Manual for Coupled Coherent States Code

Stewart K. Reed

April 14, 2010

Contents

Chapter 1

Theory

This section gives a summary of the equations required for performing CCS calculations. For a fuller derivation of the equations presented, the reader is referred to, for example, ref. [1]. Please note the slight difference in notation: $H_{ord} = \widetilde{H}$.

In quantum dynamics simulations, the wave function, $|\mathbf{\Psi}\rangle$, is expressed in terms of a superposition of coherent states. The projection of a multidimensional coherent state $|\mathbf{z}\rangle$ on to spacial coordinates may be written in the Klauder form [**?**]:

$$\langle \mathbf{x}|\mathbf{z}\rangle = \prod_{\alpha} \left(\frac{\gamma_\alpha}{\pi}\right)^{\frac{1}{4}} \exp\left[-\frac{\gamma_\alpha}{\hbar}\left(x^{(\alpha)} - q^{(\alpha)}\right)^2 + \frac{i}{\hbar}p^{(\alpha)}\left(x^{(\alpha)} - q^{(\alpha)}\right) + \frac{ip^{(\alpha)}q^{(\alpha)}}{2\hbar}\right] , \tag{1.1}$$

where (α) indexes the degrees of freedom of the system and γ_α is dependent on the characteristic frequency of the α degree of freedom. The position and momentum vectors can be expressed terms of the complex coordinate $\mathbf{z}$:

$$\mathbf{q} = \frac{\gamma^{-\frac{1}{2}}}{\sqrt{2}}\left(\mathbf{z}^* + \mathbf{z}\right) \tag{1.2}$$

and

$$\mathbf{p} = i\hbar\frac{\gamma^{\frac{1}{2}}}{\sqrt{2}}\left(\mathbf{z}^* - \mathbf{z}\right) , \tag{1.3}$$

where γ is actually a diagonal matrix but we denote the elements γ_α for brevity.

The positions and momenta of the coherent states evolve independently of those of the other coherent states and according to Hamilton's equations:

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{dt}} = -\frac{i}{\hbar}\frac{\partial\widetilde{H}\left(\mathbf{z}^*, \mathbf{z}\right)}{\partial\mathbf{z}^*} \quad \text{and} \quad \frac{\mathrm{d}\mathbf{z}^*}{\mathrm{dt}} = \frac{i}{\hbar}\frac{\partial\widetilde{H}\left(\mathbf{z}^*, \mathbf{z}\right)}{\partial\mathbf{z}} . \tag{1.4}$$

In these equations $\widetilde{H}$ is the averaged or reordered Hamiltonian, $\widetilde{H} = \widetilde{T} + \widetilde{V}$. The averaging arises because the coherent states are diffuse objects. Consequently, the whole potential surface is, in principle at least, sampled by each coherent state at all times.

The potential energy of a coherent state $\mathbf{z}$ is obtained by evaluating the matrix element

$$\langle \mathbf{z}_1|\hat{V}|\mathbf{z}_2\rangle = \int \langle \mathbf{z}_1|\mathbf{x}\rangle\langle \mathbf{x}|\hat{V}|\mathbf{x}\rangle\langle \mathbf{x}|\mathbf{z}_2\rangle \,\mathrm{dx} , \tag{1.5}$$

for $\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}$. The widths of the coherent states, the γ_α's which appear in equation 1.1, depend on the curvature of the potential but not on time. As a result, we can compute an averaged potential energy surface,

$$\widetilde{V}\left(z_1^{(\alpha)}, z_2^{(\alpha)}\right) = \frac{1}{\langle z_1^{(\alpha)} | z_2^{(\alpha)} \rangle} \int \langle z_1^{(\alpha)} | x^{(\alpha)} \rangle \langle x^{(\alpha)} | \hat{V}^{(\alpha)} | x^{(\alpha)} \rangle \langle x^{(\alpha)} | z_2^{(\alpha)} \rangle \mathrm{dx}^{(\alpha)} \ , \tag{1.6}$$

and then consider the coherent states as point objects for the purposes of calculating potential energy. Elsewhere in the literature $\widetilde{V}$ is denoted by V_{ord} where ord stands for ordered. This alternative notation arises because the appropriate energy operator can be written in terms of creation and annihilation operators in such a way that within products creation operators appear to the left of any annihilation operators. The reordering required is not totally trivial because the operators do not commute, $\left[\hat{a}, \hat{a}^\dagger\right] = 1$, so transposing pairs of operators introduces extra terms. This reordering is necessary because $\hat{a}|z\rangle = z|z\rangle$ and $\langle z|\hat{a}^\dagger = \langle z|z^*$ but other combinations, such as $\hat{a}^\dagger|\mathbf{z}\rangle$, are not defined.

Expressed in terms of creation and annihilation operators, the position and momentum vectors operators are

$$\hat{\mathbf{q}} = \frac{\gamma^{-\frac{1}{2}}}{\sqrt{2}}\left(\hat{\mathbf{a}}^\dagger + \hat{\mathbf{a}}\right) \quad \text{and} \quad \hat{\mathbf{p}} = i\hbar\frac{\gamma^{\frac{1}{2}}}{\sqrt{2}}\left(\hat{\mathbf{a}}^\dagger - \hat{\mathbf{a}}\right) \ . \tag{1.7}$$

As an example of reordering, consider the kinetic energy operator $\hat{T} = \hat{\mathbf{p}}/2m$. Substituting in the expression for $\hat{\mathbf{p}}$ gives

$$\langle \mathbf{z}|\hat{\mathbf{T}}|\mathbf{z}\rangle = \frac{-\hbar^2\gamma}{2}\langle \mathbf{z}|\left(\hat{\mathbf{a}}^{\dagger^2} - \hat{\mathbf{a}}\hat{\mathbf{a}}^\dagger - \hat{\mathbf{a}}^\dagger\hat{\mathbf{a}} + \hat{\mathbf{a}}^2\right)|\mathbf{z}\rangle = \frac{-\hbar^2\gamma}{2}\langle \mathbf{z}|\left(\hat{\mathbf{a}}^{\dagger^2} - 2\hat{\mathbf{a}}^\dagger\hat{\mathbf{a}} - 1 + \hat{\mathbf{a}}^2\right)|\mathbf{z}\rangle \tag{1.8}$$

Using $\hat{a}|z\rangle = z|z\rangle$, $\langle z|\hat{a}^\dagger = \langle z|z^*$ and equations 1.2 and 1.3, and defining the averaged kinetic energy as $\widetilde{T} = \langle \mathbf{z}|\hat{\mathbf{T}}|\mathbf{z}\rangle / \langle \mathbf{z}|\mathbf{z}\rangle$ in analogy to equation 1.2 gives

$$\widetilde{T} = \frac{\mathbf{p}^2}{2m} + \sum_\alpha \frac{\hbar^2\gamma_\alpha}{4m} \ . \tag{1.9}$$

Although the motion of coherent states in phase space is independent, the contribution of each coherent state to the wave function is dependent on all the coherent states. The projection of the wave function onto coherent state j is

$$\langle \mathbf{z}_j|\mathbf{\Psi}\rangle = C_j \exp\left(i\frac{S_j}{\hbar}\right) \ , \tag{1.10}$$

where S_j is the classical action, which is defined as

$$S_j(t) = \int_0^t \left[\frac{i\hbar}{2}\left(\mathbf{z}^*\frac{\mathrm{d}\mathbf{z}}{\mathrm{dt}'} - \frac{\mathrm{d}\mathbf{z}^*}{\mathrm{dt}'}\mathbf{z}^*\right) - \widetilde{H}\left(\mathbf{z}^*, \mathbf{z}\right)\right] \mathrm{dt}' \ . \tag{1.11}$$

Coherent states are not orthogonal so the identity matrix has the nontrivial form:

$$\hat{I} = \sum_{j,k}^{N} |\mathbf{z}_j\rangle \left(\Omega^{-1}\right)_{jk} \langle \mathbf{z}_k| \ , \tag{1.12}$$

where N is the number of coherent states and $\Omega_{jk} = \langle \mathbf{z}_j | \mathbf{z}_k \rangle$ is the overlap of states j and k. It follows that the wave function may be written:

$$|\mathbf{\Psi}\rangle_j = \sum_k D_k \exp\left(i\frac{S_k}{\hbar}\right)|\mathbf{z}_k\rangle \ , \tag{1.13}$$

where the calculation of D_k using

$$C_j \exp\left(i\frac{S_j}{\hbar}\right) = \sum_k \Omega_{jk} D_k \exp\left(i\frac{S_k}{\hbar}\right) \tag{1.14}$$

is the slowest step of the method.

The time evolution of the amplitudes C_j is obtained from Schrödinger's equation,

$$\frac{\mathrm{d}|\mathbf{\Psi}\rangle}{\mathrm{dt}} = -\frac{i}{\hbar}\hat{H}|\mathbf{\Psi}\rangle \ . \tag{1.15}$$

Multiplying both sides from the left by $\langle \mathbf{z}_j|$ gives, after a little algebra,

$$\langle \mathbf{z}_j | \frac{\mathrm{d}|\mathbf{\Psi}\rangle}{\mathrm{dt}} = -\frac{i}{\hbar}\sum_{k,l}^{N} \langle \mathbf{z}_j | \mathbf{z}_k \rangle \widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_k\right)\left(\Omega^{-1}\right)_{k,l}\langle \mathbf{z}_l | \mathbf{\Psi}\rangle \ . \tag{1.16}$$

Finally, we consider

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle \mathbf{z}_j | \mathbf{\Psi}\rangle = \langle \dot{\mathbf{z}}_j | \mathbf{\Psi}\rangle + \langle \mathbf{z}_j | \dot{\mathbf{\Psi}}\rangle \ , \tag{1.17}$$

where the dot indicates the derivative with respect to time. After some manipulation and upon defining

$$\delta^2\widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_k\right) = \widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_k\right) - \widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_j\right) - \frac{\partial \widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_j\right)}{\partial\, \mathbf{z}_j}(\mathbf{z}_k - \mathbf{z}_j) \ , \tag{1.18}$$

we eventually arrive at

$$\begin{aligned} \frac{\mathrm{dC_j}}{\mathrm{dt}} &= \frac{i}{\hbar}\sum_{k,l}^{N}\langle \mathbf{z}_j | \mathbf{z}_k \rangle \delta^2\widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_k\right)\left(\Omega^{-1}\right)_{kl} C_l \exp\left(\frac{i}{\hbar}\left(S_l - S_j\right)\right) \\ &= \frac{i}{\hbar}\sum_{k}^{N}\langle \mathbf{z}_j | \mathbf{z}_k \rangle \delta^2\widetilde{H}\left(\mathbf{z}_j^*, \mathbf{z}_k\right) D_k \exp\left(\frac{i}{\hbar}\left(S_k - S_j\right)\right) \ . \end{aligned} \tag{1.19}$$

Chapter 2

Compiling the program

Compilation of the program is handled by Make. All platform specific details such as compiler name and libraries are contained in the file `make.options`. The makefile itself is therefore platform independent. `make.options` files for different platforms are stored in the makeoptions directory. The appropriate file should be copied to make.options in the src directory. The program should then compile by simply typing make.

The program comes with a limited version of the lapack library. This is for use on systems which do not have optimised lapack libraries already installed and may be compiled by typing `make lapack`. It may be that even on systems with lapack libraries, these routines may be quicker/easier to use. To link the main program to this library, add `-L. -lccslapack` to the `LIBS` variable in the `make.options` file. The program also comes with a dummy parallel library. This must be used if a serially executing program is required. It is made by typing `make parallellib`. This library is in src/mpiscalapack along with mpif.h files. Please note that these are not to be used if the program is to be executed in parallel. In that case the mpif.h that comes with the mpi distribution that is to be used. For creating a serial program, add `-L. -ldummyparallel` to the `LIBS` in the `make.options` file and `-Impiscalapack` to the F90FLAGS variable. If they are to be used, the lapack and dummy parallel libraries should be made before the main program.

To compile in parallel, replace `-Impiscalapack` in the F90FLAGS variable with `-DUSE_SCALAPACK` and remove `-ldummyparallel` from the LIBS variable. Specifying the compiler as mpif90 normally includes the requisite MPI libraries but you will also need to add appropriate options to the LIBS variable for the ScaLapack routines. By default, the program uses the INPLACE option provided by the MPI2 specification. It is possible to use MPI1 instead by adding `-DMPI1` to the F90FLAGS variable.

The program also comes with several utility programs, see chapter 4. Some of these have make files but not all. Some are perl scripts and as such do not need one. Makefiles for these utility programs don't use the make.options file which the main program does. Consequently, the makefile should be edited to take account of platform specifics.

Chapter 3

Using the program

The program requires several input files. The behaviour of the program is controlled by a file called `run.inpt`. This file contains details such as the number of steps to run and whether a classical or quantum calculation is to be performed. See section 3.1. Details of the interactions are given in the file called `hamiltonian.inpt` whose format is described in section 3.4. It is also necessary to have a file called `states.inpt` the contents of which are described in section 3.3.

3.1 Inputs

The following options may be specified in the run.inpt file

General Options

nsteps `n` *Total number of steps to perform.*

dt `timestep` *Set the time step to* `timestep` *fs.*

ndf `N` *Number of degrees of freedom*

nbasis `nfcts` *Number of basis functions*

basis `basistype` *How to choose basis functions. Requires other options immediately after. See section 3.2*

energy `unit` *Energy units to use.* `unit` *must be either au, eV, H or akma*

gamma *Specify gammas. Should be followed by* `ndf` *value pairs* `i G` *each on a separate line. Clearly must come after ndf option!*

classical *To perform a classical simulation. Omit to perform a quantum simulation.*

semiclassical *To perform a semiclassical simulation. The initial D's are calculated from the initial C's. These are then used as in the quantum calculations but they are kept constant throughout the simulation.*

restart *Do a continuation simulation*

verbosity `v` *How info should be written out. 0 - standard, 1 - more, and 2 - everything*

omega0 *When is zero zero for overlap matrix? Default:* 10^{-10}

minprob `mp` *For basis function removal*

maxz `mz` *For basis function removal*

randseed `rs` *Start random number generator with rs as a seed*

Options to control output files

writepos `posfreq` *Frequency with which to write positions to a file.*

writezpos `posfreq` *Frequency with which to write z-positions to a file.*

writeacf `acffreq` *Frequency with which to write auto correlation function*

crosscorr `ccorlog` *Frequency with which to write cross corrlelation to file*

harmcorr `harmcorr` *Frequency with which to write harmonic correlation*

writeCD `CDfreq` *Frequency with which to write C and D*

writeamp `ampfreq` *Frequency with which to write amplitudes, includes basis function positions too in same file.*

writeaction `actopmfreq` *Frequency with which to write the classical action. Default: 0.*

writeenergy `energyfreq` *Frequency with which to write the energy. Default: 100*

writenorm `normfreq` *Frequency with which to write the norm of the wave function. Default: 100*

writebasisnorm `basisnormfreq` *Frequency with which to write the norm of each basis function. I.e.* $C_i^* D_i \forall i$. *Default: never*

dumpfreq `df` *Frequency with which to write sysstate.out for restarts*

solve `method` *Method by which to calculate D given C. By default it uses LAPACK when run in serial and ScaLapack when run in parallel. Can now use Lapack when run in parallel. There is also the option of using a conjugate gradients routine but this seems to be not as accurate and takes as long if not longer than using LAPACK.* `method` *should be either* `LAPACK`, `CG` *or* `SCALAPACK`.

cgepsilon `v` *Convergence parameter for CG routine.*

3.1.1 Energy units

The energy units of the output depend upon the various parameters of the hamiltonian etc. However, there are a few options although at the moment I can't remember the exact details.

The akma units are $\sqrt{(\text{kcal/mol/amu})}$ Å. These are the units of the potential supplied by Emilio. To convert to kcal/mol it is necessary to multiply by 0.3105. The code can do this for you. To get the energy in kcal/mol from akma units specify

```
energy akma kcalmol
```

in the run.inpt file.

3.1.2 writepos and writezpos

Usage: `writepos freq`, `writezpos freq`

Write out positions of basis functions every `freq` steps. Defaults to 0 indicating no output. writepos gives position and momentum q and p. writezpos gives $\Re(z)$ and $\Im(z)$ pairs. The first number on each line is the time in fs.

3.1.3 verbosity

Specify how information should be written out. Usage: `verbosity n` where n is an integer s.t. $n \geq 0$

3.1.4 restart

This option specifies that the simulation should continue from where a previous run finished. This requires a restart.inpt file. At the end of the run, the program writes out a file called `systate.out`. To restart from this, simply copy it to `restart.inpt` and add the keyword `restart` to the `run.inpt` file.

3.1.5 omega0

Specify how the tolerance on overlap matrix elements being zero. If $|\langle z|z'\rangle| < tol$ then the overlap is set to zero.

Usage: `omega0 tol`

where tol is a real number and $tol > 0$.

3.1.6 minprob and maxz

These two options which may be used separately or together allow basis functions which have escaped to be removed from the calculation and thereby preventing the energy becoming infinite. Usage: `minprob mp`

`maxz mz`

The first causes basis functions whose probability is less than mp to be removed. While the second removes basis functions which have $|z| > mz$. Once removed, a variable (incbf) is set to false and the basis function is no longer updated. When calculating energy and dynamics it is not included.

3.2 Basis Functions

There are several ways of choosing basis functions. The method is chosen:
`basis method`
where `method` may be one of the following

- Grid
- Random gaussian
- Initial states
- Harmonic Oscillator
- From file

3.2.1 Grid

This doesn't quite work. There should be $(n_q * n_p)^N$ basis functions where N is the number of degrees of freedom and n_q and n_p are the number of divisions along the q and p axes. Instead I think there are only $(n_q * n_p)$.

3.2.2 Random Gaussin

Choose initial basis functions randomly from a gaussian distribution. A maximu energy may be specified. The widths of the distributions in q and p for each degree of freedom then needs to be specified. The position and momentum for each function is chosen separately at random from a Gaussian distribution. The Gaussian is centred on the appropriate value of z given in the states.inpt file.

For example, the run.inpt file might contain

```
basis rgauss
enmax 5
1.0 1.0
0.1 0.1
0.1 0.1
0.1 0.1
0.1 0.1
0.1 0.1
0.1 0.1
```

The numbers are the widths of the distributions in q and p for each mode. The energy cutoff enmax is optional.

3.2.3 Initial states

It is possible to use the initial states as the basis functions. This simply requires the inclusion of the key word `istates`. The number of states given in the states.inpt file must be the same as the number of basis functions specified in the run.inpt file. If this is not the case the program stops. The program could easily be amended but it would require reallocating all arrays which have a dimension dependent on the number of basis functions. The initial states are given in terms of q and p.

Initial states may now also be used in conjunction with harmonic oscillator basis functions. See the section on the Harmonic Oscillator for details.

Also, it is now possible to read in basis functions from another file. This has the advantage that it allows you to separate the initial state and the initial basis functions. In the new file it is also possible to specify the initial amplitudes.

3.2.4 Harmonic Oscillator

The initial basis functions may be sampled from the harmonic oscillator wave functions. In coherent states the ground state wave function is

$$\psi_n(z) = \frac{\langle n|z\rangle}{\pi^{1/2}} = \frac{z^n}{(\pi n!)^{1/2}} \exp\left(-\frac{|z|^2}{2}\right) . \tag{3.1}$$

The probability for a particular z is the $|\langle n|z\rangle|^2 = \langle z|n\rangle\langle n|z\rangle$.

Keyword for run.inpt file: harmosc

There are a few optional inputs which maybe omitted. These are.

maxtries n Default value for n is 20.

rmax r Default value for r is 3.5.

super ns nb `ns` of the modes are in vibrational states which are superpositions of harmonic oscillator states. The maximum number of harmonic oscillator states used to describe the superpositions is `nb`.

The next line should contain the keyword `vibstates` and this should be followed by a further ndf sets of lines. Each line begins with an integer denoting which method to use. The remainder of the line is governed by the method. The choice is

0 Put all the basis functions at the origin in this mode

1 To be followed by an integer `n` and a real number `comp`. Choose basis fns randomly from the harmonic oscillator state $|n\rangle$ but with a compression factor of `comp`.

2 As 1 except two extra real numbers are required `dx` and `dy` which give a displacement for the final basis function with respect to the harmonic oscillator wave function.

3 Same inputs as 1. Basis function however is read from the states.inpt file, rather than being chosen randomly. In other words, this option is like using the istates option for choosing basis functions.

4 As 3 but allows an offset to be specified for the harmonic oscillator in the same way as option 2. The basis functions in the states.inpt file should be offset already. The offset is applied in reverse which projecting the wave function on to the harmonic oscillator wave function.

5 Randomly choose basis functions from a wave function which is a superposition of harmonic oscillator states. Samples from a rectangular box from qmin to qmax and from pmin to pmax. The maximum probability density is also required. Usage 5 qmin qmax pmin pmax maxprob nbf the nbf coefficients.

6 currently unused. They have been left for sampling states which are superpositions of harmonic oscillator states.

7 Use the basis functions in the states.inpt file. The vibrational mode however is a super position of harmonic oscillator states. The number 7 should be followed by the number of states in the expansion ($\leq$ `ns` above) which should in turn be followed by the specified number of coefficients for the harmonic oscillator states (ground state, ν=1, etc.)

8 As for 7 except that the 8 is followed by $\Re(z_{offset})$ and $Im(z_{offset})$. The number of states in the expansion comes next followed by the coefficients.

The compression factor cf reduces the width (compresses) the distribution from which the basis functions are chosen. For the ground state this is achieved simply by dividng the chosen z by cf. This may not be the best way of doing things.

I have found a way of compressing higher order states. It stems from the fact that in p-space higher order states are circular with a maximum at a set value of $|z|$. Integrating over θ gives a probability distribution which looks a bit like an offset and distorted gaussian. The distribution in r is

$$|\psi_n(r)|^2 = \frac{1}{\pi n!} \exp\left(-(r-r_0)^2/\sigma^2\right) r^(2n+1)\,. \tag{3.2}$$

If $r_0 = 0$ and $\sigma = 1$ then this reduces to the usual expression. Varying σ changes the width of the distribution. However, it also changes the position of the maximum probability. The program therefore calculates r_0 so as to keep the maximum in the same place.

$$r_0 = \frac{(2n+1)(1-\sigma^2)}{\sqrt{4n+2}}\,. \tag{3.3}$$

3.2.5 Initial basis from file

Rather than specifying the basis functions in the initial states file, `states.inpt`, the initial basis functions may be given in a file called `basis.inpt`. This option is selected by including

```
basis file
```

in the `run.inpt` file. The first line of the file should contain the keyword `basis` followed by the number of states which appear in the file. This should be the same as the number given in the `run.inpt` file. The second line may optionally contain the words `amplitudes` or `unitamp`. The former indicates amplitudes are specified as well as position, q, and momentum, p, while the latter indicates that the amplitudes should all be equal such that the total amplitude is 1.

For a system with 6 degrees of freedom, a basis file containing a single basis function might look like the following:

```
nbasis 1
amplitudes
q1 p1 ℜ(A1) ℑ(A1)
```

q_2 p_2 $\Re(A_2)$ $\Im(A_2)$
q_3 p_3 $\Re(A_3)$ $\Im(A_3)$
q_4 p_4 $\Re(A_4)$ $\Im(A_4)$
q_5 p_5 $\Re(A_5)$ $\Im(A_5)$
q_6 p_6 $\Re(A_6)$ $\Im(A_6)$

Obviously, q_i, p_i, $\Re(A_i)$ and $\Im(A_i)$ should be replaced by the appropriate numerical values. If the word amplitudes is omitted, or if unitamp is specified, the amplitudes may be omitted and will be ignored if they are there.

Basis set files can be created using the utility program createbasis, see section 4.8.

3.3 Initial States

The file `states.inpt` contains details of coherent states. These were originally used for calculating correlation functions and initial amplitudes. This coherent states listed in this file can also be used as the initial basis functions or as the centres of distributions. See elsewhere for details.

The first line of the file should contain the keyword `nstates` followed by the number of states which appear in the file. This could be 0. Even if it is 0, this file still needs to exist and contain this first line. Each state starts with the keyword `state` which, for ease of reference may be followed by a number/other text but only the keyword is considered. Each state is given by a position q and momentum p separated by one or more spaces for each degree of freedom. Each q p pair should be on a new line. Only the number of states given on the very first line will be read.

3.4 Potentials

The following systems are available

- Quartic tunnelling + harmonic bath
- Henon-Heiles - correct?
- Polynomial, Morse and Gaussians
- Classical Polynomial, Morse and Gaussians
- Cartesian potentials

3.4.1 Quartic tunnelling + harmonic bath

3.4.2 Henon-Heiles

3.4.3 Polynomial, Morse and Gaussian Potentials

So far this is the most general type of potential which this program can handle. Each mode can have any combination of polynomial, morse and gaussian potentials. At the moment polynomials up to and including x^{10} are supported. Further extension is possible especially as there is actually a reccurrence relation which simplifies the reordering process. This however has not been used here.

To use these potentials the hamiltonian file must begin with the keyword `polymorse`. This is to be followed by the number of degrees of freedom and then the masses. It may be that the masses are incorporated into the generalised coordinates in which case the masses should be set to 1.0. After the masses each mode should be listed together with its potential. An example is shown below.

```
polymorse
ndf 3
mass
1.0
1.0
1.0
mode 1
poly 2
x0 0.0
x1 0.0
x2 5.0
mode 2 morse
A 2.0
alpha 0.5
q0 1.0
mode 3
gaussian
A 2.0
q0 0.5
sigma 1.0
end
```

Potential types

There are three types of potentials available within the polymorse Hamiltonian. The first is a polynomial and is selected by the keyword poly followed by the order p (highest power) within the polynomial. This is then followed by $p+1$ lines containing a label and the coefficient of the corresponding term. Note that the label is ignored by the program.

The next potential type is the Morse potential. The unaveraged Morse is

$$V_{morse}(q) = A\left(1 - \exp\left[-\alpha\left(q - q_0\right)\right]\right)^2 \ . \tag{3.4}$$

This potential may be averaged either by direct integration or using

$$\exp\left(\hat{A} + \hat{B}\right) = \exp\left(\hat{B}\right)\exp\left(\hat{A}\right)\exp\left(\frac{1}{2}\left[\hat{A}, \hat{B}\right]\right) \ , \tag{3.5}$$

where $\left[\hat{A}, \hat{B}\right]$ is the commutator. Care must be taken in the application of this formula however. Specifically, the exponent $\hat{A} + \hat{B}$ must be the complete exponent and may not be multiplied by any constants. The correct average

potential is

$$\widetilde{V}_{morse}(q) = A\left(1 - 2\exp\left[-\alpha(q-q_0) + \frac{1}{4}\frac{\alpha^2}{\gamma}\right] + \exp\left[-2\alpha(q-q_0) + \frac{\alpha^2}{\gamma}\right]\right) . \tag{3.6}$$

Note that this equation can not be written in the form $(1-\exp(D))^2$. This function has a minimum at

$$q_0 + \frac{3\alpha}{4\gamma} . \tag{3.7}$$

As of January 2009 I believe it makes most sense to choose $\bar{q}_0 = q_0 - 3\alpha/(4\gamma)$ for the averaged potential so that the minimum is in the same places as in the original potential. The shift in the minimum occurs because the potential is asymmetric. Whether or not to reposition the minimum obviously depends on the system which is being modelled. However, I feel that an experimentally obtained position q_0 will probably already be in the correct place even given the quantum nature of the averaging therefore $\bar{q}_0$ should be used instead.

The final potential type is a Gaussian. The unaveraged potential is

$$V_{Gaussian}(q) = A\exp\left[-\alpha(q-q_0)^2\right] . \tag{3.8}$$

It appears that the variable q_0 is actually used in the code despite being given in the list of required inputs above. Therefore all Gaussians are centred at zero. The averaged potential is

$$\widetilde{V}(q) = A\sqrt{\frac{\gamma}{\gamma+\alpha}}\exp\left[-\frac{\gamma\alpha}{\gamma+\alpha}q^2\right] . \tag{3.9}$$

In the code, it is actually written in terms of z:

$$\widetilde{V}(z_1^*, z_2) = A\sqrt{\frac{\gamma}{\gamma+\alpha}}\exp\left[-\frac{\alpha}{2(\gamma+\alpha)}(z_2+z_1^*)^2\right] . \tag{3.10}$$

Coupling

This potential type also allows a polynomial type coupling between each pair of modes to be sepecified. The coupling follows immediately after the word `end` which is there to denote the end of the uncoupled part of the potential and begins with the keyword `coupling` followed by the number of couplings given. It is not necessary to specify all couplings. Those not included will be set to zero.

```
coupling 2
1 2 50.0 1.0 1.0
1 3 20.0 1.0 1.0
```

The two integers denote the modes being coupled. The first number should be less than the second. The three real numbers indicate the coupling parameters a_{12}, a_{21} and a_{22} where the coupling potential is

$$H^{coup} = a_{12}xy^2 + a_{21}x^2y + a_{22}x^2y^2 \tag{3.11}$$

The reordered potential and therefore the one actually used is

$$\begin{aligned} H_{ord}^{coup} =& a_{12}x\left(y^2+\frac{1}{2\gamma_y}\right)+a_{21}y\left(x^2+\frac{1}{2\gamma_x}\right) \\ &+a_{22}\left(x^2+\frac{1}{2\gamma_x}\right)\left(y^2++\frac{1}{2\gamma_y}\right). \end{aligned} \tag{3.12}$$

Coupling Truncation

Given that the first two terms in the H^{coup} are essentially cubic, the coupling often tends to $-\infty$ as x and y tend to $\pm\infty$. For most potential wells this is not a problem since the potential energy increases to $\pm\infty$ as x and/or y become large. This however is not so for Morse potentials. In this case the potential tends to a constant in either the positive or negative limits. In this case it is possible for basis functions to escape in a manner such that they are accelerating away from the origin. The problem ultimately arises because H^{coup} is not a sufficiently good fit to the energy surface.

One option is to truncate the averaged coupling potential smoothly using an error function. This can be done by writing

$$H_{ord}^{t-coup}\left(q_i,q_j\right)=H_{ord}^{coup}\left(q_i,q_j\right)\times\mathcal{T}_{ord}\left(q\right). \tag{3.13}$$

where q is either q_i or q_j and

$$\mathcal{T}_{ord}(q)=\frac{1}{2}\left(\mathrm{erf}\left(\frac{-d\left(q-x_0\right)}{w}\right)+1\right). \tag{3.14}$$

The truncation occurs over a distance w and is centred on x_0. $d=\pm1$ is the direction in which the coupling is truncated. For example, if $d=-1$ then $s(q)=0$ if $q\ll x_0$ thereby truncating the potential as $q\to-\infty$.

Truncation as $q\to\pm\infty$ can be achieved using

$$\mathcal{T}_{ord}(q)=\frac{1}{2}\left(\mathrm{erf}\left(\frac{-\left(q-x_0\right)}{w}\right)+\mathrm{erf}\left(\frac{\left(q+x_0\right)}{w}\right)\right). \tag{3.15}$$

However, this is really inconsistent with the idea of averaged and unaveraged potentias. A better option therefore is to truncate the unaveraged potential using a Heaviside function. A smoother function could be used if one could be found for which the averaging integral could be calculated. The unaveraged potential is then

$$H^{t-coup}\left(q_i,q_j\right)=H^{coup}\left(q_i,q_j\right)\times\mathcal{T}(q_i,q_j) \tag{3.16}$$

where

$$\mathcal{T}(q_i,q_j)=\Theta(c_1-q_i)\times\Theta(c_2-q_i)\times\Theta(q_j-c_3)\times\Theta(c_4-q_j). \tag{3.17}$$

Here $\Theta(x)$ is the Heaviside function and is defined as

$$\Theta(x)=\begin{cases}0 & \text{if } x<0,\\ 1 & \text{if } x>0.\end{cases} \tag{3.18}$$

By choosing c_k's appropriately, this expression may be reduced to 3 cases: truncation occurs when one q approaches either $+\infty$ or $-\infty$; when one $|q|$ approaches ∞; and when $|q_i|$ and $|q_j|$ approach ∞. These correspond to truncation methods 2 - 4 in the program. See the code for the final averaged function!

To summarise, the available methods are

± 1 Truncate the reordered coupling potential using equation 3.14. The sign indicates whether the coupling is truncated towards $+\infty$ or $-\infty$.

± 2 Truncate the unordered coupling potential using a step function s(q) = 0,1 depending on x. The truncated unordered potential is then reordered by integration. Same sign convention/arguments as with ± 1. There is no width argument.

$+3$ Use the same truncation as for ± 2 except this time truncate as the specified mode goes to $\pm\infty$.

$+4$ As for 3 except truncate with regards to both modes. This requires two cutoffs, one for each mode.

$+5$ Same as 1 except that the truncation is applied as the specified mode goes to $\pm\infty$ using equation 3.15. Currently only implemented in the classical polymorse hamiltonian.

Each coupling may be truncated separately. Truncation is achieved by adding some lines to the file `hamilton.inpt` after the coupling terms. The first line to add contains the keyword `couptrunc` and gives the number of coupling trunctions:

```
couptrunc 2
```

with each subsequent line corresponding to a separate truncation. All truncations start with

```
i j tm
```

where i and j are the coupled modes with $i < j$, and tm is the truncation method. What follows then depends on the truncation method.

For method $tm = \pm 1$, the line should also contain which co-ordinate q_i, or q_j controls the truncation, the location of the point where $\mathcal{T} = 0.5$ and the width, w. For example

```
1 2 -1 1 -0.45 0.1
```

truncates the coupling between q_1 and q_2 using method -1 as $q_1 \to -\infty$. $\mathcal{T} = 0.5$ at $q_1 = -0.45$ and the cutoff width is 0.1.

The specification for methods ± 2 and $+3$ looks almost the same as for methods ± 1 except the width is not required. Also, the position refers to the location of the step in the Heaviside function. The specification for the final method, $+4$, requires only an addition two cutoffs, one for q_i and the second for q_j:

```
1 2 4 0.45 0.5
```

Coupling switch

As well as truncating the coupling when one of the coupled degrees of freedom gets too large in absolute terms, the truncation can also be switched by the position of a third mode. The coupling then becomes

$$H^{tscoup}(q_i, q_j, q_k) = H^{coup}(q_i, q_j) \times \mathcal{T}(q_i, q_j) \times w(q_k) \,, \tag{3.19}$$

where $w(q_k)$ is the switching function. There are two switching functions available. The first is a Heaviside function although this becomes an error function on averaging. The second is a gaussian located at the origin. Using a Heaviside function gives an impulse to the switching co-ordinate which causes unphysical behaviour whereas using a gaussian gives a gradual. Obviously the gaussian still affects the equations of motion for the switching mode q_k but in a much smoother manner. In the case of the gaussian switch,

$$w(q_k) = \mathrm{e}^{-\gamma_k \omega_k q_k} \,, \tag{3.20}$$

where γ_k is the mode frequency used through out the program, and ω is the width of the switching gaussian.

Coupling switching may be specified for each coupling individually. The switching is given in the `hamiltonian.inpt` file and is announced using the keyword `coupswitch` followed by the number of switchings. As with the coupling truncation, each switch is specified in terms of the two coupled modes, the switching method, and the mode which controls the switching. If the Heaviside method is used, method 1, then the direction of the switch and its position are also required. If however, the gaussian switch is used it is sufficient to specify the width of the gaussian, ω. For example

```
coupswitch 2
1 2 1 3 1 0.5
1 3 2 2 1.5
```

will switch the coupling between q_1 and q_2 using a Heaviside function of q_3 with the coupling being turned off for $q_3 \gg 0.5$. Meanwhie, the coupling between modes q_1 and q_3 will be switch using a gaussian function of q_2 with a width of 1.5.

3.4.4 Classical Polynomial, Morse and Gaussian Potentials

This set of Hamiltonians is based on thePolynomial, Morse and Gaussian Potentials except that the unaveraged potentials are used. The use and options should all be as described in the previous section except that a keyword of `classicalpolymorse` should be used instead of `polymorse`. The hamiltonian file needs to contain the particle masses and the interactions between particles.

3.4.5 Cartesian Potentials

To date, the only potential that I have managed to integrate in 3 cartesian dimensions is the gaussian potential centred at the origin. This is okay as Morse potentials can be expressed as the sum of such gaussians. An example file for a 2 particle, 3D system is

```
cartesian
ndf 6
mass
79.904 Br
79.904 Br
interactions 1
maxterms 3
gauss 1 2 3
461264.0 0.417655
46126.418 0.65923
-370.9415 1.385
```

The number of degrees of freedom is the product of the number of particles and the number of cartesian dimensions. The masses of all the particles (not degrees of freedom as with other hamiltonian types) need to be given.

The line interactions indicates the number of pair potentials which will be specified. maxterms gives the maximum number of terms used to describe these interactions. Currently it is only possible to give pair potentials as sums of gaussians, so each pair potential starts with the word gauss followed by two numbers corresponding to the particles between which the potential is to be applied. The third number is the number of gaussians used to describe the potential. This must be less than or equal to maxterms.

The gaussian parameters are A and σ such that

$$V_g(q) = A \exp\left(-\frac{q^2}{2\sigma^2}\right) \tag{3.21}$$

The averaged potential for $q = q^{(\alpha)} - q^{(\beta)}$ is

$$\begin{aligned}\widetilde{V}\left(\mathbf{z}_1^*, \mathbf{z}_2\right) =& \sqrt{\frac{2\sigma^2\gamma_\alpha\gamma_\beta}{2\gamma_\alpha\gamma_\beta\sigma^2 + \gamma_\alpha + \gamma_\beta}} \\ & \times \exp\left[-\frac{\left(\sqrt{\gamma_\beta}\left(z_1^{(\alpha)^*} + z_2^{(\alpha)}\right) - \sqrt{\gamma_\alpha}\left(z_1^{(\beta)^*} + z_2^{(\beta)}\right)\right)^2}{4\gamma_\alpha\gamma_\beta\sigma^2 + 2\gamma_\alpha + 2\gamma_\beta}\right] .\end{aligned} \tag{3.22}$$

Constraints

Constraints may optionally be added by adding something like the following to the `hamiltonian.inpt` file.

```
constraints 1
1 1 2 3.8628339
```

The first line indicates that there is 1 constraint. Currently, it is not actually possible to handle any more than this. The second line begins with the constraint number and this is followed by the indices of the two particles whose separation is to be constrained, in this case particles 1 and 2. Finally, the real number is the desired separation in Å.

The constraints are achieved using a modified Hamiltonian with Lagrange multipliers. The constrained Hamiltonian is

$$H_c = T + V - \lambda_1\sigma_1 - \lambda_2\sigma_2 \tag{3.23}$$

in which σ_1 and σ_2 are the constraints and λ_1 and λ_2 are the Lagrange multipliers. Since the constraint terms are identically zero when the constraints are satisfied, no attempt has been made to average or reorder these terms. Instead we assume that any such process would simply result in a different choice of the Lagrange multipliers. This assumption gives

$$\widetilde{H} = \widetilde{T} + \widetilde{V} - \lambda_1\sigma_1 - \lambda_2\sigma_2 \tag{3.24}$$

Unusually when compared to most constrained simulations that I am aware of, we need to constraints. This is because the basis functions z describe both position and momentum and a constraint is needed for each.

The first constraint is simply the required separation:

$$\sigma_1 = \sum_\alpha \left(x^{(1\alpha)} - x^{(2\alpha)}\right)^2 - d^2 \,, \tag{3.25}$$

in which $x^{(1\alpha)}$ is the position of the 1st constrained particle in the cartesian α direction. d is the required separation of particles 1 and 2. When this constraint is satisfied, $\sigma_1 = 0$.

The second constraint states that the relative velocity of the two particles parallel to the separation vector; this should be zero. It is of course possible to have a non-zero relative velocity perpendicular to the separation vector as this corresponds to a rotation of the molecule. The constraint is

$$\sigma_2 = \frac{\sum_\alpha \left(v^{(1\alpha)} - v^{(2\alpha)}\right).\left(x^{(1\alpha)} - x^{(2\alpha)}\right)}{\left[\sum_\alpha \left(x^{(1\alpha)} - x^{(2\alpha)}\right)^2\right]^{\frac{1}{2}}} \,, \tag{3.26}$$

in which $v^{(1\alpha)}$ is the velocity of the 1st constrained particle in the cartesian α direction. As with the first constraint, when this constraint is satisfied, $\sigma_2 = 0$.

3.4.6 Classical Cartesian Potentials

These are the same as the Cartesian potentials except that they are introduced by the keyword `classicalcartesian` and they use the unordered potentials.

3.5 Output Files

3.5.1 amplitudes.out

This file contains the complex amplitude of each basis function and the full position in phase space. Each entry starts with a comment giving the time/step number and is terminated by a blank line. In between there is one line for each basis function. Each of these lines records the time, real and imaginary parts of the amplitude and the position and momentum for each degree of freedom.

The wave function is

$$|\mathbf{\Psi}\rangle = \sum_{j=1}^{N} D_j \exp\left(iS_j\right) |\mathbf{z}\rangle_j \ . \tag{3.27}$$

The amplitude written out in the amplitudes file is $D_j \exp\left(iS_j\right)$.

3.5.2 energy.out

The system energy is

$$E = \sum_{j}^{\mathrm{ndf}} \sum_{k}^{\mathrm{ndf}} D_j^* D_k H_{ord}\left(z_j^*, z_k\right) \mathrm{e}^{\mathrm{i(S_k - S_j)}} \ , \tag{3.28}$$

where ndf is the number of degrees of freedom. When the polymorse hamiltonian is used, the program also writes out the energies of the individual modes. In this case energy.out contains the time and total energy followed by the energy of each mode followed by the sum of the coupling energy terms which that mode each mode is involved in. It follows therefore that the total energy is the sum of the individual modes plus one half of the sum of the coupling energy.

3.5.3 hexpect.out

This contains the expectation values for the total energy and the energy in each mode. Also included is the change in energy of each due to basis functions being dropped.

In quantum mechanics we can use the density operator $\hat{\rho}$ to calculate the expectation value of some property A without complete knowledge of the wave function using

$$\langle A \rangle = \mathrm{Tr}\hat{\rho}\hat{A} \ , \tag{3.29}$$

where Tr denotes the trace of the matrix. The trace of the density operator has to be 1 for the system to be properly normalised

$$\mathrm{Tr}\hat{\rho} = 1. \tag{3.30}$$

The energy expectation value is therefore

$$\langle H \rangle = \sum_{i} C_i^* D_i H_{ord}\left(z_i^*, z_i\right) \ . \tag{3.31}$$

This quantity is typically conserved better than the energy given above. Also contains the expectation values of the individual modes if the polymorse hamiltonian is used.

3.5.4 norm.out

This file contains the norm of the wave function. The norm is

$$\langle \Psi | \Psi \rangle = \sum_{i} C_i^* D_i \ . \tag{3.32}$$

This should be almost perfectly conserved.

3.5.5 systate.out

This file contains sufficient information to restart the simulations. See the run.inpt file entry above for more details.

Chapter 4

Utility Programs

This chapter describes the many utility programs that I have written for creating basis sets and analysing the output. The more recent programs make use of some general purpose modules that reside in the directory utilities/Modules. The programs that make use of these modules are compiled using Makefiles that will find these modules in this directory without them needing to copied into the programs own directory. This achieved using the following line in the Makefile:

```
VPATH= src:../Modules
```

Modifications to these modules should not change the manner in which the existing functions are called. This is so as not to break any of the other programs that make use of these modules. Currently the modules contained within this directory are as follows:

basisset *This module contains fortran types for storing basis sets. It also contains various routines include routines for calculating overlap matrix elements and reading in configurations from the* `amplitudes.out` *files.*

errors *This simple module contains two simple routines for dealing with errors from opening and reading from files.*

histogram *This module defines a histogram type and functions for adding data to any histograms that are created. It also contains routines for normalising these histograms and writing them to files.*

morse *This module contains data and functions associated with a Morse potential. There available routines include those for reading parameters from a file and projecting coherent states on to a Morse wave function.*

polpak *Strictly this is not a module but a set of routines that I found at http://orion.math.iastate.edu/burkardt/f_src/polpak/polpak.f90. These routines calculate Laguerre polynomials amongst other things. It is used here as Laguerre polynomials appear in the expressions for the wave functions in a Morse oscillator.*

randgen *This file contains a set of functions for choosing numbers from a variety of distributions. I obtained it from the web pages of either Richard*

Chandler or Paul Northrop in the Department of statistics at UCL. I have adapted the set up routine to use system calls for choosing a seed rather than using a temporary file. I have also added the option to write out this seed to the file unit of your choice.

vectorutiles *This module contains a series of functions for rotating vectors and solving 2 and 3 dimensional systems of linear equations by explicit matrix inversion.*

4.1 extractbasisfn.pl

A perl script to extract single basis functions from either `amplitudes.out` or `zpos.out`.

4.2 plotwavefn

A fortran program to project the wave function for a single mode at a particular step onto the x axis.

4.3 trainbasis

This little fortran program takes a simulation of a single basis function and turns it in to a set of basis functions and writes a `states.inpt` file for use with either the istates basis functions or the harmonic oscillator basis. An useful addition would be to perturb slightly the positions. The options are currently:

```
The control file,trains.inpt must contain at least
   nmodes <n>
   nsteps <ns>
 and possibly
   trajmode <mode>  can be either separate or single. Separate corresponds to
                    each mode having a separate file in its own sequentially
                    numbered directory.
                    Single describes the case when all modes are in one file.

 Other entries may be:
   dirstem <stem>  Default: current directory.
   ampfile <name>  Name of file containing qs and ps. Default: amplitudes.out
   qcol <c>       Column containing q for separate mode. Default: 4.
   qcol <c>       Column containing p for separate mode. Default: 5.
   firstcol <c>   Column of first q for single mode. Default: 4.

   randomise      add a random delta q and delta p to each state
                  perturbations are drawn from a circular gaussian
                  probability distribution for z
   rscale  s      Scale the widths of the aforementioned distributions.
                  The widths are s/gamma(i).
```

There is also a script called lotsoftrains whose job it is to run the trainbasis program several times and combine the results. This only really makes sense if the randomise option is used. The lotsoftrains script requires an input file for trainbasis but called trainsinput.inpt in order to distinguish it from trains.inpt. It is called with the first part of the name for the output files and the number of times which trainbasis should be run.

4.4 redistribute

The aim of this program is to read in a trajectory (zpos.out) and write out a new amplitudes.out file in which the points are evenly distributed in space rather than in time. The program is controlled by a file called `redistribute.inpt`. This file should contain the following.

nsteps `ns` *Number of steps in the initial zpos file*

nsegments `nsegs` *Number of equally spaced segments to split the trajectory into.*

The default behaviour maybe modified using the following options.

gamma `g` *Gamma for this mode - used for converting between z a and q,p*

trajfiles *Including this causes the program to read a trajectory file that contains q, p, prob rather than the default zpos.out file.*

traj *The line after should contain the name of the trajectory file to use. If trajfiles is not specified, then this has no effect.*

redisttraj *The line after should contain the name of the new trajectory file to be written. If trajfiles is not specified, then this has no effect. Instead,a file called newamps.out is written.*

closed *This indicates that the trajectory is a closed loop and the first point is added to the end of the trajectory.*

displacefirst `v` *Displace the first point around the new trajectory by vtrajlength/nsegments.*

randdisplacefirst *Same as displacefirst except that v is chosen randomly from the interval [0,1].*

The zpos.out file should contain

```
# Optionally, some comment lines
# Step
t Re(z) Im(z)
# Step
t Re(z) Im(z)
```

Alternatively, a trajectory file should contain

```
# Optionally, some comment lines
q1 p1 prob1
q2 p2 prob2
```

4.5 energyaverage

A program for averaging `energy.out` files. The program can either do a straight average or it can weight the values according to the overlap of the initial state with a specified harmonic oscillator vibrational state. This includes vibrational states which are actually super positions of harmonic oscillator states. The main input file is called `energyaverage.inpt`. The energy.out files are assumed to be in separate directories whose names comprise a common stem followed by a number. If a weighted mean is required then the `states.inpt` files are also read.

The input file `energyaverage.inpt` has the following arguments. Sometimes order matters.

```
                  energy average

Requires an input file called energyaverage.inpt. Reads in
energy.out files from sequentially named directories.
Weighted means also require the states.inpt files.
This file should contain the following.

atype type = type may be either mean or weighted
dirstem Run = first part of directory name.
nruns  46 = Number of different runs e.g. Run1 -> Run46
nsteps 10000 = number of entries in energy.out
ndf 9 = number of degrees of freedom.
      Note: the energy files should contain 2*ndf+3 columns

gamma = a list of gammas as per main ccs code. To be
               followed by ndf lines n gamma.
basis harmosc = harmonic basis. Needed for the weighted
                         mean. Please see man ccs code manual
                         for details
```

4.6 average.pl

This is a perl script for averaging data files. It will average files containing 2 or more columns assuming that the first columns are all the same. It checks to make sure they are and gives an error if it finds any deviation. It will also calculate the standard deviation of the values and insert extra columns if desired. These are useful as the can be used for error bars. Not all files need to have the same number of entries but only in so much as some files may contain fewer or more lines; those lines that are there must have the same values in the first column. Run as

```
average.pl -h
```

to find out more about the usage of this script.

4.7 recombinestates

When a basis set is created by joining trajectories for several different single-mode simulations, the order in which the basis functions are combined matters. In particular, if corresponding points are taken from each mode to form the combined basis function then the norm of the resulting basis set will be low. If points are chosen randomly then the resulting norm is significantly higher. This is the aim of this program. It reads a `states.inpt` file and randomly creates new sets. This program is interactive.

4.8 createbasis

This is a program for creating basis sets. Currently, it can create basis sets for normal mode type systems, triatomic systems and tetraatomic systems. The code requires an input file called `createbasis.inpt`. The system type is now selected using keywords. If no system type is given, the it is assumed to be of normal mode type. Note that the type does not change the basis file that it is created so it is perfectly possible to use the normal mode system type to create a basis function for a triatomic system. It depends what sort of basis set is required.

Rather than using γ, this program actually uses ξ for selecting the basis functions. ξ is taken to be the coherent state width associated with the potential. I.e. the value that gives the ground state wave function. It reserves γ for the width of the basis functions. These may be the same. However, the program allows the basis functions chosen with width ξ to be reprojected on to a set of basis functions with widths γ with the same mean values as the original set. This option has not been used properly nor has it been thoroughly tested.

The input file should contain

ncartdim `n` *Number of carteisan dimensions. Default: 0*

nparticles `n` *Number of particles. Default: 0*

ndf `n` *Number of degrees of freedom. Default: 1*

nbasis `n` *Number of basis functions to create. Default: 1*

mass *This should be followed by a mass for each particle. Default: none*

xi *The keyword should be followed by ndf lines containing an index integer i (which is actually ignored) and then ξ for that degree of freedom. Default: none*

mxi *As for xi, except it only needs to be followed by nparticles lines. The program automatically gives all ncartdim degrees of freedom associated with a particle the same value for γ.*

gamma *The keyword should be followed by ndf lines containing an index integer i (which is actually ignored) and then γ for that degree of freedom. Default: none*

pgamma *As for gamma, except it only needs to be followed by nparticles lines. The program automatically gives all ncartdim degrees of freedom associated with a particle the same value for γ.*

displace *Should be followed by ndf lines containing 3 numbers: s, q0 and p0. s is a dummy index. q0 and p0 are the amounts to displace the mode by. Not sure this option works how it was envisaged.*

systemtype *The type of system to create a basis for. systemtype may be: normalmodes, planartriatomic, tetraatomic or diatomic.*

maxnterms `nt` *This section deals with how to create the basis functions in each mode. What needs to follow will be explained below.*

randseed `rs` *Specify the seed for the random number generator.*

reproject *Reproject basis functions on to a set of functions with the same centres but different widths.*

minsep `m ms` *Specify that the basis functions must be separated by at least ms in mode m.*

minbfprob `prob` *Specify a minimum probability for a basis function.*

engrange `minbfeng maxbfeng` *Specify an energy range in which the basis function must lie.*

The options for choosing basis functions for each degree of freedom or mode follow the maxnterms key word. The keyword is followed by an integer. This is actually only needed when the wave function to be sampled is expressed as the sum of the harmonic oscillator vibrational wave functions. If this is method is not used the maximum number of terms should be set to 0.

The maxnterms line should be followed by a line containing `Method meth`, where meth is the method to be used for the first degree of freedom. The possible methods are single, sum, file and morse. Each of these require their own options to follow. Once all the options have been given, there should be another Method line followed by options for the next degree of freedom. This pattern needs to be repeated for all degrees of freedom.

single

Choose basis functions from a single harmonic oscillator wave function. The Method line should be

```
Method single vibmode comp q0 p0
```

In this, `vibmode` is the vibrational wave function to choose from. It should therefore be an integer equal to or greater than zero. `comp` is a compression factor to use when choosing the basis function. `q0` and `p0` are the position of the centre of the wave function. The basis functions are generated using essentially the same routines as are in the main simulation program for doing the same.

sum

This method is very similar to single except that the wave function is expressed as the sum of harmonic oscillator vibrational wave functions. The method line is followed by another line containing the coefficients of the harmonic oscillator wave functions:

```
Method sum vibmode comp nterms
ec0 ec1 ec2 ec3
```

The second line contains `nterms` coefficients. `nterms` should be less than or equal to `maxnterms` at the beginning of the section.

file

Read basis functions in from a file. There are various options that include cswavefun and pair. The options are quite complicated.

morse

With this method, the basis functions are chosen using the morse module. This module chooses basis functions from the Morse oscillator wave function. Alternatively it can use a file but projects the points on to the Morse wave function in order to obtain the amplitude. The details of the Morse potentials are given in a separate file called `morseham.inpt`. When specifying the Morse method, a number needs to be given that determines which potential in morseham.inpt should be used for this mode. The Method line therefore looks like:

```
Method morse n
```

where n is the integer to choose the potential to be used.

The morseham.inpt file contains details of the morse potentials. A typical file may contain

```
 nmodes 2
mode 1
A 45.314
alpha 2.045
r0 2.667
gamma 192.79
mass 39.975
viblevel 17
phasespace 2.2 4.5 -120 120
file
BrBrEquispaceTraj
mode 2
A 0.50243
alpha 1.67
r0 3.9
gamma 14.74
mass 16.11
```

```
viblevel 0
phasespace 2.5 5.0 -25 25
wfnmc
```

The first line specifies that the file contains two Morse modes. The first begins on the second line and is announced by the keyword mode. A, alpha and r0 are the standard parameters of the Morse potential:

$$V_{morse}(r) = A\left[1 - \exp\left(-\alpha\left(r - r_0\right)\right)\right]^2 \ . \tag{4.1}$$

gamma is the value of gamma corresponding to this potential. It is expected that it is

$$\gamma = \frac{\mu\omega}{\hbar} \tag{4.2}$$

in which μ is the reduced mass of the mode however there is some evidence that for highly excited vibrational levels that a reduced value may give better results. The reduced mass of the mode μ is given on the following line and is introduced by the keyword mass. The next line gives the vibrational level that is to be sampled. Unfortunately, I don't know a form for the Morse wave function in phase space comparable to equation 3.1 for the harmonic oscillator. Consequently, with the exception of using a predetermined list of points, basis functions are chosen at random from a rectangular region of phase space whose limits are given by the numbers following the keyword phasespace. The first two denote maximum and minimum positions in Å where as the second two numbers give the maximum and minimum allowable momenta in Å u (157.461 fs)$^{-1}$ where u is the unified atomic mass and 157.461 fs is the internal unit of time used by the program. This should really be modified, perhaps by using a different keyword, to accept momenta in Å u fs^{-1} .

The next line indicates when the basis functions should be accepted after they have been randomly chosen from the specified region of phasespace. There options are:

energy *Accept a basis function chosen if its energy lies within a specified range. This range should be given on the next two lines. The first line should start with the keyword* `mineng` *and be followed by the corresponding minimum acceptable energy. The second line should start with the keyword* `maxeng` *and be followed by the corresponding maximum acceptable energy.*

montecarlo *This option specifies that the basis function should be accepted according to a Mont Carlo move dependent on its energy. The acceptance probability is* $\exp\left(-(E - E_n)^2/(2\sigma^2)\right)$ *in which* E_n *is the energy of the vibrational level specified above. The width* σ *is specified on the next line and should be introduced by the keyword* `sigma`*. Acceptance is then governed by the value of a number selected randomly on the range [0,1].*

wfnmc *This denotes that a wave function Mont Carlo acceptance should be used. This is similar to the montecarlo method except that the acceptance probability is given by* $|\langle z|\psi_n\rangle|^2$ *in which* ϕ_n *is the wave function of the vibrational level specified above.*

grid *This option does not actually use points chosen randomly from the specified region of phase space. Instead it chooses points from a grid that coves*

it. The number of rows and columns may be specified by specifying the keywords `nq` *and* `np` *followed by the desired integers. If either of these lines are omitted, their corresponding value defaults to 21. The points to be included can be reduced by specifying minimum and maximum acceptable energies using the optional keywords* `mineng` *and* `maxeng` *followed by the desired values. If either of these are omitted then the correspond bound default to* 10^{10} *or* -10^{10} *as appropriate.*

file *Read in points for this mode from the file whose name appears on the next line. With this option, although necessary for the parsing routine, the phasespace region is redundant.*

The contribution to the amplitude for the multidimensional basis function is obtained by projecting the basis function on the desired Morse wave function. The energy of the n*th* vibrational level of a Morse oscillator is

$$E_n = 2|\alpha|\hbar \left(\frac{A}{2\mu}\right)^{\frac{1}{2}} \left(n+\frac{1}{2}\right) - \frac{\alpha^2\hbar^2}{2\mu}\left(n+\frac{1}{2}\right)^2 . \tag{4.3}$$

and the corresponding wave function is

$$\psi_n(r) = N_n \exp(\xi(r))\,\xi(r)^{b_n} L_n^{2b_n}(\xi(r)) \tag{4.4}$$

in which

$$\xi(r) = 2\frac{\sqrt{2\mu A}}{\alpha\hbar}\exp\left[-\alpha(r-r_0)\right] \tag{4.5}$$

and

$$b_n = \frac{\sqrt{-2\mu E_n}}{\alpha\hbar} = \frac{\sqrt{2\mu A}}{\alpha\hbar} - \frac{1}{2} - n . \tag{4.6}$$

In equation 4.4, $L_n^{2b_n}(\xi)$ are associated Laguerre polynomials and N_n is the normalisation constant. A, α and r_e are parameters of the Morse potential, equation 4.1, and μ is the reduced mass of the bond.

The projection of a 1D basis function onto the Morse wave function is

$$\begin{aligned} C &= \langle z|\psi_n\rangle \\ &= \int \langle z|x\rangle\langle x|\psi_n\rangle \,\mathrm{dx} . \end{aligned} \tag{4.7}$$

in which$\langle z|x\rangle$ is given by the complex conjugate of equation 1.1 and $\langle x|\psi_n\rangle = \psi_n(x)$. The program evaluates this integral numerically.

4.9 decay

Program to calculate the dissociation in systems of the form Br_2Ne_N. The dissociation is calculated using the distance between the centre of the Br-Br vector and each Ne in turn.

It is possible to specify several options in the input file `decay.inpt`.

nbasis `n` *Number of basis functions*

rcut `rc` *Cutoff Radius*

nneons `ne` *Number of Neon atoms*

nbromine `nbr` *Number of Bromine atoms*

gammaBr `gbr` *Gamma for Bromine atoms. Default: 385.57 / $Å^2$*

gammaNe `gne` *Gamma for Neon atoms. Default: 10.1 / $Å^2$*

writedissconfig *Write out configuration at dissociation. Default: false*

t0 `t0` *Specify the initial time*

pad *Start the plot of dissociation rate from 0 even if t0 ¿ 0. Default: false*

dt `dt` *Time between configurations*

simweight `sw` *For doing secondary dissociation. Simply writes this as the weight in the dissociation file header to be used by he averaging program. Default: 1*

multifile `nf` *The trajectory has been split into nf files. The names of which follow on separate lines.*

nfiles `nf` *Calculate the dissociation rate using trajectories in several files, the names of which follow separate lines. Used for send stage decay. The final dissociation probability is obtained by summing the dissociation probability calculated from each file.*

ampfile *Name of a single ampltiudes.out file to use follows on a separate line. Default: amplitudes.out*

4.10 brenergy

Calculates the brenergy upon dissociation of a Ne. Dissociation is as for the decay program. The program is controlled through a file called brenergy.inpt which may contain the following options.

nbasis `nbf` *Number of basis functions*

rcut `rc` *Only include basis functions for which $r_{BrBr} > rc$. Default: 12 Å.*

nbins `nb` *Number of bins for the energy distributions. Default: 100*

min `engmin` *Minimum energy for Br-Br energy histogram. Default: -12 kJ/mol*

max `engmax` *Maximum energy for Br-Br energy histogram. Default: 02 kJ/mol*

brnemin `engmin` *Minimum energy for energy histogram*

brnemax `engmax` *Maximum energy for energy histogram*

gammaBr `gBr` *Br gamma. Default: 385.57 $Å^{-2}$*

gammaNe `gNe` *Ne gamma. Default: 10.15 $Å^{-2}$*

quantise *Calculate projections of wave function/dissociated basis function on to Br-Br vibrational modes.*

vrange `vmin vmax` *The initial Br-Br vibrational state ν_i is specified in morseham.inpt. This gives the range of states over which to calculate the projection. I.e. project on to vibrational levels $\nu_i + \delta\nu$ with $\delta\nu \in$* [`vmin, vmax`].

ampfile *The name of a single amplitude file should appear on the following line.*

multifile `nf` *The trajectory has been split into nf files. The names of which follow on separate lines.*

suffix *A suffix to append to the names of the output files should appear on the following line.*

tmin `tmin` *Do not include dissociation events that happen before tmin in the distributions. Default: tmin* $= -10^{20}$.

tmax `tmax` *Stop calculating the distributions at tmax. Default: tmax* $= 10^{20}$.

4.11 autocorrelation

This program calculates an autocorrelation function from a series of amplitudes files. The autocorrelation function is

$$P(t) = |\langle\psi(0)|\psi(t)\rangle|^2 \ . \tag{4.8}$$

In CCS,

$$\psi(t) = \sum_{j=1}^{N} D_j(t) \exp\left(iS_j(t)\right) |\mathbf{z}_j\rangle \ . \tag{4.9}$$

The autocorrelation function is therefore

$$P(t) = \left| \sum_{j=1}^{N} \sum_{k=1}^{N} D_j^* D_k \exp\left(i\left(S_k(t) - S_j(t)\right)\right) \langle\mathbf{z}(0)|\mathbf{z}(t)\rangle \right|^2 \ . \tag{4.10}$$

4.12 Histogram

This is a small program to create a histogram from a file containing a column of data. The program allows the column containing the data to be specified. Currently only handles implicitly equally weighted data.

Chapter 5

Variables

This chapter lists the main variables located in each module. Large arrays should go into the arrays module. Parameters for the hamiltonian are in the hamiltonian module and the options module holds other details.

However, I am afraid that this chapter is probably now some what out of date. A lot of it should still be relevant though.

5.1 Options module

Units

tu [real] factor to convert system time units to femtoseconds for output

energyunits [character string] energy units for output files.

massunits [character string] containing mass units for output files.

Control variables

nsteps [integer] Number of steps to perform

dt [real] Time step

classical [logical] Do a purely classical simulation

basismethod [integer] Method for assigning basis functions

1. grid
2. random from a gaussian distribution
3. initial states (from states.inpt file)
4. harmonic basis functions

minprob [real] Minimum probability for basis functions. If probability becomes less than this for a particular basis function then that basis function is effectively removed. Set to a negative value to turn off basis function removal.

maxz [real] As a above but where z in each degree of freedom is constrained to be less than this value. Set to a negative value to turn off removal.

verbosity [integer] How much info should be written to output file.

Output Control

The following are all the period at which the quantities are calculated and written to file. Set to zero to not calculate/write out a particular quantity.

ccorlog [integer] correlation function.

hcorfreq [integer]harmonic oscillator correlation function.

writeposfreq [integer] positions

writeCDfreq [integer] C and D.

writeampfreq [integer] amplitudes

writeSfreq [integer] classical action

writeEfreq [integer] energy

writeacffreq [integer] autocorrelation function

writenormfreq [integer] wave function norm

dumpfreq [integer] restart file.

5.2 Arrays module

t [real] Current time in this simulation

pt [real] Accumulated time in previous simulations. Total time is t+pt.

psteps [integer] The number of steps done before this run.

toteng [real] Total (quantum) energy of the system.

dtoteng [real] Change in total energy due to the removal of some basis functions.

zb [complex array, dimension ndf,nbasis] The basis functions.

S [real array, dimension nbasis] The classical action.

C [complex array, dimension nbasis] wave function coefficient.

D [complex array, dimension nbasis] wave function coefficient.

initnorm [real] Initial norm of the wave function.

incbf [logical array, dimension nbf] Determines if each basis function is included or not.

oldincbf [logical array, dimension nbf] Working array for determining if each basis function is included or not.

Initial states/setting up basis functions

Working arrays

Unless you have just assigned values, don't assume that they have correct or even sensible values.

zbnew, zk1, zk2, zk3, zk4 [complex arrays, dimension ndf,nbasis]

Snew, Sk1, Sk2, Sk3, Sk4 [real arrays, dimension nbasis]

Cnew, Ck1, Ck2, Ck3, Ck4, Dold [complex arrays, dimension nbasis]

overlpm [complex array, dimension nbasis,nbasis] The overlap matrix

FERR, BERR, RWORK [real arrays, dimension] For the LAPACK routines.

ipiv [integer array, dimension nbasis] For the LAPACK routines.

WORK [complex array, dimension 2×nbasis] For the LAPACK routines.

LWORK [integer] size of WORK.

overlpmf [complex array, dimension nbasis,nbasis]

Bibliography

[1] D. V. Shalashilin and M. S. Mark S. Child. The phase space CCS approach to quantum and semiclassical molecular dynamics for high-dimensional systems:. *Chem. Phys.*, 304:103–120, 2004.