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

Developers's Guide

Version 8

Release 2 Revision 2

Authors:

G. A. Worth, M. H. Beck, A. Jäckle, and H.-D. Meyer

Address:

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

August 12, 2024

Contents

List of Tables

List of Figures

List of Examples

Copyright

Acknowledgments

The very first MCTDH program, later called version 1, was written by Uwe Manthe as part of his PhD work in Heidelberg. Over the years several graduate students, post-docs and visitors have made contributions to the MCTDH package. We list them in chronological order:
M. Ehara, M.-C. Heitz, A. Raab, S. Wefing, S. Sukiasyan, C. Cattarius F. Gatti, F. Otto, M. Nest, A. Markmann and M. R. Brill.
We are very, very grateful to all of them!

Citations

When citing the MCTDH program package in the literature, the following citation should be used:

G. A. Worth, M. H. Beck, A. Jäckle, and H.-D. Meyer. The MCTDH Package, Version 8.2, (2000), University of Heidelberg, Heidelberg, Germany

A comprehensive description of the methods incorporated in the programs is in:

M. H. Beck, A. Jäckle, G. A. Worth, and H.-D. Meyer. The multiconfiguration time-dependent Hartree (MCTDH) method: A highly efficient algorithm for propagating wavepackets. *Phys. Rep.* **324:1** (2000), 1.

The original paper is:

H.-D. Meyer, U. Manthe, and L. S. Cederbaum. The multi-configurational time-dependent Hartree approach. *Chem. Phys. Lett.* **165** (1990), 73.

A list of publications concerning the MCTDH method is given at the end of this Guide. The latest version of this list can be found on the MCTDH homepage:

http://www.pci.uni-heidelberg.de/tc/usr/mctdh/

From this URL a review on the MCTDH scheme, Ref. [**?**], may also be downloaded.

Introduction

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.

Part I

MCTDH program

Chapter 1

The Structure of the MCTDH Program

1.1 Program-Hierarchy

Figure 1.1 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.

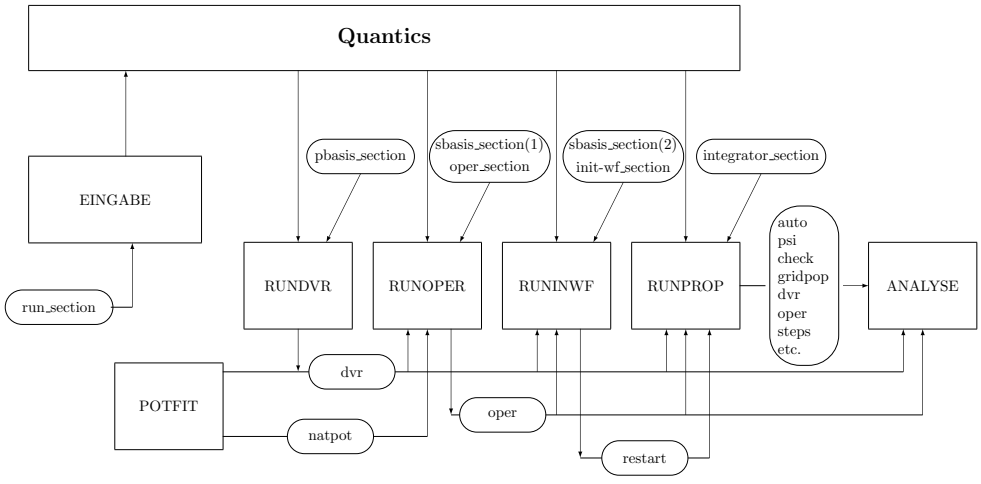

Figure 1.1: The structure of the MCTDH programs. See text for details.

1.2 File-Hierarchy

Chapter 2

The Processing of Input-Files

Chapter 3

The operator

3.1 Operator representation in the program

3 levels:

`hamilton(f,k) = h`	Hamiltonian terms
`hterm(nh,h) = hl`	operators
`hoplab(hl)`	labels

Associated information with indices `k, h, hl`

See operdef.inc, arrays dimensioned with `maxkoe`, `maxhtm`, `maxhop`.

`hamilton(f,k)`

fkoe, ftime: "DOF" for coefficient and time

`ki(k), kf(k)`	indices for state labels (multi-set)
	$\langle \Phi^{(kf)} \mid H_k^{(kf,ki)} \mid \Phi^{(ki)} \rangle$
`hsym(m,k)`	symmetry flag for operator
`diag(m,k)`	if .true., unit operator
`kact(k)`	type of operator (for dissipative operators)

....

k

Multiple operators add index to `kzahl`, `nmulpot`, `zhun`, `khzahl`, `khtot`.
labelled by `hamlab(n), n=1,nham`
k = zham(n), zham(n)+khtot(n)
hamlab(1) = 'system'
ktot = $\sum$khtot

`hterm(nh,h)`

Integers defining operator h, pointing to labels in `hoplab`
$1 <$ nh $<$ htmfac(h)
`htmfac(h)` no. of factors in operator.
e.g. for `dqdvr^2*-1*0.5*mass_rd^-1` with

```
hoplab(2) = -1
hoplab(4) = 0.5
hoplab(5) = mass_rd^-1
hoplab(9) = dqdvr^2
```

then

```
htmfac(h) = 4
hterm(1,h) = 9, hterm(2,h) = 2, hterm(3,h) = 4,
     hterm(4,h) = 5
```

If operator is sum of terms, e.g.
`KEdvr*1*w10a + q^2*0.5*w10a`
terms separated in list by 0. Here `htmfac(h)=7`

htmdof(h), htmmode(h), htmmuld(h)

One of these integers is non-zero and indicates the coordinates.

`htmdof(h) = f`	One-D operator on DOF f
`htmmode(h) = m`	mode operator on mode m
`htmmuld(h) = muld`	multi-D operator with coordinates defined in `mulddof(n,muld)`

htmform(h)

If set to 1, `hterm(nh,h)` integers point to other hterm operators rather than hoplab.

htmsym(h)

Symmetry of operator.

1 hermitian
-1 anti-hermitian

htmtype(h)

Type of operator. See genoper/asshop.F

```
######################################################################
C
C ASSIGNHOP
C
C assigns the type of operator
C
C hopz:    operator matrix
C
C hoptype: type of matrix, from folowing list
C 1D types
C 0 = zero
C 1 = unit
C 2 = real vector (diagonal matrix)
C 3 = imaginary vector (imaginary part stored as real)
C 4 = complex vector
C 5 = FFT, real vector
C 6 = FFT, imaginary vector
C 7 = FFT, complex vector
C 8 = real matrix
C 9 = imaginary matrix (imaginary part stored as real)
C 10 = complex matrix
C 2D combined-mode types
C 17 = real 3D tensor (for jm^2)
```

```
C 18 = real 3D tensor (for j_p)
C 19 = real 3D tensor (for j_m)
C
C analytic potential
C 101 = multi-dimensional real diagonal analytic
C
C#####################################################################
```

`hoplab(hl)`

Defines function used to build up operators.

`hopilab(hl)`, `hopifile(hl)` ,
: define where function is in OPFUNCS library

`hopipar(n,hl)`, `hoprpar(n,hl)`
: associate parameters with function, e.g. CAP[a,b,c]

3.2 The OPFUNCS library

File nos.	Description
1-100	Operators defined only on a grid (non-local operators, natural potentials etc.).
101-200	Complex analytic functions.
201-300	Multi-dimensional potential energy surfaces.
301-400	Real analytic functions.

`hopifile(hl)`	File name	Description
1		diagonal potentials from file
2		non-diagonal potentials from file
3	funcgrd	1D non-local operators
101	funcanlz	1D complex analytic functions
201	funcsrf	multi-dimensional PES
301		real number
302	funcanld	1D analytic functions
303	func1d	1D potential energy curves

`hopilab(hl)` defines which operator in file

opfuncs/funcgrd.f

```
C#####################################################################
C ifile = 3
C#####################################################################

      subroutine fdefgrd(hopilab,hoplab,hopfile,hlabipar,hlabrpar,
     +           hoppar,maxhpar)
....
C---------------------------------------------------------------------
C assign function number
C---------------------------------------------------------------------
      if (hoplab .eq. 'KEdvr') then
         hopilab=1

....

      else if (hoplab .eq. 'dqdvr^2') then
         hopilab=4

....

C#####################################################################
```

```
      subroutine funcgrd(hmat,gdim,ifunc,ipar,rpar,mass,d1mat,d2mat,
     +           kinsph,fftp,ort,sphdim)
....
      go to (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17) ifunc

C KEdvr
    1 call kedvrterm(d2mat,hmat,mass,gdim)
      return
....
C dqdvr^2
    4 call cpqxdz(d2mat,hmat,gdim)
      return
```

3.3 Overwiew of Build algorithm

Part I: genoper/heingabe.F

(called from mctdh/eingabe.F)
reads .op file and makes initial set-up of arrays.

1. reads parameters from .par / .op files, loads into `apar(n)`, `rpar(n)` arrays (parameters in .inp file already read in einoper).

2. reads HAMILTONIAN-SECTION.
   - breaks up operator keyword to find factors.
   - Stores unique factors in hoplab(hl).
   - Stores unique operator in `hterm(nh,h)`, and associates with coordinates
   - Stores `hamilton(f,k)`

3. reads any definitions in LABELS-SECTIONS (those in alter-labels already read in `einoper`)

Now ready for manipulations of operator.
New labels, operators, terms added to arrays.
Redundant terms removed from hamilton.

4. labels substituted / expanded with definitions.
   Expansions occur for **potfit** /srf files.

5. alterations made for electronic basis:
   - if multi-set expand **1**, mirror S1&2, etc.

6. Labels assigned to functions in library.

Part II: genoper/runoper.F

(called from mctdh.F)
After `heingabe`, memory can be allocated. The operators can then be built, typed, and further sorted.

1. Types hterm operators
   - using information from hoplabs, e.g. product of factors from analytic functions is a "real vector"
   - OR sets up operator matrix and sees what it is

2. Sums up uncorrelated terms of same type. (e.g. DVR and FFT cannot be mixed)
   $\left.\begin{array}{ccc} k_1 & | & A \\ k_2 & | & B \end{array}\right\}$ $\begin{array}{ccc} 1 \\ | \\ 1 \end{array}$ $\begin{array}{ccc} | \\ | \\ | \end{array}$ $\begin{array}{ccc} k_1A + k_2B \\ | \\ 1 \end{array}$
   Types any new operators

3. Sums up (product) correlated terms
   $\left.\begin{array}{ccc} k_1 & | & A \\ k_2 & | & B \end{array}\right\}$ $\begin{array}{ccc} C \\ | \\ C \end{array}$ $\begin{array}{ccc} 1 \\ | \\ 1 \end{array}$ $\begin{array}{ccc} | \\ | \\ | \end{array}$ $\begin{array}{ccc} k_1A + k_2B \\ | \\ C \end{array}$
   Types any new operators

4. Pack coefficients into operators
   $\left.\begin{array}{ccc} k_1 & | & A \\ k_2 & | & A \end{array}\right\}$ $\begin{array}{ccc} C \\ | \\ 1 \end{array}$ $\begin{array}{ccc} 1 \\ | \\ k_2 \end{array}$ $\begin{array}{ccc} | \\ | \\ | \end{array}$ $\begin{array}{ccc} A \\ | \\ A \end{array}$ $\begin{array}{ccc} k_1C \\ | \\ 1 \end{array}$
   Types any new operators

5. Sets up product mode operators for combined modes
   $\left.\begin{array}{ccc} k_1 & | & A \\ k_2 & | & B \end{array}\right\}$ $\begin{array}{ccc} C \\ | \\ D \end{array}$ $\begin{array}{ccc} k_1 \\ | \\ k_2 \end{array}$ $\begin{array}{ccc} | \\ | \\ | \end{array}$ $\begin{array}{ccc} AC \\ | \\ BD \end{array}$ $\begin{array}{ccc} 1 \\ | \\ 1 \end{array}$
   Types any new operators

6. Tries to sum terms again.
   $\left.\begin{array}{ccc} k_1 & | & AC \\ k_2 & | & BD \end{array}\right\}$ $\begin{array}{ccc} 1 \\ | \\ 1 \end{array}$ $\begin{array}{ccc} | \\ | \\ | \end{array}$ $\begin{array}{ccc} k_1AC + k_2BD \\ | \\ 1 \end{array}$
   Types any new operators

7. Set up operators and write to tmp file

8. Delete any zero terms

9. Order hamilton array

10. Set up pointers for hops array
    Write oper file.

End result is oper file.

`hamilton(f,k)` represents Hamiltonian,

`hops(zhop(h))` is representation of operator `hterm(h)`.

Analytic operators (time-dependent, CDVR PES) have no hops representation, but can be calculated on the fly from `hterm(h)`

From subroutine **hop** (propwf/hphi.F)

```
      subroutine hop (f,k,hpsi,psi,hops,subdim1,fftfak,hin,rueck,
     +           exphin,exprueck,vec,workc)
....
      real*8     hops(hopsdim)
      complex*16 hpsi(subdim1),psi(subdim1)
....
C-----------------------------------------------------------------------
C Go to selected operator
C-----------------------------------------------------------------------
      h=hamilton(f,k)
      goto (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19) htmtype(h)

      routine='Hop'
      write(message,'(a,i5)')  'Unknown htmtype :',htmtype(h)
      call errormsg

    1 call cpvxz(psi,hpsi,subdim1)
      return

    2 if (htmdof(h) .gt. 0) then
         call dtxxdz(hops(zhop(h)),psi,hpsi,vgdim(f),gdim(f),ngdim(f))
      else
         call dvxxdz(hops(zhop(h)),psi,hpsi,subdim1)
      endif
      return

....
    8 if (htmdof(h) .gt. 0) then
         call qtxxdz(hops(zhop(h)),psi,hpsi,vgdim(f),gdim(f),ngdim(f))
      else
         call qvxxdz(hops(zhop(h)),psi,hpsi,subdim1)
      endif
      return
```

$$\text{khtot}\left\{\begin{array}{l}\left.\begin{array}{l}1\\ \vdots\\ \text{kzahl}\end{array}\right\}\text{correlated}\\ \left.\begin{array}{l}\text{kzahl}+1\\ \vdots\\ \text{kzahl}+\text{nmulpot}\end{array}\right\}\text{multi}-\text{D}\\ \left.\begin{array}{l}\vdots\\ \text{zhun(m,s)}\\ \vdots\\ \text{zhun(m,s)}+\text{khzahl(m,s)}\\ \vdots\end{array}\right\}\text{uncorrelated}\end{array}\right.$$

Figure 3.1:

Chapter 4

The wave function

4.1 The A and Φ vectors

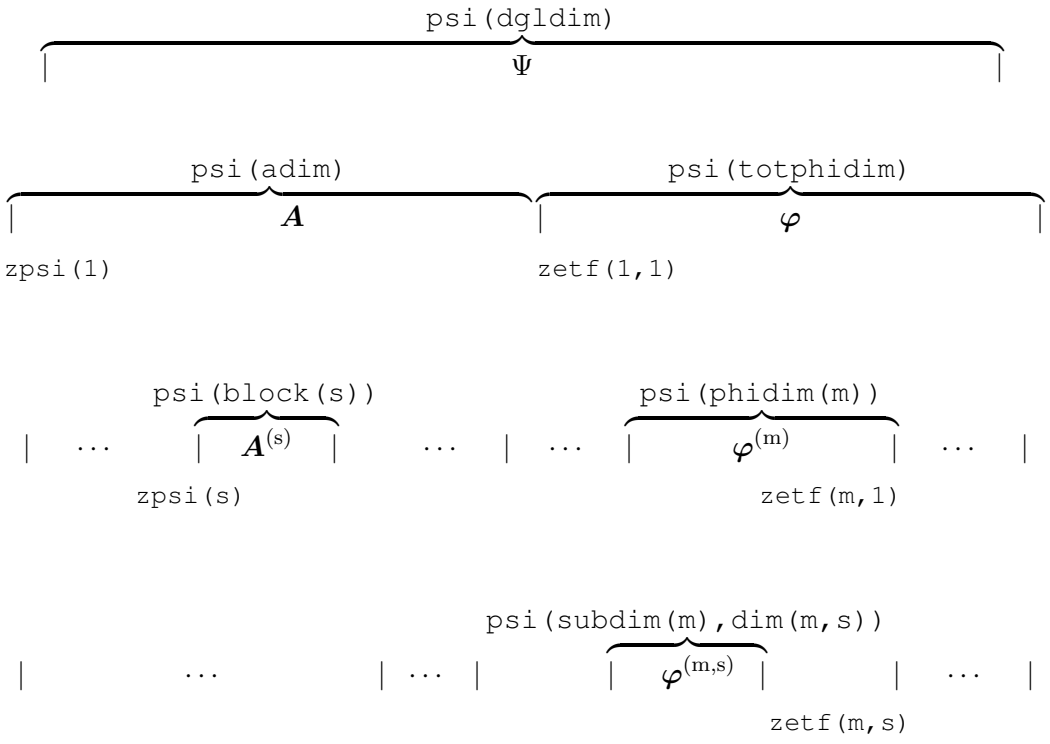

Figure 4.1: The structure of the wave function

4.2 The density matrix

The density matrix ρ is stored in the `mc(mcdim)` and `mr(mrdim)` arrays:

- `mc(mcdicht1)`: The inverse of the density matrix. (Size: `dmatdim`)
- `mc(mcdicht2)`: The density matrix itself. (Size: `dmatdim`)
- `mr(mrdicht3)`: The eigenvalues of the density matrix (Array size doubles; size: `d3matdim`)
- `mc(mcdicht4)`: Eigenvectors of the density matrix. (Size: `dmatdim`).

- `dmatdim`=$\sum$ `dim`^{2}
- `d3matdim`=2$\sum$ `dim`^{2}

- `dmat(m,s)`: Pointer to start of density matrix for mode m and state s.
- `dim(m,s)`: Dimension.

Chapter 5

Internal details of some subroutines

5.1 General subroutines

5.2 Subroutines belonging to wavefunction propagation

5.3 Subroutines belonging to the propagation of density operators of Typ I

5.4 Subroutines belonging to the propagation of density operators of Typ I

5.4.1 `d2opwfit`

This subroutine calculates the a new density operator $\rho = -i[D, \rho_0]$, where ρ_0 is the initial density operator and D an arbitrary, but hermitian, operator. The used method is an iterative algorithm, with the original initial density operator

$$\rho_0 = \sum_M \sum_N A^0_{M,N} \left|\Phi^0_M\right\rangle \left\langle\Phi^0_N\right| \tag{5.1}$$

and the iterated density operator

$$\rho_k = \sum_M \sum_N A^k_{M,N} \left|\Phi^k_M\right\rangle \left\langle\Phi^k_N\right|, \tag{5.2}$$

where

$$\begin{aligned} \left|\Phi^k_M\right\rangle &= \left|\varphi^{(1),k}_{m_1}\right\rangle \cdot \ldots \cdot \left|\varphi^{(f),k}_{m_f}\right\rangle & (5.3)\\ D &= d_1 \otimes \ldots \otimes d_f. & (5.4) \end{aligned}$$

The equations for the iterations ($k = 0, 1, 2, 3, \cdots$) are for the A-vector

$$\begin{aligned} A^{k+1}_{I,J} &= -i \sum_M \left\langle\Phi^k_I|D|\Phi^0_M\right\rangle \sum_N A_{M,N} \left\langle\Phi^0_N|\Phi^k_J\right\rangle \\ &\quad +i \sum_M \left\langle\Phi^k_I|\Phi^0_M\right\rangle \sum_N A_{M,N} \left\langle\Phi^0_N|D|\Phi^k_J\right\rangle \end{aligned} \tag{5.5}$$

and for the spfs

$$|\varphi_l^{(\nu),k+1}\rangle = -i\sum_m \underline{\underline{S}}_{lm}^{(\nu)} |d_\nu \varphi_m^{(\nu),k}\rangle + i\sum_m \underline{\underline{T}}_{lm}^{(\nu)} |\varphi_m^{(\nu),k}\rangle \tag{5.6}$$

with

$$\underline{\underline{S}}_{lm}^{(\nu)} = \sum_{I_\nu J}\sum_{M_\nu N} A^k_{J,I_\nu l} \langle \Phi^k_{I_\nu}|D_\nu|\Phi^0_{M_\nu}\rangle A^0_{M_\nu m,N} \langle \Phi^0_N|\Phi^k_M\rangle \tag{5.7}$$

$$\underline{\underline{T}}_{lm}^{(\nu)} = \sum_{I_\nu J}\sum_{M_\nu N} A^k_{J,I_\nu l} \langle \Phi^k_{I_\nu}|\Phi^0_{M_\nu}\rangle A^0_{M_\nu m,N} \langle \Phi^0_N|D|\Phi^k_M\rangle. \tag{5.8}$$

5.4.2 `d2trafonat`

This routine transforms the density operators of typ II into the natural orbital picture. The used transformation looks like as follows.

Transforming the ket $|\tilde{\Phi}_J\rangle$

$$|\tilde{\Phi}_J\rangle = \sum_L U^*_{JL} |\Phi_L\rangle \tag{5.9}$$

leads to the corresponding equations for the bra:

$$\langle\tilde{\Phi}_J| = \sum_L \langle\Phi_L|U_{LJ}\ . \tag{5.10}$$

And accordingly:

$$|\Phi_J\rangle = \sum_L U_{JL} |\tilde{\Phi}_L\rangle \tag{5.11}$$

$$\langle\Phi_J| = \sum_L \langle\tilde{\Phi}_L|U^*_{LJ}\ . \tag{5.12}$$

Putting these transformations into the density operator ρ then give:

$$\begin{aligned}
\rho &= \sum_{JL} B_{JL} |\Phi_J\rangle\langle\Phi_L| \\
&= \sum_{JLK} B_{JL}U_{JK} |\tilde{\Phi}_K\rangle\langle\Phi_L| \\
&= \sum_{JLKM} B_{JL}U_{JK} |\tilde{\Phi}_K\rangle\langle\tilde{\Phi}_M|U^*_{ML} \\
&= \sum_{JLKM} U_{JK}B_{JL}U^*_{ML} |\tilde{\Phi}_K\rangle\langle\tilde{\Phi}_M| \\
&= \sum_{KM} \tilde{B}_{KM} |\tilde{\Phi}_K\rangle\langle\tilde{\Phi}_M|
\end{aligned} \tag{5.13}$$

with the transformation

$$\tilde{B}_{KM} = \sum_{JL} U_{JK}B_{JL}U^*_{ML}\ . \tag{5.14}$$

Part II

Tools

Index