------------------
    0          14501.84          12.40
    1          15288.35          13.08
    2          14399.39          12.32
```

```
    3         13243.32         11.33
    4         17475.32         14.95
    5         13699.10         11.72
    6         13507.77         11.55
    7         14790.03         12.65
```

The first column denotes the MPI process, 0 being the master process, the second column shows the cpu time spent in each process and the third column gives the percentage of the cpu time. The Example shows a well paralellized case where the cpu time is equally distributed over the processes.

If the MPI parallelization is combined with the usage of POSIX threads the fourth column appears that shows the sum of real time spent in the threads of each prozess. This is similar to the summation done for the `ptiming`-files but here the summation was additionally done over the different routines. The following example is an QUANTICS-run with 2 MPI-processes with 4 POSIX-threads each, denoted by '(4)':

```
 Process          cpu           %cpu    sum real time ( 4)
---------------------------------------------------------
    0         51811.34         50.31     51574.81
    1         51168.49         49.69     51095.79
```

If the sum of real time is comparable to the cpu time the shared memory parallelization works well for each prozess (see the comments to `ptiming`-file in sec. 3.10).

For combined calculations, distributed and shared memory parallelization, a further option for the `ptiming`-keyword can be set: `ptiming=all`. If this is done a `ptiming`-file is created for each MPI-prozess, containing the timing information for the parallel routines. Thereby the quality of the shared memory parallelization within the different MPI-prozesses can be checked in more detail as by the `mpitiming`-file. These files are denoted by `ptiming0`, `ptiming1` and so on. This is mainly for testing.

To combine distributed and shared memory parallelization both, the `usepthreads` and the `usempi` keywords, must be set in the RUN-section and QUANTICS must be started with the `mpirun` command. For a parallel calculation where each MPI-prozess uses 4 POSIX-threads the RUN-section may look like:

```
RUN-SECTION
usepthreads=4
usempi
...
end-run-section
```

Here the parallelization for all routines is working. The MPI-parallel ones are called in each MPI-prozess where they run in shared memory parallel mode with 4 POSIX-threads. Additionally in the master-process the other shared memory parallel routines for the SPF-propagation are used.
Example:

```
mpirun -np 5 quantics.mpi-g77 <inputfile>
```

The g77-built QUANTICS is now run with 5 MPI-processes and 4 POSIX-threads per MPI-process i.e. 20 cores are used. The specific command to be given may depend on the queueing system of your computer and hence may differ from the example above.

To be able to start the MPI-parallel QUANTICS the program must be compiled with some additional options. For the standard compilers (g77, gfortran, pgf77, and ifort) additional sections are included in the script compile.cnf. Depending on the compiler the command is:

```
compile -m quantics

   or

compile -c mpi-xxx quantics
```

where `xxx` must be replaced by `g77`, `gfortran`, `pgf77`, or `intel`, respectively. If another compiler should be used an appropriate section in the compile.cnf script must be created. The `-m` option is a shorthand form. It requires that your default compiler is one out of the four above mentioned compilers. Note that it is essentially that your MPI package was created with a compiler compatible to the compiler you would like to use [3].

[3]Try to check `mpif77 -v` and see the manpages for `mpif77`.

Chapter 4

Selecting a DVR/FBR-representation for the primitive basis

DVR/FBR-representations are used to set up the single-particle functions of an MCTDH or the product grid of a numerically exact calculation. The DVR/FBR basis is also called primitive basis in the following. A comprehensive discussion of the DVR/FBR technique can be found in Appendix B of the review [1].

4.1 Available DVR/FBR-representations

The DVR/FBR-representations that are implemented in the MCTDH package are compiled in Tab. 4.1, along with examples for typical applications of them. Also given are the keywords by which the corresponding primitive basis types are selected in the input file.

The representations of the primitive basis are specified in the PRIMITIVE-BASIS-SECTION of the input file. The program evaluates this section and stores the information in the dvr file. The PRIMITIVE-BASIS-SECTION also serves to define the system coordinates, by allocating a label to each degree of freedom in the system. These labels are then used in other sections, such as the INIT_WF-SECTION, the SPF-BASIS-SECTION, and the HAMILTONIAN-SECTION to map the different sets of information onto the system coordinates.

A complete input file — for the photo-dissociation process of NOCl — is shown in Example 4.1. There, three Jacobian coordinates are defined, labelled `rd` (dissociative degree of freedom), `rv` (vibrational degree of freedom), and `theta` (angular degree of freedom). Their DVR-representations are sine, Hermite, and Legendre, respectively. The parameters associated with these representations are explained in the following sections.

4.2 Hermite and radial Hermite DVR

In the Hermite or harmonic oscillator DVR the harmonic oscillator functions

$$\chi_j(x) = \left(2^j j!\right)^{-1/2} (m\omega/\pi)^{1/4} \, H_j\Big(\sqrt{m\omega}\,(x-\tilde{x})\Big)\, e^{-m\omega(x-\tilde{x})^2/2} \tag{4.1}$$

are taken as basis functions. The Hermite DVR is thus typically used for vibrational modes. In the above equation H_j denotes the jth Hermite polynomial and j starts from zero, $j = 0, 1, \ldots, N-1$. Note that the Hermite DVR depends only on the product $m\omega$, which determines the width, and on $\tilde{x}$, which defines the centre of the grid.

Table 4.1: The DVR/FBR-representations for the primitive basis which are available in the MCTDH package. Also displayed are the corresponding keywords that are used in the input file, and a typical application for each basis type.

DVR/FBR-representation	Keyword	Typical applications
Hermite (harmonic osci.) DVR	`HO`	Vibrational modes
Radial Hermite DVR	`rHO`	Vibrational modes
Legendre (rotator) DVR	`Leg`	Angular modes (θ)
Restricted Legendre DVR	`Leg/R`	Angular modes (θ)
Sine DVR	`sin`	Vibrational, angular, and dissociative modes
Laguerre DVR	`Lagu`	Boundary condition $\varphi(r) \sim r^{1/2}$
Exponential (plane-wave) DVR	`exp`	perodic boundaries, ϕ-angular and diffractive modes
Fast Fourier transform (FFT)	`FFT`	Dissociative modes with large grids
Spherical harmonics FBR	`sphFBR`	Combined (θ, ϕ) angular modes
Extended Legendre DVR	`KLeg`	Combined (θ, ϕ) angular modes
Two–Dimensional Legendre DVR	`PLeg`	Combined (θ, ϕ) angular modes
Three–Dimensional rotational DVR	`wigner`	Combined (α, β, γ) angular modes

A variant of the Hermite DVR is the radial Hermite DVR, which is an appropriate DVR when the wavefunction is defined on a half-axis $[\tilde{x}, \infty)$ only, and satisfies the boundary condition $\psi(\tilde{x}) = 0$. The odd harmonic oscillator functions,

$$\chi_j(x) = \sqrt{2}\,\chi_{2j-1}^{\mathrm{HO}}(x)\,, \tag{4.2}$$

are chosen as basis, with $j = 1, \ldots, N$ and χ_k^{HO} given by Eq. (4.1).

A Hermite DVR may be selected in two ways. One is given by the following PRIMITIVE-BASIS-SECTION:

```
PRIMITIVE-BASIS-SECTION
    X    HO    36    0.00    0.10        1822.89
    Y    HO    36    0.00    2.721,eV    1.0,AMU
end-primitive-basis-section
```

Here we have assumed that the system under consideration has two degrees of freedom, labelled `X` and `Y`. The keyword `HO` specifies the harmonic oscillator DVR. The next entry denotes the number N of basis functions or, equivalently, grid points. The next three numbers define the equilibrium position $\tilde{x}$, the frequency ω, and the mass m. If the mass entry is missing, the program sets m to 1.

The above example also demonstrates the use of units. The default unit for times is femtoseconds, for all other input variables it is atomic units. (Hence exactly the same DVR is selected for the two modes `X` and `Y`.) A complete list of the available units can be found in the HTML documentation.

The second way to choose a Hermite DVR is to specify the first and last grid point, by employing the keyword `xi-xf`:

```
PRIMITIVE-BASIS-SECTION
    X    HO    36    xi-xf    -0.528    0.528
    Y    HO    36    xi-xf    -0.528    0.528
end-primitive-basis-section
```

Here the two parameters define the grid points x_1 and x_N. The program then computes the corresponding product $m\omega$, as well as the equilibrium position $\tilde{x} = (x_1 + x_N)/2$. The above example is thus equivalent to the previous one.

```
RUN-SECTION
  name = nocl1   propagation
  tfinal = 25.0   tout = 1.0
  psi=double auto steps gridpop
  title = NOCl, propagation, CMF 5 6 6, spf 5 5 5, prim 36 24 60
end-run-section

OPERATOR-SECTION
  opname = nocl1
  alter-labels
    CAP_rd = CAP [ 5.0d0  0.3d0  3 ]
  end-alter-labels
end-operator-section

SPF-BASIS-SECTION
  rd = 5    rv = 5    theta = 5
end-spf-basis-section

PRIMITIVE-BASIS-SECTION
#Label    DVR   N    Parameter
    rd    sin   36   3.800   5.600
    rv    HO    24   2.136   0.272,ev  7.4667,AMU
    theta Leg   60   0       all
end-primitive-basis-section

INTEGRATOR-SECTION
  CMF/var = 0.5 , 1.0d-5
  BS/spf  =  10 , 1.0d-6
  SIL/A   =  12 , 1.0d-6
end-integrator-section

INIT_WF-SECTION
  file = nocl0
end-init_wf-section

end-input
```

Example 4.1: An input file for a wavepacket propagation on the S1 surface of NOCl.

A radial Hermite DVR can be selected in exactly the same way than a Hermite DVR, but with `rHO` rather than `HO` as keyword.

4.3 Legendre DVR

A Legendre or rotator DVR is employed for angular degrees of freedom θ because the associated Legendre functions $P_l^m(\cos\theta)$ are eigenfunctions of the angular momentum operator $\hat{l}^2$. The basis functions are thus the $\mathcal{L}^2$-normalised associated Legendre functions,

$$\chi_{l-m+1}(\theta) = \sqrt{\frac{2\,l+1}{2}\,\frac{(l-m)!}{(l+m)!}}\,P_l^m(\cos\theta)\,, \tag{4.3}$$

with $m \geq 0$ and l restricted to $m \leq l \leq m+N-1$. The parameter m denotes the magnetic quantum number and is treated as a fixed parameter. The associated Legendre function P_l^m

is given by the polynomial

$$P_l^m(x) = \frac{(-1)^m}{2^l l!} \left(1 - x^2\right)^{m/2} \frac{d^{l+m}}{dx^{l+m}} \left(x^2 - 1\right)^l \, , \tag{4.4}$$

for $0 \leqslant m \leqslant l$.

A Legendre DVR is selected for a coordinate named `theta` by the line

```
theta     Leg     60     0     all
```

in the PRIMITIVE-BASIS-SECTION. The first number specifies the number N of basis functions or grid points, the second one denotes the magnetic quantum number m. The last keyword controls which symmetry is to be used. For `all`, all values $l = m, m+1, \ldots, m+N-1$ are employed, for `odd`/`even`, only odd/even values of l are taken, i.e. $l = m, m+2, \ldots, m+2N-2$ or $l = m+1, m+3, \ldots, m+2N-1$.

4.4 Sine DVR

The sine or Colbert-Miller DVR uses the particle-in-a-box eigenfunctions as a basis. The box boundaries are x_0 and x_{N+1}, and $L = x_{N+1} - x_0$ denotes the length of the box. The basis functions are thus

$$\chi_j(x) = \begin{cases} \sqrt{2/L}\, \sin\left(j\pi(x - x_0)/L\right) & \text{for } x_0 \leq x \leq x_{N+1} \\ 0 & \text{else} \end{cases} \, . \tag{4.5}$$

Note that the grid boundaries x_0 and x_{N+1} do not belong to the grid since the wavefunction vanishes there by construction. The sine DVR has been successfully used in MCTDH calculations for vibrational, angular, and dissociative degrees of freedom.

A sine DVR is turned on for a coordinate named, say, `r` by the line

```
r     sin     24     1.0     24.0     short
```

in the PRIMITIVE-BASIS-SECTION. Again the first number denotes the number N of grid points. The meaning of the next two numbers depends on the last (optional) keyword, which can be `short` (the default) or `long`. When `short` is selected, the two numbers are the first and last grid point, x_1 and x_N. If the keyword `long` is present, the two numbers specify the box boundaries, x_0 and x_{N+1}. The line

```
r     sin     24     0.0     25.0     long
```

is hence equivalent to the previous one. The grid spacing is $\Delta x = (x_N - x_1)/(N-1)$ in the case of `short` and $\Delta x = (x_{N+1} - x_0)/(N+1)$ in the case of `long`.

4.5 Exponential DVR and fast Fourier transform

The exponential DVR uses plane waves as basis functions. It is therefore often used for dissociative degrees of freedom. Moreover, exponential DVR and FFT are the only primitive-basis representations within the MCTDH program that satisfy periodic boundary conditions. These occur for instance for angular motion or motion parallel to a corrugated surface.

As our implementation of the exponential DVR requires an odd number of basis functions, we set $N = 2n + 1$. The basis functions are then written as

$$\chi_j(x) = L^{-1/2} \exp\left(2i\pi j(x - x_0)/L\right) \, , \tag{4.6}$$

with $-n \leqslant j \leqslant n$ and $L = x_N - x_0$. The wavefunctions to be represented satisfy periodic boundary conditions, $\psi(x_0) = \psi(x_N)$.

The fast Fourier transform (FFT) method may be considered as an exponential DVR where, however, the derivative matrices are not built but the action of them on the wavefunction is evaluated by two FFTs. The MCTDH program uses a Temperton FFT which allows one to use grids the length of which can be factorised into powers of 2, 3, and 5, i.e. $N = 2^j 3^k 5^l$, where j, k, and l are non-negative integers. For optimal performance, one should work with a grid length of $N = 2^j 3^k$.

Although the exponential DVR and the fast Fourier transform (FFT) are formally equivalent, this does not hold for their efficiency. It is our experience that the exponential DVR performs faster than FFT for small grids ($N \lesssim 16$), while FFT is faster for large grids ($N \gtrsim 64$). Between these limits both representations are similarly fast.

FFT and exponential DVR have an identical set of input parameters. Supposed there are two degrees of freedom labelled `X` and `Y`, an exponential DVR and an FFT representation are employed for these coordinates by a PRIMITIVE-BASIS-SECTION reading

```
PRIMITIVE-BASIS-SECTION
    X    exp    25    2.2    5.8    linear
    Y    fft    48    1.5    4.3
end-primitive-basis-section
```

As usual, the first number defines the number N of grid points. Remember that N must be odd for an exponential DVR and factorise into powers of 2, 3, and 5 (better 2 and 3 only) for FFT. If the last keyword is missing or `linear`, then the remaining two numbers, which will be called x_i and x_f in the following, are interpreted as the grid points $x_0 = x_i$ and $x_{N-1} = x_f$. The grid spacing is $\Delta x = (x_f - x_i)/(N-1)$. Due to the periodic boundary conditions the first grid point, x_0, and the point x_N, which is the one following the last point on the grid, are to be identified.

Instead of `linear`, the keywords `periodic` or `s-periodic` may be specified, which changes the interpretation of x_i and x_f. In both cases, x_i and x_f are considered identical due to the periodic boundary conditions. The grid spacing is now $\Delta x = (x_f - x_i)/N$. For `periodic` the parameter x_f is taken as Nth grid point, $x_N = x_f$. For `s-periodic` the grid points are additionally shifted by $\Delta x/2$, i.e. $x_0 = x_i + \Delta x/2$ and $x_N = x_f + \Delta x/2$. As an example, the lines

```
    X    fft    32    0.00    3.0434    linear
```

and

```
    X    fft    32    0.00    3.1416    periodic
```

are equivalent.

The keywords `periodic` and `s-periodic` are particularly useful to describe angular degrees of freedom. For angular modes ranging from 0 to some integer fraction of 2π, one may alternatively use the keywords `2pi` or `s-2pi`. The example above is thus equivalent to

```
    X    fft    32    2pi/2
```

For a more detailed discussion see the HTML documentation.

4.6 *Advanced topic*: Spherical harmonics FBR

The spherical harmonics FBR is the appropriate choice when there is rotational motion which must be described by two angles, θ and ϕ. The spherical harmonic functions

$$Y_{jm}(\theta,\phi) = \sqrt{\frac{2\,j+1}{4\pi}\,\frac{(j-m)!}{(j+m)!}}\,P_j^m(\cos\theta)\,e^{im\phi} \tag{4.7}$$

serve as basis functions, where P_j^m denotes the Legendre polynomial (4.4). The matrix elements of the angular momentum operators $\hat{j}^2$, $\hat{j}_+$, $\hat{j}_-$, and $\hat{j}_z$ are then given by simple formulas.

Examples for a PRIMITIVE-BASIS-SECTION defining a spherical harmonics FBR for the set of coordinates `alpha` and `beta` are

```
PRIMITIVE-BASIS-SECTION
    alpha    sphFBR    9  nosym
    beta     phiFBR
end-primitive-basis-section
```

and

```
PRIMITIVE-BASIS-SECTION
    alpha    sphFBR    8  sym
    beta     phiFBR    4  2
end-primitive-basis-section
```

The keyword `sphFBR` selects the spherical harmonics FBR and the label `phiFBR` indicates the second coordinate on which the FBR is based. The number $j_{\max}$ after the keyword `sphFBR` or defines the maximum value for j. The j values are $j = 0, 1, \ldots, j_{\max}$ for `nosym` and $j = 0, 2, \ldots, j_{\max}$ or $j = 1, 3, \ldots, j_{\max}$ for `sym`, depending on the parity of $j_{\max}$. Without the optional data $m_{\max}$, which follows the keyword `phiFBR`, m takes on the values $m = -j, -j+1, \ldots, -1, 0, 1, \ldots, j-1, j$. With $m_{\max}$ (and possibly also Δm) given, m takes the values $m = -\min(m_{\max}, j), -\min(m_{\max}, j) + \Delta m, \ldots, \min(m_{\max}, j) - \Delta m, \min(m_{\max}, j)$. In the second example we thus have j-values of $j = 0, 2, 4, 6, 8$. The corresponding values for m are $m = 0$ for $j = 0$, $m = -2, 0, 2$ for $j = 2$, and $m = -4, -2, 0, 2, 4$ for $j = 4, 6, 8$, giving an overall number of 19 basis functions.

The order in which the primitive basis sets are declared is, in general, arbitrary. However, a sphFBR line **must** be followed directly by a phiFBR line. Moreover, the corresponding degrees of freedom (`alpha` and `beta` in the example above) must be declared as combined in the SPF-BASIS-SECTION.

4.7 *Advanced topic*: Restricted Legendre DVR

The restricted Legendre DVR, `Leg/R`, is very similar to the (ordinary) Legendre DVR, but can make use of the fact that the angular motion may be restricted to an interval smaller than $(0, \pi)$. In such a case one may drop the unused grid-points, assuming that the wavefunction vanishes there. The FBR/DVR transformation matrix is now rectangular rather than square. The propagation, however, is performed exclusively in the (smaller) set of DVR grid-points and some speed-up is obtained. For example, one may replace the `theta` line in the `PRIMITIVE-BASIS-SECTION` of the nocl1.inp input file (see Example 4.1 and the Tutorial) to

```
  theta  Leg/r  60  0   all    1.4   2.7
```

The two last numbers define the range (in radians) to be covered by the grid-points. In the log file one finds

```
****** Primitive Basis *********

 mode      kappa     DVR     N         xi          xf          dx         p-max

 rd            1     sin    36       3.800       5.600    0.051429      59.436
 rv            2     HO     24       1.620       2.652    0.044849      70.171
 theta         3   Leg/R 27/ 60      2.739       1.389    0.051924   m= 0, sym= 0
```

The last line tells us, that there are 60 FBR functions but only 27 DVR grid-points. For technical reasons, the locations of the first and last grid-points differs slightly from the inputted numbers. Comparing the timing files, one finds that the restricted Legendre DVR reduces the total CPU time by a factor of 1.4 and reduces the time spend for propagating the single-particle functions of the theta degree of freedom by more than a factor of 4.

The analyse programs **plgpop** and **showd1d84** are useful for determining an appropriate theta-range. Try `plgpop -z 1.e-10 2 3` and `showd1d84 -a -y 0.000001 f3`.

When using the restricted Legendre DVR it should be the last entry in the `PRIMITIVE-BASIS-SECTION`. Due to a bug it otherwise sometimes performs incorrectly.

4.8 *Advanced topic*: Extended Legendre DVR (KLeg) and Two-Dimensional Legendre DVR (PLeg)

Similarly to the spherical harmonics FBR, the extended Legendre DVR employs the spherical harmonics as basis set. It thus defines a two-dimensional representation. As the name indicates, there is a FBR/DVR transformation from the angular momentum indices ℓ to grid points θ_i. There is, however, no such transformation for the $\{m, \phi\}$ pair. The extended Legendre DVR will typically be used to describe angular motion of a molecule with total angular momentum $J > 0$. The projection of the angular momentum of the angular motion under discussion onto the body fixed frame is usually called K or Ω (rather than m). The angle ϕ, which is the coordinate canonical conjugate to K, is an Euler angle and does not appear in the potential. There is thus no point to perform a $K \leftrightarrow \phi$ transformation. The 2D single-particle functions, $\varphi(\theta, K)$, are hence given in a mixed DVR–FBR representation.

An example for a PRIMITIVE-BASIS-SECTION defining an extended Legendre DVR for the set of coordinates `theta` and `K` is given by

```
PRIMITIVE-BASIS-SECTION
    theta  KLeg      31     even
    K       K        -4        4
end-primitive-basis-section
```

The keyword `KLeg` selects the extended Legendre DVR and the label `K` indicates the second coordinate on which the KLeg representation is based. Similarly to the spherical harmonics FBR, the `KLeg` line **must** be followed directly by a `K` line, and the corresponding degrees of freedom (here `theta` and `K`) must be declared as combined in the SPF-BASIS-SECTION. The first number after `KLeg` specifies the number N of θ-grid points, the following keyword controls which symmetry is to be used. For `all`, all values $\ell = K_1, K_1+1, \ldots, K_1+N-1$ are employed, for `odd/even`, only odd/even values of ℓ are taken, i.e. $\ell = K_1, K_1 + 2, \ldots, K_1 + 2N - 2$ or $\ell = K_1 + 1, K_1 + 3, \ldots, K_1 + 2N - 1$. Here, K_1 denotes the minimum of $|K|$, i. e. $K_1 = 0$ if $K_{\min} \cdot K_{\max} \leq 0$ and $K_1 = \min(|K_{\min}|, |K_{\max}|)$ else. The numbers following `K` are the minimal, $K_{\min}$, and maximal, $K_{\max}$, values for K. The K-grid thus consists of 9 points for the example above.

The two-dimensional Legendre DVR, `PLeg`, is similar to `KLeg`, except that a Fourier-transformation from K to ϕ is performed. An example for a PRIMITIVE-BASIS-SECTION defining a PLeg two-dimensional Legendre DVR for the set of coordinates `theta` and `phi` is given by

```
PRIMITIVE-BASIS-SECTION
    theta  PLeg      31      even
    phi     exp      11      2pi
end-primitive-basis-section
```

The keyword `PLeg` selects the two-dimensional Legendre DVR and the label `phi` indicates the second coordinate on which the PLeg representation is based. Similarly to KLeg and to the spherical harmonics FBR, the `PLeg` line **must** be followed directly by a `exp` line, and the corresponding degrees of freedom (here `theta` and `phi`) must be declared as combined in the SPF-BASIS-SECTION. The number of grid-points of the exponential DVR must be odd. The 11 points chosen here allow to represent K in the interval $-5 \leq K \leq 5$.

4.9 *Advanced topic*: Three-Dimensional rotational DVR (Wigner)

Wigner-DVR uses the L^2-normalized Wigner-D functions, $\overline{D}^j_{m,k}(\alpha, \beta, \gamma)$, as a basis set. These are defined as:

$$\overline{D}^j_{m,k}(\alpha, \beta, \gamma) = \sqrt{\frac{2j+1}{8\pi^2}} D^j_{m,k}(\alpha, \beta, \gamma) \tag{4.8}$$

$$D^j_{m,k}(\alpha, \beta, \gamma) = e^{-im\alpha} d^j_{m,k}(\beta) e^{-ik\gamma} \tag{4.9}$$

where α and γ are the Euler angles representing rotation around the space-fixed (SF) and body-fixed (BF) z-axes, respectively, and β is the Euler angle between the SF and BF z-axes (where the z-y-z right-handed axis convention is used). The Wigner-(small)-d function, $d^j_{m,k}(\beta)$, is defined as:

$$d^j_{m,k}(\beta) = \left\langle j, m \left| e^{-i\beta \hat{J}_Y} \right| j, k \right\rangle \tag{4.10}$$

Wigner-DVR defines a three-dimensional representation which can be used to model rotation of polyatomic molecules. An FBR/DVR transform is used to convert between the angular momentum index j and the grid points β_i. The $\{m, \alpha_i\}$ and $\{k, \gamma_i\}$ momentum-coordinate conjugate pairs can be used in either momentum or coordinate representation in MCTDH; in the latter case a discrete Fourier transform switches between grid points and momentum indices.

A PRIMITIVE-BASIS-SECTION input block defining Wigner-DVR uses three input lines, one for each Euler angle. For example:

```
PRIMITIVE-BASIS-SECTION
---------------------------------------------------
# Mode    DVR      N
---------------------------------------------------
  beta    wigner  20   all
  gamma   k       -7   7
  alpha   exp     15   2pi
---------------------------------------------------
END-PRIMITIVE-BASIS-SECTION
```

The keyword `wigner` is used to select the L^2-normalized Wigner-(small)-d functions as the basis functions for the β- angle. The number following the `wigner` keyword denotes the number of j-values to be used, and the `all` keyword denotes that both odd and even j-values are to be included. For the γ and α angles, the coordinate representation is declared by an `exp` line; alternatively, a `K` line can be used to declare the momentum representation. In the example above, the `alpha` DOF is used in coordinate representation, while the `gamma` DOF is used in momentum representation. The input line for the β angle must appear first and be immediately followed by a line for the γ angle, which must then be followed by a line for the α angle; that is, the order of the degrees of freedom must be given as $|J, K, M\rangle$. All three DOF must be declared as combined in the SPF-BASIS-SECTION.

Defining the single-particle basis

Chapter 5

Defining the single-particle basis

In an MCTDH calculation (propagation or relaxation) the single-particle basis has to be defined. This is done in the SPF-BASIS-SECTION of the input file. The SPF-BASIS-SECTION also enables one to treat degrees of freedom as a combined mode, or to select only a subset of the system degrees of freedom.

5.1 Specifying the number of single-particle functions

In the SPF-BASIS-SECTION the number $n_\varkappa$ of single-particle functions to be used for each degree of freedom $\varkappa$ of the system are listed. Supposed that there are two degrees of freedom, the SPF-BASIS-SECTION

```
SPF-BASIS-SECTION
X = 3
Y = 3
end-sbasis-section
```

assigns three single-particle functions to each mode. Here `X` and `Y` are the mode labels which must coincide with those in the PRIMITIVE-BASIS-SECTION.

If only one single-particle function is used for each degree of freedom, a calculation actually employs the time-dependent Hartree (TDH) method rather than the MCTDH scheme. Another special case is obtained for equal numbers of single-particle and primitive basis functions in each degree of freedom $\varkappa$, i.e. $n_\varkappa = N_\varkappa$, with $N_\varkappa$ being the number of primitive basis functions. An MCTDH calculation is then equivalent to a numerically exact one, but note that numerically exact calculations can be performed much more efficiently by using the `exact` keyword (see Sec. 3.6).

Typical numbers of single-particle functions range from $n_\varkappa = N_\varkappa/20$ to $n_\varkappa = N_\varkappa/3$. Note that the numbers of single-particle functions should obey

$$n_\varkappa^2 \leqslant \prod_{\varkappa'=1}^{f} n_{\varkappa'} \,, \tag{5.1}$$

since otherwise there will be redundant configurations in the MCTDH wavefunction. Hence, one must choose $n_1 = n_2$ if there are two degrees of freedom.

5.2 *Advanced topic*: Selecting degrees of freedom from a large system

For some systems it may be useful to treat only a subset of the specified coordinates. The program allows one to select coordinates simply by listing only those required in the SPF-BASIS-SECTION. For instance, if five modes `V`, `W`, `X`, `Y`, and `Z` are specified in the PRIMITIVE-BASIS-SECTION

```
PRIMITIVE-BASIS-SECTION
    V    HO    16    0.0d0    1.0d0    1.0d0
    W    HO    22    0.0d0    1.0d0    1.0d0
    X    HO    32    0.0d0    1.0d0    1.0d0
    Y    HO    21    0.0d0    1.0d0    1.0d0
    Z    HO    12    0.0d0    1.0d0    1.0d0
end-primitive-basis-section
```

and you wish to include only `W`, `X`, and `Z` in your calculations, this can be accomplished by the SPF-BASIS-SECTION

```
SPF-BASIS-SECTION
    W = 7
    X = 8
    Z = 6
end-spf-basis-section
```

(or by commenting out the corresponding lines for `V` and `Y`).

The approximation being made here is that the coupling between the included and excluded modes is negligible. As a result, the Hamiltonian is built simply ignoring all terms that include contributions from the excluded degrees of freedom.

5.3 Combining modes to produce multi-dimensional single-particle functions

For large systems, or when certain degrees of freedom are strongly coupled, it may be advantageous to combine degrees of freedom together and use multi-mode single-particle functions (see Sec. 4.5 in Ref. [1] for further details).

A combination scheme can be easily specified by grouping together the degrees of freedom to be combined in the SPF-BASIS-SECTION. As an example we consider a system consisting of eight degrees of freedom named `r1`,...,`r8`. The SPF-BASIS-SECTION

```
SPF-BASIS-SECTION
    r1, r4     = 15
    r2, r3, r7 = 10
    r5, r6, r8 = 12
end-spf-basis-section
```

then combines, for instance, the modes `r1` and `r4` together, and treats them as a single coordinate in the calculation. The single-particle functions for this coordinate are then two-dimensional functions in the system coordinates. The number of two-dimensional single-particle functions is 15 for the combined `r1`–`r4`–mode.

In the example given, eight modes have been combined together to produce three modes for the MCTDH calculation. As the length of the expansion coefficient vector grows exponentially with the number of modes included in a calculation this drastically reduces the computational resources required.

It is important to choose a ”good” combination scheme. One should combine those DOF, which are most strongly coupled with each other. Then, these correlations are taken care of at the SPF-level, and the number of SPFs, necessary for convergence, is decreased. Sometimes physical intuition and knowledge of the system helps to identify the strongly coupled DOFs, but most of the time, this information is missing. Then one takes a more practical approach. Firstly, one combines those DOFs which have similar frequencies, because modes with very different frequencies are less likely to couple strongly. Secondly, one should set up a combination scheme such, that the combined grids have similar size.

One must neither over- nor under-combine. The A-vector length should be larger than the number of data points needed to represent the SPFs. Otherwise it is likely that one over-combines. It is always useful to inspect the timing file. The time needed to propagate the SPFs should take between 2% and 25% of the total effort (CMF scheme assumed).

Setting up the Hamiltonian

Chapter 6

Setting up the Hamiltonian

For a quantum dynamical calculation a Hamiltonian operator has to be defined. This is typically done using the operator file, which is a text file that is read and interpreted by the MCTDH program. The MCTDH program is capable to parse a variety of mathematical expressions. This often allows the implementation of a new Hamiltonian without programming any routines.

6.1 The operator file

The operator file contains all the information the program needs to set up the Hamiltonian. It must have the extension .op, e.g. hamiltonian.op. Similarly to the input file, the operator file is divided into sections. In the OP_DEFINE-SECTION the Hamiltonian is characterised. Numerical constants can be specified in the PARAMETER-SECTION. The definition of the Hamiltonian is contained in the HAMILTONIAN-SECTION. Finally, any labels that are used to describe the Hamilton operator are compiled in the LABELS-SECTION. The operator file for the propagation of NOCl on the S_1 surface is given as Example 6.1.

In order to select a particular operator file, an OPERATOR-SECTION is required in the input file. The keywords `opname` and `oppath` then point to this file. For instance, with the OPERATOR-SECTION

```
OPERATOR-SECTION
oppath = /usr/people/quantics/operators
opname = nocl1
end-operator-section
```

the MCTDH or Potfit program would look in the directory /usr/people/quantics/operators for the operator file nocl1.op. If the `oppath` item is not given, the program first looks in the startup directory (i. e. the directory where the input file is located), then in the directory specified by the default operator path. (The default operator path is displayed when typing **quantics -max**). The operator files that are available in the MCTDH package are tabulated in the HTML documentation.

If you desire to create a new operator file, the first step is to write the OP_DEFINE-SECTION. In this section the Hamiltonian is given a title, between the keywords `title` and `end-title`. The title will be printed e.g. in the log file. In the NOCl example 6.1 the operator is labeled “NOCl S1 surface”, by employing the OP_DEFINE-SECTION

```
OP_DEFINE-SECTION
```

```
OP_DEFINE-SECTION
title
   NOCl S1 surface
end-title
end-op_define-section

PARAMETER-SECTION
mass_rd = 16.1538 , AMU
mass_rv =  7.4667, AMU
end-parameter-section

HAMILTONIAN-SECTION
------------------------------------------
modes         |  rd    |  rv    |  theta
------------------------------------------
-0.5/mass_rd  |  dq^2  |  1     |   1
-0.5/mass_rv  |  1     |  dq^2  |   1
 0.5/mass_rd  |  q^-2  |  1     |  j^2
 0.5/mass_rv  |  1     |  q^-2  |  j^2
 1.0          |  V
------------------------------------------
end-hamiltonian-section

LABELS-SECTION
V = srffile {nocl1um, default}
end-labels-section

end-operator
```

Example 6.1: An operator file for the NOCl S_1 state.

```
title
NOCl S1 surface
end-title
end-op_define-section
```

There is no restriction on the format or number of lines used. The next step for creating a new operator file is described in the following section.

6.2 Defining numerical constants

A Hamiltonian typically contains some numerical constants, such as (reduced) masses, frequencies, or coupling strengths. In the PARAMETER-SECTION of the operator file, labels can be associated with these constants, which can then be used in the definition of the Hamiltonian. This makes the operator file not only easier to read but also very simple to change, as is depicted in Sec. 6.11.

For example, a PARAMETER-SECTION reading

```
PARAMETER-SECTION
mh =  1837.15   # hydrogen mass
mc = 21874.7    # carbon mass
end-parameter-section
```

defines the labels mh and mc as the masses of the hydrogen and carbon nuclei (in a.u.). Note that it is possible to specify the unit of a parameter. The just mentioned example can hence also be written

```
PARAMETER-SECTION
mh =  1.0, H-mass
mc = 12.0, AMU
end-parameter-section
```

The keywords H-mass and AMU stand for hydrogen mass and atomic mass unit. See the HTML documentation for available units. Furthermore, elementary algebraic operations can be performed in the PARAMETER-SECTION. Thus, the section

```
PARAMETER-SECTION
a = 1.0
b = 2.0
c = 3.0
d = 2.0*a+b*c+1.0
e = EXP[ d/9.0-a ]
end-parameter-section
```

sets d to 9.0 and e to 1.0. The algebraic expression must not contain spaces or brackets! Only the operators $+ - * / \hat{}$ are allowed. An exponent acts only on the label to which it is attached and $* /$ are evaluated before $+-$. Otherwise the order of operation is strictly from left to right. Exponents must be numbers (not parameters), they may be signed and real (e. g.: alpha^-0.5 is a valid construct). The exponential form (e. g. 2.3d-5) is allowed for numbers only. The decimal exponent must be indicated by a d, not by a D, e or E ! See the HTML documentation (Hamiltonian-Documentation/Parameter-Section) for further details.

The last line of the example above demonstrates the use of functions in parameter arithmetic. Available functions are:
EXP, LOG, LOG10, SIN, COS, TAN, ASIN, ACOS, ATAN, SINH, COSH, TANH, ABS, INT, ATAN2, MIN, MAX.
The definitions of these functions are the usual FORTRAN definitions. Note that the last three functions depend on two arguments. The two arguments, which may be numbers, parameters or numerical expressions, must be separated by a blank or a comma. The keyword for the function must follow directly the equal sign, not even a minus is allowed in between. No operations must follow the closing bracket of the function, except possibly a unit.

Some labels have a special, pre-defined meaning. A complete list of these is given in the HTML documentation. Here we only mention the mass_x label (cf. Example 6.1). The value of this parameter is taken as the mass of the degree of freedom specified by x (as defined in the PRIMITIVE-BASIS-SECTION of the input file). By default the mass is otherwise set to 1.0 au. The mass_x label can then be used to define the kinetic energy, by employing the KE operator. The symbol KE belongs to the list of expressions the MCTDH program can interpret. These will be explained in the following section. Other parameters of special meaning are PI, jtot, and jbf.

Remarks:

- Parameters cannot be re-defined. A second definition is simply ignored (but protocoled in the log file). E. g. if a parameter is defined on the command line, then a following re-definition in the operator-file will be ignored. This, however, makes operations like

```
par=par+1
```
invalid.

- Since version 8.2.2 it is allowed to put spaces around $+$ and $-$. However, such spaces are allowed only in a PARAMETER-SECTION, but not in parameter arithmetic anywhere else (e. g. not within a parameter bracket of an symbolic expression or function).

- One better does not mix parameter arithmetic and units. The statement
```
par = par1*par2+par3,ev
```
is valid. The whole expression (not only par3) is divided by 27.211. However, the statement
```
par = par1,ev/par2,AMU
```
is invalid. A unit can only be appended to the end of an expression and it will modify the value of the full expression.

- A function name must be in upper case letters and the argument(s) of the function must be in square brackets. No arithmetic is possible with the function. Do the arithmetic with the parameter which was defined by the function.

- The parameters and their values are protocoled in the op.log file. One may check if the program has performed the parameter arithmetic in the way expected.

6.3 Using symbolic expressions to define the Hamiltonian

The MCTDH and Potfit programs are able to parse a variety of mathematical symbols, such as powers, exponential and trigonometric functions, and first and second derivatives. A complete list of these built-in symbols can be found in the HTML documentation. A selection of these symbols is compiled in Tab. 6.1. For a complete list, please refer to App. C. The symbolic expressions are used in the HAMILTONIAN-SECTION to define the Hamiltonian.

Within the MCTDH framework the Hamiltonian must in general be given in product form,

$$H = \sum_{r=1}^{s} c_r \, h_r^{(1)} \cdot \ldots \cdot h_r^{(f)} \,, \tag{6.1}$$

where f and s denote the numbers of degrees of freedom and Hamiltonian terms, respectively. This structure is reflected in the HAMILTONIAN-SECTION, which is mainly a table with s lines and $f+1$ columns, containing the coefficients c_r and single-particle operators h_r.

To make things concrete, let us explain the usage of the symbolic expressions with the aid of the following two examples. The first one is a fairly simple one, namely a modified Henon-Heiles Hamiltonian, i.e. two coupled anharmonic oscillators.

The Hamiltonian is

$$H = -\frac{1}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + \frac{x^2+y^2}{2} + \lambda\left(xy^2 - \frac{x^3}{3}\right) + \lambda^2\left(\frac{x^4+y^4}{16} + \frac{x^2y^2}{8}\right) \,. \tag{6.2}$$

The modification with respect to the original Henon-Heiles Hamiltonian is the last (quartic) term. It makes the system bound. The corresponding operator file is included as Example 6.2. As one can see, the Hamiltonian (6.2) is represented by the symbols in the HAMILTONIAN-SECTION, one product term per line. The mode labels have to match with those in the

Table 6.1: A selection of built-in symbolic expressions that can be used to define the Hamiltonian. (For a complete list please refer to the HTML documentation or to Appendix C.) The variables x and θ represent the mode labels associated with the corresponding degrees of freedom.

Symbol	Operator	Description
`1`	1	Unit operator
`q`	x	Multiply by position coordinate x
`q^r`	x^r	Multiply by rth power of x
`sin`	$\sin x$	Sine of coordinate
`cos`	$\cos x$	Cosine of coordinate
`tan`	$\tan x$	Tangent of coordinate
`exp`	e^x	Exponential of coordinate
`dq`	∂_x	First derivative
`dq^n`	∂_x^n	nth derivative[a]
`KE`	$-\frac{1}{2m}\,\partial_x^2$	Kinetic energy term[b]
`j^2`	$-\frac{1}{\sin\theta}\,\partial_\theta \sin\theta\,\partial_\theta$	Angular momentum squared[c]

[a]Derivatives with $n > 2$ are only allowed for an FFT primitive-basis.
[b]The variable m stands for the `mass_`x label defined in the PARAMETER-SECTION.
[c]The given formula is only valid if a Legendre-DVR with magnetic quantum number $m = 0$ is used.

```
OP_DEFINE-SECTION
title
Henon-Heiles PES
end-title
end-op_define-section

PARAMETER-SECTION
mass_X = 1.0  mass_Y = 1.0
lambda = 0.2
end-parameter-section

HAMILTONIAN-SECTION
-----------------------------
  modes        |  X   | Y
-----------------------------
1.0            |  KE  | 1
1.0            |  1   | KE
0.5            |  q^2 | 1
0.5            |  1   | q^2
lambda         |  q   | q^2
-lambda/3      |  q^3 | 1
lambda^2/16    |  q^4 | 1
lambda^2/16    |  1   | q^4
lambda^2/8     |  q^2 | q^2
-----------------------------
end-hamiltonian-section

end-operator
```

Example 6.2: An operator file for a wavepacket propagation using the modified Henon-Heiles Hamiltonian.

PRIMITIVE-BASIS-SECTION of the input file. The coupling parameter λ and the masses used for the `KE` keyword are defined in the PARAMETER-SECTION.

The second example is the kinetic energy of a three-atomic molecule with total angular momentum $J = 0$ described by Jacobian coordinates r_1, r_2, and θ. The kinetic energy reads

$$T = -\frac{1}{2\mu_1}\frac{\partial^2}{\partial r_1^2} - \frac{1}{2\mu_2}\frac{\partial^2}{\partial r_2^2} - \frac{1}{2}\left(\frac{1}{\mu_1 r_1^2} + \frac{1}{\mu_2 r_2^2}\right)\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\sin\theta\,\frac{\partial}{\partial\theta}\,, \tag{6.3}$$

where μ_1 and μ_2 specify the reduced masses associated with r_1 and r_2. The representation of T in the HAMILTONIAN-SECTION is given by

```
HAMILTONIAN-SECTION
-------------------------------------------
modes          |  r1    |  r2    |  theta
-------------------------------------------
 1.0           |  KE    |  1     |  1
 1.0           |  1     |  KE    |  1
 0.5/mass_r1   |  q^-2  |  1     |  j^2
 0.5/mass_r2   |  1     |  q^-2  |  j^2
-------------------------------------------
end-hamiltonian-section
```

Here we have assumed that the keywords `mass_r1` and `mass_r2` have been correspondingly defined in the PARAMETER-SECTION. (See also Example 6.1). Note that all blank lines and all lines which start with `-----` (5 minus) are ignored.

Another example, the operator file for a 4-mode model of pyrazine, is discussed is Sec. 9. This example also demonstrates how to treat non-adiabatic systems.

6.4 Defining labels

In the tableau of the HAMILTONIAN-SECTION there must appear only simple labels with or without exponents and products of those. I. e. `cos*q^2` is a valid entry but `q+q^2` is not. Simply use two lines to perform the sum. Symbolic expressions with parameters must not appear in the tableau, one rather must link them to a simple label. This is done in a LABELS-SECTION. For example

```
LABELS-SECTION
bcw=expcos[1.1,theta0]
cap1=CAP[ 5.0 0.3 3 ]
dcos=cos1[2.0 5.0*t1]^2
V =  natpot {name}
end-labels-section
```

See the Appendix C for a list of symbolic expressions. In the parameter list, $[\cdots]$, there may appear numbers, parameters or simple algebraic expressions of those. The different entries may be separated by a comma or a blank. The use of units is not allowed here. File names, etc. are given in curly brackets. If a path is relative it is interpreted as relative to the location of the input file. The entry `name` simply is a shortcut for the path of the name-directory.

A label may consist of upper or lower case letters (case sensitive!) and numbers. Even special characters like ".", ":", "$", "%", "&", or "?" are allowed. The underscore "_" however has a special meaning. If one writes a label as

label_modelabel

where *modelabel* denotes a modelabel of one of the dof, then the program puts the corresponding operator in the *modelabel* column of the HAMILTONIAN-SECTION, assuming unit operators for all other degrees of freedom. This feature is frequently used for complex absorbing potentials (CAPs). See Section 6.10 for an example. Remember not to make the underscore part of a label, except when using this modelabel feature.

6.5 Implementing user-defined 1D-operators

The examples in the previous section have underlined how easily a Hamiltonian operator can be implemented in many cases into the MCTDH program. Nevertheless, it might happen that a new 1D-operator is required for a particular problem. When the desired 1D-operator is a real function, i.e. a potential, then there are three simple ways to implement it. The first way is to define the function by a set of data points $\{x_i,\ f(x_i)\}$. This data is written in ASCII format to a file called, say, data. When one includes in a LABELS-SECTION a statement like:

```
V2=external1d{data}
```

then the data will be read by MCTDH, quadratically interpolated and assigned to the label `V2`. As usual, the path given as argument to external1d may be absolute or relative. In the latter case it is, as usual, relative to the location of the input file. MCTDH reads the data file (two columns, free format) till it finds an end-of-file. Blank lines and lines beginning with a hash '#' are ignored. The x–values have to be equally spaced. Extrapolation is not possible, hence the first x-point must be smaller than the first grid-point and the last x-point must be larger than the last grid-point. For optimal performance, there should be two data points below the first and above the last grid-point, respectively. The data points should be dense enough such that the interpolation error is negligible.

The second way is similar to the first one, but here the data must be given precisely at the grid points. A file of either ASCII or binary format must be generated such that the first line contains the data for the first grid point, the second line for the second point, etc. Let us call this file again data. The LABELS-SECTION should now contain the statement

```
V2=read1d{data ascii}
```

where `ascii` is to be replaced by `binary` if the file data is in binary format. See also HTML documentation: "Hamiltonian Documentation"/"Labels-Section".

The third way is to use the pre-defined label `my1d` and to edit the subroutine **my1d** which is on source/opfuncs/func1d.F. The label `my1d` is to be used in the same way as e.g. the label `cos`, i.e. with or without parameters. One may make use of up to five parameters.

Grid based 1D operators, i. e. matrix representations of 1D operators with respect to the DVR functions, may also be read in. See the HTML documentation "Hamiltonian Documentation" / "Labels-Section" for details.

6.6 *Advanced topic*: Defining new symbolic expressions

When a new operator with a new label is to be implemented, this operator must be added to the operators in the source/opfuncs directory. Operators are divided into four classes, which are handled differently by the program. The first of these are grid based operators, such as the kinetic energy operator in a DVR basis, or a natural potential expansion operator, which are only defined with respect to a grid. The remaining three classes are all different types of analytic functions: complex functions (e.g. CAPs), multi-dimensional functions (e.g. non-separable potential energy surfaces), and real one-dimensional functions. In this and the following section we will give some examples as to how to implement real one-dimensional functions. The implementation of multi-dimensional functions is the topic of Sec. 6.8.

Important note: Since version 8.3.10 there are four new files: install/user_surfaces, install/user_surfdef, opfuncs/user1d.F and opfuncs/usersrf.F. These were introduced to separate changes done by a user from changes done by the authors. This new procedure will simplify an update of the package. When implementing a new one-dimensional potential function please use now opfuncs/user1d.F. Otherwise the procedure remains unchanged.

There is a default example from MCTDH in the file opfuncs/user1d.F, showing how users can define their own one-dimensional potentials. There are three subroutines in this file: `ufdef1d`, `ufunc1d`, and `my1d`. Suppose we use the label “my1d” in the operator file

```
LABELS-SECTION
     my = my1d[p1,p2,p3,p4,p5]^p0
end-labels-section
```

The subroutine `my1d` will be executed during running MCTDH. Although there is no real potential coded in this subroutine, one can find instructions on how to implement the user-defined potentials. To call the subroutine `my1d`, the program has to identify the label “my1d.” This is done in the subroutine `ufdef1d`, where MCTDH reads the label and its parameters, e.g. `p1`. The arrays `hoprpar` and `hopipar` serve to pass real or integer arguments to the function definition in subroutine `ufunc1d`. Here we only use the real one. The program by default stores an exponent r in `hoprpar(1)` or `hopipar(1)`, depending on its type. If any parameters `[`*p1*`,`*p2*`,...]` given in square brackets are present, this is indicated by the counter `np` being greater than zero, and the parameters are automatically pushed to the array elements `hoppar(1,np)`,`hoppar(2,np)`,.... Since `hoprpar(1)` is reserved for the exponent, there is a shift of arguments between `hoprpar` and `hoppar`. If no parameters have been specified, i.e. `np=0`, default values are taken for the parameters. Besides the issue of parameters, each label is given a different function number `ifunc` and the program will call different subroutine according to the `ifunc` number. For example, we have defined `ifunc` equals to 1 if the label is `my1d` in subroutine `ufdef1d` and in subroutine `ufunc1d`, subroutine `my1d` will be called if `ifunc` equals 1.

As an example, if you wish to add the cotangent function $\cot(a * (x - b))$ to the program, using the label `cot`. You would then have to edit both the subroutines `ufdef1d` and `ufunc1d` in the file opfuncs/user1d.F, and additionally write a subroutine of the cotangent function. First, the new function must be coded:

```
subroutine cot(x,v,a,b)
C cot:   cot(a*(x-b))
C this subroutine is called in subroutine ufunc1d
C the parameters are defined in subroutine ufdef1d

real*8  x, v, a, b, r

      r=a*(x-b)
      v=1.d0/tan(r)

end subroutine
```

Any valid FORTRAN expression can be employed to define a new function. Then you have to define the label "cot" in the subroutine `ufdef1d` and give it a function number `ifunc`. Go to the end of the subroutine `ufdef1d` where the code reads like (near line 60 of file opfuncs/user1d.F)

```
                ifunc=1

C newfunc     ! This is, of course, just an example.
C     elseif(label(1:ilbl) .eq. 'newfunc') then
C        ifunc=2


C------------------------------------------------------------------------
C end of if loop
C------------------------------------------------------------------------
      endif
```

The next free function number is 2, so you should replace the three out-commented lines with

```
C cot:  cot(a*(x-b))
      else if (label(1:ilbl) .eq. 'cot') then
         if (np .gt.0) then
            hoprpar(2)=hoppar(1,np)
            hoprpar(3)=hoppar(2,np)
         else
            hoprpar(2)=1.0d0
            hoprpar(3)=0.0d0
         endif
         ifunc=2
```

Finally, you have to modify the subroutine `ufunc1d`. Find the following comments

```
C newfunc EXAMPLE
C Set here the routine you want to call
C
C     elseif (ifunc .eq. 2) then
C       call newfunc(x,v)
```

and replace those comments by the following lines

```
C cot:  cot(a*(x-b))
      elseif (ifunc .eq. 2) then
         call cot(x,v,hoprpar(2),hoprpar(3))
```

This completes the modifications for implementing the cotangent function. After recompiling the program (type compile quantics), the new label `cot` may be used in the HAMILTONIAN-SECTION. When the label `cot` is used with parameters, it must – as usual – be assigned to a simple label in a LABELS-SECTION (see Sec. 6.4).

6.7 *Advanced topic*: Implementing separable potentials

In favorable cases the potential energy surface may be expressed by the built-in symbols known to the program, as has been discussed in Sec. 6.3. If this is not possible, new symbolic expressions have to be introduced as detailed in the previous section. In some cases, however, it may be more convenient to set up a set of labels specifically for a complicated separable potential, i.e.

$$V(x_1,\ldots,x_f) = \sum_{k=1}^{s} \prod_{i=1}^{f} V_{ki}(x_i)\,, \tag{6.4}$$

where f and s denote the numbers of degrees of freedom and Hamiltonian terms, respectively. Let us further assume that you provide a FORTRAN-routine, say `mypot`, stored in a file source/opfuncs/mypot.f, to evaluate the one-dimensional potentials $\mathrm{v} = V_{ki}(x)$ in dependence of k, i, and x (v and x in a.u.):

```
subroutine mypot (k,i,x,v)
integer k,i
real*8  x,v
...
end
```

The first step then is to add this file to the operator library, by inserting the line

```
$(AR_OPFUNCS)($(PATH_OPFUNCS)/mypot.o) \
```

at the corresponding position into the install/Makefile.

The second step is to establish labels for the one-dimensional potentials. Although the choice of these labels is arbitrary, we strongly recommend the use of some systematic nomenclature, e.g. `mypot1:1`, `mypot2:1`, `mypot1:2`, etc., where the first number denotes k and the second i. For instance, with $s = 3$ and $f = 2$ the potential would then be defined by the lines

```
1.0  |  mypot1:1  |  mypot1:2
1.0  |  mypot2:1  |  mypot2:2
1.0  |  mypot3:1  |  mypot3:2
```

in the HAMILTONIAN-SECTION.

Next, a link between these labels and the potential routine is needed. This is done by adding the lines

```
8 call mypot(ipar(2),ipar(3),x,v)
  return
```

to subroutine `callanld` in the file source/opfuncs/callanld.f. The file number 8 should be the next free number in that subroutine. For this to work properly, you have to store the indices k and i in `ipar(2)` and `ipar(3)` before. (Remember that `ipar(1)` is reserved for an exponent, if present.) This can be accomplished by inserting a new subroutine `defmypot` into your mypot.f file:

```
subroutine defmypot (label,file,k,i)
integer       file,k,i,a,b,c
character*(*) label

if (label(1:5) .eq. 'mypot') then
   a = 5
   b = index(label,':')
   c = index(label,' ')
   read(label(a+1:b-1),*) k
   read(label(b+1:c-1),*) i
```

```
      file = 308
   endif
   return
   end
```

The number 5 must match the number of characters in the name `mypot`, and `file` must be 300 plus the file number introduced above. Finally, add the lines

```
call defmypot(buffer,ifile,hopipar(2),hopipar(3))
if (ifile .ne. 0) go to 99
```

to the subroutine `defanld` in the file source/opfuncs/callanld.f, and recompile.

We close in noting that it is possible — analogous to what was said in Sec. 6.6 — to pass additional parameters to the potential routine `mypot`, by employing the arrays `hopipar`, `hoprpar`, and `hoppar`.

6.8 *Advanced topic*: Implementing non-separable potentials (potential surfaces)

It is also possible to include non-separable potentials into the MCTDH program, i.e. potentials that cannot be written in the product form (6.4). Because the direct evaluation of a non-separable potential makes an MCTDH calculation extremely inefficient, they are typically used in numerically exact calculations (propagation, relaxation, or diagonalisation) or to generate a separable potential fit using the Potfit program. The MCTDH program can however use them as they are, which may be useful for checking purposes. Note that the use of a multi-dimensional potential is not of disadvantage, if it operates on the coordinates of *one* MCTDH particle (combined mode) exclusively. See Section 6.13.

Another application of non-separable potentials is the MCTDH method in combination with the CDVR scheme (see Sec. 8.4.3). If the CDVR method is to be used during the propagation, the keyword `analytic_pes` should be included in the OPERATOR-SECTION. This means that the generated potential operator will not be explicitly calculated on the primitive grid points, but will be stored in the oper file in an "analytic" form which can be evaluated on-the-fly at any point in coordinate space.

For the convenience of the user, there is already a dummy routine source/surfaces/mysrf.f and one merely places the code of the new potential energy surface there. When editing mysrf.f one finds a brief description of how to make the necessary modifications on the file source/opfuncs/usersrf.F. This simplified procedure is convenient for a quick implementation of a new surface. However, it does not allow to pass surface options to the program. If options are needed or if more than one potential energy surface is to be implemented, one has to go the proper, slightly more elaborate way described next.

However, the most easiest way to include a multi-dimensional potential into MCTDH or POTFIT is via the `readsrf` keyword. In the LABELS-SECTION of an MCTDH operator file there would appear the statement

```
Vmd=readsrf{data ascii}
```

where `data` gives the path of the data file and `ascii` might be replaced by `binary`, if the file is in binary format. For POTFIT one would write

```
pes=readsrf{data ascii}
```

in the OPERATOR-SECTION of the POTFIT input file.

The file data must contain the potential energy values at the grid points, just the energies and one energy per line. The order is implicitly defined by the Primitive-Basis-Section, or (in quantics) by a `|i&j&k` (i,j,k=1,2,...) construct of the Hamiltonian-Section. Note: first index runs fastest. There is no check for the consistency of the data.

Important note: Since version 8.3.10 there are four more files, which were introduced to separate changes done by a user from changes done by the authors. This new procedure will simplify an update of the package. Rather than editing the Makefile, please edit now the two files install/user_surfdef and install/user_surfaces. Similarly, rather than editing opfuncs/funcsrf.F, please edit now the file opfuncs/usersrf.F. For implementing new 1D functions, please use the file opfuncs/user1d.F.

The implementation of a non-separable potential follows the same philosophy as that of separable potentials, now with source/opfuncs/usersrf.F being the relevant file to modify. Let us again assume that you provide a FORTRAN-routine, say `newsurf`, stored in a file source/surfaces/newsurf.f, to evaluate the non-separable potential $\mathrm{v} = V(\vec{x})$ in dependence of the coordinate vector `x(i)` $= \vec{x}_i$ (v and $\vec{x}$ in a.u.):

```
      subroutine newsurf (x,v)
      real*8 x(*),v
      ...
      end
```

As a first step one has to provide a default routine, in case the new surface is not linked. I. e. one creates a file DEF_newsurf.f on the directory source/surfaces, and writes to it e. g. (see source/surfaces/ for examples)

```
      subroutine newsurf (x,v)
      real*8 x(*),v
      write(6,'(a,/a)') '############################################',
     +          'newsurf is not linked. Run compile with -i option.'
      write(2,'(a,/a)') '############################################',
     +          'newsurf is not linked. Run compile with -i option.'
      stop
      end
```

Note that the file DEF_newsurf.f may also need to include dummy routines for other subroutines which are on the source/surfaces/newsurf.f file. In particular, if the routine newsurf contains an entry point, the dummy routine must have a corresponding entry point as well. If the program compiles but does not link, after having added a new surface, it is likely that there is a mistake in the dummy routine.

The next step then is to ensure that these two file will be compiled, by inserting the line

```
SURFDF1 = $(AR_SURFDEF)($(PATH_SURFACES)/DEF_newsurf.o)
```

into install/user_surfdef. Finally insert

```
newsurf.o     : $(PATH_SURFACES)/newsurf.f
	$(FC) $(FFLAGS) -c -o $@ $?
```

into install/user_surfaces. Note that the second line starts with a tab and not with a series of blanks. The two files install/user_surfdef and install/user_surfaces will be sourced (i. e. read) by the Makefile. They hence contain the *personal* additions to the Makefile. Similarly, the files opfuncs/user1d.F and opfuncs/usersrf.F contain the *personal* additions to the FORTRAN code.

The subroutine `newsurf` will be called from the MCTDH or Potfit program via the subroutine `uvpoint` (i. e. user V point) which is located on the file opfuncs/usersrf.F.

At the end of the subroutine `uvpoint` a call to the new surface routine must be added. For this in-comment

```
        elseif(ifunc .eq. 2) then
```

(near line 107 of file opfuncs/usersrf.F) and replace the line following this if-statement with

```
        call newsurf(gpoint,v)
```

The array `gpoint` contains the coordinates in the order specified in the HAMILTONIAN-SECTION, via the mode-line and the `|i&j&k` construct. (See Section 6.13).

The surface number `hopilab`, 2 in our example, then has to be defined. (Note, `hopilab` is called `ifunc` in some routines). This is done in subroutine `udefsrf`, which also is stored on source/opfuncs/usersrf.F. Add lines such as

```
      else if (label(1:11) .eq. 'newsurf') then
         write(ilog,'(a)')  'newsurf, <remarks> '
         hopilab = 2
```

to this subroutine. If the surface depends of parameters or options, these options have to be read here. See subroutine `defsrf` on opfuncs/funcsrf.F for examples.

Finally, we want the program to write some information to the log file, when the multi-dimensional potential energy surface is used. To this end one has to edit the subroutine `usersurfinfo` on the file opfuncs/usersrf.F. One should briefly describe the surface and then name the coordinates. The string `mlabel(j)` contains the modelabel of the j-th coordinate of `mysurf`, as assigned in the Hamiltonian-Section. For example, the code added to `usersurfinfo` may read

```
      elseif( hopilab .eq. 2 ) then
         write(ilog,'(a)') 'newsurf. V(x,y,z) = beta-function'
         write(ilog,'(2a)') 'x : ',mlabel(1)
         write(ilog,'(2a)') 'y : ',mlabel(2)
         write(ilog,'(2a)') 'z : ',mlabel(2)
         jj=4   ! This is dimension+1
```

Finally, recompile. Include the new surface by running compile with the option **-i newsurf**. (See HTML documentation *Installation and Compilation/Compiling the Programs*)

Please be reminded again, that only real, multi-dimensional potential functions should be implemented on `usersrf.F`. For one-dimensional real functions please use `user1d.F` (or `funcan1d.F`), for complex functions use `funcan1z.F`, and for grid based operators use `funcgrd.f`.

The new potential surface is selected in the HAMILTONIAN-SECTION using the `V` (or any other not pre-defined) label and the LABELS-SECTION. Assuming that the Hamiltonian is given by $H = -1/2m\left(\partial_x^2 + \partial_y^2 + \partial_z^2\right) + V(x, y, z)$, the HAMILTONIAN-SECTION reads

```
HAMILTONIAN-SECTION
-------------------------------
modes      |  x   |  y   |  z
-------------------------------
```

```
 1.0       |  KE  |  1   |  1
 1.0       |  1   |  KE  |  1
 1.0       |  1   |  1   |  KE
 1.0       |  V
-------------------------------
end-hamiltonian-section
```

The label `V` is defined in the LABELS-SECTION

```
LABELS-SECTION
V = newsurf
end-labels-section
```

in the operator file.

If the order of the arguments of V differ from the order defined in the mode-line then the order has to be explicitly specified. E. g. turning to the above example but cyclic interchanging the modes the HAMILTONIAN-SECTION reads

```
HAMILTONIAN-SECTION
-------------------------------
modes      |  z   |  x   |  y
-------------------------------
 1.0       |  KE  |  1   |  1
 1.0       |  1   |  KE  |  1
 1.0       |  1   |  1   |  KE
 1.0       |2&3&1  V
-------------------------------
end-hamiltonian-section
```

because the first argument of V is the second mode etc. See the HTML documentation and Section 6.13 for further details.

6.9 Incorporating natural potentials

With the aid of the Potfit program potential surfaces can be fitted to the product form 6.4. These fits are known as natural potential fits. Natural potentials are the method of choice to employ non-separable potential surfaces in an MCTDH calculation. How such a fit is generated will be discussed in Sec. 12.1. We only mention that the Potfit program requires the (non-separable) potential to be implemented in the operator library in exactly the same way as described in the previous section.

After having constructed a natural potential fit, it may be used in the MCTDH program by correspondingly defining the label `V` in the above example:

```
LABELS-SECTION
V = natpot{directory}
end-labels-section
```

Here *directory* denotes the path to the directory containing the natpot file, which is created by the Potfit program. Replacing the path by the keyword `name`, i. e. `natpot{name}`, indicates that the natpot file is in the name-directory, i.e. the directory the output is directed to.

Note that MCTDH uses the modelables to let the potential operate on the degrees of freedom (or on combined modes, i. e. MCTDH particles) in the correct order. It is hence recommended to use an unnumbered Hamiltonian line (e. g. `| V`) rather than a numbered one (e. g. `|1&2&3 V`) when V refers to a natural potential. One may put the symbol `| V` in any column, and it may be convenient to place it in the first column. If a numbered Hamiltonian line is used, however, the numbers must be consistent with the modelabels. Otherwise the program will stop. If one gives the keyword `ignore` as argument to `natpot{···}` in the Labels-Section, then the modelabels are ignored and the assignment of the modes or DOFs is done exclusively by using the numbers. As a final remark we note that the modelabels of a natpot may be altered by running **chnpot84**.

There are some restrictions when using natural potentials (natpots). natpots cannot be multiplied with another operator (except a constant). Hence a construct like `| cos*V` is not allowed. However, if the natpot does not operate on all DOFs or particles, it may be combined with operators acting on the remaining DOFs or particles. E. g. if the natpot `V` does not operate on the first DOF, the following Hamiltonian line is allowed

```
const | dq^2 | V
```

This, however, does not hold if the other operator is also a natpot. I. e.

```
const | V1 | V2
```

is not a valid Hamiltonian line if `V1` and `V2` are both natpots. One cannot multiply natpots with each other.

6.10 Using complex absorbing potentials (CAPs)

Complex absorbing potentials (CAPs) can be employed to reduce the lengths of the primitive grids. CAPs are also useful for computing S-matrix elements in scattering processes. Please refer to Chaps. 4.7 and 8.6 of the review [1] for details.

The CAPs $-iW$ that can be employed in the MCTDH program are one-dimensional and monomial, i.e. of the form

$$-iW(x) = -i\eta\,|x - x_c|^b\,\Theta\big(\pm\,(x - x_c)\big)\,. \tag{6.5}$$

The parameters x_c, η, and b denote the starting point, strength, and order of the CAP, respectively. Θ specifies Heaviside's step function. When the positive sign is used, the CAP lies to the right of x_c, otherwise it is located to the left of x_c.

Let us assume that your system under investigation has three degrees of freedom, labeled `x`, `y`, and `z`. To turn on CAPs for, say, the first two modes, add the lines

```
 1.0  |  cap1  |  1     |  1
 1.0  |  1     |  cap2  |  1
```

to the HAMILTONIAN-SECTION. The labels `cap1` and `cap2` (or any other labels you have chosen) are then defined in the LABELS-SECTION:

```
LABELS-SECTION
cap1 = CAP [ 5.0  0.375  3  +1]
cap2 = CAP [ 1.0  0.240  2  -1]
end-labels-section
```

The parameters in square brackets are from left to right the starting point x_c, strength η (both in a.u.), and order b of the CAP. The last number specifies whether the CAP lies to the right ($+1$, which is also the default), or left (-1) of x_c. Note, the input $[\cdots]$ may consist of numbers, parameters or algebraic expressions containing numbers and parameters. Hence

```
cap1 = CAP [ 3.0+x0  1.0d-3*strength  3  +1 ]
```

is a valid statement, provided the parameters `x0` and `strength` are defined.

The described way of including CAPs has the disadvantage that the CAPs are hard-wired in the operator file. The MCTDH program therefore offers the opportunity to switch on CAPs from the input file, without any change of the HAMILTONIAN- or LABELS-SECTION in the operator file. To this end include the lines

```
alter-labels
cap_x = CAP [ 5.0  0.375  3  +1]
cap_y = CAP [ 1.0  0.240  2  -1]
end-alter-labels
```

in the OPERATOR-SECTION of the input file. The special keyword `cap1_`*x*, where *x* stands for one of the mode labels, tells the program to add a CAP to the corresponding degree of freedom. See Section 6.4 for more details on the special meaning of the underscore (_) within a label.

The optimal values of the CAP parameters, i. e. the Cap length, CAP strength, and CAP order, need to be determined. It is our experience, that the optimal CAP order is 2 or 3 (the larger the energy range, E_{max}/E_{min}, the larger the optimal CAP order). The CAP length should be as small as possible in order not to waist grid points. On the other hand, a short CAP requires a large CAP strength which in turn, produces unwanted CAP reflexions. The program **reflex84** computes an estimate of the CAP transmission and CAP reflexion. This estimate, derived in Ref. [19], is very accurate for a free particle. To determine the optimal CAP parameters, one needs to know the lowest and highest kinetic energy component for the CAP degree of freedom with which the particle enters CAP. These energies are sometimes difficult to estimate. If one has used FFT (or exponential DVR) for the CAP degree of freedom, the command **showd1d84 -a -pop2 -y 0.01 fx** (*x* denotes the number of the degree of freedom of the CAP) displays the momentum distribution from which the desired energies can be calculated.

The program **reflex84** is most conveniently called through the shell script **plcap**. The parameters necessary for the calculation are given as options and arguments. The program prompts for missing input. To give an example, let us turn to NOCl. Type

```
plcap -n 3 -m 16 -l 0.6 -e 0.3 0.1 2.0 ev
```

This computes the reflection– and transmission–probability for a CAP order of 3, a (reduced) mass of 16 atomic mass units, a CAP length of 0.6 a.u., a CAP strength of 0.3 a.u. and for the energy interval 0.1 – 2.0 eV. A GNUPLOT window pops up which displays the reflection– and the absorption–probabilities and their sum. **plcap** then prompts you for new options. Type `-h` to see the list of options, or type `-z 1.e-7` to arrive at a more convenient scale. If one inputs `-e ?`, the program will compute the optimal CAP strength for the given energy interval.

The distribution of kinetic energies of dissociating NOCl lies between 0.2 eV and 1.6 eV, thus the reflection– and absorption–probabilities are below 10^{-4}, which is a fairly good value. The precise values of the CAP parameters are usually not very critical, except when very low kinetic energies are present. Very low kinetic energies may appear when an internal excitation takes almost all of the available energy. The low kinetic energy contributions are more strongly reflected from the CAP and these reflections lead to artificial oscillatory structures in e. g. the reaction probability. Fig. 3 of Ref. [9] shows such an small artificial structure near 1.25 eV, i. e. close to the $v = 2$ threshold. As discussed in Ref. [9] this small artificial structure disappears when using a longer and weaker CAP.

```
# Parameter file for the Henon-Heiles system
lambda = 0.4
end-parameter-file
```

Example 6.3: A parameter file for the Henon-Heiles Hamiltonian.

6.11 *Advanced topic*: Altering a Hamiltonian from input file or command line

The concrete form of the Hamiltonian, i.e. the values of parameters defined in the PARAMETER-SECTION and the meaning of labels specified in the LABELS-SECTION of the operator file, can be overridden in different ways. Let us first consider the change of parameters.

One possibility is to write the parameters to be changed into a file. Example 6.3 shows an example for such a parameter file, again for the Henon-Heiles system. To read this file and use its settings insert the keyword

```
parfile = mypara/hh.par
```

into the OPERATOR-SECTION of your input file. Here it was assumed that the parameter file is named hh.par and resides in the directory mypara (relative to the path of the input file).

A second way is to include the `alter-parameter` and `end-alter-parameter` keywords in the OPERATOR-SECTION of the input file. All parameter definitions in-between replace the corresponding parameters in the PARAMETER-SECTION of the operator file. For instance, the lines

```
alter-parameter
lambda = 0.4
end-alter-parameter
```

in the OPERATOR-SECTION of the input file set the coupling parameter λ of the Henon-Heiles potential, Example 6.2, to 0.4 a.u. The format of the parameters is the same as in the PARAMETER-SECTION of the operator file.

The third method makes use of command line parameters. Starting a calculation employing the **-p** option, e.g.

```
quantics -D new -p lambda 0.4 hh
```

would run a new calculation with the input file hh.inp, but with the coupling parameter λ twice as strong as defined in the operator file. The **-D** option means that the results are written this time to the name-directory *new*. There may be more than one parameter definition and the parameters may carry a unit. Thus

```
quantics -p lambda 10.9,eV -p mass_Y 1.5  hh
```

is also a valid command.

The order of precedence of parameters defined from different sources is command line, input file, parameter file, operator file. Thus parameters in the operator file are the default values, which can be altered in a run in a variety of ways.

The labels in the LABELS-SECTION of the operator file may also be modified without altering the operator file. This is done using the `alter-labels` keyword. One example for this was presented before in Sec. 6.10, where complex absorbing potentials were added to the Hamiltonian. Another typical example is to switch between different implementations of a potential. Supposed the potential to be used is represented by the label `V` in the line

```
# Uwe Manthe fit NOCl S1 surface

PARAMETER-SECTION
theta0 = 127.4, deg

c000 = 0.0384816  c001 = 0.0247875  c002 = 0.0270933  c003 = 0.00126791
  c004 = 0.00541285  c005 = 0.0313629  c006 = 0.0172449
...
end-parameter-section

HAMILTONIAN-SECTION
modes          |  rd          |  rv      | theta

1.0            |  1           |  v:NO    |   1
c000           | bcrd^0*1qd   | bcrv^0   | bcw^0
c001           | bcrd^0*1qd   | bcrv^0   | bcw^1
c002           | bcrd^0*1qd   | bcrv^0   | bcw^2
c003           | bcrd^0*1qd   | bcrv^0   | bcw^3
c004           | bcrd^0*1qd   | bcrv^0   | bcw^4
c005           | bcrd^0*1qd   | bcrv^0   | bcw^5
c006           | bcrd^0*1qd   | bcrv^0   | bcw^6
c010           | bcrd^1*1qd   | bcrv^0   | bcw^0
c011           | bcrd^1*1qd   | bcrv^0   | bcw^1
.....
end-hamiltonian-section

LABELS-SECTION
bcrd = exp1[-1.5,4.315]
1qd = exp[-1.5,4.315]
bcrv = q[2.136]
bcw=expcos[-1.1,theta0]
end-labels-section

end-operator
```

Example 6.4: A surface file containing the MANTHE analytic fit to the NOCl S_1 potential energy surface.

```
 1.0  |  V
```

of the HAMILTONIAN-SECTION and defined as

```
LABELS-SECTION
V = mysurf
end-labels-section
```

in the LABELS-SECTION of the operator file. Then you may add

```
alter-label
V = natpot{name}
end-label-parameter
```

to the OPERATOR-SECTION of your input file. The program then uses the natural potential fit stored in the name-directory rather than the potential surface labeled `mysurf`.

It is also possible to move some part of the Hamiltonian section, e.g. the potential, to a separate file, called surface file. Use of this is made in Example 6.1, where only the kinetic energy part is defined in the HAMILTONIAN-SECTION. The (separable) potential is stored in the surface file nocl1.srf, part of which is displayed in Example 6.4. (See

$MCTDH_DIR/operators/nocl1um.srf for a complete listing and compare it with Eqs. (32,33) of Ref. [3]).

The surface is defined in the Hamiltonian by

```
LABELS-SECTION
V = srffile {nocl1, directory}
end-labels-section
```

Here *directory* denotes the path to the directory containing the nocl1.srf file. Replacing the directory by the keyword `oppath` indicates that the surface file is in the same directory as the .op file. The keyword `default` will make the program look for the .srf file in the default operator directory. Using again the `alter-label` keyword in the OPERATOR-SECTION of the input file, one may select a different potential with a minimum of effort.

Finally we note that is is possible to impose an energy cut-off on a non-separable potential energy surface by using the `v` keyword in the operator section of the input file. This is detailed in the HTML documentation.

6.12 Setting up Auxiliary Operators

In addition to the system Hamiltonian, other operators may be required, e.g. to generate eigenfunctions of a zero-order Hamiltonian for the initial wavepacket (see Sec. 7.6), or to calculate the time-evolution of an expectation value (either using the `expect` keyword in the RUN-SECTION, or the **ANALYSE** program **EXPECT**). Operators needed during a run must be included in the .op file. They are defined exactly as the system Hamiltonian, but are delimited by

```
HAMILTONIAN-SECTION_XXX
   .
   .
   .
end-hamiltonian-section
```

where XXX is a label to distinguish the operator. Operators to be used in any post-propagation analysis can also be set up in a separate .op file, i.e. one not containing the system Hamiltonian. The RUN-SECTION keyword `genoper = S` can then be used to set up the operators in the file S.op to the read-write file oper_S.

To check that these operators are used correctly it is necessary to understand the working of the program internal flag `diag` assigned to each operator of which the total operator is composed. If this flag is set to `.true.` then the operator is a unit operator and it will not be explicitly evaluated. This has an obvious advantage for the efficiency of the program. The program also uses these flags to determine which operator terms are separable, i.e. product terms in which all operators except for one are unit operators.

In some cases, however, unit operators must be explicitly evaluated. An example is when the matrix elements $\langle \tilde{\varphi}_a \mid h \mid \varphi_b \rangle$ are required, where $\{\tilde{\varphi}\}$ and $\{\varphi\}$ are different basis sets, which happens when e.g. the `operate` keyword is used. For this reason there is a flag `nodiag` assigned to each operator, which when set to `.true.` turns off the use of the `diag` flag. This flag is set by the program, but can also be set by hand using `nodiag` or `usediag` as the very first keyword in the HAMILTONIAN-SECTION_XXX. How this flag is set is listed in the log and op.log files. Note that the system Hamiltonian and the operators used for `eigenf`, `meigenf`, `expect`, and **pexpect** must be of `usediag` type. For `operate`,

fmat, and **flux** the `nodiag` variant is required. It could become necessary to implement the same operator twice, once with `usediag`, once with `nodiag`.

6.13 DOF, mode, and muld potentials

The potential functions from which the Hamiltonian is build are mostly one-dimensional. They are accessed through symbols like `q`, `sin`, or `vh2`. However, MCTDH can also use multi-dimensional potential functions. These are defined in funcsrf.F of usersrf.F. If the multi-dimensional potential operates on one particle (i. e. on one combined mode) it is a particle (or mode) operator. Otherwise it is a so called muld–potential. The use of muld potentials makes the propagation slow and in general one will use potfit to generate a natural potential, which is a sum of products of DOF (or mode) potentials. However, there is no disadvantage in using (multi-dimensional) mode potentials. The MCTDH program will recognise a multi-dimensional potential as mode potential, if it operates on all DOFs of one mode, but on no other DOF. Assume that there is a symbol `fxy` which refers to a potential function $f_1(x, y)$. A Hamiltonian-Section may then read

```
HAMILTONIAN-SECTION
-----------------------------------------
modes     |  R  |  x  |  y  |  z  | theta
-----------------------------------------
        .
        .
 const    |2&3 fxy  |5 cos
        .
        .
-----------------------------------------
end-hamiltonian-section
```

or equivalently

```
HAMILTONIAN-SECTION
-----------------------------------------
modes     |  R  |  x  |  y  |  z  | theta
-----------------------------------------
        .
        .
 const    |  1  |& fxy      |  1  | cos
        .
        .
---------------------------------------
end-hamiltonian-section
```

If `x` and `y` are uncombined, then `fxy` will be treated as a muld potential, but when `x` and `y` are combined to form a MCTDH particle, then `fxy` will be treated as mode operator. Inspect the op.log file. After the operator terms are summed (if possible), they are listed under the heading

```
Hamiltonian Operator Terms [h,htmdof,htmmode,htmmuld,htmtype,htmsym,string]
 No.   f   m  md Typ Sym      Term
```

A non-zero entry in the column `f` or `m` denotes that the operator term acts on the DOF `f` or the particle `m`, respectively.

If however `x`,`y`, and `z` would form one MCTDH particle, then `fxy` would not be recognised as a mode operator, because it does not act on all DOFs of the mode. One has to include `z` as a dummy DOF

```
HAMILTONIAN-SECTION
-----------------------------------------
modes      |  R  |  x  |  y  |  z  | theta
-----------------------------------------
        .
        .
 const     |2&3&4 fxy  |5 cos
        .
        .
-----------------------------------------
end-hamiltonian-section
```

or equivalently

```
HAMILTONIAN-SECTION
-----------------------------------------
modes      |  R  |  x  |  y  |  z  | theta
-----------------------------------------
        .
        .
 const     |  1  |&& fxy           | cos
        .
        .
---------------------------------------
end-hamiltonian-section
```

Assume that there is another symbol `fzx`, which refers to the function $f_2(z, x)$. An inclusion of this function may look like

```
HAMILTONIAN-SECTION
-----------------------------------------
modes      |  R  |  x  |  y  |  z  | theta
-----------------------------------------
        .
        .
 const     |4&2&3 fzx  |5 cos
        .
        .
-----------------------------------------
end-hamiltonian-section
```

Note that the dummy variable(s) must be the last one(s). The symbol `fzx` is simply interpreted here as $f_2(z, x, y)$ with no dependence on y. Note that one can freely re-order the arguments of a multi-dimensional function when using a numbered Hamiltonian line.

The use of muld potentials is restricted as there must be no "holes" in the order of modes and of DOFs within a mode. I. e. a muld potential may operate on mode 2, 3, and 4 but must not operate on mode 2 and 4 only. In the latter case there is a hole (mode 3). Holes are only allowed at the beginning and the end of a muld potential. One may fill the holes by incorporating dummy variable as done above. However, this will make the application of a muld potential even more expensive. One rather should try to re-order the modes and DOFs to avoid the holes. In any case, it is advisable to transform a muld potential to a separable natpot by using **potfit** (see Section 12.1). How a natpot is incorporated is discussed in Section 6.9.

Finally an important note. It is not possible to re-order the arguments or to add dummy variables if the special multi-dimensional “function” `readsrf` is used. In this case the re-ordering and/or addition of dummy variables must be done on the `readsrf` data file.

6.14 Golden rules for writing operator files

General remarks

File-names, modelabels, labels and parameters are case sensitive! Hence most parts of the operator file are case sensitive (in contrast to the input file). All input is assumed to be in atomic units. In contrast to the input-file this holds also for times. One hence has to explicitly give the unit `fs` when a parameter value is given in femto-seconds.

Parameter-Section (See also Section 6.2)

Parameters are real numbers. When real a number appears in the Parameter-Section it should include a dot. E. g. one should use `2.0` rather than `2` . Numbers in exponential format, e. g. `1.0d-2` should be avoided, if the exponent is not large. Use `0.01` in this case. The letter indicating the exponent must be a lower case `d`; a `D`, `e`, or `E` will not work. One may perform simple arithmetic with the parameters (see Section 6.2). In particular one may exponentiate a parameter. The exponent, however, must be a number and cannot be another parameter. (Use the function `EXP` for exponentiation). If the exponent is integer, write it as integer, e. g. write `par^3` rather than `par^3.0` . Note that the exponent can be real and also negative, e. g. `par^-0.5` is possible. The use of brackets is not allowed. Rather than writing

`cent = j*(j+1)/(2*mass)` one has to write

`cent = 0.5*j^2/mass + 0.5*j/mass .`

The string which is used to specify a parameter may consist of upper or lower case letters, numbers, and the special characters

`. _ ~ @ $ % & ?`

Note in particular that the colon (:) is **not** allowed to be part of a parameter name. It is recommended to choose names which start with a letter. Note that there is a pre-defined parameter `PI` with obvious meaning.

There is a special parameter, called `mass_`*modelabel* where *modelabel* is a label which was assigned to one of the degrees of freedom in the Primitive-Basis-Section. This special parameter should be set to the (reduced) mass of the indicated degree of freedom. This parameter is used in connection with the `KE` keyword and strange results may occur if `KE` is used but `mass_`*modelabel* is not set. Do not use this construct when the second part is not a valid modelabel. E. g. for the total mass do not use `mass_tot` but rather use `mass@tot` or `mass.tot` or `MassTot` instead.

Labels-Section (See also Section 6.4)

One may use the same set of letters, numbers and special characters for labels, as are allowed for parameters. Again, the colon (:) is not allowed to be part of the name, although pre-defined labels (i.e. those listed in Appendix C) may contain a colon. Moreover, the underscore has a special meaning for labels. If a label has the structure *label_modelabel* then the (QUANTICS program will put the corresponding operator in that column of the Hamiltonian-Section which refers to *modelabel*. One must not put it explicitly there. Unit operators are assumed for all other degrees of freedom. This feature, which is often used to include CAPS, excludes the general use of the underscore in a label. E. g. defining a label as `exp_1` may produce an error, because `1` may not be a modelabel.

Note that there must not be a parameter and a label which have the same name. E. g. `q` cannot be used as parameter because it is pre-defined as a label. The program checks that parameter and label names are disjoint.

Hamiltonian-Section (See also Section 6.3)
Only simple labels may appear in a Hamiltonian-Section. Operators with arguments must be assigned to a simple label in the Labels-Section.

With the aid of the caret ^ one may apply a power to operators. The power may be integer or real and may carry a sign. This, however, works only for potential like operators. Inspect Appendix C to learn, which operators can be exponentiated. Note, that symbols like `dq^2` or `j^2` are operator labels of their own right, they do not denote that the second power of the operators `dx` or `j` is taken literally.

Time-dependent operators can easily be implemented. The time is simply treated as an additional DOF of the Hamiltonian-Section. The modelabel of this additional DOF must be `Time`. See the HTML-documentation ("Hamiltonian/Liouvillian Documentation" and then "Time-dependent Operators") for further details. See also the operator file nocl1T.op on $MCTDH_DIR/operators.

Chapter 7

Generating the initial wavepacket

For a quantum dynamical calculation an initial wavepacket $\Psi(0)$ is required. This is done in the INIT_WF-SECTION of the input file. The initial wavepacket must have a particular form, depending on the method to be used. If the MCTDH scheme is employed, $\Psi(0)$ has to be represented as a multi-configurational Hartree product (i.e. a linear combination of products of single-particle functions), while in a numerically exact calculation it must be mapped onto a product of DVR grids. Usually, $\Psi(0)$ is a simple Hartree product, i.e. a product of one-dimensional functions (unless spherical harmonics are employed, which are two-dimensional). The MCTDH program offers a number of function types to be used for each factor in the product, i.e. each degree of freedom.

7.1 Building Gaussian functions as initial functions

A possible choice for the one-dimensional initial functions are Gaussian functions. These can be defined in two manners which differ only by the way the width is specified, namely either as

$$\varphi(x) = N\, e^{-1/4\,((x-x_0)/\Delta x)^2}\, e^{ip_0(x-x_0)} \tag{7.1}$$

or as

$$\varphi(x) = N\, e^{-1/2\, m\omega\,(x-x_0)^2}\, e^{ip_0(x-x_0)}\,, \tag{7.2}$$

with corresponding keywords `gauss` and `HO`. Here N is a normalisation constant, x_0 and p_0 are the centre and initial momentum, Δx is the width, and m and ω denote mass and frequency.

Suppose there are two degrees of freedom `X` and `Y`, then the initial wavepacket may be defined by an INIT_WF-SECTION reading

```
INIT_WF-SECTION
build
    X    gauss    4.315    0.0    0.0794
    Y    HO       2.151    0.0    0.218,eV    13615.5
end-build
end-init_wf-section
```

The keywords `build` and `end-build` enclose the lines that specify how to build the initial wavefunction. The first and second number in each line denote x_0 and p_0, respectively. The next numbers are Δx in the first case, and ω and m in the second. As the example shows, one may add a unit to the parameters. Note that Δx and ω may be complex.

Plane waves may be generated by setting the frequency within the HO line to zero. E. g.

```
build
    X    HO    0.0    0.0    0.0
    Y    HO    0.0    2.5    0.0
end-build
```

will generate a flat function for the X degree of freedom and a plane wave with momentum 2.5 au for the Y degree of freedom.

In a numerically exact calculation, the initial wavepacket is simply the product of the functions (7.1) or (7.2) for the degrees of freedom involved. In an MCTDH calculation, however, the program interprets each line in the INIT_WF-SECTION as first single-particle function for that degree of freedom. Higher single-particle functions are then constructed by multiplying the preceding function by x (so producing a series of powers of x), followed by Schmidt-orthogonalisation onto the lower functions. The set of all products of the functions of the included modes then defines the initial configurational space.

The initial wavefunction in an MCTDH calculation is then chosen as one of these configurations. The default is to use the product of the first single-particle functions of each mode, thus arriving at the same initial wavefunction as in a numerically exact calculation: the product of the functions (7.1) or (7.2). One may however also populate a different configuration, with the aid of the pop keyword, e.g.

```
INIT_WF-SECTION
build
    X    gauss    4.315    0.0    0.0794                 pop = 2
    Y    HO       2.151    0.0    0.218,eV    13615.5    pop = 3
end-build
end-init_wf-section
```

The initial wavepacket is in this example the product of the second single-particle function in X and the third in Y.

7.2 Setting up Legendre functions as initial functions

An initial function frequently employed for angular modes is an associated Legendre function

$$\phi_{l-m+1}(\theta) = \sqrt{\frac{2\,l+1}{2}\,\frac{(l-m)!}{(l+m)!}}\,P_l^m(\cos\theta)\,, \tag{7.3}$$

with $0 \leqslant m \leqslant l$. The parameter m denotes the magnetic quantum number and is treated as a fixed parameter. P_l^m is the (unnormalised) associated Legendre function (4.4).

The following example defines the initial wavepacket of a system with two degrees of freedom rd and theta as product of a Gaussian function in rd and a Legendre polynomial in theta:

```
INIT_WF-SECTION
build
    rd       gauss    4.50d0    -8.76d0    0.18d0
    theta    Leg      0         0          sym
end-build
end-init_wf-section
```

The numbers after the keyword `Leg` denote m and l, respectively. If the corresponding type of the primitive basis is not `Leg`, then m must be zero; if it is `Leg`, then m must coincide with the value in the PRIMITIVE-BASIS-SECTION.

In an MCTDH calculation the program uses the Legendre polynomial specified in the INIT_WF-SECTION to define not only the initial wavepacket but also the first single-particle function. Which higher single-particle functions are used depends on the last parameter, which can be `sym` or `nosym`. In the latter case all values of l (both even and odd) are employed, in the former case only those Legendre functions having the same symmetry as l (either even or odd) are taken.

7.3 Setting up extended Legendre functions as initial functions

When the extended Legendre DVR is used an initial associated Legendre function and the corresponding K-function (which is a Kronecker δ) should be generated via the `KLeg` and `K` keywords.

```
INIT_WF-SECTION
build
    ...........
    theta KLeg        2          sym
    K       K         1
    ...........
end-build
end-init_wf-section
```

The number after the keyword `KLeg` denotes the initial ℓ, and the keyword `sym` accomplishes that for the generation of higher single-particle functions only every second one will be taken. I. e. there will be only even or only odd ℓ's depending on whether the initial ℓ is even or odd. The keyword `sym` may be replaced by `nosym` with obvious meaning. The number following `K` denotes the initial value of K. Note that the `KLeg` and `K` keywords of the INIT_WF-SECTION may also be used when the `PLeg` DVR is employed.

7.4 Generating spherical harmonics as initial functions

If spherical harmonics have been employed as primitive basis functions for a combined mode `alpha` and `beta`, a normalised spherical harmonic

$$Y_{jm}(\alpha,\beta) = \sqrt{\frac{2j+1}{4\pi}\,\frac{(j-m)!}{(j+m)!}}\,P_j^m(\cos\alpha)\,e^{im\beta} \qquad (7.4)$$

is the appropriate initial function for that mode. Here P_j^m denotes the associated Legendre function (4.4).

Spherical harmonics can be selected similarly to the primitive basis by the INIT_WF-SECTION

```
INIT_WF-SECTION
build
    ...........
    alpha    sphfbr    0
    beta     phifbr    0
```

```
    ...........
end-build
end-init_wf-section
```

The two numbers specify j and m. Of course, the initial spherical harmonic must be part of the primitive basis set.

In an MCTDH calculation, this again defines not only the initial wavepacket but also the first single-particle function. Higher single-particle functions are other spherical harmonics whose quantum numbers are as close as possible to the quantum numbers of the first one.

7.5 Generating Wigner functions as initial functions

For rotational motion of polyatomic molecules in three dimensions, Wigner-D functions can be used for the initial wave function. These are defined as:

$$\overline{D}^{j}_{m,k}(\alpha,\beta,\gamma) = \sqrt{\frac{2j+1}{8\pi^2}} D^{j}_{m,k}(\alpha,\beta,\gamma) \tag{7.5}$$

$$D^{j}_{m,k}(\alpha,\beta,\gamma) = e^{-im\alpha} d^{j}_{m,k}(\beta) e^{-ik\gamma} \tag{7.6}$$

$$d^{j}_{m,k}(\beta) = \left\langle j,m \left| e^{-i\beta \hat{J}_Y} \right| j,k \right\rangle \tag{7.7}$$

where $\overline{D}^{j}_{m,k}(\alpha,\beta,\gamma)$ is the normalized Wigner-D function and $d^{j}_{m,k}(\beta)$ is the Wigner (small)-d function, and $-j \leqslant m, k \leqslant j$.

A Wigner-D function can be generated as an inital wave function by specifying the `wigner` keyword, followed by two `K` lines, as in the example below:

```
INIT_WF-SECTION
build
-----------------------------------------------------------
# mode    type  q.n.
-----------------------------------------------------------
...
  beta   wigner  5    nosym excite=mkj  print  # j of initial wigner
  gamma      k   -3    -7  7 1    # initial-k, k-range, k-step size
  alpha      k    1    -2  2 1    # initial-m, m-range, m-step size
...
-----------------------------------------------------------
end-build
END-INIT_WF-SECTION
```

The Wigner-(big)-D function is generated in the above example as the product of a Wigner (small)-d function for the β Euler angle and a Kronecker-δ for the associated momentum quantum number of each of the α and γ angles. The first number following `wigner` denotes the j-value of the initial wavefunction; initial values of the k and m quantum numbers along with their ranges and step sizes are given in the first and second `K` lines, respectively. The corresponding DVR for `wigner` initial wavefunctions must be `wigner` for the first degree of freedom in the combined mode; either `exp` or `K` are allowed DVR/FBR types for the second and third degrees of freedom. The degrees of freedom are assumed to be given in the order $|J, K, M\rangle$. The `excite` keyword can be used to choose between two different schemes for generating unoccupied single-particle functions: `excite=mkj` preferentially excites m-states, then k, then j, while `excite=kmj` preferentially excites k, then m, then j-states.

7.6 Generating eigenfunctions of a one-dimensional Hamiltonian

It may be useful to start a calculation with the wavepacket in a particular eigenstate of a zero-th order Hamiltonian. This occurs, for example, in an atom-diatom scattering calculation when the diatom starts in a particular vibrational eigenstate. To do this, one must first define the zero-th order operator. This is done by including a HAMILTONIAN-SECTION_OPER section in the operator file (see Sec. 6.12) to define an operator labelled OPER (any other string except SYSTEM can be chosen for this name).

As an example, a one dimensional H_2 Hamiltonian operator can be defined by adding the section

```
HAMILTONIAN-SECTION_H2
usediag
-------------------------------------------
  modes        |  rd    |  rv  |  theta
-------------------------------------------
1.0            |  1     |  KE  |    1
1.0            |  1     |  vh2 |    1
-------------------------------------------
end-hamiltonian-section
```

to the operator file. In fact, the simpler input

```
HAMILTONIAN-SECTION_H2
------------------
  modes    |  rv
------------------
1.0        |  KE
1.0        |  vh2
------------------
end-hamiltonian-section
```

works as well, because `usediag` is default for `eigenf`, and for DOF's which are not listed a unit operator is assumed by default.

The desired functions are then generated by using the `eigenf` keyword in `build` block of the INIT_WF-SECTION, e.g.

```
INIT_WF-SECTION
build
    rd       gauss   4.50d0    -8.76d0   0.18d0
    rv       eigenf    H2   pop=2
    theta    leg         jbf      sl0         sym
end-build
end-init_wf-section
```

generates a three dimensional wavepacket with a Gaussian along mode rd and the second eigenfunction (i. e. the first excited state) of the operator H2 for rv. (NB. pop=1 is default and may be dropped). For the theta degree of freedom an associated Legendre function is taken. The associated Legendre function is specified by the value of the parameters `jbf` and `sl0`.

If the `veigen` keyword has been included in the RUN-SECTION, then the eigenfunctions and eigenvalues are written to the file veigen. In this way this procedure can be used to numerically exactly diagonalise a one-dimensional operator. If the `veigen` keyword is not given, the eigenvalues are still written to the log file.

NB. This is only possible if the primitive basis for the degree of freedom is a DVR basis (i.e. not an FFT) as the program generates the eigenfunctions by diagonalising the operator represented as a real matrix.

7.7 Reading the initial wavepacket from file

Instead of building a new initial wavepacket, one may also read a wavefunction that has been created in a previous calculation from the restart file. This is done by the `file` keyword in the INIT_WF-SECTION:

```
INIT_WF-SECTION
file = oldrun, orthopsi
end-init_wf-section
```

Here `oldrun` is the path of the directory where the restart file is stored. If no path is specified, the restart file is searched for in the name-directory. The second (optional) parameter, which can be `orthopsi` (the default) or `noorthopsi`, specifies whether or not the single-particle functions are Schmidt-orthogonalised after being read.

The primitive basis must be defined in the PRIMITIVE-BASIS-SECTION identically to the one of the previous calculation from which the initial wavepacket is being read. This can be ensured by reading the definition of the primitive basis of the previous run from file using the `readdvr` keyword in the RUN-SECTION, rather than defining the primitive basis in a PRIMITIVE-BASIS-SECTION. The number of single-particle functions, however, may differ.

Since version 8.3.10 there is also a `Read-Inwf ... end-read-inwf` block. In contrast to the simple `file` keyword, this allows to distribute the SPFs and the blocks of the A-vector freely among the electronic states. In particular, the current system and the wavefunction read in do no longer need to have the same number of electronic states. Example:

```
INIT_WF-SECTION
Read-Inwf
   file = gs
   SPF 1 -> 1,2,3
   A   1 -> 2
end-read-inwf
end-init_wf-section
```

Here, the file which is read in, gs/restart, has only one electronic state. Its SPFs are copied to all the three states of the current system and its A-vector is copied to state 2. The A-vector blocks for state 1 or 3 are hence zero. See the HTML documentation for more information.

7.8 *Advanced topic*: Diagonalising a multi-dimensional operator to create multi-dimensional SPFs

With the `meigenf` feature it is possible to diagonalise one– or multi–dimensional hermitian Hamiltonians to create one– or multi–dimensional SPFs. As `meigenf` uses the Lanczos algorithm (with full re-orthogonalisation) for diagonalisation, it needs some initial guess for the SPF. Hence the keyword `meigenf` must not be given in a build-block, it may come after a build block. However, `meigenf` can also alter the SPFs of a wavefunction which is read from file.

Example: `meigenf = 3,oper,0`

Here, `meigenf` will diagonalise the operator `oper`, which must be defined in a Hamiltonian-Section (similar to `eigenf`). The eigenfunctions of `oper` will then replace the SPFs of the third mode (first argument). The third argument, `0`, finally indicates that the ground state is taken as the first SPF. (In contrast to `eigenf`, `meigenf` counts the eigenfunctions from zero). The Lanczos iteration is stopped, when the selected state (here the ground state) is converged. Adding the argument `full` forces `meigenf` to perform as many iterations as there are grid points, leading to a numerically exact full diagonalisation of the operator. (This is not recommended if the particular mode under discussion is represented by many grid points, more than 300 say). With an additional integer argument one may limit the number of iterations. Finally, if the integer argument for the selected eigenstate is replaced by the argument `follow`, then that eigenfunction, which has the largest overlap with the initial function, will be taken as first SPF.

Example: `meigenf = 3,oper,follow,full,select,write,125`

In this example the maximum number of arguments is given. See the HTML documentation for explanation and more information.

7.9 *Advanced topic*: Generating an initial wavepacket using an operator

It is also possible to first generate an initial wavepacket, and then to apply an operator to this wavepacket before starting the propagation. This is required, e.g., when the initial wavepacket to be generated is the dipole operator acting on a ground state wavefunction.

To do this, a wavepacket must be build or read in as described in this section above. The operator must also be defined in the operator file using a HAMILTONIAN-SECTION_OPER section and setting `nodiag` (see Sec. 6.12). This sets up an operator labelled OPER (any other string except SYSTEM can be chosen for this name). Once this has been done, adding the keyword `operate=OPER` to the INIT_WF-SECTION generates a wavepacket by appling this operator to the initial packet.

For a numerically exact wavefunction this procedure is simple. For an MCTDH wavefunction however the optimal single-particle functions for the final wavepacket may be different from those of the initial wavepacket. To optimise the basis functions for the new wavefunction, an iterative procedure is used. Details of the iterations are output in the log file.

7.10 *Advanced topic*: Creating a set of initial wavepackets

Instead of propagating only a single wavepacket, one may also define a set of P initial wavepackets $\Psi_1, \ldots, \Psi_P$, which are then propagated simultaneously. This is called a multi-packet calculation.

For example, to propagate $P = 2$ wavefunctions with coordinates X and Y, one first has to add the line

```
packets = 2
```

to the SPF-BASIS-SECTION in order to specify the number P of packets. The definition for the initial wavefunction has to be given for each wavepacket, e.g.

```
INIT_WF-SECTION
build
    X    gauss    4.315    0.0    0.0794                  pack = 1
    Y    gauss    3.2      0.0    0.053                   pack = 1
    X    HO       2.151    0.0    0.218,eV    13615.5     pack = 2
    Y    gauss    3.840    0.0    0.1378                  pack = 2
end-build
end-init_wf-section
```

The pack keyword defines the packet to which an input line belongs. It is possible to specify more initial packets in this section than given by the packets argument. All data with pack $>$ packets will then be ignored.

Note that the auto file now contains the cross-correlation matrix

$$c_{\alpha\beta}(2t) = \langle \Psi^*_\alpha(t) \,|\, \Psi_\beta(t) \rangle \,, \quad \alpha, \beta = 1, \ldots, P \,, \tag{7.8}$$

rather than the auto-correlation function. See the HTML documentation for the exact format of the auto file.

7.11 *Advanced topic*: Setting up (a)diabatically corrected initial wavepackets

The flux-analysis and similarly **twprob** require the knowledge of the energy distribution of the (initial) wavepacket. This energy distribution is written to the enerd file, if the keyword correction is given in the INIT_WF-SECTION. (NB: The file enerd is called adwkb in older versions.) The keyword correction requires an argument. If correction = edstr is given, it is assumed that the wavepacket is located far outside such that the interaction potential can savely be neglected. The energy distribution is then given by Eq.(140) of the MCTDH review [1], i. e. essentially by a fourier transform of the single-particle function of the translational degree of freedom. It is assumed that the translational degree of freedom is the dof number 1, i. e. is the first item in the PRIMITIVE-BASIS-SECTION. Otherweise the keyword trans = I1 (,I2) must be given, where I1 denotes the number of the translational dof and (optional) I2 its electronic state.

The influence of the interaction potential is (partly) corrected for when giving correction = dia. The energy distribution is now evaluated as the overlap of the translational single-particle function with a distorted wave (rather than a plane wave). The distorted wave is the solution of a 1D Schrödinger equation employing the translational mean field as interaction. (See the MCTDH review [1] Chapter 7.2 for details). The distorted wave is no longer calculated through the WKB approximation, but evaluated numerically using the Numerov method.

The quality of the energy distribution may be further improved by replacing the argument `dia` by `ad`. An adiabatic correction is now performed which modifies the single-particle functions of the internal degrees of freedom. This improves the mean-field and in turn the distorted wave. Adiabatic correction, however, is presently only implemented for the $H+H_2$ system and its isotopic variants.

Remarks:

- The translational degree of freedom must not be combined with other dof's.
- There are special routines for the $H+H_2$ system (and its isotopic variants). These are used when the argument `hh2` is additionally given with the keyword `correction`, e. g. `correction = hh2, ad`.
- Presently, the adiabatic correction works only in combination with the `hh2` argument.

Chapter 8

Choosing an integration scheme

In a propagation or relaxation calculation an ordinary differential equation has to be solved. This can be accomplished by different integration methods. The first two sections, 8.1 and 8.2, are dealing with integration techniques developed especially for the MCTDH method. Section 8.3 describes general integrators, which can be used in both MCTDH and numerically exact calculations.

8.1 Using the VMF integration scheme in an MCTDH calculation

In an MCTDH calculation one possible integration method is the variable mean-field, or VMF scheme, which is described in Sec. 5.1 of Ref. [1] and in Ref. [20]. As the name implies, in the VMF scheme the mean-fields are determined in each integration step. The VMF scheme is the default in an MCTDH calculation.

As can be seen from Tab. 8.1, the equations of motion in the VMF scheme can be solved with an ABM (the default), BS or RKx integrator. ABM stands for Adams-Bashforth-Moulton predictor-corrector method, BS for Bulirsch-Stoer extrapolation scheme and RKx for a Runge-Kutta integrator or fixed order x, where $x = 5$ and $x = 8$ are available. We recommend the use of the ABM method because it is generally more efficient.

To choose a VMF calculation employing the ABM integrator, the INTEGRATOR-SECTION in the input file should read, e.g.,

```
INTEGRATOR-SECTION
  VMF
  ABM = 6, 1.0d-7, 0.01d0
end-integrator-section
```

The parameters after the `ABM` keyword are explained in Sec. 8.3. When the BS integrator is desired, a possible INTEGRATOR-SECTION is

```
INTEGRATOR-SECTION
  BS = 8, 1.0d-6
end-integrator-section
```

Here we have omitted the `VMF` keyword since it is the default.

Note that one may also not define at all the INTEGRATOR-SECTION. This is equivalent to specifying the `VMF` and `ABM` keywords, together with some default parameters for the ABM integrator that can be found in the HTML documentation.

Table 8.1: Available integrators in dependence of the calculation type. The table displays which of the integrators ABM, BS, RKx and SIL can be chosen depending on whether a VMF, CMF, or numerically exact calculation is being made. An underlined checkmark "$\underline{\checkmark}$" indicates the default.

	Integrator			
Calculation type	ABM	BS	SIL	RKx
VMF	$\underline{\checkmark}$	✓	—	✓
CMF, A-vector	✓	✓	$\underline{\checkmark}$	✓
CMF, φ-vector	✓	$\underline{\checkmark}$	—	✓
Numerically exact	$\underline{\checkmark}$	✓	✓	✓

8.2 Using the CMF integration scheme in an MCTDH calculation

In many cases an MCTDH calculation is more efficient if the VMF scheme is replaced by the constant mean-field, or CMF scheme. In the CMF scheme the numerical effort is reduced by holding the mean-fields, density matrices, and Hamiltonian matrix elements constant for some time, rather than evaluating them in each integration step. Note that the CMF scheme does (presently) not work in combination with the CDVR approximation. The CMF scheme is detailed in Sec. 5.2 of Ref. [1] and in Ref. [20].

Table 8.1 displays the integrators being compatible with the CMF method. Since the MCTDH coefficients (i.e. the A-vector) and the single-particle functions (i.e. the φ-vector) are propagated separately in the CMF scheme, different integrators can be chosen for each of them. This is indicated by appending `/A` or `/spf` to the `ABM`, `BS`, `RK5`, `RK8` or `SIL` keyword. The default is `SIL/A` and `BS/spf`, which is in general the most efficient combination.

An example for the INTEGRATOR-SECTION in the input file is

```
INTEGRATOR-SECTION
  CMF    = 0.5d0, 1.0d-6
  SIL/A  = 15, 1.0d-7
  BS/spf =  9, 1.0d-7
end-integrator-section
```

This starts a CMF calculation with an initial stepsize of $0.5\,\mathrm{fs}$ and an error tolerance of 10^{-6}. The parameters for the SIL and BS integrator are described in Sec. 8.3. If the same integrator (e.g. ABM) is to be used for the MCTDH coefficients and the single-particle functions, the shortcut `/all` can be appended to the integrator keyword:

```
INTEGRATOR-SECTION
  CMF     = 1.0d0, 1.0d-5
  ABM/all = 5, 1.0d-4, 0.05d0
end-integrator-section
```

Note, however, that the ABM integrator typically will not give you the optimal performance of the CMF scheme. As a final example, the CMF scheme may also be selected by

```
INTEGRATOR-SECTION
  CMF
end-integrator-section
```

The program then uses default integrators and parameters, which are compiled in the HTML documentation.

In the above examples, two (optional) parameters are used to concretise the CMF calculation. The first one is the initial stepsize (in fs). A good guess for the initial stepsize is to use the output interval `tout` specified in the RUN-SECTION. Whether the initial stepsize was chosen reasonably can be checked by looking at the update file, which will be generated in a calculation if the `update` keyword in the RUN-SECTION is set. The update file indicates whether repetition steps were necessary in the beginning of the propagation. If so, one should use a smaller initial stepsize in the following calculations.

The second parameter defines the CMF error tolerance, which controls the stepsizes during the propagation. Typical values lie between 10^{-4} (very low accuracy) and 10^{-8} (very high accuracy). For many applications an error tolerance of 10^{-5} or 10^{-6} will be sufficient. The convergence of a calculation with respect to the CMF error can be checked by comparing the results of two calculations performed with different error tolerances.

We finally note that it is possible to perform a CMF calculation with fixed or variable stepsizes. To choose among the possible options use the keywords `CMF/var`, `CMF/varphi`, `CMF/vara` or `CMF/fix`, respectively. With the extension `var` the stepsize becomes variable and is controlled by both the single particle functions and the A-vector. As `var` is default, `CMF/var` is identical to `CMF`. Using the extension `varphi` or `vara`, the stepsizes are controlled only by the single particle functions or only the A-vector, respectively. Finally, the extension `fix` enforces the use of a fixed stepsize. To discriminate these CMF stepsizes from the integrator step sizes, the former are often called update times.

8.3 Description of the available integrators

The integrators that are available are an Adams-Bashforth-Moulton (ABM) predictor-corrector method with fixed order and variable stepsize, a Bulirsch-Stoer (BS) extrapolation scheme with polynomial extrapolation and variable order and stepsize, two Runge-Kutta (RK5/8) integrators with adaptive stepsize and fixed order (5 or 8, respectively), and a short iterative Lanczos (SIL) algorithm with variable order and stepsize. Note that the (hermitian) SIL integrator is automatically replaced by a complex SIL integrator, also known as Lanczos-Arnoldi integrator, if the Hamiltonian is complex. More precisely, the Lanczos-Arnoldi routine is automatically chosen, if there is a complex potential (e.g. a CAP) in one of the separable parts of the Hamiltonian. However, the Hamiltonian may be non-hermitian for various other reasons. In these case one has to replace the `SIL` keyword by `CSIL`, which enforces the use of the Lanczos-Arnoldi integrator. Note that the integration will be incorrect, if the (hermitian) SIL integrator is used for a non-hermitian Hamiltonian! Which integrator has been used is protocoled in the log-file.

Each integrator is associated with up to three parameters. In case of the ABM, BS and SIL integrators the first one is the integration order and the second one the error tolerance, while the last one depends on the integrator. Typical error tolerances range from 10^{-3} or 10^{-4} (low accuracy) over 10^{-5} or 10^{-6} (normal accuracy) to 10^{-7} or 10^{-8} (high accuracy). For the RK5 and RK8 integrators (where the order is fixed), the first parameter specifies the error tolerance and the second one the initial stepsize. For all integrators it is in general not useful to work with an error tolerance less accurate than 10^{-5}.

When performing a calculation, one should first select the desired error tolerance. The second step is to define the integration order. The meaning of this parameter is slightly different for the three integrators which provide it. For the ABM integrator, which runs with a fixed order, the order-parameter in the INTEGRATOR-SECTION is the true integration or-

Table 8.2: Optimal orders for the ABM and BS integrators in dependence of the error tolerance. The optimal ABM order was found empirically and might differ slightly in other cases. The values for the BS integrator, on the other hand, can be proved to be the optimal orders. (What is called "optimal BS order" in this guide is actually the maximum number of extrapolations.)

Error tolerance	Optimal ABM order	Optimal BS order
10^{-3}	3	4
10^{-4}	4	5
10^{-5}	5	7
10^{-6}	5	8
10^{-7}	6	9
10^{-8}	6	10

der. For the BS and SIL integrators, which continously adapt their integration order during a run, the order-parameter denotes the maximum number of extrapolations and the maximum integration order, respectively. What the order-parameter hence defines is actually the memory being allocated, as all three integrators have in common that with each increase of the order-parameter by one, one additional wavefunction vector must be stored.

The order-parameter of the ABM and BS integrators should be chosen according to Tab. 8.2. Larger values do not increase the efficiency but only enlarge the memory requirements. (In the case of the ABM integrator a larger value in fact decreases the efficiency.) Smaller values for the order-parameter lead to longer CPU times. They might however be used if memory must be saved.

The optimal order-parameter of the SIL integrator unfortunately cannot be predicted but has to be found out empirically for each system. Typical values range from 6 to 16. After a calculation the largest order the SIL integrator has used is given in the log file. If this value is smaller than the order-parameter, you should decrease the order-parameter accordingly, to avoid the waste of memory in future calculations. If the largest order equals the order-parameter, this indicates that the efficiency might become higher if a larger order-parameter is chosen, so increase the order-parameter for optimal performance. When memory-intensive systems are investigated, it again might become necessary to use a smaller than optimal order-parameter, at the price of a longer computation time.

If the ABM, BS or RKx integrator is employed, the last parameter to be specified is the initial stepsize (in fs).. In the case of the BS integrator, the output interval `tout` defined in the RUN-SECTION is normally a good choice. For the ABM integrator, the initial stepsize should in general be by a few orders of magnitude smaller than the output interval. This is because the ABM integrator — although being a multi-step method — has to be started as a one-step method, i.e. with an order of two, since initially the wavefunction is given for a single point of time only. In case of the RKx integrators, the initial stepsize can also be omitted or set to zero. The integrator then tries to guess a suitable value for the initial stepsize by employing a single explicit Euler step and estimating the second derivative of the solution. (However, in our experience this guess is often too conservative.) Whether the initial ABM, BS or RKx stepsize was chosen reasonably can be decided with the aid of the steps file, which is generated when the `steps` keyword in the RUN-SECTION is set. From this ASCII file it can be seen how large the first (successful) step actually was. This value may then be used as initial stepsize in future calculations.

For the complex SIL method, two different error estimates, called standard and improved estimate, are implemented, which can be specified by the third parameter. The standard error criterion is based on the product of the sub-diagonal elements of the Lanczos matrix. The improved one uses the norm of the difference between the wave functions propagated with two consecutive orders. The improved estimate requires slightly more computation time but is more reliable when the stepsize is large. For details we refer the reader to Refs. [1,20]. The estimates can be activated by the keywords `standard` or `novel`, respectively. The former is the default.

8.4 Fine-tuning the Equations of Motion and the Integration Scheme.

There are a number of keywords that can be added to the INTEGRATOR-SECTION that change the form of the equations of motion, or change the way the integration is performed. Examples of these are given in this section.

8.4.1 *Advanced topic*: Propagating in natural or interaction picture orbitals

Instead of the standard single-particle functions one may employ natural or interaction picture orbitals. Natural orbitals are those single-particle functions that diagonalise the MCTDH density matrices. Interaction picture orbitals are obtained by moving from the Schrödinger to the interaction picture. For details see Secs. 3.3 and 3.4 of the review [1]. In normal use natural orbitals have no advantages over normal single-particle functions: they span the same space, and may even force the integrator to take smaller steps. The interaction picture may allow the equations of motion to be integrated more efficiently than the standard VMF scheme. This is especially true if an operator has a large separable part. It is however usually less efficient than the CMF scheme.

The type of orbitals to be used is selected in the INTEGRATOR-SECTION of the input file, since the orbital type affects the form of the equations of motion to be integrated. Place the keyword `natorb` or `interpic` into the INTEGRATOR-SECTION, i.e.

```
INTEGRATOR-SECTION
   .
   .
   .
   natorb
end-integrator-section
```

or

```
INTEGRATOR-SECTION
   .
   .
   .
   interpic
end-integrator-section
```

in order to move from standard to natural or interaction picture orbitals, respectively.

8.4.2 Suitable integrator settings for improved relaxation

The integrator settings for improved relaxation are somewhat different from those for propagation. Improved relaxation requires a CMF/fix or CMF/varphi integration scheme. The best is simply to use CMF, this defaults to CMF/var for propagation runs and to CMF/varphi for improved relaxation. Improved relaxation furthermore requires a Davidson "integrator" (actually a diagonalizer), i.e. the keyword `DAV`, `rDAV`, `rrDAV`, or `cDAV`. A typical setting might read:

```
INTEGRATOR-SECTION
  CMF     = 1.0, 3.0d-3
  RK8/spf = 1.0d-9
  rrDAV/A = 200, 1.0d-8
  natorb
  eps_inv=1.0d-10
end-integrator-section
```

Note that the CMF-accuracy is rather low, whereas the accuracy of the integrators is rather high. Note also that the parameter for regularizing the density matrices, `eps_inv`, is also set to a low value (its default value is 10^{-8}). This is because the lowest natural SPF populations are in the range $10^{-6} \cdots 10^{-10}$ for improved relaxation runs, whereas they are typically in the range $10^{-3} \cdots 10^{-6}$ for propagation runs.

The RK8 integrator was found to perform best for SPF relaxation. If a hight accuracy of the results is not required, one may set the RK8 accuracy to `1.0d-8` and `eps_inv=1.0d-9`, or even remove the `eps_inv` line.

As orbital-type `natorb` was chosen in this example. The default for improved relaxation is `energyorb` (may be abbreviated to `enorb`). Energy orbitals make the Hamiltonian matrix more diagonal dominant than other orbital choices, which accelerates the convergence of the Davidson digonalizer. However, the computation of the energy orbitals is a bit costly and we noticed that, in particular when a preconditioner is used, it is often more efficient to use natural orbitals. Standard orbitals, `stdorb`, which are default for propagation, can also be used. Note that the use of energy orbitals requires that the keyword `orben` is set in the Run-Section.

8.4.3 *Advanced topic*: Evaluating potentials using the TDDVR or CDVR method

The time-dependent DVR (TDDVR) and Correlation DVR (CDVR) methods offer the possibility of employing non-separable potentials within the MCTDH scheme, without the strong increase of computational labour that arises when such potentials are evaluated directly. The disadvantage of these procedures is their introduction of an additional, unpredictable error in the calculation. For a discussion of the TDDVR and CDVR methods see Ref. [1].

The TDDVR or CDVR method can be used by simply inserting either the keyword

```
TDDVR
```

or the keyword

```
CDVR
```

into the INTEGRATOR-SECTION of the input file. Presently, both TDDVR and CDVR work only in combination with the VMF integration scheme.

Chapter 9

Treating non-adiabatic systems

To treat a non-adiabatic system, i.e. a system which involves a manifold of coupled potential energy surfaces, the Hamiltonian operator has to be set up appropriately. Furthermore, the primitive and the single-particle basis, as well as the initial wavepacket, have to be defined in a special way when the system is non-adiabatic.

9.1 Setting up the Hamiltonian for a non-adiabatic system

The Hamiltonian of a non-adiabatic system with σ electronic states can be written as $(\sigma \times \sigma)$-matrix

$$\boldsymbol{H} = \begin{pmatrix} H_{11} & \dots & H_{1\sigma} \\ \vdots & & \vdots \\ H_{\sigma 1} & \dots & H_{\sigma\sigma} \end{pmatrix} . \tag{9.1}$$

For the sake of simplicity we assume in the following that only two electronic states have to be accounted for, i.e. $\sigma = 2$. The generalisation to larger numbers of states is straightforward. To implement the Hamiltonian matrix, which is now two-dimensional, into the MCTDH program, it must have the product form required by the MCTDH method:

$$\boldsymbol{H} = \sum_{k=1}^{s} c_k \, h_k^{(1)} \dots h_k^{(f)} \begin{pmatrix} \gamma_{11}^{(k)} & \gamma_{12}^{(k)} \\ \gamma_{21}^{(k)} & \gamma_{22}^{(k)} \end{pmatrix} , \tag{9.2}$$

where f denotes the number of molecular degrees of freedom and $h_k^{(\varkappa)}$ is a one-dimensional operator acting exclusively on the $\varkappa$th degree of freedom. It is no restriction to assume that the elements of the γ-matrices are only zero and one, i.e. $\gamma_{ij}^{(k)} \in \{0, 1\}$.

The program knows a number of built-in symbolic expressions that can be used to define the γ-matrices in the Hamiltonian section of the operator file. These symbols are compiled in Tab. 9.1. For instance, the symbol `S1&1` specifies the symmetric matrix that couples states 1 and 1, while `Z1&2` stands for the unsymmetric matrix that couples initial state 2 with final state 1. Note that the symbol `S1&2` implies that initial state 2 couples with final state 1 *and* vice versa, because the corresponding matrix is symmetric.

Table 9.1: The built-in symbolic expressions that can be used to define the couplings of a non-adiabatic Hamiltonian. (Two electronic states are assumed.)

Matrix	Symbol	Matrix	Symbol
$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	`1`	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	`S1&2` (or `S2&1`)
$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	`S1&1`	$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$	`Z1&2`
$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$	`S2&2`	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$	`Z2&1`

As an example, the operator file for the 4-mode 2-state model of the pyrazine molecule is shown in Example 9.1. The Hamiltonian for this system reads

$$\boldsymbol{H} = \sum_i \frac{\omega_i}{2} \left(-\partial^2_{Q_i} + Q_i^2\right) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -\Delta & 0 \\ 0 & \Delta \end{pmatrix} + \sum_{i \neq 10a} \begin{pmatrix} \varkappa_i^{(1)} & 0 \\ 0 & \varkappa_i^{(2)} \end{pmatrix} Q_i + \begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix} Q_{10a} \,, \tag{9.3}$$

with $i = 10a, 6a, 1, 9a$. The numerical parameters are defined in the PARAMETER-SECTION, e.g. `w10a` corresponds to ω_{10a} and `k6a1` to $\varkappa_{6a}^{(1)}$. This Hamiltonian is represented in the operator file by the HAMILTONIAN-SECTION (see Example 9.1). The modelabel `el` labels the electronic states. For more details of the pyrazine calculations see Ref. [6].

9.2 Defining the primitive basis for a non-adiabatic system

For the treatment of a non-adiabatic system not only the operator but also the input file has to be set up appropriately. One modification concerns the PRIMITIVE-BASIS-SECTION, where an electronic basis has to be specified for the mode labelled `el` in the operator file:

```
PRIMITIVE-BASIS-SECTION
    v10a   HO   22   0.0   1.0   1.0
    v6a    HO   32   0.0   1.0   1.0
    v1     HO   21   0.0   1.0   1.0
    v9a    HO   12   0.0   1.0   1.0
    el     el    2
end-primitive-basis-section
```

The primitive-basis type for the electronic basis is `el` and the number denotes the number σ of states in the system, in this case two. This sets up a discrete (vector) representation for the σ states.

The complete input file for the 4-mode 2-state pyrazine model, from which the above lines were taken, is displayed in Example 9.2.

9.3 Defining the single-particle basis for a non-adiabatic system

In an MCTDH propagation or relaxation a single-particle basis is needed for the representation of the wavefunction. For non-adiabatic systems there are two possible representations, termed single- or multi-set (see Sec. 3.5 of Ref. [1] for details).

```
OP_DEFINE-SECTION
title
  Pyrazine 4-mode model
end-title
end-op_define-section

PARAMETER-SECTION
w10a    = 0.09357, ev
w6a     = 0.0740 , ev
w1      = 0.1273 , ev
w9a     = 0.1568 , ev
delta   = 0.46165, ev
k6a1    =-0.0964 , ev
k6a2    = 0.1194 , ev
k11     = 0.0470 , ev
k12     = 0.2012 , ev
k9a1    = 0.1594 , ev
k9a2    = 0.0484 , ev
lambda  = 0.1825 , ev
end-parameter-section

HAMILTONIAN-SECTION
---------------------------------------------
    modes    | v10a | v6a | v1  | v9a | el
---------------------------------------------
1.0*w10a     | KE   | 1   | 1   | 1   | 1
0.5*w10a     | q^2  | 1   | 1   | 1   | 1
1.0*w6a      | 1    | KE  | 1   | 1   | 1
0.5*w6a      | 1    | q^2 | 1   | 1   | 1
1.0*w1       | 1    | 1   | KE  | 1   | 1
0.5*w1       | 1    | 1   | q^2 | 1   | 1
1.0*w9a      | 1    | 1   | 1   | KE  | 1
0.5*w9a      | 1    | 1   | 1   | q^2 | 1
-delta       | 1    | 1   | 1   | 1   | S1&1
delta        | 1    | 1   | 1   | 1   | S2&2
k6a1         | 1    | q   | 1   | 1   | S1&1
k6a2         | 1    | q   | 1   | 1   | S2&2
k11          | 1    | 1   | q   | 1   | S1&1
k12          | 1    | 1   | q   | 1   | S2&2
k9a1         | 1    | 1   | 1   | q   | S1&1
k9a2         | 1    | 1   | 1   | q   | S2&2
lambda       | q    | 1   | 1   | 1   | S1&2
----------------------------------------------
end-hamiltonian-section
end-operator
```

Example 9.1: An operator file for the pyrazine 4-mode 2-state model system.

In the single-set formalism, which is the default, the wavepackets on each surface are represented by the same single-particle function basis. As there is thus only one single-particle basis, the SPF-BASIS-SECTION has the same form as for adiabatic systems, e.g.

```
SPF-BASIS-SECTION
    v10a = 5
    v6a  = 6
    v1   = 4
    v9a  = 4
end-spf-basis-section
```

```
###################################################################
###            pyrazine 4-mode multi-set                        ###
###################################################################

RUN-SECTION
  name = pyr4mode
  propagate
  tfinal=120.0   tout=0.50   tpsi=1.00
  psi  auto=twice  steps  gridpop
end-run-section

OPERATOR-SECTION
  opname = pyrmod
end-operator-section

SPF-BASIS-SECTION
  multi-set
    v10a = 4, 3
    v6a  = 5, 4
    v1   = 3, 3
    v9a  = 3, 3
end-spf-basis-section

PRIMITIVE-BASIS-SECTION
    v10a   HO   22   0.0   1.0   1.0
    v6a    HO   32   0.0   1.0   1.0
    v1     HO   21   0.0   1.0   1.0
    v9a    HO   12   0.0   1.0   1.0
    el     el    2
end-primitive-basis-section

INTEGRATOR-SECTION
  CMF/var = 0.5 ,  1.0d-5
  BS/spf  =   7 ,  1.0d-5 ,  2.5d-4
  SIL/A   =   5 ,  1.0d-5
end-integrator-section

INIT_WF-SECTION
  build
    init_state = 2
    v10a  HO   0.0   0.0   1.0
    v6a   HO   0.0   0.0   1.0
    v1    HO   0.0   0.0   1.0
    v9a   HO   0.0   0.0   1.0
  end-build
end-init_wf-section

end-input
```

Example 9.2: An input file for the pyrazine 4-mode 2-state model system.

Note that no single-particle basis needs to be specified for the electronic “degree of freedom”, as this is a complete basis set.

In the multi-set formalism, which is often more efficient than the single-set formalism, the wavepackets on each surface are represented in a different single-particle function basis. The number of functions desired for each state must therefore be given. The SPF-BASIS-SECTION may read

```
SPF-BASIS-SECTION
multi-set
    v10a = 4, 3
    v6a  = 5, 4
    v1   = 3, 3
    v9a  = 3, 3
end-spf-basis-section
```

The keyword `multi-set` selects the multi-set formalism. For instance, the line

```
    v10a = 4, 3
```

requests four functions to be used for the wavepacket in the lower state 1, and three functions for the wavepacket in the upper state 2. The multi-set formalism usually requires fewer single-particle functions (per state) than the single-set formalism. This makes the former more efficient in most cases.

9.4 Building the initial wavepacket for a non-adiabatic system

For a non-adiabatic system with σ electronic states also σ initial wavefunctions have to be built (unless they are read from file). When generating the initial wavepacket the MCTDH program however assumes that only one electronic state is initially populated and hence sets all wavefunctions on other states to zero.

The initial wavefunction can therefore be defined in the same way as for adiabatic systems. The program has solely be supplied with the information which state is to be populated at the beginning. This is achieved using the `init_state` keyword in the INIT_WF-SECTION, as shown in Example 9.2. If this keyword is missing, state 1 is initially populated.

When a `multi-set` wavefunction is to be read from file it is convenient to use a `Read-Inwf` block, because then the wavefunction read and the system wavefunction do not need to have the same number of electronic states. See Section 7.7.

Chapter 10

Treating bosonic systems

While MCTDH is designed for distinguishable particles, it also allows for the treatment of *indistinguishable* particles. The 'only' conceptual complication that needs to be taken care of is the permutation symmetry encoded in the general rules of quantum mechanics: More precisely, Ψ is a valid wave function only if for any permutation P_{ij} of two particles we have $P_{ij}\Psi = \pm\Psi$. The $+$ sign holds for *bosonic* particles, which are the subject of this chapter; for *fermions* $(-)$ there already exist specially modified versions of MCTDH. [1] The consequence is that bosons live only in the symmetry-restricted Hilbert space

$$\mathbb{H}_+ = \{\Psi \mid P_{ij}\Psi = \Psi \quad \forall i,j\} \subset \mathbb{H}.$$

Obviously, the most elegant way to extend the MCTDH ansatz

$$\Psi(Q,t) = \sum_J A_J(t)\Phi_J(Q,t) \tag{10.1}$$

would be to include only basis function $\Phi_J \in \mathbb{H}_+$, a demand clearly not met by the Hartree products employed in MCTDH. However, it is possible to circumvent this by simply projecting any wave function onto $\mathbb{H}_+$. This amounts to keeping the expansion coefficients A_J symmetric rather than the basis vectors themselves. In fact, any wave function will usually stay permutation symmetric under (real or imaginary) time evolution if both the initial state and the Hamiltonian are chosen as outlined below and if numerical errors are kept at bay.

10.1 Setting up the Hamiltonian

The Hamiltonian for identical bosonic atoms should of course be symmetric in all particles. For bosonic atoms with no more than binary interactions, it usually has the form

$$H = \sum_i h(p_i, x_i) + \sum_{i<j} V(x_i - x_j),$$

so the one-particle operator h (including kinetic energy) has to be listed in the operator file for any boson i, while the interaction potential V requires a manual entry for any combination

[1] For simplicity, we assume a system of spin-polarized one-dimensional bosons. The extension to higher dimensions is slightly more complicated insofar as one has to distinguish between particles ($i = 1, \ldots, N$) and degrees of freedom ($\varkappa = 1, \ldots, f$). To match these two, different modes $\varkappa$ belonging to one and the same physical particle #i have to be combined, cf. Sec. 5.3.

$i < j$. (Altogether, these are $N(N-1)/2$ terms, which naturally limits the application to few particles.)

As an example, the operator file for $N = 3$ one-dimensional bosons in a harmonic trap is shown in Example 10.1 (for more details, the reader is referred to [21]). In that case, $h(p, x) = \frac{1}{2}p^2 + \frac{1}{2}x^2$; the two-body potential $V(x) = g\delta_\sigma(x)$, shaped as a normalized Gaussian of width σ, has to be fitted to the direct-product form imposed by MCTDH. This is carried out as usual via **potfit** (set there e. g. `pes = gauss1d{width=0.05}`), the resulting natpot is included in the LABELS-SECTION. The numerical parameters are defined in the PARAMETER-SECTION, even though some of the values are conveniently reset in the input file (see below).

10.2 Modifying the input

The symmetrization mentioned above brings about some minor modifications of the way the wave function Ψ is handled. The MCTDH ansatz (10.1) is now simplified insofar as the *single-particle functions* are now identical, i.e., we have $\Phi_J \equiv \varphi_{j_1} \otimes \cdots \otimes \varphi_{j_N}$ with a single set of functions $\{\varphi_j \mid j \leq n\}$.

This reflects in the input file as illustrated in Example 10.2: The SPF-BASIS-SECTION only gives the first orbital φ_1, while all others are mapped via the entry `x2 = id,1`, etc.

It goes without saying that, by extension, the *primitive basis* also has to be identical for every boson. Again this is reflected in the input file, where the PRIMITIVE-BASIS-SECTION contains repetitions of the very same line

```
x1      HO   125  xi-xf  -4.0  4.0
```

for any x_i.

A little less trivial is the choice of the *initial wave function* Ψ_0. As stated above, it must be permutation symmetric, a demand which can be met by the following standard choices.

- A Hartree state $\Psi_0 = \varphi^{\otimes N}$ is implemented trivially by including the following lines in the INIT_WF-SECTION:

  ```
  build
    x1  eigenf   spo
    x2  map x1
    x3  map x1
  end-build
  ```

  Here `eigenf spo` specially selects the (lowest) eigenfunction φ of the single-particle operator `spo` previously defined in the operator file. This particular feature is not essential—in this case, one may as well use harmonic-oscillator functions (`HO`) instead—but we have included it simply to give a realistic example.

- More generally, a number (or Fock) state $|n_1, n_2, \ldots\rangle$ can be selected, which denotes how many particles $n_a \in \mathbb{N}$ occupy a given single-particle mode φ_a (where $\sum_a n_a = N$). Cast in standard MCTDH form, this a sum over all permutations of the single configuration

  $$J = (\underbrace{j_1, \ldots, j_1}_{n_1 \times}, \ldots, \underbrace{j_n, \ldots, j_n}_{n_n \times}).$$

  This is a little more cumbersome, since we have to keep track of all permutations of J. A typical statement from the INIT_WF-SECTION now reads:

```
build
  x1  eigenf   spo
  x2  map x1
  x3  map x1
end-build

A-coeff
  2 2 3 (1.0,0.0)
end-A-coeff

symcoeff
```

Note that the last block is the same as for the Hartree product above, telling MCTDH to select eigenstates φ_a of the single-particle Hamiltonian `spo`. On top of that, the block `A-coeff` defines just which orbitals a should be selected. In our example, this produces a single configuration $J = (2, 2, 3)$ (forget about normalization.) To make this permutation symmetric —i.e., a number state $|n_1 = 0, n_2 = 2, n_3 = 1\rangle$— the statement `symcoeff` has been added.

- If one performs a series of relaxations, e.g. with increasing interaction parameter g, then the most convenient choice for the initial state is one already obtained in some previous MCTDH calculation. This is done by simply casting the INIT_WF-SECTION as follows:

```
file= p3d1_19
symcoeff
```

This ensures that the restart file from the directory p2d1_19 is read as initial wave function. The `symcoeff` statement is added just to be one the safe side and make sure that the initial state is absolutely symmetric.

The input for potfit is very simple. One usually uses as many natpot terms as grid points. The fit is therefore exact.

```
RUN-SECTION
name = ngaussHOfit_N125
end-run-section

OPERATOR-SECTION
pes = gauss1d{width=0.05}
end-operator-section

PRIMITIVE-BASIS-SECTION
x1     HO  125  xi-xf  -4.0  4.0
x2     HO  125  xi-xf  -4.0  4.0
end-primitive-basis-section

NATPOT-BASIS-SECTION
x1 = contr
x2 = 125
end-natpot-basis-section

end-input
```

```
#####################################################################
###  3 bosonic particles, 1D, in a harmonic trap                  ###
#####################################################################

OP_DEFINE-SECTION
title
  p3d1_DW, 3 one-dimensional bosons in a harmonic trap
end-title
end-op_define-section

PARAMETER-SECTION
mass_x1 = 1.0
mass_x2 = 1.0
mass_x3 = 1.0
g       = 1.0 # int. strength    | only dummy value, reset in INP file!
end-parameter-section

HAMILTONIAN-SECTION
---------------------------------
    modes     | x1  | x2  | x3
---------------------------------
# Kinetic energy
 1.0          | KE  | 1   | 1
 1.0          | 1   | KE  | 1
 1.0          | 1   | 1   | KE

# Harmonic trap
 0.5          | q^2 | 1   | 1
 0.5          | 1   | q^2 | 1
 0.5          | 1   | 1   | q^2

# Two-particle interaction // N(N-1)/2 entries
 g            |1&2 vv
 g            |1&3 vv
 g            |2&3 vv
---------------------------------
end-hamiltonian-section

HAMILTONIAN-SECTION_spo
-----------------------
    modes     | x1
-----------------------
# Kinetic energy
 1.0          | KE
# Harmonic trap
 0.5          | q^2
-----------------------
end-hamiltonian-section

LABELS-SECTION
  vv = natpot{ngaussHOfit_N125 ignore} # pot-fitted interaction potential
end-labels-section  # 'ignore' is set to ignore the modelabels of the fit

end-operator
```

Example 10.1: An operator file for $N = 3$ one-dimensional bosons in a harmonic trap.

```
########################################################################
###  3 bosonic particles, 1D, in a harmonic trap                    ###
########################################################################

RUN-SECTION
name = p3d1_20
energy-not-ev  time-not-fs   # A dimensionless model is treated
relaxation = 0  rlxunit=au
tfinal = 20.0   tout = all  tpsi = 1.0
gridpop  steps  cross  orben
title = 3 particles, 1D in double-well trap (p3d1_DW)
end-run-section

OPERATOR-SECTION
opname = p3d1_DW
alter-parameters
  g = 10.
end-alter-parameters
end-operator-section

SPF-BASIS-SECTION
x1 = 15
x2 = id,1
x3 = id,1
end-spf-basis-section

PRIMITIVE-BASIS-SECTION
x1     HO   125  xi-xf  -4.0  4.0
x2     HO   125  xi-xf  -4.0  4.0
x3     HO   125  xi-xf  -4.0  4.0
end-primitive-basis-section

INTEGRATOR-SECTION
CMF/varphi = 0.2, 1.0d-2
RK8/spf    = 1.0d-8 , 0.1
RRDAV/A    = 200 , 1.0d-9
end-integrator-section

INIT_WF-SECTION
file=p3d1_19
symcoeff
end-init_wf-section

end-input
```

Example 10.2: An input file for $N = 3$ one-dimensional bosons in a harmonic trap.

Chapter 11

Analysing the results employing the Analyse programs

The set of Analyse programs can be used to analyse the information from a calculation, which is stored in the various data files. For a complete list of programs, see the HTML documentation. If a program is started using the `-h` option, i.e.

```
analyse -h
```

where *analyse* is the name of the program, e.g. **rdgpop**, a brief description of how to use the program, and a list of options will appear.

The programs are designed to be used together with the GNUPLOT program. In many cases, the option `-g` will produce a file complete with GNUPLOT commands, ready for immediate plotting. Some programs also support interactive plotting in conjunction with GNUPLOT. In these cases, starting the program with

```
analyse -inter
```

brings up a menu with options to choose what is to be plot, to change plotting boundaries, etc.

Here, a brief overview of only the most important programs will be given. As an example, we take the results from the wavepacket propagation of the NOCl system. First the system is relaxed on the S_0 surface using the input file inputs/nocl0.inp. Propagation is then made using the file inputs/nocl1.inp. The data files containing all the information about the calculation and the system evolution are then contained in the directory nocl1. This is the system used in the first tutorial.

11.1 The Analysis Interface

The **analysis** program provides a menu driven interface for running many of the **ANALYSE** programs. On typing

```
analysis
```

a menu appears, as shown in Example 11.1.

An option is selected by entering the appropriate number. This may lead to further menus which allow the examination or plotting of various quantities of interest.

```
************************************************************************

     THE HEIDELBERG MCTDH PROGRAM ANALYSIS PACKAGE

          Program Version :     8
          Release         :     2

************************************************************************

Present directory is: /workb/graham/mctdh82.0

  0 = stop
  1 = list / change directory
  2 = analyse convergence
  3 = analyse integrator
  4 = analyse results
  5 = analyse system evolution
  6 = analyse potential surface
  7 = compare calculations
```

Example 11.1: The start-up menu in the **analysis** program which provides an interface for running the various **ANALYSE** programs.

A browse function is included to move between directories containing data (option 1). Keep typing the name of the new directory, either absolute or relative names are allowed, until the directory of choice is found. Then type "no" to return to the main menu. This option may also be used to list the contents of the present directory.

If one knows the MCTDH package well, it is more convenient (and faster) to use directly the routines, which are called by the **analysis** interface. But for the beginner, **analysis** can be a big help, as one is guided through the large selection of analyse–tools. Note, however, that there are more tools available than accessible through **analysis**. (See the HTML documentation).

11.2 Interpreting the MCTDH output

During the propagation of a wavepacket, information about the system evolution is output to allow an easy visual check of how the calculation is progressing. If the keyword `output` was included in the RUN-SECTION of the input file, this information is written to the file output in the directory specified by the `name` keyword. If this keyword was omitted, this information is written to the screen during the calculation.

A section of the output from our example NOCl propagation is shown in Example 11.2. After information about the program version used, and where and when the calculation was run, starts the information about the system. This is output every t fs, where `tout` = t is the time specified in the RUN-SECTION of the input file.

At each output step the following information is give:

- Norm: The norm of the wavefunction expansion coefficients $\sqrt{\sum_J A_J^* A_J}$. This should remain close to 1.0, unless a CAP is used in which case the norm will disappear with the wavepacket.

```
******************************************************************************
                              MCTDH version   8
                                 Release     1
                                 Revision    6
******************************************************************************

 ------ Host: "bose" ----------Tue Feb  8 10:58:01 2000

 /usr/people/graham/runs1/nocl1
   NOCl S1  Propagation, (sin,HO,Leg/36,24,60). CAP
-----------------------------------------------------------------------------

 Time  =        .00 fs,       CPU =        .76 s,    Norm    = 1.00000000
 E-tot =   1.188024 eV,    E-corr =   1.029664eV,    Delta-E =      .0000 meV

 Natural weights *1000 :
rd          : 999.2709    .7283    .0008   0.0000   0.0000
rv          : 999.3842    .6153    .0005   0.0000   0.0000
theta       : 998.9065   1.0845    .0088    .0002   0.0000

 Mode expectation values and variances :
rd          : <q>=   4.3143  <dq>=    .0794  <n>=   2.1160  <dn>=   2.2115
rv          : <q>=   2.1549  <dq>=    .0670  <n>=    .0349  <dn>=    .2249
theta       : <q>=   2.2283  <dq>=    .0767  <j>=   4.6297  <dj>=   3.9988
-----------------------------------------------------------------------------

 Time  =       1.00 fs,       CPU =       1.32 s,    Norm    = 1.00000000
 E-tot =   1.188024 eV,    E-corr =   1.023057eV,    Delta-E =     0.0000 meV

 Natural weights *1000 :
rd          : 997.8587   2.1345    .0068    .0001   0.0000
rv          : 997.8951   2.0843    .0201    .0005   0.0000
theta       : 998.5412   1.4346    .0233    .0009   0.0000

 Mode expectation values and variances :
rd          : <q>=   4.3155  <dq>=    .0798  <n>=   2.2337  <dn>=   2.3028
rv          : <q>=   2.1578  <dq>=    .0668  <n>=    .0588  <dn>=    .2809
theta       : <q>=   2.2276  <dq>=    .0767  <j>=   4.8200  <dj>=   4.1250


   :
   :
   :
   :

Propagation was successful.

Total time     [h:m:s] :    0 :  0 : 12.1

 ---------------------------------------------------------------------------
 ------ Host: "bose" ----------Tue Feb  8 10:58:13 2000

 /usr/people/graham/runs1/nocl1
   NOCl S1  Propagation, (sin,HO,Leg/36,24,60). CAP
```

Example 11.2: A section of an output file from the wavepacket propagation on the S1 surface of NOCl.

- E-tot: The total energy, i.e. expectation value of the Hamiltonian. This should remain constant, unless a CAP is used when the energy will dissappear with the wavepacket.
- E-corr: The correlated energy, i.e. the expectation value of the terms in the Hamiltonian which correlate the degrees of freedom. (The correlated and uncorrelated Hamiltonian terms are listed in the op.log file).

- Delta-E: The loss in energy during the calculation, i.e. difference between the energy at time t and at time 0.
- Natural weights: The natural weights, i.e. eigenvalues of the one-dimensional density matrices, are given for each mode in the calculation. A weight of 1.0000 given here indicates that the least important natural orbital is present in 0.1% of the wavefunction.
- Mode expectation values: $\langle q\rangle$ and $\langle dq\rangle$ are the expectation values and variances for the position operator, where $\langle dq\rangle = \sqrt{\langle q^2\rangle - \langle q\rangle^2}$. This gives an idea of the spread of the wavepacket and a check on the grid used. If a DVR is used, $\langle n\rangle$ and $\langle dn\rangle$ are the expectation values and variances of the number representation in the corresponding FBR basis, a measure of which functions are populated. If a Legendre DVR is used (i. e. Leg, Leg/R or KLeg) the symbol is changed to $\langle j\rangle$ and $\langle dj\rangle$ to indicate, that the number representation is in this case just the expectation and variance of the angular momentum. If an FFT basis or the exponential-DVR is used, $\langle p\rangle$ and $\langle dp\rangle$, the expectation value and variance of the momentum operator are given.

At the end of the file should be written:

```
Propagation was successful.
```

and the CPU-time used for the calculation.

11.3 Checking the accuracy of a calculation

The accuracy of an MCTDH calculation depends on both the size of the primitive and the single-particle function bases. Analyse programs are available for both these tasks. Note that one has to be in the name directory when applying the analyse routines and scripts as shown below.

11.3.1 Checking the primitive basis size

The program **rdgpop** reads and evaluates the populations of the primitive basis functions, e.g. the grid points. This is used to check that enough primitive basis functions have been used for the calculation. The program requires the gridpop file, which is obtained by specifying the `gridpop` keyword in the RUN-SECTION of the input file.

The program can be used either to calculate the maximum population, or to evaluate the change of population with time of the points at the ends of the grid.

In the directory containing the data files, typing

```
rdgpop
```

results in some information about the calculation and the primitive basis used for each degree of freedom in the calculation. The question

```
 Number of grid-points to be summed over: nz =?
```

then appears. If 1 is input, then the populations output are those on the end grid points. If 2 is input, the output population for the beginning of the grid is the sum of the populations of the first and second points, while the output population for the end of the grid is the sum of the populations of the last and last-but-one points. And so on.

The next question asked is

```
Runing-number of degree of freedom: dof =?
dof = 0 -> Print only maximum over time
```

Selecting 0 here results in the maximum population at the end grid points being displayed.

```
-------------------------------------------------------------------------------

 Maximal values (all times);  final time:    30.00 fs
   #  dof     grid(begin)    grid(end)      basis(begin)   basis(end)
   1  rd       .000245781     .000000064     .260884702     .000002414
   2  rv      0.000000000     .003499307     .972981513     .000017179
   3  theta   0.000000000    0.000000000     .150721535     .000005773
-------------------------------------------------------------------------------
```

We see that the beginning of the `rv` grid is unpopulated, and this grid point could be removed without affecting the propagation quality. Likewise, the ends of the `theta` grid are unpopulated.

To avoid answering the questions, this result could be obtained by calling the program as

```
rdgpop 1 0
```

If the number of grid points to be summed over is changed to 2, i.e.

```
rdgpop 2 0
```

then the output is

```
-------------------------------------------------------------------------------

 Maximal values (all times);  final time:    30.00 fs
   #  dof     grid(begin)    grid(end)      basis(begin)   basis(end)
   1  rd       .001164114     .000002005     .593797892     .000013545
   2  rv      0.000000000     .010178351     .993075095     .000088905
   3  theta   0.000000000    0.000000000     .316521645     .000008258
-------------------------------------------------------------------------------
```

The start of the `rv` grid is still unpopulated, and so 2 grid points could be removed at the start of this degree of freedom. By increasing the number of points over which the sum is made, it is possible to evaluate how many grid points can be removed. Whether it is possible to remove the grid points depends on the primitive basis being used. Thus the Leg-DVR used for the `theta` degree of freedom by definition runs from π to 0 (if symmetry is used `theta` runs from $\pi/2$ to 0), and these points cannot be removed. (However, the use of the restricted Legendre DVR, `Leg/R`, allows to remove unused angular grid-points).

To chart the population of the end grid points, select a degree of freedom. Thus

```
rdgpop 1 1
```

produces the file gpop.pl, which contains the populations of the end of grid points as a function of time. The option `-g` writes information for GNUPLOT to this file, and so

```
rdgpop -g 1 1
gnuplot -persist gpop.pl
```

produces a plot of the time-dependence of the end of grid points. A more convenient way to plot the time evolution of the population of the end grid points is provided by the **plgpop** script.

```
plgpop 1 1
```

11.3.2 Checking the single-particle function basis size

The quality of the single-particle function basis is reflected in the populations of the natural orbitals (see Sec. 3.3 in Ref. [1]). If the calculation contains natural orbitals with a low population, these are not significant for the representation of the wavefunction, and the calculation is of a reasonable quality. Unfortunately, different properties have different convergence criteria, and it is not possible to give absolute figures for when the natural orbitals are insignificant. As a general rule of thumb, when the population of the highest (least populated) natural orbital is below 1 % (i.e. a population below 0.01), the calculations will be reasonable, although convergence may be a way off.

Experience has shown that it is important that the single-particle function bases for all the modes are balanced, i.e. the lowest natural orbital populations are similar for all. There is little point spending effort on converging the single-particle function basis for one mode when the dynamics can be seriously affected by the poor representation of another mode.

The program **rdcheck** is used to check the natural orbital populations, and so show where more functions are required. Two basic pieces of information are required by the program: a state and a mode for analysis. If no arguments are given, the program prompts for what it requires. The basic information is provided by typing

```
rdcheck  0  0
```

when in the directory containing the data files from a calculation. The arguments 0 0 select no particular state or mode. The program then prints some information about the system, and most importantly the maximum population of the highest natural orbital (lowest natural weight) is displayed.

```
---------------------------------------------------------------------------

 Maximum over time of lowest nat.-weight;   final time :    30.00 fs
 mode     s = 1
   1   1.810E-03
   2   2.472E-04
   3   9.870E-05
---------------------------------------------------------------------------
```

This information says that the calculation should be of reasonable quality, as all mode contain natural orbitals that remain fairly insignificant.

If a mode is selected, the populations as a function of time can also be graphically displayed. The modes are numbered in the order in which they are listed in the SPF-BASIS-SECTION of the input file. For NOCl the order corresponds to the degrees of freedom `rd, rv, theta`, and so 2 selects the vibrational mode rv. The NOCl system has only one state, and so

```
rdcheck 1  2
```

produces a file, nat.pl which contains the natural populations as a function of time. The GNUPLOT program can be conveniently used by including GNUPLOT data in this file, i.e.

```
rdcheck -g 1  2
gnuplot -persist nat.pl
```

would produce the plot shown in Fig. 11.1. Again, a more convenient way to produce this plot is provided by a pl-script. Just type:

```
plnat 1 2
```

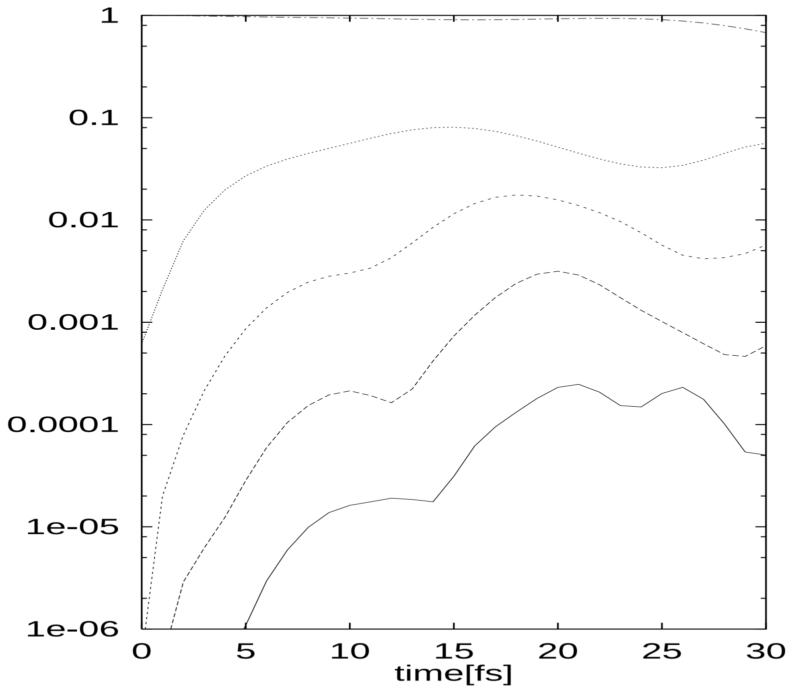

Figure 11.1: The natural orbital populations for the single-particle function basis for the vibrational degree of freedom as a function of time for the photo-dissociation of NOCl.

11.4 Checking the efficiency of a calculation

The timing file, which is obtained by adding the keyword `timing` to the RUN-SECTION, contains information about how much time is spent in the various sections and subroutines of the program. This information can be used to improve the efficiency of a calculation. For instance, if in a CMF run the BS-integrator (used to propagate the single-particle functions) takes less than one or two percent of the total effort, one should combine more single-particle functions. If, on the other hand, the BS-integrator takes more than 80% of the total effort, one should remove some of the combinations. If the propagation of one certain mode takes much longer than the propagation of the other modes although the (combined) grid sizes are comparable, then check whether the (DVR) representation is appropriate. The information listed in the timing file can be extremely helpful. It is a good practice to always include the timing file.

11.5 Watching the system's evolution

The program **showd1d** is designed to plot the evolution of the system density along a degree of freedom. It reads the gridpop file, which can be created by adding the keyword `gridpop` in the RUN-SECTION of the input file. This file contains the one-dimensional densities

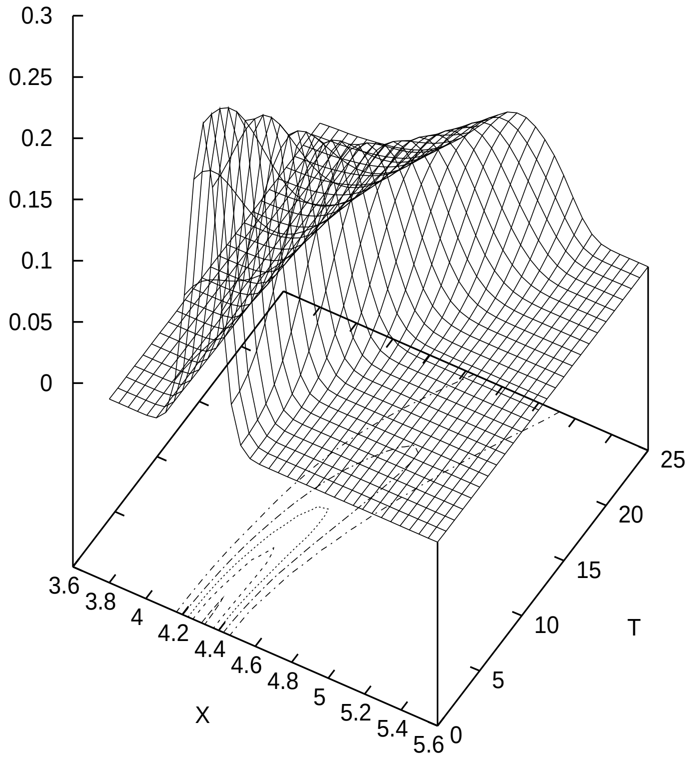

Figure 11.2: The density along the dissociative degree of freedom as a function of time for the photo-dissociation of NOCl.

output at intervals specified by the `tout` keyword. This density can be plotted using the **showd1d** program in conjunction with GNUPLOT.

For example, the NOCl photo-dissociation calculation has been run using the example input file 4.1. The command

```
showd1d -a -T f1
```

requests that the density for the first degree of freedom (as listed in the PRIMITIVE-BASIS-SECTION of the input file, i.e. the dissociative mode, `rd`) is written to a file den1d_f1. The options make this file into a GNUPLOT grid file, complete with commands. The option `-a` (automatic) lets **showd1d** call GNUPLOT. The 3D plot of density is shown in Fig. 11.2. Try also the other format options. A complete list of options is obtained trough the command **showd1d -h**.

The similar program is **showspf** which displays the single-particle functions (uncombined modes only, of course). Note that **showd1d** reads the gridpop file while **showspf** reads the psi file. Finally, there is **showrst** which plots the single-particle functions of the restart file.

11.6 Determining photo-dissociation and photo-absorption spectra

The spectrum of a system can be generated using **autospec**. This reads the autocorrelation function from the file auto, which is created if the keyword `auto` is included in the RUN-SECTION of the input file. The spectrum is then created over a chosen interval in energy space by Fourier transform. The shell script **plspec** calls **autospec** and then GNUPLOT. It is often more convenient to use this script.

For example, the NOCl spectrum can be displayed by typing

```
plspec  0.6  2.0 ev
```

This spectrum was produced in the tutorial, and is shown in Fig. 2.1.

Before the autocorrelation function is Fourier transformed, it gets modified. To reduce artifacts of the Gibs phenomenon, the autocorrelation function is multiplied with $cos^n(\pi t/2T)$, where $n = 0, 1, 2$ and where T denotes the final time (plus one time step) of the autocorrelation function. Due to the $t/2$–trick, see Eq. (167) of the MCTDH review, the propagation time is only $T/2$. The columns 2–4 of the spectrum.pl file list the results for the different n's. To introduce a damping, i. e. a Lorentzian or Gaussian broadening, the autocorrelation function is further multiplied with $exp(-(|t|/\tau)^i)$ where τ and $i = 1, 2$ are the two last arguments of **autospec**. The exponential is ignored if $\tau = 0$.

Finally the Fourier transform is multiplied with the photon energy ω to arrive at an absorption spectrum. This multiplication is omitted if the option `-FT` is given. Note that only in this case one may use an energy interval containing negative energies (compare with Fig. 2.4).

11.7 Computing excitation and reaction probabilities

The analysis of scattering processes, i. e. the computation of excitation and reaction probabilities, is performed by evaluating the quantum flux going into a particular channel. The quantum flux is determined by the interaction of the time-dependent wavepacket with a CAP. The program **flux** performs the necessary analysis. The psi file is read and the energy resolved flux is computed and written to the flux file. Additionally this flux is divided by the energy distribution of the initial wavepacket to obtain the transition or reaction probabilities. The latter step requires in general that the QUANTICS program has computed the energy distribution of the initial wavepacket and has written it to the enerd file (see Section 7.11). (NB: The file enerd is called adwkb in older versions). Besides the flux file, **flux** creates the files flux.log, gtau and wtt. When the file gtau is present the flux program skips the time consuming evaluation of the integrals $\langle\Psi(t)|W|\Psi(t+\tau)\rangle$ but reads the function $g(\tau)$ from the gtau file. (See the MCTDH review, section 8.6.3, Eq.(199), for more details). This allows to re–do the Fourier integrals for another energy interval very quickly. The option `-w` enforces the re–calculation of $g(\tau)$.

The file wtt contains the expectation values $W_{tt} = \langle\Psi(t)|W|\Psi(t)\rangle$. This information is useful for checking that the absorption process has finished. The shell script **plwtt** visualises the function W_{tt}, while **plflux** and **plflux -r** visualise the flux and the transition probability, respectively.

The program **flux** cannot only determine the total energy resolved flux going into a particular arrangement channel (i. e. going into a particular CAP) but can also determine the flux which is projected onto final quantum states or which is weighted by an operator. This is probably best demonstrated by an example. Copy the file $MCTDH_DIR/operators/nocl1.op to your tutorial directory and add the following lines to this operator file.

```
HAMILTONIAN-SECTION_vib
usediag
--------------------------------------
modes      |  rd    |  rv    | theta
--------------------------------------
1.0        |  1     |  KE    |   1
1.0        |  1     |  v:NO  |   1
--------------------------------------
end-hamiltonian-section
```

Then edit the input file nocl1.inp and set the propagation time to `tfinal=60` and re-run. Since **flux** analyses the wavepacket as it is absorbed by the CAP, a longer propagation time is required as for converging the spectrum. Then execute the commands

```
flux -w -s 19 -lo 12  0.61 2.0 ev rd
mv flux flux.0
flux -w -s 19 -lo 12 -O vib -u 200. 0.61 2.0 ev rd
mv flux flux.op
flux -ed flux.0 -s 19 -lo 12 -O vib -u ev 0.61 2.0 ev rd
mv flux flux.op_r
flux -w -s 19 -lo 12  -P 2 eigenf vib 1 % 0.61 2.0 ev rd
mv flux flux.1
flux -w -s 19 -lo 12  -P 2 eigenf vib 2 % 0.61 2.0 ev rd
mv flux flux.2
flux -w -s 19 -lo 12  -P 2 eigenf vib 3 % 0.61 2.0 ev rd
mv flux flux.3
autospec  -FT  0.61 2.0 ev 0 1
```

The first flux-run evaluated the total flux. To display it, type

```
plflux -f flux.0
```

As you will notice, the plot looks very similar to the absorption spectrum shown in figure 2.1. It is not identical, though, as the definition of an absorption spectrum contains a factor ω, the energy of the absorbed photon. This multiplication is omitted when **autospec** (or **plspec**) is run with the `-FT` option, as we have done above. Now the spectra are identical as one observes when typing

```
plflux -G -f flux.0 -d spectrum.pl
```

This shows the flux and the Fourier transform of the autocorrelation function on top of each other.

Next we modify the flux by letting the operator of the vibrational energy act on it. Type

```
plflux -G -f flux.0 -d flux.op
```

and you will see the total flux in comparison with the vibrational energy weighted flux. It is now clear that the structures in the spectrum are due to vibrational excitation of the NO fragment. By the way, via the option `-u 200` the modified flux was multiplied by the factor 200. This was done to make it comparable with the total flux. Usually the option `-u`

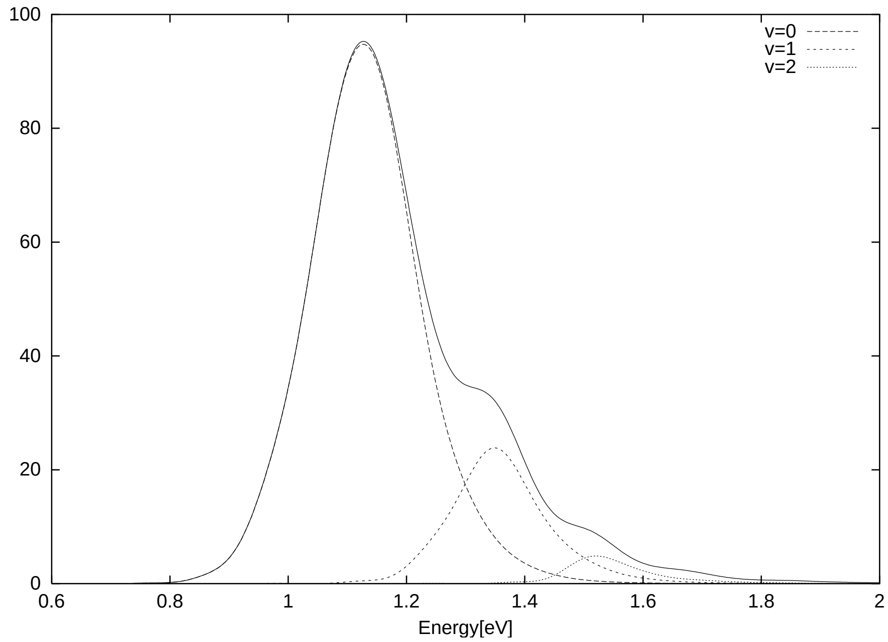

Figure 11.3: The total (full line) and projected flux of dissociating NOCl. The projection is on the vibrational states of the NO fragment, v=0, v=1, and v=2, respectively.

is followed by an energy keyword, e.g. `-u ev`, to transform the weighted flux from `a.u.` to a desired energy unit.

One may divide the weighted flux by the total flux to observe the vibrational energy content. The option `-ed flux.0` was used to input the file flux.0 as energy distribution (the default is the file enerd, the generation of which, however, is only useful for a scattering – not half-scattering – problems). To visualise the quotient weighted-flux/total-flux, type

```
plflux -G -r -f flux.op_r
```

The structures below 0.8 eV are numerical noise because one divides a very small number by another very small number. Obviously, the higher the energy of the absorbed photon, the larger is the vibrational energy of the NO fragment. Compare this plot with the eigenenergies displayed in the flux.log file.

Finally we demonstrate the use of projectors. Type

```
plflux -G -f flux.0 -d flux.1 -e flux.2
plflux -G -a 1.0 -y 25. -f flux.0 -d flux.2 -e flux.3
```

The first plot shows the flux projected onto the vibrational ground state and the first excited state, respectively, in comparison with the total flux. The second plot is similar, but shows the flux projected onto the first and second excited state, respectively. The Fig. 11.3 displays the total and projected projected flux, similarly to the plots generated above.

11.8 Monitoring state populations of non-adiabatic systems

11.8.1 Diabatic populations

The (electronic) state populations of a `multi-set` run are written to the check file and are read by **rdcheck** which writes them to the file chk.pl. One then may use GNUPLOT to plot the state populations, but it is easier to use **plstate** :

```
plstate
```

Note that **plstate** may also be used for runs evolving on one single potential energy surface. In this case the norm-squared of the wavefunction is plotted. In the presence of CAP's this quantity is non constant.

In case of a multi-state calculation which uses the `single-set` formulation, the state populations presently have to be read from the output file. Use the command:

```
fgrep population output | sed 's/population : //' > file
```

to write the state populations to the file file.

11.8.2 Adiabatic populations computed with adpop

Because QUANTICS usually works in the diabatic representation the diabatic populations can be easily calculated (see above). But if one is interested in the adiabatic state populations, these are more difficult to obtain. There are two possible ways.

The first possibility is to use the analyse program **adpop**. This program reads the psi file of a QUANTICS- run and the so-called pes file, which is a special operator file in which all derivative operators are ignored. The pes file is conveniently generated by setting the option `-pes`, i. e. running "**quantics** `-pes` inpfile". Moving to the name directory, one may then calculate the diabatic and adiabatic state populations by running **adpop**. If needed, one- and two-dimensional adiabatic densities can be calculated by adding the modelabels to the program call:

```
adpop v9a v10a,v6a
```

In this case the one dimensional adiabatic density for the `v9a` degree of freedom and the two dimensional density for the `v10a`-`v6a` degrees of freedom will be created as well as the diabatic and adiabatic state populations. The created files are: adp (for the populations), adp_v9a (for the one dimensional density), adp_v10a_v6a (for the two dimensional density) and the log file adp.log. To create two-dimensional densities, the two modelabes must be separated by a comma. The modelabels for one-dimensional densities stand alone. To visualise the density files created by **adpop** use the shell script **pladpop** that can plot one-and two-dimensional densities. For plotting the state populations simply use **plgen**.

As the calculation runs over the full primitive grid, the calculation is slow. Analyzing one wavefunction takes about [grid-dimension]*[A-vector length]*1.5 $* 10^{-8}$ s on a 3 GHz P4. The adpop calculations can be accelerated by setting the options `-q` or `-mc`. The first option enables the *quick modus* where all points are ignored for which the product of the one-particle grip-populations (i. e. the 1D-densities) are smaller than some threshold. The loss in accuracy is usually negligible (and can be controlled by setting the parameter `qtol`). The Monte-Carlo integration is, of course, less accurate, but allows to attack problems which are not feasible for adpop using direct integration.

11.8.3 Adiabatic populations computed with adproj

If one is interested in the adiabatic state populations alone one should use the second variant, which is more efficient, especially for larger systems (≥5 degrees of freedom). However, this way is more cumbersome than the **adpop** calculation. The procedure is explained in the following.

First a pes file must be created, often this has already been done by the previous **quantics**-run. After this, running **adproj** generates several vpot files which contain the adiabatic surfaces and the projection operator matrix elements for each electronic state. As an example the file apr_p1_12 contains the matrix element (1,2) of the projector (p = projector) for the first adiabatic electronic state. In contrast, the file apr_v2 contains the second adiabatic potential energy surface (v = potential). Then the vpot files must be fitted by **potfit** to bring them into the MCTDH product representation. To calculate the expectation value of the projector one needs the projector in form of an MCTDH oper file. This oper file can be created by an **quantics**-run with the keyword `genoper` in the RUN-SECTION. After the oper file has been created, the expectation value of the projector can easily be calculated with the **expect** analyse routine. This requires that the wavefunction has been stored (psi file). Alternatively one may compute the adiabatic populations *on the fly* by setting the keyword `expectation` in the RUN-SECTION of the **quantics** propagation.

```
RUN-SECTION
name = projector111
readvpot = apr_p1_11    # path of the vpot file
direct = 1
end-run-section

OPERATOR-SECTION
pes = none              # no PES from library needed
end-operator-section    # as the PES is read from vpot

PRIMITIVE-BASIS-SECTION
    v10a   HO     20   0.0     1.0     1.0
    v6a    HO     30   0.0     1.0     1.0
    v1     HO     20   0.0     1.0     1.0
end-primitive-basis-section

NATPOT-BASIS-SECTION
v10a = 10
v6a  = contr
v1   = 10
end-natpot-basis-section

end-input
```

Example 11.3: The input file, projfit.inp, for the **potfit** calculation.

Here's a list of all steps needed to compute the adiabatic populations:

1. Create an pes file by an **quantics**-`genpes`-run.
2. Create vpot files with **adproj** as explained above. (Just run **adproj** in the name directory of step 1. **adproj** will read the pes file.)
3. Fit the vpot files of the projector with **potfit** to create natpot files. As **potfit** requires an input file, an example is shown in Example (11.3). Here **potfit** will read the vpot

file apr_p1_11 and will create the natpot file in the name directory projector111. In the OPERATOR-SECTION the keyword pes = none must be given. Note that there must not appear an electronic degree of freedom in the PRIMITIVE-BASIS-SECTION. A similar input file must be generated for each matrix element of the projector.

4. Create a projection operator file by performing an **quantics**-genoper-run using the just created natpot files. For this **quantics**-run an input and an operator file must be written. As an example, these files for the creation of a projection operator are shown in Example (11.4) and (11.5). In the operator file, projector1.op, one uses the natpot files which are in the directories projector111, projector112 and projector122. Note that the SPF- and PRIMITIVE-BASIS-SECTIONs must be identical to the ones used in the **quantics**-propagation-run where the psi-file was created that will be used for the calcualtion of the adiabatic populations.

5. Calculate the expectation value of the projection operator with the analyse routine **expect**. This requires the wave function that is contained in the psi file. Alternatively the adiabatic populations can be computed *on the fly* by setting the expectation keyword in the RUN-SECTION. In this case it is not necessary to store the wavefunction.

Notice that the **adproj** run provides also the vpot files of the adiabatic surfaces. This makes it possible to perform propagation in the adiabatic approximation.

```
RUN-SECTION
name = proj1 genoper
title =  pyrazine 3D projector.
end-run-section

OPERATOR-SECTION
opname = projector1
end-operator-section

SPF-BASIS-SECTION
multi-set
      v10a =    7, 7
      v6a  =   10, 10
      v1   =    6, 5
end-spf-basis-section

PRIMITIVE-BASIS-SECTION
    v10a   HO     20   0.0     1.0     1.0
    v6a    HO     30   0.0     1.0     1.0
    v1     HO     20   0.0     1.0     1.0
    el     el      2
end-primitive-basis-section

end-input
```

Example 11.4: The input file, proj1.inp, for the **mctdh**-genoper-run.

Typical calculation times for the **adpop**-run are about 20s (for 3D) up to 9h (for 6D). Each calculation was performed on a 2.6GHz processor and the psi files contained 121 timesteps

```
OP_DEFINE-SECTION
title
projection-operator for 3D pyrazine
end-title
end-op_define-section

LABELS-SECTION
P11 = natpot{projector111}
P12 = natpot{projector112}
P22 = natpot{projector122}
end-labels-section

HAMILTONIAN-SECTION_projector1
usediag
------------------------------------------
    modes        |  el   | v10a | v6a | v1
------------------------------------------
1.0              | S1&1  | P11
1.0              | S2&2  | P22
1.0              | S1&2  | P12
------------------------------------------
end-hamiltonian-section

end-operator
```

Example 11.5: The operator file, projection1.op, for the **mctdh**-`genoper`-run.

(3D and 6D pyrazine models). In the case of using **adproj** it takes about 10s (3D) up to 5min (6D) for the same Hamiltonians and wavefunctions. Note that in the **potfit** step of the second variant there are, as usual, less natpot terms than grid points (See Example 11.3). The numbers in the NATPOT-BASIS-SECTION were chosen such that there was virtually no difference to the numerically exact **adpop**-run.

11.9 Plotting 2D cuts through the system density

The program **showsys** can be used to display one- and two-dimensional cuts through the system density using the psi file. This must have been generated during a propagation run by including the `psi` keyword in the RUN-SECTION of the input file. If this file has not been generated, one-dimensional densities may be still be plotted from the gridpop file using the **showd1d** program (see Sec. 11.5).

The options are accessed by typing in the number given and responding to the questions. The words in brackets indicate what the present option is. Some options lead to a change in options being displayed.

Option 10 can be used to change between various plot tasks. What is possible depends on the system: for a non-adiabatic system the choices include not only plotting the reduced density, but plotting a cut through the adiabatic or diabatic wavefunction. These plots should be used with caution as the values are often extremely low in a cut. If a pes file is present (see Sec. 11.10), cuts through the PES may also be plotted. If there is more than one electronic state, one may chose between a adiabatic or diabatic representation of the potential.

```
  0 = stop
  1 = plot to screen
  2 = print plot
  3 = save plot to a postscript file
  4 = save plot to a gnuplot file (use after 1 or 2)
  5 = save data to an xyz file
  9 = toggle re-plot (get new set of contours)

 10 = change plot task (plot reduced density)
 20 = change coordinate section (  rd=x  rv=y  theta=1.545  Time=0.000)
 30 = change coordinate bounds
 40 = show coordinate info
 50 = change a single coordinate
 80 = change coordinate units
 90 = change Z-axis units (au)

110 = toggle contour mode (linear)
120 = change number of contours (21)
150 = toggle grid (off)
160 = toggle surface (off)
170 = toggle contour lines (on)
240 = toggle key (on)
245 = toggle title (on)
250 = toggle show points (off)
260 = toggle stick spectrum (off)
270 = toggle smooth curve (off)
280 = change time-slice format, e.g. movie(step-through)
285 = change time-step (  1)
290 = toggle no-weigths (off)

400 = Overlay plots (off)

900 = toggle gnuplot output format (on)
910 = change printer (lpr)
920 = change GNUplot command
```

Example 11.6: The start-up menu in the **showsys** program to enable interactive plotting of the system density and potential energy surfaces.

Option 20 allows a different cut to be chosen. Enter either `x`, `y`, or a number for each DOF. Information about the mode boundaries is given by option 40. The program then chooses the grid point nearest to the selected coordinate for the plotted cut. Note that when densities are plotted, the values of the numbers given are irrelevant. They merely serve as space-holders. The density is the integral of $|\Psi|^2$ over all those DOFs, which are labelled with a number.

Option 400 allows to generate overlay plots, i. e. plotting a density on top of the contour lines of the potential. Before using this menu point, the potential plot data has to be stored to some file by using menu point 5. (See the tutorial Sec. (2.1) and Fig. 2.2 for an example).

11.10 Plotting cuts through the potential energy surfaces

The program **showsys** can be used to display one- and two-dimensional cuts through the potential energy surfaces of the system. Before this can be done, the **MCTDH** program must be used to generate a pes file from the Hamiltonian information in the operator file (see

Chapter 6). The pes file is an operator file from which all terms are removed which contain derivative operators or CAPs.

To generate a pes file, set up an input file specifying the system and operator with a PRIMITIVE-BASIS-SECTION, A SPF-BASIS-SECTION, and an OPERATOR-SECTION. While no information is required about the single-particle function basis, the SPF-BASIS-SECTION is required as it also defines which degrees of freedom defined in the PRIMITIVE-BASIS-SECTION are included. In the RUN-SECTION, the keyword `genpes` is then required, along with the `name` of a directory in which to store the new file. Now the **MCTDH** program is run, and the file name/pes generated. The op.log.pes file contains information on the function that has been set up, and the log is now called log.pes. This makes it possible to use the same name-directory as the propagation run.

Rather than editing the input file it is often more convenient to use options. E. g.

```
quantics -pes nocl1
```

will generate a pes file of the NOCl S1-surface and stores it in the name directory.

Now change to the directory where the pes file has been stored and start the showsys program. The program is also able to generate two-dimensional plots of the system density from the psi file. If only pes plotting is required then start the program using the `-pes` option:

```
showsys -pes
```

If both pes cuts and density plots are wanted, this option should not be used. A menu appears which allows the interactive generation of plots. This is shown in Example 11.6.

When the potential used is given by a natpot file, generated by the **potfit** program (see next Section), then it is more convenient to use the **showpot** program to visualise the potential energy surface. **showpot** is menu driven, similar to **showsys**, and it allows to plot 1D and 2D cuts of the original surface, of the natural potential fit, and of differences between them. When the parameter natpot-cut (menu point 500) is larger than zero, then all natural potential terms, the supremums norm of which is smaller than natpot-cut, are removed. A proper use of natpot-cut may reduce the number of potential terms (by a factor 1/2, or so) with only marginally reducing the accuracy of the fit. The parameter natpot-cut is also available in QUANTICS. The number of omitted terms is protocolled in the showpot.log file.

Chapter 12

Using the Potfit program

12.1 Transforming a potential to product form

For optimal performance, the MCTDH algorithm requires the Hamiltonian to be given as a sum of products of single-particle operators (MCTDH product form). The kinetic energy operator usually is in MCTDH product form, but the potential is often given as a multidimensional function. The program **potfit** is able to transform a given potential energy surface to MCTDH product form. For small systems (e. g. 3D) it does this job fast and reliable. However, in contrast to the QUANTICS program, which avoids using the primitive product grid, **potfit** has to employ the full primitive product grid. The computational resources used by **potfit** thus increase much more strongly with the size of the system than the ones required by QUANTICS.

The numerical effort of **potfit** can be reduced, if the potential surface is partly given in product form. For example, if a 6 D surface reads

$$V(u,v,w,x,y,z) = V_1(u,v,w)V_2(x,y,z) + V_3(u,v,w)V_4(x,y,z) \tag{12.1}$$

one should, of course, apply **potfit** to V_1, ..., V_4 individually, rather than applying it directly to V.

Similar to **quantics**, **potfit** is started with giving the path of an input file as argument.

```
potfit <inputfile>
```

The input file is structured similar to the QUANTICS input file. To understand it, one must be familiar with the basics of the potfit algorithm. The potfit algorithm is described in Ref. [1] and in the original papers [22, 23]. In short, the fit to product form is performed in two steps. First, the potential density matrices are diagonalised to obtain the natural potentials. The thus generated product representation of the potential minimises (to a very good approximation) the overall L^2 error. As there are regions of greater and lesser physical importance, a better representation can be achieved by introducing weights which emphasise the regions of physical importance. Separable weights, i. e. weights that act on one degree of freedom only, can be incorporated into the first step. More powerful, however, are correlated weights, i. e. weights which cannot be written in product form. In a second step one thus may iteratively improve the representation by employing correlated weights. The correlated weights are implemented as *relevant regions*. A relevant region may be defined, e. g., as those areas where the potential energy is below some threshold, $V(\boldsymbol{R}) < V_{max}$. The program then iteratively improves the fit in the relevant region (while making it worse in the non-relevant region).

```
RUN-SECTION
niteration = 10
name = noclfit
iteration  prodwei
end-run-section

OPERATOR-SECTION
pes = nocl1sch          # Schinkes surface
vcut <  5.0d0,ev        # Potential is cutted above 5 eV
vcut > -1.0d0,ev        # and below -1 ev
end-operator-section

NATPOT-BASIS-SECTION
rd = 5       # rd = 15   These are the values for nat.pot. I
rv = 4       # rv = 15   review Section 9.1
theta = contr
end-natpot-basis-section

PRIMITIVE-BASIS-SECTION
  rd    sin     36   3.80    5.60
  rv    HO      24   2.136   0.272,ev  7.4667,AMU
  theta Leg     60     0    all
end-primitive-basis-section

SEPARABLE-WEIGHT-SECTION
  rd       5   3.904   5.83d-03
  rv       2     v:NO   1.d-3  1.d0
  theta    3    2.22   6.d0
end-separable-weight-section

CORRELATED-WEIGHT-SECTION
v  < 2.0,eV
rd <  5.d0
end-correlated-weight-section

end-input
```

Example 12.1: A potfit input file for the NOCl S1 surface.

The output files of potfit are structured similarly to the QUANTICS ones. There are the files output, input, log and timing. The prodwei file lists the separable weights and the file iteration compiles various error measures for each iteration step. The script **plpweight** reads prodwei and plots the separable weights, and the script **plpit** reads iteration and plots the error measures versus the number of iterations. The file natpot finally contains the natural potential fit which may be read by the QUANTICS program.

A potfit input file, noclpot.inp, is shown in Example 12.1. The RUN-SECTION defines the number of iterations to be performed, the name directory, and the files to be opened. The potential is first evaluated on the full product grid and written to the file vpot. Note, vpot is needed by **showpot** when plotting the exact potential.

The OPERATOR-SECTION specifies the potential energy surface to be used and defines cuts which remove large (positive and/or negative) potential values which, if kept, would slow down the integrator.

The NATPOT-BASIS-SECTION defines how many natural potentials are used for the fit. The more natural potentials one includes, the more accurate is the fit but the slower is the MCTDH calculation. One of the degrees of freedom should have the argument `contr`. This is the mode, over which a contraction is performed (see Refs. [1, 22, 23]). Contract over that degrees of freedom which converges most slowly. Inspect the output to see which one it is. One should avoid to contract over a mode if it is defined on a much larger grid than the other modes. To decide how may natural single particle potentials should be included in the potfit, one should inspect the output file which displays the natural populations as well as the sums of neglected natural populations. This will be discussed below.

MODE COMBINATION is also possible. In the NATPOT-BASIS-SECTION, the degrees of freedom can be combined into a single mode in the same way, as it is done in the spf-basis-section for an quantics run. If the resulting natpot file is used in a quantics run, the degrees of freedom combined in potfit must also be combined in the spf-basis-section (note that the order of the degrees of freedom in the mode must be identical in the natpot-basis-section and in the spf-basis-section). If the degrees of freedom are not combined in the natpot-basis-section, they still can be combined in a quantics run. There are, however, restrictions. For each combined mctdh-mode, potfit must use precisely the same combination, or may treat all degrees of freedom of this mode uncombined. Combining degrees of freedom can reduce the number of natural potentials needed for convergence. This will be important for large systems.

The PRIMITIVE-BASIS-SECTION must be identical to the PRIMITIVE-BASIS-SECTION of the following QUANTICS calculation, except for the ordering of the degrees of freedom. This ordering, however, must be consistent with the ordering of the arguments of the pes to be fitted. If one is insecure about the latter ordering, inspect the file source/opfuncs/funcsrf.F and search for the name of the particular pes under discussion. See also the HTML documentation. NB: One may use the `order` keyword (OPERATOR-SECTION of the potfit input file) to define a new order in which the arguments are passed to the surface routine. However, this does not work for the `readsrf` surface. See the HTML documentation for details.

Note that **quantics** uses the modelabels to associate the natpot terms with the DOFs. If this is not wanted one may give the keyword `ignore` as a parameter to the `natpot` keyword in the LABELS-SECTION. In this case one must use a numbered input (e. g. `|1&2&3 V`) in the HAMILTONIAN tableau to indicate on which DOFs and in which order the natpot shall operate. See the HTML documentation for details.

The SEPARABLE-WEIGHT-SECTION and the CORRELATED-WEIGHT-SECTION finally define the separable and correlated weights, respectively. See the HTML documentation for details.

The OUTPUT FILE contains important information on the natural populations and on error measures. Shown in Example 12.2 is an excerpt of an potfit output-file, which is generated by running the input Example 12.1. The block named `Trace - Sum of all preceding Natural Weights [eV**2]` displays the sum of "neglected" reduced natural weights, e. g. the second entry, $0.5371E-03$, is the sum of eigenvalues 3 to 36. This sum is directly related to the fit error. As in this example we took 5 and 4 single particle potentials into account, the estimated error is given by $\sqrt{0.2817E{-}04 + 0.2474E{-}05} = 0.5535E-02$ in eV, i. e. 5.5 meV. This estimate is printed in the output below the line `Global (weighted) L2 error estimated from neglected ...`, and it

```
*************      Mode:   1   rd            ************************

 Trace of reduced density matrix :  0.8297 au,  red. trace  0.8399 eV**2
 Number of eigenvalues considered:    5  /   36
 Reduced Eigenvalues (Natural Weights) [eV**2] :
    1  0.8363E+00  0.3063E-02  0.3058E-03  0.1667E-03  0.3651E-04  0.1198E-04
    7  0.5736E-05  0.4436E-05  0.2863E-05  0.1240E-05  0.1052E-05  0.4074E-06
   13  0.1324E-06  0.1037E-06  0.8599E-07  0.4732E-07  0.3625E-07  0.1433E-07
   19  0.7491E-08  0.7248E-08  0.5348E-08  0.3652E-08  0.1783E-08  0.7816E-09
   25  0.4284E-09  0.3753E-09  0.2785E-09  0.2162E-09  0.1371E-09  0.6923E-10

  Trace - Sum of all preceding Natural Weights [eV**2] :
    1  0.3600E-02  0.5371E-03  0.2313E-03  0.6467E-04  0.2817E-04  0.1618E-04
    7  0.1045E-04  0.6010E-05  0.3147E-05  0.1907E-05  0.8552E-06  0.4478E-06
   13  0.3154E-06  0.2118E-06  0.1258E-06  0.7847E-07  0.4222E-07  0.2789E-07
   19  0.2040E-07  0.1315E-07  0.7802E-08  0.4150E-08  0.2367E-08  0.1585E-08
   25  0.1157E-08  0.7814E-09  0.5028E-09  0.2866E-09  0.1495E-09  0.8030E-10

 Abs(Trace - Sum of relevant Natural Weights) [eV**2] :  2.81665E-05
 Sqrt(Abs(Trace - Sum of rel. Nat. Weights))    [meV] :    5.3072

*************      Mode:   2   rv            ***********************

 Trace of reduced density matrix :  0.8297 au,  red. trace  0.8399 eV**2
 Number of eigenvalues considered:    4  /   24
 Reduced Eigenvalues (Natural Weights) [eV**2] :
    1  0.8370E+00  0.2502E-02  0.3699E-03  0.2115E-04  0.1505E-05  0.5984E-06
    7  0.1717E-06  0.9290E-07  0.4180E-07  0.3013E-07  0.1483E-07  0.7315E-08
   13  0.4230E-08  0.3149E-08  0.1693E-08  0.1479E-08  0.9545E-09  0.3832E-09
   19  0.1571E-09  0.2029E-10  0.6050E-12  0.1463E-18  0.2231E-36 -0.2449E-18

  Trace - Sum of all preceding Natural Weights [eV**2] :
    1  0.2896E-02  0.3935E-03  0.2362E-04  0.2474E-05  0.9691E-06  0.3708E-06
    7  0.1990E-06  0.1061E-06  0.6434E-07  0.3421E-07  0.1938E-07  0.1207E-07
   13  0.7838E-08  0.4688E-08  0.2995E-08  0.1516E-08  0.5612E-09  0.1780E-09

 Abs(Trace - Sum of relevant Natural Weights) [eV**2] :  2.47378E-06
 Sqrt(Abs(Trace - Sum of rel. Nat. Weights))    [meV] :    1.5728

*************      Mode:   3   theta         ************************

 Contracted mode. Dimension =     60  /     60


 Global (weighted) L^2 error estimated from neglected natural weights:
        5.5354 meV          44.646cm^-1      2.0342E-04 a.u.

 Weighted   rms-error on rel. grid points [meV]:   3.9734  1.4602E-04 au
 Weighted   rms-error on all  grid points [meV]:   5.4941  2.0190E-04 au
 Unweighted rms-error on rel. grid points [meV]:  31.2338  1.1478E-03 au
 Unweighted rms-error on all  grid points [meV]: 133.0489  4.8895E-03 au
 Max.  absolute error on rel. grid points [meV]: 313.2904  1.1513E-02 au
 Max.  absolute error on all  grid points [ eV]:   2.2438  8.2457E-02 au
```

Example 12.2: An excerpt of a potfit output file for the NOCl S1 surface.

compares well with the numerically evaluated error (`Weighted rms-error on all..`) which is 5.4941 meV. By the way, the choice 5/4 for the number of natural potentials is a bit unbalanced, as the sums of neglected weights of the two DOFs are quite different. A more balanced choice would be 5/3 or 9/4, which would lead to (estimated) errors of 7.2 meV or 2.4 meV, respectively.

The following 10 iterations reduce the fit error to 2.202 meV on the relevant region, whereas the global error on all grid points increases to 9.067 meV. Note that *weighted* always refers to separable weights and *relevant* refers to correlated weights. For large systems the numerical evaluation of the rms-error may become costly. In such a situation it may be useful to switch this evaluation off or perform it only every n-th iteration step. See the HTML documentation for details.

12.2 Using *ab initio* data

An interesting feature of the QUANTICS package is the possibility to define the potential energy operator (or a part of it, or in general any local operator in configurational space) directly from *ab initio* data, in a way that can later be used by a QUANTICS calculation. The multidimensional grid, on which the *ab initio* data is collected, however, must be a product grid. In general, there are no further restrictions, e.g., an equidistant distribution of the grid points is not required. The **potfit** program can be used in order to transform the supplied data into a product form, practically a natpot file, that can later be used in the QUANTICS simulations. This is similar to the transformation of general multidimensional functions into product form that has been covered in chapter 12.1. Also, the primitive grid where the *ab initio* data is collected happens to be usually rather sparse, due to the cost of evaluating the desired property on each point. The QUANTICS package provides the **chnpot** utility, which allows to interpolate between natural potentials defined in different primitive grids. Therefore, the user may collect the *ab initio* data in a rather sparse primitive grid that can be later be interpolated into a more suitable one for the dynamical simulation phase. These operations will be discussed in detail in this chapter.

The practical implementation of the MCTDH algorithm uses DVR's for the primitive basis, whose points define the primitive grid (see Chapter 4). It is assumed in our discussion that the value of some property, e.g., the potential energy, has been collected on the points defined by the primitive grid using some external program. One has to differentiate between the actual points of the primitive grid in each coordinate and the associated value of a certain property. As will be immediately seen, both pieces of information are given separately to the programs that have to use them.

There are mainly three ways to use *ab initio* data in a QUANTICS calculation:

12.2.1 Using *ab initio* data directly with the QUANTICS program

This is the least flexible of the possibilities being discussed, but the concepts that will be introduced apply equally to the other procedures. It corresponds to the direct path to the usage of QUANTICS as depicted in Fig. 12.1. First, let's assume that we have the *ab initio* data values at the points defined by some primitive grid. The primitive grid being used should correspond, for each coordinate, with some of the DVR's defined in the QUANTICS program (What to do when this is not the case, will be covered in the following sections). As usual, the primitive grid is defined in the PBASIS-SECTION of the input file. For example one could have something like:

```
PRIMITIVE-BASIS-SECTION
x    sin  17    -2.4  2.4
y    sin  21    -2.5  3.5
end-primitive-basis-section
```

for coordinates *x* and *y*. The information concerning where the actual data values are found is given in the LABELS-SECTION of the operator file. One has to define a new label making use of the `readsrf` keyword:

```
LABELS-SECTION
vdat  = readsrf{pathtofile S}
end-labels-section
```

`S` can be either `ascii` or `binary` depending on the file to be read. `pathtofile` is the absolute or relative path to the file containing the data. The newly defined `vdat` label can then be used in the HAMILTONIAN-SECTION of the operator, for example:

```
HAMILTONIAN-SECTION
---------------------
modes      | x | y
---------------------
1.0        |1&2 vdat
other operator lines
end-hamiltonian-section
```

The file containing the data values consists of a single column of numeric entries, written so that the first index runs fastest. The order of the indexes is defined by the `|i&j&k&...` construct in the HAMILTONIAN-SECTION as shown above. In our example the file has to be created so that it could be read by the following pseudo-code, where the *i* index runs on the `y` coordinate:

```
iterate i in range 1 to 21:
    iterate j in range 1 to 17:
        read v(j,i)
    end iterate
end iterate
```

One should note the following: the described procedure implies that both the `x` and `y` coordinates belong to the same combined mode and there are no other coordinates are present in this mctdh-particle. Otherwise the program would treat the new potential as a `muld`-potential, i. e. a multi-dimensional potential. This slows down the performance of QUANTICS. Therefore, the procedure outlined above is most useful when the potential operates on one combined mode (MCTDH particle) exclusively.

12.2.2 Using the potfit program

A second alternative is to use the **potfit** program to convert the *ab initio* data to product form, and then use **quantics**, as it is shown in Fig. 12.1. This possibility circumvents the inconveniences described at the end of the previous section, since the different degrees of freedom can then be used in different combined modes, or be simply uncombined. The usage of **potfit** has been covered in chapter 12.1 of this guide, so here it will only be covered how to make it read an *ab initio* surface defined on a primitive grid. The `readsrf` keyword (see previous section for details) comes now into play in the OPERATOR-SECTION of the input file of **potfit**:

```
OPERATOR-SECTION
pes = readsrf{path-to-file S}
.......
end-operator-section
```

As in the previous case, the file containing the data values has to be written with the index of the first specified degree of freedom running fastest (the most internal loop), the second degree of freedom in the second most internal loop and so on. The order of the degrees of freedom is given by the PRIMITIVE-BASIS-SECTION in the **potfit** input file. The program **potfit** will generate a natpot file to be used directly by **quantics**. Remember that **quantics** uses the modelabels to associate the natpot terms with the DOFs.

12.3 Extra flexibility, combining potfit and chnpot

The maximum flexibility in the usage of *ab initio* data is accomplished by combining **potfit** with the **chnpot** utility. There are two main reasons why the initial primitive grid in which the *ab initio* data points are given should be transformed into a more suitable one. First, the given points may be too sparse, and a more dense grid is desired for the dynamical simulation phase. Second, the primitive grid where the points are supplied does not correspond to a DVR defined in the QUANTICS code.

12.3.1 Dealing with an arbitrary primitive grid

In case that the *ab initio* data values are given in a grid that does not correspond to a DVR known to QUANTICS, the `external` keyword can be used in the PRIMITIVE-BASIS-SECTION of the **potfit** input file:

```
PRIMITIVE-BASIS-SECTION
x   external   16  path_to_x_grid  <unit>
y   external   16  path_to_y_grid  <unit>
end-primitive-basis-section
```

`path_to_x_grid` is the absolute or relative path to a file containing, in one column, the values x_i of the `x` coordinate, in this case, a total of 16 entries. One needs to create a file with this information for every `external` entry. Optionally, a unit may be given to convert e. g. Angstroem to au or degree to radian. As before, the information concerning where the *ab initio* data points are found is given in the OPERATOR-SECTION of the same input file. After execution, the program yields a natpot file and a dvr file. One should note that these natpot and dvr files cannot be used directly in a simulation, since only the grid points of the DVR are found on the dvr file (and not the matrices representing the derivative operators). However, any natpot whose primitive grid is known (through the corresponding dvr file) can be interpolated to a desired DVR by the **chnpot** utility, yielding new natpot and dvr files to be used in the simulation phase. This is covered in the next subsection. (As a remark, the external keyword is more powerful and the non-local part of an arbitrary DVR can also be supplied, see the HTML documentation for this advanced feature).

12.3.2 Transforming between two natural potentials with chnpot

The **chnpot** utility interpolates a given natpot file into a new primitive grid corresponding to the desired DVR. It needs to know where to find the initial natpot and dvr files. An example input file for **chnpot** reads:

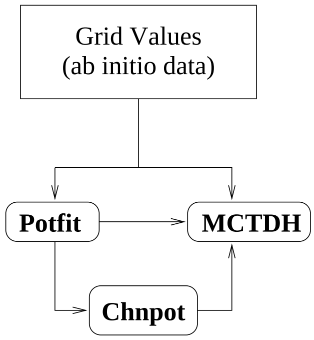

Figure 12.1: Main concepts involved in the usage of *ab initio* data with the QUANTICS package

```
RUN-SECTION
name = <S1>
dvrdir = <S2>
natpotdir = <S3>
end-run-section

PRIMITIVE-BASIS-SECTION
 x   sin  45  -2.4 2.4
 y   sin  45  -2.5 3.5
end-primitive-basis-section

Fit-Section
x  spline
y  spline
end-fit-section

end-input
```

The RUN-SECTION contains keywords that control the program execution, e.g., where the (old) initial files are found, where to store the (new) results, etc. See the HTML documentation for a detailed description of every keyword. The PRIMITIVE-BASIS-SECTION specifies the new DVR, and hence the new primitive grid. The Fit-Section describes how each degree of freedom has to be interpolated. A detailed description of the **chnpot** command line and input file options, is maintained in the HTML documentation. Here some hints are given in how to use this utility: after **chnpot** has been executed, the user should check that the natpot obtained has been correctly interpolated. This can be done conveniently with the **showpot** utility. Original potentials with discontinuities or regions where the function slope changes abruptly can lead to oscillatory behavior of the interpolated function. Depending on the starting data points, the user may have to find the most convenient interpolation scheme for each degree of freedom. The program can perform spline and essentially non oscillatory interpolation (ENO) for degrees of freedom without special boundary conditions and fourier, sine and cosine interpolation for angular or other degrees of freedom. The program can handle combined modes (combined in **potfit**) when the combination is up to 2D type, as it is capable of 1D and 2D interpolations. If the original natpot has been obtained from a sparse primitive grid, it may be a good idea to potfit it using as many natural potentials as grid points in each degree of freedom, in case it is computationally feasible. In this way, the resulting natpot is exact on the original grid points. What follows is the already described interpolation to a suitable grid using **chnpot**.

12.4 *Advanced topic*: Manipulating potentials with the projection program

While it is fairly straightforward to include program code for new potential energy surfaces into the QUANTICS program package, it is sometimes desirable to use an existing PES routine to generate a related PES. Examples for this would be:

- Models with reduced dimensionality, where you need an effective potential which is derived from the full dimensional potential by averaging over some degree(s) of freedom.
- Series expansion of a PES, e.g. multipole expansions or Fourier transforms of angular degrees of freedom.

These tasks can be accomplished with the **projection** program. Basically, this program takes the potential $V(q_1, \dots, q_f)$ from an existing PES routine and "projects out" some degrees of freedom (say, $q_1, \dots, q_p$) by integrating over them with *projection functions* $\chi_\varkappa(q_\varkappa)$ to yield an $(f-p)$-dimensional *projected potential*:

$$V_{\text{proj}}(q_{p+1}, \dots, q_f) = \int \mathrm{d}q_1 \cdots \mathrm{d}q_p \; \chi_1(q_1) \cdots \chi_p(q_p) \, V(q_1, \dots, q_f) \tag{12.2}$$

Numerically this is done by evaluating the PES and the projection functions on the DVR grid points $x^{(\varkappa)}_{\alpha_\varkappa}$ and employing the DVR weights $w^{(\varkappa)}_{\alpha_\varkappa}$:

$$V_{\text{proj}}(x^{(p+1)}_{\alpha_{p+1}}, \dots, x^{(f)}_{\alpha_f}) = \sum_{\alpha_1} \cdots \sum_{\alpha_p} w^{(1)}_{\alpha_1} \chi_1(x^{(1)}_{\alpha_1}) \cdots w^{(p)}_{\alpha_p} \chi_p(x^{(p)}_{\alpha_p}) \, V(x^{(1)}_{\alpha_1}, \dots, x^{(f)}_{\alpha_f}) \tag{12.3}$$

projection will store the resulting V_{proj} as a vpot file which must then be processed in a subsequent **potfit** run.

During the **projection** run, the first p degrees of freedom are removed, and so one is absolutely free in choosing the DVR basis (i.e. basis type as well as basis parameters and basis size) for these DOFs. For the $f-p$ remaining DOFs however, the DVR basis must match the one for the subsequent **potfit** run.

As one is free to choose the DVR basis for the DOFs on which one projects, one can enhance the accuracy of the integration scheme (12.3) by choosing it according to the projection functions. E.g. if the projection function is a Legendre polynomial, one should use a Legendre DVR, as this will turn the integration into a Gauss quadrature. For some of the available projection functions, the HTML documentation gives hints on which DVR to choose.

12.4.1 Input and output files

As both **projection** and **potfit** take a PES on the full grid as primary input, their input files are quite similar. An example for a **projection** input file is given in example 12.3. (You can find a full version with more extensive comments in $MCTDH_DIR/pinputs/bmkpe_proj.inp .)

```
RUN-SECTION
  name = bmkpproj  # directory where output is written
  timing           # write timing information to file "timing"
  output           # write output to file "output"
end-run-section

OPERATOR-SECTION
  pes  = h4bmkp{jacobian}  # surface to use: BMKP H4 surface
  vcut < 0.0,ev            # cut off at 0.0eV (PES zero point is at ~-9.5eV)
end-operator-section

PRIMITIVE-BASIS-SECTION
  R     FFT  128  1.80    17.0
  RAB   HO   8    1.4483  0.019216  0.5,h-mass
  RCD   HO   8    1.4483  0.019216  0.5,h-mass
  AL    leg  10   0  even
  BE    leg  10   0  even
  PHI   exp  27   2pi
end-primitive-basis-section

PROJECTION-SECTION
  PROJECTOR 000
    AL   leg  0  0
    BE   leg  0  0
    PHI  cos  0
  end-projector
  PROJECTOR 200
    AL   leg  2  0
    BE   leg  0  0
    PHI  cos  0
  end-projector
  # ... (further projectors omitted)
  PROJECTOR 222
    AL   leg  2  2
    BE   leg  2  2
    PHI  cos  2
  end-projector
  error
end-projection-section

end-input
```

Example 12.3: A projection input file for the BMKP surface for $(H_2)_2$ (excerpt).

The RUN-SECTION only specifies the name directory and which output files to be opened. The OPERATOR-SECTION is the same as in **potfit**; it specifies the PES to be used and defines cuts to remove large potential values.

The PRIMITIVE-BASIS-SECTION has the same structure as in **quantics** and **potfit**. It defines the grid points of the product grid on which (12.3) is evaluated. As mentioned above, for the DOFs on which one projects one is free to use any basis definition. For the remaining DOFs, one must use the same basis definitions as for the subsequent **potfit** run (which in turn must be the same as for the following **quantics** run, unless one uses **chnpot** in between).

The PROJECTION-SECTION contains the definitions of the projector functions $\chi_\varkappa(q_\varkappa)$. During a single **projection** run, it is possible to use several different sets of projection functions. One such set of functions is defined in one PROJECTOR section. After the

`PROJECTOR` keyword one must give an arbitrary string which is used as a label for the projected potential (in example 12.3 this would be the labels "000", "200", and "222"). The definition of the individual projector functions is done by first stating the modelabel of the DOF on which to project (here e.g. "`AL`"), and then giving the type and parameters of the projection function (here e.g. "`leg 2 0`", which stands for the Legendre polynomial P_{20}). Please see the HTML documentation for which projector function types are available and what parameters they take.

For each defined projector, **projection** produces a vpot file which contains the projected potential on the product grid of the remaining DOFs. The name of this file depends on the given projector label, e.g. in case of the "200" projector the corresponding file will be called vpot_200. To use this file in the subsequent **potfit** run, one must read it with the `readvpot` keyword (in the RUN-SECTION) and also specify `pes=none` (in the OPERATOR-SECTION). Example input files for **potfit** runs which use **projection** output can be found in $MCTDH_DIR/pinputs/bmkpe_fit_*.inp .

In the case where the projected potentials $V^{(i)}_{\text{proj}}$ (where i numbers the different sets of projector functions) constitute a series expansion of V, it is worthwhile to re-assemble the projected potentials into another full-dimensional potential $\tilde{V}$ which can then be compared to the original potential V. Thus one can check whether enough terms of the series expansion have been taken into account to faithfully represent the original potential. To this end, let $\chi^{(i)}_{\varkappa}$ be the projector function for the $\varkappa$-th DOF in the i-th set, and let $\tilde{\chi}^{(i)}_{\varkappa}$ be the corresponding *complementary projector function* such that

$$\int \mathrm{d}q_{\varkappa}\, \chi^{(i)}_{\varkappa}(q_{\varkappa})\, \tilde{\chi}^{(i)}_{\varkappa}(q_{\varkappa}) = 1 \tag{12.4}$$

If we assume that the projector sets i and j are orthonormal, i.e.

$$\prod_{\varkappa=1}^{p} \int \mathrm{d}q_{\varkappa}\, \chi^{(i)}_{\varkappa}(q_{\varkappa})\, \tilde{\chi}^{(j)}_{\varkappa}(q_{\varkappa}) = \delta_{ij} \tag{12.5}$$

then we can define the re-assembled potential

$$\tilde{V}(q_1,\ldots,q_f) = \sum_i \tilde{\chi}^{(i)}_1(q_1)\cdots\tilde{\chi}^{(i)}_p(q_p)\, V^{(i)}_{\text{proj}}(q_{p+1},\ldots,q_f) \tag{12.6}$$

An error measure that depends on the remaining coordinates and which has the unit of energy is:

$$\Delta V(q_{p+1},\ldots,q_f) = \left(\frac{\int \mathrm{d}q_1\cdots\mathrm{d}q_p\, |V(q_1,\ldots,q_f) - \tilde{V}(q_1,\ldots,q_f)|^2}{\int \mathrm{d}q_1\cdots\mathrm{d}q_p}\right)^{1/2} \tag{12.7}$$

If you give the `error` keyword in the PROJECTION-SECTION, this error measure will be calculated and stored in another vpot file, named projerr.vpot . This can then be inspected with the **showpot** program.

Finally, the log file lists some brief information about the projected potentials, and (in case that `error` was requested) the mean, minimum and maximum values of ΔV.

12.4.2 Generating a Fourier-transformed potential

In the case of a four-atomic system which is described in Jacobi coordinates (see Fig. 12.2), it is of technical advantage for the QUANTICS package to replace the relative torsional angle φ by two torsional angles φ_1 and φ_2 such that $\varphi = \varphi_1 - \varphi_2$ and then describe the system in terms of their conjugate momenta k_1 and k_2. (For more details see [24] or [25].) In this case the transition from $\varphi_{1,2}$ to $k_{1,2}$ is simply done by a Fourier transform (in the following we will abbreviate the set of coordinates R, r_1, r_2, θ_1, θ_2 simply by Q):

$$\tilde{\Psi}(Q, k_1, k_2) = \frac{1}{(2\pi)^2} \int_0^{2\pi} \mathrm{d}\varphi_1 \, \mathrm{e}^{-ik_1\varphi_1} \int_0^{2\pi} \mathrm{d}\varphi_2 \, \mathrm{e}^{-ik_2\varphi_2} \, \Psi(Q, \varphi_1 - \varphi_2) \tag{12.8}$$

Likewise, we Fourier-transform the potential $V(Q, \varphi)$:

$$\tilde{V}_\Omega(Q) = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{d}\varphi \, \mathrm{e}^{-i\Omega\varphi} \, V(Q, \varphi) \tag{12.9}$$

or vice versa:

$$V(Q, \varphi) = \sum_{\Omega=-\infty}^{+\infty} \mathrm{e}^{i\Omega\varphi} \, \tilde{V}_\Omega(Q) \tag{12.10}$$

Then it is straightforward to see that the action of the potential operator $\hat{V}$ on the wavefunction in terms of $k_{1,2}$ is given by

$$(\hat{V}\Psi)(Q, k_1, k_2) = \sum_{\Omega=-\infty}^{+\infty} \tilde{V}_\Omega(Q) \, \tilde{\Psi}(Q, k_1 - \Omega, k_2 + \Omega) \tag{12.11}$$

Since the potential V is real and often symmetric, $V(Q, \varphi) = V(Q, -\varphi)$, the Fourier-transformed potential is also symmetric, $\tilde{V}_\Omega(Q) = \tilde{V}_{-\Omega}(Q)$, and (12.9) simplifies to

$$\tilde{V}_\Omega(Q) = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{d}\varphi \, \cos(\Omega\varphi) \, V(Q, \varphi) \tag{12.12}$$

The latter form can easily be evaluated with the **projection** program. A PROJECTION-SECTION of the corresponding input file would look like this:

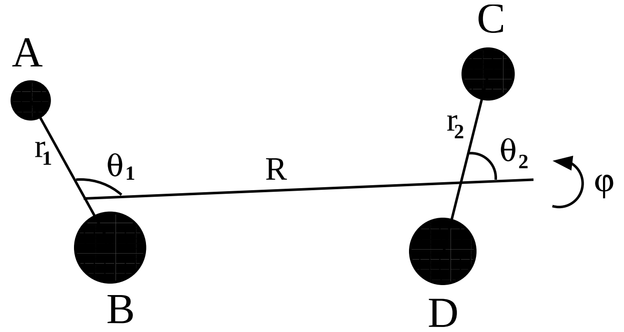

Figure 12.2: Jacobi coordinates for a 4-atomic system

```
PROJECTION-SECTION
  PROJECTOR K0
    PHI  cos  0
  end-projector
  PROJECTOR K1
    PHI  cos  1
  end-projector
  PROJECTOR K2
    PHI  cos  2
  end-projector
  # ...
  error
end-projection-section
```

where `PHI` is the modelabel of the torsional coordinate in question. Take note that the definition of the `cos` projector function already includes the factor $1/2\pi$! The usage of the `error` keyword is recommended here, as it will calculate the error measure described in the last section, so one can check whether one has calculated enough Fourier components to faithfully represent the original potential (look at the log file).

Once this **projection** run is finished, you will find the files vpot_K0, vpot_K1, etc., which contain the individual Fourier components $\tilde{V}_\Omega$. As these are defined on the full grid of the remaining coordinates Q, one must use **potfit** to bring them into the product form required by MCTDH. It will be necessary to write separate input files to fit the separate Fourier components, but these input files will look very much alike. When writing the input files, note the following:

- in the RUN-SECTION, use `readvpot` to read the vpot_Kx files created by **projection**
- in the OPERATOR-SECTION, use `pes=none`
- in the PRIMITIVE-BASIS-SECTION, use exactly the basis definitions from the **projection** input file, but only for the remaining coordinates Q; omit all the coordinates that were projected out.

After the **potfit** runs are complete, you can optionally change the primitive basis by using **chnpot**.

12.4.3 Using a Fourier-transformed potential in QUANTICS

The last section described how to generate the Fourier components $\tilde{V}_\Omega$ in a form usable by QUANTICS; however this is only one ingredient of (12.11). The other is the shifting of the k coordinates, $\tilde{\Psi}(Q, k_1, k_2) \to \tilde{\Psi}(Q, k_1 - \Omega, k_2 + \Omega)$. In QUANTICS this can be accomplished with the simple `shift` operator, which we will denote here formally as $\hat{S}_q^\Delta$ (where q is the DOF on which it operates and Δ is the amount by which the DOF is shifted):

$$(\hat{S}_q^\Delta \tilde{\Psi})(\ldots, q, \ldots) = \tilde{\Psi}(\ldots, q - \Delta, \ldots) \tag{12.13}$$

In this notation, (12.11) becomes (recall that $\tilde{V}_\Omega = \tilde{V}_{-\Omega}$):

$$\begin{aligned}(\hat{V}\Psi)(Q, k_1, k_2) &= \tilde{V}_0(Q)\,\tilde{\Psi}(Q, k_1, k_2) \qquad (12.14)\\ &+ \sum_{\Omega=1}^{\infty} \tilde{V}_\Omega(Q) \left[(\hat{S}_{k_1}^{\Omega} \hat{S}_{k_2}^{-\Omega} \tilde{\Psi})(Q, k_1, k_2) + (\hat{S}_{k_1}^{-\Omega} \hat{S}_{k_2}^{\Omega} \tilde{\Psi})(Q, k_1, k_2)\right]\end{aligned}$$

```
HAMILTONIAN-SECTION
------------------------------------------------------
 modes   |  R  |  r1   |   r2 | th1 |  k1   | th2 |  k2
------------------------------------------------------
#  ...   (kinetic energy omitted)
1.0      |1&2&3&4&6  V0           |5 1         |7 1
1.0      |1&2&3&4&6  V1           |5 kp1       |7 km1
1.0      |1&2&3&4&6  V1           |5 km1       |7 kp1
1.0      |1&2&3&4&6  V2           |5 kp2       |7 km2
1.0      |1&2&3&4&6  V2           |5 km2       |7 kp2
------------------------------------------------------
end-hamiltonian-section

LABELS-SECTION
V0  = natpot{fitK0}
V1  = natpot{fitK1}
V2  = natpot{fitK2}
kp1 = shift[1]
km1 = shift[-1]
kp2 = shift[2]
km2 = shift[-2]
end-labels-section
```

Example 12.4: An operator file showing the use a Fourier-transformed potential (excerpt). Only Fourier components up to Ω=2 are included.

or even more formally on the operator level:

$$\hat{V} \;=\; \tilde{V}_0 + \sum_{\Omega=1}^{\infty} (\, \tilde{V}_\Omega \hat{S}^{\Omega}_{k_1} \hat{S}^{-\Omega}_{k_2} \;+\; \tilde{V}_\Omega \hat{S}^{-\Omega}_{k_1} \hat{S}^{\Omega}_{k_2} \,) \tag{12.15}$$

This form can be easily translated into an QUANTICS operator file. (Of course one will truncate the series at a certain $\Omega_{\max}$.) Parts of the corresponding HAMILTONIAN- and LABELS-SECTIONs are shown in example 12.4. A few explanations are in order:

- During the following QUANTICS propagation, we will be using the two-dimensional KLeg DVR for the combined coordinates (θ_1, k_1) and (θ_2, k_2). (The use of this DVR is part of the technical reason why we did the Fourier transform in the first place.) This makes it necessary to order the coordinates differently than in the above theoretical considerations. Namely, k_2 can not directly follow k_1 but the k_i must be paired with the θ_i.
- This different coordinate ordering makes it necessary to use the slightly cumbersome `|1&2&3&4&6` syntax for the potfitted Fourier components `V0`, `V1`, `V2`. This tells the program to apply them to the set of DOFs (1, 2, 3, 4, 6), which corresponds to (R, r_1, r_2, θ_1, θ_2). (Also see section 6.13.)
- Here we assume that the natpot files of the potfitted Fourier components are located in the directories fitK0 etc. That is, fitK0 should contain the potfit of vpot_K0, fitK1 that of vpot_K1, etc.
- The `shift` operators must be put into the LABELS-SECTION and then referenced by their labels (`kp1`, `km1`, etc.) because of a syntactical restriction of QUANTICS: it is not allowed to use operators which take an argument directly in the HAMILTONIAN-SECTION.

This completes the setup of the operator file for a Fourier-transformed potential.

12.5 Downsizing previous *potfits*: the cutnpot and rdnpot functions

Under certain circumstances, it may be necessary to make use of previous *potfitted* potentials, *i.e.* natpot files. Possibly, it may be convenient to reduce the number of single particle potentials and the corresponding contracted coefficients. For instance, this may be useful to investigate the quality of the potential expansion (*e.g.* the fitting error) upon reduction of the expansion coefficients or, simply, in order to decrease the cost of the energy evaluation. For such a purpose, the **cutnpot** and **rdnpot** functions have been developed. It should be highlighted that they play a major role in the **mgpf** program (under development).

The process can be summarized in two steps: (i) elimination of unwanted single particle potentials and the corresponding contracted coefficients from a previously existing natpot file; and (ii) reading of the newly generated (reduced) natpot file for the evaluation of error measurement. The first step is accomplished by means of the **cutnpot** function. This is invoked by including the `cutnpot` keyword in the RUN-SECTION of the **potfit** input file. The NATPOT-BASIS-SECTION should contain the desired (smaller) values of the expansion coefficients while preserving the definitions of the contracted mode and the particles. The natpot file to be reduced (*cut*)[1] should either be placed in the working directory as natpot1 or under a different name (or location) in which case the name (with absolute path) should be provided as: `cutnpot=path`. Then **potfit** will produce a new (shrinked) natpot file within the same directory. Finally, the subsequent measurement of the fitting error (for sufficiently small primitive grids) is obtained by using the **rdnpot** function which is invoked by adding the RUN-SECTION keyword `rdnpot` and running again **potfit**.

As previously indicated, for the sake of consistency, the DVR definitions, the contracted mode, and the mode combination scheme, if any, must remain unaltered during the whole process. Hence, if a new particle scheme is needed, then a new **potfit** or a new **mgpf** calculation has to be performed.

The `cutnpot` and `rdnpot` keywords appear in the RUN-SECTION, otherwise the input is the standard one.

[1]Note that **cutnpot** yields the same results as a normal potfit with the reduced number of single particle potentials, provided no correlated weights are used.

Appendix A

The concept of the input file

With the exception of a continuation run, where a previous wavepacket propagation is carried on to longer times, all calculations require an input file, *name*.inp. This file is a text file, with the required options input as keywords. As all lines beginning with a # are treated as comments, title and other text can be usefully added to make the file easier to understand.

Example input A.1 shows the input file required for a simple wavepacket propagation calculation, using a modified Henon-Heiles Hamiltonian (see Ref. [2] for more details of this calculation). As the example shows, it is possible to include the information of the operator file in the input file. This is particularly useful for systems having a simple Hamiltonian.

The keywords in the input file are grouped together into sections, each with a specific set of information. The sections start with a line containing the keyword *XXX*-SECTION, and end with END-*XXX*-SECTION, where *XXX* is the name of the section. The possible sections are compiled in Tab. A.1.

Table A.1: The possible sections in the input file. Also displayed is whether a section is required for a certain calculation type.

Section	Calculation Type				
	gendvr	genoper genpes	geninwf	propagation relaxation	diagonalisation
RUN	yes	yes	yes	yes	yes
PRIMITIVE-BASIS	yes	yes	yes	yes	yes
PARAMETER[a]	yes[b]	yes[c]	yes[c]	yes[c]	yes[c]
SPF-BASIS	no	yes	yes	yes	yes
OPERATOR[d]	no	yes	yes	yes	yes
OP_DEFINE[a]	no	yes	yes	yes	yes
HAMILTONIAN[a]	no	yes	yes	yes	yes
LABELS[a]	no	yes[e]	yes[e]	yes[e]	yes[e]
INITWF	no	no	yes	yes	yes
INTEGRATOR	no	no	no	yes	no

[a] May be in the operator file
[b] Only if parameters are used in the DVR specifications
[c] Only if parameters are used in the Hamiltonian specification
[d] Only if the Hamiltonian is in an operator file
[e] Only if label definitions are required in the Hamiltonian specification

Table A.2: Description of the calculation types. The table shows the RUN-SECTION keyword required for a certain calculation type and the level associated with this type. Also given are the files that are created and needed by the different calculation types.

Level	Keyword	Description	Created files	Required files
1	`gendvr`	Sets up primitive bases	dvr	
2	`genpes`	Sets up a pes file for analysis	pes	dvr
2	`genoper`	Sets up an operator for use	oper	dvr
3	`geninwf`	Sets up an initial wavefunction	restart	dvr, oper
4	`propagation`	Propagates a wavepacket	User defined	dvr, oper, restart
4	`relaxation`	Relaxes a wavepacket	User defined	dvr, oper, restart
4	`diagonalisation`	Diagonalises a Hamiltonian	User defined	dvr, oper, restart

Which sections are required depends on the type of calculation to be made. Table A.1 lists the various sections, and indicates which are required for the various types. A possible calculation type is, for instance, a propagation, a relaxation, or a diagonalisation, symbolised in the RUN-SECTION by the corresponding keyword `propagation`, `relaxation`, or `diagonalisation`. Additionally hereto, one may use the MCTDH program to solely set up a primitive basis, a Hamiltonian operator, or an initial wavepacket. This is done with the keywords `gendvr`, `genoper`, or `geninwf` in the RUN-SECTION. (The RUN-SECTION is hence required for all calculation types.) The generated information can then be read from file in following calculations be using the keywords `readdvr`, `readoper`, or `readinwf` in the RUN-SECTION.

Each calculation type has a level associated with it, which reflects the stages for a calculation. These levels are listed in Tab. A.2. Each level keyword automatically contains the lower levels, thus the keyword `propagation` implies `gendvr`, `genoper`, `geninwf`, `propagation`, and a wavepacket propagation will be performed after first setting up a DVR, operator, and initial wavepacket. The listed files are files which contain the information from the lower level calculations.

In the input file there may appear keywords which have a UNIX filename as argument (e. g. `oppath =` $\cdots$). These filenames are interpreted relative to the location of the input file.

```
#################################################################
## Propagating a wavepacket using the Henon-Heiles Hamiltonian ##
#################################################################

RUN-SECTION
  propagation    tout=0.01    tfinal=0.50
  name = hh  psi  gridpop
end-run-section

PBASIS-SECTION
#Label   DVR    N     Parameter
    X    HO     32    0.0    1.0    1.0
    Y    HO     32    0.0    1.0    1.0
end-pbasis-section

SBASIS-SECTION
  X = 3     Y = 3
end-sbasis-section

OP_DEFINE-SECTION
  title      Henon-Heiles PES      end-title
end-op_define-section

PARAMETER-SECTION
  mass_X = 1.0
  mass_Y = 1.0
  lambda = 0.2, au
end-parameter-section

HAMILTONIAN-SECTION
-----------------------------
  modes       |  X   | Y
-----------------------------
1.0           |  KE  | 1
0.5           |  q^2 | 1
-lambda/3     |  q^3 | 1
lambda^2/16   |  q^4 | 1
1.0           |  1   | KE
0.5           |  1   | q^2
lambda^2/16   |  1   | q^4
lambda        |  q   | q^2
lambda^2/8    |  q^2 | q^2
-----------------------------
end-hamiltonian-section

INIT_WF-SECTION
  build
    X    gauss    1.80    0.00    0.75
    Y    gauss    0.00    1.20    0.50
  end-build
end-init_wf-section

INTEGRATOR-SECTION
  VMF    ABM = (6 , 1.0d-7 ,  1.0d-5)
end-integrator-section

end-input
```

Example A.1: An input file for a wavepacket propagation using the Henon-Heiles Hamiltonian.

Appendix B

The Structure of the Programs

Figure B.1 displays a flowchart of the MCTDH program package. The MCTDH program first reads the input file via the `eingabe` routines and computes the memory requirements. Depending on the input settings, it then starts some or all of the calculation types. The routines `call`*x* allocate the memory, the routines `run`*x* perform the calculations. Communication between these parts of the MCTDH program, as well as between the MCTDH and the Potfit and Analysis programs, is done employing the files indicated by ovals in the diagram.

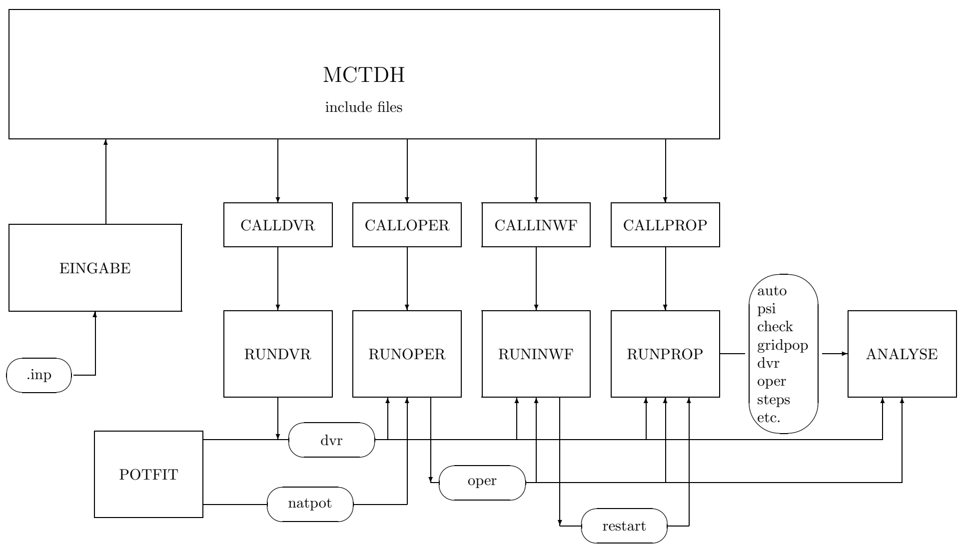

Figure B.1: The structure of the MCTDH programs. See text for details.

Appendix C

The built-in symbolic expressions

The following tables describe the symbols and related operators that can be used to set up a Hamiltonian operator.

General Remarks

With the aid of the caret ^ one may apply a power to operators. The power may be integer or real and may carry a sign. This, however, works only for potential like operators. Inspect the Tables below to learn, which operators can be exponentiated. Note, that symbols like `dq^2` or `j^2` are operator labels of their own right, they do not denote that the second power of the operators `dx` or `j` is taken literally. (Compare with Appendix B (Discrete Variable Representation) of the MCTDH review (Phys.Rep. 324 (2000) 1-105), to learn how `dq` and `dq^2` are defined).

One may multiply operators, e.g. a construct like `dq*cos*dq` is allowed. However, multiplication is allowed only among potential like operators and operators with a simple matrix representation. This excludes all `KLeg` and `PLeg` operators from multiplication. (See Table C.2 and notes to this table). Moreover, `natpots` cannot be multiplied with other operators. (See Section 6.9)

Table C.1: Simple one-dimensional operators. The expression x is the coordinate, r can be replaced by any real number, positive or negative. If $r = 1$ it is not required, e.g. q and $q\hat{}1$ are synonemous. The expression n can be replaced by any non-negative integer.

Symbol	Operator	Notes
1	1	Unit operator
I	i*1	Imaginary unit times unit operator
q^r	x^r	Multiply by rth power of x
qs^r	$(1-x^2)^{r/2}$	Multiply by rth power of $\sqrt{1-x^2}$
sin^r	$\sin^r(x)$	rth power of sine of coordinate
cos^r	$\cos^r(x)$	rth power of cosine of coordinate
tan^r	$\tan^r(x)$	rth power of tangent of coordinate
cosh^r	$\cosh^r(x)$	rth power of hyperbolic-cosine of x
sinh^r	$\sinh^r(x)$	rth power of hyperbolic-sine of x
acos^r	$\arccos^r(x)$	rth power of acosine of coordinate
asin^r	$\arcsin^r(x)$	rth power of asine of coordinate
atan^r	$\arctan^r(x)$	rth power of atangent of coordinate
exp^r	$\exp(x)^r$	exponential of coordinate
gauss^r	$\exp(-x^2)^r$	gaussian of coordinate
ngauss^r	$\exp(-x^2/2)^r/\sqrt{2\pi}$	normalized gaussian
legth:n	$P_n(\cos(x))$	nth order Legendre polynomial of cosine of x
asleg:l_m	$P_l^m(x)$	associated Legendre polynomial of x (see function `plgndr` in sorce/lib/utilities/legendre.f)
aslegth:l_m	$P_l^m(\cos(x))$	associated Legendre polynomial of cosine of x
c_p	$\sqrt{J(J+1)-K(K+1)}$	C^+_{JK} symbol appearing with the j_+ operator (see Table C.2). J is fixed.
c_m	$\sqrt{J(J+1)-K(K-1)}$	C^-_{JK} symbol appearing with the j_- operator (see Table C.2). J is fixed.
my1d		user supplied routine. See Note 8 of Table C.3.

Table C.2: Operator symbols which require no arguments. The expression n can be replaced by any positive integer. Finally, m is the mass of the relevant degree of freedom. In the MCTDH input, this mass is given by the reserved parameter `mass_`*modelabel*. If `mass_`*modelabel* is not explicitely set, it is 1 by default.

Symbol	Operator	Notes
dq	∂_x	first derivative. Cannot be used for rHO, Leg, KLeg, PLeg, Wigner or sphFBR.
dq^2	∂_x^2	second derivative. Cannot be used for Leg, KLeg, PLeg, Wigner or sphFBR.
p	$-i\partial_x$	momentum (deprecated, use dq)
KE	$-\frac{1}{2m}\partial_x^2$	Kinetic energy term. Cannot be used for modes with a Legendre DVR or sphFBR.
j^2	$-\sin^{-1}(\theta)\,\partial_\theta \sin(\theta)\,\partial_\theta - \sin^{-2}(\theta)\,\partial_\phi^2$	Angular momentum squared. In this form used for sphFBR and PLeg. For Leg and KLeg ∂_ϕ^2 is replaced by $-m^2$ (or $-K^2$).
j_p	$e^{i\phi}\,(\partial_\theta + i\cot(\theta)\,\partial_\phi)$	Angular momentum raising operator j_+. Only for KLeg and PLeg. For Wigner see below.
j_m	$e^{-i\phi}\,(-\partial_\theta + i\cot(\theta)\,\partial_\phi)$	Angular momentum lowering operator j_-. Only for KLeg and PLeg. For Wigner see below.
jpm	$C^+_{JK}\,j_+ + C^-_{JK}\,j_-$	Combined angular momentum operator. $C^\pm_{JK}$ are defined in Table C.1. Only for KLeg, PLeg and Wigner.
cjpm	$C^+_{JK}(j_{1,+} + j_{2,+}) + C^-_{JK}(j_{1,-} + j_{2,-})$	Combined operator for two angular momenta. Here $K = k_1 + k_2$, so this is different from jpm for the two individual angular momenta. Only for two successive KLegs, which furthermore have to be combined in one mode.
jz	$j_z = -i\partial_\phi$	Angular momentum operator. Only for sphFBR and KLeg. For PLeg use dq or p on the ϕ DOF.
jz^2	$j_z^2 = (-i\partial_\phi)^2$	second power of angular momentum operator j_z. Only for sphFBR and KLeg. For PLeg use dq^2 on the ϕ DOF.
jp^2	$(j_+)^2$	Square of angular momentum raising operator. Only for KLeg and PLeg.
jm^2	$(j_-)^2$	Square of angular momentum lowering operator. Only for KLeg and PLeg.
jpjm	$(j_+) * (j_-)$	Product of angular momentum raising and lowering operators. Only for KLeg and PLeg.

(continued)

Table 2, continued.

Symbol	Operator	Notes
jmjp	$(j_+) * (j_-)$	Product of angular momentum raising and lowering operators. Only for KLeg, PLeg.
sJp	$\sin(\theta) * J_+$	J is total angular momentum. Only for KLeg and PLeg.
sJm	$\sin(\theta) * J_-$	J is total angular momentum. Only for KLeg and PLeg.
sJpk	$(\sin(\theta) * J_+ * k + k * sin(\theta) * J_+)/2$	Only for KLeg and PLeg.
sJmk	$(\sin(\theta) * J_- * k + k * sin(\theta) * J_-)/2$	Only for KLeg and PLeg.
Jp	J_+	multiplication with C^-_{JK} and shift $k \to k-1$
Jm	J_-	multiplication with C^+_{JK} and shift $k \to k+1$
Jx	J_x	$J_x = (J_+ + J_-)/2$
Jy	iJ_y	$iJ_y = (J_+ - J_-)/2$
dth1	$\partial_\theta \, \sin\theta$	”first derivative”, only for Leg- KLeg- and PLeg-DVR (no symmetry).
dth2	$\frac{1}{2}(\cos\theta \, \partial_\theta \, \sin\theta + \partial_\theta \, \sin\theta \, \cos\theta)$	”first derivative”, only for Leg- KLeg- and PLeg-DVR (symmetry).
d2th1	$\cot\theta \, \partial_\theta \sin(\theta) \, \partial_\theta + \partial_\theta \, \sin\theta$	”second derivative”, only for Leg- KLeg- and PLeg-DVR (no symmetry).
qdq	$\frac{1}{2}(x \, \partial_x + \partial_x \, x)$	for rHO-DVR this replaces the ”first derivative”.
sdq	$\frac{1}{2}(\sin(x) \, \partial_x + \partial_x \, \sin(x))$	for cos-DVR this replaces the ”first derivative”.
sdq2	$\frac{1}{2}(\sin^2(x) \, \partial_x + \partial_x \, \sin^2(x))$	
cdq	$\frac{1}{2}(\cos(x) \, \partial_x + \partial_x \, \cos(x))$	
cdq2	$\frac{1}{2}(\cos^2(x) \, \partial_x + \partial_x \, \cos^2(x))$	
csdq	$\frac{1}{2}(\sin(x)\cos(x) \, \partial_x + \partial_x \, \sin(x)\cos(x))$	
udq	$\frac{1}{2}(\sqrt{1-x^2} \, \partial_x + \partial_x \, \sqrt{1-x^2})$	
uqdq	$\frac{1}{2}(\sqrt{1-x^2} \, x \, \partial_x + \partial_x \, \sqrt{1-x^2} \, x)$	
udq2	$\frac{1}{2}((1-x^2) \, \partial_x + \partial_x \, (1-x^2))$	

(continued)

Table 2, continued.

Symbol	Operator	Notes
j^2	$-\partial_\beta^2 - \cot(\beta)\,\partial_\beta - \sin^{-2}(\beta)\,[\partial_\alpha^2 + \partial_\gamma^2 - 2\cos(\beta)\,\partial_\alpha\,\partial_\gamma]$	Wigner-DVR angular momentum squared operator, with matrix elements: $j^2\,\lvert J,K,M\rangle = J(J+1)\,\lvert J,K,M\rangle$.
j_p	$(j_+)_{\mathrm{BF}} = e^{-i\gamma}\left(\frac{i}{\sin(\beta)}\partial_\alpha + \partial_\beta - i\cot(\beta)\,\partial_\gamma\right)$	Wigner-DVR body-fixed angular momentum lowering operator, which operates as: $(j_+)_{\mathrm{BF}}\,\lvert J,K,M\rangle = \sqrt{J(J+1) - K(K-1)}\,\lvert J,K-1,M\rangle$.
j_m	$(j_-)_{\mathrm{BF}} = e^{i\gamma}\left(\frac{i}{\sin(\beta)}\partial_\alpha - \partial_\beta - i\cot(\beta)\,\partial_\gamma\right)$	Wigner-DVR body-fixed angular momentum raising operator, which operates as: $(j_-)_{\mathrm{BF}}\,\lvert J,K,M\rangle = \sqrt{J(J+1)-K(K+1)}\,\lvert J,K+1,M\rangle$.
j_ps	$(j_+)_{\mathrm{SF}} = e^{i\alpha}\left(i\cot(\beta)\partial_\alpha + \partial_\beta - \frac{i}{\sin(\beta)}\partial_\gamma\right)$	Wigner-DVR space-fixed angular momentum raising operator, which operates as: $(j_+)_{\mathrm{SF}}\,\lvert J,K,M\rangle = \sqrt{J(J+1) - M(M+1)}\,\lvert J,K,M+1\rangle$.
j_ms	$(j_-)_{\mathrm{SF}} = e^{-i\alpha}\left(i\cot(\beta)\partial_\alpha - \partial_\beta - \frac{i}{\sin(\beta)}\partial_\gamma\right)$	Wigner-DVR space-fixed angular momentum lowering operator, which operates as: $(j_-)_{\mathrm{SF}}\,\lvert J,K,M\rangle = \sqrt{J(J+1) - M(M-1)}\,\lvert J,K,M-1\rangle$.
jpm	$C^+_{JK}\,(j_-)_{\mathrm{BF}} + C^-_{JK}\,(j_+)_{\mathrm{BF}}$	Wigner-DVR body-fixed combined angular momentum operator. $C^\pm_{JK}$ are defined in Table C.1.
jpms	$C^+_{JM}\,(j_+)_{\mathrm{SF}} + C^-_{JM}\,(j_-)_{\mathrm{SF}}$	Wigner-DVR space-fixed combined angular momentum operator. $C^\pm_{JM}$ are defined in Table C.1, but here M replaces K.
jp^2	$(j_+)^2_{\mathrm{BF}}$	Wigner-DVR squared body-fixed angular momentum raising operator.
jp^2s	$(j_+)^2_{\mathrm{SF}}$	Wigner-DVR squared space-fixed angular momentum raising operator.
jm^2	$(j_-)^2_{\mathrm{BF}}$	Wigner-DVR squared body-fixed angular momentum lowering operator.
jm^2s	$(j_-)^2_{\mathrm{SF}}$	Wigner-DVR squared space-fixed angular momentum lowering operator.

(continued)

Table 2, continued.

Symbol	Operator	Notes
jpjm	$(j_+)_{\mathrm{BF}} * (j_-)_{\mathrm{BF}}$	Wigner-DVR product of body-fixed angular momentum raising and lowering operators.
jpjms	$(j_+)_{\mathrm{SF}} * (j_-)_{\mathrm{SF}}$	Wigner-DVR product of space-fixed angular momentum raising and lowering operators.
jmjp	$(j_-)_{\mathrm{BF}} * (j_+)_{\mathrm{BF}}$	Wigner-DVR product of body-fixed angular momentum lowering and raising operators.
jmjps	$(j_-)_{\mathrm{SF}} * (j_+)_{\mathrm{SF}}$	Wigner-DVR product of space-fixed angular momentum lowering and raising operators.

Notes to Table C.2

The volume-element assumed for Leg/KLeg/PLeg/Wigner is $\sin\theta\, d\theta$, whereas all other DVRs in MCTDH assume the simple volume-element dq. Because of the non-trivial volume-element, ∂_θ is not an anti-hermitian operator, only $\partial_\theta \sin\theta$ (i.e. dth1) is. Note that $\sin\theta\, \partial_\theta = \partial_\theta \sin\theta - \cos\theta = \mathrm{dth1} - \cos\theta$.

The operators j_p, j_m, jpm, jp^2 and jm^2 are (KLeg or PLeg) 2D mode-operators. Similarly, j^2 becomes a 2D mode-operator, when operating on a KLeg or PLeg mode. Similarly, the operators sJp, sJm, sJpk, and sJmk are also 2D KLeg/PLeg operators, where J denotes the total angular momentum.

The 1D operators jz, jz^2, operate on the k-dof of the KLeg mode only. The 2D mode operator j_p performs a similar multiplicative and shift operation on k, but additionally performs a derivative and k-dependent multiplicative operation on the θ-dof of the KLeg mode. Similar operations are done by the j_m, sJp, sJm, sJpk, and sJmk operators.

When applied to Wigner functions, the operators j^2, j_p, j_m, jpm, jp^2, jm^2, jpjm, and jmjp are 3D mode operators and are represented as 4D tensors in MCTDH, so care must be taken when multiplying these operators with other operators. The Wigner operators j_p, j_m, jpm, jp^2, jm^2, jpjm, and jmjp operate in the BODY-fixed axis system; that is, these operators perform multiplicative operations and shifts depending on the k (second) degree of freedom in the combined 3D mode. The corresponding SPACE-fixed operators, which perform multiplications and shifts depending on the m (third) DOF, are denoted j_ps, j_ms, jpms, jp^2s, jm^2s, jpjms, and jmjps. Note that j_+ and j_- are defined as $j_+ = j_x + ij_y$ and $j_- = j_x - ij_y$ for both the SF- and BF-system. Due to the anomalous commutation relation for the BF operators, j_p = $(j_+)_{BF}$ *decreases* k by one, whereas j_ps = $(j_+)_{SF}$ increases m by one.

The operator cjpm is a 4D mode-operator (two successive KLegs). It only works if the two KLegs are combined into one mode, and for this case it replaces the use of natural potentials of the cpp/cmm surfaces (see Table C.5).

Note that the operators j_p, j_m, jpm, cjpm, jp^2, jm^2, sJp, sJm, sJpk and sJmk — as well as j^2 if the latter operates on a KLeg/PLeg combined mode — are 3D tensors in MCTDH and not matrices. Hence care must be taken when multiplying these operators with other operators. To give an example

```
HAMILTONIAN-SECTION
-----------------------------------
  modes    | ... |  theta   |  k
-----------------------------------
 ...       | ... | cos*j_p | c_p
 ...       | ... | j_p*cos | c_p
-----------------------------------
end-hamiltonian-section
```

is a valid construct. Note that the 2D operator j_p may be multiplied from right or left with and operator operating on θ only. However, it may be multiplied only from right with a local k-dependent function (here c_p).

The operators dth1, dth2, qdq, and sdq replace the first derivative operator for the Leg (and KLeg/PLeg), rHO, cos (and sin when the keyword sdq is given) DVR, repectively. In these cases the operator dq cannot be used.

Table C.3: One-dimensional operators which require arguments. The expression x stands for the coordinate, p can be replaced by any parameter from the PARAMETER-SECTION, or any real number. The exponent r can be any real number. If $r = 1$ it is not required, e.g. $q[p]$ and $q[p]$ˆ1 are synonemous.

Symbol	Operator
q[p]ˆr	$(x-p)^r$
qs[p]ˆr	$\left(\sqrt{p-x^2}\right)^r$
sin[p1,p2]ˆr	$\sin^r(p1(x-p2))$
cos[p1,p2]ˆr	$\cos^r(p1(x-p2))$
tan[p1,p2]ˆr	$\tan^r(p1(x-p2))$
exp[p1,p2]ˆr	$\exp^r(p1(x-p2))$
Exp[p1,p2]ˆr	$\exp^r(i*p1(x-p2))$
sinh[p1,p2]ˆr	$\sinh^r(p1(x-p2))$
cosh[p1,p2]ˆr	$\cosh^r(p1(x-p2))$
tanh[p1,p2]ˆr	$\tanh^r(p1(x-p2))$
cos1[p1,p2]ˆr	$(\cos(p1*x)-\cos(p1*p2))^r$
exp1[p1,p2]ˆr	$(1-\exp(p1(x-p2)))^r$
expcos[p1,p2]ˆr	$(\exp(p1\cos(x))-\exp(p1\cos(p2)))^r$
expcos1[p1,p2]ˆr	$\exp(p1(\cos(x)-p2))^r$
qtanh[p1,p2,p3]ˆr	$\tanh^r(p2(\arccos(x)-p1)^{p3})$
motanh[p1,p2,p3,p4]ˆr	$\tanh^r(p3[1-\exp(-p1(x-p2))]^{p4})$
asin[p1,p2,p3]ˆr	$(\arcsin(p1*x-p2)-p3)^r$
acos[p1,p2,p3]ˆr	$(\arccos(p1*x-p2)-p3)^r$
atan[p1,p2,p3]ˆr	$(\arctan(p1*x-p2)-p3)^r$
coschirp[p1,p2,p3]ˆr	$\cos^r(x[p2+(p1-p2)\exp(-(x/p3)^2)])$
tgauss[p1,p2]ˆr	$\exp(-p1(\arccos(x)-p2)^2)^r$
gauss[p1,p2]ˆr	$\exp(-p1(x-p2)^2)^r$
ngauss[σ, x_0]ˆr	$(2\pi\sigma^2)^{-1/2}\exp(-(x-x_0)^2/(2\sigma^2))^r$
morse[p1,p2,p3,p4,p5]	Morse function. See Note 1.
morse1[p1,p2,p3,p4]	Morse function. See Note 1.
CAP[p1,p2,p3,p4]	$-iW$. See Note 2.
ACAP[p1,p2,p3,p4,p5]	$-iW$. See Note 3.
lct_elec[strength]	Local control function (see note 11).
lct_vibr[strength]	Local control function (see note 12).
lct{operator,parameter}	Local control function (see note 13).

(continued)

Table 3, continued.

Symbol	Operator
step[p]	$\Theta(x-p)$ Step function. See Note 4.
rstep[p]	$\Theta(p-x)$ Reverse step function. See Note 4.
charfun[p1,p2]	characteristic function: if $x \in [p1, p2]$ then charfun=1 else it is zero.
regcoul[p1,p2]	regularized coulomb function: $1/\sqrt{(x-p1)^2+p2}$.
switch1[p1,p2]	$0.5*[1-\tanh(p1(x-p2))]$
switch2[p1,p2]	$0.5*[1+\tanh(p1(x-p2))]$
low$[m,\omega,s]$	lowering operator. See Note 5.
rai$[m,\omega,s]$	raising operator. See Note 5.
num$[m,\omega,s]$	number operator. See Note 5.
ramorse$[m,\omega,\Lambda,\alpha,z_0]$	raising operator for Morse potential. See Note 6.
lwmorse$[m,\omega,\Lambda,\alpha,z_0]$	lowering operator for Morse potential. See Note 6.
cspot$[J,K,csmax,m]$	centrifugal potential. See Note 7.
external1d{*file*}	external 1D function read from file *file*. See Note 8.
read1d{*file F*}	external 1D function read from file *file* of format *F*. See Note 9.
my1d[p1,p2,p3,p4,p5]	user supplied routine. See Note 10.
flux$[x_c, power]$	Flux operator for Cartesian kinetic energy. x_c = location of dividing surface, $power$ = exponent of smoothing function. See Note 11.
pgauss$[\sigma, x_0]$	Projector $\|G><G\|$, where G denotes a L^2 normalized Gauss $G=(2\pi\sigma^2)^{-1/4}\,exp[-(x-x_0)^2/(4\sigma^2)]$
shift$[\Omega]$	simple shift on the grid by Ω, $\tilde{\psi}(x_i)=\psi(x_{i-\Omega})$ $\tilde{\psi}(x_i)=0$ if $i-\Omega<1$ or $i-\Omega>N$ For a periodic shift on a periodic grid (exp-DVR of FFT) use shift$[\Omega]$ <u>and</u> shift$[\Omega-N]$ (with $\Omega>0$).

Table C.4: One-dimensional potential energy curves.

Symbol	Potential Curve
v:NO	NO potential curve
v:H2	H_2 potential (link lsth)
vbmkp:H2	H_2 potential (link h4bmkp)
v:HO	OH potential, morse function from h2o.f
v:OH	OH potential (link hoosrf)
v:CH	CH potential (link c2h)
v:C2	C_2 potential (link c2hasec)
v:OF	OF potential curve
vrho:H3	H+H_2 potential in hypersphaerical coordinates, theta=π (link lsth)
vthe:H3	H+H_2 potential in hypersphaerical coordinates, rho=2.484773 (link lsth)
vdj:000	expansion coeffiecent V_{000} for DJ H_4 surface (link h4dj)
vdj:022	expansion coeffiecent V_{022} for DJ H_4 surface (link h4dj)
vdj:224	expansion coeffiecent V_{224} for DJ H_4 surface (link h4dj)

Table C.5: Two-dimensional operators (used for molecule-surface scattering).

Symbol	Operator
coshcosth[p]	$\cosh(p * \cos(\theta))$
sinhcosth[p]	$\sinh(p * \cos(\theta))$
cossinthcosphi[p]	$\cos(p * \sin(\theta) * \cos(\phi))$
cossinthsinphi[p]	$\cos(p * \sin(\theta) * \sin(\phi))$
sinsinthcosphi[p]	$\sin(p * \sin(\theta) * \cos(\phi))$
sinsinthsinphi[p]	$\sin(p * \sin(\theta) * \sin(\phi))$
reY[l,m]	$\mathrm{Re}(Y_l^m(\theta, \phi))$
imY[l,m]	$\mathrm{Im}(Y_l^m(\theta, \phi))$

Two-dimensional C_+, C_- symbols defined on truncated k_1,k_2 grid.
See also "Hamiltonian/Liouvillian Documentation"/"Available Surfaces"

Symbol	Operator
cpp{jtot=J,dim=d}	$\sqrt{J(J+1) - (\sum_{j=1}^{d} k_j)(\sum_{j=1}^{d} k_j + 1)}$
cmm{jtot=J,dim=d}	$\sqrt{J(J+1) - (\sum_{j=1}^{d} k_j)(\sum_{j=1}^{d} k_j - 1)}$

Table C.6: Some general multi-dimensional operators. Here parameters are to be given in curly brackets. E.g: coulomb1d{a=1.3 b=2.0 c=0.0 d=1.5}. See also "Hamiltonian Documentation"/"Available Surfaces"

Symbol	Operator
readsrf{*file* F}	Potential values on grid points are read from file *file* of format F. (F=`ascii` or `binary`. See HTML Docu For 1D-potential use read1d).

gauss1d{width=w S}	$\exp(-0.5((x_1 - x_2)/w)^2)/(\sqrt{2\pi}\, w)$
	If the optional string S is set to `periodic`,
	then a 2 π periodic grid is assumed.
	In this case the DVR lines should read e. g.:
	`x1 FFT 128 2pi`
	`x2 FFT 128 2pi`

gauss2d{width=w}	$\exp(-0.5([(x_1 - x_2)/w]^2 + [(y_1 - y_2)/w]^2))/(2\pi\, w^2)$
coulomb1d{$\cdots$}	$1/\sqrt{(a\, x_1 - b\, x_2 + c)^2 + d}$

Table C.7: One-dimensional operators for treating symmetric double-well potentials by mapping each side on an artificial (single-set) electronic state. Note, the grid must be a sin-DVR, which, when doubled, lies symmetrically to zero but does not contain zero. The differential operators which are truncated are firstly defined on this doubled grid, but then projected to the working grid.

Symbol	Operator	Notes
Rf	Rf $\varphi(x_i) = \varphi(x_{N+1-i})$	Reflection operator. (Must not be multiplied with other operators).
Rfm	Rfm $\varphi(x_i) = -\varphi(x_{N+1-i})$	Reflection operator. (Must not be multiplied with other operators).
hKEh	step*KE*step	Truncated kinetic energy
hFRh	step*KE*Rf*step	Truncated kinetic energy times reflection
hdqh	step*dq*step	Truncated first derivative
hdqRh	step*dq*Rf*step	Truncated first derivative times reflection
dqR	dq*R	First derivative times reflection
dq2R	dq^2*R	Second derivative times reflection

Notes to Table C.3

All input variables $[\cdots]$ are numbers, parameters or arithmetic expressions containing numbers and parameters. (See Hamiltonian-Documentation/Parameter-Section for details). The use of units is not allowed here. Note that these symbolic expressions with parameters must not appear in a HAMILTONIAN-SECTION. They rather have to be linked to simple symbols (without parameters) in a LABELS-SECTION. (Compare with Example 6.4).

1. A morse curve can be given by $D\,(\exp(-\alpha(x-x_0))-1)^2 + E_0$, where D is the dissociation energy (depth parameter), α defines the curvature, x_0 the equilibrium position, and E_0 is an energy shift parameter. If one uses the symbol morse1 these are precisely the input parameters, i.e. $[D, \alpha, x_0, E_0]$. For the symbol morse the input parameters are $[D, \omega, x_0, ex_0, m]$, where m is the mass, ω is the frequency of the related harmonic oscillator, and ex_0 is the position at which the potential is zero. Note that ω and α are related by $\alpha^2 = m\omega^2/2D$, while ex_0 and E_0 are related by $E_0 = -D(exp(-\alpha(ex_0 - x_0)) - 1)^2$.

2. A CAP (Complex Absorbing Potential) is an imaginary, negative potential, used to absorb a wavepacket as it approaches the end of the grid. It is defined as $-iW$, where $W = \eta\,\Theta(k(x-x_0))\,(k(x-x_0))^n$ and where Θ denotes the Heaviside's step function. The input parameters are $[x_0, \eta, n, k]$, where k is used to choose to which end of the grid the CAP is placed: $k = -1$ puts the CAP at the left, and $k = 1$ and the right of the grid. $k = 1$ is default and may be left out.

3. ACAP symbolises an automatic CAP. The ACAP is useful, when one wants to place the initial wavepacket at a position, where it overlaps with the CAP. The ACAP remains disabled as long as the wavepacket overlaps with the CAP. The ACAP is enabled only when the wavepacket starts to re-enter the region where the CAP is defined. There is a fifth parameter: *timecap*. If this optional parameter is set, the ACAP will remain disabled at least as long as *time* $<$ *timecap*, where *time* is the propagation time in fs. The parameter *timecap* is useful, because the automatic enabling of the CAP may sometimes happen too early. The time, at which the ACAP is switched on, is protocoled in

the log file. Use this information to set the option `-lo` in **flux83** appropriately. When **flux83** is run, it must not evaluate matrix elements of the CAP for times, at which the CAP is switched off.

4. The symbols step and rstep symbolise a Heaviside's step function and the reverse of it. I. e. step[p] = $\Theta(x-p)$ and rstep[p] = $1-\Theta(x-p)=\Theta(p-x)$.

5. The lowering operator corresponding to a harmonic oscillator is given by

$$b = \frac{i}{\sqrt{2m\omega}}p + \sqrt{\frac{m\omega}{2}}(q-q_0)\,,$$

where p denotes the momentum operator, q denotes the position operator, m is the mass, ω is the frequency, and q_0 is the equilibrium position. The input parameters are $[m,\omega,s]$, which means the mass m, the frequency ω, and the shift

$$s = -\sqrt{\frac{m\omega}{2}}q_0\,.$$

(Note the minus sign). The corresponding raising operator is given by $b^\dagger$, and the number operator by $\hat{n}=b^\dagger b$. The parameters have the same meaning as for lowering operators. NB. The lowering, raising and number operator require the use of a simple DVR with an ordinary first derivative, e.g. sin, HO, or exp but not FFT, rHO, Leg, KLeg, PLeg or sphFBR.

6. The (approximate) raising/lowering operators (R/L) for a Morse Hamiltonian

$$H = p^2/(2m) + D\left(e^{-2\alpha(x-x_0)} - 2e^{-\alpha(x-x_0)}\right)$$

are defined as

$$L = \sqrt{\frac{\hbar}{2m\omega}}\left[\left(\Lambda-\frac{1}{2}\right)\alpha - \Lambda\alpha e^{-\alpha(x-x_0)} + \frac{i}{\hbar}p\right]$$

and

$$R = L^\dagger$$

with $\omega=\alpha\sqrt{2D/m}$ and $\Lambda=\sqrt{2Dm}/\hbar\alpha$

7. The centrifugal potential given by:

$$V_{cent}(x) = min\left(\frac{J(J+1)-2K^2}{2mx^2}\ ,\ csmax\right)\,.$$

8. An arbitrary (real) 1D–function may be defined through a set of points. The points are read from file *file* and are then interpolated to define a general 1D–function. The data is in free format with one (x,y) data pair per line. Blank lines and lines which start with a # are ignored. Currently, the x-data (called *time* in the code) must increase linearly, i.e must be equally spaced.

   An arbitrary number of these, with different data files, may be used in one operator. However, when using multiple instances of the same external1d function, do this by defining a label and referring to it in the operator rather than declaring external1d{*file*} with the same file *file* repeatedly, as this wastes buffer memory due to duplication.

9. An arbitrary (real) 1D–function may be defined through a set of points. The points must coincide with the grid points. The potential values on the grip points are read from file *file*, one value per line. The file may be in ascii or binary format. (binary is default). Give `ascii` or `binary` as second argument after *file*. (For multi-dimensional surfaces use *readsrf*).

10. A (real) 1D–function may be defined through a user written subroutine. Edit the subroutine **my1d** on `$MCTDH_DIR/source/opfuncs/func1d.F` .

11. The flux operator $[\Theta, T]$ is set up in a sine or exponential basis and then transformed to DVR representation. This operator might be used with *eigenf* to produce flux-eigenstates as initial wavefunctions. To regularize the flux operator and to make its eigenfunctions more localized, it is multiplied from right and left with $[\cos(\pi p/2p_{max})\cosh(\pi p/2p_{max})]^{power}$. The exponent $power$ may be zero or any positive real number. $power = 1$ is recommended.

12. The lct_elec[strength] operator initiates the calculation of a local control field to maximise or minimise the population of an electronic state. The field is calculated as a function of $E(t) = \pm\lambda \sum_i \mathrm{Im}\langle\psi_i(\boldsymbol{Q},t)|\mu_{ik}|\psi_k(\boldsymbol{Q},t)\rangle$. The Hamiltonian for the expectation value of the transition dipole moment is written:

    dipole | h_1 |ZSf&Si

    h_1 is usually q. Sf is the final state and Si is the initial state. If the strength parameter is positive Sf is maximised, if it is negative Sf is minimised.

13. The lct_vibr[strength] operator initiates the calculation of a local control field to maximise or minimise the population of a vibrational state. The field calculated in the same manner as note 11, but is a sum over all vibrational states in an electronic state and along a vibrational mode. The Hamiltonian for the expectation value of the transition dipole moment is written:

    dipole | h_1 | S1&1

    h_1 is usually q. This also requires a $lctvib = t, s, m$ in the run-section of the input. t=vibrational target state, s=electronic state of interest and m is vibration mode of interest.

14. The local control operator *lct{operator,parameter}* allows a field to be generated using local-control theory. The field is a response to the time-evolution of a desired observable, and is calculated from the expectation value of the commutator of the observable with the dipole moment operator. *operator* specifies the operator to be evaluated in the expectation, and refers to the operator defined in HAMILTONIAN-SECTION_OPERATOR. *parameter* refers to a parameter in the labels-section that controls the size of the field. See the HowTo on local control for more details.

Special operators

There is a number of operators especially defined for the methyl-iodine (CH3I) system. Their labels all start with MI: . See opfuncs/ch3i.f and opfuncs/ch3igrd.f for further information.

Non-adiabatic operators

If the system contains more than one electronic state, the Hamiltonian can be written in matrix form, i.e.

$$\hat{H} = \begin{pmatrix} H_{11} & H_{12} & \dots \\ H_{21} & H_{22} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{C.1}$$

To input such a form, the symbols in Table C.8 can be used. Thus the operator

$$h_1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + h_2 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + h_3 \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{C.2}$$

can be represented symbolically as

```
modes   | X   | el
1.0     | h_1 | 1
1.0     | h_2 | S1&2
1.0     | h_3 | Z1&2
```

See also Sec. 9 for more examples.

Table C.8: Matrix operator symbols, used for an electronic degree of freedom.

Symbol	Operator
Sf&i	Symmetric matrix element
Zf&i	Unsymmetric matrix element
1	Unit matrix

Appendix D

Structure of the WF array

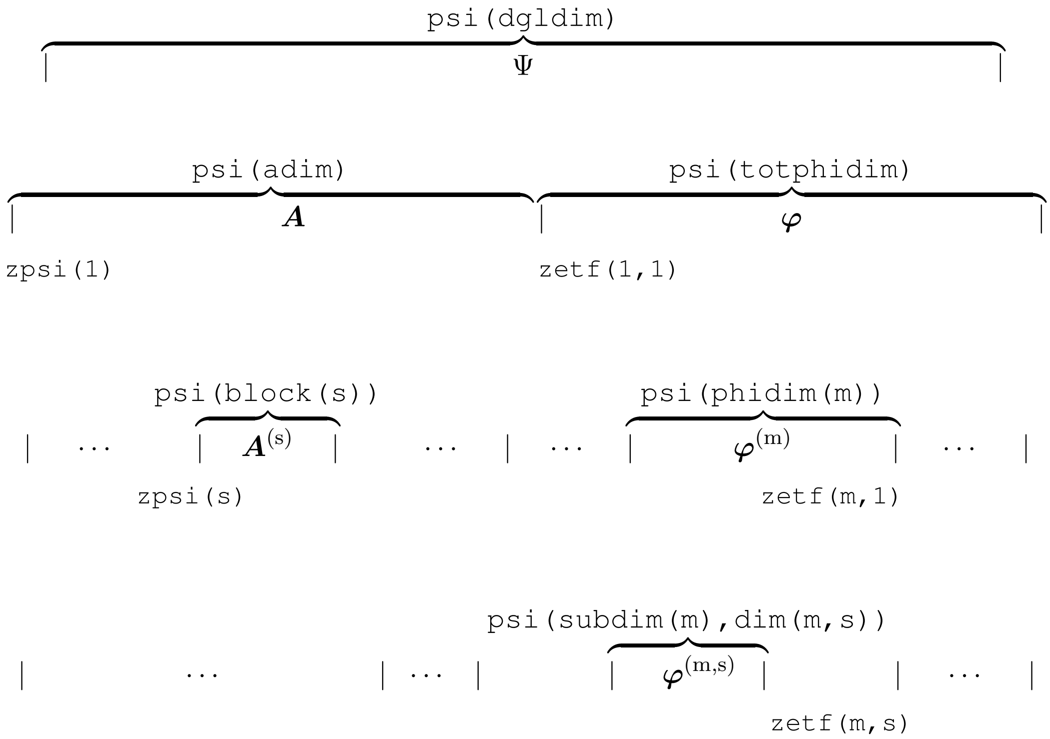

Figure D.1: The structure of the wave function

Appendix E

Installing the MCTDH package

The installation of the MCTDH package is is very easy if you install it on a PC with a not too old Linux system. On more fancier machines one may need to edit the compile scripts (to set compiler options appropriately) and/or install some open-source software like gnuplot or a GNU make. It is essential to have a bash shell (version 3.x is recommended, but lower versions may also work) and it is highly recommended that you work under bash, although it is possible to work under C-shell or kshell as well. A bash, however, must exist as there are several bash scripts. Moreover, for some tasks like OCT or Cluster Expansion python scripts are used. Hence, python 2.4 or higher (but NOT python 3) should be installed.

One should begin with creating a directory MCTDH which eventually will contain all the MCTDH stuff, but not – at least I prefer to do so – the output of production runs. Move the MCTDH tar-ball to the MCTDH directory and unzip and untar it. I. e.

```
mkdir /home/muser/MCTDH
cd /home/muser/MCTDH
mv <path>/mctdh84.x.tgz .
tar xzvf mctdh84.x.tgz
```

Here it is assumed that your login name is muser and that you have a GNU tar. If you do not have a GNU tar you first have to gunzip the tar-ball and then untar it (without the option z). The symbol x stands for the revision number of the particular package which was downloaded. When the tar command is finished, there should exist a directory mctdh84.x under /home/muser/MCTDH .

If you are familiar with Subversion (svn), it will be more convenient to download the code form the svn-repository of the Heidelberg MCTDH package. For details see Appendix **??**.

After the code is downloaded, move to the directory mctdh84.x/install and run **check_system** :

```
cd mctdh84.x/install
./check_system
```

This will create an output like

```
*********************************************************************
***************    -------  CHECK SYSTEM  -------    ***************
*********************************************************************

Mon Dec 9 14:42:09 CET 2013

The path of the MCTDH-directory is: /home/dieter/MCTDH/mctdh84.9

System           : Linux
Platform         : i686
Operating-System : GNU/Linux
Machine          : cauchy
Processor        : unknown
Kernel           : Linux

This is a 32-bit system.

These are the variables determined by platform.cnf
MCTDH_PLATFORM = i686
MCTDH_COMPILER = gfortran
MCTDH_GFORTRAN_VERSION = 4.7.2

These are the default compilers
Fortran compiler: gfortran
C compiler      : gcc
make command    : make

If you want to use other compilers please edit platform.cnf* and
possibly also compile.cnf* (or use the -c option when compiling).

The Fortran compiler gfortran is /apps/gcc-4.7/bin/gfortran
The C compiler gcc is /apps/gcc-4.7/bin/gcc
The make command  make  is /usr/bin/make
Congratulation, you have a GNU make.

GNUPLOT exist on your system: gnuplot 4.4 patchlevel 0

Python exist on your system: Python 2.6.6

Your bash is: GNU bash, version 4.1.5(1)-release

Distributed memory parallelization with MPI seems to be possible!
For this use Fortran compiler consistent with:
gcc version 4.7.2 (GCC)

Shared memory parallelization with POSIX-Threads seems to be possible!

The use of the NUMA library for the shared memory parallelization
with POSIX-Threads seems to be possible!
Compile with "-u" option to enable the use of NUMA. (See "compile -h").

The standard installation should work without problems.
If not already done, you should now run "./install_mctdh".
```

```
*******************************************************************
**************  -----   Finished System Check  -----  **************
*******************************************************************
```

This is just for fun, but if you read the message "The standard installation should work without problems", then there should be no problems. By the way, please note that the GNU compilers GCC 3.4.0 – 3.4.3 and GCC 4.0.0 – 4.1.0 are buggy, DO NOT USE THEM. We recommend to use GNU-GCC 4.4.2 or higher. For the 8.5-branch, where we start to use some fancy features of FORTRAN95/2003 we recommend to use even higher versions, e.g. 4.7.2, however, currently (2013) 4.4.2 is still sufficient. Next to GNU, the intel, pgi, and some other compilers are also fine. See install/compile.cnf for possible compilers. (You may edit the compile scripts and add new compilers to it).

Please read the README file of the install directory and start with the installation, i. e. type

```
./install_mctdh
```

while being in the install directory. The script will ask you several questions and in general you should answer yes, i. e. type `y`. (In fact, one may run `yes | ./install_mctdh` but this is not recommended for beginners.) Please, try to understand all the questions before giving an answer. If you think you have made a wrong choice, you can always stop the installation process with the `ctrl c` command and then start anew. The **install_mctdh** script LaTeX compiles the guide, compiles the code, sets some environment variables, and writes the path of the MCTDH directory to your ./bashrc file (or some other configuration file if you are not running under bash). Note that you have to source your ./bashrc (or other configuration) file to activate the changes. The path to the MCTDH directory is stored in the environment variable `MCTDH_DIR`, to inspect it type `echo $MCTDH_DIR`. But it is more convenient to run the script **menv**, which writes a list of all MCTDH environment variables to screen.

If you are running under bash the install script will also create the ~/.mctdhrc file. This enables the powerful **cdm** command (try `cdm -h`) and sets a link ~/mctdh which points to the currently active MCTDH directory, in the present case to /home/muser/MCTDH/mctdh84.x . (One may install several MCTDH packages, but only one will be active. Use the script **minstall** to switch between the different versions.)

If you want to make use of the surfaces library, move the file addsurf.tgz to your MCTDH directory, i. e. in our example to /home/muser/MCTDH, and untar it. Then you should edit the ~/.mctdhrc. In this case simply remove the # in front of `export MCTDH_ADDSURF` and source .mctdhrc to activate this change. Similarly, one can set environment variables in .mctdhrc which point to the MCTDH backup and elk directories (for users who want to change the code). Use the script **mklinks** to set a link to the PES requested. However, one should do so only when a PES is needed. After a PES is linked to the $MCTDH_DIR/source/surfaces directory, one has to compile mctdh or potfit with the `-i` option, e. g. execute the commands

```
mklinks h4bmkp
compile -i h4bmkp potfit
```

before running **potfit84** to bring the BMKP surface of the H_4 system to product form.

Load the URL `file:///home/muser/mctdh/doc/index.html` into your browser to inspect the HTML on-line documentation. Bookmark this page! A simple but quick help is provided through the script **mhelp**. It briefly explains the keywords of the input file. Try `mhelp -h`. (All MCTDH scripts and programs know the help option `-h`). If you want to inspect the code, try the scripts **mcb**, **mcg**, **mcl**, and **phelp**.

You may go to the AdvancedUser directory (e. g. type `cdm Ad`) and execute **make** there (read the README file first). This will give you access to additional scripts and routines (**mcalc** is quite useful).

There are four compile-configuration files on the install directory: compile.cnf_le, compile.cnf_be, compile.cnf_lenp, and compile.cnf_benp. The letters `le` and `be` stand for `little endians` and `big endians`, respectively, `np` denotes `no parallelization`. The **install_mctdh** script copies one of these four configuration files to compile.cnf, the default choice is compile.cnf_le. The file compile.cnf is then read by the **compile** script which is used to compile individual programs (e. g. `compile mctdh`) or the full package (`compile all`). If your compiler does not support pthreads, you have to choose compile.cnf_lenp or compile.cnf_benp. We have tried to find reasonable options for the compilers (see in particular `MCTDH_FFLAGS_OPT`), but we cannot account for any hardware and software installation on which MCTDH may run. Hence, depending on your particular hardware/software installation, the choice of compiler options may not be optimal. (To inspect the compiler option run `compile config`). Feel free to adjust the compiler options to your particular installation. If you add a new compiler, please send us the updated compile.cnf file.

If you are working on a system where computers of different kind (32 bit / 64 bit, Linux / other Unix) are interconnected by a common file system, you may store and install the MCTDH package only once, but run **compile** on each kind. MCTDH is smart and will load automatically the correct executables. Run **menv** on interconnected computers of different kind and you will see that the paths are set differently.

In general, each MCTDH user works with his own package. This allows him to change the code according to his demand. However, sometimes it may be wanted that several users have access to the same package. In this case there is a master-user who installs the package and clients who only need to add the line

```
source $MCTDH_DIR/install/MCTDH_client
```

to their .bashrc. The file MCTDH_client is generated during installation. Of course, `$MCTDH_DIR` must be replaced by the full path of the MCTDH directory, which for the present example reads /home/muser/MCTDH/mctdh84.x. Alternatively, the clients may simply copy the file MCTDH_client to their .bashrc.

If the automatic detection of platform and compiler does not work, one has to edit the platform.cnf script. (Note that during installation platform.cnf.def is copied to platform.cnf. Hence one may whish to edit platform.cnf.def as well). Around line 100, platform.cnf reads:

```
#-----------------------------------------------------------------------
# SET MACHINE-DEPENDENT OPTIONS
#-----------------------------------------------------------------------

system=`uname -s`
## system=MYSYSTEM   # Incomment, if automatic dection doesnt work.
case $system in
  MYSYSTEM)  # Here you may set the variables by hand.
    MCTDH_PLATFORM=      # Please set! (try `uname -p` or `uname -m`)
    MCTDH_COMPILER=      # Please set! (if not listed in compile.cnf, you
    ;;                   # have to edit compile.cnf as well.)
```

Change this to e. g.:

```
#------------------------------------------------------------------------
# SET MACHINE-DEPENDENT OPTIONS
#------------------------------------------------------------------------

system=`uname -s`
system=MYSYSTEM   # Incomment, if automatic dection doesnt work.
case $system in
  MYSYSTEM)  # Here you may set the variables by hand.
    MCTDH_PLATFORM=cruncher
    MCTDH_COMPILER=pgf77
    ;;
```

This is, of course, just an example. One can give any name to MCTDH_PLATFORM, a convenient choice is the output of **uname -p** or **uname -m**. The symbol which is given to MCTDH_COMPILER must be listed in the compile.cnf file. If one wants to use a compiler, which is not listed there, one has to edit compile.cnf to add the new compiler. (Note that during installation one of the files compile.cnf_⋆ is copied to compile.cnf, where ⋆ stand for le, be, lenp, or benp. Hence one may whish to edit those files as well.)

Finally, let us summarize the commands you now should be familiar with: **menv**, **compile**, **mhelp**, **cdm**, and, if you use a PES from addsurf, **mklinks**. Try the help option -h and inspect the HTML on-line documentation "The Analyse Programs / Utility Scripts" to learn more about the utility scripts. If you want to inspect the code, make yourself familiar with **mcb**, **mcg**, **mcl**, and **phelp**. There is also a backup facility and an automatic program test (Elk Test), see the HTML on-line documentation for details. For additional information on the install process see the page "Installation and Compilation" of the HTML on-line documentation.

When the installation is completed, it is advisable to work through the tutorial (Sec. 2).

List of MCTDH references

[1] M. H. Beck, A. Jäckle, G. A. Worth, and H.-D. Meyer. The multiconfiguration time-dependent Hartree method: A highly efficient algorithm for propagating wavepackets. *Phys. Rep.* **324** (2000), 1–105.

[2] H.-D. Meyer, U. Manthe, and L. S. Cederbaum. The multi-configurational time-dependent Hartree approach. *Chem. Phys. Lett.* **165** (1990), 73–78.

[3] U. Manthe, H.-D. Meyer, and L. S. Cederbaum. Wave-packet dynamics within the multiconfiguration Hartree framework: General aspects and application to NOCl. *J. Chem. Phys.* **97** (1992), 3199–3213.

[4] H.-D. Meyer and G. A. Worth. Quantum molecular dynamics: Propagating wavepackets and density operators using the multiconfiguration time-dependent Hartree (MCTDH) method. *Theor. Chem. Acc.* **109** (2003), 251–267.

[5] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum. The effect of a model environment on the S_2 absorption spectrum of pyrazine: A wavepacket study treating all 24 vibrational modes. *J. Chem. Phys.* **105** (1996), 4412.

[6] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum. Relaxation of a system with a conical intersection coupled to a bath: A benchmark 24-dimensional wavepacket study treating the environment explicitly. *J. Chem. Phys.* **109** (1998), 3518–3529.

[7] A. Raab, G. Worth, H.-D. Meyer, and L. S. Cederbaum. Molecular dynamics of pyrazine after excitation to the S_2 electronic state using a realistic 24-mode model Hamiltonian. *J. Chem. Phys.* **110** (1999), 936–946.

[8] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum. State filtering by a bath: Up to 24 mode numerically exact wavepacket propagations. *Chem. Phys. Lett.* **299** (1999), 451.

[9] A. Jäckle and H.-D. Meyer. Time-dependent calculation of reactive flux employing complex absorbing potentials: General aspects and application within MCTDH. *J. Chem. Phys.* **105** (1996), 6778.

[10] A. Jäckle and H.-D. Meyer. Calculation of $H+H_2$ and $H+D_2$ reaction probabilities within the multiconfiguration time-dependent Hartree approach employing an adiabatic correction scheme. *J. Chem. Phys.* **109** (1998), 2614.

[11] M. H. Beck and H.-D. Meyer. Extracting accurate bound-state spectra from approximate wave packet propagation using the filter-diagonalization method. *J. Chem. Phys.* **109** (1998), 3730–3741.

[12] M. H. Beck and H.-D. Meyer. Efficiently computing bound-state spectra: A hybrid approach of the multiconfiguration time-dependent Hartree and filter-diagonalization methods. *J. Chem. Phys.* **114** (2001), 2036–2046.

[13] T. Sommerfeld, H.-D. Meyer, and L. S. Cederbaum. Potential energy surface of the CO_2^- anion. *PCCP* **6** (2004), 42–45.

[14] D. J. Tannor, V. Kazakov, and V. Orlov. Control of photochemical branching: Novel procedures for finding optimal pulses and global upper bounds. In *Time Dependent Quantum Molecular Dynamics*, J. Broeckhove and L. Lathouwers, Eds. Plenum, New York, 1992, pp. 347–360.

[15] W. Zhu and H. Rabitz. A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator. *J. Chem. Phys.* **109** (1998), 385.

[16] J. P. Palao and R. Kosloff. Optimal control theory for unitary transformations. *Phys. Rev. A* **68** (2003), 062308.

[17] L. Wang, H.-D. Meyer, and V. May. Femtosecond laser pulse control of multidimensional vibrational dynamics: Computational studies on the pyrazine molecule. *J. Chem. Phys.* **125** (2006), 014102.

[18] H.-D. Meyer, F. Le Quéré, C. Léonard, and F. Gatti. Calculation and selective population of vibrational levels with the Multiconfiguration Time-Dependent Hartree (MCTDH) algorithm. *Chem. Phys.* **329** (2006), 179–192.

[19] U. V. Riss and H.-D. Meyer. Investigation on the reflection and transmission properties of complex absorbing potentials. *J. Chem. Phys.* **105** (1996), 1409.

[20] M. H. Beck and H.-D. Meyer. An efficient and robust integration scheme for the equations of motion of the multiconfiguration time-dependent Hartree (MCTDH) method. *Z. Phys. D* **42** (1997), 113–129.

[21] S. Zöllner, H.-D. Meyer, and P. Schmelcher. Ultracold few-boson systems in a double-well trap. *Phys. Rev. A* **74** (2006), 053612.

[22] A. Jäckle and H.-D. Meyer. Product representation of potential energy surfaces. *J. Chem. Phys.* **104** (1996), 7974.

[23] A. Jäckle and H.-D. Meyer. Product representation of potential energy surfaces II. *J. Chem. Phys.* **109** (1998), 3772.

[24] S. Sukiasyan. *Investigation of three– and four–atomic reactive scattering problems with the help of the multiconfiguration time-dependent Hartree method*. PhD thesis, Universität Heidelberg, 2005.

[25] F. Gatti, F. Otto, S. Sukiasyan, and H.-D. Meyer. Rotational excitation cross sections of *para*-H_2 + *para*-H_2 collisions. A full-dimensional wave packet propagation study using an exact form of the kinetic energy. *J. Chem. Phys.* **123** (2005), 174311.

[26] U. Manthe. *Mehrdimensionale Wellenpaketdynamik nach elektronischen Anregungen*. PhD thesis, Universität Heidelberg, 1991.

[27] U. Manthe, H.-D. Meyer, and L. S. Cederbaum. Multiconfigurational time-dependent Hartree study of complex dynamics: Photodissociation of NO_2. *J. Chem. Phys.* **97** (1992), 9062–9071.

[28] U. Manthe and A. D. Hammerich. Wavepacket dynamics in five dimensions. Photodissociation of methyl iodide. *Chem. Phys. Lett.* **211** (1993), 7.

[29] H.-D. Meyer, U. Manthe, and L. S. Cederbaum. The multi-configuration Hartree approach. In *Numerical Grid Methods and their Application to Schrödinger's Equation* (Dordrecht, 1993), C. Cerjan, Ed., Kluwer Academic Publishers, pp. 141–152.

[30] A. P. J. Jansen. A multiconfiguration time-dependent Hartree approximation based on natural single-particle states. *J. Chem. Phys.* **99** (1993), 4055.

[31] U. Manthe. Comment on "A multiconfiguration time-dependent Hartree approximation based on natural single-particle states". *J. Chem. Phys.* **101** (1994), 2652.

[32] A. P. J. Jansen. Response to "Comment on 'A multiconfiguration time-dependent Hartree approximation based on natural single-particle states"'. *J. Chem. Phys.* **101** (1994), 2654.

[33] A. D. Hammerich, U. Manthe, R. Kosloff, H.-D. Meyer, and L. S. Cederbaum. Time-dependent photodissociation of methyl iodide with five active modes. *J. Chem. Phys.* **101** (1994), 5623.

[34] J.-Y. Fang and H. Guo. Multiconfiguration time-dependent Hartree studies of the CH_3I/MgO photodissociation dynamics. *J. Chem. Phys.* **101** (1994), 5831.

[35] J.-Y. Fang and H. Guo. Four-dimensional quantum dynamics of the CH_3I/MgO photodissociation. *Chem. Phys. Lett.* **235** (1995), 341.

[36] J.-Y. Fang and H. Guo. Multiconfiguration time-dependent Hartree studies of the Cl_2Ne vibrational predissociation dynamics. *J. Chem. Phys.* **102** (1995), 1944.

[37] L. Liu, J.-Y. Fang, and H. Guo. How many configurations are needed in a time-dependent Hartree treatment of the photodissociation of ICN? *J. Chem. Phys.* **102** (1995), 2404.

[38] J.-Y. Fang and H. Guo. Quantum dynamics within the multiconfiguration time-dependent Hartree approximation. *J. Mol. Struct. (Theochem)* **341** (1995), 201.

[39] A. Jäckle and H.-D. Meyer. Reactive scattering using the multiconfiguration time-dependent Hartree approximation: General aspects and application to the collinear $H+H_2 \rightarrow H_2+H$ reaction. *J. Chem. Phys.* **102** (1995), 5605.

[40] A. P. J. Jansen and H. Burghgraef. MCTDH study of CH_4 dissociation on Ni (111). *Surf. Sci.* **344** (1995), 149.

[41] A. Capellini and A. P. J. Jansen. Convergence study of multi-configuration time-dependent Hartree simulations: H_2 scattering from LiF(001). *J. Chem. Phys.* **104** (1996), 3366.

[42] U. Manthe and F. Matzkies. Iterative diagonalization within the multi-configurational time-dependent Hartree approach: Calculation of vibrationally excited states and reaction rates. *Chem. Phys. Lett.* **252** (1996), 71.

[43] U. Manthe. A time-dependent discrete variable representation for (multi-configuration) Hartree methods. *J. Chem. Phys.* **105** (1996), 6989.

[44] M. Ehara, H.-D. Meyer, and L. S. Cederbaum. Multiconfiguration time-dependent Hartree (MCTDH) study on the rotational and diffractive inelastic molecule-surface scattering. *J. Chem. Phys.* **105** (1996), 8865.

[45] K. Museth and G. D. Billing. Generalization of the multiconfigurational time-dependent Hartree method to nonadiabatic systems. *J. Chem. Phys.* **105** (1996), 9191.

[46] F. Matzkies and U. Manthe. A multi-configurational time-dependent Hartree approach to the direct calculation of thermal rate constants. *J. Chem. Phys.* **106** (1997), 2646.

[47] T. Gerdts and U. Manthe. The resonance Raman spectrum of CH_3I: An application of the MCTDH approach. *J. Chem. Phys.* **107** (1997), 6584.

[48] A. Jäckle. *Die zeitabhängige Multikonfigurations-Hartree Methode und ihre Anwendung auf reaktive Streuprozesse.* PhD thesis, Universität Heidelberg, 1997.

[49] H.-D. Meyer, G. A. Worth, and J.-Y. Fang. Comment on "Generalization of the multiconfigurational time-dependent Hartree method to nonadiabatic systems" [J. Chem. Phys. 105, 9191 (1996)]. *J. Chem. Phys.* **109** (1998), 349.

[50] K. Museth and G. D. Billing. Response to "Comment on 'Generalization of the multiconfigurational time-dependent Hartree method to nonadiabatic systems' " [J. Chem. Phys. 109, 349 (1998)]. *J. Chem. Phys.* **109** (1998), 351.

[51] H.-D. Meyer. Multiconfiguration time-dependent Hartree method. In *The Encyclopedia of Computational Chemistry* (Chichester, 1998), P. v. R. Schleyer, N. L. Allinger, T. Clark, J. Gasteiger, P. A. Kollman, H. F. Schaefer III, and P. R. Schreiner, Eds., vol. 5, John Wiley and Sons, pp. 3011–3018.

[52] R. Milot and A. P. J. Jansen. Ten dimensional wavepacket simulations of methane scattering. *J. Chem. Phys.* **109** (1998), 1966.

[53] F. Matzkies and U. Manthe. Accurate quantum calculations of thermal rate constants employing MCTDH: H_2+OH $\rightarrow$ H+H_2O and D_2+OH $\rightarrow$ D+DOH. *J. Chem. Phys.* **108** (1998), 4828.

[54] F. Matzkies and U. Manthe. Accurate reaction rate calculations including internal and rotational motion: A statistical MCTDH approach. *J. Chem. Phys.* **110** (1999), 88.

[55] A. Jäckle, M.-C. Heitz, and H.-D. Meyer. Reaction cross sections for the H+$D_2(\nu = 0, 1)$ system for collision up to 2.5 eV: A multiconfiguration time-dependent Hartree wave-packet propagation study. *J. Chem. Phys.* **110** (1999), 241–248.

[56] I. Burghardt, H.-D. Meyer, and L. S. Cederbaum. Approaches to the approximate treatment of complex molecular systems by the multiconfiguration time-dependent Hartree method. *J. Chem. Phys.* **111** (1999), 2927–2939.

[57] A. Raab, I. Burghardt, and H.-D. Meyer. The multiconfiguration time-dependent Hartree method generalized to the propagation of density operators. *J. Chem. Phys.* **111** (1999), 8759–8772.

[58] R. Milot and A. P. J. Jansen. Energy distribution analysis of the wavepacket simulations of CH_4 and CD_4 scattering. *Surf. Sci.* **452** (2000), 179.

[59] R. Milot and A. P. J. Jansen. Bond breaking in vibrationally excited methane on transition-metal catalysts. *Phys. Rev. B* **61** (2000), 15657.

[60] G. A. Worth. Accurate wave packet propagation for large molecular systems: The multi-configuration time-dependent Hartree (MCTDH) method with selected configurations. *J. Chem. Phys.* **112** (2000), 8322–8329.

[61] A. Raab. *Untersuchung der Dynamik quantenmechanischer Systeme in Wechselwirkung mit Umgebungen mit Hilfe der zeitabhängigen Multikonfigurations-Hartree-Methode.* PhD thesis, Universität Heidelberg, 2000.

[62] M. H. Beck. *Berechnung von Schwingungsrotationsspektren mit Hilfe der zeitabhängigen Multikonfigurations-Hartree- und der Filter-Diagonalisierungs-Methode.* PhD thesis, Universität Heidelberg, 2000.

[63] A. Raab. On the Dirac-Frenkel/McLachlan variational principle. *Chem. Phys. Lett.* **319** (2000), 674–678.

[64] A. Raab and H.-D. Meyer. Multi-configurational expansions of density operators: Equations of motion and their properties. *Theor. Chem. Acc.* **104** (2000), 358–369.

[65] A. Raab and H.-D. Meyer. A numerical study on the performance of the multiconfiguration time-dependent Hartree method for density operators. *J. Chem. Phys.* **112** (2000), 10718–10729.

[66] F. Huarte-Larrañaga and U. Manthe. Full dimensional quantum calculations of the CH_4+H $\rightarrow$ CH_3+H_2 reaction rate. *J. Chem. Phys.* **113** (2000), 5115.

[67] U. Manthe and F. Matzkies. Rotational effects in the H_2+OH $\rightarrow$ H+H_2O reaction rate: Full-dimensional close-coupling results. *J. Chem. Phys.* **113** (2000), 5725.

[68] F. Matzkies and U. Manthe. Combined iterative diagonalization and statistical sampling in accurate reaction rate calculations: Rotational effects in O+HCl $\rightarrow$ OH+Cl. *J. Chem. Phys.* **112** (2000), 130.

[69] H. Wang. Basis set approach to the quantum dissipative dynamics: Application of the multiconfiguration time-dependent Hartree method to the spin-boson problem. *J. Chem. Phys.* **113** (2000), 9948.

[70] M.-C. Heitz and H.-D. Meyer. Rotational and diffractive inelastic scattering of a diatom on a corrugated surface: A multiconfiguration time-dependent Hartree (MCTDH) study on N_2/LiF(001). *J. Chem. Phys.* **114** (2001), 1382–1392.

[71] G. A. Worth. Quantum dynamics using pseudo-particle trajectories: A new approach based on the multi-configuration time-dependent Hartree method. *J. Chem. Phys.* **114** (2001), 1524–1532.

[72] S. Sukiasyan and H.-D. Meyer. On the effect of initial rotation on reactivity. A multi-configuration time-dependent Hartree (MCTDH) wave-packet propagation study on the H+D_2 and D+H_2 reactive scattering systems. *J. Phys. Chem. A* **105** (2001), 2604–2611.

[73] F. Gatti, M. H. Beck, G. A. Worth, and H.-D. Meyer. A hybrid approach of the multi-configuration time-dependent Hartree and filter-diagonalisation methods for computing bound-state spectra. Application to HO_2. *PCCP* **3** (2001), 1576–1582.

[74] S. Mahapatra, G. A. Worth, H. D. Meyer, L. S. Cederbaum, and H. Köppel. The $\widetilde{A}^2$E $\widetilde{B}^2B_2$ photoelectron bands of allene beyond the linear coupling scheme: An *ab initio* dynamical study including all fifteen vibrational modes. *J. Phys. Chem. A* **105** (2001), 5567–5576.

[75] C. Cattarius, G. A. Worth, H.-D. Meyer, and L. S. Cederbaum. All mode dynamics at the conical intersection of an octa-atomic molecule: Multi-configuration time-dependent Hartree (MCTDH) investigation on the butatriene cation. *J. Chem. Phys.* **115** (2001), 2088–2100.

[76] H. Wang, M. Thoss, and W. Miller. Systematic convergence in the dynamical hybrid approach for complex systems: A numerical exact methodology. *J. Chem. Phys.* **115** (2001), 2979.

[77] M. Thoss, H. Wang, and W. H. Miller. Self-consistent hybrid approach for complex systems: Application to the spin-boson model with debye spectral density. *J. Chem. Phys.* **115** (2001), 2991.

[78] C. Meier and U. Manthe. Full-dimensional quantum study of the vibrational predissociation of the I_2...Ne_2 cluster. *J. Chem. Phys.* **115** (2001), 5477.

[79] F. Huarte-Larrañaga and U. Manthe. Vibrational excitation in the transition state: The CH_4+H $\rightarrow$ CH_3+H_2 reaction rate constant in an extended temperature interval. *J. Chem. Phys.* **116** (2002), 2863.

[80] H. Nauendorf, G. Worth, H.-D. Meyer, and O. Kühn. Multi-configuration time-dependent Hartree dynamics on an ab initio reaction surface: Ultrafast laser -driven proton motion in Phthalic Acid Monomethylester. *J. Phys. Chem.* **106** (2002), 719–724.

[81] T. N. Rescigno, W. A. Isaacs, A. E. Orel, H.-D. Meyer, and C. W. McCurdy. Theoretical study of resonant excitation of CO_2 by electron impact. *Phys. Rev. A* **65** (2002), 32716.

[82] S. Sukiasyan and H.-D. Meyer. Reaction cross section for the H+$D_2(\nu_0 = 1)$ $\rightarrow$ HD+D and D+$H_2(\nu_0 = 1)$ $\rightarrow$ DH+H systems. A multi-configuration time-dependent Hartree (MCTDH) wave-packet propagation study. *J. Chem. Phys.* **116** (2002), 10641–10647.

[83] H. Köppel, M. Döscher, I. Baldea, H.-D. Meyer, and P. G. Szalay. Multistate vibronic interactions in the benzene radical cation. II. Quantum dynamical simulations. *J. Chem. Phys.* **117** (2002), 2657–2671.

[84] U. Manthe. Reaction Rates: Accurat quantum dynamical calculations for polyatomic systems. *J. Theor. Comp. Chem.* **1** (2002), 153.

[85] F. Huarte-Larrañaga and U. Manthe. Accurate quantum dynamics of a combustion reaction: Thermal rate constants of O(^{3}P) + $CH_4(X^1A_1)$ $\rightarrow$ OH($X^2\Pi$) + $CH_3(X^2\ A_2'')$. *J. Chem. Phys.* **117** (2002), 4635.

[86] M. Nest and H.-D. Meyer. Benchmark calculations on high-dimensional Henon-Heiles potentials with the Multi-Configuration Time-Dependent Hartree (MCTDH) Method. *J. Chem. Phys.* **117** (2002), 10499–10505.

[87] J. Trin, M. Monnerville, B. Pouilly, and H.-D. Meyer. Photodissociation of the ArHBr complex investigated with the Multi–Configuration Time–Dependent Hartree (MCTDH) approach. *J. Chem. Phys.* **118** (2003), 600–609.

[88] G. Worth and I. Burghardt. Full quantum mechanical molecular dynamics using Gaussian wavepackets. *Chem. Phys. Lett.* **368** (2003), 502–508.

[89] C. McCurdy, W. A. Isaacs, H.-D. Meyer, and T. Rescigno. Resonant vibrational excitation of CO_2 by electron impact: Nuclear dynamics on the coupled components of the $^2\Pi_u$ resonance. *Phys. Rev. A* **67** (2003), 042708–1–19.

[90] F. Huarte-Larrañaga and U. Manthe. Quantum mechanical calculation of the OH + HCl $\rightarrow$ H_2O + Cl reaction rate: Full-dimensional accurate, centrifugal sudden, and J-shifting results. *J. Chem. Phys.* **118** (2003), 8261.

[91] M. Nest and H.-D. Meyer. Dissipative quantum dynamics of anharmonic oscillators with the Multi-Configuration Time-Dependent Hartree (MCTH) Method. *J. Chem. Phys.* **119** (2003), 24.

[92] H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *J. Chem. Phys.* **119** (2003), 1289.

[93] H. Wang and M. Thoss. Theoretical study of ultrafast photoinduced electron transfer processes in mixed-valence systems. *J. Phys. Chem. A* **107** (2003), 2126–2136.

[94] D. Egorova, M. Thoss, W. Domcke, and H. Wang. Modeling of ultrafast electron-transfer processes: Validity of multilevel Redfield theory. *J. Chem. Phys.* **119** (2003), 2761.

[95] M. Petkovic and O. Kühn. Multidimensional hydrogen bond dynamics in Salicylaldimine: Coherent nuclear wave packet motion versus intramolecular vibrational energy redistribution. *J. Phys. Chem. A* **107** (2003), 8458–8466.

[96] M. Thoss, W. Domcke, and H. Wang. Theoretical study of vibrational wave-packet dynamics in electron-transfer systems. *Chem. Phys.* **296** (2004), 217–229.

[97] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum. Multidimensional dynamics involving a conical intersection: Wavepacket calculations using the MCTDH method. In *Conical Intersections* (Singapore, 2004), W. Domcke, D. R. Yarkony, and H. Köppel, Eds., World Scientific Co., pp. 573–607.

[98] F. Richter, M. Hochlaf, P. Rosmus, F. Gatti, and H.-D. Meyer. A study of mode–selective trans–cis isomerisation in HONO using ab initio methodology. *J. Chem. Phys.* **120** (2004), 1306–1317.

[99] F. Richter, P. Rosmus, F. Gatti, and H.-D. Meyer. Time–dependent wavepacket study on trans–cis isomerisation of HONO. *J. Chem. Phys.* **120** (2004), 6072–6084.

[100] C. Iung, F. Gatti, and H.-D. Meyer. Intramolecular vibrational energy redistribution in the highly excited Fluoroform molecule: A quantum mechanical study using the MCTDH algorithm. *J. Chem. Phys.* **120** (2004), 6992–6998.

[101] R. van Harrevelt and U. Manthe. Multiconfigurational time-dependent Hartree calculations for dissociative adsorption of H_2 on Cu(100). *J. Chem. Phys.* **121** (2004), 3829.

[102] D. J. Haxton, Z. Zhang, H.-D. Meyer, T. N. Rescigno, and C. W. McCurdy. Dynamics of dissociative attachment of electrons to water through the $^2B_1$ metastable state of the anion. *Phys. Rev. A* **69** (2004), 062714.

[103] T. Wu, H.-J. Werner, and U. Manthe. First-principles theory for the H + CH_4 $\rightarrow$ H_2 + CH_3 reaction. *Science* **306** (2004), 2227–2229.

[104] B. Lasorne, F. Gatti, E. Baloitcha, H.-D. Meyer, and M. Desouter-Lecomte. Cumulative isomerization probability studied by various transition state wave packet methods including the mctdh algorithm. benchmark: HCN $\rightarrow$ CNH. *J. Chem. Phys.* **121** (2004), 644–654.

[105] F. Gatti and H.-D. Meyer. Intramolecular vibrational energy redistribution in Toluene: A nine dimensional quantum mechanical study using the MCTDH algorithm. *Chem. Phys.* **304** (2004), 3–15.

[106] M. Petkovic and O. Kühn. Ultrafast wave packet dynamics of an intramolecular hydrogen transfer system: From vibrational motion to reaction control. *Chem. Phys.* **304** (2004), 91.

[107] E. V. Gromov, A. B. Trofimov, N. M. Vitkovskaya, H. Köppel, J. Schirmer, H.-D. Meyer, and L. S. Cederbaum. Theoretical study of excitations in furan: Spectra and molecular dynamics. *J. Chem. Phys.* **121** (2004), 4585.

[108] R. van Harrevelt and U. Manthe. Degeneracy in discrete variable representations: General considerations and applications to the multiconfigurational time-dependent hartree approach. *J. Chem. Phys.* **121** (2004), 5623.

[109] M. D. Coutinho-Neto, A. Viel, and U. Manthe. The ground state tunneling splitting of malonaldehyde: Accurate full dimensional quantum dynamics calculations. *J. Chem. Phys.* **121** (2004), 9207–9210.

[110] C. Cattarius and H.-D. Meyer. Multi-dimensional density operator propagations in open systems: Model studies on vibrational relaxation and surface sticking processes. *J. Chem. Phys.* **121** (2004), 9283–9296.

[111] O. Vendrell and H.-D. Meyer. Proton conduction along a chain of water molecules. Development of a linear model and quantum dynamical investigations using the multiconfiguration time-dependent hartree method. *J. Chem. Phys.* **122** (2005), 104505.

[112] K. Giese, H. Ushiyama, K. Takatsuka, and O. Kühn. Dynamical hydrogen atom tunneling in dichlorotropolone: A combined quantum, semiclassical, and classical study. *J. Chem. Phys.* **122** (2005), 124307.

[113] S. Woittequand, C. Toubin, B. Pouilly, M. Monnerville, S. Briquez, and H.-D. Meyer. Photodissociation of a HCl molecule adsorbed on ice. *Chem. Phys. Lett.* **406** (2005), 202–209.

[114] B.Pouilly, M. Monnerville, F. Gatti, and H.-D. Meyer. Wave packet study of the UV photodissociation of the Ar_2HBr complex. *J. Chem. Phys.* **122** (2005), 184313.

[115] S. Zöllner, H.-D. Meyer, and P. Schmelcher. Multi-electron giant dipole resonances of atoms in crossed electric and magnetic fields. *Eur. Phys. Lett.* **71** (2005), 373–379.

[116] K. Giese and O. Kühn. The all-Cartesian reaction plane Hamiltonian: Formulation and application to the H-atom transfer in tropolone. *J. Chem. Phys.* **123** (2005), 054315.

[117] R. van Harrevelt and U. Manthe. Multidimensional time-dependent discrete variable representations in multiconfiguration hartree calculations. *J. Chem. Phys.* **123** (2005), 064106.

[118] S. Zöllner, H.-D. Meyer, and P. Schmelcher. N-electron giant dipole states in crossed electric and magnetic fields. *Phys. Rev. A* **72** (2005), 033416.

[119] A. Markmann, G. Worth, S. Mahapatra, H.-D. Meyer, H. Köppel, and L. Cederbaum. Simulation of a complex spectrum: Interplay of five electronic states and 21 vibrational degrees of freedom in $C_5H_4^+$. *J. Chem. Phys.* **123** (2005), 204310.

[120] C. Crespos, H.-D. Meyer, R. C. Mowrey, and G. J. Kroes. Multiconfiguration time-dependent Hartree method applied to molecular dissociation on surfaces: H_2+Pt(111). *J. Chem. Phys.* **124** (2006), 074706.

[121] G. Pasin, F. Gatti, C. Iung, and H.-D. Meyer. Theoretical investigation of Intramolecular Vibrational Energy Redistribution in highly excited HFCO. *J. Chem. Phys.* **124** (2006), 194304.

[122] D. V. Tsivlin, H.-D. Meyer, and V. May. Vibrational excitations in α-helical polypeptides: Multiexiton self-trapping and related infrared transient absorption. *J. Chem. Phys.* **124** (2006), 134907.

[123] S. Zöllner, H.-D. Meyer, and P. Schmelcher. Correlations in ultracold trapped few-boson systems: Transition from condensation to fermionization. *Phys. Rev. A* **74** (2006), 063611.

[124] G. Pasin, C. Iung, F. Gatti, and H.-D. Meyer. Theoretical investigation of highly excited vibrational states in DFCO: Calculation of the out-of-plane bending states and simulation of the intramolecular vibrational energy redistribution. *J. Chem. Phys.* **126** (2007), 024302.

[125] T. S. Venkatesan, S. Mahapatra, H.-D. Meyer, H. Köppel, and L. S. Cederbaum. Multimode Jahn-Teller and Pseudo-Jahn-Teller interactions in the cyclopropane radical cation: Complex vibronic spectra and nonradiative decay dynamics. *J. Phys. Chem. A* **111** (2007), 1746.

[126] S. Zöllner, H.-D. Meyer, and P. Schmelcher. Excitations of few-body systems in one-dimensional harmonic and double wells. *Phys. Rev. A* **75** (2007), 043608.

[127] C. Matthies, S. Zöllner, H.-D. Meyer, and P. Schmelcher. Quantum dynamics of two bosons in an anharmonic trap: Collective versus internal excitations. *Phys. Rev. A* **76** (2007), 023602.

[128] M. R. Brill, F. Gatti, D. Lauvergnat, and H.-D. Meyer. Photoinduced nonadiabatic dynamics of ethene: Six dimensional wave packet propagations using two different approximations of the kinetic energy operator. *Chem. Phys.* **338** (2007), 186–199.

[129] O. Vendrell, F. Gatti, and H.-D. Meyer. Dynamics and infrared spectroscopy of the protonated water dimer. *Angew. Chem. Int. Ed.* **46** (2007), 6918–6921.

[130] A. N. Panda, F. Otto, F. Gatti, and H.-D. Meyer. Rovibrational energy transfer in ortho-H_2 + para-H_2 collisions. *J. Chem. Phys.* **127** (2007), 114310.

[131] F. Richter, F. Gatti, C. Léonard, F. Le Quéré, and H.-D. Meyer. Time–dependent wave packet study on trans–cis isomerisation of HONO driven by an external field. *J. Chem. Phys.* **127** (2007), 164315.

[132] S. Woittequand, D. Duflot, M. Monnerville, B. Pouilly, C. Toubin, S. Briquez, and H.-D. Meyer. Classical and quantum studies of the photodissociation of a HX (X=Cl,F) molecule adsorbed on ice. *J. Chem. Phys.* **127** (2007), 164717.

[133] O. Vendrell, F. Gatti, D. Lauvergnat, and H.-D. Meyer. Full dimensional (15D) quantum-dynamical simulation of the protonated water dimer I: Hamiltonian setup and analysis of the ground vibrational state. *J. Chem. Phys.* **127** (2007), 184302.

[134] O. Vendrell, F. Gatti, and H.-D. Meyer. Full dimensional (15D) quantum-dynamical simulation of the protonated water dimer II: Infrared spectrum and vibrational dynamics. *J. Chem. Phys.* **127** (2007), 184303.

Index