QE main modules

$QEROOT

```
degironc@degironc-Laptop:~/QE/espresso-5.0.2$ ls
archive  configure             flib        License   PlotPhon  README
atomic   CPV                   include     Makefile  PP        S3DE
bin      dev-tools             install     make.sys  pseudo    TODO
BLAS     Doc                   iotk        Modules   PW        upftools
clib     environment_variables lapack-3.2  PHonon    QHA
```

$QEROOT

```
degironc@degironc-Laptop:~/QE/espresso-5.0.2$ ls
archive   configure             flib        License   PlotPhon   README
atomic    CPV                   include     Makefile  PP         S3DE
bin       dev-tools             install     make.sys  pseudo     TODO
BLAS      Doc                   iotk        Modules   PW         upftools
clib      environment_variables lapack-3.2  PHonon    QHA
```

$QEROOT/Modules

```
degironc@degironc-Laptop:~/QE/espresso-5.0.2/Modules$ ls *.f90
atom.f90                    input_parameters.f90          read_upf_v1.f90
autopilot.f90               io_files.f90                  read_upf_v2.f90
basic_algebra_routines.f90  io_global.f90                 read_uspp.f90
becmod.f90                  ions_base.f90                 read_xml_cards.f90
bfgs_module.f90             kernel_table.f90              read_xml.f90
cell_base.f90               kind.f90                      read_xml_fields.f90
check_stop.f90              mm_dispersion.f90             recvec.f90
clocks.f90                  mp_base.f90                   recvec_subs.f90
compute_dipole.f90          mp.f90                        run_info.f90
constants.f90               mp_global.f90                 set_signal.f90
constraints_module.f90      mp_image_global_module.f90    sic.f90
control_flags.f90           mp_wave.f90                   splinelib.f90
coulomb_vcut.f90            noncol.f90                    stick_base.f90
descriptors.f90             open_close_input_file.f90     stick_set.f90
dspev_drv.f90               parallel_include.f90          timestep.f90
electrons_base.f90          parameters.f90                upf.f90
environment.f90             parser.f90                    upf_to_internal.f90
error_handler.f90           paw_variables.f90             uspp.f90
fd_gradient.f90             plugin_arguments.f90          version.f90
fft_base.f90                plugin_flags.f90              wannier_new.f90
fft_custom.f90              pseudo_types.f90              wave_base.f90
fft_interfaces.f90          ptoolkit.f90                  wavefunctions.f90
fft_parallel.f90            radial_grids.f90              wrappers.f90
fft_scalar.f90              random_numbers.f90            write_upf_v2.f90
fft_types.f90               read_cards.f90                ws_base.f90
funct.f90                   read_input.f90                xc_vdW_DF.f90
generate_function.f90       read_namelists.f90            xml_input.f90
griddim.f90                 read_ncpp.f90                 xml_io_base.f90
image_io_routines.f90       read_pseudo.f90               zhpev_drv.f90
```

$QEROOT/Modules

```
degironc@degironc-Laptop:~/QE/espresso-5.0.2/Modules$ ls *.f90
atom.f90                      input_parameters.f90          read_upf_v1.f90
autopilot.f90                 io_files.f90                  read_upf_v2.f90
basic_algebra_routines.f90    io_global.f90                 read_uspp.f90
becmod.f90                    ions_base.f90                 read_xml_cards.f90
bfgs_module.f90               kernel_table.f90              read_xml.f90
cell_base.f90                 kind.f90                      read_xml_fields.f90
check_stop.f90                mm_dispersion.f90             recvec.f90
clocks.f90                    mp_base.f90                   recvec_subs.f90
compute_dipole.f90            mp.f90                        run_info.f90
constants.f90                 mp_global.f90                 set_signal.f90
constraints_module.f90        mp_image_global_module.f90    sic.f90
control_flags.f90             mp_wave.f90                   splinelib.f90
coulomb_vcut.f90              noncol.f90                    stick_base.f90
descriptors.f90               open_close_input_file.f90     stick_set.f90
dspev_drv.f90                 parallel_include.f90          timestep.f90
electrons_base.f90            parameters.f90                upf.f90
environment.f90               parser.f90                    upf_to_internal.f90
error_handler.f90             paw_variables.f90             uspp.f90
fd_gradient.f90               plugin_arguments.f90          version.f90
fft_base.f90                  plugin_flags.f90              wannier_new.f90
fft_custom.f90                pseudo_types.f90              wave_base.f90
fft_interfaces.f90            ptoolkit.f90                  wavefunctions.f90
fft_parallel.f90              radial_grids.f90              wrappers.f90
fft_scalar.f90                random_numbers.f90            write_upf_v2.f90
fft_types.f90                 read_cards.f90                ws_base.f90
funct.f90                     read_input.f90                xc_vdW_DF.f90
generate_function.f90         read_namelists.f90            xml_input.f90
griddim.f90                   read_ncpp.f90                 xml_io_base.f90
image_io_routines.f90         read_pseudo.f90               zhpev_drv.f90
```

constants.f90 *contains values of physical (HARTREE_SI, BOHR_RADIUS_SI) and mathematical constants (pi,tpi,..)*

parameters.f90 *max dimension of some qts*

```
MODULE parameters

  IMPLICIT NONE
  SAVE

  INTEGER, PARAMETER :: &
       ntypx  = 10,     &! max number of different types of atom
       npsx   = ntypx,  &! max number of different PPs (obsolete)
       nsx    = ntypx,  &! max number of atomic species (CP)
       npk    = 40000,  &! max number of k-points
       lmaxx  = 3,      &! max non local angular momentum (l=0 to lmaxx)
       lqmax= 2*lmaxx+1  ! max number of angular momenta of Q

END MODULE parameters
```

input_parameters.f90	*contains input vars*
read_input.f90	*reads input data by*
read_namelists.f90	*… reading namelists*
read_cards.f90	*… reading cards*

These are common input routines, then CP and Pwscf each have their own input.f90 routine that moves input data in their internal location.

fft_base.f90 *FFT base module*

fft_custom.f90 *FFT w cutomized cutoff*

fft_interfaces.f90 *PUBLIC :: fwfft, invfft*

fwfft (grid_type, fft_data,fft_dlay_descriptor)

invfft(grid_type, fft_data,fft_dlay_descriptor)

fft_parallel.f90

fft_scalar.f90

fft_types.f90 *TYPE fft_dlay_descriptor*

grid_type

```
! grid_type = 'Dense'
!   forward fourier transform of potentials and charge density
!   on the dense grid . On output, f is overwritten
! grid_type = 'Smooth'
!   forward fourier transform of potentials and charge density
!   on the smooth grid . On output, f is overwritten
! grid_type = 'Wave'
!   forward fourier transform of  wave functions
!   on the smooth grid . On output, f is overwritten
! grid_type = 'Custom'
!   forward fourier transform of potentials and charge density
!   on a custom defined grid specified by dfft. On output, f
!   is overwritten
! grid_type = 'CustomWave'
!   forward fourier transform of  wave functions
!   on a custom defined grid specified by dfft. On output, f
!   is overwritten
```

fft_dlay_descriptor

```
TYPE fft_dlay_descriptor
  INTEGER :: nst      ! total number of sticks
  INTEGER, POINTER :: nsp(:)   ! number of sticks per processor ( potential )
                               ! using proc index starting from 1 !!
                               ! on proc mpime -> nsp( mpime + 1 )
  INTEGER, POINTER :: nsw(:)   ! number of sticks per processor ( wave func )
                               ! using proc index as above
  INTEGER :: nr1    = 0  !
  INTEGER :: nr2    = 0  ! effective FFT dimensions of the 3D grid (global)
  INTEGER :: nr3    = 0  !
  INTEGER :: nr1x   = 0  ! FFT grids leading dimensions
  INTEGER :: nr2x   = 0  ! dimensions of the arrays for the 3D grid (global)
  INTEGER :: nr3x   = 0  ! may differ from nr1 ,nr2 ,nr3 in order to boost performances
  INTEGER :: npl    = 0  ! number of "Z" planes for this processor = npp( mpime + 1 )
  INTEGER :: nnp    = 0  ! number of 0 and non 0 sticks in a plane ( ~nr1*nr2/nproc )
  INTEGER :: nnr    = 0  ! local number of FFT grid elements  ( ~nr1*nr2*nr3/proc )
                         ! size of the arrays allocated for the FFT, local to each processor:
                         ! in parallel execution may differ from nr1x*nr2x*nr3x
                         ! Not to be confused either with nr1*nr2*nr3
```

… and much more

cell_base.f90	*system defs*
recvec.f90	*reciprocal lattice defs*
recvec_subs.f90	*reciprocal lattice subs*
	ggen
wavefunctions.f90	*wfc module* *evc(npw,nbndx)* *psic(nrxx)*

MPI – Message Passing Interface

data are distributed - communication is explicit

definition of processor subgroups

synchronization

broadcast

collect

all_to_all

MPI modules in Modules

mp_base.f90	*basic MPI calls*
mp.f90	*main MPI interfaces*
mp_global.f90	*MPI groups defs*

mp_image_global_module.f90

mp_wave.f90

mp.f90

main MPI interfaces

```
PUBLIC :: mp_start, mp_end, &
  mp_bcast, mp_sum, mp_max, mp_min, mp_rank, mp_size, &
  mp_gather, mp_get, mp_put, mp_barrier, mp_report, mp_group_free, &
  mp_root_sum, mp_comm_free, mp_comm_create, mp_comm_group, &
  mp_group_create, mp_comm_split, mp_set_displs
```

MPI parallelization layers

R & G space data distribution

K-points distribution (pools)

Task group parallelization (many procs)

Band group parallelization (mainly CP and EXX)

Image parallelization (NEB only)

mp_global.f90

World group (all processors)

world_comm

Image groups (processors within an image)

Pot groups (processors within a cooking-pot)

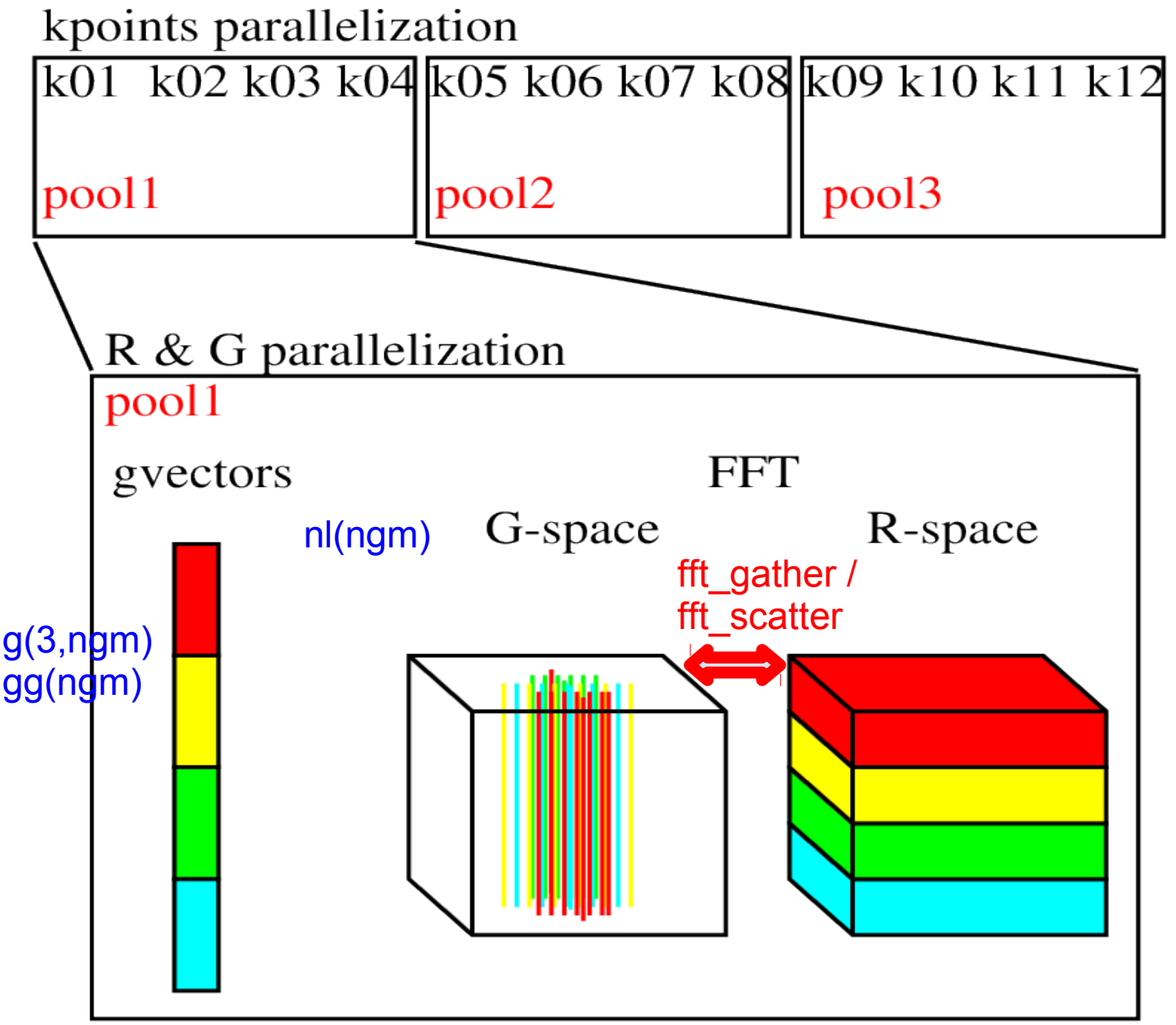
Pool groups (processors within a pool of k-points)

inter_pool_comm, intra_pool_comm

Band groups (processors within a pool of bands)

inter_bgrp_comm, intra_bgrp_comm

Ortho (or linear-algebra) groups


kpoints parallelization
k01 k02 k03 k04
k05 k06 k07 k08
k09 k10 k11 k12
pool1
pool2
pool3
R & G parallelization
pool1
gvectors
FFT
nl(ngm)
G-space
R-space
fft_gather /
fft_scatter
g(3,ngm)
gg(ngm)

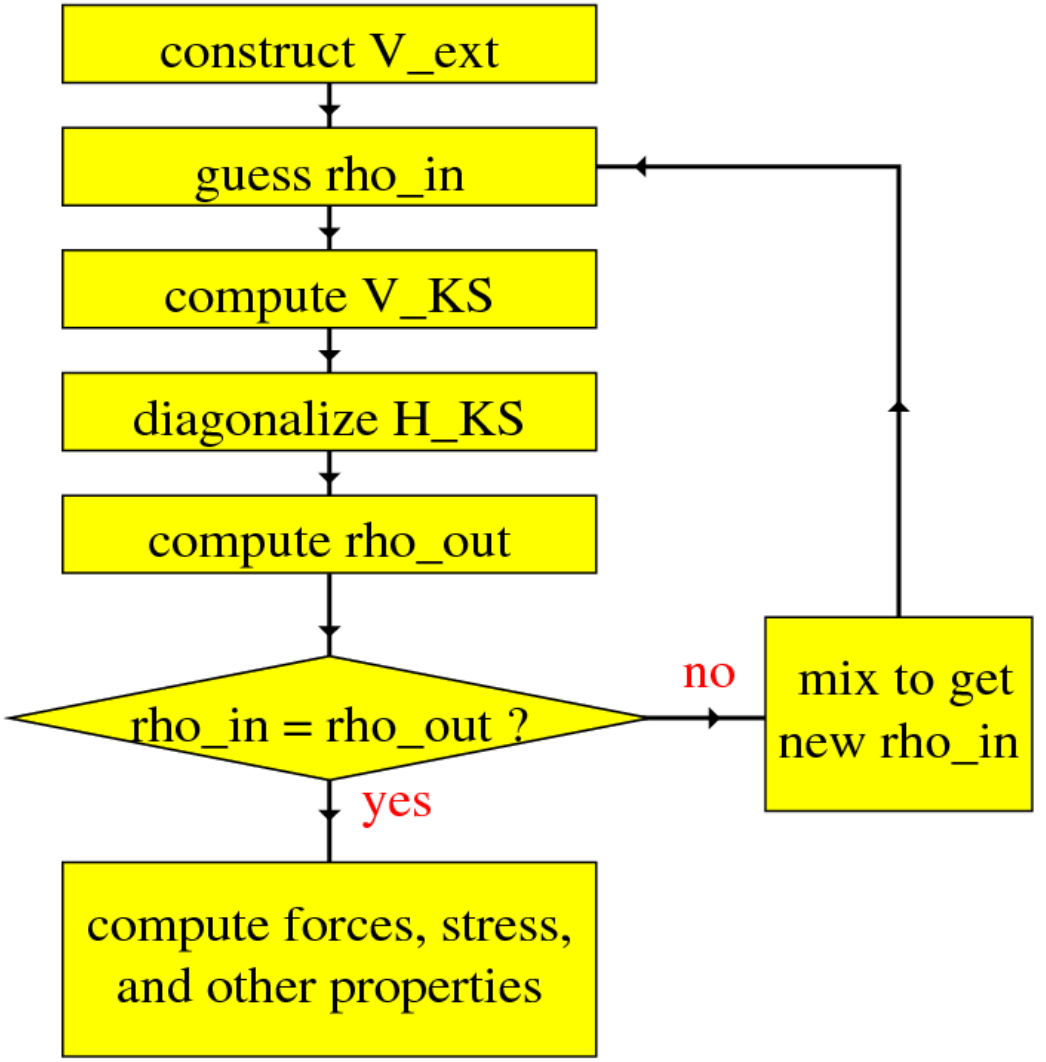
codes

$QEROOT/PW/src/pwscf.f90 → pw.x

$QEROOT/CPV/src/cp.f90 → cp.x

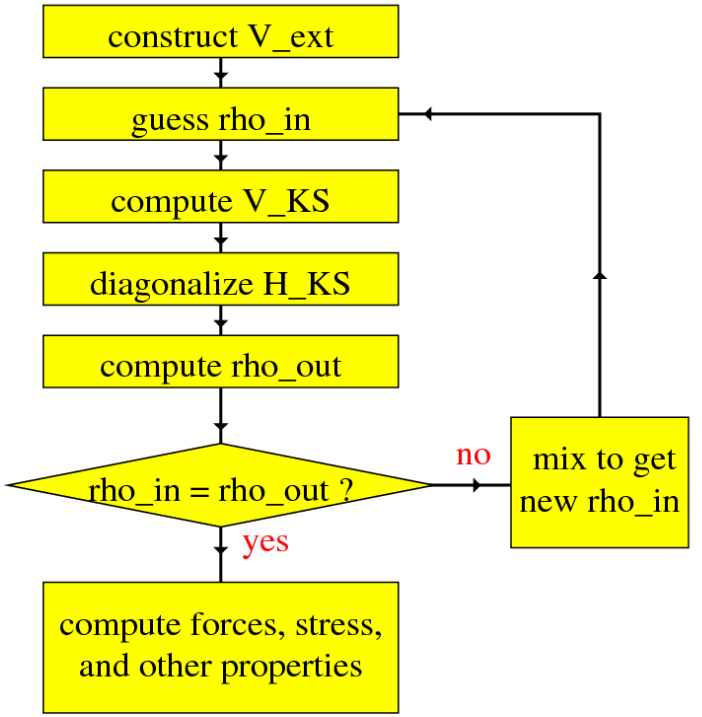
$QEROOT/PHonon/PH/phonon.f90 → ph.x

PWscf structure



pwscf.f90

```
call read_input_file ;  call iosys
call setup ; call init_run
main_loop: do
    call electrons
    If (lforce) call forces
    If (lstres) call stress
    If (md .or. lbfgs) call move_ions
end do mail_loop
```



pwscf.f90

```
call read_input_file ;  call iosys
call setup ; call init_run
main_loop: do
   call electrons
   If (lforce) call forces
   If (lstres) call stress
   If (md .or. lbfgs) call move_ions
end do mail_loop
```

setup.f90

define dimensions and system parameters

init_run.f90

allocate stuff

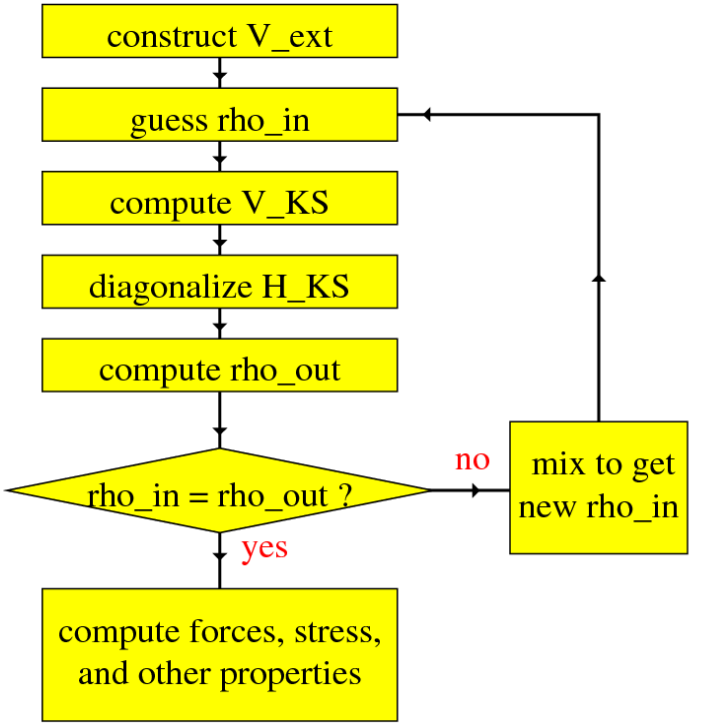
call ggen

call openfil

call hinit0

call potinit

call wfcinit

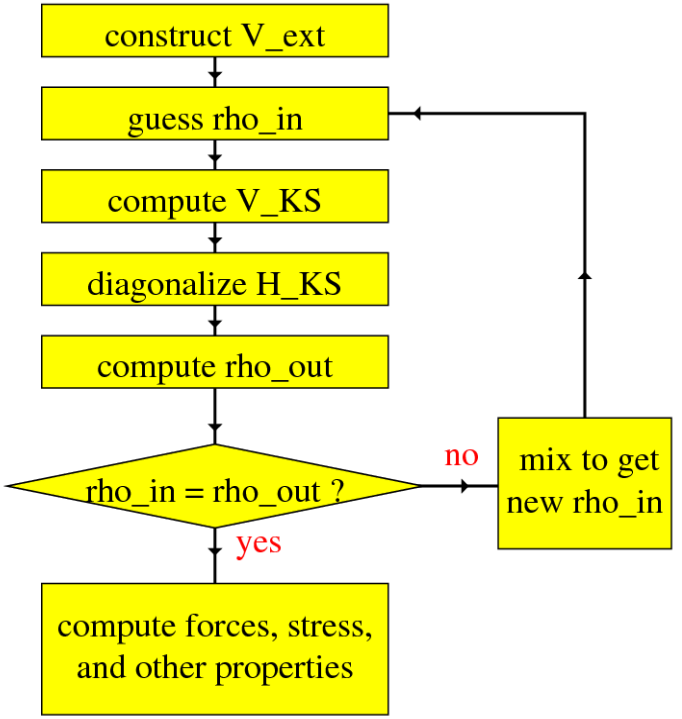


pwscf.f90

```
call read_input_file ;  call iosys
call setup ; call init_run
main_loop: do
   call electrons
   If (lforce) call forces
   If (lstres) call stress
   If (md .or. lbfgs) call move_ions
end do mail_loop
```

electrons.f90

```
ewld = ewald (…)
do iter =1, niter
    call c_bands
    call sum_bands
    call mix_rho
    call v_of_rho
    If (conv_elec) return
 end do
```



electrons.f90

```
ewld = ewald (…)
do iter =1, niter
    call c_bands          diagonalization of H_KS
    call sum_bands        density builder
    call mix_rho          scf test and extrapolation
    call v_of_rho         new potential
    If (conv_elec) return
 end do
```