

# Exact Diagonalisation Program EDP\*

Matthias Meister

August 2007

## 1 Purpose of the program

The program finds the eigenvalues and eigenvectors of a Hamiltonian matrix by numerical diagonalisation and outputs these results into files. The Hamiltonian can be of the following general structure:

$$\hat{H} = \sum_{k=1}^F \varepsilon_k c_k^\dagger c_k + \sum_{s=1}^{N_B} \hbar\omega_s (A_s^\dagger A_s + \frac{1}{2}) + \sum_{kl=1}^F t^{kl} c_k^\dagger c_l + \sum_{klmn=1}^F V^{klmn} c_k^\dagger c_l^\dagger c_m c_n + \hat{W} \quad (1)$$

where the quantities have the following meaning:

$\varepsilon_k$  : energy of single-fermion state  $k$ , in total there are  $F$  single-fermion states;  $c_k$  is the corresponding annihilation operator. The fermion states are orthonormal. There can only be one type of fermion, these fermions are indistinguishable.

$N_B$  : number of harmonic oscillators in the system; oscillator quanta  $\hbar\omega_s$ , corresponding annihilation operator is  $A_s$ .

$t^{kl}$  : fermion hopping integrals between single-fermion states  $k$  and  $l$ . The relation  $t_{kl} = t_{lk}$  must hold.

$V^{klmn}$  : fermion-fermion interaction coefficients.

$\hat{W}$  : coupling between fermions and oscillators. Has not been specified further here as it can be rather diverse.

These quantities, alongside others, can be specified in the program, see below section 2. The program takes these parameters, builds the Hamilton matrix based on them, diagonalises the matrix and writes a file with the eigenvalues (in ascending order) and a file with the corresponding eigenvectors.

## 2 Program Structure

The source code of the program consists of the following parts:

- EDP.f90 : main program; calling subroutines and writing the output.
- EDP-SETUP.f90 : module; parameters are specified, subroutines generate coefficients for the Hamiltonian.
- EDP-HAMILTON.f90 : module; builds the Hamilton matrix from the coefficients generated before. No changes are required in this module.
- CONSTANTS.f90 : module; used in various programs, contains some useful constants, only some are used in the program.

---

\*For lack of a better name

## 2.1 Main program EDP.f90

The variables `lwork`, `work`, and `info` are required by the `acml`-routine `dsyev` which is used in this program unit to diagonalise the Hamilton matrix. The other variables occurring have been defined in the modules. The program first calls the routine `GETKOEFF` from the `SETUP`-module to read the parameters and generate the coefficients. Then the routine `HAMGEN` from the `HAMILTON`-module is called, which builds the Hamilton matrix from these coefficients.

The Hamilton matrix `Hamil` is copied into the array `EigVec`. This is not really necessary, but allows to keep `Hamil` for other purposes and instead use `EigVec` as parameter in the `dsyev`-call, where it is changed and will hold the eigenvectors after the routine has finished.

The filenames for the results are specified in the `OPEN`-instructions. The eigenvectors and -values are written into the files. The eigenvalue file has two columns, first column is just an index numbering the eigenvalues, second column holds the eigenvalue. The eigenvalues are in ascending order. The eigenvector file has three columns, the second column holds the number of the eigenvalue to which the vector corresponds, the first column gives the number of the component within the eigenvector, and the third column the value of the component.

## 2.2 Module EDP-SETUP.f90

In this module the system is specified by giving parameters of the Hamiltonian and other quantities relevant for the program. The module also holds the routines to build the coefficients for the contributions to the Hamiltonian. The relevant parameters are specified in these subroutines. The quantities that have to be defined before the `CONTAINS`-statement are

- `SINFER` : Number of single-fermion states
- `NUMFER` : Number of fermions in the system
- `FERSTA` : Number of fermion states
- `NBOS` : Number of oscillators in the system
- `DHAM` : Total (linear) dimension of the Hamiltonian, i.e. number of states of the system
- `BOSSTA` : One-dimensional array of length `NBOS`, holding the number of oscillator states to be used for each oscillator. Please note that in this program the oscillator states are counted from 1, i.e. the oscillator ground state corresponds to the state with index 1.
- `MBOS` : Maximum value of the entries of the array `BOSSTA`

Note:

$$\text{FERSTA} = \frac{\text{SINFER}!}{(\text{SINFER} - \text{NUMFER})!\text{NUMFER}!}, \quad \text{DHAM} = \text{FERSTA} \prod_{b=1}^{\text{NBOS}} \text{BOSSTA}(b).$$

### 2.2.1 Routine GETKOEFF

Calls the other routines in this module.

### 2.2.2 Routine GETSFER

Specify the array `SFE` of length `SINFER`. This array holds the energies of the single-fermion states, i.e. `SFE(k)` corresponds to  $\varepsilon_k$  in (1).

### 2.2.3 Routine GETSBOS

Specify array `quosc` of length `NBOS`. The array holds the oscillator quanta; `quosc(s)` corresponds to  $\hbar\omega_s$  in (1). From the array `quosc` the auxiliary array `SBOS` is derived.

### 2.2.4 Routine FIXHOPP

In this routine the hopping integrals  $t^{kl}$  in (1) are specified as array elements `HOPP(k,l)`. If many of these are equal, it is more convenient to assign a value to the variable `chop` and set the elements of the array via this variable.

### 2.2.5 Routine FIXBFI

The fermion-oscillator interaction is specified in this routine through the array `BFI`. This array is five-dimensional. Index five specifies the oscillator, so this index takes values from 1 to `NBOS`. Index one and two give the states of this oscillator connected by this interaction, and indices three and four determine the fermion states involved. Note that though a fermion can interact with all oscillators separately, each such interaction is with one oscillator only. Also, there is always only one electron involved, i.e. it is a two-particle (oscillator-fermion) interaction. To better illustrate the meaning of the indices, consider the case of a system containing only one oscillator and one fermion. The states of the system shall be denoted  $|N; k\rangle$ , where  $N$  is an oscillator state and  $k$  a fermion state. Then, with  $\hat{W}$  from (1) we have  $\langle M, k | \hat{W} | N, l \rangle = \text{BFI}(M, N, k, l, 1)$ .

### 2.2.6 Routine FIXFFI

The fermion-fermion interaction coefficients  $V^{klmn}$  from (1) are given here as elements of the array `FFI`,  $V^{klmn} = \text{FFI}(k, l, m, n)$ . For an easy overall scaling of the interaction the variable `cff` can be used.

### 2.2.7 Function IOC

*No change should be required in this function.* The function takes two one-dimensional arrays, `bvec` and `fvec` as input and returns an integer number. The array `bvec` has length `NBOS` and holds the states of the oscillators of the system, e.g. in the case of four oscillators `bvec` =  $(N_1, N_2, N_3, N_4)$  means that oscillator 1 is in state  $N_1$ , oscillator 2 in state  $N_2$ , and so on. In the same fashion `fvec`, of length `NUMFER`, holds the states of the fermions in the system. The entries of `fvec` are always in ascending order. The number returned by the function corresponds to the index of the basis state of the total system with oscillators and fermions as given by the input arrays.

### 2.2.8 Routine STI

*No change should be required in this routine.* The routine is the ‘inverse’ of `IOC`. It converts a basis state index  $r$  into oscillator and single-fermion indices and fills vectors `bvec`, `fvec` with these results.

### 2.2.9 Function SKP

*No change should be required in this function.* The function is used to evaluate the inner product of two many-fermion states, represented by arrays `A` and `B` that hold the single-fermion state indices. These single-state indices can be in any order (the function works out the correct sign).

It is even possible that within one array indices occur more than once (in which case the result will be 0). Note that the contents of the arrays **A** and **B** are changed (rearranged).

#### **2.2.10 Function ISORT**

*No change should be required in this function.* Takes an array **A** as input and rearranges its elements in ascending order. The function returns the parity of the permutation necessary to achieve this reordering or 0 if some elements occur more than once. The function is used by SKP.

#### **2.2.11 Function delta**

*No change should be required in this function.* Ordinary Kronecker delta, i.e.  $\text{delta}(m, n) = \delta_{mn}$ .

### **2.3 Module EDP-HAMILTON.f90**

*No change should be required in this module.* The routines in this module use the quantities generated in the **SETUP**-module to build the Hamilton matrix. The module also contains the definitions for the arrays **Hamil** and **EigVec** that are intended to hold the Hamilton matrix and the eigenvectors, respectively, as well as the array **EigVal** for the eigenvalues.

#### **2.3.1 Routine HAMGEN**

The routine calls the other routines in this module to build the Hamiltonian. Then the Hamilton matrix is checked for symmetry and an error message is issued if the symmetry is violated by more than a certain threshold. All the routines listed below work through nested loops to perform their calculation. **OpenMP** loop-parallelism has been implemented in each routine.

#### **2.3.2 Routine GETSINGLE**

Adds the single-particle energies for fermions and oscillator quanta to the Hamiltonian.

#### **2.3.3 Routine GETHOPP**

The hopping term matrix elements are added to the Hamiltonian.

#### **2.3.4 Routine GETBF**

Puts the fermion-oscillator interaction in the Hamilton matrix.

#### **2.3.5 Routine GETFF**

The fermion-fermion interaction is added to the Hamiltonian.