# The TEOQuS library: open source software for multimode driven quantum systems
# User manual

G. A. Sinuco-León

*Department of Physics and Astronomy, University of Sussex,Brighton, BN1 9QH, UK*

January 17, 2019.

## 1 Software description

In section we provide the header of each one of the subroutines of the library, including the argument declaration to help the user to identify the type of variable expected by each function.

## 2 Multimode expansion of the time-dependent Schrodinge equation

The library can be used to calculate the time-evolution operator, $U(t',t),\ t'>t$, of systems whose Hamiltonian has the form:

$$H = \sum_{i,j}^{D} E_{i,j} |i\rangle\langle j| + \sum_{i,j}^{D}\sum_{\ell=1}^{N}\sum_{n\in Z} V_{i,j}^{\ell,n} e^{in\omega_\ell t} |i\rangle\langle j| + \text{h.c.} \tag{1}$$

where $D$ is the dimension of the Hilbert space, $E_{i,j}$ defines a static component of $H$, $V_{i,j}^{\ell,n}$ is the coupling between the states $i$ and $j$ oscillating at frequency $n\omega_\ell$ (i.e. the $n$-th harmonic of the $\ell$-th fundamental frequency $\omega_\ell$) and $N$ is the number of incommensurately frequencies.

To calculate the time-evolution operator we generalise the Rotating (or Resonant) Wave Approximation (RWA), taking into account the complex time dependence of eq. (1). For this, we rephrase the problem in terms of building a time-dependent unitary transformation, $U_F(t)$ to a new basis $\{|\bar{i}\rangle\}$, that leads to a *time-independent* and diagonal Hamiltonian, $\bar{H}$. After applying the standard quantum-mechanical transformation rule to the Schrödinger equation [?, ?], this condition becomes:

$$U_F^\dagger(t)\left[H(t) - i\hbar\partial_t\right]U_F(t) \;\;=\;\; \sum_{\bar{i}} \bar{E}_{\bar{i}} |\bar{i}\rangle\langle\bar{i}| \tag{2}$$

Importantly, in the basis of states defined by this transformation the time evolution operator is diagonal and has the form:

$$\bar{U}(t',t) = \sum_{\bar{i}} e^{-i\bar{E}_{\bar{i}}(t'-t)} |\bar{i}\rangle\langle\bar{i}| \tag{3}$$

which let us to calculate the time evolution operator in the original basis $\{|i\rangle\}$, just by inverting the transformation $U_F(t)$, according to [?]:

$$U(t',t) = U_F(t')\bar{U}(t',t)U_F(t) \tag{4}$$

To formulate a fully defined computational problem, we express the unitary transformation $U_F(t)$ as the multifrequency Fourier series [?]:

$$U_F(t) = \sum_{\vec{n}} U^{\vec{n}}_{i,\bar{i}} e^{-i\vec{\omega}\cdot\vec{n}t} |i\rangle \langle\bar{i}| \tag{5}$$

where $\vec{\omega} = (\omega_1, \omega_2, \ldots, \omega_N)$ and $\vec{n}$ is a $N$-dimensional vector of integers. After plugging this expansion in eq. (2) and performing an integral over time, we obtain a fully defined eigenproblem for the eigenvalues $\bar{E}_{\bar{i}}$ and Fourier components of the unitary transformation $U^{\vec{n}}_{i,\bar{E}}$:

$$\sum_{\ell,m}\sum_{i,i'}\sum_{\vec{n}} V^{\ell,m}_{i,i'} U^{\vec{n}_m *}_{i,\bar{i}} U^{\vec{n}_m}_{i',\bar{i}'} - \hbar\vec{\omega}\cdot\vec{n}\delta_{\bar{i},\bar{i}'}\delta_{\vec{n},\vec{n}'} = \bar{E}_{\bar{j}}\delta_{\bar{i},\bar{i}'} \tag{6}$$

where $\vec{n}_{\ell,m} = \vec{n} + mP_\ell$ with $P_\ell = (0,\ldots,1,\ldots,0)$ the projector at the $\ell-$th position. To obtain a finite matrix representation of this problem we truncate the sum over the number of modes of the Fourier expansion eq. (5). Below, in Appendix A, we show an specific example of the shape of the matrix.

This formulation to calculate the time-evolution operator is equivalent to the multimode Floquet representation of the Hamiltonian that introduces an extended Hilbert space $|E_i, \vec{n}\rangle$ [?, ?]. However, the semiclassical description presented here makes emphasis in the physically accessible states.

## 3 Usage

As a concrete example, here we illustrate the use of the library functionality considering a two

the code is

### 3.1 Setting the stage

The initialising two derived types:

```
TYPE(ATOM)                                      ID
TYPE(MODE),       DIMENSION(:),   ALLOCATABLE :: FIELDS
```

`IDwill` store information about the type of system and the number of levels. `FIELDSis` an type that will store the componet of the hamiltonian or information needed to buiild the matrices .

The parameters of `IDare` initialised by calling the subroutine:

```
INFO = 0
CALL FLOQUETINIT('qubit','U',0.2D1,ID,INFO)
```

The first argument (qubit) labels the type of systems, the second indicate, the third the number of levles and returen ID initilised. the ... of this funciton is

all of arguments are optional. If none argument is passed the function return error. If the system to study does not belong to the listed ones, then the componenet of ID must be initialised by hand: ID

```
ID%id_system  = 4
ID%D_BARE =  Number of energy states
ALLOCATE(ID%E_BARE(n)) : an array to store the energies
```

system init For a number of systems the library includes this the subroutine FLOQUETINIT(atomicspecie,manifold,JTOTAL,ID,info)

! ATOMICSPECIE: 87Rb,6Li,Cs,41K,qubit,lattice, SPIN ! MANIFOLD : ”U” UPPER HYPERFINE MANIFOLD, ”L” LOWER HYPERFIND MANIFOLD, ”B” BOTH ! JTOTAL : IF ATOMICSPECIE .EQ. SPIN THEN JTOTAL IS THE TOTAL ANGULAR MOMENTUM OF THE SPIN ! IF ATOMICSPECIE .EQ. LATTICE, THEN JTOTAL IS THE NUMBER OF SITES

For an arbitrary system, it is not necesary to call this function and the matrix componets can be inditialized directly

the number of driving fields in the vector

This sequence of instructino indicates that the qubit is driven by two fundamental frequencies. The first frequency has a a single harmonic (modes(2) = 1) while the second frequency has the fundamental and the first harmonic (`modes_num(3)=2`).

The driving fields are then initialize in the derived tye FIELDS. For the system of interes the Hamiltonian componet can be by scalar parameteers and they are defined explicitily. The frequency and the number of the expansio are defined by , respectively. Notice that the dimension of the `total_frequencies=3` corresponding to the static plus two driving frequencies.

## 3.2 Hamiltonian components

The user should allocated space of each matrix component of the Hamiltonian. If the system is one of the defaults, them the Hamiltonian componet have defined forms which are initialized, after defieinnin scalar parameters correwsondin tothe couplings the . In the particular case, this is done as follows

`CALLSETHAMILTONIANCOMPONENTS(ID,size(modes_num,1),total_frequencies,MODES_NUM,`
.

defines each one of the contributiosn to the Hamtilton and stores them in the matrix `fields()%V`.

a spares representation of the hamiltnia cmponets is build in the and stored in the arrays, whose name is selfexplanatory. This representation is needed to build the

For an arbitray system, each matrix should be defined element by element (see e.g random matrix example)

## 3.3 Multimode Floquet matrix and diagonalisation

Once the coonents of the hamiltonian are defined (ie.e the complete set of matrices `FIELDS(mode)%V`), the Multimode Hamiltonian cna be initialized callin the funciton.

```
  CALL MULTIMODEFLOQUETMATRIX(ID,size(modes_num,1),total_frequencies,MODES_NUM,F
```

As a result of this call, the system stores the full multimode Floquet matrix in `H_FLOQUET`,whose size is calcuated internally. A sparse representation of this matrix is obtained using teh call.....

```
  CALL MULTIMODEFLOQUETMATRIX_SP(ID,size(modes_num,1),total_frequencies,MODES_NUM
```

which produces three vectors .....

The library includes wrappers to diagonalisation subroutines of the lapack of the MKL-intel library (for the sparse representation). These function are called using:

lapack `CALLLAPACK_FULLEIGENVALUES(H_FLOQUET,SIZE(H_FLOQUET,1),E_FLOQUET,INFO)`.

MKL

In all cases the eigenvalues are stores in .. and the eigen fuciton in ... These eigenvectors are the multimode Foureire decomposition of the tiem evluion operator ....

## 3.4 Time-evolution operator

The time evoluiton is evlauted using the, betwteen t1 and t2 callin the funciton.... `CALLMULTIMODETIMEEVOLUTINOPERATOR(SIZE(U_F,1),SIZE(MODES_NUM,1),MODES_NUM,U_F,E_`
.

## 3.5 Micromotion operator

The micromotion operator is the instantatnous transofrmation. Assuming we know the foure decomposiontino, then the intant tante ou evaluated using .... This is done using the subroutine

`CALLMULTIMODEMICROMOTION(ID,SIZE(U_F,1),NM,MODES_NUM,U_F,E_FLOQUET,ID%D_BARE,`
.

## 3.6 Identifying the dressing modes

In several application is useful to define a dressed.. for this, the user can identify a subset of driving field, e.g. The library uses the if withn the set of `modes_num`identify a subset of , e.g. ....

lines .. allocate needed memory space, and the

`CALLMICROMOTIONFOURIERDRESSEDBASIS(ID,DRESSINGFIELDS_INDICES,MODES_NUM,FIELDS,`

```
CALL MICROMOTIONDRESSEDBASIS(ID,MODES_NUM,DRESSINGFIELDS_INDICES,FIELDS,U_FD,&
 & E_DRESSED,T1,U_F1_red,INFO)
```

then the (micromotion operator) is defined calling the function....

and the micromotion operator at tiem t is evaluated with the function. Then, the time evolution operator in the dressed basis can be writen as:

## 3.7 driven routine

The time evolution perato can be calculated calling the funcition.. ..

# 4 MODULES

The library include several

## 4.1 Physical Constants

The module `physical_constants`defines the default values of commonly used parameters defining the Hamiltonian of atomic systems. The values are based on the .... . The user can in src/

```
MODULE physical_constants
  IMPLICIT NONE
  DOUBLE PRECISION, PARAMETER :: pi           = 4.0*ATAN(1.0)
  DOUBLE PRECISION, PARAMETER :: e            = 1.602176462E-19
  DOUBLE PRECISION, PARAMETER :: h_P          = 6.62606957E-34
  DOUBLE PRECISION, PARAMETER :: hbar         = h_P/(2.0*4.0*ATAN(1.0))
  DOUBLE PRECISION, PARAMETER :: mu_B         = 9.27400968E-24
  DOUBLE PRECISION, PARAMETER :: k_B          = 1.3806488E-23
  DOUBLE PRECISION, PARAMETER :: mu_cero      = 12.566370614E-7
  DOUBLE PRECISION, PARAMETER :: epsilon_cero = 8.854187817E-12
  DOUBLE PRECISION, PARAMETER :: amu          = 1.660538921E-27
  DOUBLE PRECISION, PARAMETER :: g_t          = 9.8
  DOUBLE PRECISION, PARAMETER :: SB_ct        = 5.6704E-8
  COMPLEX*16,       PARAMETER :: J_IMAG       = DCMPLX(0.0,1.0)
  DOUBLE PRECISION, PARAMETER :: speedoflight = 299792458.0
  DOUBLE PRECISION            :: TOTAL_TIME
END MODULE physical_constants
```

## 4.2 Arrays

The module `ARRAYS`provides global definitions of matrices. The user cannot define variables using any of the names in this.

```
MODULE ARRAYS

  DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: Identity,CLEBSH_GORDAN_JtoF,j
  COMPLEX*16,       DIMENSION(:,:), ALLOCATABLE :: H_hyperfine,HAMILTONIAN,H_RF,H
  COMPLEX*16,       DIMENSION(:,:), ALLOCATABLE :: H_RF_DAGGER,H_ALPHA_DAGGER,H_A
  COMPLEX*16,       DIMENSION(:,:), ALLOCATABLE :: observable, observable_extende
  DOUBLE PRECISION, DIMENSION(:),   ALLOCATABLE :: W_SPACE,W_SPACEF,W_SPACEF_0,E_
  DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: Fx,Fy,Fz,g_F_matrix
  COMPLEX*16,       DIMENSION(:,:), ALLOCATABLE :: Hamiltonian_F,Identity_F,H_AUX
  COMPLEX*16,       DIMENSION(:,:), ALLOCATABLE :: H_FLOQUET_INTERACTION,H_FLOQUE
  INTEGER,          DIMENSION(:,:), ALLOCATABLE :: F_t,H_w,H_J,H_M,Jz_dash,Fz_das
  INTEGER,          DIMENSION(:),   ALLOCATABLE :: index_state
  INTEGER                                       :: KD
  DOUBLE PRECISION, DIMENSION(3)                :: POSITION,DELTA_POSITION
```

```
END MODULE ARRAYS
```

## 4.3 Atomic properties

The ATOMIC_PROPERTIESmodule defines the default physical parameters of

```
MODULE ATOMIC_PROPERTIES
  USE physical_constants
  IMPLICIT NONE
  DOUBLE PRECISION :: L=0.0,  S = 0.5
  DOUBLE PRECISION :: mass_at = 87*amu
  DOUBLE PRECISION :: I,g_I,g_J
  DOUBLE PRECISION :: J,F,gf,mf
  DOUBLE PRECISION :: gF_2,gF_1,G_F
  DOUBLE PRECISION :: A,a_s,alpha_E
  INTEGER          :: Fup,Fdown,Ftotal
  INTEGER          :: Total_states_LSI
  CHARACTER(LEN=7) :: ID_name

  !87Rb
  DOUBLE PRECISION :: I_87Rb   =  1.5
  DOUBLE PRECISION :: J_87Rb   =  0.5
  DOUBLE PRECISION :: gJ_87Rb  =  2.0
  DOUBLE PRECISION :: gI_87Rb  = -0.000995
  DOUBLE PRECISION :: A_87Rb   =  2*pi*hbar*3.417341E9
  DOUBLE PRECISION :: a_s_87Rb = 5.77E-9
  DOUBLE PRECISION :: alpha_E_87Rb = 2*pi*hbar*0.0794*1E-4
  INTEGER          :: Fup_87Rb     =  2
  INTEGER          :: Fdown_87Rb   =  1
  CHARACTER(LEN=7) :: ID_name_87Rb = "87Rb"

  !6Li
  DOUBLE PRECISION :: I_6Li   =  1.0
  DOUBLE PRECISION :: J_6Li   =  0.5
  DOUBLE PRECISION :: gJ_6Li  =  2.0
  DOUBLE PRECISION :: gI_6Li  = -0.000995
  DOUBLE PRECISION :: A_6Li   =  2*pi*hbar*152.137E6
  DOUBLE PRECISION :: a_s_6Li = 5.77E-9
  DOUBLE PRECISION :: alpha_E_6Li = 2*pi*hbar*0.0794*1E-4
  INTEGER          :: Fup_6Li     =  1
  INTEGER          :: Fdown_6Li   =  1
  CHARACTER(LEN=7) :: ID_name_6Li = "6Li"

  !qubit
  DOUBLE PRECISION :: I_qubit   =  0.0
  DOUBLE PRECISION :: J_qubit   =  0.0
  DOUBLE PRECISION :: gJ_qubit  =  1.0
  DOUBLE PRECISION :: gI_qubit  =  0.0
  DOUBLE PRECISION :: A_qubit   =  1.0
  DOUBLE PRECISION :: a_s_qubit =  0.0
  DOUBLE PRECISION :: alpha_E_qubit = 0.0
  INTEGER          :: Fup_qubit     =  1
  INTEGER          :: Fdown_qubit   =  1
  CHARACTER(LEN=7) :: ID_name_qubit = "qubit"


  !spin
  DOUBLE PRECISION :: I_spin   =  0.0
  DOUBLE PRECISION :: J_spin   =  0.0
  DOUBLE PRECISION :: gJ_spin  =  1.0
  DOUBLE PRECISION :: gI_spin  =  0.0
  DOUBLE PRECISION :: A_spin   =  1.0
  DOUBLE PRECISION :: a_s_spin =  0.0
  DOUBLE PRECISION :: alpha_E_spin = 0.0
  INTEGER          :: Fup_spin     =  1
  INTEGER          :: Fdown_spin   =  1
  CHARACTER(LEN=7) :: ID_name_spin = "spin"


  !lattice
  CHARACTER        :: PERIODIC
  CHARACTER(LEN=7) :: ID_name_lattice = "lattice"

END MODULE ATOMIC_PROPERTIES
```

### 4.4 MKL

```
MODULE FEAST
  integer     fpm(128)
  real*8      Emin,Emax
  real*8      epsout
  integer     loop
  integer     M0 ! initial guess
  integer     M1 ! total number of eigenvalues found
  integer     info_FEAST
  real*8,     DIMENSION(:),   ALLOCATABLE :: E, RES ! vector of eigenvalues
  complex*16, DIMENSION(:,:), ALLOCATABLE :: X      ! matrix with eigenvectore
END MODULE FEAST
```

## 5 DERIVED TYPES (src/modes.f90)

The derived type defined

```
MODULE TYPES
```

```
  TYPE :: MODE
     DOUBLE PRECISION :: OMEGA
     COMPLEX*16       :: X,Y,Z
     DOUBLE PRECISION :: phi_x,phi_y,phi_z
     INTEGER          :: N_Floquet
     COMPLEX*16, DIMENSION(:,:), ALLOCATABLE :: V
     COMPLEX*16, DIMENSION(:),   ALLOCATABLE :: VALUES
     INTEGER,    DIMENSION(:),   ALLOCATABLE :: ROW,COLUMN
  END TYPE MODE

  TYPE :: ATOM
     INTEGER          :: id_system
     INTEGER          :: D_BARE
     DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: E_BARE
  END TYPE ATOM

  TYPE :: HARMONIC_FACTORS
     COMPLEX*16,DIMENSION(:,:), ALLOCATABLE :: U,U_r,U_AVG
     INTEGER,   DIMENSION(:),   ALLOCATABLE :: n
  END type HARMONIC_FACTORS

  TYPE :: MWCOUPLING
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: TOP
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: TOP_DAGGER
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: DC
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: DC_DAGGER
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: MW
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: MW_DAGGER
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: RF
     COMPLEX*16,DIMENSION(:,:),ALLOCATABLE :: RF_DAGGER
  END type MWCOUPLING
END MODULE TYPES
```

## 6 COMPUTATIONAL SUBROUTINES

The `ATOMIC_PROPERTIES`module defines the default physical parameters of

```
SUBROUTINE FLOQUETINIT(atomicspecie,manifold,JTOTAL,ID,info)
  ! ATOMICSPECIE: 87Rb,6Li,Cs,41K,qubit,lattice, SPIN
  ! MANIFOLD : "U" UPPER HYPERFINE MANIFOLD, "L" LOWER HYPERFIND MANIFOLD, "B" BO
  ! JTOTAL   :  IF ATOMICSPECIE .EQ. SPIN THEN JTOTAL IS THE TOTAL ANGULAR MOMENT
  !             IF ATOMICSPECIE .EQ. LATTICE, THEN JTOTAL IS THE NUMBER OF SITES


  ! calculate the dimenson of the Hilbert space
  ! initialize all the matrices required for a full Floquet calcuations
  ! Calculate the nuclear, electron and total angular momentum operators

  USE physical_constants ! Standard Module with constants
  USE ATOMIC_PROPERTIES  ! gF, F , etc. factors for several species
  USE subinterface       ! To ubroutines for representation of I and J operators
  USE ARRAYS
  !USE FLOQUET            ! Number of floquet modes
  USE SUBINTERFACE_LAPACK
  USE TYPES
  IMPLICIT NONE

  CHARACTER (LEN=*),OPTIONAL, INTENT(IN)    :: ATOMICSPECIE
  CHARACTER (LEN=*),OPTIONAL, INTENT(IN)    :: MANIFOLD  !
  !INTEGER,          OPTIONAL, INTENT(IN)    :: JTOTAL
  DOUBLE PRECISION, OPTIONAL, INTENT(IN)    :: JTOTAL
  TYPE(ATOM),       OPTIONAL, INTENT(OUT)   :: ID
  INTEGER,                    INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE SETHAMILTONIANCOMPONENTS(ID,NM,NF,MODES_NUM,FIELD,INFO)
  ! ID  tYPE OF ATOM
  ! MODES_NUM, VECTOR. THE SIZE OF THE VECTOR TELL US THE NUMBER OF FREQUENCIES,
  ! FIELDS : IN AND OUTPUT THE MATRICES
  ! INFO

  USE ARRAYS
  USE ATOMIC_PROPERTIES
  USE TYPES
  USE SUBINTERFACE_LAPACK ! write_matrix interface

  IMPLICIT NONE
  INTEGER,                   INTENT(IN)    :: NM,NF
  TYPE(ATOM),                INTENT(IN)    :: ID
  INTEGER,    DIMENSION(NM), INTENT(IN)    :: MODES_NUM
  TYPE(MODE), DIMENSION(NF), INTENT(INOUT) :: FIELD
  INTEGER,                   INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE F_representation(Fx,Fy,Fz,Ftotal)

  USE FUNCIONES

  IMPLICIT NONE
  DOUBLE PRECISION, DIMENSION(:,:), INTENT(OUT):: Fx,Fy,Fz
  DOUBLE PRECISION, INTENT(IN) :: Ftotal
  !INTEGER, INTENT(IN) :: Ftotal_

  !DOUBLE PRECISION
  INTEGER k,p,N_k
  double precision k_!,Ftotal

  Fx = 0.0
  Fy = 0.0
  Fz = 0.0
```

---

```
SUBROUTINE I_and_J_representations(j_x,j_y,j_z,I_x,I_y,I_z,L,S,I)

  USE FUNCIONES

  IMPLICIT  NONE
  DOUBLE PRECISION, DIMENSION(:,:),INTENT(INOUT) :: j_x,j_y,j_z,I_x,I_y,I_z
  DOUBLE PRECISION, INTENT(IN) :: L,S,I
```

---

```
SUBROUTINE MULTIMODETIMEEVOLUTINOPERATOR(D,NM,MODES_NUM,U_F_MODES,E_MULTIFLOQUET

  ! TIME EVOLUTION OPERATOR OF A MULTIMODE DRESSED SYSTEM. THE EVOLUTION OPERATOR
  ! MULTIMODE FLOQUET HAMILTONIAN
  ! U : MATRIX OF AMPLITUED OF PROBABILITIES FOR TRANSITIONS BETWEEN T1 TO T2
!!$  D              (IN)    : DIMENSION OF THE EXTENDED HILBERT SPACE (SIZE OF THE
!!$  NM             (IN)    : NUMBER OF MODES
!!$  MODES_NUM      (IN)    : VECTOR (NM) INDICATING THE NUMBER OF HARMONICS OF EA
!!$  U_F_MODES      (IN)    : TRANSFORMATION, DIMENSOON (D,D)
!!$  E_MULTIFLOQUET (IN)    : MULTIMODE FLOQUET SPECTRUM
!!$  D_BARE         (IN)    : DIMENSION OF THE BARE HILBERT SPACE
!!$  FIELD          (IN)    : STRUCTURE DESCRIBING THE COUPLINGS
!!$  T1             (IN)    : INITIAL TIME
!!$  T2             (IN)    : FINAL TIME
!!$  U              (OUT)   : TRANFORMATION BETWEEN THE EXTENDED BARE BASIS AND TH
!!$  INFO           (INOUT): (POSSIBLE) ERROR FLAG

  USE TYPES
  USE SUBINTERFACE_LAPACK


  IMPLICIT NONE
  INTEGER,                                    INTENT(IN)    :: D,D_BARE,NM ! DIM
  INTEGER,                                    INTENT(INOUT) :: INFO
  INTEGER,          DIMENSION(NM),            INTENT(IN)    :: MODES_NUM
  TYPE(MODE),       DIMENSION(NM),            INTENT(IN)    :: FIELD  ! FIELDS P
  DOUBLE PRECISION,                           INTENT(IN)    :: T1,T2  ! IN SECON
  DOUBLE PRECISION, DIMENSION(D),             INTENT(IN)    :: E_MULTIFLOQUET ! S
  COMPLEX*16,       DIMENSION(D,D),           INTENT(IN)    :: U_F_MODES   ! TRA
  COMPLEX*16,       DIMENSION(D_BARE,D_BARE), INTENT(OUT)   :: U           ! EVO
```

---

```
SUBROUTINE MULTIMODEFLOQUETMATRIX(ATOM_,NM,NF,MODES_NUM,FIELD,INFO)
  !ID,size(modes_num,1),total_frequencies,MODES_NUM,FIELDS,INFO
  !  USE FLOQUET
  !ATOM_ type atom, -> dimension of the bare Hilbert space
  !NM -> number of modes
  !NF -> Number of Fields
  !MODES_NUM -> number of harmonics of each mode
  !FIELD -> Field couplings
  !INFO


  USE ARRAYS
  USE ATOMIC_PROPERTIES
  USE TYPES
  USE SUBINTERFACE_LAPACK

  IMPLICIT NONE
  INTEGER,                  INTENT(IN)    :: NM,NF
  INTEGER,                  INTENT(INOUT) :: INFO
  INTEGER,   DIMENSION(NM), INTENT(IN)    :: MODES_NUM
  TYPE(MODE),DIMENSION(NF), INTENT(IN)    :: FIELD
  TYPE(ATOM),               INTENT(IN)    :: ATOM_
```

---

```
SUBROUTINE MULTIMODEFLOQUETMATRIX_SP(ATOM__,NM,NF,MODES_NUM,FIELDS,VALUES_,ROW_I

!ATOM_      (IN)    : type of quantum system
!NM         (IN)    : number of modes
!NF         (IN)    : number of driving fields
!MODES_NUM  (IN)    : vector indicating the number of harmonics of each driving
!FIELDS     (IN)    : Fields
!VALUES_    (OUT)   : Hamiltonian values
!ROW_INDEX_ (OUT)   : vector indicating the row position of values
```

```
!COLUMN_    (OUT)   : vector indicating the column position of the values
!INFO       (INOUT) : error flag. INFO=0 means there is no error

  USE TYPES         !(modes.f90)
  USE MERGINGARRAYS !(utils.f90)

  IMPLICIT NONE
  INTEGER                  ,            INTENT(IN)    :: NM,NF
  TYPE(MODE), DIMENSION(NF),            INTENT(INOUT) :: FIELDS
  TYPE(ATOM),                           INTENT(IN)    :: ATOM__
  INTEGER,    DIMENSION(NM),            INTENT(IN)    :: MODES_NUM
  INTEGER,                              INTENT(INOUT) :: INFO
  COMPLEX*16, DIMENSION(:), ALLOCATABLE,INTENT(OUT)   :: VALUES_
  INTEGER,    DIMENSION(:), ALLOCATABLE,INTENT(OUT)   :: COLUMN_
  INTEGER,    DIMENSION(:), ALLOCATABLE,INTENT(OUT)   :: ROW_INDEX_
```

---

```
SUBROUTINE MULTIMODEFLOQUETTRANSFORMATION(D,NM,MODES_NUM,U_F_MODES,E_MULTIFLOQUE

  ! TIME-DEPENDENT TRANSFORMATION BETWEEN THE EXTENDED BARE BASIS AND THE FLOQUE
  ! U(T1) = sum_ U^n exp(i n omega T1)
  !
!!$  D              (IN)   : DIMENSION OF THE EXTENDED HILBERT SPACE (SIZE OF TH
!!$  NM             (IN)   : NUMBER OF MODES
!!$  MODES_NUM      (IN)   : VECTOR (NM) INDICATING THE NUMBER OF HARMONICS OF E
!!$  U_F_MODES      (IN)   : TRANSFORMATION, DIMENSOON (D,D)
!!$  E_MULTIFLOQUET (IN)   : MULTIMODE FLOQUET SPECTRUM
!!$  D_BARE         (IN)   : DIMENSION OF THE BARE HILBERT SPACE
!!$  FIELD          (IN)   : STRUCTURE DESCRIBING THE COUPLINGS
!!$  T1             (IN)   : TIME. THE BARE 2 DRESSED TRANSFORMATINO IS TIME DEP
!!$  U              (OUT)  : TRANFORMATION BETWEEN THE EXTENDED BARE BASIS AND T
!!$  INFO           (INOUT): (POSSIBLE) ERROR FLAG

  USE TYPES

  IMPLICIT NONE
  INTEGER,                                    INTENT(IN)    :: D,D_BARE,NM ! DIM
  INTEGER,                                    INTENT(INOUT) :: INFO
  INTEGER,          DIMENSION(NM),            INTENT(IN)    :: MODES_NUM
  TYPE(MODE),       DIMENSION(NM),            INTENT(IN)    :: FIELD  ! FIELDS P
  DOUBLE PRECISION,                           INTENT(IN)    :: T1 ! IN SECONDS
  DOUBLE PRECISION, DIMENSION(D),             INTENT(IN)    :: E_MULTIFLOQUET ! 
  COMPLEX*16,       DIMENSION(D,D),           INTENT(IN)    :: U_F_MODES ! TRANF
  COMPLEX*16,       DIMENSION(D_BARE,D),      INTENT(OUT)   :: U ! TIME-DEPENDEN
```

```
SUBROUTINE MULTIMODEMICROMOTION(ID,D,NM,MODES_NUM,U_F_MODES,E_MULTIFLOQUET,D_BAR

  ! TIME-DEPENDENT TRANSFORMATION BETWEEN THE EXTENDED BARE BASIS AND THE FLOQUE
  ! U(T1) = sum_ U^n exp(i n omega T1)
  !
!!$  D              (IN)   : DIMENSION OF THE EXTENDED HILBERT SPACE (SIZE OF TH
!!$  NM             (IN)   : NUMBER OF MODES
!!$  MODES_NUM      (IN)   : VECTOR (NM) INDICATING THE NUMBER OF HARMONICS OF E
!!$  U_F_MODES      (IN)   : TRANSFORMATION, DIMENSOON (D,D)
!!$  E_MULTIFLOQUET (IN)   : MULTIMODE FLOQUET SPECTRUM
!!$  D_BARE         (IN)   : DIMENSION OF THE BARE HILBERT SPACE
!!$  FIELD          (IN)   : STRUCTURE DESCRIBING THE COUPLINGS
!!$  T1             (IN)   : TIME. THE BARE 2 DRESSED TRANSFORMATINO IS TIME DEP
!!$  U              (OUT)  : TRANFORMATION BETWEEN THE EXTENDED BARE BASIS AND T
!!$  INFO           (INOUT): (POSSIBLE) ERROR FLAG

  !USE TYPES_C
  USE TYPES
  !USE MODES_4F
  USE SUBINTERFACE_LAPACK
  USE ATOMIC_PROPERTIES

  IMPLICIT NONE
  TYPE(ATOM),                  INTENT(IN)    :: ID
  INTEGER,                                   INTENT(IN)    :: D,D_BARE,NM ! DIM
  INTEGER,                                   INTENT(INOUT) :: INFO
  INTEGER,          DIMENSION(NM),           INTENT(IN)    :: MODES_NUM
  TYPE(MODE),       DIMENSION(NM),           INTENT(IN)    :: FIELD  ! FIELDS P
  DOUBLE PRECISION,                          INTENT(IN)    :: T1 ! IN SECONDS
  DOUBLE PRECISION, DIMENSION(D),            INTENT(IN)    :: E_MULTIFLOQUET ! 
  COMPLEX*16,       DIMENSION(D,D),          INTENT(IN)    :: U_F_MODES ! TRANF
  COMPLEX*16,       DIMENSION(D_BARE,D_BARE), INTENT(OUT)   :: U ! TIME-DEPENDEN
```

---

```
SUBROUTINE MICROMOTIONFOURIERDRESSEDBASIS(ID,DRESSINGFIELDS_INDICES,MODES_NUM,FI
! ID        (in)    :: TYPE(ATOM) system ID
! DRESSINGFIELDS_INDICES (in) :: integer array indicating the indices of the dre
! MODES_NUM (in)    :: integer array indicating the number of harmonics of all d
! FIELDS    (in)    :: Array of TYPE(MODE) of dimension
! U_FD      (out)   :: complex*16 matrix fourier decomposition of the micromotio
! E_DRESSED (out)   :: dressed energies
! INFO      (inout) :: error flag
```

```
  USE TYPES

  TYPE(ATOM),                      INTENT(IN)  :: ID
  INTEGER,    DIMENSION(:),        INTENT(IN)  :: DRESSINGFIELDS_INDICES
  INTEGER,    DIMENSION(:),        INTENT(IN)  :: MODES_NUM
  TYPE(MODE), DIMENSION(:),        INTENT(IN)  :: FIELDS
  COMPLEX*16, DIMENSION(:,:),      ALLOCATABLE, INTENT(OUT) :: U_FD
  DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE, INTENT(OUT) :: E_DRESSED
```

---

```
SUBROUTINE MICROMOTIONDRESSEDBASIS(ID,MODES_NUM,DRESSINGFIELDS_INDICES,FIELDS,U_

! ID (in)         :: TYPE(ATOM) system ID
! MODES_NUM (in) :: integer array indicating the number of harmonics of each dri
! DRESSINFIELDS_INDICES :: integer array indicating the indices of the dressing 
! FIELDS         :: Array of TYPE(MODES) with NM components (all driving fields)
! U_F_MODES      :: complex*16 matrix of dimension DxD. Fourier decomposition of
! E_MULTIFLOQUET :: dressed energies
! T1             :: double precision, time
! U              :: complex*16 matrix of dimension D_BARE x D_BARE. micromotion 
! INFO           :: error flag


  USE TYPES
  IMPLICIT NONE
  TYPE(ATOM),                        INTENT(IN)    :: ID
  INTEGER,          DIMENSION(:),   INTENT(IN)    :: MODES_NUM
  INTEGER,          DIMENSION(:),   INTENT(IN)    :: DRESSINGFIELDS_INDICES
  COMPLEX*16,       DIMENSION(:,:), INTENT(IN)    :: U_F_MODES
  DOUBLE PRECISION, DIMENSION(:),   INTENT(IN)    :: E_MULTIFLOQUET
  TYPE(MODE),       DIMENSION(:),   INTENT(IN)    :: FIELDS
  DOUBLE PRECISION ,                 INTENT(IN)    :: T1
  COMPLEX*16,       DIMENSION(:,:), INTENT(OUT)   :: U
  INTEGER,                           INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE MULTIMODETRANSITIONAVG(D,NM,FIELD,MODES_NUM,U_F_MODES,E_MULTIFLOQUET,
!!$   AVERAGE TIME EVOLUTION OPERATOR OF A MULTIMODE DRESSED SYSTEM. THE AVERAGE
!!$   MULTIMODE FLOQUET HAMILTONIAN
!!$   U : MATRIX OF AVERAGE TRANSITION PROBABILITIES
!!$
!!$  D              (IN)   : DIMENSION OF THE EXTENDED HILBERT SPACE (SIZE OF TH
!!$  NM             (IN)   : NUMBER OF MODES
!!$  MODES_NUM      (IN)   : VECTOR (NM) INDICATING THE NUMBER OF HARMONICS OF EA
!!$  U_F_MODES      (IN)   : TRANSFORMATION, DIMENSOON (D,D)
!!$  E_MULTIFLOQUET (IN)   : MULTIMODE FLOQUET SPECTRUM
!!$  D_BARE         (IN)   : DIMENSION OF THE BARE HILBERT SPACE
!!$  U              (OUT)  :  MATRIX OF AVERAGE TRANSITION PROBABILITIES
!!$  INFO           (INOUT): (POSSIBLE) ERROR FLAG

  USE TYPES

  IMPLICIT NONE
  TYPE(MODE),DIMENSION(NM), INTENT(IN)     :: FIELD
  INTEGER,   DIMENSION(NM), INTENT(IN)     :: MODES_NUM

  INTEGER,                                    INTENT(IN)    :: D,D_BARE,NM ! DIME
  INTEGER,                                    INTENT(INOUT) :: INFO
  DOUBLE PRECISION, DIMENSION(D),             INTENT(IN)    :: E_MULTIFLOQUET ! S
  COMPLEX*16,       DIMENSION(D,D),           INTENT(IN)    :: U_F_MODES    ! TRAN
  DOUBLE PRECISION, DIMENSION(D_BARE,D_BARE), INTENT(OUT)   :: U            ! EVOL
```

---

## 7 DRIVER SUBROUTINES

```
SUBROUTINE DRESSEDBASIS(D,ID,NM,MODES_NUM,FIELDS,U_FD,E_DRESSED,INFO)

!!$ THIS SUBROUTINES CALCULATES THE FOURIER COMPONENTS OF THE
!!$ TRANSFORMATION BETWEEN THE BARE BASIS TO THE DRESSED BASIS DEFINDED BY THE FU
!!$
!!$ D                             : DIMENSION OF THE MULTIMODE EXTENDED HILBERT SP
!!$ ID (IN)                       : TYPE OF QUANTUM SYSTEM
!!$ NM (IN)                       : NUMBER OF MODES == NUMBER OF DRIVING FIELDS
!!$ MODES_NUM                     : VECTOR INDICATING THE NUMBER OF HARMONICS OF E
!!$ FIELDS (IN)                   : AMPLITUDE, FREQUENCY AND PHASES OF ALL DRIVING
!!$ U_FD (OUT)                    : THIS IS THE TRANSFORMATION WE ARE LOOKING FOR
!!$ E_DRESSED (OUT)               : DRESSED ENERGIES
!!$ INFO (INOUT)                  : INFO = 0 MEANS SUCESS


  USE ATOMIC_PROPERTIES
  USE TYPES
  USE SUBINTERFACE
  USE SUBINTERFACE_LAPACK
  USE FLOQUETINIT_
  USE ARRAYS
  IMPLICIT NONE
  TYPE(MODE), DIMENSION(NM),     INTENT(IN)    :: FIELDS
  TYPE(ATOM),                    INTENT(IN)    :: ID
  INTEGER,    DIMENSION(NM),     INTENT(IN)    :: MODES_NUM
  COMPLEX*16, DIMENSION(D,D),       INTENT(OUT)   :: U_FD
  DOUBLE PRECISION, DIMENSION(D), INTENT(OUT)   :: E_DRESSED
  INTEGER,                       INTENT(IN)    :: NM,D
  INTEGER,                       INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE DRESSEDBASIS_SP(D,ID,NM,MODES_NUM,FIELDS,U_FD,E_DRESSED,INFO)

!!$THIS SUBROUTINES CALCULATES THE TRANSFORMATION BETWEEN THE BARE BASIS TO THE I
!!$ D                               : DIMENSION OF THE MULTIMODE EXTENDED HILBERT SI
!!$ ID (IN)                         : TYPE OF QUATUM SYSTEM
!!$ NM (IN)                         : NUMBER OF MODES == NUMBER OF DRIVING FIELDS
!!$ MODES_NUM                       : VECTOR INDICATING THE NUMBER OF HARMONICS OF I
!!$ FIELDS (IN)                     : AMPLITUDE, FREQUENCY AND PHASES OF ALL DRIVING
!!$ U_FD (OUT)                      : THIS IS THE TRANSFORMATION WE ARE LOOKING FOR
!!$ E_DRESSED (OUT)                 : DRESSED ENERGIES
!!$ INFO (INOUT)                    : INFO = 0 MEANS SUCESS


  USE ATOMIC_PROPERTIES
  USE TYPES
  USE SPARSE_INTERFACE
  USE SUBINTERFACE
  USE SUBINTERFACE_LAPACK
  USE FLOQUETINIT_
  USE ARRAYS

  IMPLICIT NONE
  TYPE(MODE), DIMENSION(NM),      INTENT(INOUT)    :: FIELDS
  TYPE(ATOM),                     INTENT(IN)    :: ID
  INTEGER,    DIMENSION(NM),      INTENT(IN)    :: MODES_NUM
  COMPLEX*16, DIMENSION(D,D),     INTENT(OUT)   :: U_FD
  DOUBLE PRECISION, DIMENSION(D), INTENT(OUT)   :: E_DRESSED
  INTEGER,                        INTENT(IN)    :: NM,D
  INTEGER,                        INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE TIMEEVOLUTIONOPERATOR(ID,D_BARE,NM,MODES_NUM,FIELD,T1,T2,U,INFO)
 ! TIME EVOLUTION OPERATOR OF A MULTIMODE DRESSED SYSTEM. THE EVOLUTION OPERATOR
```

```
  ! MULTIMODE FLOQUET HAMILTONIAN
  ! U : MATRIX OF AMPLITUED OF PROBABILITIES FOR TRANSITIONS BETWEEN T1 TO T2
!!$  NM             (IN)   : NUMBER OF MODES
!!$  MODES_NUM      (IN)   : VECTOR (NM) INDICATING THE NUMBER OF HARMONICS OF EA
!!$  D_BARE         (IN)   : DIMENSION OF THE BARE HILBERT SPACE
!!$  FIELD          (IN)   : STRUCTURE DESCRIBING THE COUPLINGS
!!$  T1             (IN)   : INITIAL TIME
!!$  T2             (IN)   : FINAL TIME
!!$  U              (OUT)  : TRANFORMATION BETWEEN THE EXTENDED BARE BASIS AND TH
!!$  INFO           (INOUT): (POSSIBLE) ERROR FLAG

    USE ATOMIC_PROPERTIES
    USE TYPES
    USE SUBINTERFACE
    USE SUBINTERFACE_LAPACK
    USE FLOQUETINIT_
    USE ARRAYS


    IMPLICIT NONE
    TYPE(ATOM) ,                                INTENT(IN)    :: ID
    INTEGER,                                    INTENT(IN)    :: D_BARE
    INTEGER,                                    INTENT(IN)    :: NM
    INTEGER,          DIMENSION(NM),            INTENT(IN)    :: MODES_NUM
    TYPE(MODE),       DIMENSION(NM),            INTENT(IN)    :: FIELD  ! FIELDS
    DOUBLE PRECISION,                           INTENT(IN)    :: T1
    DOUBLE PRECISION,                           INTENT(IN)    :: T2
    COMPLEX*16,       DIMENSION(D_BARE,D_BARE), INTENT(OUT)   :: U
    INTEGER,                                    INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE MICROMOTIONFOURIERDRESSEDBASIS(ID,DRESSINGFIELDS_INDICES,MODES_NUM,FI
! THIS SUBROUTINE CALCULATES THE FOURIER COMPONENTS (U_FD) AND PHASES (E_DRESSED)
! ID        (in)    :: TYPE(ATOM) system ID
! DRESSINGFIELDS_INDICES (in) :: integer array indicating the indices of the dres
! MODES_NUM (in)    :: integer array indicating the number of harmonics of all dr
! FIELDS    (in)    :: Array of TYPE(MODE) of dimension
! U_FD      (out)   :: complex*16 matrix fourier decomposition of the micromotion
! E_DRESSED (out)   :: dressed energies
! INFO      (inout) :: error flag

  USE TYPES
  IMPLICIT NONE
  TYPE(ATOM),                     INTENT(IN)  :: ID
  INTEGER,    DIMENSION(:),       INTENT(IN)  :: DRESSINGFIELDS_INDICES
  INTEGER,    DIMENSION(:),       INTENT(IN)  :: MODES_NUM
  TYPE(MODE), DIMENSION(:),       INTENT(IN)  :: FIELDS
  COMPLEX*16, DIMENSION(:,:),     ALLOCATABLE, INTENT(OUT) :: U_FD
  DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE, INTENT(OUT) :: E_DRESSED
  INTEGER, INTENT(INOUT) :: INFO


END SUBROUTINE MICROMOTIONFOURIERDRESSEDBASIS
```

---

```
SUBROUTINE MICROMOTIONDRESSEDBASIS(ID,MODES_NUM,DRESSINGFIELDS_INDICES,FIELDS,U_F
! THIS SUBROUTINE CALCULATES U: THE TIME-DEPENDENT MICROMOTION OPERATOR OF A SUBS

! ID (in)        :: TYPE(ATOM) system ID
! MODES_NUM (in) :: integer array indicating the number of harmonics of each driv
! DRESSINFIELDS_INDICES :: integer array indicating the indices of the dressing m
! FIELDS         :: Array of TYPE(MODES) with NM components (all driving fields)
! U_F_MODES      :: complex*16 matrix of dimension DxD. Fourier decomposition of
! E_MULTIFLOQUET :: dressed energies
! T1             :: double precision, time
! U              :: complex*16 matrix of dimension D_BARE x D_BARE. micromotion o
! INFO           :: error flag


  USE TYPES
  IMPLICIT NONE
  TYPE(ATOM),                         INTENT(IN)    :: ID
  INTEGER,          DIMENSION(:),     INTENT(IN)    :: MODES_NUM
  INTEGER,          DIMENSION(:),     INTENT(IN)    :: DRESSINGFIELDS_INDICES
  COMPLEX*16,       DIMENSION(:,:),   INTENT(IN)    :: U_F_MODES
  DOUBLE PRECISION, DIMENSION(:),     INTENT(IN)    :: E_MULTIFLOQUET
  TYPE(MODE),       DIMENSION(:),     INTENT(IN)    :: FIELDS
  DOUBLE PRECISION ,                  INTENT(IN)    :: T1
  COMPLEX*16,       DIMENSION(:,:),   INTENT(OUT)   :: U
  INTEGER,                            INTENT(INOUT) :: INFO
```

## 7.1 Utility subroutines

```
SUBROUTINE PACKINGBANDMATRIX(N,A,KD,AB,INFO)

! brute force packing of a banded matrix

  IMPLICIT NONE
  INTEGER, INTENT(INOUT) :: INFO
  INTEGER, INTENT(IN)    :: N,KD
  COMPLEX*16, DIMENSION(N,N)    :: A
  COMPLEX*16, DIMENSION(KD+1,N) :: AB
```

---

```
SUBROUTINE LAPACK_FULLEIGENVALUES(H,N,W_SPACE,INFO)
!eigenvalues/vectors of matrix ab
!H, inout, packed banded matrix
! , out,eigenvectors
!N, in,matrix dimension
!W_space, out, eigenvalues
!INFO,inout, error flag

  !H is COMPLEX*16 array, dimension (N, N)
  !  69 *>          On entry, the Hermitian matrix A.  If UPLO = 'U', the
  !  70 *>          leading N-by-N upper triangular part of A contains the
  !  71 *>          upper triangular part of the matrix A.  If UPLO = 'L',
  !  72 *>          the leading N-by-N lower triangular part of A contains
  !  73 *>          the lower triangular part of the matrix A.
  !  74 *>          On exit, if JOBZ = 'V', then if INFO = 0, A contains the
  !  75 *>          orthonormal eigenvectors of the matrix A.
  !  76 *>          If JOBZ = 'N', then on exit the lower triangle (if UPLO='L')
  !  77 *>          or the upper triangle (if UPLO='U') of A, including the
  !  78 *>          diagonal, is destroyed.
  !
  ! The eigenvector H(:,r) corresponds to the eigenvalue W_SPACE(r)
  !
  IMPLICIT NONE
  INTEGER,                          INTENT(IN)    :: N
  COMPLEX*16,       DIMENSION(N,N), INTENT(INOUT) :: H
  DOUBLE PRECISION, DIMENSION(N),   INTENT(INOUT) :: W_SPACE
  INTEGER,                          INTENT(OUT)   :: INFO

SUBROUTINE LAPACK_FULLEIGENVALUESBAND(AB,Z,KD,N,W,INFO)
!eigenvalues/vectors of banded matrix ab
!AB, inout, packed banded matrix
!Z, out,eigenvectors
!KD out, calcuated eigenvectors
!N, in,matrix dimension
!W, out, eigenvalues
!INFO,inout, error flag

  !H is COMPLEX*16 array, dimension (N, N)
  !  69 *>          On entry, the Hermitian matrix A.  If UPLO = 'U', the
  !  70 *>          leading N-by-N upper triangular part of A contains the
  !  71 *>          upper triangular part of the matrix A.  If UPLO = 'L',
  !  72 *>          the leading N-by-N lower triangular part of A contains
  !  73 *>          the lower triangular part of the matrix A.
  !  74 *>          On exit, if JOBZ = 'V', then if INFO = 0, A contains the
  !  75 *>          orthonormal eigenvectors of the matrix A.
  !  76 *>          If JOBZ = 'N', then on exit the lower triangle (if UPLO='L')
  !  77 *>          or the upper triangle (if UPLO='U') of A, including the
  !  78 *>          diagonal, is destroyed.
  !
  ! The eigenvector H(:,r) corresponds to the eigenvalue W_SPACE(r)
  !
  IMPLICIT NONE
  INTEGER,                                INTENT(IN)    :: N,KD
  COMPLEX*16,       DIMENSION(KD+1,N), INTENT(INOUT)    :: AB
  COMPLEX*16,       DIMENSION(N,N),       INTENT(INOUT) :: Z
  DOUBLE PRECISION, DIMENSION(N),         INTENT(INOUT) :: W
  INTEGER,                                INTENT(OUT)   :: INFO
```

---

```
SUBROUTINE LAPACK_SELECTEIGENVALUES(H,N,W_SPACE,L1,L2,Z,INFO)
!selected eigenvalues/vectors of hermitian matrix
!H, inout, packed banded matrix
! , out,eigenvectors
!N, in,matrix dimension
!W_space, out, eigenvalues
!L1 ordinal lowest eigenvalue
!L2 ordinal highest eigenvlaue
!Z : eigenvectors
!INFO,inout, error flag

  !USE FLOQUET
  IMPLICIT NONE
  INTEGER,                          INTENT(IN)    :: N,L1,L2
  COMPLEX*16, DIMENSION(:,:),       INTENT(INOUT) :: H
  COMPLEX*16, DIMENSION(:,:),       INTENT(OUT)   :: Z
  DOUBLE PRECISION, DIMENSION(:), INTENT(OUT)     :: W_SPACE
  INTEGER,                          INTENT(OUT)   :: INFO
```

---

```
SUBROUTINE MKLSPARSE_FULLEIGENVALUES(D,DV,VALUES,ROW_INDEX,COLUMN,E_L,E_R,E_FLOQU

!CALCULATES THE ENERGY SPECTRUM OF THE MATRIX REPRESENTED BY VALUES, ROW_INDEX AN
! D (IN), MATRIX DIMENSION == NUMBER OF EIGENVALUES
! DV (IN), NUMBER OF VALUES != 0
! VALUES (IN) ARRAY OF VALUES
! ROW_INDEX (IN), ARRAY OF INDICES
! COLUMN (IN),    ARRAY OF COLUMN NUMBERS
! E_L (IN),       LEFT BOUNDARY OF THE SEARCH INTERVAL
! E_R (IN),       RIGHT BOUNDARY OF THE SEARCH INTERVAL
! E_FLOQUET (OUT), ARRAY OF EIGENVALUES
! INFO     (INOUT)  ERROR FLAG and VERBOSITY FLAG
!                 0 display no information
!                 1 DISPLAY INFORMAITON ABOUT THE SIZE OF THE ARRAYS
!                 10 DISPLAY INFORMAITON ABOUT THE ARRAYS AND THE ARRAYS
  USE FEAST
  IMPLICIT NONE
  INTEGER,                          INTENT(IN)    :: D,DV
  COMPLEX*16,       DIMENSION(DV),  INTENT(INOUT) :: VALUES
  INTEGER,          DIMENSION(DV),  INTENT(INOUT) :: COLUMN
  INTEGER,          DIMENSION(D+1), INTENT(INOUT) :: ROW_INDEX
  DOUBLE PRECISION,                 INTENT(IN)    :: E_L,E_R
  DOUBLE PRECISION, DIMENSION(D),   INTENT(OUT)   :: E_FLOQUET
  COMPLEX*16,       DIMENSION(D,D), INTENT(OUT)   :: U_F
  INTEGER,                          INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE QUICK_SORT_I_T(v,index_t,N)

  IMPLICIT NONE
  INTEGER, INTENT(IN) :: N

  !INTEGER, DIMENSION(N),INTENT(INOUT) :: v
  DOUBLE PRECISION, DIMENSION(N),INTENT(INOUT) :: v
  INTEGER, DIMENSION(N),INTENT(INOUT) :: index_t

  INTEGER, PARAMETER :: NN=2500, NSTACK=500
```

---

```
SUBROUTINE TESTUNITARITY(N,U,DELTA,INFO)
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: N
  COMPLEX*16, DIMENSION(N,N), INTENT(IN) :: U_F
  INTEGER, INTENT(INOUT) :: INFO
  DOUBLE PRECISION, INTENT(OUT) :: DELTA
```

---

```
SUBROUTINE WRITE_MATRIX(A)
! it writes a matrix of doubles nxm on the screen
  DOUBLE PRECISION, DIMENSION(:,:) :: A
  CHARACTER(LEN=105) STRING
  CHARACTER(LEN=105) aux_char
  integer :: aux
```

---

```
SUBROUTINE WRITE_MATRIX_INT(A)
!it writes a matrix of integer nxm on the screen
  INTEGER, DIMENSION(:,:) :: A
```

---

```
SUBROUTINE COORDINATEPACKING(D,A,V,R,C,index,INFO)
  IMPLICIT NONE
  INTEGER,INTENT(IN):: D
  COMPLEX*16,DIMENSION(D,D),INTENT(IN)  :: A
  COMPLEX*16,DIMENSION(D*D),INTENT(OUT) :: V
  INTEGER, DIMENSION(D*D),  INTENT(OUT) :: R,C
  INTEGER, INTENT(OUT)   :: index
  INTEGER, INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE APPENDARRAYS(V,B,INFO)
  COMPLEX*16, DIMENSION(:),ALLOCATABLE, INTENT(INOUT) :: V
  COMPLEX*16, DIMENSION(:),INTENT(IN)    :: B
  INTEGER,                 INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE APPENDARRAYSI(V,B,INFO)
  INTEGER, DIMENSION(:),ALLOCATABLE, INTENT(INOUT) :: V
  INTEGER, DIMENSION(:),INTENT(IN)    :: B
  INTEGER,                 INTENT(INOUT) :: INFO
```

---

```
SUBROUTINE VARCRCPACKING(N,DIM,UPLO,zero,A,VALUES,COLUMNS,ROWINDEX,INFO)

  INTEGER,                    INTENT(IN)    :: N
  INTEGER,                    INTENT(INOUT) :: INFO,DIM
  CHARACTER,                  INTENT(IN)    :: UPLO
  DOUBLE PRECISION,           INTENT(IN)    :: ZERO
  COMPLEX*16,DIMENSION(N,N), INTENT(IN)    :: A

  COMPLEX*16, DIMENSION(DIM), INTENT(OUT) :: VALUES
  INTEGER,    DIMENSION(DIM), INTENT(OUT) :: COLUMNS
  INTEGER,    DIMENSION(N+1), INTENT(OUT) :: ROWINDEX
```

## 8 Convention C++ wrappers

a challenge subroutines with ... and allocatable arrays. This is overcome by denifning global variables ...