ecalj/fpgw/ code document

0.1

Generated by Doxygen 1.8.6

Fri Nov 11 2016 17:43:19

Contents

Chapter 1

Data Type Index

1.1 Data Types List

Here are the data types with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Type Documentation

3.1 m_anf Module Reference

Antiferro condition module. We have line AFcond at the bottom of 'LMTO' file. Currently(feb2016), only laf is used (thus AF symmetry is not used yet for hx0fp0_sc) To access laf, need to call anfcond() in advance.

Public Member Functions

- subroutine anfcond ()

Public Attributes

- logical, protected laf
- integer, dimension(:),
  allocatable, protected ibasf

3.1.1 Detailed Description

Antiferro condition module. We have line AFcond at the bottom of 'LMTO' file. Currently(feb2016), only laf is used (thus AF symmetry is not used yet for hx0fp0_sc) To access laf, need to call anfcond() in advance.

Definition at line 5 of file m_anf.F.

3.1.2 Member Function/Subroutine Documentation

3.1.2.1 subroutine m_anf::anfcond ( )

Definition at line 14 of file m_anf.F.

Here is the caller graph for this function:

3.1.3 Member Data Documentation

3.1.3.1 integer, dimension(:), allocatable, protected m_anf::ibasf

Definition at line 8 of file m_anf.F.

3.1.3.2 logical, protected m_anf::laf

Definition at line 7 of file m_anf.F.

The documentation for this module was generated from the following file:

- gwsrc/m_anf.F

3.2 m_freq Module Reference

Frequency mesh generator.

Public Member Functions

- subroutine getfreq (epsmode, realomega, imagomega, tetra, omg2max, wemax, niw, ua, mpi__root)

    Get data set for m_freq. All arguments are input.

Public Attributes

- real(8), dimension(:),
  allocatable, protected frhis
- real(8), dimension(:),
  allocatable, protected freq_r
- real(8), dimension(:),
  allocatable, protected freq_i
- real(8), dimension(:),
  allocatable, protected wiw
- integer, protected nwhis
- integer, protected npm
- integer, protected nw_i
- integer, protected nw

3.2.1 Detailed Description

Frequency mesh generator.

- OUTPUT
    - fhris :histgram bins to accumlate im part
    - freq_r: omega along real axis
    - freq_i: omega along imag axis
    - wiw: integration weight along im axis
    - npm: npm=1 means only positive omega;npm=2 means positive and negative omega.
- NOTE: change of frequency mesh defined here may destroy consistency or not. Need check

Definition at line 9 of file m_freq.F.

3.2.2 Member Function/Subroutine Documentation

3.2.2.1 subroutine m_freq::getfreq (logical, intent(in) *epsmode,* logical, intent(in) *realomega,* logical, intent(in) *imagomega,* logical, intent(in) *tetra,* real(8), intent(in) *omg2max,* real(8) *wemax,* integer, intent(in) *niw,* real(8), intent(in) *ua,* logical, intent(in) *mpi__root*)

Get data set for m_freq. All arguments are input.

- This read GWinput (dw,omg_c) and TimeReversal()
- All arguments are input

dw∗(nw_input-3)) then !omg is in unit of Hartree

Definition at line 19 of file m_freq.F.

Here is the caller graph for this function:

3.2.3 Member Data Documentation

3.2.3.1 real(8), dimension(:), allocatable, protected m_freq::freq_i

Definition at line 10 of file m_freq.F.

3.2.3.2 real(8), dimension(:), allocatable, protected m_freq::freq_r

Definition at line 10 of file m_freq.F.

3.2.3.3 real(8), dimension(:), allocatable, protected m_freq::frhis

Definition at line 10 of file m_freq.F.

3.2.3.4 integer, protected m_freq::npm

Definition at line 11 of file m_freq.F.

3.2.3.5 integer, protected m_freq::nw

Definition at line 11 of file m_freq.F.

3.2.3.6 integer, protected m_freq::nw_i

Definition at line 11 of file m_freq.F.

3.2.3.7 integer, protected m_freq::nwhis

Definition at line 11 of file m_freq.F.

3.2.3.8 real(8), dimension(:), allocatable, protected m_freq::wiw

Definition at line 10 of file m_freq.F.

The documentation for this module was generated from the following file:

- gwsrc/m_freq.F

3.3 m_genallcf_v3 Module Reference

get basic settings of crystal structure and nlm info

Public Member Functions

- subroutine genallcf_v3 (incwfx)

Public Attributes

- character(120), protected symgrp
- character(6), dimension(:), allocatable, protected clabl
- integer, dimension(:), allocatable, protected iclass
- integer, dimension(:,:), allocatable, protected nindxv
- integer, dimension(:,:), allocatable, protected nindxc
- integer, dimension(:,:,:), allocatable, protected ncwf
- integer, dimension(:), allocatable, protected invg
- integer, dimension(:,:), allocatable, protected il
- integer, dimension(:,:), allocatable, protected in
- integer, dimension(:,:), allocatable, protected im
- integer, dimension(:), allocatable, protected ilnm
- integer, dimension(:), allocatable, protected nlnm
- integer, dimension(:), allocatable, protected ilv
- integer, dimension(:), allocatable, protected inv
- integer, dimension(:), allocatable, protected imv
- integer, dimension(:), allocatable, protected ilnmv
- integer, dimension(:), allocatable, protected nlnmv
- integer, dimension(:), allocatable, protected ilc

- integer, dimension(:), allocatable, protected inc
- integer, dimension(:), allocatable, protected imc
- integer, dimension(:), allocatable, protected ilnmc
- integer, dimension(:), allocatable, protected nlnmc
- integer, dimension(:,:), allocatable, protected nindx
- integer, dimension(:,:), allocatable, protected konf
- integer, dimension(:,:), allocatable, protected icore
- integer, dimension(:), allocatable, protected ncore
- integer, dimension(:,:,:), allocatable, protected occv
- integer, dimension(:,:,:), allocatable, protected unoccv
- integer, dimension(:,:,:), allocatable, protected occc
- integer, dimension(:,:,:), allocatable, protected unoccc
- integer, dimension(:,:,:), allocatable, protected nocc
- integer, dimension(:,:,:), allocatable, protected nunocc
- integer, dimension(:), allocatable, protected iantiferro
- integer, protected nclass
- integer, protected natom
- integer, protected nspin
- integer, protected nl
- integer, protected nn
- integer, protected nnv
- integer, protected nnc
- integer, protected ngrp
- integer, protected nlmto
- integer, protected nlnx
- integer, protected nlnxv
- integer, protected nlnxc
- integer, protected nlnmx
- integer, protected nlnmxv
- integer, protected nlnmxc
- integer, protected nctot
- real(8), dimension(:,:), allocatable, protected plat
- real(8), dimension(:,:), allocatable, protected pos
- real(8), dimension(:), allocatable, protected z
- real(8), dimension(:,:,:), allocatable, protected symgg
- real(8), protected alat
- real(8), protected deltaw

- logical, protected done_genallcf_v3 =.false.
- character(8), dimension(:),
  allocatable, protected spid
- real(8), dimension(:,:),
  allocatable ecore
- real(8) delta
- integer niw
- real(8) esmr

3.3.1 Detailed Description

get basic settings of crystal structure and nlm info

- genallcf_v3(nwin,efin,incwfx) set data
- This is old routine. Confusing. We need to clean up.

Definition at line 20 of file genallcf_mod.F.

3.3.2 Member Function/Subroutine Documentation

3.3.2.1 subroutine m_genallcf_v3::genallcf_v3 (integer(4) *incwfx*)

BZ&

frequencies&

coulomb

product basis&

core&

dimensions and constants

combine nocc,nunocc,nindx

index for allowed core states

core energies

index for core and LMTO basis

Definition at line 69 of file genallcf_mod.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.3.3 Member Data Documentation

3.3.3.1 real(8), protected m_genallcf_v3::alat

Definition at line 58 of file genallcf_mod.F.

3.3.3.2 character(6), dimension(:), allocatable, protected m_genallcf_v3::clabl

Definition at line 43 of file genallcf_mod.F.

3.3.3.3 real(8) m_genallcf_v3::delta

Definition at line 63 of file genallcf_mod.F.

3.3.3.4 real(8), protected m_genallcf_v3::deltaw

Definition at line 58 of file genallcf_mod.F.

3.3.3.5 logical, protected m_genallcf_v3::done_genallcf_v3 =.false.

Definition at line 59 of file genallcf_mod.F.

3.3.3.6 real(8), dimension(:,:), allocatable m_genallcf_v3::ecore

Definition at line 62 of file genallcf_mod.F.

3.3.3.7 real(8) m_genallcf_v3::esmr

Definition at line 65 of file genallcf_mod.F.

3.3.3.8 integer, dimension(:), allocatable, protected m_genallcf_v3::iantiferro

Definition at line 44 of file genallcf_mod.F.

3.3.3.9 integer, dimension(:), allocatable, protected m_genallcf_v3::iclass

Definition at line 44 of file genallcf_mod.F.

3.3.3.10 integer, dimension(:,:), allocatable, protected m_genallcf_v3::icore

Definition at line 44 of file genallcf_mod.F.

3.3.3.11 integer, dimension(:,:), allocatable, protected m_genallcf_v3::il

Definition at line 44 of file genallcf_mod.F.

3.3.3.12 integer, dimension(:), allocatable, protected m_genallcf_v3::ilc

Definition at line 44 of file genallcf_mod.F.

3.3.3.13 integer, dimension(:), allocatable, protected m_genallcf_v3::ilnm

Definition at line 44 of file genallcf_mod.F.

3.3.3.14 integer, dimension(:), allocatable, protected m_genallcf_v3::ilnmc

Definition at line 44 of file genallcf_mod.F.

3.3.3.15 integer, dimension(:), allocatable, protected m_genallcf_v3::ilnmv

Definition at line 44 of file genallcf_mod.F.

3.3.3.16 integer, dimension(:), allocatable, protected m_genallcf_v3::ilv

Definition at line 44 of file genallcf_mod.F.

3.3.3.17 integer, dimension(:,:), allocatable, protected m_genallcf_v3::im

Definition at line 44 of file genallcf_mod.F.

3.3.3.18 integer, dimension(:), allocatable, protected m_genallcf_v3::imc

Definition at line 44 of file genallcf_mod.F.

3.3.3.19 integer, dimension(:), allocatable, protected m_genallcf_v3::imv

Definition at line 44 of file genallcf_mod.F.

3.3.3.20 integer, dimension(:,:), allocatable, protected m_genallcf_v3::in

Definition at line 44 of file genallcf_mod.F.

3.3.3.21 integer, dimension(:), allocatable, protected m_genallcf_v3::inc

Definition at line 44 of file genallcf_mod.F.

3.3.3.22 integer, dimension(:), allocatable, protected m_genallcf_v3::inv

Definition at line 44 of file genallcf_mod.F.

3.3.3.23 integer, dimension(:), allocatable, protected m_genallcf_v3::invg

Definition at line 44 of file genallcf_mod.F.

3.3.3.24 integer, dimension(:,:), allocatable, protected m_genallcf_v3::konf

Definition at line 44 of file genallcf_mod.F.

3.3.3.25 integer, protected m_genallcf_v3::natom

Definition at line 53 of file genallcf_mod.F.

3.3.3.26 integer, protected m_genallcf_v3::nclass

Definition at line 53 of file genallcf_mod.F.

3.3.3.27 integer, dimension(:), allocatable, protected m_genallcf_v3::ncore

Definition at line 44 of file genallcf_mod.F.

3.3.3.28 integer, protected m_genallcf_v3::nctot

Definition at line 53 of file genallcf_mod.F.

3.3.3.29 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::ncwf

Definition at line 44 of file genallcf_mod.F.

3.3.3.30 integer, protected m_genallcf_v3::ngrp

Definition at line 53 of file genallcf_mod.F.

3.3.3.31 integer, dimension(:,:), allocatable, protected m_genallcf_v3::nindx

Definition at line 44 of file genallcf_mod.F.

3.3.3.32 integer, dimension(:,:), allocatable, protected m_genallcf_v3::nindxc

Definition at line 44 of file genallcf_mod.F.

3.3.3.33 integer, dimension(:,:), allocatable, protected m_genallcf_v3::nindxv

Definition at line 44 of file genallcf_mod.F.

3.3.3.34 integer m_genallcf_v3::niw

Definition at line 64 of file genallcf_mod.F.

3.3.3.35 integer, protected m_genallcf_v3::nl

Definition at line 53 of file genallcf_mod.F.

3.3.3.36 integer, protected m_genallcf_v3::nlmto

Definition at line 53 of file genallcf_mod.F.

3.3.3.37 integer, dimension(:), allocatable, protected m_genallcf_v3::nlnm

Definition at line 44 of file genallcf_mod.F.

3.3.3.38 integer, dimension(:), allocatable, protected m_genallcf_v3::nlnmc

Definition at line 44 of file genallcf_mod.F.

3.3.3.39 integer, dimension(:), allocatable, protected m_genallcf_v3::nlnmv

Definition at line 44 of file genallcf_mod.F.

3.3.3.40 integer, protected m_genallcf_v3::nlnmx

Definition at line 53 of file genallcf_mod.F.

3.3.3.41 integer, protected m_genallcf_v3::nlnmxc

Definition at line 53 of file genallcf_mod.F.

3.3.3.42 integer, protected m_genallcf_v3::nlnmxv

Definition at line 53 of file genallcf_mod.F.

3.3.3.43 integer, protected m_genallcf_v3::nlnx

Definition at line 53 of file genallcf_mod.F.

3.3.3.44 integer, protected m_genallcf_v3::nlnxc

Definition at line 53 of file genallcf_mod.F.

3.3.3.45 integer, protected m_genallcf_v3::nlnxv

Definition at line 53 of file genallcf_mod.F.

3.3.3.46 integer, protected m_genallcf_v3::nn

Definition at line 53 of file genallcf_mod.F.

3.3.3.47 integer, protected m_genallcf_v3::nnc

Definition at line 53 of file genallcf_mod.F.

3.3.3.48 integer, protected m_genallcf_v3::nnv

Definition at line 53 of file genallcf_mod.F.

3.3.3.49 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::nocc

Definition at line 44 of file genallcf_mod.F.

3.3.3.50 integer, protected m_genallcf_v3::nspin

Definition at line 53 of file genallcf_mod.F.

3.3.3.51 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::nunocc

Definition at line 44 of file genallcf_mod.F.

3.3.3.52 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::occc

Definition at line 44 of file genallcf_mod.F.

3.3.3.53 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::occv

Definition at line 44 of file genallcf_mod.F.

3.3.3.54 real(8), dimension(:,:), allocatable, protected m_genallcf_v3::plat

Definition at line 56 of file genallcf_mod.F.

3.3.3.55 real(8), dimension(:,:), allocatable, protected m_genallcf_v3::pos

Definition at line 56 of file genallcf_mod.F.

3.3.3.56 character(8), dimension(:), allocatable, protected m_genallcf_v3::spid

Definition at line 60 of file genallcf_mod.F.

3.3.3.57 real(8), dimension(:,:,:), allocatable, protected m_genallcf_v3::symgg

Definition at line 56 of file genallcf_mod.F.

3.3.3.58 character(120), protected m_genallcf_v3::symgrp

Definition at line 42 of file genallcf_mod.F.

3.3.3.59 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::unoccc

Definition at line 44 of file genallcf_mod.F.

3.3.3.60 integer, dimension(:,:,:), allocatable, protected m_genallcf_v3::unoccv

Definition at line 44 of file genallcf_mod.F.

3.3.3.61 real(8), dimension(:), allocatable, protected m_genallcf_v3::z

Definition at line 56 of file genallcf_mod.F.

The documentation for this module was generated from the following file:

- gwsrc/genallcf_mod.F

3.4 m_hamindex Module Reference

This is in lm7K/subs/m_hamindex.F and in fpgw/gwsrc/m_hamindex.F We will need to unify make system and source code in fpgw and lmf. norbtx is given in gwsrc/readeigen.F init_readeigen2.

Public Member Functions

- integer function getikt (qin)

  get index ikt such that for qin(:)=qq(:,ikt)
- subroutine writehamindex ()

  write info for wave rotation.
- subroutine readhamindex ()

  read info for wave rotation.

Public Attributes

- integer, protected ngrp =null
- integer, protected lxx =null
- integer, protected kxx =null
- integer, protected norbmto =null
- integer, protected nbas
- integer, protected nqtt
- integer, protected ndimham =null
- integer, dimension(:), allocatable, protected ltab
- integer, dimension(:), allocatable, protected ktab
- integer, dimension(:), allocatable, protected offl
- integer, dimension(:), allocatable, protected ispec
- integer, dimension(:), allocatable, protected iclasst
- integer, dimension(:,:,:), allocatable, protected offlrev
- integer, dimension(:), allocatable, protected ibastab
- integer, dimension(:), allocatable, protected iqimap
- integer, dimension(:), allocatable, protected iqmap
- integer, dimension(:), allocatable, protected igmap
- integer, dimension(:), allocatable, protected invgx
- integer, dimension(:,:), allocatable, protected miat
- integer, dimension(:), allocatable, protected ibasindex
- real(8), dimension(:,:,:), allocatable, protected symops
- real(8), dimension(:,:), allocatable, protected ag

- real(8), dimension(:,:,:),
  allocatable, protected tiat
- real(8), dimension(:,:),
  allocatable, protected shtvg
- real(8), dimension(:,:,:,:),
  allocatable, protected dlmm
- real(8), dimension(:,:),
  allocatable, protected qq
- real(8), dimension(3, 3), protected plat
- real(8), dimension(3, 3), protected qlat
- real(8), dimension(:,:),
  allocatable, protected qtt
- real(8), dimension(:,:),
  allocatable, protected qtti
- integer, dimension(:,:,:),
  allocatable, protected igv2
- integer, dimension(:),
  allocatable, protected napwk
- integer, dimension(:,:,:,:),
  allocatable, protected igv2rev
- integer, protected napwmx =null
- integer, protected lxxa =null
- integer norbtx =null
- integer nqi
- integer nqnum
- integer ngpmx
- integer imx =null

Private Attributes

- integer, parameter, private null =-999999
- logical, private debug =.false.

3.4.1 Detailed Description

This is in lm7K/subs/m_hamindex.F and in fpgw/gwsrc/m_hamindex.F We will need to unify make system and source code in fpgw and lmf. norbtx is given in gwsrc/readeigen.F init_readeigen2.

Definition at line 4 of file m_hamindex.F.

3.4.2 Member Function/Subroutine Documentation

3.4.2.1 integer function m_hamindex::getikt (real(8), dimension(3) *qin*)

get index ikt such that for qin(:)=qq(:,ikt)

Definition at line 24 of file m_hamindex.F.

3.4.2.2 subroutine m_hamindex::readhamindex ( )

read info for wave rotation.

Definition at line 65 of file m_hamindex.F.

3.4.2.3 subroutine m_hamindex::writehamindex ( )

write info for wave rotation.

Definition at line 43 of file m_hamindex.F.

3.4.3 Member Data Documentation

3.4.3.1 real(8), dimension(:,:), allocatable, protected m_hamindex::ag

Definition at line 12 of file m_hamindex.F.

3.4.3.2 logical, private m_hamindex::debug =.false. [private]

Definition at line 6 of file m_hamindex.F.

3.4.3.3 real(8), dimension(:,:,:,:), allocatable, protected m_hamindex::dlmm

Definition at line 12 of file m_hamindex.F.

3.4.3.4 integer, dimension(:), allocatable, protected m_hamindex::ibasindex

Definition at line 11 of file m_hamindex.F.

3.4.3.5 integer, dimension(:), allocatable, protected m_hamindex::ibastab

Definition at line 10 of file m_hamindex.F.

3.4.3.6 integer, dimension(:), allocatable, protected m_hamindex::iclasst

Definition at line 10 of file m_hamindex.F.

3.4.3.7 integer, dimension(:), allocatable, protected m_hamindex::igmap

Definition at line 11 of file m_hamindex.F.

3.4.3.8 integer, dimension(:,:,:), allocatable, protected m_hamindex::igv2

Definition at line 15 of file m_hamindex.F.

3.4.3.9 integer, dimension(:,:,:,:), allocatable, protected m_hamindex::igv2rev

Definition at line 15 of file m_hamindex.F.

3.4.3.10 integer m_hamindex::imx =null

Definition at line 20 of file m_hamindex.F.

3.4.3.11 integer, dimension(:), allocatable, protected m_hamindex::invgx

Definition at line 11 of file m_hamindex.F.

3.4.3.12 integer, dimension(:), allocatable, protected m_hamindex::iqimap

Definition at line 11 of file m_hamindex.F.

3.4.3.13 integer, dimension(:), allocatable, protected m_hamindex::iqmap

Definition at line 11 of file m_hamindex.F.

3.4.3.14 integer, dimension(:), allocatable, protected m_hamindex::ispec

Definition at line 10 of file m_hamindex.F.

3.4.3.15 integer, dimension(:), allocatable, protected m_hamindex::ktab

Definition at line 10 of file m_hamindex.F.

3.4.3.16 integer, protected m_hamindex::kxx =null

Definition at line 8 of file m_hamindex.F.

3.4.3.17 integer, dimension(:), allocatable, protected m_hamindex::ltab

Definition at line 10 of file m_hamindex.F.

3.4.3.18 integer, protected m_hamindex::lxx =null

Definition at line 8 of file m_hamindex.F.

3.4.3.19 integer, protected m_hamindex::lxxa =null

Definition at line 16 of file m_hamindex.F.

3.4.3.20 integer, dimension(:,:), allocatable, protected m_hamindex::miat

Definition at line 11 of file m_hamindex.F.

3.4.3.21 integer, dimension(:), allocatable, protected m_hamindex::napwk

Definition at line 15 of file m_hamindex.F.

3.4.3.22 integer, protected m_hamindex::napwmx =null

Definition at line 16 of file m_hamindex.F.

3.4.3.23 integer, protected m_hamindex::nbas

Definition at line 9 of file m_hamindex.F.

3.4.3.24 integer, protected m_hamindex::ndimham =null

Definition at line 9 of file m_hamindex.F.

3.4.3.25 integer m_hamindex::ngpmx

Definition at line 20 of file m_hamindex.F.

3.4.3.26 integer, protected m_hamindex::ngrp =null

Definition at line 8 of file m_hamindex.F.

3.4.3.27 integer, protected m_hamindex::norbmto =null

Definition at line 8 of file m_hamindex.F.

3.4.3.28 integer m_hamindex::norbtx =null

Definition at line 19 of file m_hamindex.F.

3.4.3.29 integer m_hamindex::nqi

Definition at line 20 of file m_hamindex.F.

3.4.3.30 integer m_hamindex::nqnum

Definition at line 20 of file m_hamindex.F.

3.4.3.31 integer, protected m_hamindex::nqtt

Definition at line 9 of file m_hamindex.F.

3.4.3.32 integer, parameter, private m_hamindex::null =-999999 `[private]`

Definition at line 5 of file m_hamindex.F.

3.4.3.33 integer, dimension(:), allocatable, protected m_hamindex::offl

Definition at line 10 of file m_hamindex.F.

3.4.3.34 integer, dimension(:,:,:), allocatable, protected m_hamindex::offlrev

Definition at line 10 of file m_hamindex.F.

3.4.3.35 real(8), dimension(3,3), protected m_hamindex::plat

Definition at line 13 of file m_hamindex.F.

3.4.3.36 real(8), dimension(3,3), protected m_hamindex::qlat

Definition at line 13 of file m_hamindex.F.

3.4.3.37 real(8), dimension(:,:), allocatable, protected m_hamindex::qq

Definition at line 12 of file m_hamindex.F.

3.4.3.38 real(8), dimension(:,:), allocatable, protected m_hamindex::qtt

Definition at line 14 of file m_hamindex.F.

3.4.3.39 real(8), dimension(:,:), allocatable, protected m_hamindex::qtti

Definition at line 14 of file m_hamindex.F.

3.4.3.40 real(8), dimension(:,:), allocatable, protected m_hamindex::shtvg

Definition at line 12 of file m_hamindex.F.

3.4.3.41 real(8), dimension(:,:,:), allocatable, protected m_hamindex::symops

Definition at line 12 of file m_hamindex.F.

3.4.3.42 real(8), dimension(:,:,:), allocatable, protected m_hamindex::tiat

Definition at line 12 of file m_hamindex.F.

The documentation for this module was generated from the following file:

- gwsrc/m_hamindex.F

3.5 m_readefermi Module Reference

Public Member Functions

- subroutine readefermi ()

Public Attributes

- real(8), protected bandgap
- real(8) ef

3.5.1 Detailed Description

Definition at line 1 of file genallcf_mod.F.

3.5.2 Member Function/Subroutine Documentation

3.5.2.1 subroutine m_readefermi::readefermi ( )

Definition at line 6 of file genallcf_mod.F.

Here is the caller graph for this function:

3.5.3 Member Data Documentation

3.5.3.1 real(8), protected m_readefermi::bandgap

Definition at line 2 of file genallcf_mod.F.

3.5.3.2 real(8) m_readefermi::ef

Definition at line 3 of file genallcf_mod.F.

The documentation for this module was generated from the following file:

- gwsrc/genallcf_mod.F

3.6 m_readq0p Module Reference

Public Member Functions

- subroutine readq0p ()

Public Attributes

- real(8), dimension(:),
  allocatable, protected wqt
- real(8), dimension(:,:),
  allocatable, protected wgt0
- real(8), dimension(:,:),
  allocatable, protected q0i
- integer, protected nq0i
- integer, protected nq0iadd
- integer, dimension(:),
  allocatable, protected ixyz

3.6.1 Detailed Description

Definition at line 1 of file readqg.F.

3.6.2 Member Function/Subroutine Documentation

3.6.2.1 subroutine m_readq0p::readq0p ( )

Definition at line 7 of file readqg.F.

Here is the caller graph for this function:

3.6.3 Member Data Documentation

3.6.3.1 integer, dimension(:), allocatable, protected m_readq0p::ixyz

Definition at line 4 of file readqg.F.

3.6.3.2 integer, protected m_readq0p::nq0i

Definition at line 3 of file readqg.F.

3.6.3.3 integer, protected m_readq0p::nq0iadd

Definition at line 3 of file readqg.F.

3.6.3.4 real(8), dimension(:,:), allocatable, protected m_readq0p::q0i

Definition at line 2 of file readqg.F.

3.6.3.5 real(8), dimension(:,:), allocatable, protected m_readq0p::wgt0

Definition at line 2 of file readqg.F.

3.6.3.6 real(8), dimension(:), allocatable, protected m_readq0p::wqt

Definition at line 2 of file readqg.F.

The documentation for this module was generated from the following file:

- gwsrc/readqg.F

3.7 m_readqg Module Reference

Return QGcou and QGpsi ===.

Public Member Functions

- subroutine readngmx (key, ngmx)
- subroutine readqg (key, qin, ginv, qu, ngv, ngvec)

  Get ngv and ngvec(3,ngv) for given qin(3) key=='QGcou' or 'QGpsi'.
- subroutine readqg0 (key, qin, ginv, qu, ngv)

  Get ngv key=='QGcou' or 'QGpsi'.
- subroutine init_readqg (ifi, ginv)

  initialization. readin QGpsi or QGcou.
- subroutine tabkk (kkin, kktable, n, nout)
- subroutine iqindx2qg (q, ifi, iqindx, qu)

Find index as q=qq(:,iq) with modulo of premitive vector. ginv is the inverse of plat (premitive translation vector). Use kk1,kk2,kk3,nkey(1:3),iqkkk to get iqindx.

- subroutine sortea (ea, ieaord, n, isig)

  mini-sort routine.
- subroutine iswap (i, j)

Private Attributes

- real(8), dimension(:,:), allocatable, target, private qc
- real(8), dimension(:,:), allocatable, target, private qp
- logical, dimension(2), private init =.true.
- real(8), private qpgcut_cou
- real(8), private qpgcut_psi
- integer(4), target, private nqnumc
- integer(4), target, private nqnump
- integer(4), target, private ngcmx
- integer(4), target, private ngpmx
- integer(4), dimension(:,:,:), allocatable, private ngvecp
- integer(4), dimension(:), allocatable, private ngp
- integer(4), dimension(:,:,:), allocatable, private ngvecc
- integer(4), dimension(:), allocatable, private ngc
- integer, pointer, private nqtt
- real(8), dimension(:,:), pointer, private qtt
- real(8), private epsd =1d-7
- integer, dimension(:), pointer, private nkey
- integer, dimension(:), pointer, private kk1
- integer, dimension(:), pointer, private kk2
- integer, dimension(:), pointer, private kk3
- integer, dimension(:,:,:), pointer, private iqkkk
- integer, dimension(3), target, private nkeyp
- integer, dimension(3), target, private nkeyc
- integer, dimension(:,:), allocatable, target, private keyp
- integer, dimension(:), allocatable, target, private kk1p
- integer, dimension(:), allocatable, target, private kk2p
- integer, dimension(:), allocatable, target, private kk3p
- integer, dimension(:,:,:), allocatable, target, private iqkkkp

- integer, dimension(:,:),
  allocatable, target, private keyc
- integer, dimension(:),
  allocatable, target, private kk1c
- integer, dimension(:),
  allocatable, target, private kk2c
- integer, dimension(:),
  allocatable, target, private kk3c
- integer, dimension(:,:,:),
  allocatable, target, private iqkkkc
- real(8), dimension(3, 3), private ginv_

3.7.1 Detailed Description

Return QGcou and QGpsi ===.

Definition at line 55 of file readqg.F.

3.7.2 Member Function/Subroutine Documentation

3.7.2.1 subroutine m_readqg::init_readqg (integer(4), intent(in) *ifi,* real(8), dimension(3,3), intent(in) *ginv*)

initialization. readin QGpsi or QGcou.

Definition at line 164 of file readqg.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.7.2.2 subroutine m_readqg::iqindx2qg (real(8), dimension(3), intent(in) *q,* integer, intent(in) *ifi,* integer, intent(out) *iqindx,* real(8), dimension(3), intent(out) *qu*)

Find index as q=qq(:,iq) with modulo of premitive vector. ginv is the inverse of plat (premitive translation vector). Use kk1,kk2,kk3,nkey(1:3),iqkkk to get iqindx.

Definition at line 344 of file readqg.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.7.2.3 subroutine m_readqg::iswap (integer, intent(inout) *i,* integer, intent(inout) *j*)

Definition at line 412 of file readqg.F.

Here is the caller graph for this function:

3.7.2.4 subroutine m_readqg::readngmx (character∗(∗) *key,* integer(4) *ngmx*)

Definition at line 72 of file readqg.F.

Here is the caller graph for this function:

3.7.2.5 subroutine m_readqg::readqg (character∗(∗), intent(in) *key,* real(8), dimension(3), intent(in) *qin,* real(8), dimension(3,3), intent(in) *ginv,* real(8), dimension(3), intent(out) *qu,* integer(4), intent(out) *ngv,* integer(4), dimension(3,∗), intent(out) *ngvec*)

Get ngv and ngvec(3,ngv) for given qin(3) key==’QGcou’ or ’QGpsi’.

Definition at line 93 of file readqg.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.7.2.6 subroutine m_readqg::readqg0 (character∗(∗), intent(in) *key,* real(8), dimension(3), intent(in) *qin,* real(8), dimension(3,3), intent(in) *ginv,* real(8), dimension(3), intent(out) *qu,* integer(4), intent(out) *ngv*)

Get ngv key==’QGcou’ or ’QGpsi’.

Definition at line 130 of file readqg.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.7.2.7 subroutine m_readqg::sortea (real(8), dimension(n), intent(in) *ea,* integer(4), dimension(n), intent(inout) *ieaord,* integer, intent(in) *n,* integer, intent(out) *isig*)

mini-sort routine.

Definition at line 391 of file readqg.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.7.2.8 subroutine m_readqg::tabkk (integer *kkin*, integer, dimension(n) *kktable*, integer *n*, integer *nout*)

Definition at line 296 of file readqg.F.

Here is the caller graph for this function:

3.7.3 Member Data Documentation

3.7.3.1 real(8), private m_readqg::epsd =1d-7 [private]

Definition at line 64 of file readqg.F.

3.7.3.2 real(8), dimension(3,3), private m_readqg::ginv_ [private]

Definition at line 69 of file readqg.F.

3.7.3.3 logical, dimension(2), private m_readqg::init =.true. [private]

Definition at line 58 of file readqg.F.

3.7.3.4 integer, dimension(:,:,:), pointer, private m_readqg::iqkkk [private]

Definition at line 65 of file readqg.F.

3.7.3.5 integer, dimension(:,:,:), allocatable, target, private m_readqg::iqkkkc [private]

Definition at line 68 of file readqg.F.

3.7.3.6 integer, dimension(:,:,:), allocatable, target, private m_readqg::iqkkkp [private]

Definition at line 67 of file readqg.F.

3.7.3.7 integer, dimension(:,:), allocatable, target, private m_readqg::keyc [private]

Definition at line 68 of file readqg.F.

3.7.3.8 integer, dimension(:,:), allocatable, target, private m_readqg::keyp [private]

Definition at line 67 of file readqg.F.

3.7.3.9 integer, dimension(:), pointer, private m_readqg::kk1 [private]

Definition at line 65 of file readqg.F.

3.7.3.10 integer, dimension(:), allocatable, target, private m_readqg::kk1c [private]

Definition at line 68 of file readqg.F.

3.7.3.11 integer, dimension(:), allocatable, target, private m_readqg::kk1p [private]

Definition at line 67 of file readqg.F.

3.7.3.12 integer, dimension(:), pointer, private m_readqg::kk2 [private]

Definition at line 65 of file readqg.F.

3.7.3.13 integer, dimension(:), allocatable, target, private m_readqg::kk2c [private]

Definition at line 68 of file readqg.F.

3.7.3.14 integer, dimension(:), allocatable, target, private m_readqg::kk2p [private]

Definition at line 67 of file readqg.F.

3.7.3.15 integer, dimension(:), pointer, private m_readqg::kk3 [private]

Definition at line 65 of file readqg.F.

3.7.3.16 integer, dimension(:), allocatable, target, private m_readqg::kk3c [private]

Definition at line 68 of file readqg.F.

3.7.3.17 integer, dimension(:), allocatable, target, private m_readqg::kk3p [private]

Definition at line 67 of file readqg.F.

3.7.3.18 integer(4), dimension(:), allocatable, private m_readqg::ngc [private]

Definition at line 61 of file readqg.F.

3.7.3.19 integer(4), target, private m_readqg::ngcmx [private]

Definition at line 60 of file readqg.F.

3.7.3.20 integer(4), dimension(:), allocatable, private m_readqg::ngp [private]

Definition at line 61 of file readqg.F.

3.7.3.21 integer(4), target, private m_readqg::ngpmx [private]

Definition at line 60 of file readqg.F.

3.7.3.22 integer(4), dimension(:,:,:), allocatable, private m_readqg::ngvecc [private]

Definition at line 61 of file readqg.F.

3.7.3.23 integer(4), dimension(:,:,:), allocatable, private m_readqg::ngvecp [private]

Definition at line 61 of file readqg.F.

3.7.3.24 integer, dimension(:), pointer, private m_readqg::nkey [private]

Definition at line 65 of file readqg.F.

3.7.3.25 integer, dimension(3), target, private m_readqg::nkeyc [private]

Definition at line 66 of file readqg.F.

3.7.3.26 integer, dimension(3), target, private m_readqg::nkeyp [private]

Definition at line 66 of file readqg.F.

3.7.3.27 integer(4), target, private m_readqg::nqnumc [private]

Definition at line 60 of file readqg.F.

3.7.3.28 integer(4), target, private m_readqg::nqnump [private]

Definition at line 60 of file readqg.F.

3.7.3.29 integer, pointer, private m_readqg::nqtt [private]

Definition at line 62 of file readqg.F.

3.7.3.30 real(8), dimension(:,:), allocatable, target, private m_readqg::qc [private]

Definition at line 57 of file readqg.F.

3.7.3.31 real(8), dimension(:,:), allocatable, target, private m_readqg::qp [private]

Definition at line 57 of file readqg.F.

3.7.3.32 real(8), private m_readqg::qpgcut_cou [private]

Definition at line 59 of file readqg.F.

3.7.3.33 real(8), private m_readqg::qpgcut_psi [private]

Definition at line 59 of file readqg.F.

3.7.3.34 real(8), dimension(:,:), pointer, private m_readqg::qtt [private]

Definition at line 63 of file readqg.F.

The documentation for this module was generated from the following file:

- gwsrc/readqg.F

3.8 m_sxcfsc Module Reference

this module is only because name=name argument binding. No data

Public Member Functions

- subroutine sxcf_fal3_scz (kount, qip, itq, ntq, ef, esmr, nsp, isp, qbas, ginv, qibz, qbz, wk, nstbz, irkip, nrkip, freq_r, nw_i, nw, freqx, wx, dwdummy, ecore, nlmto, nqibz, nqbz, nctot, nbloch, ngrp, niw, nq, nblochpmx, ngpmx, ngcmx, wgt0, nq0i, q0i, symgg, alat, nband, ifvcfpout, exchange, screen, cohtest, ifexsp, nbmx, ebmx, wklm, lxklm, eftrue, jobsw, hermitianW, zsec, coh, nbandmx)
- subroutine weightset4intreal (nctot, esmr, omega, ekc, freq_r, nw_i, nw, ntqxx, nt0m, nt0p, ef, nwx, nwxi, nt_max, wfaccut, wtt, we_, wfac_, ixss, ititpskip, iirx)

3.8.1 Detailed Description

this module is only because name=name argument binding. No data

Definition at line 2 of file sxcf_fal2.sc.F.

3.8.2 Member Function/Subroutine Documentation

3.8.2.1 subroutine m_sxcfsc::sxcf_fal3_scz (integer, dimension(nqibz,nq), intent(in) *kount*, real(8), dimension(3,nq), intent(in) *qip*, integer, dimension(ntq), intent(in) *itq*, integer, intent(in) *ntq*, real(8), intent(in) *ef*, real(8), intent(in) *esmr*, integer, intent(in) *nsp*, integer, intent(in) *isp*, real(8), dimension(3,3), intent(in) *qbas*, real(8), dimension(3,3), intent(in) *ginv*, real(8), dimension(3,nqibz), intent(in) *qibz*, real(8), dimension(3,nqbz), intent(in) *qbz*, real(8), dimension(nqbz), intent(in) *wk*, integer, dimension(nqbz), intent(in) *nstbz*, integer, dimension(nqibz,ngrp,nq), intent(in) *irkip*, integer, dimension(nqibz,ngrp,nq), intent(in) *nrkip*, real(8), dimension(nw_i:nw), intent(in) *freq_r*, integer *nw_i*, integer *nw*, real(8), dimension(niw), intent(in) *freqx*, real(8), dimension(niw), intent(in) *wx*, real(8), intent(in) *dwdummy*, real(8), dimension(nctot), intent(in) *ecore*, integer, intent(in) *nlmto*, integer, intent(in) *nqibz*, integer, intent(in) *nqbz*, integer, intent(in) *nctot*, integer, intent(in) *nbloch*, integer, intent(in) *ngrp*, integer, intent(in) *niw*, integer, intent(in) *nq*, integer, intent(in) *nblochpmx*, integer, intent(in) *ngpmx*, integer, intent(in) *ngcmx*, real(8), dimension(nq0i,ngrp), intent(in) *wgt0*, integer, intent(in) *nq0i*, real(8), dimension(1:3,1:nq0i), intent(in) *q0i*, real(8), dimension(3,3,ngrp), intent(in) *symgg*, real(8), intent(in) *alat*, integer, intent(in) *nband*, integer, intent(in) *ifvcfpout*, logical, intent(in) *exchange*, logical, intent(in) *screen*, logical, intent(in) *cohtest*, integer, intent(in) *ifexsp*, integer, dimension(2), intent(in) *nbmx*, real(8), dimension(2), intent(in) *ebmx*, real(8), dimension((lxklm+1)∗∗2), intent(in) *wklm*, integer, intent(in) *lxklm*, real(8), intent(in) *eftrue*, integer, intent(in) *jobsw*, logical *hermitianW*, complex(8), dimension(ntq,ntq,nq), intent(out), optional *zsec*, complex(8), dimension(ntq,nq), intent(out), optional *coh*, integer, dimension(nq), intent(in) *nbandmx*)

Calcualte full simga_ij(e_i)= <i|Re[Sigma](e_i)|j>

Parameters

exchange	• T : Calculate the exchange self-energy • F : Calculate correlated part of the self-energy
zsec	• S_ij= <i|Re[S](e_i)|j> • Note that S_ij itself is not Hermite becasue it includes e_i. i and j are band indexes
coh	dummy
screen	dummy

Remarks

```
Jan2013: eftrue is added.
  ef=eftrue(true fermi energy) for valence exchange and correlation mode.
  but ef is not the true fermi energy for core-exchange mode.

Jan2006
    "zsec from im-axis integral part"  had been symmetrized as
    &        wtt*.5d0*(   sum(zwzi(:,itp,itpp))+ !S_{ij}(e_i)
    &        dconjg( sum(zwzi(:,itpp,itp)) )   ) !S_{ji}^*(e_j)= S_{ij}(e_j)
    However, I now do it just the 1st term.
    &        wtt* sum(zwzi(:,itp,itpp))   !S_{ij}(e_i)
    This is OK because the symmetrization is in hqpe.sc.F
    Now zsec given in this routine is simply written as <i|Re[S](e_i)|j>.
    ( In the version until Jan2006 (fpgw032f8), only the im-axis part was symmetrized.
    But it was not necessary from the begining because it was done in hqpe.sc.F

    (Be careful as for the difference between
    <i|Re[S](e_i)|j> and transpose(dconjg(<i|Re[S](e_i)|j>)).
    ---because e_i is included.
    The symmetrization (hermitian) procedure is inlucded in hqpe.sc.F

    NOTE: matrix element is given by "call get_zmelt". It returns  zmelt or zmeltt.

jobsw switch
 1-5 scGW mode.
  diag+@EF      jobsw==1 SE_nn'(ef)+delta_nn'(SE_nn(e_n)-SE_nn(ef))
  xxx modeB (Not Available now)  jobsw==2 SE_nn'((e_n+e_n')/2)  !we need to recover comment out for jobsw==2, an
  mode A        jobsw==3 (SE_nn'(e_n)+SE_nn'(e_n'))/2 (Usually usued in QSGW).
  @Ef           jobsw==4 SE_nn'(ef)
  diagonly      jobsw==5 delta_nn' SE_nn(e_n) (not efficient memoryuse; but we don't use this mode so often).

Output file in hsfp0 should contain hermitean part of SE
   ( hermitean of SE_nn'(e_n) means SE_n'n(e_n')^* )
            we use that zwz(itp,itpp)=dconjg( zwz(itpp,itp) )
Caution! npm=2 is not examined enough...

Calculate the exchange part and the correlated part of self-energy.
T.Kotani started development after the analysis of F.Aryasetiawan's LMTO-ASA-GW.
We still use some of his ideas in this code.

See paper
[1]T. Kotani and M. van Schilfgaarde, ??Quasiparticle self-consistent GW method:
    A basis for the independent-particle approximation, Phys. Rev. B, vol. 76, no. 16, p. 165106[24pages], Oct. 
[2]T. Kotani, Quasiparticle Self-Consistent GW Method Based on the Augmented Plane-Wave
   and Muffin-Tin Orbital Method, J. Phys. Soc. Jpn., vol. 83, no. 9, p. 094711 [11 Pages], Sep. 2014.

--------------------------------------------------------------------------------
Omega integral for SEc
```

```
  The integral path is deformed along the imaginary-axis, but together with contribution of poles.
  See Fig.1 and around in Ref.[1].

---Integration along imaginary axis.---
  ( Current version for it, wintzsg_npm, do not assume time-reversal when npm=2.)
  Integration along the imaginary axis: -----------------
   (Here is a memo by F.Aryasetiawan.)
    (i/2pi) < [w'=-inf,inf] Wc(k,w')(i,j)/(w'+w-e(q-k,n) >
   Gaussian integral along the imaginary axis.
   transform: x = 1/(1+w')
    this leads to a denser mesh in w' around 0 for equal mesh x
   which is desirable since Wc and the lorentzian are peaked around w'=0
    wint = - (1/pi) < [x=0,1] Wc(iw') (w-e)x^2/{(w-e)^2 + w'^2} >

    the integrand is peaked around w'=0 or x=1 when w=e
    to handel the problem, add and substract the singular part as follows:
    wint = - (1/pi) < [x=0,1] { Wc(iw') - Wc(0)exp(-a^2 w'^2) }
    * (w-e)/{(w-e)^2 +w'^2}x^2 >
    - (1/2) Wc(0) sgn(w-e) exp(a^2 (w-e)^2) erfc(a|w-e|)

    the second term of the integral can be done analytically, which
    results in the last term a is some constant

    when w = e, (1/pi) (w-e)/{(w-e)^2 + w'^2} ==> delta(w') and
    the integral becomes -Wc(0)/2
    this together with the contribution from the pole of G (s.u.)
    gives the so called static screened exchange -Wc(0)

---Integration along real axis (contribution from the poles of G: SEc(pole))
   See Eq.(34),(55), and (58) and around in Ref.[1]. We now use Gaussian Smearing.
-------------------------------------------------------------------------------


----------------------------------------------
    q     =qip(:,iq)  = q-vector in SEc(q,t).
   itq     = states t at q
   ntq     = no. states t
   eq      = eigenvalues at q
    ef      = fermi level in Rydberg
  WVI, WVR: direct access files for W. along im axis (WVI) or along real axis (WVR)
  freq_r(nw_i:nw)   = frequencies along real axis. freq_r(0)=0d0

   qbas    = base reciprocal lattice vectors
   ginv    = inverse of qbas s. indxrk.f

    wk     = weight for each k-point in the FBZ
   qbz     = k-points in the 1st BZ

   wx      = weights at gaussian points x between (0,1)
    ua_      = constant in exp(-ua^2 w'^2) s. wint.f
    expa    = exp(-ua^2 w'^2) s. wint.f

   irkip(k,R,nq) = gives index in the FBZ with k{IBZ, R=rotation

  nqibz   = number of k-points in the irreducible BZ
  nqbz    =                          full BZ
   natom   = number of atoms
   nctot   = total no. of allowed core states
   nbloch  = total number of Bloch basis functions
   nlmto   = total number of MTO+lo basis functions
   ngrp    = no. group elements (rotation matrices)
   niw     = no. frequencies along the imaginary axis
   nw_i:nw  = no. frequencies along the real axis. nw_i=0 or -nw.
   zsec(itp,itpp,iq)> = <psi(itp,q(:,iq)) |SEc| psi(iq,q(:,iq)>

----------------------------------------------
```

Definition at line 4 of file sxcf_fal2.sc.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.8.2.2 subroutine m_sxcfsc::weightset4intreal (integer, intent(in) *nctot,* real(8), intent(in) *esmr,* real(8), dimension(ntqxx), intent(in) *omega,* real(8), dimension(ntqxx), intent(in) *ekc,* real(8), dimension(nw_i:nw), intent(in) *freq_r,* integer, intent(in) *nw_i,* integer, intent(in) *nw,* integer, intent(in) *ntqxx,* integer, intent(in) *nt0m,* integer *nt0p,* real(8), intent(in) *ef,* integer, intent(in) *nwx,* integer, intent(in) *nwxi,* integer, intent(in) *nt_max,* real(8), intent(in) *wfaccut,* real(8), intent(in) *wtt,* real(8), dimension(nt_max,ntqxx), intent(out) *we_,* real(8), dimension(nt_max,ntqxx), intent(out) *wfac_,* integer, dimension(nt_max,ntqxx), intent(out) *ixss,* logical, dimension(nt_max,ntqxx), intent(out) *ititpskip,* integer, dimension(ntqxx), intent(out) *iirx*)

Definition at line 1247 of file sxcf_fal2.sc.F.

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- gwsrc/sxcf_fal2.sc.F

3.9 m_tetwt Module Reference

Get the weights and index for tetrahedron method for the Lindhard function.

Public Member Functions

- subroutine tetdeallocate ()
- subroutine gettetwt (q, iq, is, isf, nwgt, frhis, nwhis, npm,

Public Attributes

- real(8), dimension(:),
  allocatable, protected whw
- integer, dimension(:,:,:),
  allocatable, protected ihw
- integer, dimension(:,:,:),
  allocatable, protected nhw
- integer, dimension(:,:,:),
  allocatable, protected jhw
- integer, dimension(:,:,:,:),
  allocatable, protected ibjb
- integer, protected nbnbx
- integer, protected nhwtot
- integer, dimension(:,:,:),
  allocatable, protected n1b
- integer, dimension(:,:,:),
  allocatable, protected n2b
- integer, dimension(:,:),
  allocatable, protected nbnb

3.9.1 Detailed Description

Get the weights and index for tetrahedron method for the Lindhard function.

- nbnb = total number of weight.
- n1b = band index for occ. 1 n1b nband+nctot. "Valence index->core index" ordering(Core index follows valence index).
- n2b = band index for unocc. 1 n2b nband
- wwk(ibib,...) = (complex)weight for the pair for n1b(ibib...),n2b(ibib...).

NOTE: 'call getbzdata1' generates nteti,ntetf,... See mkqg.F about how to call it.

Definition at line 10 of file m_tetwt.F.

3.9.2 Member Function/Subroutine Documentation

3.9.2.1 subroutine m_tetwt::gettetwt (real(8), dimension(3), intent(in) *q,* integer, intent(in) *iq,* integer, intent(in) *is,* integer, intent(in) *isf,* integer, dimension(:), intent(in) *nwgt,* real(8), dimension(1:nwhis+1), intent(in) *frhis,* integer, intent(in) *nwhis,* integer, intent(in) *npm*)

Definition at line 22 of file m_tetwt.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.9.2.2 subroutine m_tetwt::tetdeallocate ( )

Definition at line 17 of file m_tetwt.F.

Here is the caller graph for this function:

3.9.3 Member Data Documentation

3.9.3.1 integer, dimension(:,:,:,:), allocatable, protected m_tetwt::ibjb

Definition at line 12 of file m_tetwt.F.

3.9.3.2 integer, dimension(:,:,:), allocatable, protected m_tetwt::ihw

Definition at line 12 of file m_tetwt.F.

3.9.3.3 integer, dimension(:,:,:), allocatable, protected m_tetwt::jhw

Definition at line 12 of file m_tetwt.F.

3.9.3.4 integer, dimension(:,:,:), allocatable, protected m_tetwt::n1b

Definition at line 14 of file m_tetwt.F.

3.9.3.5 integer, dimension(:,:,:), allocatable, protected m_tetwt::n2b

Definition at line 14 of file m_tetwt.F.

3.9.3.6 integer, dimension(:,:), allocatable, protected m_tetwt::nbnb

Definition at line 14 of file m_tetwt.F.

3.9.3.7 integer, protected m_tetwt::nbnbx

Definition at line 13 of file m_tetwt.F.

3.9.3.8 integer, dimension(:,:,:), allocatable, protected m_tetwt::nhw

Definition at line 12 of file m_tetwt.F.

3.9.3.9 integer, protected m_tetwt::nhwtot

Definition at line 13 of file m_tetwt.F.

3.9.3.10 real(8), dimension(:), allocatable, protected m_tetwt::whw

Definition at line 11 of file m_tetwt.F.

The documentation for this module was generated from the following file:

- gwsrc/m_tetwt.F

3.10 m_zmel Module Reference

Get the matrix element zmel = ZO^-1 <MPB psi|psi> , where ZO is ppovlz. To use this module, set data in this module, and call "call get_zmelt" or "call get_zmelt2". Then we have matrix elements zmel (exchange=F for correlation) or zmeltt (exchange=T). In future, they may be unified...

Public Member Functions

- subroutine get_zmelt (exchange, q, kx, kvec, irot, rkvec, kr, isp, ngc, ngb, nmmax, nqmax, nctot, ncc)
- subroutine get_zmelt2 (exchange,

Public Attributes

- integer, parameter null =-99999
- integer, dimension(:,:),
  allocatable miat
- real(8), dimension(:,:,:),
  allocatable tiat

- real(8), dimension(:,:), allocatable shtvg
- integer nband =NULL
- integer ngcmx =NULL
- integer ngpmx =NULL
- integer ntq =NULL
- integer, dimension(:), allocatable itq
- real(8), dimension(:,:,:), allocatable ppbir
- complex(8), dimension(:,:), allocatable, target ppovlz
- complex(8), dimension(:,:,:), allocatable zmel
- complex(8), dimension(:,:,:), allocatable zmeltt

Private Attributes

- real(8), dimension(3, 3), private qbasinv
- real(8), dimension(3), private q_bk =1d10
- real(8), dimension(3), private qk_bk =1d0
- logical, private init =.true.
- complex(8), dimension(:,:), allocatable, private cphiq
- complex(8), dimension(:,:), allocatable, private cphim
- real(8), dimension(:,:,:), allocatable, private rmelt
- real(8), dimension(:,:,:), allocatable, private cmelt
- integer, private kxold =-9999

3.10.1 Detailed Description

Get the matrix element zmel = ZO^-1 <MPB psi|psi> , where ZO is ppovlz. To use this module, set data in this module, and call "call get_zmelt" or "call get_zmelt2". Then we have matrix elements zmel (exchange=F for correlation) or zmeltt (exchange=T). In future, they may be unified...

Definition at line 5 of file m_zmel.F.

3.10.2 Member Function/Subroutine Documentation

3.10.2.1 subroutine m_zmel::get_zmelt (logical *exchange*, real(8), dimension(3) *q*, integer *kx*, real(8), dimension(3) *kvec*, integer *irot*, real(8), dimension(3) *rkvec*, integer *kr*, integer *isp*, integer *ngc*, integer *ngb*, integer *nmmax*, integer *nqmax*, integer *nctot*, integer *ncc*)

Definition at line 60 of file m_zmel.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.10.2.2 subroutine m_zmel::get_zmelt2 (logical *exchange*)

Definition at line 113 of file m_zmel.F.

Here is the call graph for this function:

Here is the caller graph for this function:

3.10.3 Member Data Documentation

3.10.3.1 real(8), dimension(:,:,:), allocatable, private m_zmel::cmelt `[private]`

Definition at line 55 of file m_zmel.F.

3.10.3.2 complex(8), dimension(:,:), allocatable, private m_zmel::cphim `[private]`

Definition at line 54 of file m_zmel.F.

3.10.3.3 complex(8), dimension(:,:), allocatable, private m_zmel::cphiq `[private]`

Definition at line 54 of file m_zmel.F.

3.10.3.4 logical, private m_zmel::init =.true. `[private]`

Definition at line 53 of file m_zmel.F.

3.10.3.5 integer, dimension(:), allocatable m_zmel::itq

Definition at line 43 of file m_zmel.F.

3.10.3.6 integer, private m_zmel::kxold =-9999 `[private]`

Definition at line 56 of file m_zmel.F.

3.10.3.7 integer, dimension(:,:), allocatable m_zmel::miat

Definition at line 39 of file m_zmel.F.

3.10.3.8 integer m_zmel::nband =NULL

Definition at line 42 of file m_zmel.F.

3.10.3.9 integer m_zmel::ngcmx =NULL

Definition at line 42 of file m_zmel.F.

3.10.3.10 integer m_zmel::ngpmx =NULL

Definition at line 42 of file m_zmel.F.

3.10.3.11 integer m_zmel::ntq =NULL

Definition at line 42 of file m_zmel.F.

3.10.3.12 integer, parameter m_zmel::null =-99999

Definition at line 37 of file m_zmel.F.

3.10.3.13 real(8), dimension(:,:,:), allocatable m_zmel::ppbir

Definition at line 44 of file m_zmel.F.

3.10.3.14 complex(8), dimension(:,:), allocatable, target m_zmel::ppovlz

Definition at line 45 of file m_zmel.F.

3.10.3.15 real(8), dimension(3), private m_zmel::q_bk =1d10 [private]

Definition at line 52 of file m_zmel.F.

3.10.3.16 real(8), dimension(3,3), private m_zmel::qbasinv [private]

Definition at line 52 of file m_zmel.F.

3.10.3.17 real(8), dimension(3), private m_zmel::qk_bk =1d0 [private]

Definition at line 52 of file m_zmel.F.

3.10.3.18 real(8), dimension(:,:,:), allocatable, private m_zmel::rmelt [private]

Definition at line 55 of file m_zmel.F.

3.10.3.19 real(8), dimension(:,:), allocatable m_zmel::shtvg

Definition at line 40 of file m_zmel.F.

3.10.3.20 real(8), dimension(:,:,:), allocatable m_zmel::tiat

Definition at line 40 of file m_zmel.F.

3.10.3.21 complex(8), dimension(:,:,:), allocatable m_zmel::zmel

Definition at line 49 of file m_zmel.F.

3.10.3.22 complex(8), dimension(:,:,:), allocatable m_zmel::zmeltt

Definition at line 49 of file m_zmel.F.

The documentation for this module was generated from the following file:

- gwsrc/m_zmel.F

Chapter 4

File Documentation

4.1 exec/makefile File Reference

Variables

- PLATFORM
- doxygen
- cd latex
- make echo fpgw latex refman pdf generated dep
- make echo fpgw latex refman pdf generated and read CallCaller sh echo echo Now generating a file callcaller dat Wait!It takes minute or so echo If you like to apply this to other programs
- make echo fpgw latex refman pdf generated and read CallCaller sh echo echo Now generating a file callcaller dat Wait!It takes minute or so echo If you like to apply this to other modify this script echo NOTE

4.1.1 Variable Documentation

4.1.1.1 make echo fpgw latex refman pdf generated dep

Definition at line 66 of file makefile.

4.1.1.2 doxygen

Definition at line 61 of file makefile.

4.1.1.3 cd latex

Definition at line 61 of file makefile.

4.1.1.4 make echo fpgw latex refman pdf generated and read CallCaller sh echo echo Now generating a file callcaller dat Wait ! It takes minute or so echo If you like to apply this to other modify this script echo NOTE

Definition at line 66 of file makefile.

4.1.1.5 PLATFORM

Definition at line 9 of file makefile.

4.1.1.6 make echo fpgw latex refman pdf generated and read CallCaller sh echo echo Now generating a file callcaller dat Wait ! It takes minute or so echo If you like to apply this to other programs

Definition at line 66 of file makefile.

4.2 makefile

```
00001 ### I think that you don’t needs to modify this file. ###
00002 ### This file is not machine-dependent. #####
00003 ### Machine dependence in make.inc
00004
00005
00006 # ---- Machine-specific compiler flags ---
00007 #include make.inc.ifort_asahi_kino
00008 #include make.inc.thinkpad_gfortran_tkotani
00009 PLATFORM=ifort
00010 LIBMATH=/usr/lib/x86_64-linux-gnu/libfftw3.so.3 /usr/lib/liblapack.so.3gf /usr/lib/libblas.so.3gf
00011
00012 #PLATFORM=ifort
00013 #LIBMATH=-mkl
00014
00015 include make.inc.$(PLATFORM)
00016
00017 BINDIR = $(HOME)/bin
00018
00019 #-------------------------------------------------------
00020 # src directories
00021 main    = ../main/
00022 gwsrc   = ../gwsrc/
00023 tote = ../tote/
00024 tags   = ../
00025
00026 #maxloc = ../Miyake/maxloc/
00027 # tag directory
00028 #
00029 #progs  = hbasfp0 hvccfp0 hx0fp0 hsfp0 hef hqpe hchknw qg4gw gwinit heftet hmergewv hparainfo hbndout
       rdata4gw_v2 convgwin hx0fp0_sc hsfp0_sc hqpe_sc kino_input_test hecor eout eout2 h_uumatrix hsigmconv
00030 # lmf_exec
00031 #progs  = hbasfp0 hvccfp0 hx0fp0 hsfp0 hef hqpe hchknw qg4gw gwinit heftet hmergewv hparainfo hbndout
       rdata4gw_v2  hx0fp0_fal hx0fp1
00032
00033 progs  = hbasfp0 hvccfp0 hx0fp0 hsfp0 hef hqpe hqpe_qsgw qg4gw gwinit heftet hmergewv
      rdata4gw_v2 convgwin hx0fp0_sc hsfp0_sc hqpe_sc kino_input_test hecor eout eout2
00034
00035 # progs  = hbasfp0 hvccfp0 hx0fp0 hsfp0 hef hqpe hchknw qg4gw gwinit heftet hmergewv hbndout rdata4gw_v2
       convgwin hx0fp0_sc hsfp0_sc hqpe_sc kino_input_test hecor eout eout2 h_uumatrix hsigmconv hwmat hmaxloc huumat
       qpwf hpsig hnocc_mlw hx0fp0_mlw hphig
00036
00037 # hmaxloc1D
00038 progs2 = $(progs) $(tags)TAGS
00039 #checkmod
00040
00041 #script = cleargw* dqpu dtote eps* ex* gw* hqpemetal* inf* lmgw* plotg save* tote_lmfh2 xqp mkG*
00042 script = cleargw* dqpu eps* gw* mkG*
00043
00044 #### You can choose these options. all is default.
00045
00046 all :$(progs2)
00047
00048 clean:
00049          rm -f  $(progs)
00050
00051 install:
00052          cp  $(progs)  $(BINDIR)
00053
00054 install2:
```

```
         cp  $(script) $(BINDIR)

cleanall:
         rm -f  $(progs2) $(main)*.o $(gwsrc)*.o  *.mod  $(tote)*.o

doxygen:
        cd $(tags);doxygen;cd ./latex;make
        echo 'fpgw/latex/refman.pdf generated'

dep:
        @echo  'This generate a call-caller data set for fpgw/'
        @echo  'HELP --> ../TOOLS/FparserTools/f_calltree.py --help, and read CallCaller.sh'
        @echo
        @echo  '--- Now generating a file 'callcaller.dat' ... Wait!!! It takes 1 minute or so!'
        @echo '        If you like to apply this to other programs, modify this script'
        @echo  ' NOTE: T.Kotani is not sure whether this is relaiable enough or not... let me know something wrong...'
        $(tags)/../TOOLS/FparserTools/f_calltree.py $(main)/*.F $(gwsrc)/*.F $(tote)/*.F >callcaller.dat 2>callcaller.err
        -egrep -e '^(ERROR|Error)' callcaller.err
        @echo
        @echo '-------------------------------------------------------------------------------'
        @echo '--- If no ERROR is shown above (if ERROR is not in callcaller.err), it is succeeded. ---'
        @echo '       Note that Unsed files might be used by other mainprogram.'
        @echo '--- If ERROR is shown above, look into callcaller.err. Something wrong.'
        @echo
        @echo ' If you want to make a callcaller-tree picture, try'
        @echo ' >GenCCtree.sh callcaller.dotdata'
        @echo ' --> Then you get ccmap.ps.; it is better to use smaller callcaller.dotdata(need to modify this script to make it).'
        @echo ' Note that you need graphviz for GenCCtree.sh. as apt-get install graphviz'

# This is necesaly to compile *.f in right order.
# When you recompile and link, just repeat 'make' (not necessary to repeat 'make init').
# When checkmodule recompile source, you have to repeat 'make'.
init:
        rm -f $(main)time_hsfp0.sc.m.F
        rm -f $(main)time_hx0fp0.sc.m.F
        rm -f $(gwsrc)time_sxcf_fal2.sc.F
        rm -f $(gwsrc)time_rppovl.F
        rm -f $(gwsrc)time_x0kf_v4h.F
        rm -f $(gwsrc)time_ppbafp.fal.F
        exec ../../TOOLS/checkmodule ../gwsrc/*.F ../main/*.F ../tote/*.F

checkmod:
        init
#../../lm7K/subs/m_hamindex.F
# m_hamindex

## tete (total energy)  ###############################
## these are experimental code ###
ECOR = \
$(tote)hecor.o

EO= \
$(tote)eout.o \

EO2= \
$(tote)eout2.o

hecor: $(ECOR) $(GWLIB)   $(MPI)  $(GWLIB) $(COMM)
        $(LK) $(LKFLAGS1) $(ECOR) $(GWLIB) $(MPI)  $(COMM) $(LKFLAGS2) -o $@

eout: $(EO)   $(GWLIB) $(MPI)
        $(LK) $(LKFLAGS1) $(EO) $(GWLIB) $(MPI) $(LKFLAGS2) -o $@

eout2: $(EO2)  $(GWLIB) $(MPI)
        $(LK) $(LKFLAGS1) $(EO2) $(GWLIB) $(MPI) $(LKFLAGS2) -o $@
########################################################


# BNDCONN= \
# $(gwsrc)bndconn.o   ### This is not linked but bndconn.o is used in lm/lmfgw.
# It is now included in lm/gw/
DERFC=
# $(gwsrc)derfc.o
#          $(gwsrc)d1mach.o \
#          $(gwsrc)i1mach.o
#
# test_genallcf =  \
# $(main)test_genallcf.o \
# $(gwsrc)genallcf_dump.o \
# $(GWLIB)


 kino_input_test =  \
 $(main)kino_input_test.o
```

```

convg =  \
$(main)convgwin.o

GWINIT =  \
$(main)gwinit.m.o

QG =  \
$(main)qg4gw.m.o

RDAT_v2 = \
$(main)rdata4gw_v2.m.o

BAS = \
$(main)hbasfp0.m.o

VCC= \
$(main)hvccfp0.m.o

SXC_SC = \
$(main)hsfp0.sc.m.o

SXC = \
$(main)hsfp0.m.o

# WMAT = \
# $(maxloc)hwmat.o \
# $(maxloc)maxloc0.o \
# $(maxloc)wmat.o

# MLOC = \
# $(maxloc)hmaxloc.o \
# $(maxloc)maxloc0.o \
# $(maxloc)maxloc1.o \
# $(maxloc)maxloc2.o \
# $(maxloc)maxloc3.o

# MLOC1D = \
# $(maxloc)hmaxloc1D.o \
# $(maxloc)maxloc0.o \
# $(maxloc)maxloc1.o \
# $(maxloc)maxloc2.o \
# $(maxloc)maxloc3.o

heftet = \
$(main)heftet.m.o

# hnocc_mlw = \
# $(maxloc)hnocc_mlw.o

hef = \
$(main)hef.m.o


X0_SC = \
$(main)hx0fp0.sc.m.o

X0 = \
$(main)hx0fp0.m.o

# X0mlw = \
# $(maxloc)hx0fp0.m.o \
# $(maxloc)wcf.o \
# $(gwsrc)tetwt5$(tet5_g) \
# $(gwsrc)m_tetwt.o \
# $(gwsrc)diagcv2.o \
# $(tote)rpaq.o \
# $(gwsrc)cinvrx.o\
# $(gwsrc)m_freq.o
#
# UU = \
# $(main)h_uumatrix.m.o \
# $(gwsrc)wcf.o \
# $(gwsrc)tetwt5$(tet5_g) \
# $(gwsrc)gintxx.o \
# $(gwsrc)pplmat.o \
# $(gwsrc)getgv2.o \
# $(gwsrc)x0kf_v4h$(x0kf_g) \
# $(gwsrc)rs.o \
# $(gwsrc)u_lat_0.o \
# $(gwsrc)wronkj.o \
# $(gwsrc)mklegw.o \
# $(gwsrc)bessl.o \
# $(gwsrc)cross.o \
# $(gwsrc)diagcv2.o
#
# UU2 = \
# $(maxloc)huumat.o \
# $(gwsrc)wcf.o \
# $(gwsrc)tetwt5$(tet5_g) \
# $(gwsrc)gintxx.o \
# $(gwsrc)pplmat.o \
# $(gwsrc)getgv2.o \
# $(gwsrc)rs.o \
# $(gwsrc)u_lat_0.o \
# $(gwsrc)wronkj.o \
# $(gwsrc)mklegw.o \
# $(gwsrc)bessl.o \
# $(gwsrc)cross.o
#
# PSIG = \
# $(maxloc)hpsig.o \
# $(gwsrc)wcf.o \
# $(gwsrc)tetwt5$(tet5_g) \
# $(gwsrc)m_tetwt.o \
# $(gwsrc)gintxx.o \
# $(gwsrc)pplmat.o \
# $(gwsrc)getgv2.o \
# $(gwsrc)rs.o \
# $(gwsrc)u_lat_0.o \
# $(gwsrc)wronkj.o \
# $(gwsrc)mklegw.o \
# $(gwsrc)bessl.o \
# $(gwsrc)cross.o
#
# PHIG = \
# $(maxloc)hphig.o \
# $(gwsrc)wcf.o \
# $(gwsrc)tetwt5$(tet5_g) \
# $(gwsrc)m_tetwt.o \
# $(gwsrc)gintxx.o \
# $(gwsrc)pplmat.o \
# $(gwsrc)getgv2.o \
# $(gwsrc)rs.o \
# $(gwsrc)u_lat_0.o \
# $(gwsrc)wronkj.o \
# $(gwsrc)mklegw.o \
# $(gwsrc)bessl.o \
# $(gwsrc)cross.o

MPI =  $(gwsrc)MPI_fpgw2.o

GWLIB =  \
$(gwsrc)m_w0w0i.o \
$(gwsrc)getwemax.o \
$(gwsrc)genallcf_dump.o \
$(gwsrc)wse.o \
$(gwsrc)bzints2.o \
$(gwsrc)wintzsg.o \
$(gwsrc)gintxx.o \
$(gwsrc)gwinput_v2.o \
$(gwsrc)pplmat.o \
$(gwsrc)rs.o \
$(gwsrc)conv2gwinput.o \
$(gwsrc)getbzdata1.o \
$(gwsrc)getgv2.o \
$(gwsrc)wcf.o \
$(gwsrc)tetwt5$(tet5_g) \
$(gwsrc)m_tetwt.o \
$(gwsrc)x0kf_v4h$(x0kf_g) \
$(gwsrc)cinvrx.o \
$(gwsrc)zsvd.o \
$(gwsrc)m_zmel.o \
$(gwsrc)m_freq.o \
$(gwsrc)m_hamindex.o\
$(gwsrc)readpomat.o \
$(gwsrc)keyvalue.o \
$(gwsrc)rppovl.o \
$(gwsrc)nocctotg.o \
$(gwsrc)ppbafp.fal$(para_g) \
$(gwsrc)psi2b_v2$(para_g) \
$(gwsrc)psi2b_v3$(para_g) \
$(gwsrc)wfacx.o \
$(gwsrc)sortea.o \
$(gwsrc)rydberg.o \
$(gwsrc)polinta.o \
$(gwsrc)efsimplef.o \
$(gwsrc)extension.o \
$(gwsrc)rangedq.o \
$(gwsrc)nword.o \
$(gwsrc)scg.o \
$(gwsrc)matm.o \
$(gwsrc)rdpp.o \
$(gwsrc)mptauof.o \
 $(gwsrc)genallcf_mod.o \
 $(gwsrc)rgwinf_mod.o \
 $(gwsrc)rotdlmm.o \
 $(gwsrc)iopen.o \
 $(gwsrc)cputid.o \
 $(gwsrc)rw.o \
 $(gwsrc)ext.o \
 $(gwsrc)ext2.o \
 $(gwsrc)cross.o \
 $(gwsrc)mate.o \
 $(gwsrc)mate1.o \
 $(gwsrc)icopy.o \
 $(gwsrc)bib1.o \
 $(gwsrc)index.o \
 $(gwsrc)idxk.o \
 $(gwsrc)maxnn.o \
 $(gwsrc)reindx.o \
 $(gwsrc)iprint.o \
 $(gwsrc)bz.o \
 $(gwsrc)bzmesh.o \
 $(gwsrc)genqbz.o \
 $(gwsrc)switches.o \
 $(gwsrc)rwbzdata.o \
 $(gwsrc)llnew.o  \
 $(gwsrc)readeigen.o \
 $(gwsrc)readqg.o \
 $(gwsrc)iqindx.o  \
 $(gwsrc)alloclist.o \
 $(gwsrc)m_pkm4crpa.o \
 $(gwsrc)m_anf.o \
 $(gwsrc)qpe1.sc.o \
 $(gwsrc)icompvv2.o \
 $(gwsrc)iopenxx.o \
 $(gwsrc)qpe1.o \
 $(gwsrc)mopen.o \
 $(gwsrc)checksymlon.o \
 $(gwsrc)mkqg.o \
 $(gwsrc)m_q0p.o \
 $(gwsrc)q0irre.o \
 $(gwsrc)basnfp.o \
 $(gwsrc)excore.o \
 $(gwsrc)mkjp.o \
 $(gwsrc)strxq.o \
 $(gwsrc)sxcf_fal2.sc$(sxcf_g) \
 $(gwsrc)sxcf_fal2$(sxcf_g) \
$(gwsrc)amix.o \
$(gwsrc)dsifa.o \
$(gwsrc)dsisl.o \
$(gwsrc)dsidi.o \
$(gwsrc)diagcv2.o \
$(gwsrc)wronkj.o \
$(gwsrc)rxx.o \
$(gwsrc)hsmq.o \
$(gwsrc)u_lat_0.o \
$(gwsrc)mklegw.o \
$(gwsrc)bessl.o \
$(gwsrc)lgen.o \
$(gwsrc)hansr5.o \
$(gwsrc)hansr4.o \
$(gwsrc)lattc.o \
$(gwsrc)qdist.o \
$(gwsrc)dlmtor.o \
$(gwsrc)dpcopy.o \
$(gwsrc)dpadd.o \
$(gwsrc)dpzero.o \
$(gwsrc)ropyln.o \
$(gwsrc)ropcsm.o \
$(gwsrc)rpaq.o \
$(gwsrc)m_readeps.o

 QPE_QSGW = \
 $(main)hqpe_qsgw.m.o\
 $(gwsrc)qpe1.qsgw.o

 QPE_SC = \
 $(main)hqpe.sc.m$(hqpe_g)

 QPE = \
 $(main)hqpe.m$(hqpe_g)

 MERGE = \
  $(main)hmergewv.m.o

 # PARAINFO = \
 # $(main)hparainfo.m.o \
 # $(gwsrc)charext.o

 # BNDOUT = \
 # $(main)hbndout.m.o \
 # $(gwsrc)iqagree.o \
 # $(gwsrc)iopenxx.o \
 # $(gwsrc)iopen.o \
 # $(gwsrc)polinta.o \
 # $(gwsrc)rydberg.o \
 # $(gwsrc)extension.o \
 # $(gwsrc)rangedq.o \
 # $(gwsrc)switches.o \
 # $(gwsrc)keyvalue.o
 #
#         $(gwsrc)setpr.o \
        # $(gwsrc)sylm.o \
        # $(gwsrc)sylmnc.o \
 # SIGMCONV = \
 # $(gwsrc)switches.o \
 # $(gwsrc)keyvalue.o \
 # $(gwsrc)iopen.o \
 # $(main)hsigmconv.m.o

###########################################

# bndconn.o:    $(BNDCONN)
#
############### dependency for use ##################



 # hsigmconv:   $(SIGMCONV)  $(MPI)  $(COMM)
 #       $(LK) $(LKFLAGS1) $(SIGMCONV) $(MPI)  $(COMM) $(LKFLAGS2) -o $@


 gwinit:        $(GWINIT)  $(MPI)  $(GWLIB)
         $(LK) $(LKFLAGS1) $(GWINIT) $(MPI) $(GWLIB)  $(LKFLAGS2) -o $@


 # qpwf:                $(maxloc)qpwf.o $(GWLIB) $(MPI)  $(COMM)
 #      $(LK) $(LKFLAGS1) $(maxloc)qpwf.o $(GWLIB) $(MPI) $(COMM)  $(LKFLAGS2) -o $@

 qg4gw:         $(QG)  $(MPI) $(GWLIB) $(COMM)
         $(LK) $(LKFLAGS1) $(QG) $(MPI) $(GWLIB) $(COMM)    $(LKFLAGS2) -o $@

 rdata4gw_v2:   $(RDAT_v2)  $(MPI)  $(COMM) $(GWLIB)
         $(LK) $(LKFLAGS1) $(RDAT_v2) $(MPI) $(COMM)  $(GWLIB) $(LKFLAGS2) -o $@

 hbasfp0:       $(BAS)  $(MPI)  $(COMM) $(GWLIB)
         $(LK) $(LKFLAGS1) $(BAS) $(MPI)  $(COMM)  $(GWLIB) $(LKFLAGS2) -o $@

 hvccfp0:       $(MPI) $(VCC)   $(DERFC) $(MPI)  $(COMM) $(GWLIB)
         $(LK) $(LKFLAGS1) $(VCC) $(DERFC) $(MPI)  $(COMM) $(GWLIB) $(LKFLAGS2) -o $@

 hx0fp0:        $(MPI) $(X0) $(GWLIB) $(MPI)  $(COMM)
         $(LK) $(LKFLAGS1) $(X0)     $(GWLIB) $(MPI)  $(COMM) $(LKFLAGS2) -o $@

# # for maxloc
#  hx0fp0_mlw:  $(X0mlw) $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(X0mlw)     $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  h_uumatrix:  $(UU) $(GWLIB)   $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(UU)     $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  huumat:      $(UU2) $(GWLIB) $(MPI) $(COMM)
#       $(LK) $(LKFLAGS1) $(UU2)  $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  hphig:       $(PHIG) $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(PHIG)     $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) $(LIBSLA) -o $@

#  hpsig: $(PSIG) $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(PSIG)     $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

 hx0fp0_sc:     $(MPI) $(X0_SC) $(GWLIB)   $(MPI)  $(COMM)
         $(LK) $(LKFLAGS1) $(X0_SC)     $(GWLIB) $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  hwmat:               $(WMAT) $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(WMAT)    $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  hmaxloc:     $(MLOC)   $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(MLOC)  $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

#  hmaxloc1D:   $(MLOC1D)   $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(MLOC1D)  $(GWLIB)   $(MPI)  $(COMM) $(LKFLAGS2) -o $@

 hsfp0:         $(MPI) $(SXC) $(GWLIB)   $(MPI)  $(COMM)
         $(LK) $(LKFLAGS1) $(SXC)    $(GWLIB) $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

 hsfp0_sc:      $(MPI) $(SXC_SC) $(GWLIB)   $(MPI)  $(COMM)
```

```
        $(LK) $(LKFLAGS1) $(SXC_SC)    $(GWLIB)  $(MPI)  $(COMM) $(LKFLAGS2) -o $@

#  hnocc_mlw:   $(hnocc_mlw) $(GWLIB) $(MPI)  $(COMM)
#       $(LK) $(LKFLAGS1) $(hnocc_mlw) $(GWLIB)  $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

 heftet:        $(heftet) $(GWLIB)  $(MPI) $(MPI)  $(COMM)
        $(LK) $(LKFLAGS1) $(heftet) $(GWLIB)  $(MPI)  $(COMM) $(LKFLAGS2) -o $@

 hef:           $(hef) $(GWLIB)   $(MPI) $(COMM)
        $(LK) $(LKFLAGS1) $(hef)    $(GWLIB)  $(MPI)  $(COMM) $(LKFLAGS2) -o $@


 hqpe:          $(QPE) $(MPI) $(COMM) $(GWLIB)
        $(LK) $(LKFLAGS1) $(QPE) $(MPI) $(COMM) $(GWLIB) $(LKFLAGS2) -o $@

 hqpe_sc:               $(QPE_SC)  $(MPI) $(COMM) $(GWLIB)
        $(LK) $(LKFLAGS1) $(QPE_SC) $(GWLIB) $(MPI)  $(COMM) $(LKFLAGS2) -o $@

 hqpe_qsgw:             $(QPE_QSGW)  $(GWLIB) $(MPI) $(COMM)
        $(LK) $(LKFLAGS1) $(QPE_QSGW) $(GWLIB) $(MPI)  $(COMM) $(LKFLAGS2) -o $@

 hmergewv:      $(MERGE) $(MPI) $(GWLIB)  $(COMM)
        $(LK) $(LKFLAGS1) $(MERGE) $(GWLIB) $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

 # hparainfo:   $(PARAINFO) $(GWLIB) $(MPI)   $(COMM)
 #      $(LK) $(LKFLAGS1) $(PARAINFO) $(GWLIB)  $(MPI)  $(COMM) $(LKFLAGS2) -o $@

# hbndout:      $(BNDOUT) $(MPI)   $(COMM)
#       $(LK) $(LKFLAGS1) $(BNDOUT) $(MPI)  $(COMM)  $(LKFLAGS2) -o $@

 convgwin:      $(convg)
        $(LK) $(LKFLAGS1) $(convg) $(LKFLAGS2) -o $@

 kino_input_test:       $(kino_input_test) $(GWLIB) $(MPI)  $(COMM)
        $(LK) $(LKFLAGS1) $(kino_input_test) $(GWLIB)  $(MPI)  $(COMM) $(LKFLAGS2) -o $@

################################ test
#
# test_genallcf:        $(test_genallcf)
#       $(LK) $(LKFLAGS1) $(test_genallcf) $(LKFLAGS2) -o $@


 $(tags)TAGS: $(progs)
       cd $(tags);etags ./*/*/*.F ./*/*.F


# --- Make rules ---
.SUFFIXES:
.SUFFIXES: .F .o
#.SUFFIXES: .f .o .c1_o .c2_0 .c3_o .c4_o .F

.F.o:
       $(FC) $(FFLAGS) $*.F -c -o $*.o
#      etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

#.F.o:
#      $(FC) $(FFLAGS) $*.F -c -o $*.o
#      etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

#.f.o:
#      $(FC) $(FFLAGS) $*.f -c -o $*.o
#      etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

.f.c1_o:
       $(FC) $(FFLAGS_c1) $*.f -c -o $*.c1_o
       etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

.f.c2_o:
       $(FC) $(FFLAGS_c2) $*.f -c -o $*.c2_o
       etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

.f.c3_o:
       $(FC) $(FFLAGS_c3) $*.f -c -o $*.c3_o
       etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags

.f.c4_o:
       $(FC) $(FFLAGS_c4) $*.f -c -o $*.c4_o
       etags $*.f -o $(tags)‘echo $*.f| sed ’s/..\///’ | sed ’s/\//-/g’‘.tags


check:
       (cd ../TESTinstallGW;./testgw.py --enforce --all)

# test for f90 dependency
#../main/hvccfp0.m.o    :       ../main/hx0fp0.m.o
#
#../main/hvccfp0.m.o    :       ../main/hbasfp0.m.o
```

```

include moduledepends.inc


################################################################################
##### You can comment out these blocks to commnet out memory and time check (verbose output)
addtime=script/addtime.awk
septhen=script/then_separate.awk
alloclist=script/add_alloclist.awk
$(main)hsfp0.sc.m.o:  $(main)hsfp0.sc.m.F
        gawk -f $(addtime) -vSTART=1 $(main)hsfp0.sc.m.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(main)time_hsfp0.sc.m.F
        $(FC) $(FFLAGS) $(main)time_hsfp0.sc.m.F -c -o $*.o

$(main)hx0fp0.sc.m.o: $(main)hx0fp0.sc.m.F
        gawk -f $(addtime) -vSTART=1 $(main)hx0fp0.sc.m.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(main)time_hx0fp0.sc.m.F
        $(FC) $(FFLAGS) $(main)time_hx0fp0.sc.m.F -c -o $*.o

$(gwsrc)sxcf_fal2.sc$(sxcf_g): $(gwsrc)sxcf_fal2.sc.F
        gawk -f $(addtime) -vSTART=100 $(gwsrc)sxcf_fal2.sc.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(gwsrc)time_sxcf_fal2.sc.F
        $(FC) $(FFLAGS) $(gwsrc)time_sxcf_fal2.sc.F -c -o $*.o

#$(gwsrc)rppovl.o: $(gwsrc)rppovl.F
#       gawk -f $(addtime) -vSTART=200 $(gwsrc)rppovl.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(gwsrc)time_rppovl.F
#       $(FC) $(FFLAGS) $(gwsrc)time_rppovl.F -c -o $*.o

$(gwsrc)x0kf_v4h$(x0kf_g): $(gwsrc)x0kf_v4h.F
        gawk -f $(addtime) -vSTART=100 $(gwsrc)x0kf_v4h.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(gwsrc)time_x0kf_v4h.F
        $(FC) $(FFLAGS) $(gwsrc)time_x0kf_v4h.F -c -o $*.o

$(gwsrc)ppbafp.fal$(para_g): $(gwsrc)ppbafp.fal.F
        gawk -f $(addtime) -vSTART=300 $(gwsrc)ppbafp.fal.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(gwsrc)time_ppbafp.fal.F
        $(FC) $(FFLAGS) $(gwsrc)time_ppbafp.fal.F -c -o $*.o
#$(gwsrc)ppbafp.fal$(para_g): $(gwsrc)ppbafp.fal.F
#       gawk -f $(addtime) -vSTART=300 $(gwsrc)ppbafp.fal.F | gawk -f $(septhen) | gawk -f $(alloclist) > $(gwsrc)time_ppbafp.fal.F
#       $(FC) $(FFLAGS) $(gwsrc)time_ppbafp.fal.F -c -o $*.o
###################################################################################


$(gwsrc)wintzsg.o :  $(gwsrc)wintzsg.F
        $(FC)  $(FFLAGS) $(gwsrc)wintzsg.F  -c -o $*.o


# DO NOT DELETE
```

4.3 gwsrc/genallcf_mod.F File Reference

Data Types

- module m_readefermi
- module m_genallcf_v3

  get basic settings of crystal structure and nlm info

Functions/Subroutines

- subroutine idxlnmc (nindxv, nindxc, nl, nn, nnv, nnc, nlnmx, nlnmxv, nlnmxc, nclass, il, in, im, ilnm, ilv, inv, imv, ilnmv, ilc, inc, imc, ilnmc)
- integer function noflmto (nindx, iclass, nl, nclass, natom)
- integer function nalwln (nocc, nunocc, nindx, nl, nn)
- integer function nofln (nindx, nl)
- integer function noflnm (nindx, nl)
- integer function nallow (nocc, nunocc, nindx, nl, nn)
- subroutine incor (ncwf, nindxc, iclass, nl, nnc, nclass, natom, icore, ncore, nctot)

4.3.1 Function/Subroutine Documentation

4.3.1.1 subroutine idxlnmc (dimension(0:nl-1,nclass) *nindxv*, dimension(0:nl-1,nclass) *nindxc*, *nl*, *nn*, *nnv*, *nnc*, *nlnmx*, *nlnmxv*, *nlnmxc*, *nclass*, dimension(nlnmx,nclass) *il*, dimension(nlnmx,nclass) *in*, dimension(nlnmx,nclass) *im*, dimension(nn,nl∗nl,nclass) *ilnm*, dimension(nlnmxv,nclass) *ilv*, dimension(nlnmxv,nclass) *inv*, dimension(nlnmxv,nclass) *imv*, dimension(nnv,nl∗nl,nclass) *ilnmv*, dimension(nlnmxc,nclass) *ilc*, dimension(nlnmxc,nclass) *inc*, dimension(nlnmxc,nclass) *imc*, dimension(nnc,nl∗nl,nclass) *ilnmc*)

Definition at line 392 of file genallcf_mod.F.

Here is the caller graph for this function:

4.3.1.2 subroutine incor (dimension(0:nl-1,nnc,nclass) *ncwf*, dimension(0:nl-1,nclass) *nindxc*, dimension(natom) *iclass*, *nl*, *nnc*, *nclass*, *natom*, dimension(nl∗nl∗nnc,nclass) *icore*, dimension(nclass) *ncore*, *nctot*)

Definition at line 568 of file genallcf_mod.F.

Here is the caller graph for this function:

4.3.1.3 integer function nallow (dimension(0:nl-1,nn) *nocc*, dimension(0:nl-1,nn) *nunocc*, dimension(0:nl-1) *nindx*, *nl*, *nn*)

Definition at line 526 of file genallcf_mod.F.

4.3.1.4 integer function nalwln (dimension(0:nl-1,nn) *nocc*, dimension(0:nl-1,nn) *nunocc*, dimension(0:nl-1) *nindx*, *nl*, *nn*)

Definition at line 475 of file genallcf_mod.F.

4.3.1.5 integer function noflmto (dimension(0:nl-1,nclass) *nindx*, dimension(natom) *iclass*, *nl*, *nclass*, *natom*)

Definition at line 462 of file genallcf_mod.F.

Here is the caller graph for this function:

4.3.1.6 integer function nofln (dimension(0:nl-1) *nindx*, *nl*)

Definition at line 504 of file genallcf_mod.F.

4.3.1.7 integer function noflnm (dimension(0:nl-1) *nindx*, *nl*)

Definition at line 515 of file genallcf_mod.F.

4.4 genallcf_mod.F

```
      module m_readefermi
      real(8),protected:: bandgap
      real(8)::ef
      contains

      subroutine readefermi()
      implicit none
      integer:: ifief,ifile_handle
      ifief=ifile_handle()
      open(ifief,file='EFERMI')
      read(ifief,*) ef,bandgap
      close(ifief)
      write(6,"(a,f12.6)")' --- READIN ef from EFERMI. ef=',ef
      end subroutine
      end module m_readefermi

!> get basic settings of crystal structure and nlm info
!!  - genallcf_v3(nwin,efin,incwfx) set data
!!  - This is old routine. Confusing. We need to clean up.
      module m_genallcf_v3
!!----------------------------------------------------------------
!! - structure
!!  -  o                   plat,alat,natom,nclass,pos,
!!  -  o                   ngrp, symgg,
!!  -  o                   invg, ef,
!! - l,n and dimensions
!!   - o                   clabl, nspin,nl,nn,nnv,nnc,
!!   - o                   nindx, nindxv, nindxc, iclass,
!!   - d                   nlmto,nlnx,nlnxv,nlnxc,nlnmx,nlnmxv,nlnmxc,
!!   - o                   z,
!! - l,n,m indices for Phi (atomic basis)
!!   -  o                   il, in, im,  ilnm,  nlnm,
!!   -  o                   ilv,inv,imv, ilnmv, nlnmv,
!!   -  o                   ilc,inc,imc, ilnmc, nlnmc,
!! - core
!!   -  o                   ncwf, ecore, konf, icore, ncore,nctot,
!! - frequency
!!   -                   niw,diw,nw,dw,delta,deltaw,esmr, freq)
!!             symgrp
!!             ,nocc, nunocc, occv, unoccv, occc, unoccc
      implicit none
      character(120),protected:: symgrp
      character(6),allocatable,protected :: clabl(:)
      integer,allocatable,protected:: iclass(:),
     &    nindxv(:,:),nindxc(:,:),ncwf(:,:,:) ,
     o    invg(:), il(:,:), in(:,:), im(:,:),   ilnm(:),  nlnm(:),
     o    ilv(:),inv(:),imv(:),  ilnmv(:), nlnmv(:),
     o    ilc(:),inc(:),imc(:),  ilnmc(:), nlnmc(:),
     o    nindx(:,:),konf(:,:),icore(:,:),ncore(:),
     &    occv(:,:,:),unoccv(:,:,:)
     &   ,occc(:,:,:),unoccc(:,:,:),
     o    nocc(:,:,:),nunocc(:,:,:),   iantiferro(:)
      integer,protected::
     o  nclass,natom,nspin,nl,nn,nnv,nnc, ngrp,
     o  nlmto,nlnx,nlnxv,nlnxc,nlnmx,nlnmxv,nlnmxc, nctot!,nw
      real(8), allocatable,protected::
     o  plat(:,:),pos(:,:),z(:),  symgg(:,:,:)   !w(igrp) freq(:),
      real(8),protected :: alat,deltaw !ef,,diw,dw
      logical,protected:: done_genallcf_v3=.false.
      character(8),allocatable,protected:: spid(:)
!! unprotected --> need to be protected
      real(8), allocatable :: ecore(:,:)
      real(8):: delta
      integer:: niw
      real(8):: esmr
c-----------------------------------------------------------------------
      contains

      subroutine genallcf_v3(incwfx)
!!> Readin GWIN_V2 and LMTO(crystal) data and allocate all required.
!!r Return iclass=ibas.
!! efin,incwfx, are used as switches.
!! input: efin,incwfx,
!!        GWIN_V2, LMTO
!! output: All the output are given in the declear section above.
!! --------------------------------------------------------
      implicit none
      integer(4)::iflmto,ifinin,incwfx,ifec,i,j,
     & lmx, lmx2,nlmto2,nprodxc,nlnaxc,nlnaxv,nprodx,ifi,ig,is
     & ,iopen,iclose,nprodxv,nlnax
     & ,noflmto,maxnn
      integer(4):: infwfx
      integer(4):: n1,n2,n3,imagw,lcutmx,n,ic
      logical :: nocore
      real(8)::efin
      real(8),allocatable::tolbas(:)
      character(120):: symgrpt
      real(8),   allocatable:: ecoret(:,:,:,:)
      integer(4),allocatable::ncwf2(:,:,:)
      integer:: ia,l,m,ic1,isp,lt,nt,nsp,nr,ncorex,ifix
      real(8)::a,b,zz, efdummy,dw,diw
      integer:: nwdummy
c      allocate(nclass,natom,nspin,nl,nn,nnv,nnc, ngrp,
c     o  nlmto,nlnx,nlnxv,nlnxc,nlnmx,nlnmxv,nlnmxc, nctot, niw,nw)
      if(done_genallcf_v3) call rx('genallcf_v3 is already called')
      done_genallcf_v3=.true.

c      allocate(alat,ef, diw,dw,delta,deltaw,esmr,  symgrp)
      iflmto     = iopen('LMTO',1,0,0)
      if (iflmto < 0) call rx( 'unit file for GWIN_V2 < 0')

c--- readin these by rgwinf_v3
c      character*120 symgrp
c      integer(4)::nclass,natom,nspin,nl,nnv,nnc
c      real(8)::alat
c      integer(4),allocatable::
c     &   iclass(:)
c     &  ,nindxv(:,:),nindxc(:,:)
c     &  ,occv(:,:,:),unoccv(:,:,:)
c     &  ,occc(:,:,:),unoccc(:,:,:)
c     &  ,ncwf(:,:,:)
c      real(8),allocatable:: plat(:,:),pos(:,:),z(:)
c      character*6,allocatable:: clabl(:)
c      write(6,*)' goto rgwin'
c      call rgwinf_v3 (iflmto,ifinin,nwin,efin,incwfx) !these are inputs
c      write(6,*)' end of rgwinf_v3'
c--------------------------------------------------------------------------
c--- rgwinf ---
      ifi = iflmto
c      nw  = nwin
c      ef  = efin
      read(ifi,*);  read(ifi,*)
      read(ifi,*)symgrpt        !SYMMETRY
      j = 0
      symgrp='  '//trim(adjustl(symgrpt))
      write(6,*)' symgrp=', symgrp
      read(ifi,*)
      read(ifi,*)
      read(ifi,*)
      read(ifi,*)alat              !lattice constant
      allocate(plat(3,3))          !primitive lattice vectors
      read(ifi,*)
      read(ifi,*)plat(1:3,1)
      read(ifi,*)plat(1:3,2)
      read(ifi,*)plat(1:3,3)
      read(ifi,*)
      read(ifi,*) natom            !Number of atoms
!!
      nclass = natom  !We set nclass = natom through the GW calculations
      write(6,*)'genalloc: alat natom=',alat,natom
      allocate(pos(3,natom))       !positions of atoms
      read(ifi,*)
      do n = 1,natom
        read(ifi,*) pos(1,n),pos(2,n),pos(3,n)
      end do
      read(ifi,*)
      read(ifi,*)
      read(ifi,*)
      read(ifi,*)nspin           !spin (1=paramagnetic  2=ferromagnetic)
      read(ifi,*)
      read(ifi,*)nl              !max. no. valence and core l
      read(ifi,*)
      read(ifi,*)nnv,nnc   !max. no. valence and core n
      write(6,*)' nspin nl nnv nnc =',nspin,nl,nnv,nnc
c--------------------------
      if(nnv==1) nnv=2 ! for backword compatibility!takao apr 2002
      ! nnv=2 corresponds to phi and phidot
      ! nnv=3 corresponds to
c--------------------------
      read(ifi,*)
      read(ifi,*)  !nrx is not readin
      read(ifi,*)
      allocate(clabl(nclass),z(nclass)) !class-label, z
      do ic = 1,nclass
        read(ifi,*) clabl(ic),z(ic) !,nrofi is not readin
      end do

      allocate(iclass(natom)) !atom and its class.
      do n = 1,natom          !!We set nclass = natom through the GW calculations
        iclass(n)=n
      end do

      allocate(nindxv(nl,nclass), nindxc(nl,nclass),
     &        occv(nl,nnv,nclass),unoccv(nl,nnv,nclass),
     &        occc(nl,nnc,nclass),unoccc(nl,nnc,nclass))
      allocate(ncwf2(nl,nnc,nclass),ncwf(nl,nnc,nclass))
      allocate(tolbas(0:2*(nl-1)))
      ifix=ifi
      call rgwinaf(ifi,ifinin,nl,nnv,nnc,nclass, !ifi can be changed.
c> BZ
     o                  n1,n2,n3,efdummy,
c> frequencies
     o                  niw,diw,nwdummy,dw,delta,deltaw,esmr,imagw,
c> coulomb
c     o                  tolvc,alp,alptx,h,ng,
c> product basis
     o                  tolbas,lcutmx,nindxv,nindxc,
     o                  occv,unoccv, occc,unoccc,
c> core
     o                  ncwf,ncwf2 )
c----
      allocate(iantiferro(1:natom),spid(1:natom))
      read(ifix,*)
      read(ifix,*)iantiferro(1:natom) !may2015
      read(ifix,*)
      read(ifix,*)spid(1:natom)

      inquire(file='NoCore',exist=nocore)
      if(nocore) then
        occc=0    ! call iclear(nl*nnc*nclass, w(ioccc))
        unoccc=0    ! call iclear(nl*nnc*nclass, w(iunoccc))
        ncwf  =0    ! call iclear(nl*nnc*nclass, w(incwf))
      elseif( incwfx==-1 ) then
        write(6,*)' ### incwf=-1 Use ForSxc for core'
        ncwf = ncwf2  !call icopy(nl*nnc*nclass,w(incwf2),w(incwf))
      elseif( incwfx==-2 ) then
        write(6,*)' ### incwf=-2 Use NOT(ForSxc) for core and Pro-basis '
        call notbit(nl*nnc*nclass, ncwf2)
        ncwf  = ncwf2 ! call icopy (nl*nnc*nclass, w(incwf2),w(incwf))
        occc= ncwf  ! call icopy (nl*nnc*nclass, w(incwf),w(ioccc))
        unoccc= 0     ! call iclear(nl*nnc*nclass, w(iunoccc))
      elseif( incwfx==-3 ) then
        call ibiton(nclass,nl,nnc,nindxc, occc, ncwf)
        unoccc= 0     ! call iclear(nl*nnc*nclass, w(iunoccc))
        write(6,*)' ### incwf=-3  occ=1 unocc=0 incwf=1 for all core '
      elseif( incwfx==-4 ) then
        write(6,*)' ### incwf=-4  occ=0 and unocc=0 for all core '
        occc=0  !call iclear(nl*nnc*nclass, w(ioccc))
        unoccc=0 !call iclear(nl*nnc*nclass, w(iunoccc))
        ncwf=0  !call iclear(nl*nnc*nclass, w(incwf))
      elseif(incwfx==0) then
        write(6,*)' ### Use unocc occ ForX0 for core'
      else
        call rx( ' ### proper incwf is not given for genallcf2:rgwinf ')
      endif
      deallocate(ncwf2)
C... End of rgwinf section ----------------------------


c> dimensions and constants
      lmx        = 2*(nl-1)
      lmx2       = (lmx+1)**2
      nlmto      = noflmto(nindxv,iclass,nl,nclass,natom)
      nlmto2     = nlmto*nlmto
      nn         = maxnn(nindxv,nindxc,nl,nclass)

c>> combine nocc,nunocc,nindx
      allocate(nindx(nl,nclass))
      allocate(nocc(nl,nn,nclass),nunocc(nl,nn,nclass))
      call reindx(occv,unoccv,nindxv,
     i              occc,unoccc,nindxc,
     d              nl,nn,nnv,nnc,nclass,
     o              nocc,nunocc,nindx)
      call maxdim(occc,unoccc,nindxc,nl,nnc,nclass,
     o              nprodxc,nlnxc,nlnmxc,nlnaxc)
      call maxdim(occv,unoccv,nindxv,nl,nnv,nclass,
     o              nprodxv,nlnxv,nlnmxv,nlnaxv)
      call maxdim(nocc,nunocc,nindx,nl,nn,nclass,
     o              nprodx,nlnx,nlnmx,nlnax)

c      nlnx4      = nlnx**4
c      nphi       = nrx*nl*nn*nclass
c      pi         = 4d0*datan(1d0)
c      tpia       = 2d0*pi/alat

c$$$c> frequency mesh
c$$$c      call defdr   (ifreq,nw)
c$$$      write(6,*)' nw from rgwinaf=',nw
c$$$      if(nw>0) then
c$$$        allocate(freq(nw))
c$$$        call genfreq (nw,dw,0.d0,
c$$$     o              freq )
c$$$      endif
```

```

c> index for allowed core states
c      call defi    (iicore,nl*nl*nnc*nclass)
c      call defi    (incore,nclass)
      allocate(icore(nl**2*nnc,nclass),ncore(nclass))
      icore=9999999
      ncore=9999999
      call incor(ncwf,nindxc,iclass,
     d              nl,nnc,nclass,natom,
     o              icore,ncore,nctot )
ccccccccccccccccccccccccccccccccccccccccccccccccc
c      write(6,*)' nnc=',nnc,nl,nclass,natom
c      write(6,*)' ncwf ',ncwf
c      write(6,*)' nindxc ',nindxc
c      write(6,*)' iclass ',iclass
c      write(6,*)' --- icore=',icore
c      write(6,*)' --- ncore nctot=',ncore,nctot
ccccccccccccccccccccccccccccccccccccccccccccccccc

c> core energies
      ifec       = iopen('ECORE',1,0,0)
      allocate(konf(nl,nclass),ecore(nctot,2))
      konf=0
      allocate(ecoret(0:nl-1,nnc,2,nclass))
      ecoret=0d0
      do ic = 1,nclass
        write(6,*) ' read ECORE : ic=',ic
        read (ifec,*)
        read (ifec,*)
        read (ifec,*)
        read (ifec,*) !zz,ic1,nr ,a,b,nsp
        read (ifec,*)
        read (ifec,*) (konf(l+1,ic),l=0,nl-1)
        read (ifec,*)
        do  l = 0,nl-1
          ncorex = konf(l+1,ic)-l-1
          if (ncorex .gt. nnc) call rx( 'ECORE: wrong nnc')
          do n = 1,ncorex
            read (ifec,*) lt,nt,(ecoret(l,n,isp,ic),isp=1,nspin) !takao
            if(nspin==1) ecoret(l,n,2,ic) = ecoret(l,n,1,ic)        !
c           write(6,"(' read ecore=',3i4,2d13.5)")l,n,ic,ecoret(l,n,1:nspin,ic)
            if (lt .ne. l) call rx( 'rcore: wrong l')
            if (nt .ne. n) call rx( 'rcore: wrong n')
          end do
        end do
      end do
      i = 0
      do ia = 1,nclass
        ic  = iclass(ia)
        do l = 0,nl-1
        do n = 1,nnc
        do m = -l,l
          if (ncwf(l+1,n,ic) .eq. 1) then
            i = i + 1
            if (i > nctot) call rx( 'genalloc_mod: wrong nctot')
            ecore(i,1:nspin) = ecoret(l,n,1:nspin,ic)
            write(6,"(' ecore=',4i4,2d13.5)")i, l,n,ic,ecore(i,1:nspin)
          endif
        enddo
        enddo
        enddo
      enddo
      deallocate(ecoret)
c> index for core and LMTO basis
c      call defi    (iil,nlnmx*nclass)
c      call defi    (iin,nlnmx*nclass)
c      call defi    (iim,nlnmx*nclass)
c      call defi    (iilnm,nn*nl*nl*nclass)
c      call defi    (iilv,nlnmxv*nclass)
c      call defi    (iinv,nlnmxv*nclass)
c      call defi    (iimv,nlnmxv*nclass)
c      call defi    (iilnmv,nnv*nl*nl*nclass)
c      call defi    (iilc,nlnmxc*nclass)
c      call defi    (iinc,nlnmxc*nclass)
c      call defi    (iimc,nlnmxc*nclass)
c      call defi    (iilnmc,nnc*nl*nl*nclass)
      allocate(
     & il(nlnmx,nclass),
     & in(nlnmx,nclass),
     & im(nlnmx,nclass),
     & ilnm(nn*nl*nl*nclass),
     & ilv(nlnmxv*nclass),
     & inv(nlnmxv*nclass),
     & imv(nlnmxv*nclass),
     & ilnmv(nnv*nl*nl*nclass),
     & ilc(nlnmxc*nclass),
     & inc(nlnmxc*nclass),
     & imc(nlnmxc*nclass),
     & ilnmc(nnc*nl*nl*nclass)
     & )
      call idxlnmc( nindxv,nindxc,
     d              nl,nn,nnv,nnc,nlnmx,nlnmxv,nlnmxc,nclass,
     o              il,in,im,ilnm,
     o              ilv,inv,imv,ilnmv,
     o              ilc,inc,imc,ilnmc)
      allocate(nlnmv(nclass),nlnmc(nclass),nlnm(nclass))
      call nolnma(nindxv,nl,nclass,
     o              nlnmv )
      call nolnma(nindxc,nl,nclass,
     o              nlnmc )
      call nolnma(nindx,nl,nclass,
     o              nlnm )
      i=2 !see previous definition of symgrp
      if(symgrp(i+1:i+13)/= 'UseSYMOPSfile') then
        call rx( " Not: UseSYMOPSfile in LMTO file")
      endif
      write(6,*) ' symgrp==UseSYMOPSfile'
      ifi = 6661
      open (ifi, file='SYMOPS')
      read(ifi,*) ngrp
      allocate(symgg(3,3,ngrp))
      do ig = 1,ngrp
        read(ifi,*)
        do i=1,3
          read(ifi,"(3d24.16)") symgg(i,1:3,ig)
        enddo
      enddo
      close(ifi)
      allocate(invg(ngrp))
      call invgrp(symgg,ngrp,
     o              invg)
      is = iclose('LMTO')
      is = iclose('ECORE')
      call cputid(0)
      write(6,*) 'genallcf_v3'
      end subroutine genallcf_v3
      end module

      subroutine idxlnmc(nindxv,nindxc,
     d                   nl,nn,nnv,nnc,nlnmx,nlnmxv,nlnmxc,nclass,
     o                   il,in,im,ilnm,
     o                   ilv,inv,imv,ilnmv,
     o                   ilc,inc,imc,ilnmc)
c 92.jan.07
c 92.03.17 include core states
c indexing of core states and LMTO basis functions for all classes,
c follows that in TB-LMTO program
c il,in,im = l,n,m
c ilnm(n,lm) = index of n,l,m
c lm = l*l + l + m + 1
c NOTE: the indexing starts with core first and then valence on top
c       of core (not the same as index generated from nindx)
      implicit real*8(a-h,o-z)
      dimension nindxv(0:nl-1,nclass),nindxc(0:nl-1,nclass)
      dimension ilnm(nn,nl*nl,nclass),
     o          ilnmv(nnv,nl*nl,nclass),
     o          ilnmc(nnc,nl*nl,nclass),
     o          in(nlnmx,nclass),il(nlnmx,nclass),im(nlnmx,nclass),
     o         inv(nlnmxv,nclass),ilv(nlnmxv,nclass),imv(nlnmxv,nclass),
     o         inc(nlnmxc,nclass),ilc(nlnmxc,nclass),imc(nlnmxc,nclass)
      do     ic = 1,nclass
        ind       = 0
c core
        do      l = 0,nl-1
          l2        = l*l
          do      n = 1,nindxc(l,ic)
            do      m = 1,2*l+1
              ind       = ind + 1
              if (ind .gt. nlnmx) call rx( 'idxlnmc: ind > nlnmx')
              lm        = l2 + m
              il(ind,ic)= l
              in(ind,ic)= n
              im(ind,ic)= m - l - 1
              ilnm(n,lm,ic) = ind
              ilc(ind,ic)= l
              inc(ind,ic)= n
              imc(ind,ic)= m - l - 1
              ilnmc(n,lm,ic)= ind
            end do
          end do
        end do
c valence
        indv      = 0
        do      l = 0,nl-1
```

```
          l2        = l*l
          ncore     = nindxc(l,ic)
          do      n = 1,nindxv(l,ic)
            if (ncore+n .gt. nn) call rx( 'idxlnmc: ncore+n > nn')
            do      m = 1,2*l+1
              ind       = ind + 1
              indv      = indv + 1
              if (ind .gt. nlnmx) call rx( 'idxlnmc: ind > nlnmx')
              lm        = l2 + m
              il(ind,ic)= l
              in(ind,ic)= ncore + n
              im(ind,ic)= m - l - 1
              ilnm(ncore+n,lm,ic) = ind
              ilv(indv,ic)= l
              inv(indv,ic)= n
              imv(indv,ic)= m - l - 1
              ilnmv(n,lm,ic) = indv
            end do
          end do
        end do
      end do
      return
      end

      integer function noflmto(nindx,iclass,nl,nclass,natom)
c total number of LMTO basis functions
      implicit real*8(a-h,o-z)
      dimension nindx(0:nl-1,nclass),iclass(natom)
      noflmto   = 0
      do 1    i = 1,natom
        ic        = iclass(i)
      do 1    l = 0,nl-1
        noflmto   = noflmto + (2*l+1)*nindx(l,ic)
    1 continue
      return
      end

      integer function nalwln (nocc,nunocc,nindx,nl,nn)
c gives the number of allowed product radial phi
c nocc(l,n)   = 0,1 ==> unoccupied, occupied
c nunocc(l,n) = 1,0 ==> unoccupied,occupied
c nalwln    = number of allowed phi(l1,n1) phi(l2,n2)
      implicit real*8(a-h,o-z)
      parameter(lmax=6,nnx=10)
      dimension nocc(0:nl-1,nn),nunocc(0:nl-1,nn),
     i          nindx(0:nl-1)
      dimension icheck(0:lmax,nnx,0:lmax,nnx)
      if (nl-1 .gt. lmax) call rx( 'nalwln: increase lmax')
      if (nn .gt. nnx) call rx( 'nalwln: increase nnx')
      icheck=0
      nalwln     = 0
      do 10   l1 = 0,nl-1
      do 10   n1 = 1,nindx(l1)
        if(nocc(l1,n1) .eq. 0)goto 10
        do 20   l2 = 0,nl-1
        do 20   n2 = 1,nindx(l2)
          if(nunocc(l2,n2) .eq. 0)goto 20
          if((l1.ne.l2 .or. n1.ne.n2) .and. icheck(l2,n2,l1,n1).ne.0)
     . goto 20
          nalwln     = nalwln + 1
          icheck(l1,n1,l2,n2) = nalwln
   20   continue
   10 continue
      return
      end

      integer function nofln(nindx,nl)
c count the number of l,n
      implicit real*8(a-h,o-z)
      dimension nindx(0:nl-1)
      nofln      = 0
      do       l = 0,nl-1
        nofln      = nofln + nindx(l)
      end do
      return
      end
c-------------------------------------------------------------------
      integer function noflnm(nindx,nl)
c number of l,n,m
      implicit real*8(a-h,o-z)
      dimension nindx(0:nl-1)
      noflnm    = 0
      do 1    l = 0,nl-1
        noflnm    = noflnm + nindx(l)*(2*l+1)
    1 continue
      return
      end
      integer function nallow (nocc,nunocc,nindx,nl,nn)
c gives the number of allowed product basis
c nocc(n,l) = 0,1 ==> unoccupied, occupied
c nallow    = number of allowed product basis
      implicit real*8(a-h,o-z)
      parameter(lmax=6,nnx=10)
      dimension nocc(0:nl-1,nn),nunocc(0:nl-1,nn),
     i          nindx(0:nl-1)
      dimension icheck(0:lmax,nnx,0:lmax,nnx)
      if(nl-1 .gt. lmax) call rx( 'nallow: increase lmax')
      if(nn .gt. nnx) call rx( 'nallow: increase nnx')
      icheck=0
      do      l1 = 0,nl-1
        do      n1 = 1,nindx(l1)
          do      l2 = 0,nl-1
            do      n2 = 1,nindx(l2)
              icheck(l1,n1,l2,n2) = nocc(l1,n1)*nunocc(l2,n2)
              if (l1 .ne. l2 .or. n1 .ne. n2) then
                if (icheck(l1,n1,l2,n2)*icheck(l2,n2,l1,n1) .ne. 0)
     . icheck(l1,n1,l2,n2) = 0
              endif
            end do
          end do
        end do
      end do
      nallow     = 0
      do 10   l1 = 0,nl-1
      do 10   n1 = 1,nindx(l1)
      do 10   m1 = 1,2*l1+1
      do 10   l2 = 0,nl-1
      do 10   n2 = 1,nindx(l2)
      do 10   m2 = 1,2*l2+1
c     if (nocc(l1,n1) .eq. 0)goto 10
c     if (nunocc(l2,n2) .eq. 0)goto 10
        if (icheck(l1,n1,l2,n2) .eq. 0) goto 10
c temporary
        if (l1 .eq. l2 .and. n1.eq.n2 .and. m1.lt.m2)goto 10
        nallow     = nallow + 1
   10 continue
      return
      end

      subroutine incor  (ncwf,nindxc,iclass,
     d                   nl,nnc,nclass,natom,
     o                   icore,ncore,nctot)
c 92.03.18
c sorts out allowed core states and count the number of core states
c ncwf(l,n,cl) = 1 ==> allowed, 0 ==> not allowed
c nindxc(l,cl)  = no. core states/l,class
c nl,nnc = max. no. l,n
c icore(i,cl) = index for allowed core states
c ncore(cl)   = no. allowed core states
c nctot       = total no. allowed core states
      implicit real*8 (a-h,o-z)
      dimension ncwf(0:nl-1,nnc,nclass),nindxc(0:nl-1,nclass),
     i          iclass(natom)
      dimension icore(nl*nl*nnc,nclass),ncore(nclass)
      ncx        = nl*nl*nnc
      do      ic = 1,nclass
        i          = 0
        j          = 0
        do       l = 0,nl-1
          do       n = 1,nindxc(l,ic)
            do       m = -l,l
              j          = j + 1
              if (ncwf(l,n,ic) .eq. 1) then
                i          = i + 1
                if (i .gt. ncx) call rx( 'incore: wrong ncx')
                icore(i,ic)= j
              endif
            end do
          end do
        end do
        ncore(ic)  = i
      end do
c total no. allowed core states
      nctot      = 0
      do       i = 1,natom
        ic         = iclass(i)
        nctot      = nctot + ncore(ic)
      end do
      return
      end
```

4.5 gwsrc/m_anf.F File Reference

Data Types

- module m_anf

  Antiferro condition module. We have line AFcond at the bottom of 'LMTO' file. Currently(feb2016), only laf is used (thus AF symmetry is not used yet for hx0fp0_sc) To access laf, need to call anfcond() in advance.

4.6 m_anf.F

```
!> Antiferro condition module. We have line AFcond at the bottom of 'LMTO' file.
!! Currently(feb2016), only laf is used (thus AF symmetry is not used yet for hx0fp0_sc)
!! To access laf, need to call anfcond() in advance.

      module m_anf
      implicit none
      logical,protected:: laf !! - laf: antiferro switch
      integer,allocatable,protected:: ibasf(:) !! - ibasf(ibas) specify AF pair atom.
c       integer:: natom
c      ,ldima(:),iantiferro(:),iclasst(:)
c       real(8),allocatable:: pos(:,:),anfvec(:),qlat(:,:),plat(:,:)
      contains

      subroutine anfcond()
      implicit none
      integer,allocatable:: iantiferro(:)
      integer:: ifile_handle,ilmto,ildima,ificlass
      character(256):: aaa,keyplat
      real(8)::vecs(3),vece(3),basdiff(3)
      integer:: ibas,lkeyplat,i,ibasx,natom
      character(3)::iaaa
!! read LMTO file
      write(6,*) 'Read AFcond section in LMTO file, call anfcond in m_anf.F:'
      ilmto=ifile_handle()
      open(ilmto,file='LMTO')
      do
        read(ilmto,"(a)",end=1011,err=1011) aaa
        aaa = adjustl(aaa)
        if(trim(aaa)=='number of atoms (natom)') then
          read(ilmto,*) natom
          read(ilmto,*)
          allocate(iantiferro(natom),ibasf(natom))
        endif
        if(aaa(1:6)=='AFcond') then
          read(ilmto,*) iantiferro(1:natom)
          ibasf=-999
          do ibas=1,natom
            do ibasx=ibas+1,natom
              if(abs(iantiferro(ibas))/=0 .and. iantiferro(ibas)+iantiferro(ibasx)==0) then
                ibasf(ibas)=ibasx
                exit
              endif
            enddo
            if(ibasf(ibas)/=-999) write(6,"(a,2i5)")' AF pair: ibas ibasf(ibas)=',ibas,ibasf(ibas)
          enddo
        endif
      enddo
 1011 continue
      close(ilmto)
      if(sum(abs(iantiferro))==0) then
        laf=.false. !no AF case
        return
      endif
!! Antiferro case --------------
      laf=.true.
      if(laf) write(6,"(a,100i4)") ' Antiferromode=',iantiferro
      end subroutine anfcond
      end module

```

4.7 gwsrc/m_freq.F File Reference

Data Types

- module m_freq

    Frequency mesh generator.

4.8 m_freq.F

```
!>Frequency mesh generator
!! - OUTPUT
!!   - fhris :histgram bins to accumlate im part
!!   - freq_r: omega along real axis
!!   - freq_i: omega along imag axis
!!   - wiw: integration weight along im axis
!!   - npm: npm=1 means only positive omega;npm=2 means positive and negative omega.
!! - NOTE: change of frequency mesh defined here may destroy consistency or not. Need check
      module m_freq
      real(8),allocatable,protected:: frhis(:),freq_r(:),freq_i(:),wiw(:)
      integer,protected:: nwhis,npm,nw_i,nw

c      real(8),allocatable,protected:: frhis0(:)
c      integer,protected:: nwhis0
      contains
!> Get data set for m_freq. All arguments are input.
!! - This read GWinput (dw,omg_c) and TimeReversal()
!! - All arguments are input
      subroutine getfreq(epsmode,realomega,imagomega,tetra,omg2max,wemax,niw,ua,mpi__root)
      use m_keyvalue,only:getkeyvalue

      implicit none
      integer,intent(in):: niw !,nw_input
      logical,intent(in):: realomega,imagomega,tetra,mpi__root,epsmode
      real(8),intent(in):: omg2max,ua

      real(8),allocatable:: freqx(:),wx(:),expa(:)
      logical:: timereversal,onceww
      integer:: nw2,iw,ihis
      real(8)::omg_c,dw,omg2,wemax
      real(8), allocatable :: freqr2(:)  ,frhis_tmp(:)
      real(8)::  pi = 4d0*datan(1d0), aa,bb,ratio,oratio,daa
      integer::nee,noo,ifif,ifile_handle

      logical,save:: done=.false.
      if(done) call rx('gerfreq is already done') !sanity check
      done =.true.
      nw=-99999 !for sanity check

c      nw = nw_input
!! Histogram bin divisions
!! We first accumulate Imaginary parts.
!! Then it is K-K transformed to obtain real part.

c      call getkeyvalue("GWinput","dw",dw )
c      call getkeyvalue("GWinput","omg_c",omg_c )
c      write(6,"('dw, omg_c= ',2f13.5)") dw, omg_c
      call getkeyvalue("GWinput","HistBin_ratio",oratio, default=1.03d0)
      call getkeyvalue("GWinput","HistBin_dw",dw, default=1d-5) !a.u.
      aa = oratio-1d0
      bb = dw/aa
      iw = 0d0
      do
        iw=iw+1
        if( bb*( exp(aa*(iw-1)) - 1d0 ) >omg2max+1d-6) exit
      enddo
      nwhis = iw+2 !+2 for margin. Necessary?
      allocate(frhis(1:nwhis+1))
      do iw = 1,nwhis+1
        frhis(iw) = bb*( exp(aa*(iw-1)) - 1d0 )
      enddo
      write(6,"('dw, omg_ratio, nwhis= ',d9.2,f13.5,i6)") dw, aa,nwhis

!! Determine nw. Is this correct?
      do iw=3,nwhis
         omg2 = (frhis(iw-2)+frhis(iw-1))/2d0
         if (omg2 > wemax/2d0 ) then !>dw*(nw_input-3)) then !omg is in unit of Hartree
            nw=iw
            exit
         endif
      enddo
!! document need to be fixed...
c      nw=nw2-1      ! nw+1 is how many points of real omega we use
                    ! for dressed coulomb line W(iw=0:nw) iw=0 corresponds omg=0
```

```
                       ! maximum nw=nw2-1 because nwhis=nw2-1
!! document need to be fixed...
                       !nw is chosen from condition that frhis_m(nw-3)<dw*(nw_input-3) <frhis_m(nw-2).
                       !Here frhis_m(iw)= (freqr2(iw)+freqr2(iw+1))/2d0
                       !nw was constructed such that omg=dw*(nw-2)> all relevant frequensies needed
                       ! for correlation Coulomb Wc(omg),
                       ! and one more point omg=dw*(nw-1) needed for extrapolation.
                       ! Now, frhis_m(nw-1)> all relevent frequensies for Wc(omg)
                       ! and one more point omg=frhis_m(nw) needed for extropolation
                       ! used in subroutine alagr3z in  sxcf.f.

!! Determine freq_r
      if(epsmode) then
        nw  = nwhis-1
      endif
      allocate(freq_r(0:nw))
      freq_r(0)=0d0
      do iw=1,nw
        freq_r(iw)=(frhis(iw)+frhis(iw+1))/2d0
      enddo

!! Timereversal=F is implimented only for tetra=T and sergeyv=T
!! nw_i and npm
      npm=1
      nw_i=0
      if(.not.timereversal()) then
         write(6,"('TimeReversal off mode')")
         npm=2
         nw_i=-nw
         if(.not.tetra)   call rx( ' tetra=T for timereversal=off')
      endif
      write(6,*)'Timereversal=',timereversal()

!! Write freq_r
      if(realomega .and. mpi__root) then
        ififf=ifile_handle()
         open(unit=ifif,file='freq_r') !write number of frequency points nwp and frequensies in 'freq_r'
 file
         write(ifif,"(2i8,'  !(a.u.=2Ry)')") nw+1, nw_i
         do iw= nw_i,-1
            write(ifif,"(d23.15,2x,i6)") -freq_r(-iw),iw
         enddo
         do iw= 0,nw
            write(ifif,"(d23.15,2x,i6)") freq_r(iw),iw
         enddo
         close(ifif)
      endif

!! Determine freq_i  : gaussian frequencies x between (0,1) and w=(1-x)/x
      if (imagomega) then
         write(6,*)' freqimg: niw =',niw
         allocate( freq_i(niw) ,freqx(niw),wx(niw),expa(niw) )
         call freq01(niw,ua,
     o        freqx,freq_i,wx,expa)
         allocate(wiw(niw))
         do iw=1,niw
           wiw(iw)=wx(iw)/(2d0*pi*freqx(iw)*freqx(iw))
         enddo
         deallocate(freqx,wx,expa)
      endif

!! Plot frhis
      if(onceww(1)) then
        write(6,*)' we set frhis nwhis noo-->nee=',nwhis,noo,nee
        write(6,*)' --- Frequency bins to accumulate Im part  (a.u.) are ---- '
        do ihis= 1, nwhis !min(10,nwhis)
          write(6,"(' ihis Init  End=', i5,2f18.11)") ihis,frhis(ihis),frhis(ihis+1)
        enddo
      endif
      end subroutine getfreq
      end module m_freq
```

4.9 gwsrc/m_hamindex.F File Reference

Data Types

- module m_hamindex

  This is in lm7K/subs/m_hamindex.F and in fpgw/gwsrc/m_hamindex.F We will need to unify make system and source code in fpgw and lmf. norbtx is given in gwsrc/readeigen.F init_readeigen2.

4.10 m_hamindex.F

```
!> This is in lm7K/subs/m_hamindex.F and in fpgw/gwsrc/m_hamindex.F
!! We will need to unify make system and source code in fpgw and lmf.
!! norbtx is given in gwsrc/readeigen.F init_readeigen2
      module m_hamindex
      integer,parameter,private:: null=-999999
      logical,private:: debug=.false.

      integer,protected:: ngrp=null, lxx=null, kxx=null,norbmto=null
      integer,protected:: nbas,nqtt,ndimham=null
      integer,allocatable,protected:: ltab(:),ktab(:),offl(:),ispec(:), iclasst(:),offlrev(:,:,:),ibastab(:
)
      integer,allocatable,protected:: iqimap(:),iqmap(:),igmap(:),invgx(:),miat(:,:),ibasindex(:)
!,ngvecp(:,:,:),ngvecprev(:,:,:,:)
      real(8),allocatable,protected:: symops(:,:,:),ag(:,:),tiat(:,:,:),shtvg(:,:), dlmm(:,:,:,:),qq(:,:)
      real(8),protected:: plat(3,3),qlat(3,3)
      real(8),allocatable,protected:: qtt(:,:),qtti(:,:)
      integer,allocatable,protected:: igv2(:,:,:),napwk(:),igv2rev(:,:,:,:)
      integer,protected::  napwmx=null,lxxa=null


      integer:: norbtx=null
      integer:: nqi, nqnum,ngpmx,imx=null
      contains

!> get index ikt such that for qin(:)=qq(:,ikt)
      integer function getikt(qin) !return
      integer::i
      real(8):: qin(3)
c      if(debug) print *,'nkt=',nkt
      do i=1, nqnum !*2 !nkt
        if(debug) print *,i,qin, qq(:,i)
        if(sum(abs(qin-qq(:,i)))<1d-8) then
          getikt=i
          return
        endif
      enddo
      print *,' getikt: xxx error nqnum qin=',nqnum,qin
      do i=1, nqnum !*2 !nkt
        write(*,"('i qq=',i3,3f11.5)")i, qq(:,i)
      enddo
      call rx( ' getikt can not find ikt for given q')
      end function

!> write info for wave rotation.
      subroutine writehamindex()
      integer(4):: ifi
      logical::pmton
      logical,save:: done=.false.
      if(done) call rx('writehamindex is already done')
      done=.true.
      ifi=1789
      open(ifi,file='HAMindex',form='unformatted')
      write(ifi)ngrp,nbas,kxx,lxx,nqtt,nqi,nqnum,imx,ngpmx,norbmto
      write(ifi)symops,ag,invgx,miat,tiat,shtvg,qtt,qtti,iqmap,igmap,iqimap
      write(ifi)lxxa
      write(ifi)dlmm
      write(ifi)ibastab,ltab,ktab,offl,offlrev !for rotation of MTO. recovered sep2012 for EIBZ for hsfp0
      write(ifi)qq !,ngvecp,ngvecprev
      write(ifi)plat,qlat,napwmx
      if(napwmx/=0) then !for APW rotation used in rotwvigg
         write(ifi) igv2,napwk,igv2rev
      endif
      close(ifi)
      end subroutine writehamindex

!> read info for wave rotation.
      subroutine readhamindex()
      integer(4):: ifi,nkt
      logical::pmton
      logical,save:: done=.false.
      if(done) call rx('readhamindex is already done')
      done=.true.
      ifi=1789
      open(ifi,file='HAMindex',form='unformatted')
      read(ifi)ngrp,nbas,kxx,lxx,nqtt,nqi,nqnum,imx,ngpmx,norbmto
      allocate(symops(3,3,ngrp),ag(3,ngrp),qtt(3,nqtt),qtti(3,nqi))
      allocate(invgx(ngrp),miat(nbas,ngrp),tiat(3,nbas,ngrp),shtvg(3,ngrp))
      allocate(iqmap(nqtt),igmap(nqtt),iqimap(nqtt))
      write(6,*) 'ngrp=',ngrp
      read(ifi)symops,ag,invgx,miat,tiat,shtvg,qtt,qtti,iqmap,igmap,iqimap
      allocate( ltab(norbmto),ktab(norbmto),offl(norbmto),ibastab(norbmto) )
      allocate( offlrev(nbas,0:lxx,kxx))
      read(ifi) lxxa
      allocate( dlmm(-lxxa:lxxa, -lxxa:lxxa, 0:lxxa, ngrp))
      read(ifi) dlmm
      read(ifi)ibastab,ltab,ktab,offl,offlrev
c      allocate( ngvecprev(-imx:imx,-imx:imx,-imx:imx,nqnum) )
c      allocate( ngvecp(3,ngpmx,nqnum) )
      allocate( qq(3,nqnum)) !this was qq(3,nqnum*2) until Aug2012 when shorbz had been used.
      read(ifi)qq !,ngvecp,ngvecprev
      read(ifi)plat,qlat,napwmx
      if(napwmx/=0)then !for APW rotation used in rotwvigg
        nkt=nqtt
        allocate( igv2(3,napwmx,nkt) )
        allocate( napwk(nkt))
        allocate( igv2rev(-imx:imx,-imx:imx,-imx:imx,nkt) )
        read(ifi) igv2,napwk,igv2rev
      endif
      close(ifi)
      done=.true.
      end subroutine readhamindex
      end module


```

4.11 gwsrc/m_tetwt.F File Reference

Data Types

- module m_tetwt

  Get the weights and index for tetrahedron method for the Lindhard function.

4.12 m_tetwt.F

```
!> Get the weights and index for tetrahedron method for the Lindhard function.
!!    - nbnb = total number of weight.
!!    - n1b  = band index for occ.   1\ge n1b \ge nband+nctot.
!!       "Valence index->core index" ordering(Core index follows valence index).
!!    - n2b  = band index for unocc. 1\ge n2b \ge nband
!!    - wwk(ibib,...)  = (complex)weight for the pair for n1b(ibib...),n2b(ibib...).
!!
!! - NOTE: 'call getbzdata1' generates nteti,ntetf,... See mkqg.F about how to call it.
!!
      module m_tetwt
      real(8),allocatable,protected :: whw(:)
      integer,allocatable,protected:: ihw(:,:,:),nhw(:,:,:),jhw(:,:,:),ibjb(:,:,:,:)
      integer,protected:: nbnbx,nhwtot
      integer,allocatable,protected :: n1b(:,:,:),n2b(:,:,:),nbnb(:,:)
!!
      contains !! ----------------------------------------------------------------------
      subroutine tetdeallocate()
      deallocate(ihw,nhw,jhw, whw,ibjb,n1b,n2b,nbnb)
      end subroutine

!! routine ------------------------------------------------------------
      subroutine gettetwt(q,iq,is,isf,nwgt,frhis,nwhis,npm,
     i   qbas,ginv, ef, nqibz_mtet, nband,ekxx1,ekxx2, nctot,ecore,
     i   nqbz,qbz,nqbzw,qbzw, ntetf,idtetf,ib1bz,
     i   nbmx,ebmx,mtet,eibzmode) !nov2016
!! INPUT DATA; read only
!! nqibz_mtet: is only for mtet/=(/1,1,1/) --->(we usually use only this case)
!!
!! output data in retruened in the module variables above.

!! we assume read_bzdata is called already
c      use m_read_bzdata,only: qbas,ginv, ntetf,idtetf,ib1bz !, qbzw,nqbzw,qbz, nqibz

c      use m_readeigen,only:   readeval !we assume init_readeval is called already
c      use m_genallcf_v3,only: ecore,nctot    !we assume genallcf_v3 called already.
c      use m_read_bzdata,only: nqbz,qbas,ginv,nqbzw,nteti,ntetf,idtetf,qbzw,ib1bz,nqibz,qbz
c      use m_freq,only:                 !we assume getfreq is called already.
c     &   frhis, nwhis,npm !output of getfreq
c      use m_zmel,only: nband
c      use m_readefermi,only: readefermi,ef

      implicit none
      integer,intent(in):: is,isf,iq,nwgt(:),nqibz_mtet,nqbz,nqbzw,nband,npm,nwhis,nctot,nbmx,mtet(3)
      integer,intent(in):: ntetf,idtetf(0:3,ntetf),ib1bz(nqbzw)
      real(8),intent(in):: q(3),qbas(3,3),ginv(3,3),ef,qbz(3,nqbz),qbzw(3,nqbzw),ebmx
      real(8),intent(in):: ekxx1(nband,nqbz),ekxx2(nband,nqbz) !qbzw(:,: )
      real(8),intent(in):: frhis(1:nwhis+1),ecore(nctot,2)
```

```fortran

      real(4),allocatable :: demin(:,:,:,:),demax(:,:,:,:)
      logical,allocatable :: iwgt(:,:,:,:)
      integer,allocatable:: nbnbtt(:,:),noccxvv(:) !  &         idtetf(:,:),ib1bz(:)
      logical :: eibzmode,tetra,tmpwwk=.false.,debug,eibz4x0
      integer::kx,ncc,job,jpm,noccxvx(2)=-9999,ik,jhwtot,ib1,ib2,ibib,noccx,noccxv,verbose,ifief,ifile_handle
      real(8),allocatable:: ecore_(:,:)
      if(nctot==0) then
        allocate(ecore_(1,2))    !this is dummry
      else
        allocate(ecore_(nctot,2))
        ecore_=ecore
      endif

      tetra=.true.
c      eibzmode = eibz4x0()
      debug=.false.
      if(verbose()>=100) debug=.true.

c      if(.not.allocated(nbnb))
      allocate( nbnb(nqbz,npm))
      allocate( nbnbtt(nqbz,npm)) !,ekxx1(nband,nqbz),ekxx2(nband,nqbz))

!!===========tetraini block tetra==.true.===============================1ini
c      if(tetra) then
      write(6,"(' tetra mode nqbz nband ispin q=',2i7,i2,3f13.6)") nqbz,nband,is,q

!! move to upper level nov2016
c$$$!!     ekxx1 for rk
c$$$!!     ekxx2 for q+rk See tetwt4
c$$$      do kx = 1, nqbz
c$$$cccccccccccccccc
c$$$c        write(6,"('kkkkk kx ',i4,3f9.4,3x,3f9.4)") kx,qbz(:,kx),qbzw(:,kx)
c$$$cccccccccccccccc
c$$$        call readeval(qbz(:,kx),   is,  ekxx1(1:nband, kx) )
c$$$        call readeval(q+qbz(:,kx), isf, ekxx2(1:nband, kx) )
c$$$      enddo

c     takao-feb/2002 i replaced tetwt4(1d30) with tetwt5(job=0) -----
c     ... get pairs(n1b n2b) with non-zero tetrahedron wieghts.
c     the pairs are not dependent on the energy otemega
c     in the denominator of the dielectric function.
      write(6,"(' -- First tetwt5 is to get size of array --')")
      job = 0
      if(npm==1) then
        ncc=0
      else
        ncc=nctot
      endif
      allocate( demin(nband+nctot,nband+ncc,nqbz,npm),
     &          demax(nband+nctot,nband+ncc,nqbz,npm) )
      allocate( iwgt(nband+nctot,nband+ncc,nqbz,npm) )
!     wgt, demin, demax may require too much memory in epsilon mode.
!     We will have to remove these memory allocations in future.
!     tetwt5x_dtet2 can be very slow because of these poor memory allocation.
c      if(nctot==0) then
c        deallocate(ecore)
c        allocate(ecore(1,2))    !this is dummry
c      endif
      allocate(ibjb(1,1,1,1),ihw(1,1,1),jhw(1,1,1),nhw(1,1,1),whw(1)) !dummy
c--- efermi
c      ifief=ifile_handle()
c      open(ifief,file='EFERMI')
c      read(ifief,*) ef
c      close(ifief)
c      call readefermi() !comment out,since ef is passed nov2016
cccccccccccccccc
c      print *,'nqbz,nqbzw,nteti,ntetf,nqibz_mtet=',nqbz,nqbzw,nteti,ntetf,nqibz_mtet

      call tetwt5x_dtet4(npm,ncc,
     i q, ekxx1, ekxx2, qbas,ginv,ef,
     d ntetf,nqbzw, nband,nqbz,
     i nctot,ecore_(1,is),idtetf,qbzw,ib1bz,
     i job,
     o iwgt,nbnb,               !job=0
     o demin,demax,             !job=0
     i frhis, nwhis,            ! job=1    not-used
     i nbnbx,ibjb,nhwtot,       ! job=1    not-used
     i ihw,nhw,jhw,             ! job=1    not-used
     o whw,                     ! job=1    not-used
     i iq,is,isf,nqibz_mtet, eibzmode,nwgt,
     i   nbmx,ebmx,mtet) !nov2016

      deallocate(ibjb,ihw,jhw,nhw,whw) !dummy
      nbnbx = maxval(nbnb(1:nqbz,1:npm)) !nbnbx = nbnbxx
      if(debug) write(6,*)' nbnbx=',nbnbx
      allocate(  n1b(nbnbx,nqbz,npm)
     &          ,n2b(nbnbx,nqbz,npm))
      n1b=0; n2b=0
      do jpm=1,npm
        call rsvwwk00_4(jpm, iwgt(1,1,1,jpm),nqbz,nband,nctot,ncc, nbnbx,
     o   n1b(1,1,jpm), n2b(1,1,jpm), noccxvx(jpm), nbnbtt(1,jpm))
      enddo
      if(debug) then
        do kx  = 1, nqbz
          do jpm = 1, npm
            write(6,"('jpm kx  minval n1b n2b=',4i5)")jpm,kx,
     &       minval(n1b(1:nbnb(kx,jpm),kx,jpm)),
     &       minval(n2b(1:nbnb(kx,jpm),kx,jpm))
          enddo
        enddo
      endif
      if(sum(abs(nbnb-nbnbtt))/=0)then
        do ik=1,nqbz
          write(6,*)
          write(6,*)"nbnb  =",nbnb(ik,:)
          write(6,*)"nbnbtt=",nbnbtt(ik,:)
        enddo
        call rx( 'hx0fp0:sum(nbnb-nbnbtt)/=0')
      endif
      noccxv = maxval(noccxvx)
      noccx  = nctot + noccxv
      write(6,*)' Tetra mode: nctot noccxv= ',nctot,noccxv
      deallocate(iwgt)
c      endif
c=========end of tetraini block==========================================1end

!! TetrahedronWeight_5 block. tetwt5  ixc==,4,6,11 =======4ini
c     if(ixc==11) then !sf 21May02
c     --- method(tetwt5) for the tetrahedron weight
!     Histogram secstions are specified by frhis(1:nwp)
!     The 1st   bin  is     [frhis(1),  frhis(2)]   ...
!     The last  bin  is     [frhis(nw), frhis(nwp)].
!     nwp=nw+1; frhis(1)=0
!     takao-feb/2002
      if(abs(frhis(1))>1d-12) call rx( ' hx0fp0: we assume frhis(1)=0d0')
      write(6,*)' ----------------nbnbx nqbz= ',nbnbx,nqbz
!!     ... make index sets
      allocate(ihw(nbnbx,nqbz,npm),nhw(nbnbx,nqbz,npm),jhw(nbnbx,nqbz,npm))
      ihw=0; nhw=0; jhw=0
      jhwtot = 1
      do jpm =1,npm
        do ik   = 1,nqbz
          do ibib = 1,nbnb(ik,jpm)
            call hisrange( frhis, nwhis,
     i       demin(n1b(ibib,ik,jpm),n2b(ibib,ik,jpm),ik,jpm),
     i       demax(n1b(ibib,ik,jpm),n2b(ibib,ik,jpm),ik,jpm),
     o       ihw(ibib,ik,jpm),nhw(ibib,ik,jpm))
            jhw(ibib,ik,jpm)= jhwtot
            jhwtot = jhwtot + nhw(ibib,ik,jpm)
          enddo
        enddo
      enddo
      nhwtot = jhwtot-1
      write(6,*)' nhwtot=',nhwtot
      deallocate(demin,demax)
      allocate( whw(nhwtot),    ! histo-weight
     & ibjb(nctot+nband,nband+ncc,nqbz,npm) )
      whw=0d0
      ibjb = 0
      do jpm=1,npm
        do ik   = 1,nqbz
          do ibib = 1,nbnb(ik,jpm)
            ib1  = n1b(ibib,ik,jpm)
            ib2  = n2b(ibib,ik,jpm)
            ibjb(ib1,ib2,ik,jpm) = ibib
          enddo
        enddo
      enddo
!!     ... Generate the histogram weights whw
      job=1
      write(6,*) 'goto tetwt5x_dtet4 job=',job
      allocate(demin(1,1,1,1),demax(1,1,1,1),iwgt(1,1,1,1)) !dummy
      call tetwt5x_dtet4(  npm,ncc,
     i q, ekxx1, ekxx2, qbas,ginv,ef,
     d ntetf,nqbzw, nband,nqbz,
     i nctot,ecore_(1,is),idtetf,qbzw,ib1bz,
     i job,
     o iwgt,nbnb,               ! job=0
     o demin,demax,             ! job=0
     i frhis,nwhis,             ! job=1
     i nbnbx,ibjb,nhwtot,       ! job=1
     i ihw,nhw,jhw,             ! job=1
     o whw,                     ! job=1
     i iq,is,isf,nqibz_mtet, eibzmode,nwgt,
     i   nbmx,ebmx,mtet) !nov2016
      deallocate(demin,demax,iwgt,nbnbtt)
!! ======TetrahedronWeight_5 block end =========
      end subroutine gettetwt
      end module
```

4.13 gwsrc/m_zmel.F File Reference

Data Types

- module m_zmel

  Get the matrix element zmel = ZO^-1 <MPB psi|psi> , where ZO is ppovlz. To use this module, set data in this module, and call "call get_zmelt" or "call get_zmelt2". Then we have matrix elements zmel (exchange=F for correlation) or zmeltt (exchange=T). In future, they may be unified...

Functions/Subroutines

- subroutine timeshowx (info)

4.13.1 Function/Subroutine Documentation

4.13.1.1 subroutine timeshowx (character∗(∗) *info*)

Definition at line 382 of file m_zmel.F.

Here is the caller graph for this function:

4.14 m_zmel.F

```
!> Get the matrix element zmel =  ZO^-1 <MPB psi|psi> , where ZO is ppovlz.
!! To use this module, set data in this module, and call "call get_zmelt" or "call get_zmelt2".
!! Then we have matrix elements zmel (exchange=F for correlation)
!! or zmeltt (exchange=T). In future, they may be unified...
      module m_zmel

!! Base data for crystal structure.
!! these are set by 'call genallcf_v3' usually in the main routine.
      use m_genallcf_v3,only:
     i  nclass,natom,nspin,nl,nn,nnv,nnc, ngrp,
     i  nlmto,nlnx,nlnxv,nlnxc,nlnmx,nlnmxv,nlnmxc, niw,
     i  alat,delta,deltaw,esmr,symgrp,iclass,nlnmv, !,diw,dw
c  clabl,nindxv,nindxc,ncwf,
c      &  il, in, im, ilnm, nlnm, ilv,inv,imv,  ilnmv,
c      &  ilc,inc,imc,  ilnmc,
     i   invg, nlnmc, !nindx,konf
     i  icore,ncore,occv,unoccv ,
     i  occc,unoccc, nocc, nunocc, plat, pos,z,ecore,  symgg,
     i  done_genallcf_v3
!! Get eigenfuncitons. cphi is coefficients of MTO+lo part, geig is IPW parts.
!! Before calling them (get coefficients of eigen funcitons),
!! We need to call init_readeigen, init_readeigen2 in main rouitne.
      use m_readeigen,only: readcphi,readgeig
!! Basic data set to get zmel*
!! these are set by 'call rdpp' in main routine
      use m_rdpp,only:
     i  nblocha, lx, nx,  ppbrd , mdimx,nbloch, cgr,
     i  done_rdpp
!! BZ data. To set these data 'call read_BZDATA' in main rouitne.
      use m_read_bzdata,only:
     i  nqbz,nqibz,  qbas,ginv,qbz,qibz,wbz,
     i  done_read_bzdata
!! general purpose routine to read values in GWinput file.
      use m_keyvalue,only: getkeyvalue
      implicit none
!! ---------------------------------------------
      integer,parameter:: NULL=-99999
!! These are set by mptauof in main routine. 'call mptauof'
      integer,allocatable :: miat(:,:)
      real(8),allocatable :: tiat(:,:,:),shtvg(:,:)
!! We set these values in main routine.
      integer:: nband=NULL,ngcmx=NULL,ngpmx=NULL,ntq=NULL !set in main routine
      integer,allocatable :: itq(:)                       !set in main routine
      real(8),allocatable:: ppbir(:,:,:)                  !set in main routine, call pbafp_v2.
      complex(8),allocatable,target :: ppovlz(:,:)        !set in main rouitne
c      integer,allocatable:: imdim(:)                     !set in main routine

!! OUTPUT:  zmel for exchange=F, zmeltt for exchange=T.
      complex(8),allocatable :: zmel(:,:,:),zmeltt(:,:,:) !output

!! local save.
      real(8),private:: qbasinv(3,3),q_bk(3)=1d10,qk_bk(3)=1d0
      logical,private:: init=.true.
      complex(8),allocatable,private :: cphiq(:,:), cphim(:,:)
      real(8),allocatable,private :: rmelt(:,:,:),cmelt(:,:,:)
      integer,private::kxold=-9999

      contains
!! ------------------------------------
      subroutine get_zmelt(exchange,q,kx, kvec,irot,rkvec,kr,isp,  ngc,ngb,nmmax,nqmax, nctot,ncc)
!! Get <phiq(q,ncc+nqmax,ispq) |phim(q-rkvec,nctot+nmmax,ispm) MPB(rkvec,ngb)> ZO^-1
!!
!! ncc=0
!! kvec is in the IBZ, rk = Rot_irot(kvec), kx,kr are dummy.
!! \parameter all inputs
!! \parameter output=rmelt,clemt  matrix <MPB psi|psi>
      implicit none
      logical:: exchange
      integer:: kx,kr,isp,ngc,ngb,nmmax,nqmax,irot,ispq,ispm,nmini,nqini, nctot,ncc
      real(8) ::  quu(3),q(3), kvec(3),rkvec(3)
      ispq = isp
      ispm = isp
      nmini=1
      nqini=1
      call get_zmelt2(exchange,
     &   kvec,irot,rkvec,ngc,ngb,    !MPB        for  MPB_rkvec
     &      nmini,nmmax,ispm,nctot,  !middle-phi for  phi_{q-rkvec}
     &    q,nqini,nqmax,ispq,ncc  )  !end-phi    for  phi_q
      end subroutine get_zmelt
!! ----------------------------------------
cold  ntqxx--->nqmax
cold  nbmax -->nmmax
!!note: For usual correlation mode, I think nctot=0
!!note: For self-energy mode;   we calculate <iq1|\Sigma |iq2> , where iq1 and iq2 are in nqmax.
!!       nstate = nctot+nmmax
!!       allocate(zmeltt(ngb,    nstate,  nqmax))
!!       zmeltt= < MPB     phi   | phi   > (but true matrix elements are for <phi|phi MPB> (complex conjugate).
!!                <rkvec q-rkvec  |  q    >
 !                       cphim    | cphiq
!                        ispm     | ispq
!            nctot+  nmini:nmmax | ncc + nqini:nqmax
!                    middle state| end state
!
!!--- For dielectric funciton, we use irot=1 kvec=rkvec=q. We calulate \chi(q).
!!             q      rkvec     | q + rkvec
!                   nkmin:nkmax | nkqmin:nkqmax
!                  (we fix nkmin=1)
!           or
!              nt0=nkmax-nkmin+1 | ntp0=nkqmax-nkqmin+1
!                      1:nt0     | 1:ntp0
!                         occ    | unocc
!                      (cphi_k   | cphi_kq !in x0kf)
!                    middle state| end state
!
!!     rkvec= rk(:,k)-qq  ! <phi(q+rk,nqmax)|phi(rk,nctot+nmmax)  MPB(q,ngb )>
!!     kvec = rk(:,k)-qq  ! k
!!
!! NOTE: dimension
!!  nmtot = nctot+ nmmax-mnini+1
!!  nqtot = ncc  + nqmax-nqini+1
!! <rkvec,1:ngb   q-rkvec, 1:nmtot | q, 1:nqtot>
!! ----------------------------------------
      subroutine get_zmelt2(exchange,
     &   kvec,irot,rkvec,ngc,ngb,  !MPB for       MPB_rkvec
     &   nmini,nmmax,ispm,nctot,   !middle for    phi_{q-rkvec}
     &    q,nqini,nqmax,ispq,ncc)  !end state for phi_q
!! \parameter all inputs
!! \parameter output=rmelt,clemt  matrix <MPB psi|psi>
      implicit none
      logical:: exchange
      integer:: invr,nxx,itp,irot,isp,kr,no,nmmax,ngc,ngb,nqmax,nbcut
      integer:: iatomp(natom),nmini,nqini,nctot,ncc
      real(8) :: symope(3,3),shtv(3),tr(3,natom),qk(3),det
     &  , quu(3),q(3), kvec(3),rkvec(3),wtt
      complex(8),allocatable :: zzzmel(:,:,:),zw (:,:)
      integer:: nmtot,nqtot
      real(8),allocatable :: drealzzzmel(:,:,:), dimagzzzmel(:,:,:) ,ppb(:)
      logical:: debug=.false.
      complex(8),parameter:: img=(0d0,1d0),tpi= 8d0*datan(1d0)
      complex(8):: expikt(natom)
      integer:: it,ia,kx,verbose,nstate,imdim(natom)
      logical:: oncew
      real(8),parameter::tolq=1d-8
      integer::ispq,ispm,iii,itps
!TIME0_1001
      if(verbose()>80) debug=.true.
      if(debug) write(*,*) 'get_zmel in m_zmel: start'
      call getkeyvalue("GWinput","nbcutlow_sig",nbcut, default=0 )
      if(.not.done_genallcf_v3) call rx('m_zmel: not yet call genallcf_v3')
      if(.not.done_rdpp)        call rx('m_zmel: not yet call rdpp')
      if(.not.done_read_bzdata) call rx('m_zmel: not yet call read_bzdata')

      if(init) then
         call minv33(qbas,qbasinv)
         allocate( cphiq(nlmto,nband), cphim(nlmto,nband))
         init=.false.
      endif

      if(sum(abs(q-q_bk))>tolq) then
         call readcphi(q, nlmto,ispq, quu, cphim )
         cphiq(1:nlmto,1:ntq) = cphim(1:nlmto,itq(1:ntq))
         q_bk=q
      endif

      allocate( rmelt(ngb, nctot+nmmax, ncc+nqmax), ! nstate= nctot+nband
     &  cmelt(ngb, nctot+nmmax, ncc+nqmax))
      if(debug) write(*,*) 'get_zmel in m_zmel: 22222222'

!! qk = q-rk. rk is inside 1st BZ, not restricted to the irreducible BZ
      qk =  q - rkvec
      if(sum(abs(qk-qk_bk))>tolq) then
         call readcphi(qk, nlmto,ispm, quu, cphim)
         qk_bk=qk
      endif
c      call getsrdpp2( nclass,nl,nxx)
!! Rotate atomic positions invrot*R = R' + T
      invr  =  invg(irot)       !invrot (irot,invg,ngrp)
      tr    = tiat(:,:,invr)
      iatomp= miat(:,invr)
      symope= symgg(:,:,irot)
      shtv  = matmul(symope,shtvg(:,invr))
!! ppb= <Phi(SLn,r) Phi(SL'n',r) B(S,i,Rr)>
!! Note spin-dependence. Look for ixx==8 in hbas.m.F calling basnfp.F, which gives ppbrd.
      allocate( ppb(nlnmx*nlnmx*mdimx*nclass))
      ppb = ppbir(:,irot,ispq)
      if(debug) write(*,*) 'get_zmel in m_zmel: 3333333333'

!TIME1_1001 "init"
!TIME0_1101

!! phase factors expikt(ia) is for exp(ik.T(R))
      do ia = 1,natom
        imdim(ia)  = sum(nblocha(iclass(1:ia-1)))+1
        expikt(ia) = exp(img *tpi* sum(kvec*tr(:,ia)) )
      end do
      nmtot  = nctot + nmmax -nmini+1      ! = phi_middle
      nqtot  = ncc   + nqmax -nqini+1      ! = phi_end
      allocate( zzzmel(nbloch,nmtot,nqtot))
      zzzmel=0d0
!! MTO Core
      if(ncc>0.or.nctot>0) then
        call psicb_v3( nctot,ncc,nmmax,nqmax,iclass,expikt,
     i              cphim(1,nmini),  !middle phi
     i              cphiq(1,nqini),  !end phi
     i              ppb,!ppb,
     i              nlnmv,nlnmc,nblocha, !mdim,
     i              imdim,iatomp,
     i              mdimx,nlmto,nbloch,nlnmx,natom,nclass,
     i              icore,ncore,nl,nnc,
     o              zzzmel)
      endif
      if(debug) write(6,'("Goto psi2b_v3 nctot ncc nmmax nqmax=",4i4)') nctot,ncc,nmmax,nqmax
      if(debug) write(6,'("4444 zzzmelsum ",3i5,3d13.5)') nbloch,nmtot,nqtot,sum(abs(zzzmel)),sum(zzzmel)
!! MTO Valence
      if(nmmax*nqmax>0) then      ! val num of nm  ! val num of nq
        call psi2b_v3( nctot,ncc, nmmax-nmini+1,   nqmax-nqini+1, iclass,expikt, !phase,
     i              cphim(1,nmini),
     i              cphiq(1,nqini),
     i              ppb,! ppb,
     i              nlnmv, nlnmc,nblocha, !mdim,
     i              imdim,iatomp,
     d              mdimx,nlmto,nbloch,nlnmx, natom,nclass,
     o              zzzmel)
      endif
      if(debug) write(6,'("5555 zzzmelsum ",3i5,3d13.5)') nbloch,nmtot,nqtot,sum(abs(zzzmel)),sum(zzzmel)
!TIME1_1101 "psi2b_v3"

!TIME0_1201
!! IPW
      allocate(drealzzzmel(nbloch,nmtot,nqtot),dimagzzzmel(nbloch,nmtot,nqtot))
      drealzzzmel=dreal(zzzmel)
      dimagzzzmel=dimag(zzzmel)
      deallocate(zzzmel)
!    qk =  q - rkvec   !ncc+nqmax? nqtot?
      itps = nqini
      call drvmelp( q, nqmax-nqini+1, ! q     nt0 (in FBZ)
     i  qk,  nmmax-nmini+1,            ! q-rk  ntp0
     i  kvec,        ! k in IBZ for mixed product basis. rk = symope(kvec)
     i  ispq,ispm,ginv,
     i  ngc,ngcmx, ngpmx,nband,itq,
     i  symope, shtv, qbas, qbasinv,qibz,qbz,nqbz,nqibz,
     i  drealzzzmel, dimagzzzmel, nbloch, nctot,ncc,itps,
     o  rmelt,cmelt)
      if(debug) write(6,*) ' sxcf_fal1: end of drvmelp2 sum rmelt cmelt',sum(rmelt),sum(cmelt)
      deallocate(drealzzzmel,dimagzzzmel)
      if(verbose()>50) call timeshowx("5 after drvmelp")
      if(nbcut/=0.and.(.not.exchange)) then
         do it= nctot+1,nctot+min(nbcut,nmmax)
            rmelt(:, it,:) =0d0
            cmelt(:, it,:) =0d0
         enddo
      endif
!TIME1_1201 "drvmelp"

!! NOTE:=========================================
!! zmelt = rmelt(igb(rkvec), iocc(q), iunocc(q-rkvec)) + i* cmelt
!! iunocc: band index at target  q.
!! iocc:   band index at intermediate vector qk = q - rkvec
!! igb: index of mixed product basis       at rkvec (or written as rk)
!!   igb=1,ngb
!!   ngb=nbloch+ngc  ngb: # of mixed product basis
!!                   nbloch: # of product basis (within MTs)
!!                   ngc: # of IPW for the Screened Coulomb interaction.
!!                   igc is for given
!! See readgeig in drvmelp2.
!! ==================================================
c-------------------------------------------------------------------------
c$$$!! smbasis
c$$$!! smbasis ---need to fix this
!!  Read pomatr
c$$$      if(smbasis()) then  !this smbasis if block is from hsfp0.sc.m.F
c$$$         write(6,*)' smooth mixed basis : augmented zmel'
c$$$         ifpomat = iopen('POmat',0,-1,0) !oct2005
c$$$         nkpo = nqibz+nq0i
c$$$         nnmx=0
c$$$         nomx=0
c$$$         do ikpo=1,nkpo
c$$$            read(ifpomat) q_r,nn_,no,iqx !readin reduction matrix pomat
c$$$            if(nn_>nnmx) nnmx=nn_
c$$$            if(no>nomx) nomx=no
c$$$            allocate( pomat(nn_,no) )
c$$$            read(ifpomat) pomat
c$$$            deallocate(pomat)
c$$$         enddo
c$$$         isx = iclose("POmat")
c$$$         ifpomat = iopen('POmat',0,-1,0) !oct2005
c$$$         allocate( pomatr(nnmx,nomx,nkpo),qrr(3,nkpo),nor(nkpo),nnr(nkpo) )
c$$$         do ikpo=1,nkpo
c$$$            read(ifpomat) qrr(:,ikpo),nn_,no,iqx !readin reduction matrix pomat
c$$$            nnr(ikpo)=nn_
c$$$            nor(ikpo)=no
c$$$            read(ifpomat) pomatr(1:nn_,1:no,ikpo)
c$$$         enddo
c$$$         isx = iclose("POmat")
c$$$         write(6,*)"Read end of POmat ---"
c$$$      endif
c-------------------------------------
c$$$               if(smbasis()) then !
c$$$                  ntp0= nqmax
c$$$                  nn= nnr(kx)
c$$$                  no= nor(kx)
c$$$                  allocate( pomat(nn,no) )
c$$$                  pomat= pomatr(1:nn,1:no,kx)
c$$$                  if( sum(abs(kvec-qrr(:,kx)))>1d-10 .and.kx <= nqibz ) then
```

```
c$$$                     call rx( 'qibz/= qrr')
c$$$                  endif
c$$$                  if(no /= ngb.and.kx <= nqibz) then
c$$$!!     A bit sloppy check only for kx<nqibz because qibze is not supplied...
c$$$                     write(6,"(' q  ngb  ',3d13.5,3i5)")  kvec,ngb
c$$$                     write(6,"(' q_r  nn no',3d13.5,3i5)") q_r,nn,no
c$$$                     call rx( 'x0kf_v2h: POmat err no/=ngb')
c$$$                  endif
c$$$                  if(timemix) call timeshow("xxx2222 k-cycle")
c$$$                  ngb = nn      ! Renew ngb !!!
c$$$                  allocate ( zmel(nn, nctot+nmmax, ntp0) )
c$$$                  call matm( pomat, dcmplx(rmelt,cmelt), zmel,
c$$$     &                 nn, no, (nctot+nmmax)*ntp0 )
c$$$                  deallocate(rmelt, cmelt)
c$$$                  allocate( rmelt(ngb, nctot+nmmax, ntp0), !ngb is reduced.
c$$$     &                 cmelt(ngb, nctot+nmmax, ntp0) )
c$$$                  rmelt = dreal(zmel)
c$$$                  cmelt = dimag(zmel)
c$$$                  deallocate(zmel,pomat)
c$$$               else
c$$$                  nn=ngb
c$$$                  no=ngb
c$$$               endif

c      if( oncew() ) then
c         write(6,"('ngb nn no=',3i6)") ngb,nn,no
c      endif
c               if(timemix) call timeshow("22222 k-cycle")
      if(allocated(zzzmel))deallocate(zzzmel) !rmel,cmel)
      if(debug) write(6,*) ' sxcf: goto wtt'
      if(debug) write(6,"('sum of rmelt cmelt=',4d23.16)")sum(rmelt),sum(cmelt)
!! === End of zmelt ; we now have matrix element zmelt= rmelt + img* cmelt ===
!TIME0_1301

!! Multipled by ppovlz and reformat
      if(exchange) then
         if(debug) write(*,*) 'exchange mode 0000 ngb nmtot nqtot',ngb,nmtot,nqtot
         allocate( zmel(ngb, nmtot, nqtot))
         zmel = dcmplx(rmelt,cmelt)
         if(debug) write(*,*) 'exchange mode 1111'
         deallocate(rmelt,cmelt)
         if(debug) then
            do it = 1,nmtot
               write(6,"('wwwwwsc ',i5,2f10.4)") it,sum(abs(zmel(:,it,1)))
            enddo
            write(*,*) 'eeeeeeeeeeeee end of wwwwsc',nctot,nmmax
            write(6,*)'sumcheck ppovlz=',sum(abs(ppovlz(:,:)))
         endif
!! OUTPUT zmeltt for exchange
         allocate(zmeltt(nmtot,nqtot,ngb))

         if(verbose()>39) then
            write(*,*)'info: USE GEMM FOR SUM (zmeltt=zmel*ppovlz) in sxcf_fal2.sc.F'
            write(*,*)'zgemmsize',nqtot*nmtot,ngb,ngb
            write(*,*)'size ,zmel',size(zmel,dim=1),size(zmel,dim=2),size(zmel,dim=3)
            write(*,*)'size ,ppovlz',size(ppovlz,dim=1),size(ppovlz,dim=2)
            write(*,*)'size ,zmeltt',size(zmeltt,dim=1),size(zmeltt,dim=2),size(zmeltt,dim=3)
         endif
         call flush(6)
         call zgemm('T','N',nqtot*nmtot,ngb,ngb,(1d0,0d0),
     .     zmel,ngb,ppovlz,ngb,(0d0,0d0),zmeltt,nqtot*nmtot )
         deallocate(zmel)
      else
!! Correlation case. Get zmel
         if(debug) write(*,*) 'correlation mode 0000'
c         nstate = nctot + nmmax ! = nstate for the case of correlation
         allocate(zmeltt(ngb, nmtot, nqtot))
         zmeltt= dcmplx(rmelt,-cmelt) !zmeltt= <itp|it,ib>
         deallocate(rmelt,cmelt)
!! zmel(igb,it*itp) = C(ppovlz)*N(zmeltt(:,it*itp))
!!  C means Hermitian conjugate, N means normal
!! http://www.netlib.org/lapack/lapack-3.1.1/html/zgemm.f.html
!! OUTPUT
         allocate( zmel(ngb, nmtot, nqtot) )

         if(debug) write(6,'("4 zzzppp222aaa ",3d13.5)') sum(abs(zmeltt)),sum(zmeltt)
         call zgemm('C','N',ngb, nmtot*nqtot,ngb,(1d0,0d0),
     .     ppovlz, ngb, zmeltt,ngb, (0d0,0d0),zmel,ngb)
         deallocate(zmeltt)
         if(debug) write(*,*)'zz000 nmtot,ngb,nstate ',nmtot,ngb,nqtot
         if(debug) write(*,*)'zz000 sumchk zmel ',sum(abs(zmel(1:ngb,1:nmtot,1:nqtot)))
         if(debug) write(*,*) 'correlation mode end'
!TIME1_1301 "matmul_zmelp_povlz"
      endif
      end subroutine get_zmelt2
      end module m_zmel
      subroutine timeshowx(info)
      character*(*) :: info
      write(*,'(a,$)')info
      call cputid(0)
      end

```

4.15 gwsrc/mkjp.F File Reference

Functions/Subroutines

- subroutine vcoulq_4 (q, nbloch, ngc,nbas, lx, lxx, nx, nxx,alat, qlat, vol, ngvecc,strx, rojp, rojb, sgbb, sgpb, fouvb,nblochpmx, bas, rmax,eee, aa, bb, nr, nrx, rkpr, rkmr, rofi,
- subroutine mkjp_4 (q, ngc, ngvecc, alat, qlat, lxx, lx, nxx, nx, bas, a, b, rmax, nr, nrx, rprodx, eee, rofi, rkpr, rkmr, rojp, sgpb, fouvb)
- real(8) function fac2m (i)
- subroutine genjh (eee, nr, a, b, lx, nrx, lxx, rofi, rkpr, rkmr)
- subroutine mkjb_4 (lxx, lx, nxx, nx, a, b, nr, nrx, rprodx, rofi, rkpr, rkmr, rojb, sgbb)
- subroutine sigint_4 (rkp, rkm, kmx, a, b, nr, phi1, phi2, rofi, sig)
- subroutine intn_smpxxx (g1, g2, int, a, b, rofi, nr, lr0)
- subroutine sigintan1 (absqg, lx, rofi, nr, a1int)
- subroutine sigintpp (absqg1, absqg2, lx, rmax, sig)

4.15.1 Function/Subroutine Documentation

4.15.1.1 real(8) function fac2m (*i*)

Definition at line 629 of file mkjp.F.

Here is the caller graph for this function:

4.15.1.2 subroutine genjh (real(8) *eee,* integer(4) *nr,* real(8) *a,* real(8) *b,* integer(4) *lx,* integer(4) *nrx,* integer(4) *lxx,* real(8), dimension(nrx) *rofi,* real(8), dimension(nrx,0:lxx) *rkpr,* real(8), dimension(nrx,0:lxx) *rkmr*)

Definition at line 643 of file mkjp.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.15.1.3 subroutine intn_smpxxx (double precision, dimension(nr) *g1,* double precision, dimension(nr) *g2,* double precision, dimension(nr) *int,* double precision *a,* double precision *b,* double precision, dimension(nr) *rofi,* integer *nr,* integer *lr0*)

Definition at line 782 of file mkjp.F.

Here is the caller graph for this function:

4.15.1.4 subroutine mkjb_4 (integer(4) *lxx,* integer(4) *lx,* integer(4) *nxx,* integer(4), dimension(0:lxx) *nx,* real(8) *a,* real(8) *b,* integer(4) *nr,* integer(4) *nrx,* real(8), dimension(nrx,nxx,0:lxx) *rprodx,* real(8), dimension(nrx) *rofi,* real(8), dimension(nrx,0:lxx) *rkpr,* real(8), dimension(nrx,0:lxx) *rkmr,* real(8), dimension(nxx, 0:lxx) *rojb,* real(8), dimension(nxx, nxx, 0:lxx) *sgbb*)

Definition at line 681 of file mkjp.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.15.1.5 subroutine mkjp_4 (real(8), dimension(3) *q,* integer(4) *ngc,* integer(4), dimension(3,ngc) *ngvecc,* real(8) *alat,* real(8), dimension(3,3) *qlat,* integer(4) *lxx,* integer(4) *lx,* integer(4) *nxx,* integer(4), dimension(0:lxx) *nx,* real(8), dimension(3) *bas,* real(8) *a,* real(8) *b,* real(8) *rmax,* integer(4) *nr,* integer(4) *nrx,* real(8), dimension(nrx,nxx,0:lxx) *rprodx,* real(8) *eee,* real(8), dimension(nrx) *rofi,* real(8), dimension(nrx,0:lxx) *rkpr,* real(8), dimension(nrx,0:lxx) *rkmr,* complex(8), dimension(ngc, (lxx+1)∗∗2) *rojp,* complex(8), dimension(ngc, nxx, (lxx+1)∗∗2) *sgpb,* complex(8), dimension(ngc, nxx, (lxx+1)∗∗2) *fouvb*)

Definition at line 430 of file mkjp.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.15.1.6 subroutine sigint_4 (real(8), dimension(nr) *rkp,* real(8), dimension(nr) *rkm,* integer(4) *kmx,* real(8) *a,* real(8) *b,* integer(4) *nr,* real(8), dimension(nr) *phi1,* real(8), dimension(nr) *phi2,* real(8), dimension(nr) *rofi,* real(8) *sig*)

Definition at line 762 of file mkjp.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.15.1.7 subroutine sigintan1 (real(8) *absqg,* integer(4) *lx,* real(8), dimension(nr) *rofi,* integer(4) *nr,* real(8), dimension(nr,0:lx) *a1int*)

Definition at line 822 of file mkjp.F.

Here is the caller graph for this function:

4.15.1.8 subroutine sigintpp (real(8) *absqg1,* real(8) *absqg2,* integer(4) *lx,* real(8) *rmax,* real(8), dimension(0:lx) *sig*)

Definition at line 864 of file mkjp.F.

Here is the caller graph for this function:

4.15.1.9 subroutine vcoulq_4 (real(8), dimension(3) *q,* integer(4) *nbloch,* integer(4) *ngc,* integer(4) *nbas,* integer(4), dimension(nbas) *lx,* integer(4) *lxx,* integer(4), dimension(0:lxx,nbas) *nx,* integer(4) *nxx,* real(8) *alat,* real(8), dimension(3,3) *qlat,* real(8) *vol,* integer(4), dimension(3,ngc) *ngvecc,* complex(8), dimension((lxx+1)∗∗2, nbas, (lxx+1)∗∗2,nbas) *strx,* complex(8), dimension(ngc, (lxx+1)∗∗2, nbas) *rojp,* real(8), dimension(nxx, 0:lxx, nbas) *rojb,* real(8), dimension(nxx, nxx, 0:lxx, nbas) *sgbb,* complex(8), dimension(ngc, nxx, (lxx+1)∗∗2, nbas) *sgpb,* complex(8), dimension(ngc, nxx, (lxx+1)∗∗2, nbas) *fouvb,* integer(4) *nblochpmx,* real(8), dimension(3,nbas) *bas,* real(8), dimension(nbas) *rmax,* real(8) *eee,* real(8), dimension(nbas) *aa,* real(8), dimension(nbas) *bb,* integer(4), dimension(nbas) *nr,* integer(4) *nrx,* real(8), dimension(nrx,0:lxx,nbas) *rkpr,* real(8), dimension(nrx,0:lxx,nbas) *rkmr,* real(8), dimension(nrx,nbas) *rofi*)

Definition at line 1 of file mkjp.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.16 mkjp.F

```
00001       subroutine vcoulq_4(q,nbloch, ngc,
00002      &                  nbas, lx,lxx, nx,nxx,
00003      &                  alat, qlat, vol, ngvecc,
00004      &          strx,rojp,rojb, sgbb,sgpb, fouvb,   !sgpp,fouvp,
00005      i          nblochpmx,bas,rmax,
00006      i    eee, aa,bb,nr,nrx,rkpr,rkmr,rofi,
00007 !       These inputs are to generate sgpp on the fly.
00008      o          vcoul)
00009 Co Coulmb matrix for each q. -----------------------------------------------
00010 Ci strx:  Structure factors
00011 Ci nlx corresponds to (lx+1)**2 . lx corresponds to 2*lmxax.
00012 Ci rho-type integral
00013 Ci  ngvecc     : q+G vector
00014 Ci  rojp rojb  : rho-type integral
00015 ci  sigma-type onsite integral
00016 ci  Fourier
00017 Ci  nx(l,ibas) : max number of radial function index for each l and ibas.
00018 Ci               Note that the definition is a bit different from nx in basnfp.
00019 ci  nxx        : max number of nx among all l and ibas.
00020 ci  lx(nbas)   : max number of l for each ibas.
00021 ci  lxx        :
00022 ci
00023 ci  vol : cell vol
00024 c
00025 Co Vcoul
00026 cr vcoul is in a.u. You have to multiply e~2=2 if you want to it in Ry,
00027 cr    vcoul = 2d0*vcoul !  in Ry unit.
00028 c-----------------------------------------------------------------------
00029 c  rojp = <j_aL(r) | P(q+G)_aL > where
00030 c         |P(q+G)_aL> : the projection of exp(i (q+G) r) to aL channnel.
00031 c         |j_aL>      : \def r^l/(2l+1)!! Y_L.  The spherical bessel functions near r=0.  Energy-dependence
 is omitted.
00032 c
00033       use m_lldata,only: ll
00034       implicit none
00035       integer(4) :: nbloch, nblochpmx, nbas,
00036      &               lxx,lx(nbas), nxx, nx(0:lxx,nbas)
00037       real(8)    :: egtpi,vol,q(3),fpi
```

```

ci structure con
      complex(8) :: strx((lxx+1)**2, nbas, (lxx+1)**2,nbas)
ci |q+G|**2
      integer(4) :: ngc, ngvecc(3,ngc)
      real(8)    :: qlat(3,3),alat,absqg2(ngc),qg(3)

ci rho-type onsite integral
      complex(8) ::   rojp(ngc, (lxx+1)**2, nbas)
      real(8)    ::   rojb(nxx, 0:lxx, nbas)
ci sigma-type onsite integral
      real(8)    :: sgbb(nxx,  nxx,  0:lxx,      nbas)
      complex(8) :: sgpb(ngc,  nxx,  (lxx+1)**2, nbas)
c     &              ,sgpp(ngc,  ngc,  (lxx+1)**2, nbas)
ci Fourier
      complex(8) ::
     &             fouvb(ngc,  nxx, (lxx+1)**2, nbas)
Co
     &             ,vcoul(nblochpmx, nblochpmx)
c     &             ,fouvp(ngc,  ngc, (lxx+1)**2, nbas)

cinternals
      integer(4) :: ibl1, ibl2,ig1,ig2,ibas,ibas1,ibas2,
     &              l,m,n, n1,l1,m1,lm1,n2,l2,m2,lm2,ipl1,ipl2
      integer(4) :: ibasbl(nbloch), nbl(nbloch), lbl(nbloch),
     &              mbl(nbloch), lmbl(nbloch)
      real(8) :: pi, fpivol,tpiba
      complex(8) :: rojpstrx((lxx+1)**2,nbas)

c check
      complex(8),allocatable :: hh(:,:),oo(:,:),zz(:,:)
      real(8),allocatable    :: eb(:)

      complex(8),allocatable :: matp(:),matp2(:)
      complex(8) :: xxx
      integer(4) :: nblochngc,nev,nmx,ix
      logical :: ptest=.false. ! See ptest in basnfp.f

c-------------------
      real(8),   allocatable :: cy(:),yl(:)
      complex(8),allocatable :: pjyl_(:,:),phase(:,:)
      complex(8) :: img=(0d0,1d0)
      real(8):: bas(3,nbas),r2s,rmax(nbas)
      integer(4):: lm
c$$$#ifdef COMMONLL
c$$$      integer(4)::ll(51**2)
c$$$      common/llblock/ll
c$$$#else
c$$$      integer(4) :: ll
c$$$      external ll
c$$$#endif
      real(8)::  fkk(0:lxx),fkj(0:lxx),fjk(0:lxx),fjj(0:lxx),sigx(0:lxx),radsig(0:lxx)
      complex(8):: fouvp_ig1_ig2, fouvp_ig2_ig1, sgpp_ig1_ig2

      integer(4):: nrx,nr(nbas),ir,ig
      real(8):: eee , int1x(nrx),int2x(nrx),phi(0:lxx),psi(0:lxx)
     &  ,aa(nbas),bb(nbas),rkpr(nrx,0:lxx,nbas),rkmr(nrx,0:lxx,nbas)
     &  ,rofi(nrx,nbas)
      real(8), allocatable:: ajr(:,:,:,:), a1(:,:,:)
      logical :: debug=.false.
c--------------------------------------------------------------
      write(6,'(" vcoulq_4: nblochpmx  nbloch ngc=",3i6)') nblochpmx,nbloch,ngc
c     print *, ' sum fouvp=',sum(fouvp(:,:,:,1))
c     print *, ' sum fouvb=',sum(fouvb(:,:,:,1))
      pi    = 4d0*datan(1d0)
      fpi    = 4*pi
      fpivol = 4*pi*vol

c---for sgpp fouvp
      allocate(  !ajr(1:nr,0:lx,ngc),a1(1:nr,0:lx,ngc),rkpr(nr,0:lx),rkmr(nr,0:lx),
     &  pjyl_((lxx+1)**2,ngc),phase(ngc,nbas) )
      allocate(cy((lxx+1)**2),yl((lxx+1)**2))
      call sylmnc(cy,lxx)

c======================================================
      vcoul = 0d0
c-gvec
      tpiba = 2*pi/alat
      do ig1 = 1,ngc
        qg(1:3) = tpiba * (q(1:3)+ matmul(qlat, ngvecc(1:3,ig1)))
        absqg2(ig1)  = sum(qg(1:3)**2)
c---for spgg fourvp ----------
        do ibas=1,nbas
          phase(ig1,ibas) = exp( img*sum(qg(1:3)*bas(1:3,ibas))*alat  )
        enddo
        call sylm(qg/sqrt(absqg2(ig1)),yl,lxx,r2s) !spherical factor Y( q+G )
        do lm =1,(lxx+1)**2
```

```
          l = ll(lm)
          pjyl_(lm,ig1) = fpi*img**l *cy(lm)*yl(lm)  * sqrt(absqg2(ig1))**l  !*phase
          ! <jlyl | exp i q+G r> projection of exp(i q+G r) to jl yl  on MT
        enddo
c--------------
      enddo
c


c-- index (mx,nx,lx,ibas) order.
      ibl1 = 0
      do ibas= 1, nbas
        do l   = 0, lx(ibas)
c        write(6,'(" l ibas nx =",3i5)') l,nx(l,ibas),ibas
          do n   = 1, nx(l,ibas)
            do m   = -l, l
              ibl1  = ibl1 + 1
              ibasbl(ibl1) = ibas
              nbl(ibl1) = n
              lbl(ibl1) = l
              mbl(ibl1) = m
              lmbl(ibl1) = l**2 + l+1 +m
c        write(6,*)ibl1,n,l,m,lmbl(ibl1)
            enddo
          enddo
        enddo
      enddo
      if(ibl1/= nbloch) then
        write(6,*)' ibl1 nbloch',ibl1, nbloch
Cstop2rx 2013.08.09 kino        stop ' vcoulq: error ibl1/= nbloch'
        call rx( ' vcoulq: error ibl1/= nbloch')
      endif


c-- <B|v|B> block
c      write(6,*)' vcoulq: bvb block xxx rojbsum='
c      write(6,*) sum(rojb(:,:,1))
c      write(6,*) sum(rojb(:,:,2))
c      write(6,*) sum(rojb(:,:,3))
c      write(6,*) sum(rojb(:,:,4))
c      write(6,*)' vcoulq: bvb block xxx sgbbbsum='
c      write(6,*) sum(sgbb(:,:,:,1))
c      write(6,*) sum(sgbb(:,:,:,2))
c      write(6,*) sum(sgbb(:,:,:,3))
c      write(6,*) sum(sgbb(:,:,:,4))
      do ibl1= 1, nbloch
        ibas1= ibasbl(ibl1)
        n1   = nbl(ibl1)
        l1   = lbl(ibl1)
        m1   = mbl(ibl1)
        lm1  = lmbl(ibl1)
        do ibl2= 1, ibl1
          ibas2= ibasbl(ibl2)
          n2   = nbl(ibl2)
          l2   = lbl(ibl2)
          m2   = mbl(ibl2)
          lm2  = lmbl(ibl2)
          vcoul(ibl1,ibl2) =
     &     rojb(n1, l1, ibas1) *strx(lm1,ibas1,lm2,ibas2)
     &    *rojb(n2, l2, ibas2)
          if(ibas1==ibas2 .and. lm1==lm2) then
            vcoul(ibl1,ibl2) = vcoul(ibl1,ibl2) + sgbb(n1,n2,l1, ibas1)
            ! sigma-type contribution. onsite coulomb
          endif
        enddo
      enddo

ccccccccccccccccccccccccccc
c      goto 1112
ccccccccccccccccccccccccccc

c <P_G|v|B>
      if(debug) write(6,*)' vcoulq_4: pgvb block 1111'
      do ibl2= 1, nbloch
        ibas2= ibasbl(ibl2)
        n2   = nbl(ibl2)
        l2   = lbl(ibl2)
        m2   = mbl(ibl2)
        lm2  = lmbl(ibl2)
        do ig1 = 1,ngc
          ipl1 = nbloch + ig1
          vcoul(ipl1,ibl2) = fouvb(ig1,  n2, lm2, ibas2)

          do ibas1= 1, nbas
            do lm1  = 1, (lx(ibas1)+1)**2
              vcoul(ipl1,ibl2) = vcoul(ipl1,ibl2) -
     &      dconjg(rojp(ig1, lm1, ibas1)) *strx(lm1,ibas1,lm2,ibas2)
     &      *rojb(n2, l2, ibas2)
              if(ibas1==ibas2 .and.lm1==lm2) then
                vcoul(ipl1,ibl2) = vcoul(ipl1,ibl2) -
     &        sgpb(ig1, n2, lm2, ibas2)
              endif
            enddo
          enddo
        enddo
      enddo

      if(debug) write(6,*)' vcoulq_4: ajr allocate'
C... prepare funciton ajr and a1.
C... ajr:spherical bessel, a1: integral of (sperical bseel)*(rkp rkm)
c------------------
      allocate( ajr(nrx,0:lxx,nbas,ngc), a1(nrx,0:lxx,nbas) )
      if(debug) write(6,*)' vcoulq_4: end ajr allocate'
      do ig1 = 1,ngc
        do ibas= 1,nbas
          if(debug) write(6,"('ccc: ',10i15)")ig1,ibas
          do ir = 1,nr(ibas)
            call bessl(absqg2(ig1)*rofi(ir,ibas)**2,lxx,phi,psi)
            do l  = 0,lx(ibas)

              if(debug.and.ig==162.and.ibas==8) then
                write(6,"('ccc: ',10i15)")ig1,ibas,ir,l
                write(6,*)"ccc:", phi(l)
                write(6,*)"ccc:", rofi(ir,ibas)
              endif

              ajr(ir,l,ibas,ig1) = phi(l)* rofi(ir,ibas) **(l +1 )
              ! ajr = j_l(sqrt(e) r) * r / (sqrt(e))**l
            enddo
          enddo
        enddo
      enddo
c------------------

c <P_G|v|P_G>
      if(debug) write(6,*)' vcoulq_4: pgvpg block'
      do ig1 = 1,ngc
        ipl1 = nbloch + ig1
        rojpstrx = 0d0
        do ibas1= 1, nbas
          do lm1  = 1, (lx(ibas1)+1)**2
            do ibas2= 1, nbas
              do lm2  = 1, (lx(ibas2)+1)**2
                rojpstrx(lm2, ibas2) = rojpstrx(lm2, ibas2)+
     &      dconjg(rojp(ig1, lm1, ibas1)) *strx(lm1,ibas1,lm2,ibas2)
              enddo
            enddo
          enddo
        enddo

c----------------------
        do ibas=1,nbas
          do l = 0,lx(ibas)
            call intn_smpxxx( rkpr(1,l,ibas), ajr(1,l,ibas,ig1),int1x
     &         ,aa(ibas),bb(ibas),rofi(1,ibas),nr(ibas),0)
            call intn_smpxxx( rkmr(1,l,ibas), ajr(1,l,ibas,ig1),int2x
     &         ,aa(ibas),bb(ibas),rofi(1,ibas),nr(ibas),0)
            a1(1,         l,ibas) = 0d0
            a1(2:nr(ibas),l,ibas) =
     &             rkmr(2:nr(ibas),l,ibas) *( int1x(1)-int1x(2:nr(ibas)) )
     &           + rkpr(2:nr(ibas),l,ibas) *  int2x(2:nr(ibas))
          enddo
        enddo
c---------------------

        do ig2 = 1,ig1
          ipl2 = nbloch + ig2
          if(ig1==ig2) vcoul(ipl1,ipl2) = fpivol/(absqg2(ig1) -eee) !eee is negative
          do ibas2= 1, nbas
c... for fouvp and sgpp -------
            call wronkj( absqg2(ig1), absqg2(ig2), rmax(ibas2),lx(ibas2),
     o                 fkk,fkj,fjk,fjj)

            if(eee==0d0) then
              call sigintpp( absqg2(ig1)**.5, absqg2(ig2)**.5, lx(ibas2), rmax(ibas2),
     o             sigx)
            else
              do l = 0,lx(ibas2)
                call gintxx(a1(1,l,ibas2), ajr(1,l,ibas2,ig2)
     &                   ,aa(ibas2),bb(ibas2),nr(ibas2), sigx(l))
              enddo
            endif
            do l = 0,lx(ibas2)
              radsig(l) = fpi/(2*l+1) * sigx(l)
            enddo

c-------------------------------
            do lm2  = 1, (lx(ibas2)+1)**2
              l= ll(lm2)
c...fouvp sgpp-----------
              fouvp_ig1_ig2 = fpi/(absqg2(ig1)-eee) *dconjg(pjyl_(lm2,ig1)*phase(ig1,ibas2))
     &      * (-fjj(l)) * pjyl_(lm2,ig2)*phase(ig2,ibas2)
              fouvp_ig2_ig1 = fpi/(absqg2(ig2)-eee) *dconjg(pjyl_(lm2,ig2)*phase(ig2,ibas2))
     &      * (-fjj(l)) * pjyl_(lm2,ig1)*phase(ig1,ibas2)
              sgpp_ig1_ig2  = dconjg(pjyl_(lm2,ig1)*phase(ig1,ibas2))*radsig(l)
     &                      * pjyl_(lm2,ig2)*phase(ig2,ibas2)
c-------------------------
              vcoul(ipl1,ipl2) = vcoul(ipl1,ipl2)
     &       +  rojpstrx(lm2,ibas2)*rojp(ig2, lm2, ibas2)
c     &      -  dconjg( fouvp(ig2,  ig1, lm2, ibas2)) !BugFix Mar5-01 It was dcmplx.
c     &      -          fouvp(ig1,  ig2, lm2, ibas2)
c     &      +  sgpp(ig1, ig2, lm2, ibas2)
     &       -  dconjg( fouvp_ig2_ig1 )
     &       -          fouvp_ig1_ig2
     &       +  sgpp_ig1_ig2
            enddo
          enddo
        enddo
      enddo
ccccccccccccccccccccccccccccccc
c 1112 continue
ccccccccccccccccccccccccccccccc


c-- Right-Upper part of vcoul.
      if(debug) write(6,*)' vcoulq_4: right-upper'
      do ipl1=1, nbloch+ngc
        do ipl2=1, ipl1-1
          vcoul(ipl2,ipl1) = dconjg(vcoul(ipl1,ipl2))
        enddo
      enddo

cccccccccccccccccccccccccccccc
c test.xxxxxxxxxx
c$$$      do ibl2= 1, nbloch
c$$$        ibas2= ibasbl(ibl2)
c$$$        n2   = nbl (ibl2)
c$$$        l2   = lbl (ibl2)
c$$$        m2   = mbl (ibl2)
c$$$        lm2  = lmbl(ibl2)
c$$$        if(l2==1.and.ibas2>2) then
c$$$          vcoul(nbloch+1:nbloch+ngc, ibl2) = 0d0
c$$$          vcoul(ibl2, nbloch+1:nbloch+ngc) = 0d0
c$$$        endif
c$$$      enddo
ccccccccccccccccccccccccccc

c vcoul is in a.u. You have to multiply e~2=2 if you want to it in Ry,
c     vcoul = 2d0*vcoul !  in Ry unit.
c

c check write
      do ix = 1,nbloch+ngc,20
        write(6,"(' Diagonal Vcoul =',i5,2d18.10)") ix,vcoul(ix,ix)
      enddo
      if( allocated(yl)   ) deallocate(yl)
      if( allocated(cy)   ) deallocate(cy)
      if( allocated(phase)) deallocate(phase)
      if( allocated(pjyl_)) deallocate(pjyl_)
      if(.not.ptest) return



cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c! Below ia a plane-wave test.
c--- check! Coulomb by plane wave expansion.
      write(6,*) ' --- plane wave Coulomb matrix check 1---- '
      write(197,*) ' --- off diagonal ---- '
      nblochngc = nbloch+ngc
      allocate(matp(nblochngc),matp2(nblochngc))
      do ig1 = 1,ngc
        matp = 0d0
        do ibl2= 1, nbloch
          ibas2= ibasbl(ibl2)
          n2   = nbl(ibl2)
          l2   = lbl(ibl2)
          m2   = mbl(ibl2)
          lm2  = lmbl(ibl2)
          matp(ibl2) = fouvb(ig1, n2, lm2, ibas2)*absqg2(ig1)/fpi
        enddo
        matp(nbloch+ig1) = 1d0
        ig2=ig1
c      do ig2 = 1,ngc !off diagnal
        matp2 = 0d0
        do ibl2= 1, nbloch
          ibas2= ibasbl(ibl2)
          n2   = nbl(ibl2)
          l2   = lbl(ibl2)
          m2   = mbl(ibl2)
          lm2  = lmbl(ibl2)
          matp2(ibl2) = fouvb(ig2, n2, lm2, ibas2)*absqg2(ig2)/fpi
        enddo
        matp2(nbloch+ig2) = 1d0
        xxx= sum(
     &    matmul(matp(1:nblochngc),vcoul(1:nblochngc,1:nblochngc))
     &             *dconjg(matp2(1:nblochngc))  )
        if(ig1/=ig2) then  !off diagnal
          if(abs(xxx)>1d-1 ) then
            write(197,'(2i5, 2d13.6)') ig1,ig2, xxx
            write(197,'("    matpp ", 2d13.6)')
     &      vcoul(nbloch+ig1,nbloch+ig2)
            write(197,*)
          endif
        else
          write(196,'(2i5," exact=",3d13.6,"q ngsum=",3f8.4,i5)')
     &       ig1,ig2,fpi*vol/absqg2(ig1)
     &    , fpi*vol/absqg2(ig2),absqg2(ig1), q(1:3)
     &    , sum(ngvecc(1:3,ig1)**2)
          write(196,'("           cal  =", 2d13.6)') xxx
          write(196,'("           vcoud=", 2d13.6)')
     &       vcoul(nbloch+ig1,nbloch+ig2)
          write(196,*)
        endif
c      enddo !off diagnal
      enddo
c
      deallocate(matp,matp2)
c      stop ' *** ptest end *** See fort.196 and 197'
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      end




c==========================================================================
      subroutine mkjp_4( q,ngc,ngvecc, alat, qlat, lxx,lx,nxx,nx,
     i                   bas, a,b,rmax,nr,nrx,rprodx,
     i    eee,rofi,rkpr,rkmr,
     o        rojp,sgpb,fouvb)
C- Make integrals in each MT. and the Fourier matrix.
Cr the integrals rojp, fouvb,fouvp
Cr are for  J_L(r)= j_l(sqrt(e) r)/sqrt(e)**l Y_L,
Cr which behaves as r^l/(2l+1)!! near r=0.
Cr
Cr oniste integral is based on
Cr 1/|r-r'| = \sum 4 pi /(2k+1) \frac{r_<^k }{ r_>^{k+1} } Y_L(r) Y_L(r')
Cr See PRB34 5512(1986) for sigma type integral
Cr
      use m_lldata,only: ll
      implicit none
      integer(4) :: ngc,ngvecc(3,ngc), lxx, lx, nxx,nx(0:lxx),nr,nrx
      real(8)    :: q(3),bas(3), rprodx(nrx,nxx,0:lxx),a,b,rmax,alat,
     &              qlat(3,3)
ci rho-type onsite integral
      complex(8) :: rojp(ngc, (lxx+1)**2)
ci sigma-type onsite integral
      complex(8) :: sgpb(ngc,  nxx,  (lxx+1)**2)
c     &              sgpp(ngc,  ngc,  (lxx+1)**2)
      real(8),allocatable::cy(:),yl(:)
ci Fourier
      complex(8) ::
     &              fouvb(ngc,  nxx, (lxx+1)**2)
c     &              fouvp(ngc,  ngc, (lxx+1)**2)
c internal
      integer(4) :: nlx,ig1,ig2,l,n,ir,n1,n2,lm !, ibas
c$$$#ifdef COMMONLL
c$$$      integer(4)::ll(51**2)
c$$$      common/llblock/ll
c$$$#else
c$$$      integer(4) :: ll
c$$$      external ll
c$$$#endif
      real(8)    :: pi,fpi,tpiba, qg1(3),
     &  fkk(0:lx),fkj(0:lx),fjk(0:lx),fjj(0:lx),absqg1,absqg2,
     &  fac,radint,radsigo(0:lx),radsig(0:lx),phi(0:lx),psi(0:lx)
     &   ,r2s,sig,sig1,sig2,sigx(0:lx),sig0(0:lx) ,qg2(3)
      complex(8) :: img =(0d0,1d0),phase
      complex(8),allocatable :: pjyl(:,:)
      real(8),allocatable ::ajr(:,:,:),a1(:,:,:), !rkpr(:,:),rkmr(:,:),
     &  qg(:,:),absqg(:)


      real(8):: rofi(nrx),rkpr(nrx,0:lxx),rkmr(nrx,0:lxx),eee
      logical :: debug=.false.
c rkpr(nr,0:lx),rkmr(nr,0:lx),
c-------------------------------------------------
      if(debug) print *,' mkjp_4:'
      nlx = (lx+1)**2
      allocate(ajr(1:nr,0:lx,ngc),a1(1:nr,0:lx,ngc),
     &  qg(3,ngc),absqg(ngc),
     &  pjyl((lx+1)**2,ngc) )

      pi    = 4d0*datan(1d0)
      fpi   = 4*pi
      tpiba = 2*pi/alat
      allocate(cy((lx+1)**2),yl((lx+1)**2))
      call sylmnc(cy,lx)
!      print *,' mkjp_4: end of sylmnc'
C... q+G and <J_L | exp(i q+G r)>  J_L= j_l/sqrt(e)**l Y_L
      do ig1 = 1,ngc
        qg(1:3,ig1) = tpiba * (q(1:3)+ matmul(qlat, ngvecc(1:3,ig1)))
        qg1(1:3) = qg(1:3,ig1)
        absqg(ig1)  = sqrt(sum(qg1(1:3)**2))
        absqg1   = absqg(ig1)
        phase = exp( img*sum(qg1(1:3)*bas(1:3))*alat  )
        call sylm(qg1/absqg1,yl,lx,r2s) !spherical factor Y( q+G )
        do lm =1,nlx
          l = ll(lm)
          pjyl(lm,ig1) = fpi*img**l *cy(lm)*yl(lm) *phase  *absqg1**l
          ! <jlyl | exp i q+G r> projection of exp(i q+G r) to jl yl  on MT
        enddo
      enddo

cc rofi and aj = r**l / (2l+1)!! \times r. Sperical Bessel at e=0.
c      rofi(1) = 0d0
c      do ir   = 1, nr
c        rofi(ir) = b*( exp(a*(ir-1)) - 1d0)
c      enddo
c      do l = 0,lx
c        rkpr(1:nr,l) = rofi(1:nr)**(l      +1 )
c        rkmr(2:nr,l) = rofi(2:nr)**(-l-1   +1 )
c        rkmr(1,l)    = rkmr(2,l)
c      enddo

c rojp
      if(debug) print *,' mkjp_4: rojp'
      do ig1 = 1,ngc
        call wronkj( absqg(ig1)**2, eee, rmax,lx,
     o                    fkk,fkj,fjk,fjj)
        do lm = 1,nlx
          l = ll(lm)
          rojp(ig1,lm) = (-fjj(l))* pjyl(lm,ig1)
        enddo
      enddo

c ajr
      do ig1 = 1,ngc
        do ir  = 1,nr
          call bessl(absqg(ig1)**2*rofi(ir)**2,lx,phi,psi)
          do l   = 0,lx
            ajr(ir,l,ig1) = phi(l)* rofi(ir) **(l +1 )
            ! ajr = j_l(sqrt(e) r) * r / (sqrt(e))**l
          enddo
ccccccccccccccccccccccccccc
c        write(116,'(i3,10d13.6)') ir, rofi(ir), ajr(ir,0:lx,ig1)
ccccccccccccccccccccccccccc
        enddo
ccccccccccccccccccccccccccc
c        write(6,*) ig1,sum(ajr(1:nr,0:lx,ig1))
ccccccccccccccccccccccccccc
      enddo

c-------------------------
      if(eee==0d0) then
c        print *,' mkjp_4: use sigintAn1 eee=0(r0c=infty) mode'
        do ig1 = 1,ngc
          call sigintan1( absqg(ig1), lx, rofi, nr
     o                  ,a1(1:nr, 0:lx,ig1) )
        enddo
c      else
c We need to impliment a version of sigintAn1 to treat eee/=0 case...
      endif

c-------------------------
c sgpb
```

```
      do ig1 = 1,ngc
        do lm  = 1,nlx
          l = ll(lm)
          do n =1,nx(l)                      ! r jl        , r B(r)
            if(eee==0d0) then
              call gintxx(a1(1,l,ig1),rprodx(1,n,l),a,b,nr, sig )
ccccccccccccccccccc
c        write(6,"( ' sgpb= ',3i5,2d14.6)") ig1,n,lm, sgpb(ig1,n,lm)
ccccccccccccccccccc
            else !for a while, we use this version of sgpb
              call sigint_4(rkpr(1,l),rkmr(1,l), lx,a,b,nr, ajr(1,l,ig1),rprodx(1,n,l)
     &            , rofi, sig)
            endif
            sgpb(ig1,n,lm) = dconjg(pjyl(lm,ig1))* sig/(2*l+1)*fpi
ccccccccccccccccccc
c        write(6,"( ' sgpb= ',3i5,2d14.6)") ig1,n,lm, sgpb(ig1,n,lm)
c        write(6,*)
ccccccccccccccccccc
          enddo
        enddo
      enddo
ccccccccccccccccccccccccc
c      stop 'test end======================'
cccccccccccccccccccccccc

c---------------------------------------
c sgpp block------->removed
c---------------------------------------

c Fourier
c fouvb
      if(debug) print *,' mkjp_4: Four'
      fouvb=0d0
      do ig1 = 1,ngc
        do lm  = 1,nlx
          l = ll(lm)
          do n =1,nx(l)
cccccccccccccccccccccccccccccccccccccccccccccc
c         print *,' ig1 lm l n=',ig1,lm,l,n
cccccccccccccccccccccccccccccccccccccccccccccc
            call gintxx(ajr(1,l,ig1), rprodx(1,n,l), a,b,nr,
     o                 radint )
cccccccccccccccccccccccccccccccccccccccccccccc
c         print *,' radint=',radint
cccccccccccccccccccccccccccccccccccccccccccccc
            fouvb(ig1, n, lm) =
     &    fpi/(absqg(ig1)**2-eee) *dconjg(pjyl(lm,ig1))*radint !eee is supposed to be negative

          enddo
        enddo
      enddo
cccccccccccccccccccc
c        write(6,*)' fourvb sum=',sum (fouvb)
cccccccccccccccccccc

c------------------------------
c fouvp block --->removed
c------------------------------

      deallocate(ajr,a1,   qg,absqg,   pjyl)
      if (allocated( cy )) deallocate(cy)
      if (allocated( yl )) deallocate(yl)
      end






      real(8) function fac2m(i)
cC A table of (2l-1)!!
c     data fac2l /1,1,3,15,105,945,10395,135135,2027025,34459425/
      logical,save::  init=.true.
      real(8),save:: fac2mm(0:100)
      if(init) then
        fac2mm(0)=1d0
        do l=1,100
          fac2mm(l)=fac2mm(l-1)*(2*l-1)
        enddo
      endif
      fac2m=fac2mm(i)
      end
c=====================================================================
      subroutine genjh(eee,nr,a,b,lx, nrx,lxx,
     o      rofi,rkpr,rkmr)
C-- Generate radial mesh rofi, spherical bessel, and hankel functions
Cr  rkpr, rkmr are real fucntions --
```

```
ci eee=E= -kappa**2 <0
cr      rkpr = (2l+1)!! * j_l(i sqrt(abs(E)) r) * r / (i sqrt(abs(E)))**l
cr      rkmr = (2l-1)!! * h_l(i sqrt(abs(E)) r) * r * i*(i sqrt(abs(E)))**(l+1)
cr rkpr reduced to be r**l*r      at E \to 0
cr rkmr reduced to be r**(-l-1)*r at E \to 0
C-------------------------------------------------------------
      implicit none
      integer(4):: nr,lx, nrx,lxx,ir,l
      real(8):: a,b,eee,psi(0:lx),phi(0:lx)
      real(8):: rofi(nrx),rkpr(nrx,0:lxx),rkmr(nrx,0:lxx),fac2m
      rofi(1)    = 0d0
      do ir      = 1, nr
        rofi(ir) = b*( exp(a*(ir-1)) - 1d0)
      enddo
      if(eee==0d0) then
        do l = 0,lx
          rkpr(1:nr,l) = rofi(1:nr)**(l +1)
          rkmr(2:nr,l) = rofi(2:nr)**(-l-1 +1)
          rkmr(1,l)    = rkmr(2,l)
        enddo
      else
        do ir  = 1, nr
          call bessl(eee*rofi(ir)**2,lx,phi(0:lx),psi(0:lx))
          do l = 0,lx    !fac2m(l)= (2l-1)!!
c            print *,' phi=',l,phi(l),phi(l)*fac2m(l+1)
c            print *,' psi=',l,psi(l),psi(l)/fac2m(l)
            rkpr(ir,l) = phi(l)* rofi(ir)**(l +1) *fac2m(l+1)
            if(ir/=1) rkmr(ir,l) = psi(l)* rofi(ir) **(-l ) /fac2m(l)
          enddo
        enddo
        rkmr(1,0:lx) = rkmr(2,0:lx)
      endif
      end
c==============================================================
      subroutine mkjb_4( lxx,lx,nxx,nx,
     i                 a,b,nr,nrx,rprodx,
     i      rofi,rkpr,rkmr,
     o        rojb,sgbb)
C--make integrals in each MT. and the Fourier matrix.
      implicit none
      integer(4) :: lxx, lx, nxx, nx(0:lxx),nr,nrx
      real(8)    :: q(3), rprodx(nrx,nxx,0:lxx),a,b
ci rho-type onsite integral
      real(8)    :: rojb(nxx, 0:lxx)
ci sigma-type onsite integral
      real(8)    :: sgbb(nxx, nxx, 0:lxx)
c internal
      integer(4) :: l,n,ir,n1,n2,l1
      real(8)    ::
     &  fac, xxx,fpi,pi,sig
      real(8) :: rofi(nrx),rkpr(nrx,0:lxx),rkmr(nrx,0:lxx)
      pi    = 4d0*datan(1d0)
      fpi   = 4*pi
c      real(8),allocatable :: rkpr(:,:),rkmr(:,:)
c
c      allocate(rkpr(nr,0:lx),rkmr(nr,0:lx))
c-------------------------------------------------
c rofi and aj = r**l / (2l+1)!! \times r. Sperical Bessel at e=0.
cccccccccccccccccccccccccccccccccc
c      do l = 0,lx
c      do n = 1,nx(l)
c      do n1 = 1,nx(l)
c        call gintxx(rprodx(1:nr,n,l), rprodx(1:nr,n1,l), a,b,nr,
c     o                 xxx )
c      write(6,*)' check rprodx =',l,n,n-n1,xxx
c      enddo
c      enddo
c      enddo
c      stop 'xxx'
cccccccccccccccccccccccccccccccc

c      rofi(1)    = 0d0
c      do ir      = 1, nr
c        rofi(ir) = b*( exp(a*(ir-1)) - 1d0)
c      enddo
c      do l = 0,lx
c        rkpr(1:nr,l) = rofi(1:nr)**(l +1)
c        rkmr(2:nr,l) = rofi(2:nr)**(-l-1) *rofi(2:nr)
c        rkmr(1,l)    = rkmr(2,l)
c      enddo

C... initialize
      rojb=0d0
      sgbb=0d0
c rojb
      fac = 1d0
      do l = 0,lx
```

```
        fac = fac/(2*l+1)
        do n = 1,nx(l)
          call gintxx(rkpr(1,l), rprodx(1,n,l), a,b,nr,
     o                 rojb(n,l) )
        enddo
        rojb(1:nx(l),l) = fac*rojb(1:nx(l),l)
      enddo
c sgbb
      do l  = 0,lx
        do n1 = 1,nx(l)
          do n2 = 1,nx(l)
            call sigint_4(rkpr(1,l),rkmr(1,l),lx,a,b,nr,rprodx(1,n1,l),rprodx(1,n2,l)
     &            , rofi,sig )
            sgbb(n1, n2, l)=sig/(2*l+1)*fpi
          enddo
        enddo
      enddo
c      write(6,*) ' rojbsum=', sum(rojb(:,:)),   sum(abs(rojb(:,:)))
c      write(6,*) ' sgbbsum=', sum(sgbb(:,:,:)), sum(abs(sgbb(:,:,:)))
ccccccccccccccccccccccccccccccccccccc
c      write(6,*)' sigint 1 1 0=',sgbb(1, 1, 0) !/(16d0*datan(1d0))
c      sgbb(1, 1, 0) =0d0
cccccccccccccccccccccccccccccccccccccccc
c      deallocate(rkpr,rkmr)
      end


c-------------------------------------------------------------
      subroutine sigint_4(rkp,rkm,kmx,a,b,nr,phi1,phi2,rofi, sig)
      implicit none
      integer(4) :: nr,kmx,k,ir
      real(8):: a,b, a1(nr),a2(nr),b1(nr),rkp(nr),rkm(nr),
     &    int1x(nr),int2x(nr), phi1(nr), phi2(nr),rofi(nr),sig
      real(8),parameter:: fpi = 4d0*3.14159265358979323846d0
c
      a1(1) = 0d0;  a1(2:nr) = rkp(2:nr)
      a2(1) = 0d0;  a2(2:nr) = rkm(2:nr)
      b1(1:nr) = phi1(1:nr)
      call intn_smpxxx(a1,b1,int1x,a,b,rofi,nr,0)
      call intn_smpxxx(a2,b1,int2x,a,b,rofi,nr,0)
c
      a1(1) = 0d0; a1(2:nr) =
     &  rkm(2:nr) *( int1x(1)-int1x(2:nr) )+ rkp(2:nr) * int2x(2:nr)
      b1(1:nr) = phi2(1:nr)
      call gintxx(a1,b1,a,b,nr, sig )
      end

c---------------------------------------------------------------
      subroutine intn_smpxxx(g1,g2,int,a,b,rofi,nr,lr0)
c-- intergral of two wave function. used in ppdf
c
c int(r) = \int_(r)^(rmax) u1(r') u2(r') dr'
c
c lr0 dummy index, now not used.
c simpson rule ,and with higher rule for odd devision.
c --------------------------------------------------------------
      IMPLICIT none
      integer nr,ir,lr0
      double precision g1(nr),g2(nr),int(nr),a,b,rofi(nr),w1,w2,w3
     &    ,ooth,foth
      data ooth,foth/0.33333333333333333,1.3333333333333333333/
      data w1,w2,w3/0.41666666666666666,0.66666666666666666666,
     &              -0.083333333333333333/
      if(mod(nr,2).eq.0)
Cstop2rx 2013.08.09 kino     &  stop ' INTN: nr should be odd for simpson integration rule'
     &  call rx( ' INTN: nr should be odd for simpson integration rule')
c
      int(1)=0.0d0
      DO  10  ir = 3,nr,2
        int(ir)=int(ir-2)
     &             + ooth*g1(ir-2)*g2(ir-2)*( a*(b+rofi(ir-2)) )
     &             + foth*g1(ir-1)*g2(ir-1)*( a*(b+rofi(ir-1)) )
     &             + ooth*g1(ir)*g2(ir)*( a*(b+rofi(ir)) )
   10 CONTINUE

c At the value for odd points, use the same interpolation above
      do 20 ir = 2,nr-1,2
        int(ir)=int(ir-1)
     &             + w1*g1(ir-1)*g2(ir-1)*( a*(b+rofi(ir-1)) )
     &             + w2*g1(ir)  *g2(ir)*  ( a*(b+rofi(ir)  ) )
     &             + w3*g1(ir+1)*g2(ir+1)*( a*(b+rofi(ir+1)) )
   20 continue
      do ir=1,nr
        int(ir)=int(nr)-int(ir)
      enddo
      END

```

```
c------------------------------------------------------------------
      subroutine sigintan1( absqg, lx, rofi, nr,
     o              a1int)
c a1int(r') = r' * \int_0^a r^2 {r_{<}}^l / (r_{>})^{l+1} *
c                j_l(absqg r)/absqg**l
      implicit none
      integer(4) :: nr,l,ir,lx
      real(8):: a1int(nr,0:lx), rofi(nr),absqg
      real(8)::
     &   ak(0:lx) ,aj(0:lx), dk(0:lx), dj(0:lx),
     &   aknr(0:lx),ajnr(0:lx),dknr(0:lx),djnr(0:lx),
     &   phi(0:lx),psi(0:lx)
c---
c      print *,' sigintAn1: absqg=',absqg
      if(absqg<1d-10) then
c      if(absqg<1d-6) then !23jan2004 1d-10 ok?
Cstop2rx 2013.08.09 kino        stop "sigintAn1: absqg=0 is not supported yet. Improve here."
        call rx( "sigintAn1: absqg=0 is not supported yet. Improve here.")
c This part for absqg=0 has not been checked yet!
c       call bessl(0d0,lx,phi,psi)
c        do ir = 1,nr
c        do l  = 0,lx
c          a1int(ir,l) = .5d0* rofi(nr)**2     * rofi(ir)**l     * phi(l)
c     &                +(1d0/(2d0*l+3d0)-.5d0) * rofi(ir)**(l+2) * phi(l)
c       enddo
c        enddo
      else
        call  radkj(absqg**2, rofi(nr),lx,aknr,ajnr,dknr,djnr,0)
        a1int(1,:) = 0d0
        do ir = 2,nr
          call radkj(absqg**2, rofi(ir),lx,ak,aj,dk,dj,0)
          do l  = 0,lx
            a1int(ir,l) = ( (2*l+1)* aj(l)
     &     -((l+1)* ajnr(l)+ rofi(nr)*djnr(l) )*(rofi(ir)/rofi(nr))**l)
     &    /absqg**2
     &   *rofi(ir)
          enddo
        enddo
      endif
c      print *,' sigintAn1: end'
      end

c--------------------------------------------------
      subroutine sigintpp( absqg1, absqg2, lx, rmax,
     o              sig)
c sig(l)   =  \int_0^a r^2 {r_{<}}^l / (r_{>})^{l+1} *
c              j_l(absqg1 r)/absqg1**l
c              j_l(absqg2 r)/absqg2**l
c e1\ne0 e2\ne0
      implicit none
      integer(4) :: l,lx
      real(8)::  rmax,sig(0:lx), absqg1,absqg2, e1,e2,
     &   ak1(0:lx) ,aj1(0:lx), dk1(0:lx), dj1(0:lx),
     &   ak2(0:lx) ,aj2(0:lx), dk2(0:lx), dj2(0:lx),
     &  fkk(0:lx),fkj(0:lx),fjk(0:lx),fjj(0:lx)
c---
      e1 = absqg1**2
      e2 = absqg2**2
c
c      print *," sigintpp",e1,e2
c
      call wronkj( e1,e2, rmax,lx,   fkk,fkj,fjk,fjj )
      call  radkj( e1,    rmax,lx,   ak1,aj1,dk1,dj1,0)
      call  radkj( e2,    rmax,lx,   ak2,aj2,dk2,dj2,0)
c
      do l = 0,lx
        sig(l)= ( -l*(l+1)*rmax*aj1(l)*aj2(l)
     &            + rmax**3 * dj1(l)*dj2(l)
     &            + 0.5d0*rmax**2* (aj1(l)*dj2(l)+aj2(l)*dj1(l))
     &            - fjj(l)*(2*l+1)*(e1+e2)/2d0
     &           ) /(e1*e2)
      enddo
      end

```

4.17 gwsrc/mkqg.F File Reference

Functions/Subroutines

- subroutine mkqg2 (alat, plat, symops, ngrp, nnn, iq0pin, QpGcut_psi, QpGcut_Cou, ifiqg, ifiqgc, gammacellctrl, lnq0iadd)

4.17.1 Function/Subroutine Documentation

4.17.1.1 subroutine mkqg2 (real(8) *alat,* real(8), dimension(3,3) *plat,* real(8), dimension(3,3,ngrp) *symops,* integer *ngrp,* integer, dimension(3) *nnn,* integer *iq0pin,* real(8) *QpGcut_psi,* real(8) *QpGcut_Cou,* integer *ifiqg,* integer *ifiqgc,* integer *gammacellctrl,* logical *lnq0iadd*)

Definition at line 1 of file mkqg.F.

Here is the caller graph for this function:

4.18 mkqg.F

```
00001       subroutine mkqg2(alat,plat,symops,ngrp,nnn,iq0pin,
00002      &     qpgcut_psi, qpgcut_cou, ifiqg, ifiqgc,gammacellctrl,lnq0iadd)
00003 !! 'call getbzdata1' gives all follwing data
00004       use m_get_bzdata1,only:  getbzdata1,
00005      &  nqbz, nqibz, nqbzw,ntetf,nteti,nqbzm,
00006      &  qbz,wbz,qibz,wibz,
00007      &  qbzw, !qbasmc,
00008      &  idtetf, ib1bz, idteti,
00009      &  irk, nstar, nstbz,
00010      &  qbzm, qbzwm
00011       use m_keyvalue,only:getkeyvalue
00012 !! 'call getallq0p' give follwing data
00013       use m_q0p,only: getallq0p,
00014      &   q0i,wt,nq0i,nq0itrue, nq0iadd
00015 !! == Make required q and G in the expantion of GW. ==
00016 !!     |q+G| < QpGcut_psi for eigenfunction psi.
00017 !!     |q+G| < QpGcut_Cou for coulomb interaction
00018 !!
00019 !! OUTPUT
00020 !!     file handle= ifiqg,  which contains q and G points for eigenfunction psi. --> QGpsi
00021 !!     file handle= ifiqgc, which contains q and G points for Coulomb            --> QGcou
00022 !!
00023 !!     QGpsi(ifiqg), QGcou(ifiqgc), Q0P are written.
00024 !!     See the end of console output.
00025 !! ---------------------------------------------------
00026       implicit none
00027 c      integer,parameter:: nqibz_r=0
00028 c      real(8)::qibz_r(3,1) !dummy
00029
00030       integer ::nnn(3),ifiqg,ifiqgc,ngcxx,
00031      &     ngrp,i,j,iq,iq00,ngp,ngpmx,ngc,ngcmx,nqnum,iq0pin,
00032      &     nline,nlinemax,ifsyml,iqq,is,nk,ix,nqnumx,i1,ifkpt
00033       real(8)  :: plat(3,3),qlat(3,3),q(3),dummy,qp(3),
00034      &     qpgcut_psi, qpgcut_cou,qpgcut,alpv(3),q0smean,sumt,alp,
00035      &     volum,voltot,pi,q0(3),qlat0(3,3), alat,tripl,
00036      &     symops(3,3,ngrp),xx,qqx(3),alpm
00037       integer,allocatable:: ngvecp(:,:), ngvecc(:,:),
00038      &     ngpn(:),ngcn(:),ngvect(:,:,:),ngcx(:), nqq(:)
00039       real(8),allocatable ::
00040      &     qq(:,:),qq1(:,:),qq2(:,:),qqm(:,:)
00041       real(8) :: vol,ginv(3,3),aaa,dq(3) !,www
00042       integer :: mxkp,ifiqibz,iqibz,ifigwin,mtet(3),nm1,nm2,nm3
00043       logical ::tetrai,tetraf,tetra_hsfp0
00044       integer :: ifbz
00045 c      integer(4):: bzcase=1
00046 c     logical :: readgwinput
00047       integer:: nqnumm,ifiqmtet,verbose,q0pchoice,nn1,nn2,ifiqbz,iqbz !,auxfunq0p
00048       real(8)::aaij,bbij
00049       logical:: qbzreg
00050
00051       logical :: qreduce ,qreduce0
00052       real(8),allocatable:: qsave(:,:)
00053       integer:: imx,ifinin,il,imx0
00054       integer,allocatable :: ngvecprev(:,:,:),ngveccrev(:,:,:)
00055
00056       real(8):: ddq(3)
00057       logical :: offmesh=.false. ,offmeshg=.false.
00058       logical :: regmesh=.false. ,regmeshg=.false. ,  timereversal
00059
00060       logical :: caca,debug=.true. !,newaniso
00061       integer:: imxc,nnn3(3),imx0c,imx11(1,1)
00062       real(8):: deltaq,delta5,delta8,deltaq_scale!=1d0/3.0**.5d0
00063
00064       integer:: nqi,ifix,ig,iq0i,lm
00065       real(8),allocatable:: wti(:),qi(:,:)
```

```
      integer:: ifidml!,iclose,iopen !,ifiwqfac
      integer:: llxxx,lm1,lm2
      real(8),allocatable:: funa(:,:),wsumau(:),yll(:,:)
      real(8)::volinv,wtrue00,qg(3),alpqg2,qg2,tpiba
      character*99:: q0pf        !nov2012
      integer:: dummyia(1,1),iimx,irradd,nmax
      real(8):: epstol=1d-8,tolq=1d-8,qx(3),qxx(3)
      logical :: newoffsetg !july2014
      real(8),allocatable:: wt0(:)
      integer,allocatable::irr(:)
      real(8):: dq_(3),qlatbz(3,3)
      integer:: gammacellctrl,nnng(3),ifile_handle,ifi0,itet
      real(8)::imat33(3,3)
      logical:: lnq0iadd

c-------------------------------------------------
      print *,' mkqg2: '
      qreduce0 = qreduce()
      newoffsetg=.true. !newaniso()
      if(iq0pin == 101) then
         iq0pin=1
         newoffsetg=.false. !for old oldset Gamma case
      endif

!! I (apr2016takao)  think iq0pin==3 is used little now.
!! band case --- iq0pin == 3 ==>read syml file. E.g for Imag-part calcualtion along a symmetry line.
!!     nqq(is),qq1(1:3,is),qq2(1:3,is),is =1,nline
      if(iq0pin == 3) then
         qreduce0=.false.
         nlinemax = 50
         allocate(nqq(nlinemax),qq1(1:3,nlinemax),qq2(1:3,nlinemax))
         ifsyml = ifile_handle()
         open(ifsyml,file='SYML')
         nline = 0
         do
            nline = nline + 1
            read(ifsyml,*,err=601,end=601)
     &       nqq(nline),qq1(1:3,nline),qq2(1:3,nline)
         enddo
 601     continue
         close(ifsyml)
         nline = nline - 1
         write(6,"(/' Symmetry lines:'/' points',12x,'start',22x,'end')")
         do is=1,nline
            write(6,"(i6,2x,3f8.4,2x,3f8.4)")
     &      nqq(is),(qq1(i,is),i=1,3),(qq2(i,is),i=1,3)
         enddo
         nqnumx = sum(nqq(1:nline))
         allocate( qq(1:3,nqnumx),irr(nqnumx) )
         iqq = 0
         do is = 1,nline
            nk = nqq(is)
            do iq=1,nk
               xx = 0d0
               if(nk>1) xx=(iq-1d0)/(nk-1d0)
               qqx = xx*qq2(1:3,is)+(1d0-xx)*qq1(1:3,is)
               iqq = iqq + 1
               qq(1:3,iqq) = qqx
               write (6,"('  q=',3f7.3)") qq(1:3,iqq)
            enddo
         enddo
         nqnum = iqq
         write (6,"(' Total number of q-points:',i5/)") nqnum
         call minv33tp(plat,qlat) !it was dinv33(plat,1,qlat) by Ferdi
         goto 2001
      endif

!! we usually use negative delta (tetrahedron).
      call getkeyvalue("GWinput","delta",aaa)
      if(aaa<0d0) then
         print * ,'GWinput delta<0: tetrahedron method for x0'
         tetraf=.true.
      else
         print * ,'GWinput delta>0: not use tetrahedron method for x0'
         tetraf=.false.
      endif

!! plat,qlat,ginv
      voltot = abs(alat**3*tripl(plat,plat(1,2),plat(1,3)))
      call minv33tp(plat,qlat)
      call minv33(qlat,ginv)
      imat33=0d0
      imat33(1,1)=1d0
      imat33(2,2)=1d0
      imat33(3,3)=1d0
      if(sum(abs(matmul(transpose(qlat),plat)-imat33))>tolq) call rx('qlat*plat err')
      if(sum(abs(matmul(ginv,qlat)-imat33))>tolq)            call rx('ginv=qlat^-1 err')
      write(6,*)'=== plat ==='
      write(6,"(3d23.15)") plat
      write(6,*)'=== qlat ==='
      write(6,"(3d23.15)") qlat
c      write(6,*)'=== ginv==='
c      write(6,"(3f9.4)") ginv
!! We now use mtet=(1,1,1). If we like to recover this, examine code again.
      call getkeyvalue("GWinput","multitet",mtet,3,default=(/1,1,1/))

!! For gammacellctrl==2, we only consider tetrahedron method within the Gammacell.
!! The Gammacell is a part of BZ made from three vectors following qlatbz=(qlat(:,1)/n1q,...)
!! Then the Gamma point is in the middle of micro_qlat = (qlat(:,1)/n1q,qlat(:,2)/n2q,...)
!! To get qbz which is in the Gamma cell, we use shift in the getbzdata1 for gammacellctrl=2.
!! Tetrahedron method is applied for such qbz.
      if(gammacellctrl==2) then
        do i=1,3
          qlatbz(:,i) = qlat(:,i)/nnn(i) !qlat for Gamma cell
        enddo
        call getkeyvalue("GWinput","GammaDivn1n2n3",nnng,3)
        nnn = nnng          !division of Gamma cell
        dq_ = -matmul(qlatbz(1:3,1:3),(/.5d0,.5d0,.5d0/))
!This shift vector is to make the Gamma point centered in the Gamma cell.
        tetrai=.false.
        call minv33(qlatbz,ginv)
        write(6,*)'=== Gammacell qlatgz ==='
        write(6,"(3d23.15)") qlatbz
        write(6,*)'=== Gammacell ginv ==='
        write(6,"(3f9.4)") ginv
cccccccccccccccccccccc
c        qreduce0=.false.
cccccccccccccccccccccc
      else
        qlatbz(:,:) = qlat(:,:)
        tetrai = .true.         !used in heftet tetra_hsfp0()
        dq_ = 0d0
        if(.not.qbzreg()) dq_ = -matmul(qlat(1:3,1:3),(/.5d0/nnn(1),.5d0/nnn(2),.5d0/nnn(3)/))
                          !This dq_ is off-gamma mesh, used when qbzreg=F
      endif
ccccccccccccccccccccccccccc
c      dq_=0d0
ccccccccccccccccccccccccccc
      if(sum(abs(dq_))>tolq) write(6,'(" Shift vector (skip Gamma) by dq_=",3f9.4)')dq_

!! Get BZ data by 'call getbzdata1'
!! See following data after 'use getbzdata1' at the top of this routine.
!! In the case of gammacellctrl=2, we only calculate quantities in the Gamma cell.
!! Thus we have special meanings of nqbz. GWinput --> GammaDivn1n2n3 4 4 4
      call getbzdata1(qlatbz,nnn, !plat bzcase,
     & symops,ngrp,tetrai,tetraf,mtet,gammacellctrl) !all are inputs. output: See use.

!! Write BZDATA
      print *,' Writing BZDATA...'
      ifbz = ifile_handle()
      open (ifbz, file='BZDATA')
      write(ifbz,"(10i10)")  nqbz,nqibz, nqbzw, ntetf, nteti,ngrp !,nqibz_r
      write(ifbz,"(10i10)")  nnn(1:3) !n1q,n2q,n3q
      write(ifbz,"(3d24.16)") qlat,ginv!,qbasmc
      do iqibz = 1,nqibz
        write(ifbz,"(4d24.16,i9)") qibz(1:3,iqibz),wibz(iqibz),nstar(iqibz)
c         write(6,"('bbbbbbbbb ',4d24.16,i9)") qibz(1:3,iqibz),wibz(iqibz),nstar(iqibz)
        write(ifbz,"(100i8)") irk(iqibz,1:ngrp)
      enddo
c      write(ifbz,"(i10)") nqibz_r
c      do iqibz = 1,nqibz_r
c        write(ifbz,"(3d24.16)") qibz_r(1:3,iqibz)
c      enddo
      do iqbz = 1,nqbz
        write(ifbz,"(4d24.16,i10)") qbz(1:3,iqbz),wbz(iqbz),nstbz(iqbz)
      enddo
      if(ntetf>0) then
        write(ifbz,"(4i10)") (idtetf(0:3,itet),itet=1,ntetf)
        write(ifbz,"(i9,3d24.16)") (ib1bz(iqbz), qbzw(1:3,iqbz),iqbz=1,nqbzw)
      endif
      if(nteti>0) write(ifbz,"(5i10)") (idteti(0:4,itet),itet=1,nteti)
      write(ifbz,"(3d24.16,' !dq_')") dq_
      close(ifbz)
!! Write QIBZ
      write(6,*)' qibz are written in QIBZ file...'
      ifiqibz = ifile_handle()
      open (ifiqibz, file='QIBZ') !write q-points in IBZ.
      write(ifiqibz,"(i10)") nqibz
      do iqibz = 1,nqibz
         write(ifiqibz,"(3d24.16,3x,d24.16)") qibz(1:3,iqibz),wibz(iqibz)
      enddo
      close(ifiqibz)
!! Write QBZ
      ifiqbz = ifile_handle()
      open (ifiqbz, file='QBZ') !write q-points in IBZ.
      write(ifiqbz,"(i10)") nqbz
      do iqbz = 1,nqbz
         write(ifiqbz,"(3d24.16,3x,d24.16)") qbz(1:3,iqbz)
      enddo
      close(ifiqbz)
!!  Write KPNTin1BZ.mkqg.chk (files *.chk is only for check.).
      ifkpt = ifile_handle()
      open(ifkpt,file='KPTin1BZ.mkqg.chk')
      write(ifkpt,*)"  qbz --> shoten(qbz)"
      do      i1 = 1,nqbz
         call shorbz(qbz(1,i1),qp,qlat,plat)
         write (ifkpt,"(1x,i7,4f10.5,'    ',3f10.5)")
     &        i1,qbz(1,i1),qbz(2,i1),qbz(3,i1),wbz(i1),qp
      end do
      close (ifkpt)
      write(6,*) ' --- TOTAL num of q =',nqbz
      write(6,*)
      write(6,"( '  ngrp  = ',i3)")ngrp
      write(6,'("  qibz=",i6,3f12.5)')(i,qibz(1:3,i),i=1,min(10,nqibz))
      write(6,*)" ... QIBZ is written in QIBZ file ..."
!! alpha is for auxially function for offset Gamma method.
      call getkeyvalue("GWinput","alpha_OffG",alp,default=-1d60)
      alpv(:)=alp
      if(alp==-1d60) then
       call getkeyvalue("GWinput","alpha_OffG_vec",alpv,3,default=(/-1d50,0d0,0d0/))
       if(alpv(1)==-1d50) then
         call rx( ' mkqg: No alpha_offG nor alpha_offG_vec given in GWinput')
       endif
      endif
      print *
      print *,' alpv=',alpv
      print *
      alpm = minval(alpv)
      if(alpm<=0d0) call rx( 'alpha_offG or alpha_offG_vec <=0')

!! Determine G vectors for q points set by getgv2
      if(iq0pin==1) then   ! --- get q0x (offsetted q=0 point) ----------------------
!! I now think this QpGwut is large enough.
         qpgcut = sqrt(25d0/alpm) !a.u. !exp( -alp*QpGcut**2) !alp * QpGcut**2 = 22
         allocate( ngcx(nqbz) )
         ngcx=1
         do iq = 1, nqbz
            q   = qbz(1:3,iq)
            call getgv2(alat,plat,qlat,q, qpgcut, 1, ngcx(iq),  dummyia)
         enddo
         ngcxx = maxval(ngcx)
         allocate( ngvect(3,ngcxx,nqbz) )
         print *,' goto getgv2: ngcxx=',ngcxx
         do iq = 1, nqbz
            q  = qbz(1:3,iq)
            call getgv2( alat,plat,qlat, q, qpgcut, 2,
     &           ngcx(iq), ngvect(1:3,1:ngcx(iq),iq) )
         enddo
      endif

!! getallq0p all inputs
!! Q0P is offset Gamma or k point given in GWinput
!! see use m_q0p =>  q0i,wt,nq0i,nq0itrue are outputs
!! we now have q0i(:,nq0i+1,nq0i+nq0iadd).
!!    q0i(:,1:nq0i+n0qiadd) contains all q0x(:,i)= qlat(:,i)/nnn(i)/2d0*deltaq_scale() for i=1,3.
c      lnq0iadd=.true.
      call getallq0p(iq0pin,newoffsetg,alat,plat,qlat,nnn,alp,alpv, !apr2016
     i ngcxx,ngcx,nqbz,nqibz,nstbz,qbz,qibz,symops,ngrp,ngvect,lnq0iadd)
c      print *,'size q0i=',size(q0i),ubound(q0i),lbound(q0i)
      do i=nq0i+1,nq0i+nq0iadd
        write(6,"('  q0iadd=  ', i3, 3f10.5)") i,q0i(:,i)
      enddo

!! Four kinds of mesh points. Q0P means offset Gamma (slightly different from Gamma).
!! Which we need?
!! 1. regular
!! 2. offregular (not including Gamma)
!! 3. regular    + Q0P
!! 4. offregular + Q0P
      if(iq0pin==2) then        !this is just for dielectric case
         regmesh = qbzreg()
      else
         regmesh = .true.
      endif
      regmeshg = qbzreg()       !Gamma mesh based on regular mesh
      offmesh =  .not.qbzreg()  !we fix bzcase=1 now. apr2015.
      offmeshg = .not.qbzreg()  !Gamma mesh based on off-regular mesh
      print *,' regmesh offmeshg=', regmesh,regmeshg !regular,     regular+shifted
      print *,' offmesh offmeshg=', offmesh,offmeshg !offregmesh, offregular+shifted

!!  We check wether all q0i \in qbz or not. <--- Takao think this block is not necessary now.
```

```
      call minv33(qlat,ginv)
      nqnum = nqbz
      allocate( qq(1:3,nqnum),irr(nqnum) )
      qq(1:3,1:nqbz) = qbz(1:3,1:nqbz)
      do iq0i=1,nq0i+nq0iadd
         do iq=1,nqbz
            if(sum(abs(q0i(:,iq0i)-qq(:,iq)))<tolq) goto 2112
            call rangedq( matmul(ginv,q0i(:,iq0i)-qq(:,iq)), qx)
            if(sum(abs(qx))< tolq) goto 2112
         enddo
         goto 2111
 2112    continue
         qq(:,iq) = q0i(:,iq0i) !replaced with equivalent q0i.
      enddo
      print *,' --- We find all q0i in qbz. Skip qreduce.'
      goto 2001
 2111 continue


!! Accumulate all required q points
      deallocate(qq,irr)
      nqnum = nqbz  + nqbz*(nq0i+nq0iadd)
      nqnum = nqnum + 1         !add Gamma
      nqnum = nqnum + nq0i + nq0iadd      !add Gamma + q0i
      allocate( qq(1:3,nqnum),irr(nqnum) )
      ix = 0
      if(regmesh) then
         qq(1:3,1:nqbz) = qbz(1:3,1:nqbz)
         ix = ix+ nqbz
      endif
!!  - Off Regular mesh.
      if(offmesh) then
         do iq = 1, nqbz
            ix = ix+1
            qq(1:3,ix) = qbz(1:3,iq) - dq_
         enddo
      endif
c      nnn   =  ix               !n1q*n2q*n3q!      if(offmesh) nnn = 2*n1q*n2q*n3q
c      print *,' nnn=',nnn       !This is the number to calcualte Vxc
!!  - Shifted mesh
      if(regmeshg) then
         do iq00 = 1, nq0i+ nq0iadd
            do iq   = 1, nqbz
               ix = ix+1
               qq(1:3,ix) = qbz(1:3,iq) +  q0i(1:3,iq00)
            enddo
         enddo
      endif
      if(offmeshg) then
         do iq00 = 1, nq0i+ nq0iadd
            do iq   = 1, nqbz
               ix = ix+1
               qq(1:3,ix) = qbz(1:3,iq) - dq_ + q0i(1:3,iq00)
            enddo
         enddo
      endif
!!  - Add offset Gamma and Gamma point (these can be removed by qreduce and q0irre)
      do iq00 = 1, nq0i+ nq0iadd
         ix = ix+1
         qq(1:3,ix) = q0i(1:3,iq00)
      enddo
      ix=ix+1
      qq(1:3,ix)=0d0


!! (this mtet block is not used now) Get qqm; q point for eigenvalues.
!! Saved to Qmtet. Not so much used now...
!! We need check again if we like to use this branch again (2016apr)
      if(sum(abs(mtet))/=3) then
         nqnumm= nqbzm * (nq0i+ nq0iadd +1)
         allocate( qqm(1:3,nqnumm) )
         ix=0
         do iq00 = 1, 1 + nq0i+ nq0iadd
            do iq   = 1, nqbzm
               ix = ix+1
               if(iq00==1) then
                  qqm(1:3,ix) = qbzm(1:3,iq)
               else
                  qqm(1:3,ix) = q0i(1:3,iq00-1) + qbzm(1:3,iq)
               endif
            enddo
         enddo
         ifiqmtet=ifile_handle()
         open(ifiqmtet, file='Qmtet')
         write(ifiqmtet,"(i10)") nqnumm
         do iq=1,nqnumm
            write(ifiqmtet,"(3d24.16)") qqm(1:3,iq)
         enddo
         close(ifiqmtet)
         deallocate(qqm)
      endif

!! Remove equivalent q point by the translational symmetry
      if( qreduce0 ) then
         print *,'goto qqsave nq0i nq0iadd nqnum',nq0i,nq0iadd,nqnum
         call cputid(0)
         nmax= nq0i+nq0iadd+nqnum
         allocate(qsave(3,nmax)) !,qsavel(nmax))
         imx=0
         if(iq0pin /=1) then
            do iq=1,nq0i+ nq0iadd
               call qqsave(q0i(1:3,iq),nmax,ginv,qsave,imx)
            enddo
         endif
         do iq=1,nqnum
            call qqsave(qq(1:3,iq),nmax,ginv,qsave,imx)
         enddo
         nqnum = imx
         qq(:,1:imx)=qsave(:,1:imx)
         deallocate(qsave)
      endif
!! ------------------------------------------
 2001 continue
!! ------------------------------------------


!! Here we get all requied q points. We do reduce them by space group symmetry.
      if(allocated(wt0)) deallocate(wt0)
      allocate(wt0(nqnum+nq0i+ nq0iadd ),qi(3,nqnum+nq0i+ nq0iadd ),wti(nqnum+nq0i+ nq0iadd ))
      wt0=1d0
!! Set irreducible k-point flag. irr=1 for (irredusible point) flag, otherwise =0.
!! irr(iq)=1 for irreducile qq(:,iq), iq=1,nqnum
      call q0irre(qibz,nqibz,qq,wt0,nqnum,symops,ngrp, qi,nqi,wti,plat,.true.,0,irr)
!! nqnum is the finally obtained number of q points.
      allocate(ngpn(nqnum), ngcn(nqnum))
      if(debug) write(6,*) ' --- q vector in 1st BZ + Q0P shift. ngp ---'
      imx=0
      imxc=0
      do iq = 1, nqnum
         q = qq(1:3,iq)
         qxx=q
         if(iq0pin==1) then !use qxx on regular mesh points if q is on regular+Q0P(true).
            do iqbz=1,nqbz
            do i=1,nq0itrue+ nq0iadd  ! nq0itrue/=nq0i for anyq=F nov2015
               if(sum(abs(qbz(1:3,iqbz)-dq_+ q0i(:,i)-qxx))<tolq) then
                  qxx=qbz(1:3,iqbz)
                  exit
               endif
            enddo
            enddo
         endif
         ngpn(iq)=1
!! get nqpn. # of G vector for |q+G| < QpGcut_psi
         call getgv2(alat,plat,qlat, qxx, qpgcut_psi,1,ngpn(iq),imx11) !imx11 !nov2015
         imx0=imx11(1,1)
         if(imx0>imx) imx=imx0
         ngcn(iq)=1
!! get ngcn. # ofG vector for |q+G| < QpGcut_cou
         call getgv2(alat,plat,qlat, qxx, qpgcut_cou,1,ngcn(iq),imx11) !imx11 to avoid warning.
         imx0c=imx11(1,1)
         if(imx0c>imxc) imxc=imx0c
         if(verbose()>150)write(6,'(3f12.5,3x,2i4)') q ,ngpn(iq) !,ngcn(iq,iq00)
         if(verbose()>150)write(6,'(3f12.5,3x,2i4)') q ,ngcn(iq) !,ngcn(iq,iq00)
      enddo

!! Get G vectors and Write q+G vectors -----------
      ngpmx = maxval(ngpn)
      ngcmx = maxval(ngcn)
      write(ifiqg ) nqnum,ngpmx,qpgcut_psi,nqbz,nqi,imx,nqibz
      write(ifiqgc) nqnum,ngcmx,qpgcut_cou,nqbz,nqi,imxc
!! :nqi:   The number of irreducible points (including irr. of offset points). irr=1.
!! ::       We calcualte eigenfunction and Vxc for these points.
!! :nqnum: total number of q points.
!! :imx:   to allocate ngvecprev as follows.
      print *,' number of irrecucible points nqi=',nqi
      print *,' imx nqnum=',imx,nqnum
      write(6,*) ' --- Max number of G for psi =',ngpmx
      write(6,*) ' --- Max number of G for Cou =',ngcmx
      allocate( ngvecprev(-imx:imx,-imx:imx,-imx:imx) )       !inverse mapping table for ngvecp (psi)
      allocate( ngveccrev(-imxc:imxc,-imxc:imxc,-imxc:imxc) ) !inverse mapping table for ngvecc (cou)
      ngvecprev=9999
      ngveccrev=9999
      do iq = 1, nqnum
         q = qq(1:3,iq)
         qxx=q
         q0pf=''
         do iqbz=1,nqbz  !use qxx on regular mesh points if q is on regular+Q0P(true).
         do i=1,nq0itrue+ nq0iadd  !nq0itrue/=nq0i for anyq=F nov2015
            if(sum(abs(qbz(1:3,iqbz)-dq_+ q0i(:,i)-qxx))<tolq) then
               if(sum(abs(q0i(:,i)-qxx))<tolq) then
                 q0pf=' <--Q0P  '   ! offset Gamma points
               else
                 q0pf=' <--Q0P+R'   ! offset Gamma points-shifted nov2015
               endif
               if(iq0pin==1) then
                  qxx=qbz(1:3,iqbz)
               endif
               exit
            endif
         enddo
         enddo
         ngp = ngpn(iq)
         ngc = ngcn(iq)
         write(6,"(' iq=',i8,' q=',3f9.5,' ngp ngc= ',2i6,' irr.=',i2,a)") !irr=1 is irreducible k points.
c         write(6,"(' iq=',i8,' q=',3f17.13,' ngp ngc= ',2i6,' irr.=',i2,a)") !irr=1 is irreducible k points.
     &        iq, q, ngp, ngc, irr(iq),trim(q0pf)
         allocate( ngvecp(3,max(ngp,1)), ngvecc(3,max(ngc,1)) )
         call getgv2(alat,plat,qlat, qxx, qpgcut_psi, 2, ngp,  ngvecp) ! for eigenfunctions (psi)
         call getgv2(alat,plat,qlat, qxx, qpgcut_cou, 2, ngc,  ngvecc) ! for Coulomb        (cou)
         write (ifiqg) q, ngp, irr(iq)
         do ig = 1,ngp
            nnn3 = ngvecp(1:3, ig)
            ngvecprev( nnn3(1), nnn3(2),nnn3(3)) = ig
         enddo
         write (ifiqg)  ngvecp,ngvecprev !ngvecprev is added on mar2012takao
         do ig = 1,ngc
            nnn3 = ngvecc(1:3, ig)
            ngveccrev( nnn3(1), nnn3(2),nnn3(3)) = ig
         enddo
         write (ifiqgc) q, ngc
         write (ifiqgc) ngvecc,ngveccrev
         deallocate(ngvecp,ngvecc)
      enddo
      deallocate(ngpn,ngcn,ngvecprev,ngveccrev)
      if(iq0pin==1) deallocate(ngvect)
      if(debug) print *,'--- end of mkqg2 ---'
      end
```

4.19 gwsrc/readqg.F File Reference

Data Types

- module m_readq0p
- module m_readqg

  Return QGcou and QGpsi ===.

Functions/Subroutines

- subroutine readppovl0 (q, ngc, ppovl)

4.19.1 Function/Subroutine Documentation

4.19.1.1 subroutine readppovl0 (real(8), dimension(3), intent(in) *q*, integer, intent(in) *ngc*, complex(8), dimension(ngc,ngc), intent(out) *ppovl*)

Definition at line 34 of file readqg.F.

Here is the caller graph for this function:

4.20 readqg.F

```
      module m_readq0p
      real(8),allocatable,protected:: wqt(:), wgt0(:,:),q0i(:,:) !,nx(:,:),nblocha(:)
      integer,protected:: nq0i,nq0iadd
      integer,protected,allocatable:: ixyz(:)

      contains
      subroutine readq0p()
      implicit none
      integer:: neps,ifiq0p,ifile_handle,i,nq0ix,iq0pin
      logical:: debug=.false.
c      write(6,*) 'reading QOP'
      ifiq0p=ifile_handle()
      open (ifiq0p,file='Q0P')
      read (ifiq0p,*) nq0i,iq0pin,nq0iadd
      allocate( wqt(1:nq0i),q0i(1:3,1:nq0i+nq0iadd),ixyz(nq0i+nq0iadd) )
      do i=1,nq0i+nq0iadd
         read (ifiq0p, * ) wqt(i),q0i(1:3,i),ixyz(i)
c         write (*, * ) wqt(i),q0i(1:3,i),ixyz(i)
      enddo
      nq0ix = nq0i
      do i=1,nq0i
         if(wqt(i)==0d0 ) then
            nq0ix = i-1
            exit
         endif
      enddo
      neps=nq0i-nq0ix ! number of zero weight q0p which are used for ixc=2 or 3 mode.
      write(6,*) ' num of zero weight q0p=',neps
      write(6,"(i3,f14.6,2x, 3f14.6)" )(i, wqt(i),q0i(1:3,i),i=1,nq0i+nq0iadd)
      close(ifiq0p)
      end subroutine
      end module

      subroutine readppovl0(q,ngc,ppovl)
      implicit none
      integer, intent(in) :: ngc
      complex(8), intent(out) :: ppovl(ngc,ngc)
      real(8), intent(in) :: q(3)
      integer:: ngc_r,ippovl0,ifile_handle
      real(8):: qx(3),tolq=1d-8
      ippovl0=ifile_handle()
      open(ippovl0,file='PPOVL0',form='unformatted')
      do
        read(ippovl0) qx,ngc_r
        if(sum(abs(qx-q))<tolq) then
          if(ngc_r/=ngc) call rx( 'readin ppovl: ngc_r/=ngc')
          read(ippovl0) ppovl
          exit
        endif
      enddo
      close(ippovl0)
      end

!> Return QGcou and QGpsi ===
      module m_readqg
      implicit none
      real(8),allocatable,private,target:: qc(:,:),qp(:,:)
      logical,private:: init(2)=.true.
      real(8),private:: QpGcut_cou, QpGcut_psi
      integer(4),private,target::   nqnumc,nqnump,ngcmx,ngpmx
      integer(4),allocatable,private:: ngvecp(:,:,:),ngp(:),ngvecc(:,:,:),ngc(:)
      integer,pointer,private::nqtt
      real(8),pointer,private::qtt(:,:)
      real(8),private:: epsd=1d-7
      integer,private,pointer:: nkey(:),kk1(:),kk2(:),kk3(:),iqkkk(:,:,:)
      integer,target,private :: nkeyp(3),nkeyc(3)
      integer,target,allocatable,private:: keyp(:,:),kk1p(:),kk2p(:),kk3p(:),iqkkkp(:,:,:)
      integer,target,allocatable,private:: keyc(:,:),kk1c(:),kk2c(:),kk3c(:),iqkkkc(:,:,:)
      real(8),private:: ginv_(3,3)
      contains
c----------------------------------
      subroutine readngmx(key,ngmx)
c- get ngcmx or mgpmx
      implicit none
      integer(4):: ngmx,ifiqg=4052
      character*(*) key
      if    (key=='QGpsi') then
        open(ifiqg, file='QGpsi',form='unformatted')
        read(ifiqg) nqnump, ngpmx, qpgcut_psi
        ngmx=ngpmx
      elseif(key=='QGcou') then
        open(ifiqg, file='QGcou',form='unformatted')
        read(ifiqg) nqnumc, ngcmx, qpgcut_cou
        ngmx=ngcmx
      else
        call rx( "readngmx: key is not QGpsi QGcou")
      endif
      close(ifiqg)
      end subroutine

!> Get ngv and ngvec(3,ngv) for given qin(3)
!! key=='QGcou' or 'QGpsi'
      subroutine readqg(key,qin,ginv,  qu,ngv,ngvec)
      implicit none
      character*(*), intent(in) :: key
      real(8), intent(in) :: qin(3),ginv(3,3)
      real(8), intent(out) :: qu(3)
      integer(4), intent(out) :: ngv, ngvec(3,*)

      integer(4):: ifi, iq,verbose
      if    (key=='QGpsi') then
        ifi=1
        if(verbose()>=80) write (6,"(' readqg psi: qin=',3f8.3,i5)") qin
      elseif(key=='QGcou') then
        ifi=2
        if(verbose()>=80) write (6,"(' readqg cou: qin=',3f8.3,i5)") qin
      else
        call rx( "readqg: wrongkey")
      endif
      if(init(ifi)) then
        call init_readqg(ifi,ginv)
        init(ifi)=.false.
      endif
      if(verbose()>=40) write(6,*)'end of init_readqg'
      call iqindx2qg(qin,ifi, iq,qu)
      if(ifi==1) then
        ngv  = ngp(iq)
        ngvec(1:3,1:ngv) = ngvecp(1:3,1:ngv,iq)
        return
      elseif(ifi==2) then
        ngv  = ngc(iq)
        ngvec(1:3,1:ngv) = ngvecc(1:3,1:ngv,iq)
        return
      endif
      call rx( "readqg: can not find QGpsi or QPcou for given q")
      end subroutine readqg

!> Get ngv
!! key=='QGcou' or 'QGpsi'
      subroutine readqg0(key,qin,ginv,  qu,ngv)
      implicit none
      character*(*), intent(in) :: key
      integer(4), intent(out) :: ngv
      real(8), intent(in):: qin(3),ginv(3,3)
      real(8), intent(out):: qu(3)

      integer(4):: ifi, iq,verbose
      if    (key=='QGpsi') then
        ifi=1
        if(verbose()>=80) write (6,"('readqg0 psi: qin=',3f8.3,i5)") qin
      elseif(key=='QGcou') then
        ifi=2
        if(verbose()>=80) write (6,"('readqg0 cou: qin=',3f8.3,i5)") qin
      else
        call rx( "readqg: wrongkey")
      endif
      if(init(ifi)) then
        call init_readqg(ifi,ginv)
        init(ifi)=.false.
      endif
      call iqindx2qg(qin,ifi, iq,qu)
      if(ifi==1) then
        ngv  = ngp(iq)
        if(verbose()>=80) write(6,*)'ngp=',ngv
      elseif(ifi==2) then
        ngv  = ngc(iq)
        if(verbose()>=80) write(6,*)'ngc=',ngv
      endif
      return
      call rx( "readqg0: can not find QGpsi or QPcou for given q")
      end subroutine

!> initialization. readin QGpsi or QGcou.
      subroutine init_readqg(ifi,ginv)
      implicit none
      integer(4), intent(in) :: ifi
      real(8), intent(in) :: ginv(3,3)

      integer(4):: ifiqg,iq,verbose
      real(8)::qq(3)
      real(8),allocatable:: qxx(:,:)
      integer:: isig,i,ix,kkk,kkk3(3),ik1,ik2,ik3,ik
      integer,allocatable:: ieord(:),key(:,:)
      ginv_=ginv
      write(6,*)' init_readqg ifi=',ifi
      ifiqg=4052
      if(ifi==1) then
        open(ifiqg, file='QGpsi',form='unformatted')
        read(ifiqg) nqnump, ngpmx, qpgcut_psi
        if(verbose()>49)
     &     write (6,"('init_readqg ngnumc ngcmx QpGcut_psi=',2i5,f8.3)")
     &     nqnump, ngpmx, qpgcut_psi
        allocate(ngvecp(3,ngpmx,nqnump),qp(3,nqnump),ngp(nqnump))
        do iq=1, nqnump
          read (ifiqg) qp(1:3,iq), ngp(iq)
          read (ifiqg) ngvecp(1:3,1:ngp(iq),iq)
          if(verbose()>40)
     &     write (6,"('init_readqg psi qp ngp =',3f8.3,i5)") qp(1:3,iq),ngp(iq)
        enddo
      elseif(ifi==2) then
        open(ifiqg, file='QGcou',form='unformatted')
        read(ifiqg) nqnumc, ngcmx, qpgcut_cou
c         write (6,"('init_readqg ngnumc ngcmx QpGcut_cou=',2i5,f8.3)")
c     &     nqnumc, ngcmx, QpGcut_cou
        allocate(ngvecc(3,ngcmx,nqnumc),qc(3,nqnumc),ngc(nqnumc))
        do iq=1, nqnumc
          read(ifiqg) qc(1:3,iq), ngc(iq)
c          if(verbose()>40)  write (6,"('init_readqg cou  qc ngc =',3f8.3,i5)") qc(1:3,iq), ngc(iq)
           write (6,"('init_readqg cou  qc ngc =',3f8.3,i5)") qc(1:3,iq), ngc(iq)
          read (ifiqg) ngvecc(1:3,1:ngc(iq),iq)
        enddo
      endif
      close(ifiqg)

!! === mapping of qtt ===
!! nkey, kk1,kk2,kk3, iqkkk are to get iqindx.
!!  q --> call rangedq(matmul(ginv,q), qx) ---> n= (qx+0.5*epsd)/epsd
!!       --->  ik1,ik2,ik3= tabkk(kkk,iqk,nkey) ---> iqkkk(ik1,ik2,ik3)
      if(ifi==1) then
         nqtt => nqnump
         qtt  => qp
         nkey => nkeyp
      elseif(ifi==2) then
         nqtt => nqnumc
         qtt  => qc
         nkey => nkeyc
      endif
!! followings are the same as codes in readeigen.F
      allocate(ieord(nqtt))
      allocate(key(3,0:nqtt),qxx(3,nqtt))
      key(:,0)=0 !dummy
      key=-99999
c      print *,'ginv_=',ginv_
      do iq=1,nqtt
         call rangedq(matmul(ginv_,qtt(:,iq)), qxx(:,iq))
ccccccccccccc
c        ix=1
c        print *,' xxxx ix qxx=',ix,iq,qtt(ix,iq),matmul(ginv_,qtt(:,iq))
ccccccccccccc
      enddo
!! get key and nkey for each ix.
      do ix =1,3
ccccccccccccc
c        do i=1,nqtt
c        print *,' ix qxx=',ix,i,qtt(ix,i),qxx(ix,i)
c        enddo
ccccccccccccc
         call sortea(qxx(ix,:),ieord,nqtt,isig)
ccccccccccccc
c        do i=1,nqtt
c        print *,' ix qxx=',ix,i,qxx(ix,ieord(i))
c        enddo
ccccccccccccc
         ik=0
         do i=1,nqtt
            kkk=(qxx(ix,ieord(i))+0.5d0*epsd)/epsd  !kkk is digitized by 1/epsd
            if(i==1.or.key(ix,ik)<kkk) then
               ik=ik+1
               key(ix,ik) = kkk
            elseif (key(ix,ik)>kkk) then
               write(6,*)ix, ik,i, key(ix,ik), qxx(ix,ieord(i))
               call rx( 'iqindx: bug not sorted well')
            endif
         enddo
         nkey(ix)=ik
      enddo
      deallocate(ieord)
!!  key is reallocated. inverse mattping, iqkkk
```

```
      if(ifi==1) then
        allocate( kk1p(nkey(1)),kk2p(nkey(2)),kk3p(nkey(3)) )
        allocate( iqkkkp(nkey(1),nkey(2),nkey(3)) )
         iqkkk => iqkkkp
         kk1 =>kk1p
         kk2 =>kk2p
         kk3 =>kk3p
      elseif(ifi==2) then
        allocate( kk1c(nkey(1)),kk2c(nkey(2)),kk3c(nkey(3)) )
        allocate( iqkkkc(nkey(1),nkey(2),nkey(3)) )
         iqkkk => iqkkkc
         kk1 =>kk1c
         kk2 =>kk2c
         kk3 =>kk3c
      endif

      kk1(:) = key(1,1:nkey(1))
      kk2(:) = key(2,1:nkey(2))
      kk3(:) = key(3,1:nkey(3))
      deallocate(key)
c      write(6,*)' ifi init_qqq nqtt=',ifi,nqtt
c      write(6,*)'kkk3=',kkk3
c      write(6,*)'nkey=',nkey
c      write(6,*)'kk1=',kk1
      do i=1,nqtt
         kkk3= (qxx(:,i)+0.5*epsd)/epsd !kkk is digitized by 1/epsd
         call tabkk(kkk3(1), kk1,nkey(1), ik1)
         call tabkk(kkk3(2), kk2,nkey(2), ik2)
         call tabkk(kkk3(3), kk3,nkey(3), ik3)
         iqkkk(ik1,ik2,ik3)=i
cccccccccccccccc
c      write(6,*)' ik1,ik2,ik3 i=',ik1,ik2,ik3,i
cccccccccccccccc
      enddo
      deallocate(qxx)
      end subroutine init_readqg
!! ---
      subroutine tabkk(kkin, kktable,n, nout)
      integer:: nout,n, kkin, kktable(n),i,mm,i1,i2
      i1=1
      i2=n
      if(kkin==kktable(1)) then
         nout=1
         return
      elseif(kkin==kktable(n)) then
         nout=n
         return
      endif
      do i=1,n
         mm=(i1+i2)/2
         if(kkin==kktable(mm)) then
            nout=mm
            return
         elseif(kkin>kktable(mm)) then
            i1=mm
         else
            i2=mm
         endif
      enddo
      write(6,*) i1,i2,kkin
      write(6,*) kktable(i1),kktable(i2)
      call rx( 'takk: error')
      end subroutine

c$$$c--- release to save memory area.
c$$$      subroutine releaseqg_notusednow(key)
c$$$      implicit none
c$$$      character*(*) key
c$$$      integer(4):: ifi
c$$$      if    (key=='QGpsi') then
c$$$        ifi=1
c$$$        deallocate(qp,ngvecp)
c$$$      elseif(key=='QGcou') then
c$$$        ifi=2
c$$$        deallocate(qc,ngvecc)
c$$$      else
c$$$        stop "releaseqg: in readQGcou"
c$$$      endif
c$$$      init(ifi)=.false.
c$$$      end subroutine
!!------------------------------------------------------------

!> Find index as q=qq(:,iq) with modulo of premitive vector.
!! ginv is the inverse of plat (premitive translation vector).
!! Use kk1,kk2,kk3,nkey(1:3),iqkkk to get iqindx.
      subroutine iqindx2qg(q,ifi, iqindx,qu)
      implicit none
      integer, intent(in):: ifi
      integer, intent(out):: iqindx
      real(8), intent(in) :: q(3)
      real(8), intent(out) :: qu(3)

      integer:: i_out, iq,iqx ,kkk3(3),ik1,ik2,ik3
      real(8):: qx(3),qzz(3)
      logical::debug=.false.
      if(ifi==1) then
c         nqtt => nqnump
         qtt  => qp
         nkey => nkeyp
         iqkkk => iqkkkp
         kk1 =>kk1p
         kk2 =>kk2p
         kk3 =>kk3p
      elseif(ifi==2) then
c         nqtt => nqnumc
         qtt  => qc
         nkey => nkeyc
         iqkkk => iqkkkc
         kk1 =>kk1c
         kk2 =>kk2c
         kk3 =>kk3c
      endif
c      if(debug) write(*,"(' iqindx2_: q=',3f20.15)") q
c      write(*,"(' iqindx2_: q=',3f20.15)") q
      call rangedq(matmul(ginv_,q), qzz)
      kkk3 = (qzz+0.5*epsd)/epsd
c      write(6,*)'kkk3=',kkk3
c      write(6,*)'kk1,nkey1',kk1,nkey(1)
c      write(6,*)'kk2,nkey2',kk2,nkey(2)
c      write(6,*)'kk3,nkey3',kk3,nkey(3)
      call tabkk(kkk3(1), kk1,nkey(1), ik1)
      call tabkk(kkk3(2), kk2,nkey(2), ik2)
      call tabkk(kkk3(3), kk3,nkey(3), ik3)
c      write(6,*)' ik1ik2ik3=',ik1,ik2,ik3
      iqindx = iqkkk(ik1,ik2,ik3)
c      write(6,*)'iqindx=',iqindx
      qu = qtt(:,iqindx)
c      write(6,*)'iqindx=',iqindx
c      write(6,*)'qu=',qu
      end subroutine

!> mini-sort routine.
      subroutine sortea(ea,ieaord,n,isig)
      real(8), intent(in) :: ea(n)
      integer(4), intent(inout) :: ieaord(n)
      integer, intent(in) :: n
      integer, intent(out) :: isig
      integer :: ix,i
      isig = 1
      do i = 1,n
        ieaord(i) = i
      enddo
      do ix= 2,n
        do i=ix,2,-1
          if( ea(ieaord(i-1)) >ea(ieaord(i) ) ) then
            call iswap(ieaord(i-1),ieaord(i))
            isig= -isig
            cycle
          endif
          exit
        enddo
      enddo
      end subroutine
      subroutine iswap(i,j)
      implicit none
      integer,intent(inout) :: i, j
      integer:: iwork
      iwork = j
      j = i
      i = iwork
      end subroutine
      end module m_readqg
```

4.21 gwsrc/sxcf_fal2.F File Reference

Functions/Subroutines

- subroutine sxcf_fal3z (kount, ixc, deltaw, shtw, qip, itq, ntq, ef, ef2, esmr, esmr2,nsp, isp,qbas, ginv,qibz, qbz, wk, nstbz, wik,nstar, irkip,freq_r, freqx, wx,dwdummy, ecore,nlmto, nqibz, nqbz, nctot,

4.21.1 Function/Subroutine Documentation

4.21.1.1 subroutine sxcf_fal3z (intent(in) *kount*, integer, intent(in) *ixc*, real(8), intent(in) *deltaw*, real(8), intent(in) *shtw*, real(8), dimension(3,∗), intent(in) *qip*, intent(in) *itq*, integer, intent(in) *ntq*, real(8), intent(in) *ef*, real(8), intent(in) *ef2*, real(8), intent(in) *esmr*, real(8), intent(in) *esmr2*, integer, intent(in) *nsp*, integer, intent(in) *isp*, real(8), dimension(3∗3), intent(in) *qbas*, real(8), dimension(3∗3), intent(in) *ginv*, real(8), dimension(3,nqibz), intent(in) *qibz*, real(8), dimension(3,nqbz), intent(in) *qbz*, real(8), dimension(nqbz), intent(in) *wk*, integer(4), dimension(nqbz), intent(in) *nstbz*, real(8), dimension(nqibz), intent(in) *wik*, intent(in) *nstar*, intent(in) *irkip*, real(8), dimension(nw_i:nw) *freq_r*, real(8), dimension(niw) *freqx*, real(8), dimension(niw) *wx*, real(8) *dwdummy*, real(8), dimension(nctot) *ecore*, integer *nlmto*, integer *nqibz*, integer *nqbz*, integer *nctot*)

z1p(j,t,t') = S[i=1,nbloch] <psi(q,t') | psi(q-rk,t) B(rk,i)> v(k)(i,j) NOTE: zmel(igb, nctot+nbmax, ntp0) —> <phi phi |igb>

Definition at line 1 of file sxcf_fal2.F.

Here is the call graph for this function:

4.22 sxcf_fal2.F

```
00001       subroutine sxcf_fal3z(kount,ixc,deltaw,shtw,qip,itq, ntq,ef,ef2,esmr,esmr2,
00002      i nsp,isp,                 !tiat,miat,
00003      i qbas,ginv,
00004      i qibz,qbz,wk,nstbz,wik,
00005      i nstar,irkip,
00006      i freq_r,freqx,wx,
00007      i dwdummy,ecore,
00008      d nlmto,nqibz,nqbz,nctot,
00009 c     d nl,nnc,nclass,natom,mdimx,
00010      d nbloch,ngrp, nw_i,nw ,niw,niwx,nq, !nlnmx,
00011      & nblochpmx ,ngpmx,ngcmx,
00012      & wgt0,nq0i,q0i,symgg,alat, nband, ifvcfpout, !shtvg,
00013      & exchange,tote,screen,cohtest, ifexsp,
00014      i iwini,iwend,
00015      i nbmx,ebmx,
00016      i wklm,lxklm,
00017      i dwplot,
00018      o zsec,coh,exx)
00019       use m_readqg
00020       use m_readeigen,only: readeval
00021       use m_keyvalue,only: getkeyvalue
00022       use m_zmel,only: get_zmelt,
00023      o ppovlz, zmel,zmeltt
00024       implicit none
00025 !! TimeReversal off. when nw_i is not zero.
00026 !! Calcualte diagonal part only version of simga_ii(e_i)= <i|Re[S](e)|i>
00027 !! Similar with sxcf_fal2.sc.F
00028 Co zsec: S_ij= <i|Re[S](e)|i> where e=e_i and e_i \pm deltaw
00029 Co
00030 Cr  exchange=T : Calculate the exchange self-energy
00031 Cr          =F : Calculate correlated part of the self-energy
00032 Cr
00033 Cr
00034 Cr---- 2001 Sep. esec=omega(itp,iw). Genral iw mode for exchange =F
00035 Cr 2000 takao kotani. This sxcf is starting from sec.f F.Aryasetiawan.
00036 C-----------------------------------------------------------------
00037
00038
00039 c---- original document for sce.f (correlation case) by ferdi.Aryasetiawan.
00040 c 92.02.24
00041 c 93.10.18 from sec.f modified to take into account equivalent atoms
00042 c calculates the correlated part of the self-energy SE
00043 c SEc(q,t,t') = <psi(q,t) |SEc| psi(q,t'>
00044 c SEc(r,r';w) = (i/2pi) < [w'=-inf,inf] G(r,r';w+w') Wc(r,r';w') >
00045
```

```
c the zeroth order Green function
c G(r,r’;w)   = S[occ]   psi(kn,r) psi(kn,r’)^* /(w-e(kn)-i*delta)
c             + S[unocc] psi(kn,r) psi(kn,r’)^* /(w-e(kn)+i*delta)

c the screened coulomb potential
c Wc(r,r’;w)  = W(r,r’;w) - v(|r-r’|)
c             = < [r1,r2] v(|r-r1|) X(r1,r2;w) v(|r2-r’|) >
c W(r,r’;w)   = < [r’’] ei(r,r’’;w) v(|r’’-r’| >
c ei          = e^(-1), inverse dielectric matrix
c             = 1 + vX
c e           = 1 - vX0 in RPA

c expand Wc(r,r’;w) in optimal product basis B
c Wc(r,r’;w)  = S[k=FBZ] S[i,j=1,nbloch]
c               B(k,i,r) Wc(k,w)(i,j) B(k,j,r’)^*
c Wc(k,w)(i,j) are  the matrix elements of Wc in B

c SEc(q,t,t’) = S[k=FBZ] S[n=occ]   S[i,j=1,nbloch]
c        <psi(q,t) |psi(q-k,n) B(k,i)> <B(k,j) psi(q-k,n) |psi(q,t’)>
c        (i/2pi) <[w’=-inf,inf] Wc(k,w’)(i,j)/(w’+w-e(q-k,n)-i*delta)>
c
c             + S[k=FBZ] S[n=unocc] S[i,j=1,nbloch]
c        <psi(q,t) |psi(q-k,n) B(k,i)> <B(k,j) psi(q-k,n) |psi(q,t’)>
c        (i/2pi) <[w’=-inf,inf] Wc(k,w’)(i,j)/(w’+w-e(q-k,n)+i*delta)>

c the analytic structure of GWc for w .le. ef
c                               |
c                               |   o = pole of G
c                               ^   x = pole of Wc
c                               |
c                               |   ef-w
c                               |----<-----
c                               |          |
c                 o  o  o  o  o |o  o  o   ^
c               x  x  x  x  x  x|          |
c  -----------------------------|---->------------------------------
c                               |x  x  x  x  x  x  x  x
c                               |              o  o  o  o  o
c                               |       <----->
c                               ^        gap in insulator
c                               |
c                               |

c the analytic structure of GWc for w .gt. ef
c                               |
c                               |   o = pole of G
c                               |   x = pole of Wc
c                               |
c         gap in insulator      ^
c                <----->        |
c      o  o  o  o               |
c         x  x  x  x  x  x  x  x|
c  ------------------------>----|-----------------------------------
c                   |           |x  x  x  x  x  x  x  x
c                   ^   o  o  o  o  o  o  o
c                   |           |
c                    ------<----|
c                       w-ef    |
c                               ^
c                               |
c integration along the real axis from -inf to inf is equivalent to
c the integration along the path shown
c-----------------------------------------------------------
c integration along the imaginary axis: wint (s. also wint.f) (takao ->wintz)
c   (i/2pi) < [w’=-inf,inf] Wc(k,w’)(i,j)/(w’+w-e(q-k,n) >
c the i*delta becomes irrelevant
c-----------------------------------------------------------
c
c omit k and basis index for simplicity and denote e(q-k,n) = e
c wint = (i/2pi) < [w’=-inf,inf] Wc(w’)/(w+w’-e) >
c
c w’ ==> iw’, w’ is now real
c wint = - (1/pi) < [w’=0,inf] Wc(iw’) (w-e)/{(w-e)^2 + w’^2} >
c
c transform: x = 1/(1+w’)
c this leads to a denser mesh in w’ around 0 for equal mesh x
c which is desirable since Wc and the lorentzian are peaked around w’=0
c wint = - (1/pi) < [x=0,1] Wc(iw’) (w-e)x^2/{(w-e)^2 + w’^2} >
c
c the integrand is peaked around w’=0 or x=1 when w=e
c to handel the problem, add and substract the singular part as follows:
c wint = - (1/pi) < [x=0,1] { Wc(iw’) - Wc(0)exp(-a^2 w’^2) }
c                          * (w-e)/{(w-e)^2 +w’^2}x^2 >
c        - (1/2) Wc(0) sgn(w-e) exp(a^2 (w-e)^2) erfc(a|w-e|)
c
c the second term of the integral can be done analytically, which
c results in the last term
```

```
c a is some constant
c
c when w = e, (1/pi) (w-e)/{(w-e)^2 + w’^2} ==> delta(w’) and
c the integral becomes -Wc(0)/2
c this together with the contribution from the pole of G (s.u.)
c gives the so called static screened exchange -Wc(0)

c---------------------------------------------
c contribution from the poles of G: SEc(pole)
c---------------------------------------------
c
c for w .le. ef
c SEc(pole) = - S[k=FBZ] S[n=occ] S[i,j=1,nbloch]
c        <psi(q,t) |psi(q-k,n) B(k,i)> <B(k,j) psi(q-k,n) |psi(q,t’)>
c             Wc(k,e(q-k,n)-w)(i,j) theta(e(q-k,n)-w)
c
c for w .gt. ef
c SEc(pole) = + S[k=FBZ] S[n=unocc] S[i,j=1,nbloch]
c        <psi(q,t) |psi(q-k,n) B(k,i)> <B(k,j) psi(q-k,n) |psi(q,t’)>
c             Wc(k,w-e(q-k,n))(i,j) theta(w-e(q-k,n))
c
c theta(x)  = 1   if x > 0
c           = 1/2 if x = 0
c           = 0   if x < 0

c FBZ = 1st BZ
c NOTE: the routine only calculates the diagonal elements of the SE
c       i.e. SEc(q,t)

c q       = q-vector in SEc(q,t)
c itq     = states t at q
c ntq     = no. states t
c eq      = eigenvalues at q
c ef      = fermi level in Rydberg
c tr      = translational vectors in rot*R = R’ + T
c iatomp(R) = R’
c ifrw,ifcw,ifrwi,ifcwi
c   = direct access unit files for Re and Im coulomb matrix
c     along real and imaginary axis
c ifrb,ifcb,ifrhb,ifchb
c         = direct access unit files for Re and Im b,hb
c qbas    = base reciprocal lattice vectors
c ginv    = inverse of qbas s. indxrk.f
cxxxxx ippb,ipdb,idpb,iddb = pointers to work array w for
c  ppb     = <phi(RLn) phi(RL’n’) B(R,i)>
c  pdb     = <phi(RLn) phidot(RL’n’) B(R,i)>
c  dpb     = <phidot(RLn) phi(RL’n’) B(R,i)>
c  ddb     = <phidot(RLn) phidot(RL’n’) B(R,i)>
c freq    = frequencies along real axis
c freqx   = gaussian frequencies x between (0,1)
c freqw   = (1-freqx)/freqx
c wx      = weights at gaussian points x between (0,1)
c ua      = constant in exp(-ua^2 w’^2) s. wint.f
c expa    = exp(-ua^2 w’^2) s. wint.f
c dw      = frequency mesh along real axis
c deltaw  = energy mesh in SEc(qt,w) ---Not used now
c iclass  = given an atom, tells the class
c wk      = weight for each k-point in the FBZ
c indexk  = k-point index
c qbz     = k-points in the 1st BZ
c nstar   = no. stars for each k
c irk(k,R,nq) = gives index in the FBZ with k{IBZ, R=rotation
c mdim    = dimension of B(R,i) for each atom R
c work arrays:
c rbq,cbq     = real and imaginary part of b(q)
c rhbq,chbq   = real and imaginary part of hb(q)
c rbkq,cbkq   = real and imaginary part of b(q-k)
c rhbkq,chbkq = real and imaginary part of hb(q-k)
c   b is the eigenvector of the LMTO-Hamiltonian
c ekq     = eigenvalues at q-k
c rmel,cmel = real and imaginary part of
c             <psi(q,t’) | psi(q-k,t) B(k,R,i)>
c wr1 ... = work arrays
c dimensions:
c nqibz   = number of k-points in the irreducible BZ
c n1,n2,n3= divisions along base reciprocal lattice vectors
c natom   = number of atoms
c nctot   = no. allowed core states
c nbloch  = total number of Bloch basis functions
c nlnmx   = maximum number of l,n,m
c nlmto   = total number of LMTO basis functions
c ngrp    = no. group elements (rotation matrices)
c niw     = no. frequencies along the imaginary axis
c nw      = no. frequencies along the real axis
c niwx    = max(niw,nw)
c
c secq(t) = <psi(q,t) |SEc| psi(q,t)>
```

```
c-----------------------------------------------------------------------
      intent(in)::
     i kount,ixc,deltaw,shtw,qip,itq, ntq,ef,ef2,esmr,esmr2,
     i nsp,isp,                 !tiat,miat,
     i qbas,ginv,
     i qibz,qbz,wk,nstbz,wik,
     i nstar,irkip,
c     i iclass,mdim,nlnmv,nlnmc,
c     i icore,ncore,imdim,
c     i ppb,
     i freq_r,freqx,wx,
     i dwdummy,ecore,
     d nlmto,nqibz,nqbz,nctot,
c     d nl,nnc,nclass,natom,mdimx,
     d nbloch,ngrp, nw_i,nw ,niw,niwx,nq, !nlnmx,
     & nblochpmx ,ngpmx,ngcmx,
     & wgt0,nq0i,q0i,symgg,alat, nband, ifvcfpout, !shtvg,
     & exchange,tote,screen,cohtest, ifexsp,
     i iwini,iwend,
     i nbmx,ebmx,
     i wklm,lxklm
c     i   pomatr, qrr,nnr,nor,nnmx,nomx,nkpo,
c     i invg,!il,in,im,nn_, lx,nx_,nxx_,dwplot !ppbrd, !cgr,,nlnm

      integer :: ntq, nqbz,nqibz,ngrp,nq,nw,niw, !natom,
     & nband,  nlmto, nq0i,nctot,mbytes,iwksize,nlmtobnd,nstate,nstatex,
     & irot,  iqisp,ikpisp,isp,nsp, !nlnmx, !iq, idxk,
c     &  iwr1,iwr2,iwr3,iwr4,iwc1,iwc2,iwc3,iwc4
     & ip, it,itp,              !ifcphi,    ! ifrb,ifcb,ifrhb,ifchb,
c     i iiclass,                 !mdim(*),
     i ifrcw,ifrcwi,            !iindxk,
     & ifvcfpout,ndummy1,ndummy2,kx,kr,ngc,ngb,nbloch, !n1,n2,n3, k,
     & kp,nt0,nocc, nt0p,nt0m,irkp,i,nt0org,nmax,nt,ntp0,
     & nbmax,nblochpmx,ix,nx,iw,iwp,ixs,ixsmx, !nclass,nl,nnc,
     &  nwx,niwx,
     & itq(ntq), !,iatomp(natom),  !,miat(natom,ngrp),
     & nstar(nqibz),irkip(nqibz,ngrp,nq),kount(nqibz,nq)
c
      real(8) :: q(3),qbas(3*3),ginv(3*3), !tr(3,natom), !,tiat(3,natom,ngrp)
     & wk(nqbz),wik(nqibz),qibz(3,nqibz),qbz(3,nqbz),
     & freqx(niw),wx(niw),      !expa(niw),
     & eq(nband,nq),
     & ekq(nband), ekc(nctot+nband),
     & tpi,ef,ef2,esmr,esmr2,efp,efm,wtx,wfac,wfacx,we,esmrx, !ua,
     & dwdummy,wtt,wexx,www,exx,exxq ,wfacx2,weavx2,wex
c      complex(8) :: zsec(-1:1,ntq,nq)
c      real(8)    ::  shtw
c                      ! This shft is  to avoid some artificial resonance effects.
c                      ! shtw can be zero for esmr/=0 given by takao.
c
      integer(4):: ngpmx, ngcmx, !ngcni(nqibz), !ngpn(nqbz),
     & igc,                     !ngvecpB(3,ngpmx,nqbz),ngveccBr(3,ngcmx,nqibz),
     & nadd(3)
      real(8) :: wgt0(nq0i,ngrp),qk(3), !qfbz(3),
     & qdiff(3),add(3),symgg(3,3,ngrp),symope(3,3), !qbasinv(3,3), det,
     & qxx(3),q0i(1:3,1:nq0i),shtv(3),alat,ecore(nctot), !shtvg(3,ngrp),
c     &   ppb(1), !pdb(1),dpb(1),ddb(1), !*
     & coh(ntq,nq)              !, pos(3,natom)
      complex(8)::   alagr3zz,wintz !geigB  (ngpmx,nband,nqbz),

c
c      real(8),allocatable:: !rmel(:,:,:),cmel(:,:,:),
c     &                      rmelt(:,:,:),cmelt(:,:,:)
      complex(8),allocatable :: zz(:),zzmel(:,:,:),
     & zw(:,:), zwz(:,:,:), zwz0(:,:),zwzi(:,:),zwz00(:,:)
c for exchange --------------------
      logical :: exchange,screen,cohtest,tote
      real(8),allocatable::
     & w1p(:,:,:),w2p(:,:,:),w3p(:,:)
      complex(8),allocatable :: z1p(:,:,:),vcoul(:,:),vcoult(:,:)
      integer:: invrot,invr
c      integer:: invg(ngrp),il(*),in(*),im(*),nn_,lx(*),nx_(*),nxx_ !nlnm(*),
c      real(8)::  cgr(*),ppbrd(*)

c- debugwrite ---------------------
      logical :: debug=.false. ,onceww

cccccccccccccc
c tetra
c      integer(4) :: ntqx
c      integer(4) :: ibzx(nqbz)
c      real(8)    :: wtet  (nband,nqibz,1:ntqx), wtetef(nband,nqibz)
c                ! where the last index is 3*itq+iw-1,itq=1,ntq,iw=-1,1
c     logical    :: tetraex
ccccccczzcccccc

      complex(8) :: wintzav,wintzsg_npm,wintzsg
      integer(4) :: ibl,iii,ivsumxxx,ifexsp ,iopen
      integer(4),save::ifzwz=-999

      integer(4) :: iwini, iwend, ia
      real(8)    :: esec, omega(ntq, iwini:iwend)
      complex(8) :: zsec(iwini:iwend,ntq,nq)
c      complex(8),allocatable:: expikt(:)
      complex(8):: img=(0d0,1d0)
ctakao
c      complex(8):: cphiq(nlmto,nband), cphikq(nlmto,nband)

      integer(4) :: nt_max, igb1,igb2,iigb,  nw_i !nw_i is at feb2006 TimeReversal off case
      complex(8),allocatable:: zmel3(:) !zmel1(:),
      complex(8), allocatable :: zw_(:,:) !,zzmel(:,:)
      complex(8), allocatable :: zwz2(:,:),zw2(:,:),zmel2(:,:) !0 variant
      complex(8) ::  zz2 ,zwz3(3) ,zwz3x
      real(8) :: dd,omg_c,dw2,omg
      real(8) :: freq_r(nw_i:nw)
      complex(8), allocatable :: zw3(:,:,:)


      real(8)::weavx,wfaccut=1d-10,qqqq

      logical :: gausssmear=.true.,gass
      real(8) :: ebmx,ddw
      integer(4):: nbmx,nbmxe,nstatetot

c      integer(4):: n_index_qbz
c      integer(4):: index_qbz(n_index_qbz,n_index_qbz,n_index_qbz)

c      integer(4)::icore(*),ncore(*),imdim(*) !,iclass(*),nlnmv(*),nlnmc(*),

      integer(4)::verbose,nstbz(nqbz),bzcase=1,iqini,iqend
      real(8):: wgtq0p

      integer(4):: nrec,kxx
      real(8)::quu(3),qibz_k(3),qbz_kr(3)
      logical ::  onlyimagaxis

      logical ::zwz3mode


      real(8):: ua_,expa_(niw),ua2,freqw,freqw1,ratio,ua2_(niw)
c$$$      logical :: ua_auto !fixed to be .false.
      integer(4):: icc=0
      real(8),allocatable:: uaa(:,:)

c      logical ::testimx=.false.
ccccc zvz test cccccccccccccccccccccccc
      integer(4):: ngbx
c      complex(8):: vcoul(ngbx,ngbx)
      complex(8),allocatable:: vzz(:,:,:),aaa(:), zwzs(:)
      complex(8):: zvz,zvz1
      integer(4):: ib1,ib2,ifix
ccccccccccccccccccccccccccccccccccc
      logical ::iww2=.true., oncew


C...
c      logical::smbasis
      integer(4):: iclose,isx,iqx !nn,no,ifpomat,
c      complex(8),allocatable:: pomat(:,:)
      real(8):: q_r(3)
c      integer(4):: nnmx,nomx,nkpo, nnr(nkpo),nor(nkpo)
c      complex(8):: pomatr(nnmx,nomx,nkpo)
c      real(8):: qrr(3,nkpo)

      real(8):: elxx,ehxx,ekxx,efxx
      integer(4):: ixsmin,iwm,iir,nwxi, itini,itend, npm
      real(8)   :: fffr(3),ppp
      complex(8):: zwzz(3)

      real(8),allocatable:: ebb(:)
      integer(4):: ii,iq
      logical ::evaltest        !, imgonly

      integer:: lxklm,ivc,ifvcoud,idummy,iy,ngb0
      real(8):: wklm((lxklm+1)**2),pi,fpi,vc,qvv(3),aaaa
      complex(8)::zmelt1,zmelt0
      real(8)::voltot
c      logical :: newaniso !fixed to be T

      complex(8),allocatable:: ppovl(:,:),zcousq(:,:) !,ppovlz(:,:)
      real(8),allocatable::vcoud(:),vcousq(:)
      integer:: mrecl,nprecx,ifwd
      character(5):: charnum5

      integer:: ixc
      real(8):: qip(3,*),deltaw,shtw,eqx(nband),dwplot,tolq=1d-8
      complex(8),allocatable:: zmelt(:,:)
      integer:: ntqxx,nrot
c-------------------------------------------------------------------
      write(6,*)'sxcf_fal3z'
c      timemix=.false.
      pi  = 4d0*datan(1d0)
      fpi = 4d0*pi
      debug=.false.
      if(verbose()>=90) debug=.true.
!!
      if(.not.exchange) then
        ifwd = iopen('WV.d',1,-1,0)
        read (ifwd,*) nprecx,mrecl
        ifwd = iclose('WV.d')
c$$$!! --- gauss_img : interpolation gaussion for W(i \omega).
c$$$      call getkeyvalue("GWinput","gauss_img",ua_,default=1d0)
c$$$      if(ua_<=0d0) then
c$$$        ua_auto =.true.
c$$$        write(6,"(' ua_auto=T')")
c$$$      else
c$$$        ua_auto =.false.
c$$$        do ix = 1,niw
c$$$          freqw     = (1d0 - freqx(ix))/ freqx(ix)
c$$$          expa_(ix) = exp(-(ua_*freqw)**2)
c$$$        enddo
c$$$      endif
        call getkeyvalue("GWinput","gauss_img",ua_,default=1d0)
        do ix = 1,niw           !! Energy mesh; along im axis.
          freqw     = (1d0 - freqx(ix))/ freqx(ix)
          expa_(ix) = exp(-(ua_*freqw)**2)
        enddo
        npm = 1                 ! npm=1    Timeveversal case
        if(nw_i/=0) npm = 2     ! npm=2 No TimeReversal case. Need negative energy part of W(omega)
      endif

      tpi         = 8d0*datan(1.d0)
      if(nctot/=0) ekc(1:nctot)= ecore(1:nctot) ! core
      nlmtobnd    = nlmto*nband
      nstatetot      = nctot + nband
c      call dinv33(qbas,0,qbasinv,det)
c      allocate(expikt(natom))


!! == ip loop to spedify external q ==
      do 1001 ip = 1,nq
        if(sum(irkip(:,:,ip))==0) cycle
        q = qip(1:3,ip)
        write (*,*) ip,'  out of ',nq,'  k-points ' ! call cputid  (0)
        if(ixc==2) then
          call readeval(q,isp,eqx)
          do iw = iwini,iwend
            do i  = 1,ntq
              omega(i,iw) = eqx(itq(i)) + 2d0*(dble(iw)-shtw)*deltaw
            enddo
          enddo
        endif
!!
        if(ixc==4) then
c        dwplot=0.01
        do iw = iwini,iwend
          omega(1:ntq,iw) =  dwplot* iw + ef
        enddo
        endif

        call readeval(q, isp, eq(1,ip))
!! we only consider bzcase()==1
        if(abs(sum(qibz(:,1)**2))/=0d0) call rx( ' sxcf assumes 1st qibz/=0 ')
        if(abs(sum( qbz(:,1)**2))/=0d0) call rx( ' sxcf assumes 1st qbz /=0 ')
        If (tote) exxq = 0.d0

!! == Big loop for kx ==
!! kx is for irreducible k points, kr=irk(kx,irot) runs all k points in the full BZ.
        iqini=1
        iqend=nqibz             !no sum for offset-Gamma points.
        do 1100 kx = iqini,iqend
          if(sum(irkip(kx,:,ip))==0) cycle
          write(6,*) ' ### do 1100 start kx=',kx,' from ',iqini,' through', iqend
c          if( kx <= nqibz ) then
           qibz_k= qibz(:,kx)
c          else
c            qibz_k= 0d0
c         endif
          if(verbose()>=40)  write(6,*) ' sxcf_fal3z: loop 1100 kx=',kx
          call readqg0('QGcou',qibz_k,ginv,  quu,ngc)
          ngb = nbloch + ngc    !oct2005
          if(debug) write(6,*) ' sxcf: ngb=',ngb,nbloch

!! ===Readin diagonalized Coulomb interaction===
!!  Vcoud file is sequential file Vcoulomb matrix for qibz_k.
!!  A possible choice for paralellization is "Vcoud.ID" files where ID=kx
!!  Vould file is written in hvccfp0.m.F.
!! For correlation, W-v is read instead of Vcoud file (ifrcw,ifrcwi for WVR and WVI)
!! These can be also separeted into WVR.ID and WVI.ID files.
!! NOTE: vcoud and zcousq are in module m_zmelt.
c          if(kx<=nqibz) qxx=qibz_k
c          if(kx>nqibz ) qxx=q0i(:,kx-nqibz)
          qxx=qibz_k
          ifvcoud = iopen('Vcoud.'//charnum5(kx),0,0,0)
          do
            read(ifvcoud) ngb0
            read(ifvcoud) qvv
            if(allocated(vcoud)) deallocate(vcoud)
            allocate( zcousq(ngb0,ngb0),vcoud(ngb0) )
            read(ifvcoud) vcoud
            read(ifvcoud) zcousq
            if(sum(abs(qvv-qxx))<tolq) goto 1133
          enddo
          if(sum(abs(qvv-qxx))>tolq) then
            write(6,*)'qvv =',qvv
            write(6,*)'qxx=',qxx,kx
            call rx( 'sxcf_fal2: qvv/=qibz(:,kx) hvcc is not consistent')
          endif
 1133     continue
          if( ngb0/=ngb ) then  !sanity check
            write(6,*)' qxx ngb0 ngb=',qxx,ngb0,ngb
            call rx( 'hsfp0.m.f:ngb0/=ngb')
          endif
!! used in get_zmel
!! <I|v|J>= \sum_mu ppovl*zcousq(:,mu) v^mu (Zcousq^*(:,mu) ppovl)
!! zmel contains O^-1=<I|J>^-1 factor. zmel(phi phi J)= <phi phi|I> O^-1_IJ
!! ppovlz= O Zcousq
!! (V_IJ - vcoud_mu O_IJ) Zcousq(J, mu)=0, where Z is normalized with O_IJ.
          if(allocated(ppovl)) deallocate(ppovl,ppovlz)
          allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb))
          call readppovl0(qibz_k,ngc,ppovl)
          ppovlz(1:nbloch,:) = zcousq(1:nbloch,:)
          ppovlz(nbloch+1:nbloch+ngc,:) = matmul(ppovl,zcousq(nbloch+1:nbloch+ngc,:))
          deallocate(zcousq)
!! === open WVR,WVI for correlation mode ===
          if(.not.exchange) then
            ifrcw  = iopen('WVR.'//charnum5(kx),0,-1,mrecl)
            ifrcwi = iopen('WVI.'//charnum5(kx),0,-1,mrecl)
          endif
          nrot=0
          do irot = 1,ngrp
c            if( kx <= nqibz) then
             kr = irkip(kx,irot,ip) ! index for rotated kr in the FBZ
             if(kr==0) cycle   ! next irot
             qbz_kr= qbz(:,kr)
c            else
c              kr=-99999         !for sanity check
c              qbz_kr= 0d0
c              if( wgt0(kx-nqibz,irot)==0d0 ) cycle ! next irot
c            endif
            nrot=nrot+1
          enddo

!! == loop over rotations ==
!! We may extend
          do 1000 irot = 1,ngrp
c            if( kx <= nqibz) then
             kr = irkip(kx,irot,ip) ! index for rotated k in the FBZ
             if(kr==0) cycle
             qbz_kr= qbz(:,kr)
c            else
c              kr=-99999         !for sanity check
c              qbz_kr= 0d0
c              if( wgt0(kx-nqibz,irot)==0d0 ) cycle
c            endif
            write(*,"('ip,kx irot=',3i5, ' out of',2i4)") ip,kx,irot, iqend,ngrp

c qk = q - rk, rk is inside 1st BZ, not restricted to the irreducible BZ
            qk =  q - qbz_kr    ! qbz(:,kr)
            call readeval(qk, isp, ekq)
            ekc(nctot+1:nctot+nband) = ekq(1:nband)
            nt0 = nocc(ekc,ef,.true.,nstatetot)
            ddw= .5d0
c        if(GaussSmear) ddw= 10d0
            ddw=10d0
            efp= ef+ddw*esmr
            efm= ef-ddw*esmr
            nt0p = nocc(ekc,efp,.true.,nstatetot)
            nt0m = nocc(ekc,efm,.true.,nstatetot)
!! nbmx1 ebmx1: to set how many bands of <i|sigma|j>  do you calculate.
!! nbmx2 ebmx2: to restrict num of bands of G to calculate G \times W
            if(exchange) then
              nbmax = nt0p-nctot
              if(debug) write(6,*)' sxcf: nbmax nctot nt0p =',nbmax,nctot,nt0p
            else
              nbmax = nband
              nbmxe = nocc(ekc,ebmx,.true.,nstatetot)-nctot
              nbmax  = min(nband,nbmx,nbmxe)
              if(onceww(3)) write(6,*)' nbmax=',nbmax
            endif
            nstate = nctot + nbmax ! = nstate for the case of correlation

!! all are identical.
            ntp0 = ntq
            ntqxx= ntp0

!! Get matrix element zmelt= rmelt + img*cmelt, defined in m_zmel.F---
            if(debug) write(6,*)'zzBBB ppovlz =',sum(abs(ppovlz(:,:))),kx,irot
            if(allocated(zmel)) deallocate(zmel)
            if(allocated(zmeltt)) deallocate(zmeltt)
! this return zmeltt (for exchange), or zmel (for correlation)
            call get_zmelt(exchange,q,kx,qibz_k,irot,qbz_kr,kr,isp,
     &       ngc,ngb,nbmax,ntqxx,nctot,ncc=0)
            if(kx<= nqibz) then
              wtt = wk(kr)      !         wtx = 1d0
            else
              wtt = wk(1)*wgt0(kx-nqibz,irot) !       wtx = wgt0(kx-nqibz,irot)
              if(abs(wk(1)-1d0/dble(nqbz))>1d-10)call rx( 'sxcf:wk(1)inconsistent')
            endif
            if(debug) write(6,*) 'ssssssss',size(zmel),ntqxx*nstate*ngb
            if(debug) write(6,"(' kx wtt=',i4,f12.8)") kx,wtt
            if(debug) write(6,*)' 000 sumzmel=',ngb, nstate, ntp0,sum(abs(real(zmel))),sum(abs(imag(zmel)))

!!------------------------------------------------------------
!! === exchange section ===
!!------------------------------------------------------------
c
c S[i,j=1,nbloch] <psi(q,t) |psi(q-rk,n) B(rk,i)>
c                        v(k)(i,j) <B(rk,j) psi(q-rk,n) |psi(q,t')>
c
c> z1p(j,t,t') = S[i=1,nbloch] <psi(q,t') | psi(q-rk,t) B(rk,i)> v(k)(i,j)
!! NOTE: zmel(igb, nctot+nbmax, ntp0) ---> <phi phi  |igb>

c --- screened exchange case ----
c          if(screen) then
c            ix = 1
c            nrec=(kx-iqini)*nw+ix
c            if(bzcase()==2) nrec= (kx-1)*nw+ix
c            read(ifrcw,rec=nrec) zw  ! Readin W(0) - v
c            vcoul = vcoul + zw(1:ngb,1:ngb) !c  screen test
c          endif

c          allocate( zmel(ngb, nctot+nbmax, ntp0), w3p( nctot+nbmax,ntp0))
c          zmel  = dcmplx (rmelt,cmelt)
            if(exchange) then
              allocate( w3p( nctot+nbmax,ntp0))
              do 992 itp = 1,ntp0
                do 993 it  = 1,nctot+nbmax
                  w3p(it,itp) = 0d0
                  do 994 ivc=1,ngb
                    if(ivc==1.and.kx==1) then
                      vc= wklm(1)* fpi*sqrt(fpi) /wk(kx)
c                      write(6,*)'wklm(1) vc=',wklm(1),vc
                    else
                      vc= vcoud(ivc)
                    endif
                    w3p(it,itp) = w3p(it,itp)+ vc * abs(zmeltt(it,itp,ivc))**2
 994              continue
 993            continue
 992          continue
              if(debug) then
                do  it  = 1,nctot+nbmax
                  do  itp = 1,ntp0
                    write(6,"(' w3p =',2i4,2d14.6)") it,itp,w3p(it,itp)
                  enddo
                enddo
              endif

!! Write the Spectrum function for exchange May. 2001.
!!!!!! Probably, Need to fix this....
              if(ifexsp/=0) then
                do it  = 1, nctot+nbmax
                  do itp = 1,ntp0
                    write(ifexsp,"(3i4, 3f12.4, ' ',d23.15,'  ',d23.15)")
     &               ip,itp,it, qbz_kr, ekc(it), -wtt*w3p(it,itp)
                  enddo
                enddo
              endif

!! --- Correct weigts wfac for valence by esmr
              do it = nctot+1, nctot+nbmax
                wfac = wfacx(-1d99, ef, ekc(it), esmr) !gaussian
                w3p(it,1:ntp0) = wfac * w3p(it,1:ntp0)
              enddo

              if (.not.tote) then !total energy mode tote
                do itp = 1,ntp0 !S[j=1,nbloch]  z1p(j,t,t') <B(rk,j) psi(q-rk,n) |psi(q,t')>
                  zsec(iwini,itp,ip) = zsec(iwini,itp,ip)
     &              - wtt * sum( w3p(:,itp) )
                enddo
              else
                do itp = 1,ntp0
                  wfac = wfacx(-1d99, ef2, eq(itq(itp),ip), esmr2) !gaussian
                  w3p(1:nctot+nbmax,itp) = wfac * w3p(1:nctot+nbmax,itp)
                  exxq = exxq - wtt * sum( w3p(:,itp) )
                enddo
              endif
              deallocate( w3p)  !,rmelt,cmelt)
              cycle
            endif
c-- End of exchange section --------------



c-----------------------------------------------------------------------
c--- correlation section -----------------------------------------------
c-----------------------------------------------------------------------
c$$$c--- The matrix elements zmel.
c$$$c        allocate( zmel (ngb, nstate, ntp0) )
c$$$c        zmel = dcmplx (rmelt,-cmelt)
c$$$c        if(newaniso) then
c$$$c#ifdef USE_GEMM_FOR_SUM
c$$$          if(verbose()>39)write(*,*)'info: USE GEMM FOR SUM (zmel=zmel*ppovlz), in sxcf_fal2.F'
c$$$          allocate( zmelt (ngb, nstate) )
c$$$          do itp=1,ntp0
c$$$          zmelt = dcmplx(rmelt(:,:,itp),-cmelt(:,:,itp))
c$$$          call zgemm('C','N',ngb,nstate,ngb,(1d0,0d0),
c$$$     .      ppovlz,ngb,zmelt,ngb,(0d0,0d0),zmel(1,1,itp),ngb)
c$$$          enddo
c$$$          deallocate(zmelt)
c$$$#else
c$$$          do itp=1,ntp0
c$$$            do it=1,nstate
c$$$              zmel(:,it,itp) =  matmul(zmel(:,it,itp),dconjg(ppovlz(:,:)))
c$$$            enddo
c$$$          enddo
c$$$#endif
c$$$c        endif
c        deallocate(rmelt,cmelt)
c        if(debug) write(6,*)' end of zmel'

c================================================================
c The correlated part of the self-energy:
c S[n=all] S[i,j=1,nbloch]
c <psi(q,t) |psi(q-rk,n) B(rk,i)>
c  < [w'=0,inf] (1/pi) (w-e)/{(w-e)^2 + w'^2} Wc(k,iw')(i,j) >
c                                <B(rk,j) psi(q-rk,n) |psi(q,t)>
c e = e(q-rk,n), w' is real, Wc = W-v
c================================================================
            allocate( zw(nblochpmx,nblochpmx) )
c====================================================================
c contribution to SEc(qt,w) from integration along the imaginary axis
c====================================================================
c-------------------------------------------------
c loop over w' = (1-x)/x, frequencies in Wc(k,w')
c {x} are gaussian points between (0,1)
c-------------------------------------------------
            allocate( zwz0(nstate,ntp0) )
            ix = 1  - nw_i      !at omega=0
c        nrec=(kx-iqini)*(nw-nw_i+1) +ix ! 2---> iqini
c        if(bzcase()==2) nrec= (kx-1)*(nw-nw_i+1) +ix
            nrec=ix
            if(debug) write(6,*)' wvr nrec kx nw nw_i ix=',nrec,kx,nw,nw_i,ix
            read(ifrcw,rec=nrec) zw ! direct access read Wc(0) = W(0) - v
            zwz0=0d0
!! this loop looks complicated but just in order to get zwz0=zmel*zwz0*zmel
!! Is this really efficient???
CCC!$OMP parallel do private(itp,it,igb2,zz2)
            do itp=1,ntp0
              do it=1,nstate
                do igb2=2,ngb
```

```
                  zz2 = sum( dconjg(zmel(1:igb2-1,it,itp))*zw(1:igb2-1,igb2) )
                  zwz0(it,itp) = zwz0(it,itp)+zz2*zmel(igb2,it,itp)*2d0+
     &             dconjg(zmel(igb2,it,itp))*zw(igb2,igb2)*zmel(igb2,it,itp)
                enddo           !igb2
                zwz0(it,itp) = zwz0(it,itp)+
     &           dconjg(zmel(1,it,itp))*zw(1,1)*zmel(1,it,itp)
              enddo             !it
            enddo               !itp
            zwz0 = dreal(zwz0)
c COH term test ----- The sum of the all states for zwz00 gives the delta function.
            if(cohtest) then
              do itp = 1,ntq
                coh(itp,ip)  = coh(itp,ip)
     &           + .5d0*wtt*sum(dreal(zwz0(1:nstate,itp)))
              enddo
              deallocate(zw,zwz0,zmel)
              cycle
            endif
c
            nx  = niw
            if(niw <1) call rx( " sxcf:niw <1")
            if(allocated(zwz)) deallocate(zwz)
            if(allocated(zwzi)) deallocate(zwzi)
            allocate( zwz(niw*npm, nstate,ntp0),  zwzi(nstate,ntp0) )
            if(screen) allocate( zwz00(nstate,ntp0) )
            if(verbose()>50) write(*,'("6 before matzwz in ix cycle ",$)')
            if(verbose()>50) call cputid(0)

            zwz=0d0
            do ix = 1,nx        !*npm            ! imaginary frequency w'-loop
              nrec= ix
              if(debug) write(6,*)' wvi nrec=',nrec
              read(ifrcwi,rec=nrec) zw ! Readin W-v on imag axis
              if(npm==1) then   !then zwz is real so, we can use mode c2.
                do itp= 1,ntp0
                  do it = 1,nstate
                    ppp=0d0
                    do igb2 = 2,ngb
                      zz2 = sum( dconjg(zmel(1:igb2-1,it,itp))*zw(1:igb2-1,igb2) )
! only take real part
                      ppp = ppp + dreal(zz2*zmel(igb2,it,itp)) * 2d0
     &                 + dconjg(zmel(igb2,it,itp))*zw(igb2,igb2)*zmel(igb2,it,itp)
                    enddo       !igb2
                    zwz(ix,it,itp) = ppp +
     &               dconjg(zmel(1,it,itp))*zw(1,1)*zmel(1,it,itp)
                  enddo         !it
                enddo           !itp
              else              !we need to use mode2 because zwz is not real now.
                call matzwz( zw(1:ngb,1:ngb), zmel, ntp0,nstate,ngb,
     o           zwz(ix,1:nstate,1:ntp0))
              endif
              if(debug) write(6,*)' sumzw=',sum(abs(zw))
            enddo               !ix
            if(verbose()>50) write(*,'("xxx:6.1 before matzwz in ix cycle ",$)')
            if(verbose()>50) call cputid(0)
            if(debug) write(6,*)' sumzmel=',ngb, nstate, ntp0,sum(abs(real(zmel))),sum(abs(imag(zmel)))

c---------------------------------------------------------------
c S[i,j] <psi(q,t) |psi(q-rk,n) B(rk,i)>
c                Wc(k,0)(i,j) > <B(rk,j) psi(q-rk,n) |psi(q,t)>
c needed to take care of the singularity in the w' integration
c when w-e(q-rk,n) is small
c---------------------------------------------------------------
            if(screen) then
              zwz00 = zwz0
              zwz0  = 0d0
              do ix = 1,nx
                zwz(ix,:,:)=zwz(ix,:,:) - zwz00
              enddo
            endif

c-------------------------------------------------
c loop over w in SEc(qt,w)
c-------------------------------------------------
c$$$        if(ua_auto) then
c$$$          allocate(uaa(nstate,ntq))
c$$$          do itp = 1,ntq
c$$$            do  it = 1,nstate
c$$$              ratio = abs(zwz(niw,it,itp)/zwz0(it,itp))
c$$$              call gen_uaa(ratio,freqx(niw),  uaa(it,itp))
c$$$              if(verbose()>45) then
c$$$                write(6,"(' it itp uaa=',2i4,12f8.4)") it,itp,uaa(it,itp)
c$$$              elseif(verbose()>40.and.mod(it,10)==1.and.mod(itp,10)==1) then
c$$$                write(6,"(' it itp uaa=', 2i4,12f8.4)") it,itp,uaa(it,itp)
c$$$              endif
c$$$            enddo
c$$$          enddo
```

```
c$$$        endif
            allocate(zwzs(npm*nx))
            do iw = iwini,iwend
c frequency integration along the imaginary axis, s. wint.f
c for each e(q-rk,n) and w in SEc(qt,w)
              do 1385  itp = 1,ntq
                do 1387 it = 1,nstate
                  we =.5d0*( omega(itp,iw) -ekc(it)) != .5d0*( eq(itq(itp),ip)+2d0*(dble(iw)-shtw)*deltaw-ekc(it))
                  if(verbose()>50) then
                    do  ix = 1,niw
                      ratio  = abs(zwz(ix,it,itp)/zwz0(it,itp))
                      freqw1 = (1d0 - freqx(ix))/ freqx(ix)
                      ua2_(ix) = sqrt(- 1d0/freqw1*log(ratio))
                    enddo
                    write(6,"(' sxcf_fal2: ua=sqrt(1/w1*log(v0/v1))=',12f8.4)") ua2_(1:niw)
                  endif
c          if(ua_auto) then
c            call gen_ua(abs(zwz(niw,it,itp)/zwz0(it,itp)), niw,freqx, expa_,ua_)
c            if(iw==ini) then
c            if(verbose()>45) then
c              write(6,"(' it itp ua_=',2i4,12f8.4)")it,itp,ua_
c            elseif(verbose()>40.and.mod(it,20)==1.and.mod(itp,20)==1) then
c              write(6,"(' it itp ua_=',3i4,12f8.4)")it,itp,ua_
c            elseif(irot==1.and.mod(it,10)==1.and.itp==it) then
c              write(6,"(' it itp ua_=',3i4,12f8.4)")it,itp,ua_
c            endif
c            endif
c          endif
c$$$              if(ua_auto) then
c$$$                ua_ = .5d0*uaa(it,itp)
c$$$                call gen_expa(niw,freqx,ua_,  expa_)
c$$$              endif
                  esmrx = esmr
                  if(it <= nctot) esmrx = 0d0
                  do ix=1,nx
                    zwzs(ix   ) = dreal( zwz(ix,it,itp)) ! w(iw) + w(-iw) symmetric part
                    if(npm==2) then
                      zwzs(ix+nx) = dimag( zwz(ix,it,itp)) ! w(iw) - w(-iw)
                    endif
                  enddo
c                  if(GaussSmear) then
                    zwzi(it,itp) =
     &               wintzsg_npm(npm, zwzs,zwz0(it,itp),freqx,wx,ua_,expa_,we,nx, esmrx)
c                  else
c                    if(npm==2)
c     &                call rx( ' ###Not impliment wintzav for npm=2. Use Gausssmear.')
c                    zwzi(it,itp) =
c     &                wintzav( zwzs,zwz0(it,itp),freqx,wx,ua_,expa_,we,nx, esmrx)
c                  endif
c    .    wintz (zwz(1,it,itp),zwz0(it,itp),freqx,wx,ua,expa,we,nx)
ccccccccccccccccccccccccccccccc
c          if(verbose()>45) then
c          if(it==50.and.itp==1) then
c          write(6,"(' it itp abs(zwzi)=',2i4,12d13.5)")it,itp,abs( zwzi(it,itp))
c          icc=icc+1
c          if(icc==10) stop 'test end'
c          endif
c          endif
cccccccccccccccccccccccccccccc
 1387           continue
 1385         continue
c sum over both occupied and unoccupied states and multiply by weight
              do  itp = 1,ntq
                zsec(iw,itp,ip)  = zsec(iw,itp,ip) + wtt*sum(zwzi(:,itp))
              enddo
c end of SEc w-loop
            enddo
            deallocate(zwzs)
            if(debug) then
              write(6,*)' ntq nstate sum(zwzi)=',ntq,nstate,sum(zwzi)
              write(6,*)' ntq nstate sum(zwz )=',ntq,nstate,sum(zwz)
              do itp = 1,ntq
                write(6,'(" zsec=",i3,6d15.7)') itp,zsec(iwini:iwini+2,itp,ip)
              enddo
            endif
            deallocate(zwz,zwz0,zwzi)

c===============================================================================
c contribution to SEc(qt,w) from the poles of G
c===============================================================================
!    We assume freq_r(i) == -freq_r(-i) in this code. feb2006
c---------------------------------------
c maximum ixs finder
c---------------------------------------
c      write(6,*)' ekc at nt0p nt0m+1=', ekc(nt0p),ekc(nt0m+1)
c      write(6,*)'  nt0p nt0m+1=', nt0p, nt0m+1
```

```
           ixsmx =0
           ixsmin=0
           do 3001 iw  = iwini,iwend
             do 3002 itp = 1,ntq
               omg = omega(itp,iw)
               if (omg < ef) then
                 itini= 1
                 itend= nt0p
               else
                 itini= nt0m+1
                 itend= nstate
               endif
               do 3011 it= itini,itend
                 esmrx = esmr
                 if(it<=nctot) esmrx = 0d0
                 wfac = wfacx2(omg,ef, ekc(it),esmrx)
                 if(gausssmear) then
                   if(wfac<wfaccut) cycle
                   we = .5d0*(omg-weavx2(omg,ef,ekc(it),esmr))
                 else
                   if(wfac==0d0) cycle
                   if(omg>=ef) we = max( .5d0*(omg-ekc(it)), 0d0) ! positive
                   if(omg< ef) we = min( .5d0*(omg-ekc(it)), 0d0) ! negative
                 endif
                 do iwp  = 1,nw ! may2006
                   ixs = iwp   ! ixs = iwp= iw+1
c                 write (*,*) 'xxx freq we=',freq_r(iwp),abs(we)
                   if(freq_r(iwp) > abs(we)) exit
                 enddo
c This change is because G(omega-omg') W(omg') !may2006
c             if(ixs>ixsmx  .and. omg<=ef ) ixsmx  = ixs
c             if(ixs>ixsmin .and. omg> ef ) ixsmin = ixs
                 if(ixs>ixsmx  .and. omg>=ef ) ixsmx  = ixs
                 if(ixs>ixsmin .and. omg< ef ) ixsmin = ixs
                 wexx  = we
                 if(ixs+1 > nw) then
                   write (*,*) ' nw_i ixsmin',nw_i, ixsmin
c                   write (*,*) ' wexx, dw ',wexx,dw
                   write (*,*) ' omg ekc(it) ef ', omg,ekc(it),ef
Cstop2rx 2013.08.09 kino                stop ' sxcf 222: |w-e| out of range'
                   call rx( ' sxcf 222: |w-e| out of range')
                 endif
 3011          continue
 3002        continue          !end of SEc w and qt -loop
 3001      continue            !end of SEc w and qt -loop
           if(nw_i==0) then
             nwxi = 0
             nwx  = max(ixsmx+1,ixsmin+1)
           else
             nwxi = -ixsmin-1
             nwx  =  ixsmx+1
           endif
           if (nwx > nw   ) then
             call rx( ' sxcf nwx check : |w-e| > max(w)')
           endif
           if (nwxi < nw_i) then
             call rx( ' sxcf nwxi check: |w-e| > max(w)')
           endif
           if(debug) write(6,*)' nwxi nwx nw=',nwxi,nwx,nw

C... Find nt_max ------------------------------------
           nt_max=nt0p         !initial nt_max
           do 4001 iw  = iwini,iwend
             do 4002 itp = 1,ntq
               omg     = omega(itp,iw)
               if (omg > ef) then
                 do  it = nt0m+1,nstate ! nt0m corresponds to efm
                   wfac = wfacx2(ef,omg, ekc(it),esmr)
                   if( (gausssmear.and.wfac>wfaccut)
     &               .or.(.not.gausssmear.and.wfac/=0d0)) then
                    if (it > nt_max) nt_max=it ! nt_max is  unocc. state
                   endif       ! that ekc(it>nt_max)-omega > 0
                 enddo
               endif
 4002        continue
 4001      continue

C... Set zw3 or zwz ----------------------------------
           zwz3mode=.true.
           if(iwend-iwini>2) then
             zwz3mode=.false.
           endif
           if(zwz3mode) then
             allocate( zw3(ngb,ngb,nwxi:nwx))
             do ix = nwxi,nwx  ! real frequency w'-loop
               nrec=ix-nw_i+1
               if(debug) write(6,*)' wvr3 nrec=',nrec,nblochpmx,kx,ix,nw
```

```
                read(ifrcw,rec=nrec) zw
                zw3(1:ngb,1:ngb,ix) = zw(1:ngb,1:ngb)
                if(evaltest()) then
                  write(6,"('iii --- EigenValues for zw --------')")
                  allocate(ebb(ngb))
                  call diagcvh2((zw(1:ngb,1:ngb)-transpose(dconjg(zw(1:ngb,1:ngb))))/2d0/img,
     &             ngb, ebb)
                  do ii=1,ngb
                    if(abs(ebb(ii))>1d-8.and.ebb(ii)>0) then
                      write(6,"('iii1xxx:  iw ii eb=',2i4,d13.5)") ix,ii,ebb(ii)
                    else
                      write(6,"('iii1:  iw ii eb=',2i4,d13.5)") ix,ii,ebb(ii)
                    endif
                  enddo
                  deallocate(ebb)
                endif
              enddo
              deallocate(zw)
            else
              nstatex= max(ntp0,nt_max)
              if(allocated(zwz)) deallocate(zwz)
              allocate( zwz(nwxi:nwx,1:nstatex,ntp0) )
              do      ix = nwxi,nwx
                nrec= ix-nw_i+1
                read(ifrcw,rec=nrec) zw ! Readin (W-v)(k,w')(i,j) at k and w' on imag axis
c zwz = S[i,j] <psi(q,t) |psi(q-rk,n) B(rk,i)> Wc(k,iw')(i,j) > <B(rk,j) psi(q-rk,n) |psi(q,t)>
                call matzwz(zw(1:ngb,1:ngb), zmel(1:ngb,1:nstatex,1:ntp0), ntp0,nstatex,ngb,
     o           zwz(ix,1:nstatex,1:ntp0))
! zmel (ngb, nstate, ntp0)
              enddo
              deallocate(zmel)
              deallocate(zw)
            endif
c----------------------------------------------
            if(screen) then
              if(zwz3mode) call rx( ' this mode is not implimented')
              do ix = nw_i,nwx
                zwz(ix,:,:)=zwz(ix,:,:) - zwz00
              enddo
              deallocate(zwz00)
            endif

c------------------------------
c loop over w and t in SEc(qt,w)
c------------------------------
            if(debug) write(6,*)' sss ngb, nstate, ntp0=',ngb,nstate,ntp0
            if(debug) write(6,*)' sss zmel=',sum(abs(zmel(:,:,:)))

            if(verbose()>50) write(*,'("10 wfacx  iw,itp,it cycles ",$)')
            if(verbose()>50) call cputid(0)
            do 2001 iw  = iwini,iwend
              do 2002 itp = 1,ntq
                if(debug) write(6,*)'2011 0 zmel=',sum(abs(zmel(:,:,:)))
                omg = omega(itp,iw)
                if (omg >= ef) then
                  itini= nt0m+1
                  itend= nt_max
                  iii=  1
                else
                  itini= 1
                  itend= nt0p
                  iii= -1
                endif

                do 2011 it= itini,itend
                  if(debug) write(6,*)'2011 1 loop--- it=',iw,itp,it,sum(abs(zmel(:,:,:)))
                  esmrx = esmr
                  if(it<=nctot) esmrx = 0d0
                  wfac = wfacx2(omg,ef, ekc(it),esmrx)
                  if(gausssmear) then
                    if(wfac<wfaccut) cycle
                    we = .5d0*abs(omg-weavx2(omg,ef, ekc(it),esmr))
                  else
                    if(wfac==0d0) cycle
                    if(omg>=ef) we = 0.5d0* abs(max(omg-ekc(it), 0d0)) ! positive
                    if(omg< ef) we = 0.5d0* abs(min(omg-ekc(it), 0d0)) ! negative
                  endif

                  wfac= iii* wfac*wtt
c three-point interpolation for Wc(we)
                  do iwp = 1,nw
                    ixs=iwp
                    if(freq_r(iwp)>we) exit
                  enddo
                  if(nw_i==0) then
                    if(ixs+1>nwx) then
                      write(6,*)' ixs,nwx, we =',ixs,nwx,we
                      call rx( ' sxcf: ixs+1>nwx xxx2')
                    endif
                  else          !   write(6,*)" ixs nwxi=",ixs,nwxi,freq_r(ixs-1),we,freq_r(ixs)
                    if(omg >=ef .and. ixs+1> nwx ) then
                      write(6,*)'ixs+1 nwx=',ixs+1,nwx
                      call rx( ' sxcf: ixs+1>nwx yyy2a')
                    endif
                    if(omg < ef .and. abs(ixs+1)> abs(nwxi) ) then
                      write(6,*)'ixs+1 nwxi=',ixs+1,nwxi
                      call rx( ' sxcf: ixs-1<nwi yyy2b')
                    endif
                  endif

                  iir=1
                  if(omg < ef .and. nw_i/=0) iir = -1 !May2006 because of \int d omega' G(omega-omega') W(omega')
                  if(zwz3mode) then
                    zwz3=(0d0,0d0)
                    if(debug) write(6,"('wwwwwww ixs=',10i4)"),ixs,igb2,it,itp
                    if(debug) write(6,*)'2011 www zmel aaa=',sum(abs(zmel(:,:,:)))
                    do ix = ixs, ixs+2
                      do igb2=1,ngb
                        zz2 = sum(dconjg(zmel(1:ngb,it,itp))*zw3(1:ngb,igb2,iir*(ix-1)) )
                        zwz3(ix-ixs+1) = zwz3(ix-ixs+1)+zz2 *zmel(igb2,it,itp)
                      enddo     !igb2
                    enddo       !ix
                    if(debug) write(6,"('w xxxxxxxxxxxxx ixs loopend=',i4)"),ixs
                    if(debug) write(6,*)zwz3(1:3) !,freq_r(ixs-1),zwz3(1:3)
                    if(debug) write(6,*)'we frez zwz3=', we,ixs,freq_r(ixs-1:ixs+1)
                    if(debug) write(6,*)'2011 bbb www zmel=',sum(abs(zmel(:,:,:)))

                    zsec(iw,itp,ip) = zsec(iw,itp,ip)
     &               + wfac *alagr3zz(we,freq_r(ixs-1),zwz3) !faleev

                    if(debug) write(6,*)'2011 ccc www zmel=',sum(abs(zmel(:,:,:)))
                    if(debug) write(6,"('wwwwwww eo zsecsum')")
                  else
                    zwzz(1:3) = zwz(iir*(ixs-1):iir*(ixs+1):iir, it,itp)
                    zsec(iw,itp,ip) = zsec(iw,itp,ip)
     &               + wfac*alagr3zz(we,freq_r(ixs-1),zwzz)
                  endif
 2011           continue
 2002         continue          !end of SEc w and qt -loop
 2001       continue            !end of SEc w and qt -loop
            if(debug) write(6,*)' end of do 2001'
            if(verbose()>50) then
              write(*,'("11 after alagr3zz iw,itp,it cycles ",$)')
              call cputid(0)
            endif
            if(debug) then
              do itp = 1,ntq
                write(6,'(" zsec=",i3,6d15.7)') itp,zsec(iwini:iwini+2,itp,ip)
              enddo
            endif
            if(zwz3mode) then
              deallocate(zmel,zw3)
            else
              deallocate(zwz)
            endif
 1000     continue
c      if(newaniso) ifvcoud =iclose('Vcoud.'//charnum5(kx))
          ifvcoud =iclose('Vcoud.'//charnum5(kx))
          if(.not.exchange) then
            ifrcw  = iclose('WVR.'//charnum5(kx))
            ifrcwi = iclose('WVI.'//charnum5(kx))
          endif
 1100   continue                ! end of k-loop
        if (tote) then
          exx = exx + wik(ip) * exxq * 0.25d0
        endif
        if (allocated(zz)) deallocate(zz)
        if (allocated(zmel)) deallocate(zmel)
        if (allocated(zzmel))deallocate(zzmel)
        if (allocated(zw)) deallocate(zw)
        if (allocated(zwz)) deallocate(zwz)
        if (allocated(zwz0)) deallocate(zwz0)
        if (allocated(zwzi)) deallocate(zwzi)
        if (allocated(zwz00)) deallocate(zwz00)
        if (allocated(w1p)) deallocate(w1p)
        if (allocated(w2p)) deallocate(w2p)
        if (allocated(w3p)) deallocate(w3p)
        if (allocated(z1p)) deallocate(w1p)
        if (allocated(vcoul)) deallocate(vcoul)
        if (allocated(vcoult)) deallocate(vcoul)
c      if (allocated(zmel1)) deallocate(zmel1)
        if (allocated(zmel3)) deallocate(zmel3)
        if (allocated(zw_)) deallocate(zw_)
        if (allocated(zwz2)) deallocate(zwz2)
c      if (allocated(zw2)) deallocate(zw2)
        if (allocated(zmel2)) deallocate(zmel2)
        if (allocated(zw3)) deallocate(zw3)
        if (allocated(uaa)) deallocate(uaa)
 1001 continue
c      if (allocated(expikt)) deallocate(expikt)
      end
```

4.23 gwsrc/sxcf_fal2.sc.F File Reference

Data Types

- module m_sxcfsc

    this module is only because name=name argument binding. No data

Functions/Subroutines

- subroutine get_nwx (omega, ntq, ntqxx, nt0p, nt0m, nstate, freq_r, nw_i, nw, esmr, ef, ekc, wfaccut, nctot, nband, debug, nwxi, nwx, nt_max)

4.23.1 Function/Subroutine Documentation

4.23.1.1 subroutine get_nwx (real(8), dimension(ntq), intent(in) *omega*, integer, intent(in) *ntq*, integer, intent(in) *ntqxx*, integer, intent(in) *nt0p*, integer, intent(in) *nt0m*, integer, intent(in) *nstate*, real(8), dimension(nw_i:nw), intent(in) *freq_r*, integer, intent(in) *nw_i*, integer, intent(in) *nw*, real(8), intent(in) *esmr*, real(8), intent(in) *ef*, real(8), dimension(nctot+nband), intent(in) *ekc*, real(8), intent(in) *wfaccut*, integer, intent(in) *nctot*, integer, intent(in) *nband*, logical *debug*, integer, intent(out) *nwxi*, integer, intent(out) *nwx*, integer, intent(out) *nt_max*)

Parameters

in	*nctot*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nw_i*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nw*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nstate*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nt0p*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nt0m*	Determine indexes of a range for calculation. It is better to clean this up...
in	*ntq*	Determine indexes of a range for calculation. It is better to clean this up...
in	*nband*	Determine indexes of a range for calculation. It is better to clean this up...
in	*ntqxx*	Determine indexes of a range for calculation. It is better to clean this up...

Definition at line 1314 of file sxcf_fal2.sc.F.

Here is the caller graph for this function:

4.24 sxcf_fal2.sc.F

```
!> this module is only because name=name argument binding. No data
      module m_sxcfsc
      contains
      subroutine sxcf_fal3_scz(kount,qip,itq,ntq,ef,esmr,
     i nsp,isp,
     i qbas,ginv,
     i qibz,qbz,wk,nstbz,irkip,nrkip,
     i freq_r,nw_i,nw, freqx,wx,dwdummy,
     i ecore,
     i nlmto,nqibz,nqbz,nctot,
     i nbloch,ngrp,niw,nq,
     i nblochpmx ,ngpmx,ngcmx,
     i wgt0,nq0i,q0i,symgg, alat, nband, ifvcfpout,
     i exchange,screen,cohtest, ifexsp,
     i nbmx,ebmx,
     i wklm,lxklm,
     i eftrue,
     i jobsw,                   != iSigma_en
     i hermitianw,
     o zsec,coh,nbandmx)
      use m_readqg,only   : readqg0
      use m_readeigen,only: readeval
      use m_keyvalue,only   : getkeyvalue
      use m_zmel,only     : get_zmelt,
     i ppovlz,
     o zmel,zmeltt
      implicit none
!> \brief
!! Calcualte full simga_ij(e_i)= <i|Re[Sigma](e_i)|j>
!! ---------------------
!! \param exchange
!!   - T : Calculate the exchange self-energy
!!   - F : Calculate correlated part of the self-energy
!! \param zsec
!!   - S_ij= <i|Re[S](e_i)|j>
!!   - Note that S_ij itself is not Hermite becasue it includes e_i.
!!     i and j are band indexes
!! \param coh dummy
!! \param screen dummy
!!
!! \remark
!!
!! \verbatim
!! Jan2013: eftrue is added.
!!   ef=eftrue(true fermi energy) for valence exchange and correlation mode.
!!   but ef is not the true fermi energy for core-exchange mode.
!!
!! Jan2006
!!     "zsec from im-axis integral part"  had been symmetrized as
!!     &        wtt*.5d0*(   sum(zwzi(:,itp,itpp))+ !S_{ij}(e_i)
!!     &        dconjg( sum(zwzi(:,itpp,itp)) )   ) !S_{ji}^*(e_j)= S_{ij}(e_j)
!!     However, I now do it just the 1st term.
!!     &        wtt* sum(zwzi(:,itp,itpp))   !S_{ij}(e_i)
!!     This is OK because the symmetrization is in hqpe.sc.F
!!     Now zsec given in this routine is simply written as <i|Re[S](e_i)|j>.
!!     ( In the version until Jan2006 (fpgw032f8), only the im-axis part was symmetrized.
!!     But it was not necessary from the begining because it was done in hqpe.sc.F
!!
!!     (Be careful as for the difference between
!!     <i|Re[S](e_i)|j> and transpose(dconjg(<i|Re[S](e_i)|j>)).
!!     ---because e_i is included.
!!     The symmetrization (hermitian) procedure is inlucded in hqpe.sc.F
!!
!!     NOTE: matrix element is given by "call get_zmelt". It returns  zmelt or zmeltt.
!!
!! jobsw switch
!!  1-5 scGW mode.
!!   diag+@EF      jobsw==1 SE_nn'(ef)+delta_nn'(SE_nn(e_n)-SE_nn(ef))
!!   xxx modeB (Not Available now)  jobsw==2 SE_nn'((e_n+e_n')/2)  !we need to recover comment out for jobsw==2, and test.
!!   mode A        jobsw==3 (SE_nn'(e_n)+SE_nn'(e_n'))/2 (Usually usued in QSGW).
!!   @Ef           jobsw==4 SE_nn'(ef)
!!   diagonly      jobsw==5 delta_nn' SE_nn(e_n) (not efficient memoryuse; but we don't use this mode so often).
!!
!! Output file in hsfp0 should contain hermitean part of SE
!!    ( hermitean of SE_nn'(e_n) means SE_n'n(e_n')^* )
!!             we use that zwz(itp,itpp)=dconjg( zwz(itpp,itp) )
!! Caution! npm=2 is not examined enough...
!!
!! Calculate the exchange part and the correlated part of self-energy.
!! T.Kotani started development after the analysis of F.Aryasetiawan's LMTO-ASA-GW.
!! We still use some of his ideas in this code.
!!
!! See paper
!! [1]T. Kotani and M. van Schilfgaarde, ??Quasiparticle self-consistent GW method:
!!     A basis for the independent-particle approximation, Phys. Rev. B, vol. 76, no. 16, p. 165106[24pages], Oct. 2007.
!! [2]T. Kotani, Quasiparticle Self-Consistent GW Method Based on the Augmented Plane-Wave
!!    and Muffin-Tin Orbital Method, J. Phys. Soc. Jpn., vol. 83, no. 9, p. 094711 [11 Pages], Sep. 2014.
!!
!! ------------------------------------------------------------------------------
!! Omega integral for SEc
!!   The integral path is deformed along the imaginary-axis, but together with contribution of poles.
!!   See Fig.1 and around in Ref.[1].
!!
!! ---Integration along imaginary axis.---
!!   ( Current version for it, wintzsg_npm, do not assume time-reversal when npm=2.)
!!   Integration along the imaginary axis: -----------------
!!    (Here is a memo by F.Aryasetiawan.)
!!     (i/2pi) < [w'=-inf,inf] Wc(k,w')(i,j)/(w'+w-e(q-k,n) >
!!    Gaussian integral along the imaginary axis.
!!    transform: x = 1/(1+w')
!!     this leads to a denser mesh in w' around 0 for equal mesh x
!!    which is desirable since Wc and the lorentzian are peaked around w'=0
!!     wint = - (1/pi) < [x=0,1] Wc(iw') (w-e)x^2/{(w-e)^2 + w'^2} >
!!
!!     the integrand is peaked around w'=0 or x=1 when w=e
!!     to handel the problem, add and substract the singular part as follows:
!!     wint = - (1/pi) < [x=0,1] { Wc(iw') - Wc(0)exp(-a^2 w'^2) }
!!     * (w-e)/{(w-e)^2 +w'^2}x^2 >
!!     - (1/2) Wc(0) sgn(w-e) exp(a^2 (w-e)^2) erfc(a|w-e|)
!!
!!     the second term of the integral can be done analytically, which
!!     results in the last term a is some constant
!!
!!     when w = e, (1/pi) (w-e)/{(w-e)^2 + w'^2} ==> delta(w') and
!!     the integral becomes -Wc(0)/2
!!     this together with the contribution from the pole of G (s.u.)
!!     gives the so called static screened exchange -Wc(0)
!!
!! ---Integration along real axis (contribution from the poles of G: SEc(pole))
!!    See Eq.(34),(55), and (58) and around in Ref.[1]. We now use Gaussian Smearing.
!! ------------------------------------------------------------------------------
!! \endverbatim
!! \verbatim
!!
!! ---------------------------------------------
!!     q     =qip(:,iq)  = q-vector in SEc(q,t).
!!    itq     = states t at q
!!    ntq     = no. states t
!!    eq      = eigenvalues at q
!!     ef      = fermi level in Rydberg
!!   WVI, WVR: direct access files for W. along im axis (WVI) or along real axis (WVR)
!!   freq_r(nw_i:nw)   = frequencies along real axis. freq_r(0)=0d0
!!
!!    qbas    = base reciprocal lattice vectors
!!    ginv    = inverse of qbas s. indxrk.f
!!
!!     wk     = weight for each k-point in the FBZ
!!    qbz     = k-points in the 1st BZ
!!
!!    wx      = weights at gaussian points x between (0,1)
!!     ua_      = constant in exp(-ua^2 w'^2) s. wint.f
!!     expa    = exp(-ua^2 w'^2) s. wint.f
!!
!!    irkip(k,R,nq) = gives index in the FBZ with k{IBZ, R=rotation
!!
!!   nqibz   = number of k-points in the irreducible BZ
!!   nqbz    =                           full BZ
!!    natom   = number of atoms
!!    nctot   = total no. of allowed core states
!!    nbloch  = total number of Bloch basis functions
!!    nlmto   = total number of MTO+lo basis functions
!!    ngrp    = no. group elements (rotation matrices)
!!    niw     = no. frequencies along the imaginary axis
!!    nw_i:nw  = no. frequencies along the real axis. nw_i=0 or -nw.
!!    zsec(itp,itpp,iq)> = <psi(itp,q(:,iq)) |SEc| psi(iq,q(:,iq)>
!!
!! ---------------------------------------------
!! \endverbatim
      integer:: dummy4doxygen

! input variables
      logical, intent(in) :: exchange,screen,cohtest
      integer, intent(in) :: ntq,nqbz,nqibz,ngrp,nq,niw !,natom
      integer, intent(in) :: nband,nlmto,nq0i,nctot,isp,nsp !,mdim(*) !,nlnmx
      integer, intent(in) :: ifvcfpout,nbloch,nblochpmx !nl,nnc, nclass
      integer, intent(in) :: itq(ntq) !,nstar(nqibz) !miat(natom,ngrp),mdimx,
      integer, intent(in) :: irkip(nqibz,ngrp,nq),nrkip(nqibz,ngrp,nq)
      integer, intent(in) :: kount(nqibz,nq),ngpmx,ngcmx,ifexsp,jobsw
      integer, intent(in) :: nbmx(2) !,nlnmv(*),nlnmc(*)!,iclass(*),icore(*)
      integer, intent(in) :: nstbz(nqbz) !,nomx !,nkpo,nnmx,imdim(*)ncore(*),
      integer, intent(in) :: lxklm !,invg(ngrp) !nnr(nkpo),nor(nkpo),
c      integer, intent(in) :: il(*),in(*),im(*),nn_,lx(*),nx_(*),nxx_ !,nlnm(*)
      real(8), intent(in) :: wgt0(nq0i,ngrp),symgg(3,3,ngrp)
      real(8), intent(in) :: q0i(1:3,1:nq0i),alat,ecore(nctot) !shtvg(3,ngrp),
      real(8), intent(in) :: qbas(3,3),ginv(3,3)
      real(8), intent(in) :: wk(nqbz),qibz(3,nqibz) !tiat(3,natom,ngrp),
      real(8), intent(in) :: qbz(3,nqbz),freqx(niw),wx(niw),ef,esmr,dwdummy
      real(8), intent(in) :: ebmx(2),wklm((lxklm+1)**2) !,qrr(3,nkpo)
      real(8), intent(in) :: qip(3,nq),eftrue

c      integer,intent(in):: iwini,iwend
c      real(8),optional::exx
```

```

! output variables
c      real(8),intent(in),optional:: freqsig(iwini:iwend)
      integer, intent(in) ::nbandmx(nq)
      complex(8), intent(out),optional :: zsec(ntq,ntq,nq) , coh(ntq,nq)
c      complex(8), intent(out),optional :: zsecd(iwini:iwend,ntq,nq)

! local variables
c     complex(8) :: zsecx(ntq,ntq,nq)
c      complex(8), intent(in) :: pomatr(nnmx,nomx,nkpo)
c$$$      logical :: ua_auto !fixed to be .false.
c      real(8)::ppbrd ( 0:nl-1, nn_, 0:nl-1,nn_, 0:2*(nl-1),1:nxx_, 1:nsp*nclass)

      integer :: ifrcw,ifrcwi
      logical :: initp=.true.
      real(8),allocatable:: vcoud(:)

      integer :: ip, it, itp, i, ix, kx, irot, kr
      integer :: nt0p, nt0m,nstate , nbmax, ntqxx !iatomp(natom),
      integer :: nt,nw,ixs,iw,ivc,ifvcoud,ngb0
      integer :: nprecx,mrecl,ifwd,nrot,nwp,nw_i,ierr
      integer :: nstatetot,iqini,iqend, ngb,ngc !nbcut,
      integer :: invr,nbmxe,ia,nn,ntp0,no,itpp,nrec,npm,itini,itend
      integer :: iwp,nwxi,nwx,iir, igb1,igb2,ix0,iii

      real(8) :: tpi, ekc(nctot+nband),ekq(nband), det, q(3),ua_
      real(8) :: expa_(niw), qxx(3), symope(3,3),shtv(3) !tr(3,natom),
      real(8) :: efp,efm,wtt,wfac,we,esmrx,qbasinv(3,3)
      real(8) :: qvv(3),pi,fpi,eq(nband),omega(ntq),quu(3),freqw,ratio
      real(8) :: qibz_k(3),qbz_kr(3),ddw,vc,omega0,omg

      complex(8) :: cphiq(nlmto,nband), cphikq(nlmto,nband)
      complex(8) :: zwzs0,zz2,zwz3(3)

! local arrays
      real(8),intent(in) :: freq_r(nw_i:nw)
      real(8),allocatable :: drealzzzmel(:,:,:), dimagzzzmel(:,:,:),uaa(:,:)
      complex(8),allocatable :: vcoul(:,:),w3p(:,:,:)
      complex(8),allocatable :: zzzmel(:,:,:),zw (:,:)
      complex(8),allocatable :: zwz(:,:,:,:), zwz0(:,:,:),zwzi(:,:,:)
      complex(8),allocatable :: zwix(:,:),zwzix(:,:,:),zmel1(:) !,expikt(:)
      complex(8), allocatable :: zmel1_(:,:,:), zw3(:,:,:),zw3x(:,:)
      complex(8), allocatable :: zwz4(:,:),zwz44(:,:),pomat(:,:), zwzs(:)
      complex(8),allocatable :: ppovl(:,:),zcousq(:,:)
      complex(8),allocatable :: z1r(:,:),z2r(:,:),w3pi(:,:)

      real(8), parameter :: wfaccut=1d-8
      complex(8), parameter :: img=(0d0,1d0)

! external function
c      logical :: smbasis
c      logical :: test_symmetric_W
c      logical :: GaussSmear !fixed to be T
c      logical :: newaniso !fixed to be T
c      integer :: bzcase !fixed to be 1
      character(5) :: charnum5
      integer :: iopen,iclose
      integer :: invrot
      complex(8) ::  wintzsg_npm !wintzav,
      integer :: nocc
      real(8) :: wfacx
      real(8) :: wfacx2
      real(8) :: weavx2
      complex(8) :: alagr3z
      complex(8) :: alagr3z2

      integer:: ndummy1,ndummy2,nlmtobnd,nt0
      real(8):: wexx
c      complex(8),allocatable :: z1p(:,:,:),vcoult(:,:)
      logical :: debug, debugp,debug2=.false.
c      logical :: gass           !external
c      real(8):: wgtq0p
      integer::verbose,ififr,ifile_handle
      real(8):: ua2_(niw),freqw1
      integer :: istate,  nt_max !nbcutc,nbcutin,
      real(8):: q_r(3),qk(3),omegat
      logical::  oncew, onceww, eibz4sig,  timemix

      integer,allocatable:: ixss(:,:),iirx(:)
      real(8),allocatable:: we_(:,:),wfac_(:,:)
      complex(8),allocatable:: zw3av(:,:),zmelw(:,:,:)
      integer:: noccx
      real(8)::polinta
      logical,allocatable:: ititpskip(:,:)

      logical:: tote=.false.
      logical:: hermitianw

      real(8),allocatable:: wcorehole(:,:)
      logical:: corehole
      integer:: ifcorehole
      real(8):: tolq=1d-8
c      real(8),allocatable:: ppb(:)
c      allocate( ppb(nlnmx*nlnmx*mdimx*nclass))

c      real(8)::exxq

c-----------------------------------------------------------------
c!TIME0_0000
c      write(6,*)'sxcf_fal3_scz'
      timemix=.false.
      pi  = 4d0*datan(1d0)
      fpi = 4d0*pi
      debug=.false.
      if(verbose()>=90) debug=.true.

c      corehole=.true.
      corehole=.false.

!! core-hole
      if(corehole) then
        ifcorehole=ifile_handle()
        open(ifcorehole,file='CoreHole')
        if(allocated(wcorehole)) deallocate(wcorehole)
        allocate(wcorehole(nctot,nsp))
        do it=1,nctot
          read(ifcorehole,*) wcorehole(it,1:nsp)
        enddo
        close(ifcorehole)
        write(*,*) 'end of reading CoreHole'
      endif

      if(.not.exchange) then
        ifwd = iopen('WV.d',1,-1,0)
        read (ifwd,*) nprecx,mrecl
        ifwd = iclose('WV.d')
!! gauss_img : interpolation gaussion for W(i \omega).
        call getkeyvalue("GWinput","gauss_img",ua_,default=1d0)
        if(debug) write(6,*) ' sxcf_fal3_scz: Gausssmear=T'
        do ix = 1,niw           !! Energy mesh; along im axis.
          freqw     = (1d0 - freqx(ix))/ freqx(ix)
          expa_(ix) = exp(-(ua_*freqw)**2)
        enddo
        npm = 1                 ! npm=1    Timeveversal case
        if(nw_i/=0) npm = 2     ! npm=2 No TimeReversal case. Need negative energy part of W(omega)
      endif

c      call getkeyvalue("GWinput","nbcutlow_sig",nbcut, default=0 )
c      nbcutc=nctot+nbcut
      tpi         = 8d0*datan(1d0)
      if(nctot/=0) ekc(1:nctot)= ecore(1:nctot) ! core
      nlmtobnd    = nlmto*nband
      nstatetot      = nctot + nband


!!== ip loop to spedify external q ==
      do 1001 ip = 1,nq
        if(sum(irkip(:,:,ip))==0) cycle ! next ip
        write (6,*) ip,'  out of ',nq,'  k-points(extrnal q) '
        q(1:3)= qip(1:3,ip)
        call readeval(q,isp,eq)
        do i  = 1,ntq
          omega(i) = eq(itq(i))
        enddo

!! we only consider bzcase()==1
        if(abs(sum(qibz(:,1)**2))/=0d0) call rx( ' sxcf assumes 1st qibz/=0 ')
        if(abs(sum( qbz(:,1)**2))/=0d0) call rx( ' sxcf assumes 1st qbz /=0 ')

!! NOTE total number of
!!      kx loop(do 1100) and irot loop (do 1000) makes all the k mesh points.
!!      When iqini=1 (Gamma point), we use effective W(q=0) defined in the paper.
        iqini=1
        iqend=nqibz             !no sum for offset-Gamma points.
        do 1100 kx = iqini,iqend
          if(sum(irkip(kx,:,ip))==0) cycle ! next kx
!TIME0_01000
          write(6,*) ' ### do 1100 start kx=',kx,' from ',iqini,' through', iqend
c          if( kx <= nqibz ) then
            qibz_k= qibz(:,kx)
c          else
c            qibz_k= 0d0
c          endif
          if(timemix) call timeshow("11111 k-cycle")
          call readqg0('QGcou',qibz_k,ginv,  quu,ngc)
          ngb = nbloch + ngc
          if(debug) write(6,*) ' sxcf: ngb=',ngb,nbloch

!! ===Readin diagonalized Coulomb interaction===
!!  Vcoud file is sequential file Vcoulomb matrix for qibz_k.
!!  A possible choice for paralellization is "Vcoud.ID" files where ID=kx
!!  Vould file is written in hvccfp0.m.F.
!! For correlation, W-v is read instead of Vcoud file (ifrcw,ifrcwi for WVR and WVI)
!! These can be also separeted into WVR.ID and WVI.ID files.
!! NOTE: vcoud and zcousq are in module m_zmelt.
          qxx=qibz_k
c          if(kx<=nqibz) qxx=qibz_k
c          if(kx>nqibz ) qxx=q0i(:,kx-nqibz)
          ifvcoud = iopen('Vcoud.'//charnum5(kx),0,0,0)
          do
            read(ifvcoud) ngb0
            read(ifvcoud) qvv
            if(allocated(vcoud)) deallocate(vcoud)
            allocate( zcousq(ngb0,ngb0),vcoud(ngb0) )
            read(ifvcoud) vcoud
            read(ifvcoud) zcousq
            if(sum(abs(qvv-qxx))<tolq) goto 1133
          enddo
          if(sum(abs(qvv-qxx))>tolq) then
            write(6,*)'qvv =',qvv
            write(6,*)'qxx=',qxx,kx
            call rx( 'sxcf_fal2: qvv/=qibz(:,kx) hvcc is not consistent')
          endif
 1133     continue
          if( ngb0/=ngb ) then  !sanity check
            write(6,*)' qxx ngb0 ngb=',qxx,ngb0,ngb
            call rx( 'hsfp0.m.f:ngb0/=ngb')
          endif
!! ppovlz is used in get_zmel
!! <I|v|J>= \sum_mu ppovl*zcousq(:,mu) v^mu (Zcousq^*(:,mu) ppovl)
!! zmel contains O^-1=<I|J>^-1 factor. zmel(phi phi J)= <phi phi|I> O^-1_IJ
!! ppovlz= O Zcousq
!! (V_IJ - vcoud_mu O_IJ) Zcousq(J, mu)=0, where Z is normalized with O_IJ.
          allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb))
          call readppovl0(qibz_k,ngc,ppovl)
          ppovlz(1:nbloch,:) = zcousq(1:nbloch,:)
          ppovlz(nbloch+1:nbloch+ngc,:)=matmul(ppovl,zcousq(nbloch+1:nbloch+ngc,:))
c             write(6,*)'sumcheck ppovlz 00000 =',sum(abs(ppovlz(:,:)))
          deallocate(zcousq,ppovl)
!! === open WVR,WVI ===
          if(.not.exchange) then
            ifrcw  = iopen('WVR.'//charnum5(kx),0,-1,mrecl)
            ifrcwi = iopen('WVI.'//charnum5(kx),0,-1,mrecl)
          endif
          nrot=0
          do irot = 1,ngrp
c             if( kx <= nqibz) then
              kr = irkip(kx,irot,ip) ! index for rotated kr in the FBZ
              if(kr==0) cycle   ! next irot
              qbz_kr= qbz(:,kr)
c             else
c               kr=-99999         !for sanity check
c               qbz_kr= 0d0
c               if( wgt0(kx-nqibz,irot)==0d0 ) cycle ! next irot
c             endif
            nrot=nrot+1
          enddo
!TIME1_01000 ":BeforDo1000"


!! === loop 1000 over rotations irot ===
          do 1000 irot = 1,ngrp
c             if( kx <= nqibz) then
              kr = irkip(kx,irot,ip) ! index for rotated kr in the FBZ
              if(kr==0) cycle
              qbz_kr= qbz(:,kr)
c             else
c               kr=-99999         !for sanity check
c               qbz_kr= 0d0
c               if( wgt0(kx-nqibz,irot)==0d0 ) cycle
c             endif

!TIME0_1010
!! no. occupied (core+valence) and unoccupied states at q-rk
            qk =  q - qbz_kr
            call readeval(qk, isp, ekq)
            ekc(nctot+1:nctot+nband) = ekq(1:nband)
            nt0 = nocc(ekc,ef,.true.,nstatetot)
            ddw= .5d0
c             if(GaussSmear()) ddw= 10d0
            ddw= 10d0
```

```
           efp= ef+ddw*esmr
           efm= ef-ddw*esmr
c           nt0p = nocc (ekc,efp,.true.,nstatetot)
c           nt0m = nocc (ekc,efm,.true.,nstatetot)
           nt0p = nocc(ekq,efp,.true.,nstatetot)+ nctot
           nt0m = nocc(ekq,efm,.true.,nstatetot)+ nctot
!! nbmx1 ebmx1: to set how many bands of <i|sigma|j>  do you calculate.
!! nbmx2 ebmx2: to restrict num of bands of G to calculate G \times W

           if(exchange) then
             nbmax = nt0p-nctot
           else
             nbmax = nband
             nbmxe = nocc(ekc,ebmx(2),.true.,nstatetot)-nctot
             nbmax  = min(nband,nbmx(2),nbmxe)
             if(initp) then
               write(6,*)' nbmax=',nbmax
               initp=.false.
             endif
           endif
c$$$!! ntqxx is number of bands for <i|sigma|j>.
c$$$            ntqxx = nocc (omega-eftrue,ebmx(1),.true.,ntq)
c$$$!bug -ef is added jan2013
c$$$!previous version do not give wrong results, but inefficient.
c$$$            ntqxx = min(ntqxx, nbmx(1))
c$$$            if(ntqxx<nband) then
c$$$              do i=ntqxx,1,-1   !redudce ntqxx when band tops are degenerated. !sep2012
c$$$                if(omega(i+1)-omega(i)<1d-2) then
c$$$                  ntqxx=i-1
c$$$                else
c$$$                  exit
c$$$                endif
c$$$              enddo
c$$$            endif
c$$$            nbandmx(ip)=ntqxx   !number of bands to be calculated Sep2012.

           ntqxx = nbandmx(ip) !mar2015
           if(debug) write(6,*)' sxcf: nbmax nctot nt0p =',nbmax,nctot,nt0p
           nstate = nctot + nbmax ! = nstate for the case of correlation

!! Get matrix element zmelt= rmelt + img*cmelt, defined in m_zmel.F---
c           if(debug) write(6,*)'zzBBB ppovlz =',sum(abs(ppovlz(:,:))),kx,irot
           if(allocated(zmel)) deallocate(zmel)
           if(allocated(zmeltt)) deallocate(zmeltt)
!TIME1_1010 "Beforeget_zmelt"
! this return zmeltt (for exchange), or zmel (for correlation)
!TIME0_1088
           call get_zmelt(exchange,q,kx,qibz_k,irot,qbz_kr,kr,isp,
     &      ngc,ngb,nbmax,ntqxx,nctot,ncc=0)
           if(debug) write(6,*)' end of get_zmelt'
!TIME1_1088 "get_zmelt"

c$$$!! cccccccccc START: old version, instead of get_zmelt cccccccccccc
c$$$         call  readcphi(q, nlmto,isp, quu, cphikq)
c$$$         if(debug) write(6,*) ' sxcf: 2'
c$$$         do      it = 1,ntq
c$$$            itp        = itq(it)
c$$$            cphiq(1:nlmto,it) = cphikq(1:nlmto,itp)
c$$$            write(*,*)'svvvv ',it, itp, sum(cphiq(:,it))
c$$$         enddo
c$$$               write(*,*)'srrrrr 1c',sum(cphiq(:,1:ntq)),ntq
c$$$
c$$$         call dinv33(qbas,0,qbasinv,det)
c$$$         if(debug) write(6,*) ' sxcf: 1'
c$$$         if(allocated(expikt)) deallocate(expikt)
c$$$         allocate(expikt(natom))
c$$$cccccccccccccccccccccccccccccccccccc
c$$$!!    rotate atomic positions invrot*R = R' + T
c$$$               invr  = invrot (irot,invg,ngrp)
c$$$               tr    = tiat(:,:,invr)
c$$$               iatomp= miat(:,invr)
c$$$               symope= symgg(:,:,irot)
c$$$               shtv  = matmul(symope,shtvg(:,invr))
c$$$!TIME1 "before ppbafp_v2"
c$$$!TIME0
c$$$
c$$$!! -- ppb= <Phi(SLn,r) Phi(SL'n',r) B(S,i,Rr)>
c$$$c                call ppbafp_v2 (irot,ngrp,isp,nsp,
c$$$c     i               il,in,im,nlnm, !w(i_mnl),
c$$$c     d               nl,nn_,nclass,nlnmx,
c$$$c     i               mdimx,lx,nx_,nxx_, !Bloch wave
c$$$c     i               cgr, nl-1,  !rotated CG
c$$$c     i               ppbrd,      !radial integrals
c$$$c     o               ppb)
c$$$               ppb = ppbir(:,irot,isp)
c$$$!! qk = q-rk. rk is inside 1st BZ, not restricted to the irreducible BZ
c$$$               qk =  q - qbz_kr !qbz(:,kr)
```

```
c$$$               call readcphi(qk, nlmto,isp, quu, cphikq)
c$$$!TIME1 "before expikt"
c$$$!TIME0
c$$$
c$$$!! ====================================================
c$$$!!     matrix elements <psi(q,t') | psi(q-rk,t) B(rk,R,i)>
c$$$!!     including the phase factor exp(ik.T)
c$$$!!     B(rot*k,r) = B(k,invrot*r)
c$$$!! ====================================================
c$$$!! phase factors expikt(ia) is for exp(ik.T(R))
c$$$               do ia = 1,natom
c$$$                  expikt(ia) = exp(img*tpi* sum(qibz_k*tr(:,ia)) )
c$$$               end do
c$$$!! matrix elements
c$$$!!   core
c$$$               nt   = nctot + nbmax ! = nstate for the case of correlation
c$$$               allocate( zzzmel(nbloch,nt,ntqxx))
c$$$               call psicb_v2  (icore,ncore,ntqxx,iclass,
c$$$     i              dreal(expikt(1:natom)),dimag(expikt(1:natom)),
c$$$     i              cphiq,
c$$$     i              ppb,
c$$$     i              nlnmv,nlnmc,mdim,
c$$$     i              imdim,iatomp,
c$$$     d              mdimx,nlmto,nbloch,nlnmx,nt,ntqxx,natom,nclass,
c$$$     d              nl,nnc,
c$$$     o              zzzmel)
c$$$               if(debug)  write(6,*) ' sxcf_fal2sc: goto psi2bc1'
c$$$ccccccccc cccccccccccccccccccccccccccccccccccccccc
c$$$               write(*,*)'srrrrr 1',sum(cphikq(1:nlmto,1:ntq))
c$$$               write(*,*)'srrrrr 1',sum(cphiq(1:nlmto,1:ntq))
c$$$               write(*,*)'srrrrr 1',sum(ppb)
c$$$               write(*,*)'srrrrr 1',sum(expikt)
c$$$               write(*,*)'srrrrr 1',sum(zzzmel)
c$$$
c$$$!!   valence
c$$$               call psi2b_v2  (nbmax, ntqxx,iclass,
c$$$     i              dreal(expikt(1:natom)),dimag(expikt(1:natom)),
c$$$     i              cphikq,     !occ    q-rk nband
c$$$     i              cphiq,      !unocc  q    ntq
c$$$     i              ppb,
c$$$     i              nlnmv,nlnmc,mdim,nctot,
c$$$     i              imdim,iatomp,
c$$$     d              mdimx,nlmto,nbloch,nlnmx, nband, nt,ntqxx,
c$$$     d              natom,nclass,
c$$$     o              zzzmel)
c$$$               if(verbose()>50) call timeshow("4 after psi2bc1")
c$$$c               if(debug2) then
c$$$                  write(6,"('sum of zmel abszmel=',4d23.16)") sum(zzzmel),sum(abs(zzzmel) )
c$$$c               end if
c$$$!TIME1 "bfore psi2b_v2"
c$$$!TIME0
c$$$!! -- IPW part.
c$$$               if(debug) write(6,*) ' sxcf_fal1: goto drvmelp2 xxx111'
c$$$               allocate(drealzzzmel(nbloch,nt,ntqxx),dimagzzzmel(nbloch,nt,ntqxx))
c$$$               drealzzzmel=dreal(zzzmel)
c$$$               dimagzzzmel=dimag(zzzmel)
c$$$               deallocate(zzzmel)
c$$$               allocate( rmelt(ngb, nctot+nbmax, ntqxx), ! nstate= nctot+nband
c$$$     &              cmelt(ngb, nctot+nbmax, ntqxx))
c$$$               call drvmelp2( q,             ntqxx, ! q in FBZ
c$$$     i              q-qbz_kr,  nbmax, ! q-rk
c$$$     i              qibz_k,  ! k in IBZ for mixed product basis. rk = symope(qibz_k)
c$$$     i              isp,ginv,
c$$$     i              ngc,ngcmx, ngpmx,nband,itq,
c$$$     i              symope, shtv, qbas, qbasinv,qibz,qbz,nqbz,nqibz,
c$$$     i              drealzzzmel, dimagzzzmel, nbloch, nt,nctot,
c$$$     o              rmelt,cmelt)
c$$$               if(debug) write(6,*) ' sxcf_fal1: end of drvmelp2'
c$$$               deallocate(drealzzzmel,dimagzzzmel)
c$$$               if(verbose()>50) call timeshow("5 after drvmelp")
c$$$               if(nbcut/=0.and.(.not.exchange)) then
c$$$                  do it= nctot+1,nctot+min(nbcut,nbmax)
c$$$                     rmelt(:, it,:) =0d0
c$$$                     cmelt(:, it,:) =0d0
c$$$                  enddo
c$$$               endif
c$$$               write(6,"('sum of rmelt cmelt=',4d23.16)")sum(rmelt),sum(cmelt)
c$$$
c$$$!TIME1 "after drvmelp2"
c$$$!! NOTE:========================================
c$$$!! zmelt = rmelt(igb(qbz_kr), iocc(q), iunocc(q-qbz_kr)) + i* cmelt
c$$$!! iunocc: band index at target  q.
c$$$!! iocc:   band index at intermediate vector qk = q - qbz_kr
c$$$!! igb: index of mixed product basis       at qbz_kr (or written as rk)
c$$$!!   igb=1,ngb
c$$$!!   ngb=nbloch+ngc  ngb: # of mixed product basis
c$$$!!                   nbloch: # of product basis (within MTs)
```

```
c$$$!!                   ngc: # of IPW for the Screened Coulomb interaction.
c$$$!!                   igc is for given
c$$$!! See readgeig in drvmelp2.
c$$$!! ==================================================
c$$$!! smbasis ---need to fix this
c$$$c$$$              if(smbasis()) then !
c$$$c$$$                 ntp0= ntqxx
c$$$c$$$                 nn= nnr(kx)
c$$$c$$$                 no= nor(kx)
c$$$c$$$                 allocate( pomat(nn,no) )
c$$$c$$$                 pomat= pomatr(1:nn,1:no,kx)
c$$$c$$$                 if( sum(abs(qibz_k-qrr(:,kx)))>1d-10 .and.kx <= nqibz ) then
c$$$c$$$                    call rx( 'qibz/= qrr')
c$$$c$$$                 endif
c$$$c$$$                 if(no /= ngb.and.kx <= nqibz) then
c$$$c$$$!!     A bit sloppy check only for kx<nqibz because qibze is not supplied...
c$$$c$$$                    write(6,"(' q  ngb  ',3d13.5,3i5)")  qibz_k,ngb
c$$$c$$$                    write(6,"(' q_r  nn no',3d13.5,3i5)") q_r,nn,no
c$$$c$$$                    call rx( 'x0kf_v2h: POmat err no/=ngb')
c$$$c$$$                 endif
c$$$c$$$                 if(timemix) call timeshow("xxx2222 k-cycle")
c$$$c$$$                 ngb = nn      ! Renew ngb !!!
c$$$c$$$                 allocate ( zmel  (nn, nctot+nbmax, ntp0) )
c$$$c$$$                 call matm( pomat, dcmplx(rmelt,cmelt), zmel,
c$$$c$$$     &                    nn, no, (nctot+nbmax)*ntp0 )
c$$$c$$$                 deallocate(rmelt, cmelt)
c$$$c$$$                 allocate( rmelt(ngb, nctot+nbmax, ntp0), !ngb is reduced.
c$$$c$$$     &                    cmelt(ngb, nctot+nbmax, ntp0) )
c$$$c$$$                 rmelt = dreal(zmel)
c$$$c$$$                 cmelt = dimag(zmel)
c$$$c$$$                 deallocate(zmel,pomat)
c$$$c$$$              else
c$$$c$$$                 nn=ngb
c$$$c$$$                 no=ngb
c$$$c$$$              endif
c$$$              nn=ngb
c$$$              no=ngb
c$$$              if( oncew() ) then
c$$$                 write(6,"('ngb nn no=',3i6)") ngb,nn,no
c$$$              endif
c$$$              if(timemix) call timeshow("22222 k-cycle")
c$$$!! === End of zmelt ; we now have matrix element zmelt= rmelt + img* cmelt ===
c$$$              if(allocated(zzzmel))deallocate(zzzmel) !rmel,cmel)
c$$$              if(debug) write(6,*) ' sxcf: goto wtt'
c$$$              if(debug) write(6,"('sum of rmelt cmelt=',4d23.16)")sum(rmelt),sum(cmelt)
c$$$
c$$$!! === End of zmelt ; we now have matrix element zmelt= rmelt + img* cmelt ===
c$$$!! cccccccccc  END: old version, instead of get_zmelt cccccccccc


!! --- wtt setcion ---
c$$$              if(bzcase()==2)then
c$$$                 if(kx<=nqibz) then
c$$$                    wtt = wk(kr)
c$$$                    if(nstbz(kr)/=0) wtt = wk(kr)*(1d0-wgtq0p()/nstbz(kr))
c$$$                 elseif(kx>nqibz) then !   wtx= wgt0(kx-nqibz,irot)/dble(nqbz)
c$$$                    wtt= wgt0(kx-nqibz,irot)
c$$$                 endif
c$$$              else
c           if(kx<= nqibz) then !  wtx = 1d0
             wtt = wk(kr)
c           else               !  wtx = wgt0(kx-nqibz,irot)
c             wtt = wk(1)*wgt0(kx-nqibz,irot)
c             if(abs(wk(1)-1d0/dble(nqbz))>1d-10) call rx( 'sxcf:wk(1) inconsistent')
c           endif
!!
           if(eibz4sig()) then
             wtt=wtt*nrkip(kx,irot,ip)
           endif

!!--------------------------------------------------------
!! --- exchange section ---
!!--------------------------------------------------------
           if(exchange) then   !At the bottom of this block, cycle do 1000 irot.
!! We use the matrix elements zmeltt. Now given by "call get_zmelt"
!!
c need to check following comments ----
c     S[i,j=1,nbloch] <psi(q,t) |psi(q-rk,n) B(rk,i)>
c     v(k)(i,j) <B(rk,j) psi(q-rk,n) |psi(q,t')>
c
c     > z1p(j,n,t) = S[i=1,nbloch] <psi(q,t) | psi(q-rk,n) B(rk,i)> v(k)(i,j)
c
c     --- screened exchange case
c     if(screen) then
c     allocate( zw (nblochpmx,nblochpmx))
c     ix = 1
c     ! write(*,*)(kx-2)*(nw_w+1)+ix
```

```
c     read(ifrcw,rec=((kx-2)*nw+ix)) zw  ! Readin W(0) - v         !sf 22May02
c     !nw is number of frequency points in general mesh: freq_r(nw), freq_r(1)=0
c     vcoul = vcoul + zw(1:ngb,1:ngb) !c  screen test
c     deallocate(zw)
c     endif
!TIME0_0130
              vc = vcoud(1)     ! save vcoud(1)
              if (kx == iqini) vcoud(1) = wklm(1)* fpi*sqrt(fpi) /wk(kx)
              allocate(z1r(ntqxx,ngb),z2r(ntqxx,ngb),w3pi(ntqxx,ntqxx))
              allocate(w3p(nctot+nbmax,ntqxx,ntqxx))
              do  it = 1, nctot+nbmax
                do  ivc = 1, ngb
                  do  itp = 1, ntqxx
                    z1r(itp,ivc) = zmeltt(it,itp,ivc) * vcoud(ivc)
                    z2r(itp,ivc) = zmeltt(it,itp,ivc)
                  enddo         ! ivc
                enddo           ! it
                call zgemm('N','C',ntqxx,ntqxx,ngb,(1d0,0d0),z1r,ntqxx,
     .           z2r,ntqxx,(0d0,0d0),w3pi,ntqxx)
C              call zprm('w3pi',w3p,ntqxx,ntqxx,ntqxx)
C             Faster, but harder to parallelize
!                        call zqsmpy(11,'N','C',ntqxx,ngb,z1r,ntqxx,z2r,ntqxx,
!     .                       (0d0,0d0),w3pi,ntqxx)
C             call zprm('w3pi',w3p,ntqxx,ntqxx,ntqxx)
                do  itp = 1, ntqxx
                  do itpp = 1, ntqxx
                    w3p(it,itp,itpp) = w3pi(itp,itpp)
                  enddo
                enddo
              enddo
              vcoud(1) = vc     !restore vcoud(1)
              deallocate(z1r,z2r,w3pi)
              if(verbose()>=30) call cputid2(' complete w3p',0)
              deallocate(zmeltt)
              if(debug) then
                do  it  = 1,nctot+nbmax; do  itp = 1,ntqxx
                  write(6,"(' w3p =',2i4,2d14.6)") it,itp,w3p(it,itp,itp)
                enddo;    enddo
              endif
!TIME1_0130 "end_of_w3p"

c$$$#else
c$$$!kino 2014-08-13  !$OMP parallel  private(vc)
c$$$!kino 2014-08-13  !$OMP do
c$$$                     do itp= 1,ntqxx
c$$$                        do it = 1,nctot+nbmax
c$$$                           do ivc=1,ngb
c$$$                              zmeltt(it,itp,ivc) =  sum( zmel(:,it,itp)* ppovlz(:,ivc) )
c$$$                           enddo
c$$$                        enddo
c$$$                     enddo
c$$$!kino 2014-08-13  !$OMP end do
c$$$!kino 2014-08-13  !$OMP do
c$$$                     do 992 itpp= 1,ntqxx
c$$$                        do 993 itp = 1,ntqxx
c$$$                           if(diagonly.and.(itpp/=itp)) cycle
c$$$!! sep2013t a test:c           if(itpp>ntqxxd .and.itp/=itpp) cycle
c$$$                           do 994 it  = 1,nctot+nbmax
c$$$                              w3p(it,itp,itpp) = 0d0
c$$$                              do ivc=1,ngb
c$$$                                 if(ivc==1.and.kx==iqini) then
c$$$                                    vc= wklm(1)* fpi*sqrt(fpi) /wk(kx)
c$$$c     write(6,*)'wklm(1) vc=',wklm(1),vc
c$$$                                 else
c$$$                                    vc= vcoud(ivc)
c$$$                                 endif
c$$$c     zmelt1 =  sum( zmel(:,it,itp)  *ppovlz(:,ivc) )
c$$$c     zmelt2 =  sum( zmel(:,it,itpp) *ppovlz(:,ivc) )
c$$$                                 w3p(it,itp,itpp) = w3p(it,itp,itpp)
c$$$     &                                + vc * zmeltt(it,itp,ivc)*dconjg(zmeltt(it,itpp,ivc))
c$$$                              enddo
c$$$ 994                       continue
c$$$ 993                    continue
c$$$ 992                 continue
c$$$!kino 2014-08-13  !$OMP end do
c$$$!kino 2014-08-13  !$OMP end parallel
c$$$#endif
!KINO                     write(*,*)'kino: w3p checksum=',sum(w3p)
c                      deallocate(zmeltt)
c$$$                  else
c$$$!kino 2014-08-13  !$OMP parallel do
c$$$                     do itpp= 1,ntqxx
c$$$                        do itp = 1,ntqxx
c$$$                           if(diagonly.and.(itpp/=itp)) cycle
c$$$c sep2013t a test:c  if(itpp>ntqxxd .and.itp/=itpp) cycle
c$$$                           do it  = 1,nctot+nbmax
c$$$                              w3p(it,itp,itpp) =dcmplx(
c$$$     &                          sum ( dreal(z1p(:,it,itp))*rmelt(:,it,itpp)
c$$$     &                          +   dimag(z1p(:,it,itp))*cmelt(:,it,itpp) ) ,
c$$$     &                          sum ( dimag(z1p(:,it,itp))*rmelt(:,it,itpp)
c$$$     &                          -   dreal(z1p(:,it,itp))*cmelt(:,it,itpp) ) )
c$$$                           enddo
c$$$                        enddo
c$$$                     enddo
c$$$!kino 2014-08-13  !$OMP end parallel do
c$$$                     deallocate(z1p)
c$$$                  endif
c               deallocate(zmel)
c$$$!!-- Write the Spectrum function for exchange May. 2001
c$$$                  if(ifexsp/=0) then
c$$$                     do it  = 1, nctot+nbmax
c$$$                        do itp = 1,ntqxx
c$$$                           write(ifexsp,"(3i4, 3f12.4, ' ',d23.15,'  ',d23.15)")
c$$$     &                          ip,itp,it, qbz_kr, ekc(it), -wtt*dreal(w3p(it,itp,itp))
c$$$                        enddo
c$$$                     enddo
c$$$                  endif
c$$$!TIME1 "end of write ifsexsp"

!TIME0_0180
!! --- Correct weigts wfac for valence by esmr
              do it = nctot+1, nctot+nbmax
                wfac = wfacx(-1d99, ef, ekc(it), esmr) !gaussian
                w3p(it,1:ntqxx,1:ntqxx) = wfac * w3p(it,1:ntqxx,1:ntqxx)
              enddo

!! apr2015 correct weights for core-hole case
              if(corehole) then
              do it = 1, nctot
                w3p(it,1:ntqxx,1:ntqxx) = wcorehole(it,isp) * w3p(it,1:ntqxx,1:ntqxx)
              enddo
              endif

              do itpp=1,ntqxx
                do itp = 1,ntqxx !S[j=1,nbloch]  z1p(j,t,n) <B(rk,j) psi(q-rk,n) |psi(q,t')>
                  if(jobsw==5.and.(itpp/=itp)) cycle
c sep2013t a test:c  if(itpp>ntqxxd .and.itp/=itpp) cycle
                  zsec(itp,itpp,ip) = zsec(itp,itpp,ip)
     &             - wtt * sum( w3p(:,itp,itpp) )
                enddo
              enddo
              deallocate( w3p)
c$$$                  if(.not.newaniso()) deallocate(vcoul)
!TIME1_0180 "enddo_zsec_wtt_sum"
              cycle             ! next irot do 1000 loop
            endif               ! end of if(exchange)
!! ============== End of exchange section ======================
            if(timemix) call timeshow("33333 k-cycle")
cc!TIME1 "end of exchange section"


!!-----------------------------------------------------------
!!---  correlation section --------------------------------
!!-----------------------------------------------------------
!!  We use the matrix elements zmel, which is given by "call get_zmelt"
!!
!!==============================================================
!! need to check the following notes.
!!     The correlated part of the self-energy:
!!     S[n=all] S[i,j=1,nbloch]
!!     <psi(q,t) |psi(q-rk,n) B(rk,i)>
!!     < [w'=0,inf] (1/pi) (w-e)/{(w-e)^2 + w'^2} Wc(k,iw')(i,j) >
!!     <B(rk,j) psi(q-rk,n) |psi(q,t)>
!!     e = e(q-rk,n), w' is real, Wc = W-v
!!==============================================================
!! Get zwz0(omega=0, m, i, j), and zwz(i omega, m, i, j)
!! m intermediate state. zwz= \sum_I,J <i|m I> W_IJ(i omega) <J m|j>
!!
!! sum over both occupied and unoccupied states and multiply by weight
!     new from Jan2006! I think this should be OK.  ----------------------------
!     The output of sxcf_fal2 is  <i|Re[S](e_i)|j> ------------
!     Im-axis integral gives Hermitian part of S.
!     (Be careful as for the difference between
!     <i|Re[S](e_i)|j> and transpose(dconjg(<i|Re[S](e_i)|j>)).
!     ---because e_i is included.
!     The symmetrization (hermitian) procedure is inlucded in hqpe.sc.F
!     old befor Jan2006
!     &        wtt*.5d0*(   sum(zwzi(:,itp,itpp))+ !S_{ij}(e_i)
!     &        dconjg( sum(zwzi(:,itpp,itp)) )   ) !S_{ji}^*(e_j)= S_{ij}(e_j)
!-------------------------------------------------------------------------
!! omega integlation along im axis.
!! zwzi(istate,itqxx1,itqxx2)=\int_ImAxis d\omega' zwz(omega',istate,itqxx1,itqxx2) 1/(omt-omega')
!! ,where omt=omegat is given in the following 1385-1386 loop.
!!
```

```


!! -------------------------------------------------------------------
!! Contribution to SEc(qt,w) from integration along the imaginary axis
!!     loop over w' = (1-x)/x, frequencies in Wc(k,w')
!!     {x} are gaussian-integration points between (0,1)
!!--------------------------------------------------------------------
!! Readin W(omega=0) and W(i*omega)
!! Then get zwz0 and zwz
!! zwz0 = (zmel*)*(W(*omega=0)   -v)*zmel
!! zwz =  (zmel*)*(W(i*omega(ix))-v)*zmel
!TIME0_0200
            allocate( zwz0(        nstate,ntqxx,ntqxx))
            allocate( zwz(niw*npm,nstate,ntqxx,ntqxx))
            allocate( zw(nblochpmx,nblochpmx))
            ix = 1 + (0 - nw_i) !at omega=0 ! nw_i=0 (Time reversal) or nw_i =-nw
            read(ifrcw,rec=ix) zw ! direct access read Wc(0) = W(0) - v
            call matzwz2(2, zw(1:ngb,1:ngb), zmel, ntqxx, nstate,ngb,
     o       zwz0)
            do 1380 istate=1,nstate
              zwz0(istate,1:ntqxx,1:ntqxx) = ! w(iw) + w(-iw) Hermitian part.
     &         (zwz0(istate,1:ntqxx,1:ntqxx)
     &         + dconjg(transpose(zwz0(istate,1:ntqxx,1:ntqxx))))/2d0
 1380       continue
            do 1390 ix=1,niw    !niw is usually ~10 points.
              read(ifrcwi,rec=ix) zw ! direct access read Wc(i*omega)=W(i*omega)-v
              call matzwz2(2, zw(1:ngb,1:ngb), zmel, ntqxx, nstate,ngb,
     o         zwz(ix,1:nstate,1:ntqxx,1:ntqxx)) ! zwz = zmel*(W(0)-v)*zmel
              do 1395 istate=1,nstate
                zw(1:ntqxx,1:ntqxx)= zwz(ix,istate,1:ntqxx,1:ntqxx)
                zwz(ix,istate,1:ntqxx,1:ntqxx) = ! w(iw) + w(-iw)  Harmitian part
     &           ( zw(1:ntqxx,1:ntqxx)
     &           + dconjg(transpose(zw(1:ntqxx,1:ntqxx))) )/2d0
                if(npm==2) then ! w(iw) - w(-iw) Anti Hermitian part
                  zwz(ix+niw,istate,1:ntqxx,1:ntqxx) =
     &             ( zw(1:ntqxx,1:ntqxx)
     &             - dconjg(transpose(zw(1:ntqxx,1:ntqxx))) )/2d0/img
                endif
 1395         continue
 1390       continue
            deallocate(zw)
!TIME1_0200 "endofdo1390"
!! Integration along imag axis for zwz(omega) for given it,itp,itpp
!! itp  : left-hand end of expternal band index.
!! itpp : right-hand end of expternal band index.
!! it   : intermediate state of G.
!TIME0_0210
            allocate(zwzi(nstate,ntqxx,ntqxx))
            do 1400 itpp= 1,ntqxx
              do 1410 itp = 1,ntqxx
                if((jobsw==5).and.(itpp/=itp)) cycle
                if (jobsw==1.or.jobsw==4) then
                  omegat = ef
c               elseif (jobsw==2)               omegat=.5d0*(omega(itp)+omega(itpp))
                else
                  omegat = omega(itp)
                endif
                do 1420  it = 1,nstate
                  we =.5d0*( omegat -ekc(it))
                  if(it <= nctot) then
                    esmrx = 0d0
                  else
                    esmrx = esmr
                  endif
!! ua_auto may be recovered in future...
c     if(ua_auto) then
c     ratio = .5d0 *( abs(zwz(niw,it,itp,itp  )/zwz0(it,itp,itp  ))
c     &                     +abs(zwz(niw,it,itpp,itpp)/zwz0(it,itpp,itpp)) )
c     call gen_ua(ratio,niw,freqx,  expa_,ua_)
c     endif
!! Gaussian smearing. Integration along im axis. zwz(1:niw) and zwz0 are used.
                  zwzi(it,itp,itpp) =
     &             wintzsg_npm(npm, zwz(1,it,itp,itpp), zwz0(it,itp,itpp)
     &             ,freqx,wx,ua_,expa_,we,niw,esmrx)
c                    zwzi(it,itp,itpp) =  !rectangular smearing only for npm=1
c     &                          wintzav ( zwz(1,it,itp,itpp),zwz0(it,itp,itpp)
c     &                          ,freqx,wx,ua_,expa_,we,niw, esmrx)
 1420           continue
 1410         continue
 1400       continue
            deallocate(zwz0,zwz) !zwzs
            if(debug) print *,'zzzzzzzzz sum zwzi ',sum(abs(zwzi(:,:,:)))
!TIME1_0210 "endofdo1400"
!! Contribution to Sigma_{ij}(e_i)
            do  1500 itpp= 1,ntqxx
              do 1510 itp = 1,ntqxx
                if( jobsw==5.and.(itpp/=itp)) cycle
                zsec(itp,itpp,ip) = zsec(itp,itpp,ip) + wtt*sum(zwzi(:,itp,itpp))
 1510         continue
 1500       continue
            deallocate(zwzi)
            if(jobsw==4) goto 2002

!! ------------------------------------------------------------------
!!  Contribution to SEc(qt,w) from the poles of G (integral along real axis)
!!    Currently, jobsw =1,3,5 are allowed...
!!    The variable we means \omega_epsilon in Eq.(55) in PRB76,165106 (2007)
!! ------------------------------------------------------------------
!TIME0_0310
            if(timemix) call timeshow("goto Sec pole part k-cycle")
            if(debug)  write(6,*)'GOTO contribution to SEc(qt,w) from the poles of G'
            if (.not.(jobsw == 1 .or. jobsw == 3.or.jobsw==5)) then
              call rx( 'sxcf_fal3_scz: jobsw /= 1 3 5')
            endif
!! Get index nwxi nwx nt_max. finish quickly. We can simplify this...
            call get_nwx(omega,ntq,ntqxx,nt0p,nt0m,nstate,freq_r,
     i       nw_i,nw,esmr,ef,ekc,wfaccut,nctot,nband,debug,
     o       nwxi,nwx,nt_max)
!! assemble small arrays first.
            allocate(we_(nt_max,ntqxx),wfac_(nt_max,ntqxx),ixss(nt_max,ntqxx),ititpskip(nt_max,ntqxx),iirx(ntqxx))
            call weightset4intreal(nctot,esmr,omega,ekc,freq_r,nw_i,nw,
     i       ntqxx,nt0m,nt0p,ef,nwx,nwxi,nt_max,wfaccut,wtt,
     o       we_,wfac_,ixss,ititpskip,iirx)

!! We need zw3, the Hermitian part, because we need only hermitean part of Sigma_nn'
!! This can be large array; nwx-nwxi+1 \sim 400 or so...
            allocate( zw3(ngb,ngb,nwxi:nwx))
            allocate( zw(nblochpmx,nblochpmx))
            do ix = nwxi,nwx
              nrec = ix-nw_i+1  !freq_r(ix is in nw_i:nx)
              read(ifrcw,rec=nrec) zw ! direct access Wc(omega) = W(omega) - v
              if(hermitianw) then
                zw3(:,:,ix)=(zw(1:ngb,1:ngb)+transpose(dconjg(zw(1:ngb,1:ngb))))/2d0
              else
                zw3(:,:,ix)=zw(1:ngb,1:ngb)
              endif
            enddo
            deallocate(zw)
!! rearrange index of zmel
            allocate(zmel1(ngb))
            if(jobsw==3) then
              allocate(zmel1_(ntqxx,ngb,nstate))
              do itpp= 1,ntqxx
                do it  = 1,nstate
                  zmel1_(itpp,1:ngb,it) = zmel(1:ngb,it,itpp)
                enddo
              enddo
            endif
!! jobsw==3
            if( jobsw==3) then
              allocate(zwz44(3,ntqxx),zwz4(ntqxx,3))
              do itp=1,ntqxx
                do it=1,nt_max
                  if(ititpskip(it,itp)) cycle
                  we =  we_(it,itp)
                  ixs=  ixss(it,itp)
                  zmel1(:)=dconjg(zmel(:,it,itp))
                  zwz4=0d0
                  do ix0=1,3
                    ix=ixs+ix0-2
                    do igb2=1,ngb
! !                     **** most time consuming part ******
                      zz2=sum(zmel1(1:ngb)*zw3(1:ngb,igb2, iirx(itp)*ix)  )
                      call zaxpy(ntqxx,zz2,zmel1_(1,igb2,it),1,zwz4(1,ix0),1)
                    enddo
                  enddo
                  zwz44 = transpose(zwz4)
                  do itpp=1,ntqxx
                    if(npm==1) then
                      zsec(itp,itpp,ip) = zsec(itp,itpp,ip)
     .                 + wfac_(it,itp) * alagr3z2(we,freq_r(ixs-1),zwz44(1,itpp),itp==itpp ) !mar015,itp,itpp)
                    else
                      zsec(itp,itpp,ip) = zsec(itp,itpp,ip)
     .                 + wfac_(it,itp) * alagr3z(we,freq_r(ixs-1),zwz44(1,itpp))
                    endif
                  enddo
                enddo
              enddo
              deallocate(zwz44,zwz4)
            endif

!! jobsw=1,5 Sigma are calculated.
```

```
            if( jobsw==1.or.jobsw==5) then
              do itp=1,ntqxx
                do it=1,nt_max
                  if(ititpskip(it,itp)) cycle
                  we =  we_(it,itp)
                  ixs=  ixss(it,itp)
                  zmel1(:)=dconjg(zmel(:,it,itp))
                  zwz3=0d0
                  do ix0=1,3
                    ix=ixs+ix0-2
!!               **** most time consuming part for jobsw=1 ******
!!               To reduce computational time, confusing treatment only uses lower half of zw3 (zw3 is Hermitan)
!!               Clean up needed.

!! zwz3 contains <itp| it I> wz3_IJ(we)  <J it| itp>
!!    when zw3 is hermitian.
                    if(hermitianw) then
                      do igb2=2,ngb
                      zz2 = sum(zmel1(1:igb2-1)*zw3(1:igb2-1,igb2,iirx(itp)*ix)  ) +
     &                 .5d0* zmel1(igb2)*zw3(igb2,igb2,iirx(itp)*ix)
                      zwz3(ix0) = zwz3(ix0)+zz2*zmel(igb2,it,itp)
                      enddo       !igb2
                      zwz3(ix0) = 2d0*dreal(zwz3(ix0))+ !I think 2d0 is from upper half.
     &                zmel1(1)*zw3(1,1, iirx(itp)*ix)*zmel(1,it,itp)
!!    when zw3 is not need to be hermitian case. This gives life time
                    else
                      zwz3(ix0) = sum( matmul(zmel1(1:ngb), zw3(1:ngb,1:ngb,iirx(itp)*ix))*zmel(1:ngb,it,itp) )
                    endif
                  enddo
                  if(npm==1) then
                    zsec(itp,itp,ip) = zsec(itp,itp,ip)
     .               + wfac_(it,itp)*alagr3z2(we,freq_r(ixs-1),zwz3,.true.)
                  else
                    zsec(itp,itp,ip) = zsec(itp,itp,ip)
     .               + wfac_(it,itp)*alagr3z(we,freq_r(ixs-1),zwz3)
                  endif
                enddo
              enddo
            endif
!TIME1_0310 "EndReCorrelation"
c           goto 2012

cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c$$$
c$$$
c$$$
c$$$ccccccccc old code cccccccccccccccccccccccc
c$$$            if(timemix) call timeshow("55555 k-cycle")
c$$$            if(debug) write(*,'(a,5i6)')'kino: ntqxx,itini,itend,ngb=',ntqxx,itini,itend,ngb
c$$$c$$$               if(test_symmetric_W().and.npm==2) then
c$$$c$$$                  if(onceww(4)) write(6,*)' test_symmetric_W()=',test_symmetric_W(),nwxi,nwx
c$$$c$$$                  allocate(zw3x(ngb,ngb))
c$$$c$$$                  do ix= 1,min(abs(nwxi),nwx)
c$$$c$$$                     zw3x = 0.5d0* (zw3(:,:,ix) + zw3(:,:,-ix))
c$$$c$$$                     zw3(:,:, ix)=zw3x
c$$$c$$$                     zw3(:,:,-ix)=zw3x
c$$$c$$$                  enddo
c$$$c$$$                  deallocate(zw3x)
c$$$c$$$               endif
c$$$!TIME1 "before 2001"
c$$$!TIME0
c$$$            allocate(zwz44(3,ntqxx),zwz4(ntqxx,3))
c$$$            do 2001 itp = 1,ntqxx ! loop over states (q-k,n)
c$$$              omg = omega(itp)
c$$$              if (omg >= ef) then
c$$$                itini= nt0m+1
c$$$                itend= nt_max
c$$$                iii=  1
c$$$              else
c$$$                itini= 1
c$$$                itend= nt0p
c$$$                iii= -1
c$$$              endif
c$$$              do 2011 it = itini,itend ! nt0p corresponds to efp
c$$$                esmrx = esmr
c$$$                if(it<=nctot) esmrx = 0d0
c$$$                wfac = wfacx2(omg,ef, ekc(it),esmrx)
c$$$                if(wfac<wfaccut) cycle ! next it
c$$$                we = .5d0* abs( omg-weavx2(omg,ef, ekc(it),esmr) ) !Gaussian smearing
c$$$                if(it<=nctot .and.wfac>wfaccut) call rx( "sxcf: it<=nctot.and.wfac/=0")
c$$$c$$$                    Rectangular smearing
c$$$c$$$                        if( wfac==0d0) cycle ! next it
c$$$c$$$                        if( omg >= ef) we = 0.5d0* abs( max(omg-ekc(it), 0d0) )
c$$$c$$$                        if( omg <  ef) we = 0.5d0* abs( min(omg-ekc(it), 0d0) )
c$$$c$$$                        if( it<=nctot) then !faleev
```

```
c$$$c$$$                           if(wfac/=0) call rx( "sxcf:  it<=nctot.and.wfac/=0")
c$$$c$$$                        endif
c$$$c$$$                     endif
c$$$                if(debug) write(6,"( ' xxx1',10d13.6)") omg,ef, ekc(it),wfac
c$$$                wfac= iii* wfac*wtt
c$$$                do iwp = 1,nw
c$$$                  ixs=iwp
c$$$                  if(freq_r(iwp)>we) exit
c$$$                enddo
c$$$                if(nw_i==0) then
c$$$                  if(ixs+1>nwx) call rx( ' sxcf: ixs+1>nwx xxx2')
c$$$                else
c$$$                  if(omg >=ef .and. ixs+1> nwx ) then
c$$$                    write(6,*)'ixs+1 nwx=',ixs+1,nwx
c$$$                    call rx( ' sxcf: ixs+1>nwx yyy2a')
c$$$                  endif
c$$$                  if(omg < ef .and. abs(ixs+1)> abs(nwxi) ) then
c$$$                    write(6,*)'ixs+1 nwxi=',ixs+1,nwxi
c$$$                    call rx( ' sxcf: ixs-1<nwi yyy2b')
c$$$                  endif
c$$$                endif
c$$$                iir = 1
c$$$                if(omg < ef .and. nw_i/=0) iir = -1
c$$$                zmel1(:)=dconjg(zmel(:,it,itp))
c$$$
c$$$                if (jobsw == 1.or.jobsw==5) then
c$$$                  zwz3=(0d0,0d0)
c$$$!kino 2014-08-13  !$OMP parallel do private(ix,zz2)
c$$$                  do 2014 ix0=1,3
c$$$                    ix=ixs+ix0-2
c$$$                    do igb2=2,ngb
c$$$! !**** most time consuming part for jobsw=1 ******
c$$$                      zz2=sum(zmel1(1:igb2-1)*zw3(1:igb2-1,igb2,iir*ix)  ) +
c$$$     &                 .5d0* zmel1(igb2)*zw3(igb2,igb2,iir*ix)
c$$$                      zwz3(ix0)=zwz3(ix0)+zz2*zmel(igb2,it,itp)
c$$$                    enddo       !igb2
c$$$                    zwz3(ix0)=2d0*dreal(zwz3(ix0))+
c$$$     &               zmel1(1)*zw3(1,1, iir*ix)*zmel(1,it,itp)
c$$$ 2014             continue      !ix
c$$$!kino 2014-08-13  !$OMP end parallel do
c$$$                  if(npm==1) then
c$$$                    zsec(itp,itp,ip) = zsec(itp,itp,ip)
c$$$     .               + wfac*alagr3z2(we,freq_r(ixs-1),zwz3,itp,itp)
c$$$                  else
c$$$                    zsec(itp,itp,ip) = zsec(itp,itp,ip)
c$$$     .               + wfac*alagr3z(we,freq_r(ixs-1),zwz3)
c$$$                  endif
c$$$!!  this contribution to zsec_nn is real (hermitean)
c$$$
c$$$                elseif(jobsw == 3) then
c$$$                  zwz4=(0d0,0d0)
c$$$!$OMP parallel private(ix,zz2)
c$$$                  do 2015 ix0=1,3
c$$$                    ix=ixs+ix0-2
c$$$!$OMP do reduction(+:zwz4)
c$$$!! Next zaxpy is most time consuming part for jobsw=3.****
c$$$!! I think we can speed up this section...
c$$$                    do igb2=1,ngb
c$$$                      zz2=sum(zmel1(1:ngb)*zw3(1:ngb,igb2, iir*ix)  )
c$$$                      call zaxpy(ntqxx,zz2,zmel1_(1,igb2,it),1,zwz4(1,ix0),1)
c$$$                    enddo
c$$$ 2015             continue      !ix0
c$$$!$OMP end parallel
c$$$                  zwz44 = transpose(zwz4)
c$$$                  do itpp=1,ntqxx
c$$$                    if(jobsw==5.and.(itpp/=itp)) cycle
c$$$                    if(npm==1) then
c$$$                      zsec(itp,itpp,ip) = zsec(itp,itpp,ip)
c$$$     .                 + wfac*alagr3z2(we,freq_r(ixs-1),zwz44(1,itpp),itp,itpp)
c$$$                    else
c$$$                      zsec(itp,itpp,ip) = zsec(itp,itpp,ip)
c$$$     .                 + wfac*alagr3z(we,freq_r(ixs-1),zwz44(1,itpp))
c$$$                    endif
c$$$                  enddo         !itpp
c$$$                endif           ! inner jobsw=1 or 3
c$$$!!     this contribution to zsec_nn' is not hermitean because W(e_n)
c$$$!!     and must be made hermitean when zsec will be written on disc
c$$$ 2011         continue
c$$$ 2001       continue            !itp
c$$$cccccccccccccccccccccccccccccccccccccccccccccccccc

 2012       continue
            deallocate(we_,wfac_,ixss,ititpskip,iirx)
 2002       continue
            deallocate(zw3, zmel, zmel1)
            if(allocated(zmel1_)) deallocate(zmel1_)
            if(verbose()>50) call timeshow("11after alagr3z iw,itp,it cycles")
            if(debug) then
              write(6,*)' end of do 2001 '
              do itp = 1,ntq
                write(6,'(" zsec=",i3,2d15.7)') itp,zsec(itp,itp,ip)
              enddo
            endif
 1000     continue              ! end do irot
          ifvcoud =iclose('Vcoud.'//charnum5(kx))
          if(.not.exchange) then
            ifrcw  = iclose('WVR.'//charnum5(kx))
            ifrcwi = iclose('WVI.'//charnum5(kx))
          endif
          deallocate(ppovlz)
 1100   continue                ! end of kx-loop
        ifvcoud =iclose('Vcoud')
        if(irot==1) write(6,"('  sum(abs(zsec))=',d23.15)") sum(abs(zsec))
        if (allocated(vcoul))deallocate(vcoul)
 1001 continue                  ! end do ip
c      if(allocated(freq_r))deallocate(freq_r)
c      if (allocated(expikt))deallocate(expikt)
c!TIME1_0000 "end of sxcf_fal3_scz"
      end subroutine sxcf_fal3_scz


      subroutine weightset4intreal(nctot,esmr,omega,ekc,freq_r,nw_i,nw,
     i ntqxx,nt0m,nt0p,ef,nwx,nwxi,nt_max,wfaccut,wtt,
     o we_,wfac_,ixss,ititpskip,iirx)
!! generate required data set for main part of real part integration.
      implicit none
      integer,intent(in):: ntqxx,nctot,nw_i,nw,nt0m,nwx,nwxi,nt_max
      real(8),intent(in)::ef,omega(ntqxx),ekc(ntqxx),freq_r(nw_i:nw),esmr,wfaccut,wtt
      real(8),intent(out):: we_(nt_max,ntqxx),wfac_(nt_max,ntqxx)
      integer,intent(out) :: ixss(nt_max,ntqxx),iirx(ntqxx)
      logical,intent(out) :: ititpskip(nt_max,ntqxx)
      integer:: itini,iii,it,itend,wp,ixs,itp,iwp,nt0p
      real(8):: omg,esmrx,wfacx2,we,wfac,weavx2
      ititpskip=.false.
      do itp = 1,ntqxx          !this loop should finish in a second
        omg = omega(itp)
!  jobsw==2
!            if (jobsw==2)  omg=.5d0*(omega(itp)+omega(itpp))
        iirx(itp) = 1
        if( omg < ef .and. nw_i/=0) iirx(itp) = -1
        if (omg >= ef) then
          itini= nt0m+1
          itend= nt_max
          iii=  1
        else
          itini= 1
          itend= nt0p
          iii= -1
        endif
        ititpskip(:itini-1,itp)=.true.
        ititpskip(itend+1:,itp)=.true.
        do it = itini,itend     ! nt0p corresponds to efp
          esmrx = esmr
          if(it<=nctot) esmrx = 0d0
          wfac_(it,itp) = wfacx2(omg,ef, ekc(it),esmrx)
          wfac = wfac_(it,itp)
          if(wfac<wfaccut) then
            ititpskip(it,itp)=.true.
            cycle
          endif
          wfac_(it,itp)=  wfac_(it,itp)*wtt*iii
!   Gaussian smearing we_= \bar{\omega_\epsilon} in sentences next to Eq.58 in PRB76,165106 (2007)
!   wfac_ = $w$ weight (smeared thus truncated by ef). See the sentences.
          we_(it,itp) = .5d0* abs( omg-weavx2(omg,ef, ekc(it),esmr) )
          we= we_(it,itp)
          if(it<=nctot .and.wfac>wfaccut) call rx( .and."sxcf: it<=nctotwfac/=0")
          do iwp = 1,nw
            ixs = iwp
            if(freq_r(iwp)>we) exit
          enddo
          ixss(it,itp) = ixs
          if(nw_i==0) then
            if(ixs+1>nwx) call rx( ' sxcf: ixs+1>nwx xxx2')
          else
            if(omg >=ef .and. ixs+1> nwx ) then
              write(6,*)'ixs+1 nwx=',ixs+1,nwx
              call rx( ' sxcf: ixs+1>nwx yyy2a')
            endif
            if(omg < ef .and. abs(ixs+1)> abs(nwxi) ) then
              write(6,*)'ixs+1 nwxi=',ixs+1,nwxi
              call rx( ' sxcf: ixs-1<nwi yyy2b')
            endif
          endif
        enddo
```

```fortran
      enddo
      end subroutine weightset4intreal
      end module m_sxcfsc
!! ---------------------------------------------------------------
      subroutine get_nwx(omega,ntq,ntqxx,nt0p,nt0m,nstate,freq_r,
     i nw_i,nw,esmr,ef,ekc,wfaccut,nctot,nband,debug,
     o nwxi,nwx,nt_max)
!> Determine indexes of a range for calculation.
!! It is better to clean this up...
      implicit none
      integer,intent(in) :: nctot,nw_i,nw,nstate,nt0p,nt0m,ntq,
     & nband,ntqxx
      real(8),intent(in):: omega(ntq),esmr,ef,ekc(nctot+nband),wfaccut,
     & freq_r(nw_i:nw)
      integer,intent(out) :: nt_max,nwxi,nwx

      integer:: itp,it,itini,itend,iwp,ixs,ixsmin,ixsmx,verbose
      real(8):: omg,wfac,wfacx2,we,weavx2,esmrx,wexx
      logical::debug
!!     maximum ixs reqired.
      ixsmx =0
      ixsmin=0
      do 301 itp = 1,ntqxx
        omg  = omega(itp)
        if (omg < ef) then
          itini= 1
          itend= nt0p
        else
          itini= nt0m+1
          itend= nstate
        endif
        do 311 it=itini,itend
          esmrx = esmr
          if(it<=nctot) esmrx = 0d0
          wfac = wfacx2(omg,ef, ekc(it),esmrx)
          if(wfac<wfaccut) cycle !Gaussian case
          we = .5d0*(weavx2(omg,ef,ekc(it),esmr)-omg)
cc Gaussian=F case  keep here just as a memo
c           if(wfac==0d0) cycle ! next it
c           if(omg>=ef) we = max( .5d0*(omg-ekc(it)), 0d0) ! positive
c           if(omg< ef) we = min( .5d0*(omg-ekc(it)), 0d0) ! negative
          if(it<=nctot) then
            if(wfac>wfaccut) call rx( .and."sxcf: it<=nctotwfac/=0")
          endif
          do iwp = 1,nw
            ixs=iwp
            if(freq_r(iwp)>abs(we)) exit
          enddo
c     This change is because G(omega-omg') W(omg') !may2006
c     if(ixs>ixsmx  .and. omg<=ef ) ixsmx  = ixs
c     if(ixs>ixsmin .and. omg> ef ) ixsmin = ixs
          if(ixs>ixsmx  .and. omg>=ef ) ixsmx  = ixs
          if(ixs>ixsmin .and. omg< ef ) ixsmin = ixs
          wexx  = we
          if(ixs+1 > nw) then
            write (*,*) ' nw_i ixsmin',nw_i, ixsmin
            write (*,*) ' wexx ',wexx
            write (*,*) ' omg ekc(it) ef ', omg,ekc(it),ef
            call rx( ' sxcf 222: |w-e| out of range')
          endif
 311    continue
 301  continue                  !end of SEc w and qt -loop
!!
      if(nw_i==0) then          !time reversal
        nwxi = 0
        nwx  = max(ixsmx+1,ixsmin+1)
      else                      !no time revarsal
        nwxi = -ixsmin-1
        nwx  =  ixsmx+1
      endif
      if (nwx > nw   ) then
        call rx( ' sxcf_fal3_sc nwx check : |w-e| > max(w)')
      endif
      if (nwxi < nw_i) then
        call rx( ' sxcf_fal3_sc nwxi check: |w-e| > max(w)')
      endif
      if(debug) write(6,*)'nw, nwx=',nw,nwx
      if(verbose()>50)call timeshow("10before alagr3z iw,itp,it ")
!!  Find nt_max
      nt_max=nt0p               !initial nt_max
      do 401 itp = 1,ntqxx
        omg     = omega(itp)
        if (omg > ef) then
          do  it = nt0m+1,nstate ! nt0m corresponds to efm
            wfac = wfacx2(ef,omg, ekc(it),esmr)
c                       if( (GaussSmear().and.wfac>wfaccut)
c     &                       .or.(.not.GaussSmear().and.wfac/=0d0)) then
```

```
           if(wfac>wfaccut) then
             if (it > nt_max) nt_max=it ! nt_max is  unocc. state
           endif               ! that ekc(it>nt_max)-omega > 0
         enddo                 ! so it > nt_max does not contribute to omega pole integral
       endif
401  continue                  !end of  w and qt -loop
     end subroutine get_nwx
```

4.25 gwsrc/x0kf_v4h.F File Reference

Functions/Subroutines

- subroutine x0kf_v4hz (npm, ncc,ihw, nhw, jhw, whw, nhwtot,n1b, n2b, nbnbx, nbnb,q,nsp, isp_k, isp_kq,qbas, ginv, rk, wk,
- subroutine x0kf_v4hz_symmetrize (npm,
- subroutine dpsion5 (frhis, nwhis, freqr, nw_w, freqi, niwt,realomega,imagomega,rcxq, npm, nw_i, nmbas1, nmbas2,zxq, zxqi,
- logical function checkbelong (qin, qall, nq, ieibz)
- subroutine hilbertmat (zz, nwhis, his_L, his_C, his_R, rmat)
- real(8) function wcutef (e, ecut, ecuts)

4.25.1 Function/Subroutine Documentation

4.25.1.1 logical function checkbelong (real(8), dimension(3) *qin,* real(8), dimension(3,nq) *qall,* integer *nq,* integer *ieibz*)

Definition at line 1567 of file x0kf_v4h.F.

4.25.1.2 subroutine dpsion5 (real(8), dimension(nwhis+1) *frhis,* integer(4) *nwhis,* real(8), dimension(0:nw_w) *freqr,* integer(4) *nw_w,* real(8), dimension(niwt) *freqi,* integer(4) *niwt,* logical *realomega,* logical *imagomega,* complex(8), dimension(nmbas1,nmbas2, nwhis,npm) *rcxq,* integer(4) *npm,* integer(4) *nw_i,* integer(4) *nmbas1,* integer(4) *nmbas2,* complex(8), dimension (nmbas1,nmbas2, nw_i: nw_w) *zxq,* *zxqi*)

Definition at line 1264 of file x0kf_v4h.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.25.1.3 subroutine hilbertmat (complex(8) *zz,* integer(4) *nwhis,* real(8), dimension(-nwhis:nwhis) *his_L,* real(8), dimension(-nwhis:nwhis) *his_C,* real(8), dimension(-nwhis:nwhis) *his_R,* complex(8), dimension(-nwhis:nwhis) *rmat*)

Definition at line 1582 of file x0kf_v4h.F.

Here is the caller graph for this function:

4.25.1.4 real(8) function wcutef (real(8) *e*, real(8) *ecut*, real(8) *ecuts*)

Definition at line 1683 of file x0kf_v4h.F.

Here is the caller graph for this function:

4.25.1.5 subroutine x0kf_v4hz (integer(4) *npm*, integer(4) *ncc*, integer(4), dimension(nbnbx,nqbz,npm) *ihw*, integer(4), dimension(nbnbx,nqbz,npm) *nhw*, integer(4), dimension(nbnbx,nqbz,npm) *jhw*, real(8), dimension(nhwtot) *whw*, integer(4) *nhwtot*, integer(4), dimension(nbnbx,nqbz,npm) *n1b*, integer(4), dimension(nbnbx,nqbz,npm) *n2b*, integer(4) *nbnbx*, integer(4), dimension(nqbz,npm) *nbnb*, real(8), dimension(3) *q*, integer(4) *nsp*, integer(4) *isp_k*, integer(4) *isp_kq*, real(8), dimension(3,3) *qbas*, real(8), dimension(3,3) *ginv*, real(8), dimension(3,nqbz) *rk*, real(8), dimension(nqbz) *wk*)

Definition at line 1 of file x0kf_v4h.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.25.1.6 subroutine x0kf_v4hz_symmetrize (integer(4) *npm*)

Definition at line 866 of file x0kf_v4h.F.

Here is the caller graph for this function:

4.26 x0kf_v4h.F

```
00001       subroutine x0kf_v4hz (npm,ncc,
00002      i            ihw,nhw,jhw,whw,nhwtot, ! tetwt5
00003      i            n1b,n2b,nbnbx,nbnb,     !  use whw by tetwt5 ,
00004      i                  q,
00005      i                  nsp,isp_k,isp_kq,!symmetrize,
00006      i                  qbas,ginv,rk,wk,
00007 c     i                  mdim,
00008      d                  nlmto,nqbz,nctot,
00009 c     d                  natom,
00010      d                  nbloch,nwt,
00011      i      iq, ngbb, ngc, ngpmx,ngcmx,
00012      i      nqbze, nband,nqibz,
00013      o      rcxq,
00014      i     nolfco,zzr,nmbas, zcousq,
00015      i     chipmzzr,eibzmode,
00016      i     nwgt,igx,igxt,ngrp,eibzsym, crpa)
00017       use m_readqg,only   :  readqg
00018       use m_readeigen,only:  readeval
00019       use m_keyvalue,only   :  getkeyvalue
00020       use m_rotmpb,only   :  rotmpb2
00021       use m_readqgcou,only:
00022      o qtt_, nqnum
00023       use m_pkm4crpa,only : readpkm4crpa
00024       use m_zmel,only     : get_zmelt2,
00025      o zmel !,ppbir ,ppovlz
00026
00027 !! === calculate chi0, or chi0_pm ===
00028 !! We calculate imaginary part of chi0 along real axis.
```

```
!!
!! NOTE: rcxq is i/o variable for accumulation. We use E_mu basis when chipm=F.
!!
!!
!! ppovl= <I|J> = O , V_IJ=<I|v|J>
!! (V_IJ - vcoud_mu O_IJ) Zcousq(J, mu)=0, where Z is normalized with O_IJ.
!! <I|v|J>= \sum_mu ppovl*zcousq(:,mu) v^mu (Zcousq^*(:,mu) ppovl)
!!
!! zmelt contains O^-1=<I|J>^-1 factor. Thus zmelt(phi phi J)= <phi |phi I> O^-1_IJ
!! ppovlz(I, mu) = \sum_J O_IJ Zcousq(J, mu)
!!
!! when nmbas1=2, this works in a special manner for nolfco=T chipm=F. mar2012takao
!!
!!
!! OUTPUT:
!!  rcxq (nmbas,nmbas,nwt,npm): for given q,
!!       rcxq(I,J,iw,ipm) =
!!       Im (chi0(omega))= \sum_k <I_q psi_k|psi_(q+k)> <psi_(q+k)|psi_k> \delta(\omega- (e_i-ej))
!!      When npm=2 we calculate negative energy part. (time-reversal asymmetry)
!!
c              See also tetwt5. and check weight mode=4 of hx0fp0 and (mode=5,6).
c
c- takao kotani Apr 2002   This originated from Ferdi’s x0k.
cr daxpy dominates the cpu time
c
c
c x0(i,j)(q,w) = S[k=FBZ] S[t=occ] S[t’=unocc]
c  <M(q,i) psi(k,t) |psi(k+q,t’)> <psi(k+q,t’)| psi(k,t) M(q,j)>
c  { 1/[w-e(k+q,t’)+e(k,t)+i*delta] - 1/[w+e(k+q,t’)-e(k,t)-i*delta] }
c  ; w is real. x0 is stored into rcxq.
c
c zzmel =  <psi(k+q,t’) | psi(k,t) B(R,i)>
c zmel  =  <psi(k+q,t’) | psi(k,t) M(R,i)>
c rcxq  =  zeroth order response function along the positive real axis.
c          Note this is accmulating variable. Equivalnet with zxq. See rcxq2zxq below.
c
c q       = q-vector in x(q,iw)
c ifchi   = direct access unit file for cphi, the coefficient of eigenfunction for argumentation wave.
c qbas    = base reciprocal lattice vectors
c ginv    = inverse of qbas s. indxrk.f
c
c ppb     = <phi(RLn) phi(RL’n’) B(R,i)>
c
c iclass  = given an atom, tells the class
c iindxk  = index for k-points in the FBZ
c rk      = k-points in the 1st BZ
c wk      = weight for each k-point in the 1st BZ
c freq    = frequency points along positive imaginary axis
c
c
c mdim    = dimension of B(R,i) for each atom R
c nlnmx   = maximum number of l,n,m
c nlmto   = total number of LMTO basis functions
c nqbz    = number of k-points in the 1st BZ
c n1,n2,n3= divisions along base reciprocal lattice vectors
c natom   = number of atoms
c noccx   = maximum number of occupied states
c noccxv  = maximum number of occupied valence states
c nbloch  = total number of Bloch basis functions
c
c cphi_k cphi_kq:  b(k) and b(k+q)
c  : coefficients of eigenfunctions for argumentation waves in each MT
c
      implicit none
      integer(4):: npm,ncc,ngbb,natom,nwt,nsp,isp_k,isp_kq,nlmto !,noccx,noccxv
     &  ,nl,nclass,nnc,nlnmx,nbloch,iq,nqibz,iatom,nctot,nbmx,iopen !mdimx,
     &  ,jpm,ibib,itps,nt0,ntp0,ngp_kq,ngp_k,it,itp,iw,igb2,igb1,ngb
     &  ,nn,no,isx,iclose,k,nbnbx,nqbz
      real(8):: q(3),qbas(3,3),ginv(3,3),rk(3,nqbz),wk(nqbz),ebmx
c      complex   (8):: rcxq (ngbb,ngbb, nwt,npm),aaa
      complex   (8):: rcxq (nmbas,nmbas,nwt,npm)
      complex(8) :: imag=(0d0,1d0),trc,aaa !phase(natom),
      complex(8),allocatable:: cphi_k(:,:),cphi_kq(:,:),geig_kq(:,:),geig_k(:,:)
      integer(4):: ngpmx, ngcmx, nqbze, nband,
     &              ngc,nadd(3), !ngvecpB(3,ngpmx,nqbze), ngpn(nqbze),
     &             igc !ngveccB(3,ngcmx),
c     &              ngvecp_kq(3,ngpmx),ngvecp_k(3,ngpmx)
      complex(8),allocatable :: zmelt(:,:,:)!,zmelt1(:,:,:)
      real(8) :: qbasinv(3,3), det,qdiff(3),add(3),symope(3,3)
     &    ,shtv(3)=(/0d0,0d0,0d0/)
      data symope /1d0,0d0,0d0, 0d0,1d0,0d0, 0d0,0d0,1d0/
c      real(8) :: ppb_unused(*)

c      integer(4) :: mdim(natom)

      complex(8),allocatable :: ttx(:,:)
      complex(8),allocatable::  z1p(:,:)
      integer(4) ::  nbnb(nqbz,npm),
     &  n1b(nbnbx,nqbz,npm), n2b(nbnbx,nqbz,npm)
      complex(8),allocatable:: zzmel(:,:,:)
c      integer(4)::imdim(natom),iatomp(natom)
      logical,parameter:: debug=.false.
c---tetwt5
      logical:: hist      ,ipr
      integer(4):: nhwtot,
     &  ihw(nbnbx,nqbz,npm),nhw(nbnbx,nqbz,npm),jhw(nbnbx,nqbz,npm)
      real(8):: whw(nhwtot)
      complex(8) :: zmelt1,zmelt2,zmeltt(ngbb)      !...........................sf 21May02
c         complex(8), allocatable :: zxq_(:,:,:) !..........sf 21May02
      real(8) :: imagweight !............................sf 21May02
c         logical :: takao=.false. !.........................sf 21May02
c         allocate( zxq_( nbloch + ngc,nbloch + ngc,nwt)) !..sf 21May02
      integer(4)::nocc
      real(8):: eband(nband)!,ebandr(nband),ebandqr(nband)
c      integer(4):: n_index_qbz
c      integer(4):: index_qbz(n_index_qbz,n_index_qbz,n_index_qbz)
c-----
c      integer(4):: nlnm(*),nlnmv(*),nlnmc(*)!,iclass(*)!,icore(*),ncore(*)
      integer(4):: verbose
c---for iepsmode
      logical   :: nolfco !iepsmode
      integer(4):: nmbas, imb1,imb2, imb !nmbas1x !nmbas2,nmbas1,
      complex(8):: zq01,zq02
c      real(8)   :: zq0zq0
      complex(8)   :: zq0zq0 !This is a bug for the case of two atoms per cell!!! oct2006
c      complex(8):: rcxqmean(nwt,npm,nmbas1,nmbas2)

      real(8):: vec_kq_g(3),vec_k_g(3),vec_kq(3),vec_k(3),quu(3),tolq=1d-8,quu1(3),quu2(3) !tolqu=1d-4,

      integer(4):: nbcut,nbcut2
      logical :: iww1=.true.,iww2=.true.

      logical:: smbasis
      integer(4):: ifpomat, nbloch_r, ngbo,iqxdummy
c      integer(4),allocatable:: io(:),in(:),io_q(:),in_q(:)
      complex(8),allocatable:: pomat(:,:), zmeltn(:,:,:)
      real(8):: q_r(3)
      complex(8):: img=(0d0,1d0),zzz(ngbb)

      integer(4):: nkmin,  nkmax, nkqmin, nkqmax,nkmax1,nkqmax1
c      real(8):: qq(3)
      integer(4):: ib1, ib2,      ngcx,ix,iy

c      integer(4),allocatable:: ngvecc(:,:)
      logical :: onceww !testtr,negative_testtr

!! takao apr2012
      logical :: z1offd !, z1stcol
      complex(8),target :: zzr(ngbb,nmbas) !ppovlz(ngbb,ngbb),
      integer:: igb
c      logical:: symmetrize

!! jun2012takao
c      real(8):: qeibz(3,nqbz),  !    aik(3,3,ngrpt)
      integer:: ngrp,nwgt(nqbz) !,ngrpt, aiktimereversal(ngrpt),nwgtieibz,ieibz
      integer:: igx(ngrp*2,nqbz),igxt(ngrp*2,nqbz),ieqbz
!! nwgt(neibz
      logical:: checkbelong,eibzmode, chipmzzr
      complex(8):: zcousq(ngbb,ngbb)  !ppovl(ngc,ngc) ,
      complex(8),allocatable:: zcousqr(:,:,:),rcxq0(:,:),rcxq00(:,:),rcxq000(:,:),rcxqwww(:,:)
c      complex(8):: zcousqrsum(ngbb,ngbb,2),    zmeltx(ngbb),zmelty(ngbb),zcousqrx(ngbb,ngbb)
      complex(8):: zmeltx(ngbb),zmelty(ngbb),zcousqrx(ngbb,ngbb) ,zcousqc(ngbb,ngbb)
     & ,rzc(ngbb,ngbb),cmat(ngbb,ngbb) !zcousqinv(ngbb,ngbb),
      integer::  eibzsym(ngrp,-1:1),neibz,icc,ig,eibzmoden,ikp,i,j,itimer,icount,iele
      integer:: irotm,nrotmx,ixx,iyy,itt,ntimer, nccc, nxx,iagain,irotm1,irotm2
      integer,allocatable:: i1(:,:),i2(:,:),nrotm(:)
      complex(8),allocatable:: zrotm(:,:),zrr(:,:),zrrc(:,:),zrr_(:,:,:),zrrc_(:,:,:),zmmm(:),zrrx(:,:)
c      complex(8),pointer:: zmat(:,:)
c      complex(8),allocatable,target:: ppovl_(:,:)
c#ifdef USE_GEMM_FOR_SUM
c      complex(8),allocatable :: zmelt_tmp(:,:,:)
c#endif
      complex(8),allocatable:: rcxq_core(:,:)
!$    integer:: omp_get_num_threads
      logical:: eibz4x0
      logical :: crpa
      real(8):: wpw_k,wpw_kq
      real(8):: vec_kcrpa(3),vec_kqcrpa(3)

      logical :: exchange=.false.
      integer:: irot=1
      integer:: ntqxx,nbmax


c      if (symmetrize) goto 5000

c ------------------------------------------------
!TIME0_1001
      write(6,'(" x0kf_v4hz: q=",3f8.4,$)')q
      call cputid(0)
c$$$!! check eibzmode assumes nmbas1=nmbs2
c$$$      if(eibzmode) then
c$$$         if(nmbas1/=nmbas2) then
c$$$            write(6,*)'x0kf_v4h: eibzmode=T only allow nmbas1=nmbas2.',nmbas1,nmbas2,nmbas
c$$$            stop 'x0kf_v4h: eibzmode=T only allow nmbas1=nmbas2.'
c$$$         endif
c$$$      endif
!!
c$$$      imdim(1) = 1
c$$$      do iatom = 1,natom
c$$$        iatomp(iatom) = iatom
c$$$        if(iatom<natom) imdim(iatom+1)=imdim(iatom)+mdim(iatom)
c$$$      enddo
c      nctot      = noccx - noccxv
c      print *, 'qqqqqqqqqqq',qbas
      call minv33(qbas,qbasinv)
c      phase= (1d0,0d0) !coskt = 1d0; sinkt = 0d0
      allocate(cphi_k(nlmto,nband),cphi_kq(nlmto,nband), geig_kq(ngpmx,nband),  geig_k(ngpmx,nband) )
      call getkeyvalue("GWinput","nbcutlow",nbcut, default=0 )
      call getkeyvalue("GWinput","nbcutlowto",nbcut2, default=0 )
c      call getnemx(nbmx,ebmx,7,.true.)
c$$$!TIME0
c$$$      if(smbasis()) then !need to check again, when we will make smbasis on.
c$$$cccccccccccccccccccccccccccccccc
c$$$        if(ncc/=0) then
c$$$          write(6,*)"Timereversal=F(ncc/=0) not yet implemented for smbasis."
c$$$          write(6,*)" pomat should be generated correctly                  ."
c$$$Cstop2rx 2013.08.09 kino          stop "Timereversal=F(ncc/=0) not yet implemented for smbasis."
c$$$          call rx( "Timereversal=F(ncc/=0) not yet implemented for smbasis.")
c$$$        endif
c$$$cccccccccccccccccccccccccccccccccc
c$$$
c$$$        ifpomat = iopen('POmat',0,-1,0) !oct2005
c$$$C... smoothed mixed basis !oct2005
c$$$C This replace original zmelt with new zmelt based on smoothed mixed basis.
c$$$        do
c$$$          read(ifpomat) q_r,nn,no,iqxdummy !readin reduction matrix pomat
c$$$c          write(6,"('ttt: q  =',3f12.5)") q
c$$$c          write(6,"('ttt: q_r=',3f12.5)") q_r
c$$$          allocate( pomat(nn,no) )
c$$$          read(ifpomat) pomat
c$$$          if( sum(abs(q-q_r))<1d-10) then ! .and.kx <= nqibz ) then
c$$$            write(6,*) 'ok find the section for give qibz_k'
c$$$            exit
c$$$!          elseif (kx >nqibz ) then
c$$$!            exit
c$$$          endif
c$$$          deallocate(pomat)
c$$$        enddo
c$$$        if( sum(abs(q-q_r))>1d-10 ) then
c$$$          write(6,"('q  =',3f12.5)") q
c$$$          write(6,"('q_r=',3f12.5)") q_r
c$$$Cstop2rx 2013.08.09 kino          stop 'POmat reading err q/=q_r'
c$$$          call rx( 'POmat reading err q/=q_r')
c$$$        endif
c$$$        isx = iclose('POmat')
c$$$      endif
c$$$!TIME1 "after if smbasis"

ckino
!KINO      it=0
!KINO      do k=1,nqbz
!KINO        if(eibzmode.and.nwgt(k)==0 ) cycle
!KINO        if(sum(nbnb(k,1:npm))==0) cycle
!KINO        it=it+1
!KINO      enddo
!KINO      write(6,'(a,i3,1x,a,i4)')'iq=',iq,'active-k-points=',it
ckinoend

!TIME1_1001 "beforedo1000"
!! loop over k-points --------------------------------------------------------------------------
c      qq=0d0
      do 1000 k = 1,nqbz
        if(eibzmode.and.nwgt(k)==0 ) cycle
        if(debug) write(6,'("do 1000  k=",i4,3f10.4)')k,rk(:,k)
        ipr=(k<5.or.k==nqbz.or.debug)
        if(sum(nbnb(k,1:npm))==0) cycle
!TIME0_1101
        if(k<=5.or. (mod(k,max(10,nqbz/20))==1.or.k>nqbz-10) ) then
          write(6,"('  x0kf_v4hz: k rk=',i7,3f10.4,$)")k, rk(:,k)
          call cputid(0)
        endif
ccccccccccc
c          write(6,"('xxxxxxx  x0kf_v4hz: k rk=',i7,3f10.4,$)")k, rk(:,k)
ccccccccccc
!! --- tetra ------ override nt0 itps ntp0 ---------------
        nkmin = 999999
        nkmax= -999999
        nkqmin= 999999
        nkqmax=-999999
        do jpm=1,npm !npm
          do ibib = 1, nbnb(k,jpm)
            nkmin  = min(n1b(ibib,k,jpm),nkmin)
            nkqmin = min(n2b(ibib,k,jpm),nkqmin)
            if(n1b(ibib,k,jpm)<=nband)   nkmax = max(n1b(ibib,k,jpm),nkmax)
            if(n2b(ibib,k,jpm)<=nband)  nkqmax = max(n2b(ibib,k,jpm),nkqmax)
          enddo
        enddo
!! ebmx band cutoff is
c$$$!!
c$$$        nkmax1=nkmax
c$$$        nkqmax1=nkqmax
c$$$cccccccccccccccccccccccccccccc
c$$$c        ebandqr=eband
c$$$c        call readeval(rk(:,k),isp_kq,ebandr)
c$$$cccccccccccccccccccccccccccccc
c$$$        call readeval(q+rk(:,k),isp_kq,eband)
c$$$        nkqmax = nocc (eband,ebmx,.true.,nband)
c$$$        if(npm==2) then
c$$$          call readeval(rk(:,k),isp_k,eband)
c$$$          nkmax = nocc (eband,ebmx,.true.,nband)
c$$$        endif
c$$$        write(6,"('nnnnnnk  ',2i5,2i5)")nkmax1, nkmax
c$$$        write(6,"('nnnnnnkq ',2i5,2i5)")nkqmax1,nkqmax
c$$$!!
        itps  = nkqmin          ! nkqmin = the num of min   n2 =unocc for jpm=1
        nt0   = nkmax
        ntp0  = nkqmax - nkqmin +1
        if( npm==2.and. nkqmin/=1) then
          write(6,*)' npm==2 nkqmin nkqmax  nkmin nkmax=',nkqmin,nkqmax,nkmin,nkmax
          call rx( " When npm==2, nkqmin==1 should be.")
        endif
        if(nkmin/=1) then
          call rx( " nkmin==1 should be.")
        endif

!... feb2006
!  zzmel(1:nbloch, ib_k,ib_kq)
!      ib_k =[1:nctot]              core
!      ib_k =[nctot+1:nctot+nkmax]  valence
!      ib_kq =[1:nctot]             core
!      ib_kq =[ncc+1:ncc+nkqmax - nkqmin] valence range [nkqmin,nkqmax]
!   If jpm=1, ncc=0.
!   If jpm=2, ncc=ncore. itps=1 should be.
! There is a little confusion. n1b index contains cores are after valence.
! You can see codes to treat the confusion.
! NOTE:
!  q+rk n2b vec_kq  vec_kq_g geig_kq cphi_kq  ngp_kq ngvecp_kq  isp_kq
!    rk n1b vec_k   vec_k_g  geig_k  cphi_k   ngp_k  ngvecp_k   isp_k

c        if(ipr) then
c          write(6,"(' nkRange  nkqRange=',2i6,2x,2i6)") nkmin,nkmax,nkqmin,nkqmax
c        endif



c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$        goto 8828
c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$
c$$$
c$$$
c$$$
c$$$
c$$$!TIME1 "before readphi"
c$$$!TIME0
c$$$!!--- calculate the matrix elements <psi(k+q,t') | psi(k,t) B(R,i)>
c$$$!! Read cphi part of eigenfunctions for k and k+q
c$$$        call  readcphi(q+rk(:,k)-qq, nlmto,isp_kq, quu2,cphi_kq)
c$$$        call  readcphi(  rk(:,k)-qq, nlmto,isp_k, quu1,cphi_k) !quu is used q for eigenfunctions.
c$$$!! ... core
c$$$        if(debug)  call cputid(0)
c$$$!!          allocate( zzmel(nbloch,noccx, ntp0) )
c$$$!!                          q        k    q+k
c$$$        if(debug)  write(6,*)nbloch,nctot,nt0,ncc,ntp0
c$$$        allocate( zzmel(nbloch, nctot+nt0, ncc+ntp0) )
c$$$        if(debug)  write(6,"('zzzw nkmin nkqmin=',2i5)") nkmin,nkqmin
c$$$        if(onceww(5) ) write(6,*)' nctot ncc=',nctot,ncc
```

```
c$$$c      allocate( ppb(nlnmx*nlnmx*mdimx*nclass))
c$$$c      ppb=ppbir(:,irot,ispq)
c$$$        call psicb_v3  ( nctot,ncc,nt0,ntp0,iclass,phase,
c$$$     i              cphi_k (1,nkmin),
c$$$     i              cphi_kq(1,nkqmin),
c$$$     i              ppbir(:,irot,isp_k),!ppb,
c$$$     i              nlnmv,nlnmc,mdim,
c$$$     i              imdim,iatomp,
c$$$     i              mdimx,nlmto,nbloch,nlnmx,natom,nclass,
c$$$     i              icore,ncore,nl,nnc,
c$$$     o              zzmel)
c$$$C ... valence
c$$$      if(debug) write(6,'("4 zzzqqbbb222 ",3d13.5)') sum(abs(zzmel)),sum(zzmel)
c$$$        !call cputid(0)
c$$$        call psi2b_v3  ( nctot,ncc,nt0,ntp0,iclass,phase,
c$$$     i              cphi_k(1,nkmin),
c$$$     i              cphi_kq(1,nkqmin),
c$$$     i              ppbir(:,irot,isp_k),! ppb,
c$$$     i              nlnmv, nlnmc,mdim,
c$$$     i              imdim,iatomp,
c$$$     d              mdimx,nlmto,nbloch,nlnmx,natom,nclass,
c$$$     o              zzmel)
c$$$        if(debug) call cputid(0)
c$$$        if(debug) write(6,'("4 zzzqqbbb222 ",3d13.5)') sum(abs(zzmel)),sum(zzmel)
c$$$!TIME1 "after psicb_v3"
c$$$!TIME0
c$$$!! --- IPW set
c$$$        call readqg('QGpsi',q+rk(:,k)-qq,ginv, vec_kq, ngp_kq, ngvecp_kq)
c$$$        call readqg('QGpsi',  rk(:,k)-qq,ginv, vec_k,  ngp_k,  ngvecp_k)
c$$$!!      ngp_kq = ngpn(kp)  ! q+k   ntp0 in FBZ
c$$$!!      ngp_k = ngpn(k)   ! k     np0  in FBZ
c$$$!!      ngc              ! q          in IBZ
c$$$        ngb  = nbloch + ngc ! This is not ngbb for smbasis()=T. oct2005
c$$$        if(ngb/=ngbb) then
c$$$           write(6,*)' x0kf_v4h: ngb ngbb=',ngb,ngbb
c$$$           call rx( 'x0kf_v4h: ngb/=ngbb')
c$$$        endif
c$$$!!                        q       k        q+k
c$$$        allocate( zmel(ngb,nctot+nt0,ncc+ntp0) )
c$$$        allocate( z1p(ngb,ngb) )
c$$$!! ... read plane wave part of eigenfunction. (note isp is opposite).
c$$$        call readgeig(q+rk(:,k)-qq, ngpmx,isp_kq, vec_kq_g, geig_kq)
c$$$        call readgeig(  rk(:,k)-qq, ngpmx,isp_k,  vec_k_g,  geig_k)
c$$$        if(sum(abs(vec_kq_g-vec_kq))>tolqu) then
c$$$          write(6,"('vec_kq_g :',3d23.10)") vec_kq_g
c$$$          write(6,"('vec_kq  :',3d23.10)") vec_kq
c$$$          call rx( 'x0kf_v4hz:vec_kq_g/=vec_kq')
c$$$        endif
c$$$        if(sum(abs(vec_k_g-vec_k))>tolqu) then
c$$$          write(6,"('vec_k_g :',3d23.10)") vec_k_g
c$$$          write(6,"('vec_k:  ',3d23.10)") vec_k
c$$$          call rx( 'x0kf_v4hz:vec_k_g/=vec_k')
c$$$        endif
c$$$!!      qdiff = q   - qbkp(:) + rk(:,k)
c$$$        qdiff = q    - vec_kq    + vec_k
c$$$        ! q   - (q+k)   + k  is not zero.
c$$$        ! qc  -  q1     + q2
c$$$        add   = matmul(qbasinv, qdiff)
c$$$        nadd  = idint( add + dsign(.5d0,add))  !  write(6,*)' qdif=',qdiff,qbkp(:),rk(:,k)
c$$$        if(sum(abs(add-nadd))>1d-10) call rx( "sexc: abs(add-nadd))>1d-10")
c$$$        zmel = 0d0
c$$$!TIME1 "before melpln2t"
c$$$!TIME0
c$$$!! <Bq Pq2|Pq1> = < Bq Pqu2|  Pqu1> *exp(i2pi nadd)
c$$$        if(ngc/=0) then !Aug2005
c$$$          call melpln2t(ngp_kq, ngvecp_kq  ! q+k  ; kp ngp_kq 1:ntp0 q-point
c$$$     &             , ngp_k, ngvecp_k       ! k    ; k  ngp_k 1:nt0  occupied
c$$$     &           , ngc,  nadd,
c$$$     &       geig_kq(1:ngp_kq, itps:itps+ntp0-1), ntp0, ! q+k  ; kq
c$$$     &       geig_k(1:ngp_k, 1:nt0     ),  nt0, !    ; k
c$$$     i       shtv,  q, q,  symope,qbas,
c$$$     i       vec_kq, !qt oct2013
c$$$     o       zmel (nbloch+1:nbloch+ngc, nctot+1:nctot+nt0,ncc+1:ncc+ntp0))
c$$$        endif
c$$$!! == zmelt contain O^-1=<I|J>^-1 factor. zzmel(J,it,itp)= \sum_I <phi phi|I> O^-1_IJ ==
c$$$        zmel(1:nbloch, 1:nctot+nt0, 1:ncc+ntp0) =
c$$$     &  zzmel(1:nbloch, 1:nctot+nt0, 1:ncc+ntp0)
c$$$!!                        k            q+k
c$$$        deallocate(zzmel)
c$$$         if(debug) write(6,'("4 zzzppp222bbb ",3d13.5)') sum(abs(zmel)),sum(zmel)
c$$$        if(debug) call cputid(0)
c$$$!TIME1 "after melpln2t"
c$$$!TIME0
c$$$!! === zmelt conversion on different basis.
c$$$        allocate(zmmm(nmbas))   ! this is also obsolete if USE_GEMM_FOR_SUM
c$$$        if(chipmzzr) then       !spin moment basis.
c$$$          zmat => zzr
```

```
c$$$        elseif(nolfco .and. nmbas==1) then !for <e^iqr|x0|e^iqr>
c$$$          zmat => ppovlz
c$$$        else                    !may2013  this removes O^-1 factor from zmelt
c$$$          allocate(ppovl_(ngb,ngb))
c$$$          ppovl_=0d0
c$$$          do i=1,nbloch
c$$$            ppovl_(i,i)=1d0
c$$$          enddo
c$$$          ppovl_(nbloch+1:nbloch+ngc,nbloch+1:nbloch+ngc)=ppovl
c$$$          if(.not.eibz4x0()) then !sep2014 added for eibz4x0=F
c$$$            ppovl_=matmul(ppovl_,zcousq)
c$$$          endif
c$$$          zmat => ppovl_
c$$$        endif
c$$$!! :: zmelt conversion muplitpled by zzr.
c$$$        if(verbose()>39) write(6,*)'info: USE GEMM FOR SUM (zmelt = zmelt*zmat) in x0kf_v4h.F'
c$$$        allocate( zmelt_tmp(ngb,nctot+nt0,ncc+ntp0) )
c$$$        call zcopy(ngb*(nctot+nt0)*(ncc+ntp0),zmel,1,zmelt_tmp,1)
c$$$        call zgemm('T','N',ngb,(nctot+nt0)*(ncc+ntp0),ngb,(1d0,0d0),
c$$$     .   zmat,ngb,zmelt_tmp,ngb,(0d0,0d0),zmel,ngb)
c$$$        deallocate(zmelt_tmp)
c$$$        deallocate(zmmm)
c$$$        if(allocated(ppovl_)) deallocate(ppovl_)
c$$$!TIME1  "after matmul zmel"
c$$$c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$        print *,'xxxxxxxxxxxx 8829 xxxxxxxxxxxxxx'
c$$$         goto 8829
c$$$c        if(debug) write(6,'("4 zzzppp 111 ",3d15.6)') sum(abs(zmel)),sum(zmel)
c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$
c$$$
c$$$
c$$$
c$$$
c$$$c$$$!!march2013--> this if branch of nolfco is now unified to do 25
c$$$c$$$!! No LocalFieldCorrection mode
c$$$c$$$        if(nolfco) then !iepsmode==202) then ! just for <exp(iq r)|x0(q,\omega)|exp(iq r)>
c$$$c$$$          do jpm  = 1,npm !
c$$$c$$$            do ibib = 1, nbnb(k,jpm) !---  ibib loop
c$$$c$$$              if(jpm==1) then
c$$$c$$$                if( n1b(ibib,k,jpm) <= nbcut.and. n2b(ibib,k,jpm)>nbcut2) then !oct2005
c$$$c$$$                  if(iww2) then
c$$$c$$$                    write(6,"(' nband_chi0 nbcut nbcut2 n2b n1b=',4i6)")
 nbcut,n2b(ibib,k,jpm),n1b(ibib,k,jpm)
c$$$c$$$                    iww2=.false.
c$$$c$$$                  endif
c$$$c$$$                  cycle
c$$$c$$$                endif
c$$$c$$$              else
c$$$c$$$                if( n2b(ibib,k,jpm) <= nbcut.and. n1b(ibib,k,jpm)>nbcut2) then !oct2005
c$$$c$$$                  if(iww2) then
c$$$c$$$                    write(6,"(' nband_chi0 nbcut nbcut2 n2b n1b=',4i6)")
 nbcut,n2b(ibib,k,jpm),n1b(ibib,k,jpm)
c$$$c$$$                    iww2=.false.
c$$$c$$$                  endif
c$$$c$$$                  cycle
c$$$c$$$                endif
c$$$c$$$              endif
c$$$c$$$
c$$$c$$$              if( jpm==1.and.n2b(ibib,k,jpm) > nbmx) cycle
c$$$c$$$              if( jpm==2.and.n1b(ibib,k,jpm) > nbmx) cycle
c$$$c$$$
c$$$c$$$              if( n1b(ibib,k,jpm) <= nband) then
c$$$c$$$                it = nctot + n1b(ibib,k,jpm) !valence
c$$$c$$$              else
c$$$c$$$                it = n1b(ibib,k,jpm) - nband !core
c$$$c$$$              endif
c$$$c$$$
c$$$c$$$              if( n2b(ibib,k,jpm) <= nband) then
c$$$c$$$                itp = ncc + n2b(ibib,k,jpm) - itps + 1 !val
c$$$c$$$                if(itp > ncc + nkqmax-itps+1 ) cycle
c$$$c$$$              else
c$$$c$$$                itp =       n2b(ibib,k,jpm) - itps + 1 - nband !core
c$$$c$$$              endif
c$$$c$$$
c$$$c$$$              do imb2=1,nmbas
c$$$c$$$                zq02 = zmelt(imb2,it,itp)
c$$$c$$$                do imb1=1,imb2
c$$$c$$$                  zq01 = zmelt(imb1,it,itp)
c$$$c$$$                  zq0zq0 = dconjg(zq01)*zq02
c$$$c$$$                  do iw = ihw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1
c$$$c$$$                    if (iw .gt. nwt) stop "x0kf_v4hz: iw > nwt"
c$$$c$$$                    !iiww=iw+ihw(ibib,k)-1
c$$$c$$$                    imagweight   = whw(jhw(ibib,k,jpm)+iw-ihw(ibib,k,jpm))
c$$$c$$$                    if(eibzmode) imagweight = nwgt(k)*imagweight
c$$$c$$$c                    rcxqmean(iw,jpm,imb1,imb2) =  ! here we  sum over ibib (or n, n') and k.
c$$$c$$$c     &              rcxqmean(iw,jpm,imb1,imb2) + zq0zq0*imagweight
```

```
c$$$c$$$                    rcxq(imb1,imb2,iw,jpm) =  ! here we  sum over ibib (or n, n') and k.
c$$$c$$$     &              rcxq(imb1,imb2,iw,jpm) + zq0zq0*imagweight !sum over spin in hx0fp0
c$$$c$$$                  enddo ! iw
c$$$c$$$                enddo ! imb1
c$$$c$$$              enddo ! imb2
c$$$c$$$            enddo ! ----- ibib loop
c$$$c$$$          enddo ! ----- jpm loop
c$$$c$$$          deallocate(zmelt,z1p)
c$$$c$$$          cycle !cycye do 1000 here
c$$$c$$$        endif
c$$$c$$$!TIME1 "before jpm ibib loop"
c$$$c$$$!TIME0
c$$$
c$$$
c$$$
c$$$
c$$$
c$$$ 8828   continue


c$$$ this (ppovlz generation) is moved to hx0fp0 and/or hx0fp0_sc.
c$$$!! === zmelt conversion on different basis.
c$$$        if(chipmzzr) then       !spin moment basis.
c$$$c          if(allocated(ppovlz)) deallocate(ppovlz)
c$$$c          allocate(ppovlz(ngb,nmbas))
c$$$c          ppovlz= zzr
c$$$        elseif(nolfco .and. nmbas==1) then !for <e^iqr|x0|e^iqr>
c$$$          continue
c$$$        else                    !may2013  this removes O^-1 factor from zmelt
c$$$          allocate(ppovl_(ngb,ngb))
c$$$          ppovl_=0d0
c$$$          do i=1,nbloch
c$$$            ppovl_(i,i)=1d0
c$$$          enddo
c$$$          ppovl_(nbloch+1:nbloch+ngc,nbloch+1:nbloch+ngc)=ppovl
c$$$          if(.not.eibz4x0()) then !sep2014 added for eibz4x0=F
c$$$            ppovl_=matmul(ppovl_,zcousq)
c$$$          endif
c$$$          ppovlz = ppovl_
c$$$          deallocate(ppovl_)
c$$$        endif
c$$$        if(allocated(zmel)) deallocate(zmel)
c        nbmax= nctot+nt0
c        ntqxx= ncc+ntp0
cc        call get_zmelt(exchange,q,kx,qibz_k,irot,qbz_kr,kr,isp,
cc     &       ngc,ngb,nbmax,ntqxx,isp_k,isp_kq)
c
!! -------------------------------------------------------------------------------
!!note: for usual correlation mode, I think nctot=0
!!--- For dielectric funciton, we use irot=1 kvec=rkvec=q
!            < MPB      middle    |   end >
!!              q      rkvec     | q + rkvec
!                      nkmin:nt0 | nkqmin:ntp0
!                         occ    | unocc
!                      (nkmin=1)
!                      (cphi_k  | cphi_kq !in x0kf)
!
!!     rkvec= rk(:,k)   ! <phi(q+rk,nqmax)|phi(rk,nctot+nmmax)  MPB(q,ngb )>
!!     qbz_kr= rk(:,k)  !
!!     qibz_k= rk(:,k)  ! k
        ngb  = nbloch + ngc
!!Get the matrix element zmel   ZO^-1 <MPB psi|psi> , where ZO is ppovlz
!!  Output is zmel(ngb, nctot+nt0,ncc+ntp0)  nkmin:nt0, nkqmin:ntp0
! nt0=nkmax-nkmin+1  , ntp0=nkqmax-nkqmin+1
        call get_zmelt2(exchange,
     &      q,irot,q,ngc,ngb,                       !MPB
     &                  nkmin, nkmax, isp_k,nctot, !middle state 1:nt0   --> true index of eigen is
 mkmin:mkmin+nt0-1     + nctot
     &      q+rk(:,k),nkqmin,nkqmax,isp_kq,ncc)  !end state    1:ntp0  -->                     is
 nkqmin:nkqmin+ntp0-1
        zmel = dconjg(zmel)
        allocate( z1p(ngb,ngb) )

 8829   continue
        if(debug) write(6,'("4 zzzppp 222 ",3d15.6)') sum(abs(zmel)),sum(zmel)

!TIME1_1101 "before_get_zmelt2"
!TIME0_1201
c----------------------------
!!  z1p = <M_ibg1 psi_it | psi_itp> < psi_itp | psi_it M_ibg2 >
!!  zxq(iw,ibg1,igb2) = sum_ibib wwk(iw,ibib)* z1p(ibib, igb1,igb2)
!KINO        write(6,'(a,i4)')'kino: npm=',npm
!kino 2014-08-13  !$OMP parallel private(it,itp,iww1,iww2,zmelt2,imagweight)

ccccccccccccccccccccccccc
c        write(6,"('gggx ',3f9.4,x2x,3f9.4)") q+rk(:,k),rk(:,k)
ccccccccccccccccccccccccc
```

```

        do 25 jpm  = 1, npm !
        do 25 ibib = 1, nbnb(k,jpm) !---  ibib loop
!KINO          write(6,'(a,5i8)')'kino: ngb,hw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1=',
!KINO&      ngb,ihw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1
          if(n1b(ibib,k,jpm) <= nband) then
            it = nctot + n1b(ibib,k,jpm) !valence
            if(it > nctot + nkmax ) cycle
          else
            it = n1b(ibib,k,jpm) - nband !core
          endif
          if( n2b(ibib,k,jpm) <= nband) then
            itp = ncc + n2b(ibib,k,jpm) - itps + 1 !val
            if(itp > ncc + nkqmax-itps+1 ) cycle
          else
            itp =  n2b(ibib,k,jpm) - itps + 1 - nband !core
          endif

ccccccccccccccccccccccccc
c        write(6,"('gggx ',2i3)") ib,jb,ebandqr(it),ebandr(itp)
cccccccccccccccccccccc

ccccccccccccccccccccccccc
c          if(itp/=1) cycle
c          if(k/=1) cycle
cccccccccccccccccccccccccc


          if(jpm==1) then !nbmx is moved to tetwt5.
c$$$           if(n2b(ibib,k,jpm)>nbmx)  then  !nbmx
c$$$             if(iww1) then
c$$$               write(6,*)' nband_chi0 nbmx=',nbmx
c$$$               iww1=.false.
c$$$             endif
c$$$             cycle
c$$$           endif
            if( n1b(ibib,k,jpm) <= nbcut .and. nbcut2<n2b(ibib,k,jpm) ) then
             if(iww2) then
              write(6,"(' nband_chi0 nbcut nbcut2 n2b n1b=',4i6)") nbcut,n2b(ibib,k,jpm),n1b(ibib,k,jpm)
              iww2=.false.
             endif
             cycle
            endif
          else !jpm==2 ------------------------------
c$$$           if( n1b(ibib,k,jpm) > nbmx) then  !nbmx
c$$$             if(iww1) then
c$$$               write(6,*)' nband_chi0 nbmx=',nbmx
c$$$               iww1=.false.
c$$$             endif
c$$$             cycle
c$$$           endif
            if( n2b(ibib,k,jpm) <= nbcut .and. nbcut2<n1b(ibib,k,jpm) ) then
             if(iww2) then
              write(6,"(' nband_chi0 nbcut nbcut2 n2b n1b=',4i6)") nbcut,n2b(ibib,k,jpm),n1b(ibib,k,jpm)
                iww2=.false.
             endif
             cycle
            endif
          endif

cccccccccccccccccccccccccccccccccc takao variant begin
cc         if(takao) then
cc
cc         do ic = 1,ngb
cc           z1p(1:ngb,ic) =
cc     &     zmelt(ic,it,itp)*dconjg(zmelt(1:ngb,it,itp))
cc         end do
cc         ihww = ihw(ibib,k)
cc
c1ini------------
cc         do iw = 1, nhw(ibib,k)
cc           rviw = whw(jhw(ibib,k)+iw-1)
cC ... this part dominates the cpu time --------------------!
c!          call zaddr_(zxq(1,1,ihww+iw-1),rviw,z1p,ngb**2)
cc           call daxpy(ngb**2*2,rviw,z1p,1,
cc     &                 zxq(1,1,ihww+iw-1),1)
cc         enddo
c1end---------
c2ini --------
cc         call  rcxq_zxq(rc1p,z1p,ngb,-1)
cc         do iw = 1, nhw(ibib,k)
cc           rviw = whw(jhw(ibib,k)+iw-1)
C ... this part dominates the cpu time --------------------!
!          call zaddr_(rcxq(1,1,ihww+iw-1),rviw,z1p,ngb**2)
cc           call daxpy(ngb**2,rviw,rc1p,1,
cc     &                rcxq(1,1,ihww+iw-1),1)
cc         enddo
```

```
c2end --------------
cc         else
ccccccccccccccccccccccccc takao variant end

c$$$          if(newanisox.and.eibzmoden==1) then ! This is slow.
c$$$            zmeltx =  zmelt(:,it,itp)
c$$$            z1p=0d0
c$$$            do ieqbz =1, nwgt(k) !equivalent points for ieibz
c$$$                                 !igx,igxt specifies space-group operation (including ID)
c$$$              call rotMPB(zcousq,nbloch,ngbb,q,igx(ieqbz,k),igxt(ieqbz,k),ginv,zcousqrx)
 !zcousqr=Rotate_igx(zcousq)
c$$$              zmelty = matmul(zmeltx,zcousqrx)
c$$$              do igb2=1, ngb !...................................
c$$$                zmelt2 = zmelty(igb2)
c$$$              do igb1=1,igb2
c$$$                z1p(igb1,igb2) = z1p(igb1,igb2) + dconjg(zmelty(igb1)) * zmelt2
c$$$              enddo
c$$$              enddo
c$$$            enddo
c$$$          else
          if (ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1 >nwt) call rx( "x0kf_v4hz: iw>nwt")

!kino 2014-08-13  !$OMP do private(zmelt2)
          do igb2=1, nmbas     !...................................
             zmelt2 = zmel(igb2,it,itp) !zmelt(igb2,it,itp)
             do igb1=1,igb2
c                 z1p(igb1,igb2) = dconjg(zmelt(igb1,it,itp)) * zmelt2
                z1p(igb1,igb2) = dconjg(zmel(igb1,it,itp)) * zmelt2
             enddo
          enddo

!! ---------------------------------
          if(crpa) then
c            print *,'readout readqkm init'
            if(n1b(ibib,k,jpm) <= nband) then
              call readpkm4crpa(n1b(ibib,k,jpm),   rk(:,k), isp_k,   wpw_k) !k  n1b
            else
              wpw_k=0d0
            endif
            if(n2b(ibib,k,jpm) <= nband) then
              call readpkm4crpa(n2b(ibib,k,jpm), q+rk(:,k), isp_kq,  wpw_kq) !kq n2b
            else
              wpw_kq=0d0
            endif
c            write(6,"('rrrrrr: n1b wpw_k n2b wpw_kq fac irange=',i5,x,f9.4, i5,x,f9.4, x,f9.4, 2i5)")
c     &           n1b(ibib,k,jpm),wpw_k, n2b(ibib,k,jpm),wpw_kq, (1d0-wpw_k*wpw_kq),
c     &           ihw(ibib,k,jpm), ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1
          endif
c$$$cccccccccccccccccccccccccccccccccccc
c$$$ For SrVO3 test
c$$$          if(crpa) then
c$$$             wpw_k=0d0
c$$$             if(15<n1b(ibib,k,jpm).and.n1b(ibib,k,jpm)<19) wpw_k=1d0
c$$$             wpw_kq=0d0
c$$$             if(15<n2b(ibib,k,jpm).and.n2b(ibib,k,jpm)<19) wpw_kq=1d0
c$$$          endif
c$$$ccccccccccccccccccccccccccccccccccccc


ccccccccccccccccccccccccccccccc
!kino 2014-08-13  !$OMP end do
c$$$  endif

!$OMP parallel private(imagweight)
!$OMP master
!$        if (jpm.eq.1 .and. ibib.eq.nbnb(k,1)) then
!$          write(6,'(a,i5,a,i5)') 'OMP parallel iw, threads=', omp_get_num_threads(),
!$   .     ' nw=',ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-ihw(ibib,k,jpm)
!$        endif
!$OMP end master
!$OMP do
          do iw = ihw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1 !iiww=iw+ihw(ibib,k)-1
             imagweight = whw(jhw(ibib,k,jpm)+iw-ihw(ibib,k,jpm))
             if(crpa) imagweight = imagweight*(1d0-wpw_k*wpw_kq)
             if(eibzmode) imagweight = nwgt(k)*imagweight
             do igb2=1,nmbas      !this part dominates cpu time most time consuming..........
                do igb1=1,igb2
                   rcxq(igb1,igb2,iw,jpm) = !here we  sum over ibib (or n, n') and k.
     &                  rcxq(igb1,igb2,iw,jpm) + z1p(igb1,igb2)*imagweight !sum over spin in hx0fp0
                enddo           !igb1
             enddo              !igb2
          enddo                 ! iw
!$OMP end do
!$OMP end parallel
 25     continue
!kino 2014-08-13  !$OMP end parallel
!TIME1_1201 "after_rcxq"
```

```

c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$c        if(ipr) then
c$$$          do jpm=1,npm
c$$$            write(6,"(' k jpm sum(rcxq) ngb ngbb=',2i5,2d23.15,2i8)")
c$$$     &      k,jpm,sum(rcxq(:,:,:,jpm)),ngb,ngbb
c$$$          enddo
c$$$          do ib1 =1,ngbb
c$$$             if(ib1<4) then
c$$$             elseif(ib1>ngbb-3) then
c$$$             else
c$$$                cycle
c$$$             endif
c$$$          do ib2 =1,ngbb
c$$$             if(ib2<4) then
c$$$             elseif(ib2>ngbb-3) then
c$$$             else
c$$$                cycle
c$$$             endif
c$$$          do iw =1,nwt
c$$$            write(6,"('uuu: k iw ib1 ib2 sum(rcxq)=',4i5,4d23.15)")
c$$$     &       k,iw,ib1,ib2,(rcxq(ib1,ib2,iw,1)), (rcxq(ib1,ib2,iw,2))
c$$$          enddo
c$$$          enddo
c$$$          enddo
c$$$c        endif
c$$$cccccccccccccccccccccccccccccccccccccccccccccccccc
        deallocate(z1p,zmel) !zmelt,z1p)
        if(debug) call cputid(0)
        if(debug) write(6,*)' end of kloop k jpm=',k,jpm
 1000 continue

!! Not need to be symmetrized
      if(nolfco .and. nmbas==1) then
         write(6,*)' nmbas=1 nolfco=T ---> not need to symmetrize'
         goto 9999
      endif
!TIME0_1301


!! ==== Hermitianize. jun2012takao moved from dpsion5 ====
c      if(eibzmode) then !comment out sep2014.
        do jpm=1,npm
        do iw= 1,nwt
         do igb2= 1,nmbas !eibzmode assumes nmbas1=nmbas2
         do igb1= 1,igb2-1
           rcxq(igb2,igb1,iw,jpm) = dconjg(rcxq(igb1,igb2,iw,jpm))
         enddo
         enddo
        enddo
        enddo
c       endif
!TIME1_1301 "before_eibzmode_symmetrization"
 9999 continue
      write(6,"(' --- x0kf_v4hz: end')")
      end subroutine x0kf_v4hz


!! ---------------------------------------------------------------------------------
      subroutine x0kf_v4hz_symmetrize (npm, !ncc,
c     i            ihw,nhw,jhw,whw,nhwtot, ! tetwt5
c     i            n1b,n2b,nbnbx,nbnb,     !  use whw by tetwt5 ,
     i                  q,
     i                  nsp,isp_k,isp_kq, !symmetrize,
     i                  qbas,ginv,!rk,wk,
c     i                  mdim,
c     d                  nlmto,nqbz,nctot,
c     d                  natom,
     d                  nbloch,nwt,
     i      iq, ngbb, ngc, ngpmx,ngcmx,
     i      nqbze, nband,nqibz,
     o      rcxq,
     i     nolfco,zzr,nmbas, zcousq,
     i     chipmzzr,eibzmode,
     i     ngrp,eibzsym) !, crpa) !nwgt,igx,igxt,
c      use m_readqg,only   :  readqg
c      use m_readeigen,only:  readeval
c      use m_keyvalue,only   :  getkeyvalue
      use m_rotmpb,only   :  rotmpb2
      use m_readqgcou,only:
     o qtt_, nqnum
c      use m_pkm4crpa,only : readpkm4crpa
c      use m_zmel,only     : get_zmelt2,
c     o zmel !,ppbir ,ppovlz
!! === symmetrization for EPIBZ mode ===
      implicit none
      integer(4):: npm,ncc,ngbb,natom,nwt,nsp,isp_k,isp_kq !nlmto !,noccx,noccxv
```

```fortran
     &  ,nl,nclass,nnc,nlnmx,nbloch,iq,nqibz,iatom,nctot,nbmx,iopen !mdimx,
     &  ,jpm,ibib,itps,nt0,ntp0,ngp_kq,ngp_k,it,itp,iw,igb2,igb1,ngb
     &  ,nn,no,isx,iclose,k,nbnbx,nqbz
      real(8):: q(3),qbas(3,3),ginv(3,3),ebmx !,rk(3,nqbz),wk(nqbz)
      complex   (8):: rcxq (nmbas,nmbas,nwt,npm)
      complex(8) :: imag=(0d0,1d0),trc,aaa !phase(natom),
      integer(4):: ngpmx, ngcmx, nqbze, nband,
     &              ngc,nadd(3)
c      integer(4) ::  nbnb(nqbz,npm), n1b(nbnbx,nqbz,npm), n2b(nbnbx,nqbz,npm)
      logical,parameter:: debug=.false.
c      integer(4):: nhwtot, ihw(nbnbx,nqbz,npm),nhw(nbnbx,nqbz,npm),jhw(nbnbx,nqbz,npm)
c      real(8):: whw(nhwtot)
      complex(8) :: zmelt1,zmelt2,zmeltt(ngbb)      !............................sf 21May02
      real(8) :: imagweight !............................sf 21May02
      integer(4)::nocc
      real(8):: eband(nband)!,ebandr(nband),ebandqr(nband)
      integer(4):: verbose

      logical   :: nolfco !iepsmode
      integer(4):: nmbas, imb1,imb2, imb !nmbas1x !nmbas2,nmbas1,
      real(8):: vec_kq_g(3),vec_k_g(3),vec_kq(3),vec_k(3),quu(3),tolq=1d-8,quu1(3),quu2(3) !tolqu=1d-4,
      integer(4):: nbcut,nbcut2
      logical :: iww1=.true.,iww2=.true.
      complex(8):: img=(0d0,1d0)
      integer(4):: nkmin,  nkmax, nkqmin, nkqmax,nkmax1,nkqmax1
      integer(4):: ib1, ib2,      ngcx,ix,iy
      complex(8),target :: zzr(ngbb,nmbas) !ppovlz(ngbb,ngbb),
      integer:: igb
      integer:: ngrp !,nwgt(nqbz) !,ngrpt, aiktimereversal(ngrpt),nwgtieibz,ieibz
c      integer:: igx(ngrp*2,nqbz),igxt(ngrp*2,nqbz),ieqbz
      logical:: checkbelong,eibzmode, chipmzzr
      complex(8):: zcousq(ngbb,ngbb) ,  zcousqc(ngbb,ngbb)
      integer::  eibzsym(ngrp,-1:1),neibz,icc,ig,eibzmoden,ikp,i,j,itimer,icount,iele
      integer:: irotm,nrotmx,ixx,iyy,itt,ntimer, nccc, nxx,iagain,irotm1,irotm2
      integer,allocatable:: i1(:,:),i2(:,:),nrotm(:)
      complex(8),allocatable:: zrotm(:,:),zrr(:,:),zrrc(:,:),zrr_(:,:,:),zrrc_(:,:,:),zmmm(:),zrrx(:,:)
c      complex(8),pointer:: zmat(:,:)
      complex(8),allocatable:: rcxq_core(:,:)
      complex(8),allocatable:: zcousqr(:,:,:),rcxq0(:,:),rcxq00(:,:),rcxq000(:,:),rcxqwww(:,:)

c      logical:: eibz4x0
c      logical :: crpa
c      real(8):: wpw_k,wpw_kq
c      real(8):: vec_kcrpa(3),vec_kqcrpa(3)

      logical :: exchange=.false.
      integer:: irot=1
      integer:: ntqxx,nbmax

!! --------------------------------------------------------------------------
!! == Symmetrizer of EIBZ PRB.81,125102(2010) Eq.(51) july2012takao ==
!! This may be not so effective ---> only for limited cases?
!! --- zrotm(J,J') = <Mbar^k_J| \hat{A}^k_i Mbar^k_J'>. ---
!! We do \sum_i T_alpha_i [ zrotm_i^dagger (I,I') P_I'J' zrom_i(J'J) ]
!! (exactrly speaking, we insert conversion matrix between Enu basis and M_I basis).
!!
!! input qin = q
!! \hat{A}^k_i  is specified by symops(:,:,igx),and igxt (-1 for time-reversal).
!! Note that k= \hat{A}^k_i(k) (S_A^k)
!! See Eq.(51) around in PRB81 125102(2010)
!!
c 5000 continue
!! === zmelt conversion ===
      if(nolfco .and. nmbas==1) then
         write(6,*)' nmbas=1 nolfco=T ---> not need to symmetrize'
         goto 9999
      endif
!!
      if(eibzmode) then
        ngb  = nbloch + ngc ! This is not ngbb for smbasis()=T. oct2005
        if(ngb/=ngbb) then
           write(6,*)' x0kf_v4h: ngb ngbb=',ngb,ngbb
           call rx( 'x0kf_v4h: ngb/=ngbb')
        endif
!TIME0_1401
        call iqindx2(q, ginv, qtt_, nqnum, ikp,quu) !to get ikp for timereversal mode
!TIME1_1401 "after_iqindx2"
!TIME0_1501
        if(sum(abs(q-quu))>tolq) call rx( 'x0kf_v2h: eibz 111 q/quu')
        neibz = sum(eibzsym(:,1))+sum(eibzsym(:,-1))
             !itimer=-1 means time reversal. eibzsym(ig,itimer) where ig: space rotation.
        write(6,"(' --- goto symmetrization --- ikp neibz q=',2i3,3f12.8)")ikp,neibz,q
        call cputid2(' --- x0kf: start symmetrization  ',0)

c        allocate(rcxq0(ngb,ngb),rcxq00(ngb,ngb),rcxq000(ngb,ngb),rcxqwww(ngb,ngb))
        ntimer=1
        if(sum(eibzsym(:,-1))>0) ntimer=2 !timereversal case
        allocate(zrotm(ngb,ngb),nrotm(ngrp*2))
!!
c        zcousqinv=zcousq
c        call matcinv(ngb,zcousqinv)

!! == Assemble rotantion matrx zrr,zrrc ==
!! Rotation matrix zrrx can be a sparse matrix.
!! Thus it is stored to  "i1(nrotmx,nccc),i2(nrotmx,nccc),zrr(nrotmx,icc),nrotm(icc)".
!! See folloings: matmul(rcxqwww,zrrx) is given by
!!     do irotm1 = 1,nrotm(icc)
!!       rcxq0(:,i2(irotm1,icc)) = rcxqwww(:,i1(irotm1,icc)) * zrr(irotm1,i2(irotm1,icc))

        allocate(zrrx(nmbas,nmbas))
        nrotmx = 10000 !trial value
!TIME1_1501 "before_1011"
        do 1011  !this loop is only in order to to set large enough nrotmx.
!TIME0_1601
        if(allocated(i1)) deallocate(i1,i2,zrr,zrrc)!,zrr_,zrrc_)
        nccc=ngrp*2
        allocate(i1(nrotmx,nccc),i2(nrotmx,nccc),zrr(nrotmx,nccc),zrrc(nrotmx,nccc))
!,zrr_(ngb,ngb,nccc),zrrc_(ngb,ngb,nccc))
        i1=-99999
        i2=-99999
        zrr=-99999d0
        zrrc=-99999d0
        call cputid2(' --- x0kf:11111   :',0)
!TIME1_1601 "allocatezrr"
!!
        icc=0
        do itimer=1,-1,-2
          if(ntimer==1.and.itimer==-1) exit
          if(itimer==1 ) itt=1
          if(itimer==-1) itt=2
          do ig=1,ngrp
            if(eibzsym(ig,itimer)==1) then
              icc=icc+1
!TIME0_1701

!! Get rotation matrix zrrx, which can be a sparse matrix. Thus stored to zrr.
              call rotmpb2(nbloch,ngb,q,ig,itimer,ginv,zrotm)
              if(nolfco.and.chipmzzr) then
!!   We assume <svec_I | svec_J >= \delta_IJ, In addition, we use fact that we have no IPW parts in svec.
!!   If IPW part exist, we may have to take into account <IPW|IPW> matrix, e.g. as in ppovlz.
!!   svec --> zzr
               if(itimer==1) then
                  zrrx= matmul(transpose(dconjg(zzr)), matmul(zrotm, zzr))
                else
                  zrrx= matmul(transpose(zzr), matmul(dconjg(zrotm), zzr))
                endif
              elseif(nolfco) then
                 call rx( 'x0kf_v4h: this case is not implemented xxxxxxxxxxxxxx')
              else
!! zrotm(J,J') is the rotation matrix = <Mbar^k_J| \hat{A}^k_i Mbar^k_J'>
!! See rotMPB2 defined in readeigen.F.
!! zrrx(mu nu)= dconjg(Zcousq(I, mu)) *zrotm(I,J)* Zcousq(J, nu)
!! zrrx is very sparse matrix. Size is \sim ngb or something.

c$$$                if(itimer==1) then
c$$$                  call matmmsparse(zcousqinv,zrotm,zcousq,zrrx,ngb,1d-8,iele)
c$$$                  ! this means zrrx= matmul(zcousqinv,matmul(zrotm, zcousq))
c$$$                else
c$$$                  call matmmsparse(dconjg(zcousqinv),dconjg(zrotm),zcousq,zrrx,ngb,1d-8,iele)
c$$$                  ! this means zrrx= matmul(dconjg(zcousqinv),matmul(dconjg(zrotm), zcousq))
c$$$                endif

                if(itimer==1) then
                  zrrx=zrotm
c                  call matmmsparse(zcousqinv,zrotm,zcousq,zrrx,ngb,1d-8,iele)
                  ! this means zrrx= matmul(zcousqinv,matmul(zrotm, zcousq))
                else
                  zrrx=dconjg(zrotm)
c                  call matmmsparse(dconjg(zcousqinv),dconjg(zrotm),zcousq,zrrx,ngb,1d-8,iele)
                  ! this means zrrx= matmul(dconjg(zcousqinv),matmul(dconjg(zrotm), zcousq))
                endif

              endif
!TIME1_1701 "end_matmmsparse"
!TIME0_1801
              i1(:,icc)=0
              i2(:,icc)=0
              irotm=0
              iagain=0
              do ix=1,ngb
              do iy=1,ngb
                if(abs(zrrx(ix,iy))>1d-8) then
                  irotm=irotm+1
                  if(irotm>nrotmx) then
```

```
                    iagain=1
                  endif
                  if(iagain/=1) then
                  i1(irotm,icc)=ix
                  i2(irotm,icc)=iy
                  zrr(irotm,icc) = zrrx(ix,iy)
                  zrrc(irotm,icc)= dconjg(zrr(irotm,icc))
                  endif
                endif
              enddo
              enddo
!TIME1_1801 "before_iagain1"
!TIME0_1901
              if(iagain==1) then
                nrotmx=irotm !enlarge allocation and do things again.
                write(6,*)' warn:(slow speed) xxxx goto 1011 xxxxxx nrotmx+=nrotmx+10000 again'
                goto 1011
                !enlarge nrotmx ang try it again.
              endif
              nrotm(icc)=irotm
              if(debug) write(6,*)'ig itimer icc nrotm=',ig,itimer,icc,nrotm(icc) ,iele
!TIME1_1901 "end_ig_itimer_icc_nrotm"
            endif
          enddo
        enddo
        exit
 1011   continue !only when nrotmx overflow.
!TIME0_2001

!! === main part to obtain symmetrized rcxq  ===
!! neibz is total number of symmetrization operation.
!!      rcxq is rotated and accumulated; finally divied by neibz
        zcousqc = dconjg(transpose(zcousq))
        if(debug) call cputid2(' --- x0kf:qqqqq222ini:',0)
!$OMP parallel private(rcxq000,icc,itt,icount,rcxqwww,rcxq00,rcxq0,rcxq_core)
        allocate(rcxq0(ngb,ngb),rcxq00(ngb,ngb),rcxq000(ngb,ngb),rcxqwww(ngb,ngb),rcxq_core(ngb,ngb))
!$OMP master
!$         write(6,'(a,i5,a,i5)') 'OMP parallel nwt, threads=',omp_get_num_threads(),' nwt=',nwt
!$OMP end master
!$OMP do
        do iw=1,nwt
        do jpm=1,npm
          rcxq000 = 0d0
          icc=0
          do itimer=1,-1,-2
            if(itimer==1 ) itt=1
            if(itimer==-1) itt=2
            icount=0
            if(itimer==1) then
              rcxqwww = rcxq(:,:,iw,jpm)
            else
              rcxqwww = transpose(rcxq(:,:,iw,jpm))
            endif
            rcxq00 = 0d0
            do ig=1,ngrp
              if(eibzsym(ig,itimer)==1) then
               icount=icount+1
               icc=icc+1
               rcxq0 =0d0

c$$$               if(itimer==1) then
c$$$               do irotm1 = 1,nrotm(icc)
c$$$               do irotm2 = 1,nrotm(icc)
c$$$               rcxq0(i2(irotm2,icc),i2(irotm1,icc)) =rcxq0(i2(irotm2,icc),i2(irotm1,icc))
c$$$     &              +    zrrc(irotm2,icc)* rcxq(i1(irotm2,icc),i1(irotm1,icc),iw,jpm)*zrr(irotm1,icc)
c$$$               enddo
c$$$               enddo
c$$$               else
c$$$               do irotm1 = 1,nrotm(icc)
c$$$               do irotm2 = 1,nrotm(icc)
c$$$               rcxq0(i2(irotm1,icc),i2(irotm2,icc)) =rcxq0(i2(irotm1,icc),i2(irotm2,icc)) !transpose
c$$$     &              +    zrrc(irotm2,icc)* rcxq(i2(irotm2,icc),i1(irotm1,icc),iw,jpm)*zrr(irotm1,icc)
c$$$               enddo
c$$$               enddo
c$$$               endif

!!  Followings are equivalent with
!!            rcxq00= rcxq00 + matmul(zrrc_(:,:,icc),matmul(rcxqwww,zrr_(:,:,icc)))
               do irotm1 = 1,nrotm(icc)
c                 if(abs(zrr(irotm1,icc))<1d-8) cycle
                 rcxq0(:,i2(irotm1,icc)) =rcxq0(:,i2(irotm1,icc)) + rcxqwww(:,i1(irotm1,icc)) * zrr(irotm1,icc)
               enddo
               do irotm2 = 1,nrotm(icc)
c                 if(abs(zrrc(irotm2,icc))<1d-8) cycle
                 rcxq00(i2(irotm2,icc),:)= rcxq00(i2(irotm2,icc),:) + zrrc(irotm2,icc) * rcxq0(i1(irotm2,icc),:)
               enddo

c               if(itimer==1) then
c                 rcxq000 = rcxq000 + rcxq00
c               else
c                 rcxq000 = rcxq000 + transpose(rcxq00)
c               endif
c
c$$$                do irotm = 1,nrotm(icc)
c$$$                 iyy = i1(irotm,icc)
c$$$                 iy  = i2(irotm,icc)
c$$$                 rcxq0(:,iy)= rcxq0(:,iy)+ rcxq(:,iyy,iw,jpm)* zrr(irotm,icc)
c$$$                enddo
c$$$                do irotm = 1,nrotm(icc)
c$$$                 iyy = i1(irotm,icc)
c$$$                 iy  = i2(irotm,icc)
c$$$                 rcxq00(iy,:)= rcxq00(iy,:)+ dconjg(zrr(irotm,icc)) * rcxq0(iyy,:)
c$$$                enddo
c$$$
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c               if(iw==1.and.jpm==1) then
c                  write(6,"('bbbbbbb ig icc iw jpm rcxq', 4i3, 13d13.6)")
c     &                 ig,icc,iw,jpm, sum(abs(rcxq00)), rcxq00(1,1),sum(abs(rcxqwww)),sum((rcxqwww))
c               endif
ccccccccccccccccccccccccccccccccccccccccccccccccccccc

              endif
            enddo

c$$$           if(itimer==1) then
c$$$             rcxq000(:,:) = matmul(zcousqc,matmul(rcxq00,zcousq))
c$$$c$$$c               call zgemm("N","N",ngb,ngb,ngb, (1d0,0d0), rcxq00, ngb, zcousq,ngb, (0d0,0d0), rzc,ngb)
c$$$c$$$c               call zgemm("N","N",ngb,ngb,ngb, (1d0,0d0), zcousqc,ngb, rzc,ngb, (0d0,0d0), rcxq000,ngb)
c$$$           elseif(icount>0) then
c$$$c$$$c           write(6,*)'qqqqq icount=',icount
c$$$c$$$c           rcxq000(:,:) = rcxq000(:,:) + transpose(matmul(transpose(zcousq),matmul(rcxq00,dconjg(zcousq))))
c$$$             rcxq000(:,:) = rcxq000(:,:) +   matmul(matmul(zcousqc,transpose(rcxq00)),zcousq)
c$$$           endif

            if(itimer==1) then
               rcxq000=rcxq00
            else
               rcxq000=rcxq000+rcxq00
            endif
          enddo
          rcxq_core = rcxq000/neibz
#if 1
!! matmul(rcxq(:,:,iw,jpm),zcousq) fails in ifort 14.0.3.
!! It looks that ifort 14.0.3 has a bug
!! But, zgemm works. So I changed like that.
          call zgemm('N','N',ngb,ngb,ngb,(1.0d0,0.0d0),rcxq_core,ngb,zcousq, ngb, (0.0d0,0.0d0),rcxq000,ngb)
          call zgemm('N','N',ngb,ngb,ngb,(1.0d0,0.0d0),zcousqc  ,ngb,rcxq000,ngb, (0.0d0,0.0d0),rcxq_core,ngb)
          rcxq(:,:,iw,jpm) = rcxq_core
#else
          rcxq(:,:,iw,jpm) = matmul(zcousqc,matmul(rcxq_core,zcousq))
#endif
          enddo
          enddo
!$OMP end  do
        deallocate(rcxq00,rcxq000,rcxq0,rcxqwww)
!$OMP end parallel
!TIME1_2001 "after_sym_rcxq"
        deallocate(zrotm,i1,i2)

c$$$        allocate(zcousqr(ngb,ngb,neibz),rcxq0(ngb,ngb),rcxq00(ngb,ngb),rcxqtr(ngb,ngb))
c$$$        icc=0
c$$$        do itimer=1,-1,-2
c$$$        do ig=1,ngrp
c$$$          if(eibzsym(ig,itimer)==1) then
c$$$            icc=icc+1
c$$$            if(itimer==1) then
c$$$              call rotMPB(zcousq,nbloch,ngb,q,ig,itimer,ginv,zcousqr(1,1,icc))
c$$$            else
c$$$!! time reversal mapping ---
c$$$              call rotMPB(dconjg(zcousq),nbloch,ngb,q,ig,itimer,ginv,zcousqr(1,1,icc))
c$$$            endif
c$$$          endif
c$$$        enddo
c$$$        enddo
c$$$
c$$$        do iw=1,nwt
c$$$        do jpm=1,npm
```

```
c$$$          rcxq0=0d0
c$$$          icc=0
c$$$c          do itimer=1,1 !1,-1,-2
c$$$          do itimer=1,-1,-2
c$$$          do ig=1,ngrp
c$$$            if(eibzsym(ig,itimer)==1) then
c$$$             icc=icc+1
c$$$             rcxq00(:,:) = matmul(dconjg(transpose(zcousqr(:,:,icc))),
c$$$     &                             matmul(rcxq(:,:,iw,jpm),zcousqr(:,:,icc)))
c$$$!! time reversal mapping ---
c$$$             if(itimer==-1) rcxq00(:,:) = transpose(rcxq00)
c$$$             rcxq0(:,:) = rcxq0(:,:)+ rcxq00(:,:)
c$$$            endif
c$$$          enddo
c$$$          enddo
c$$$          rcxq(:,:,iw,jpm)=rcxq0(:,:)/neibz
c$$$        enddo
c$$$        enddo
c$$$        deallocate(zcousqr,rcxq0,rcxq00,rcxqtr)
        if(debug) call cputid2(' --- qqqqq222end:',0)
      endif
 9999 continue
!kino 2014.08.19 use automatic deallocation,      deallocate(cphi_k,cphi_kq,geig_kq,geig_k)
      write(6,"(' --- x0kf_v4hz_symmetrize: end')")
      end subroutine x0kf_v4hz_symmetrize




C=====================================================================
        subroutine dpsion5  (frhis,nwhis, freqr,nw_w, freqi,niwt,
     i                               realomega,   imagomega,      !freqr ->frhis ...sf
     i                     rcxq, npm, nw_i,nmbas1,nmbas2,
     o    zxq,zxqi,
c     i    nolfco,chipm,schi,isp,  rcxqmean,nmbas,  !iepsmode, rcxqmean, ! epsmode
     i    chipm,schi,isp,   !No nolfco mode. Apr2012.
     i    ecut,ecuts)
c     o    x0mean)
C- Calculate W-v zxqi(on the imaginary axis) and zxq(real axis) from sperctum weight rcxq.
Cr v4 works for timereversal=F (npm=2 case).
Cr  See rcxq_zcxq for rcxq, which contains the spectrum weight for given bins along the real-axis.
Cr ! Note that zxq and zxqi are not accumlating
Ci frhis(1:nwhis+1) :: specify histgram bins i-th bin is [frhis(i), frhis(i+1)].
Ci          We suppose "freqr(i)=moddle of i-th bin; freqr(0)=0."
Ci          (I think called routine hilbertmat itself is not limited by this condition).
Ci freqr (0:nw_w) : Calcualte zxq for these real energies.
Ci freqi (1:niwt) : Calcualte zxqi for these imaginary energies.
Ci   realomega  : A switch to calculate zxq or not.
Ci   imagomega: : A switch to calculate zxqi or not.
Ciw rcxq may be altered ---used as work area.
Cio   zxq :  W-v along the real axis on freqr(0:nw_w)
Cio   zxqi:  W-v along the imag axis on freqi(niwt)
C!
C1 Feb2006:  v4 for timereversal=F
C! July2005: v3Add spin chipm mode
C! July2005: This version alter rcxq----it is used as work area.
C! sergey faleev Apr 2002 ; Rebuiled by takao
C-------------------------------------------------------------------
        implicit none
        integer(4):: nw_w,niwt,igb1,igb2, iw,iwp,nwhis,ix,npm,ifxx,nmbas1,nmbas2
        real(8) :: freqi(niwt),pi,px,omp,om,om2,om1, !omg2max from hx0fp0
     &  frhis(nwhis+1), freqr(0:nw_w), aaa,d_omg
        logical :: realomega, imagomega
        complex(8):: rcxq(nmbas1,nmbas2, nwhis,npm) !sf 13June
c      logical   :: iepsmode
        logical :: chipm

        integer(4)::isp,ispx !, nmbas
c      complex(8):: rcxqmean(nwhis,npm,nmbas,nmbas)  !takao sep2006 add nmbas
C... ecut mode
        real(8):: ecut,ecuts,wcut,wcutef,dee,schi
        logical ::debug=.false.
        real(8),allocatable :: his_l(:),his_r(:),his_c(:)
        integer(4) it
        real(8):: domega_r,domega_c,domega_l,delta_l,delta_r
        real(8),allocatable ::rmat(:,:,:),rmati(:,:,:),rmatt(:,:,:),imatt(:,:,:)
        complex(8),allocatable :: rmatic(:,:,:),imattc(:,:,:)
        complex(8) ::beta,wfac
        complex(8):: zz
        complex(8),allocatable :: zxqn(:,:),zxqn1(:,:,:),rx0mean1(:,:,:),rx0mean(:)
        complex(8),allocatable:: rrr(:)

        integer(4)::nw_i,jpm,ipm,verbose,isgi
c      complex(8):: x0mean(nw_i:nw_w,nmbas,nmbas)
        complex(8)::
     o   zxq(nmbas1,nmbas2, nw_i: nw_w), !iw=0 means omg=0,
        !iw=1:nw_w corresponds to iw's bit of the frequensy histogram
     o   zxqi(nmbas1,nmbas2,niwt),img !npm), img  !zxqi(...,npm) may2006

        real(8),allocatable:: ebb(:)
        integer(4):: ii,i,ibas1,ibas2
        logical :: evaltest !,testtr

c        if(verbose()>89) debug=.true.
c ------------------------------------------------
        write(6,'(" -- dpsion5: start...   ",$)')
        write(6,"('  nw_w nwhis=',2i5)") nw_w,nwhis
        if(debug) then
          write(6,*)' nmbas1 nmbas2 nwhis npm =',  nmbas1,nmbas2,nwhis,npm
          write(6,*)' sumchk rcxq=', sum(abs(rcxq))
        endif
        pi  = 4d0*datan(1d0)
        img = (0d0,1d0)
        call cputid(0)
        ispx = isp
        if(schi<0) then
          ispx = 3-isp !flip
        endif

!! Check freqr
        if(realomega) then
          if( nwhis <= nw_w ) then
            write(6,*)nwhis,nw_w
            call rx( ' dpsion5: nwhis<=nw_w')
          endif
          if( freqr(0)/=0d0 ) call rx( ' dpsion5: freqr(0)/=0d0')
!! I think current version allows any freqr(iw), independent from frhis.
c$$$          aaa = 0d0
c$$$          if(nw_w>0) then
c$$$            do iw = 1,nw_w
c$$$              aaa = aaa + abs( freqr(iw) - (frhis(iw)+frhis(iw+1))/2d0 )
c$$$              if(debug) write(6,"(' iw freqr frhis_m=',i5,2f13.6)" )
c$$$     &        iw,freqr(iw),  (frhis(iw)+frhis(iw+1))/2d0
c$$$            enddo
c$$$            if(aaa>1d-10)call rx( 'dpsion5:freqr/=frhis_m is not implimented yet')
c$$$          endif
        endif !realomega

C---------------------------------------------------------------
!! Each histogram bins are  [his_Left, his_Right], and  his_Center is middle.
!! his_C(0) is at zero. his_R(0) and his_L(0) are not defined.
        if(debug) write(6,*)' dpsion5: RRR 2222222222 '
        allocate(his_l(-nwhis:nwhis),his_r(-nwhis:nwhis),his_c(-nwhis:nwhis))
        his_l(1:nwhis) = frhis(  1:  nwhis)
        his_r(1:nwhis) = frhis(1+1:1+nwhis)
        his_c(1:nwhis) = (his_l(1:nwhis) + his_r(1:nwhis) )/2d0
        do iw= 1,nwhis
          his_l(-iw) = -his_r(iw)
          his_r(-iw) = -his_l(iw)
          his_c(-iw) = -his_c(iw)
        enddo
        his_c(0) = 0d0; his_r(0)=-999; his_l(0)=-999
C
        if(debug) write(6,*)'sumchk 111 rcxq=', sum(abs(rcxq))

        do iw= 1, nwhis
          if(ecut<1d9) then
            wfac= wcutef(his_c(iw), ecut,ecuts)
          else
            wfac= 1d0
          endif
! rcxq is used as work---> rcxq= Average value of Im chi.
! Note rcxq is "negative" (
          do jpm=1,npm
            call dscal(2*nmbas1*nmbas2, -wfac/(his_r(iw)-his_l(iw)),rcxq(1,1,iw,jpm),1)
          enddo
c         if(debug) write(6,*) 'dpsion5: RRR 7777 iw wfac=',iw,wfac,ecut,ecuts
        enddo
        if(debug) write(6,*)'sumchk 122 rcxq=', sum(abs(rcxq))

C... Temporary. maybe, we will have better procedure...
ctakao moved this to x0kv_v4h.F jun2012takao
!! hermitianize.
c        if(nmbas1==nmbas2) then !Is this required??? apr2012takao
c          do jpm=1,npm
c            do iw= 1, nwhis
c              do igb2= 1, nmbas2
c                do igb1= 1, igb2-1
c                  rcxq(igb2,igb1,iw,jpm) = dconjg(rcxq(igb1,igb2,iw,jpm))
c                enddo
c              enddo
c            enddo
c          enddo
c        endif
```

```
cccccccccccccccccccc
        if(debug) write(6,*)'sumchk 222 rcxq=', sum(abs(rcxq))

        if(evaltest().and.nmbas1==nmbas2) then
          write(6,"('hhh --- EigenValues for rcxq --------')")
          allocate(ebb(nmbas1))
          do jpm= 1,npm
            do iw = 1, nwhis
              call diagcvh2(rcxq(:,:,iw,jpm),nmbas1,ebb)
              do ii=1,nmbas1
                write(6,"('hhh1: xxxxxxxxxxxxxxxxx',2i4)") jpm,iw
                if(abs(ebb(ii))>1d-8.and.ebb(ii)>0)
     &        write(6,"('hhh1: jpm iw eb=',2i4,d13.5)") jpm,iw,ebb(ii)
              enddo
            enddo
          enddo
          deallocate(ebb)
        endif

C--- realomega case
        if(realomega)then
          write(6,*) " --- realomega --- "
          if(npm==1) then
            allocate( rmat(0:nw_w,-nwhis:nwhis,npm), rrr(-nwhis:nwhis))
            rmat  = 0d0
            do it =  0,nw_w
              zz = freqr(it) !his_C(it)
              call hilbertmat(zz, nwhis,his_l,his_c,his_r, rrr)
              rmat(it,:,1) = dreal(rrr)
            enddo ;   if(debug) write(6,*) 'dpsion5: RRR 55555555555'
            allocate( rmatt(0:nw_w,nwhis,npm) )
            if(     chipm.and.ispx==1 ) then
              rmatt(:,:,1) = rmat(:,1:nwhis,1)
            elseif( chipm.and.ispx==2 ) then
              do iw= 1,nwhis
                rmatt(:,iw,1) = -rmat(:,-iw,1)
              enddo
            else
              do iw= 1,nwhis
                rmatt(:,iw,1) = rmat(:,iw,1) - rmat(:,-iw,1)
              enddo
            endif
            deallocate(rmat,rrr)
          else ! npm==2 case -------------------------------------------------
            allocate( rmatt(-nw_w:nw_w,nwhis,npm), rrr(-nwhis:nwhis))
            rmatt = 0d0
            do it  =  -nw_w,nw_w
              if(it<0) then
                zz = -freqr(-it) !his_C(it)
              else
                zz = freqr(it) !his_C(it)
              endif
              call hilbertmat(zz, nwhis,his_l,his_c,his_r, rrr)
              rmatt(it,:,1) =  dreal(rrr(1:nwhis))
              rmatt(it,:,2) = -dreal(rrr(-1:-nwhis:-1))
            enddo ;   if(debug) write(6,*) 'dpsion5: RRR2 55555555555'
            deallocate(rrr)
          endif
          rmatt = rmatt/pi ; if(debug) write(6,*)'dpsion5: RRR 6666'

!! takao remove if(nolfc) block here.
c          write(6,*) " --- realomega dgemm--- "


!! WARN! I think npm==2.and.chipm does not make sense. apr2012.
!!
          if(npm==2.and.chipm)
Cstop2rx 2013.08.09 kino     &      stop 'x0kf_v4h:npm==2.and.chipm is not meaningful probably'
     &      call rx( .and.'x0kf_v4h:npm==2chipm is not meaningful probably')


!! Note rcxq is negative now (converted at the top of this routine !!!
          if(     chipm .and. ispx==2 ) then
            !nothing here
            !Since the range of zxq is nw_i=0, we have no area to store negative energy part of chipm.
          elseif( chipm              ) then
            call zaxpy( nmbas1*nmbas2*nw_w, img, rcxq, 1, zxq(1,1,1), 1)
          else
            zxq = 0d0    ! not accumlating case.
            call zaxpy( nmbas1*nmbas2*nw_w, img, rcxq(1,1,1,1), 1, zxq(1,1,1), 1)
          endif

          if(npm==2) then
            do iw=1,nw_w
              call zaxpy( nmbas1*nmbas2, img, rcxq(1,1,iw,2),1, zxq(:,:,-iw),1)
            enddo
          endif

          if(npm==1) then
            call dgemm('n','t',  2*nmbas1*nmbas2, nw_w+1, nwhis, 1d0,
     &              rcxq, 2*nmbas1*nmbas2,  rmatt, nw_w+1,
     &               1d0, zxq, 2*nmbas1*nmbas2 )
          elseif(npm==2) then
            call dgemm('n','t',  2*nmbas1*nmbas2,   npm*nw_w+1, nwhis, 1d0,
     &              rcxq(1,1,1,1), 2*nmbas1*nmbas2, rmatt(:,:,1), npm*nw_w+1,
     &               1d0, zxq, 2*nmbas1*nmbas2 )
            call dgemm('n','t',  2*nmbas1*nmbas2,   npm*nw_w+1, nwhis, 1d0,
     &              rcxq(1,1,1,2), 2*nmbas1*nmbas2, rmatt(:,:,2), npm*nw_w+1,
     &               1d0, zxq, 2*nmbas1*nmbas2 )
          else
Cstop2rx 2013.08.09 kino            stop 'dpsion5: npm=1 or 2'
            call rx( 'dpsion5: npm=1 or 2')
          endif
          deallocate(rmatt)
        endif

!! === imagomega case      imatt(niwt -->niwt,npm may2005 ===
        if(imagomega) then
          allocate( rrr(-nwhis:nwhis))
          if(npm==1) then
            allocate( rmati(niwt,-nwhis:nwhis,npm))
            rmati= 0d0
          else
            allocate( rmatic(niwt,-nwhis:nwhis,npm))
            rmatic = 0d0
          endif ;   if(debug) write(6,*) 'dpsion5: III 111111155555555555'
          do it =  1,niwt
            zz = img*freqi(it) !his_C(it)
            call hilbertmat(zz, nwhis,his_l,his_c,his_r, rrr) !Im(zz)>0
            if(npm==1) then
              rmati(it,:,1) = dreal(rrr)
            else
              rmatic(it,:,1) = rrr
            endif
          enddo ;   if(debug) write(6,*) 'dpsion5: III 55555555555'
!! ==== npm=1 case ====
          if(npm==1) then
            allocate( imatt(niwt, nwhis,npm) )
            do iw= 1,nwhis
              imatt(:,iw,1) = rmati(:,iw,1) - rmati(:,-iw,1)
            enddo
            deallocate(rmati,rrr)
            imatt = imatt/pi; if(debug) write(6,*) 'dpsion5: III  '
            call dgemm('n','t',  2*nmbas1*nmbas2, niwt, nwhis, 1d0,
     &              rcxq, 2*nmbas1*nmbas2, imatt, niwt,
     &               0d0, zxqi, 2*nmbas1*nmbas2 )
            deallocate(imatt)
!! ==== npm=2 case ====
          else
            allocate( imattc(niwt, nwhis,npm) )
            do iw= 1,nwhis
              imattc(:,iw,1) =   rmatic(:, iw,1)
              imattc(:,iw,2) = - rmatic(:,-iw,1)
            enddo
            deallocate(rmatic,rrr)
            imattc = imattc/pi; if(debug) write(6,*) 'dpsion5: IIIc '
            call zgemm('n','t',  nmbas1*nmbas2, niwt, nwhis, 1d0,
     &              rcxq(1,1,1,1), nmbas1*nmbas2, imattc(1,1,1), niwt,
     &              0d0, zxqi,    nmbas1*nmbas2 )
            call zgemm('n','t',  nmbas1*nmbas2, niwt, nwhis, 1d0,
     &              rcxq(1,1,1,2), nmbas1*nmbas2, imattc(1,1,2), niwt,
     &              1d0, zxqi,    nmbas1*nmbas2 )
            deallocate(imattc)
          endif
        endif
        deallocate(his_l,his_c,his_r)
        write(6,'("         end dpsion5 ",$)')
        call cputid(0)
        end
        logical function checkbelong(qin, qall, nq,ieibz) !ieibz is also returned
        integer:: nq,ieibz
        real(8):: qin(3), qall(3,nq),tolq=1d-8
        checkbelong=.false.
        do i=1,nq
           if(sum(abs(qin-qall(:,i)))<tolq) then
              ieibz=i
              checkbelong=.true.
              return
           endif
        enddo
        end


!!---------------------------------------------------------------------------
```

```
      subroutine hilbertmat (zz,nwhis, his_L,his_C,his_R, rmat)
C- Martix for hilbert transformation, rmat.
Cr  zz is real--->  no img*delta function part
Cr   zz is complex (and Im(zz)>0) : includes all contribution when Im(zz)>eps
Co  rmat(-nwhis:nwhis) : rmat(0) is not meaningful.
Ci i-th Histgram bin on real axis are given by [his_L, his_R]. center is his_C.
Cr f(zz) = \int_-x(nwhis)^x(nwhis) f(x)/(zz-x)
Cr        = \sum_{i/=0} rmat(i)*f(i)
Cr      ,where f(i) is the average value at i-th bin.
C!!! 23May2006 I think
C!!! rmat is -------------
C!!! f(zz) = - \int_-x(nwhis)^x(nwhis) f(x)/(zz-x)
C!!!        = - \sum_{i/=0} rmat(i)*f(i)
C I forgot minus sign in the previous note.
C------------------------------
      implicit none
      integer(4):: iw,nwhis
      complex(8) ::zz,imgepsz
      real(8)    :: his_l(-nwhis:nwhis),his_c(-nwhis:nwhis),his_r(-nwhis:nwhis)
      complex(8) :: rr_fac(-nwhis:nwhis),rl_fac(-nwhis:nwhis),img=(0d0,1d0)
      real(8)::  eps=1d-8, epsz=1d-13,delta_r,delta_l,ddr,ddl
      complex(8):: domega_c,domega_r,domega_l
      complex(8) ::  rmat(-nwhis:nwhis)
      imgepsz =img*epsz
      do iw = -nwhis, nwhis
        if(iw==0) cycle
        domega_r = zz - his_r(iw) + imgepsz
        domega_c = zz - his_c(iw) + imgepsz
        domega_l = zz - his_l(iw) + imgepsz
        if( abs(domega_c)<eps .or. abs(domega_r)<eps ) then
          rr_fac(iw) = 0d0
        else
! rr_fac(his_C(is)) = \int^{his_R}_{his_C} d omega' /(his_C(is) -omega')
c            rr_fac(iw) = log( abs((domega_r/domega_c)) )
          rr_fac(iw) = log( domega_r/domega_c )
        endif
        if( abs(domega_c)<eps .or. abs(domega_l)<eps ) then
          rl_fac(iw) = 0d0
        else
! rl_fac(his_C(is)) = \int^{his_C}^{his_L} d omega' /(his_C(is) -omega')
c            rl_fac(iw) = log( abs((domega_c/domega_l)) )
          rl_fac(iw) = log( domega_c/domega_l)
        endif
      enddo
      rmat=0d0
      do iw = -nwhis, nwhis !symmetric version. iw=0 is meaningless
        if(iw==0) cycle
c          if(debug) print *,' it iw=',it, iw
        domega_c = zz - his_c(iw)
        if(iw==  nwhis) then
          delta_r = his_r(iw)   - his_c(iw)
        elseif(iw== -1) then
          delta_r =   0d0       - his_c(iw)
        else
          delta_r = his_c(iw+1) - his_c(iw)
        endif
!         if(debug) print *,' it iw RRR1'
        if(iw== -nwhis) then
          delta_l = his_c(iw)  - his_l(iw)
        elseif(iw==  1) then
          delta_l = his_c(iw)  - 0d0
        else
          delta_l = his_c(iw)  - his_c(iw-1)
        endif
!         if(debug) print *,' it iw RRR2'
!          ddr = (his_R(iw)-his_C(iw))/delta_r
!          ddl = (his_C(iw)-his_L(iw))/delta_l
        rmat(iw)  = rmat(iw  ) + rr_fac(iw)*( 1d0-domega_c/delta_r) !+ ddr
        if(iw/=nwhis.and.iw/=-1) then
          rmat(iw+1) = rmat(iw+1) + rr_fac(iw)*domega_c/delta_r     !- ddr
        endif
        rmat(iw)  = rmat(iw) + rl_fac(iw)*( 1d0+domega_c/delta_l)   !- ddl
        if(iw/=-nwhis.and. iw/=1) then
          rmat(iw-1) = rmat(iw-1) - rl_fac(iw)*domega_c/delta_l     !+ ddl
        endif
ccccccccccccccccccccccccc
c no-derivarive test
c          rmat(iw)  =  rr_fac(iw) + rl_fac(iw)
ccccccccccccccccccccccccc
      enddo
      end

c$$$      subroutine reducezmel(aold, ngbo,ngb,nx,
c$$$     i       io,   in,   nmat, pmat,
c$$$     i     io_q, in_q, nmat_q, pmat_q,
c$$$     o      anew)
c$$$c   For given q+G basis, we augment the basis within MT.
```

```
c$$$c   For given atom and l  prod and prodd at MT boundary (reserved in PPBRD
c$$$      integer(4):: nmat,io(nmat),in(nmat),nmat_q,io_q(nmat),in_q(nmat)
c$$$      complex(8):: aold(ngbo,nx), anew(ngb,nx),pmat(nmat) ,pmat_q(nmat)
c$$$      anew=0d0
c$$$      do ix=1,nmat
c$$$          anew(in(ix), :)
c$$$     &  = anew(in(ix), :)   +  pmat(ix) * aold(io(ix), :)
c$$$      enddo
c$$$      do ix=1,nmat_q
c$$$          anew(in_q(ix), :)
c$$$     &  = anew(in_q(ix), :)  + dconjg(pmat_q(ix)) * aold(io_q(ix), :)
c$$$      enddo
c$$$      end

      real(8) function wcutef(e,ecut,ecuts)
      real(8):: e,ecut,ecuts
c      wcutef = 1d0/( exp((e-ecut)/ecuts)+ 1d0)
      wcutef = exp( -(e/ecut)**2 ) ! ecuts is not used in this case
      end
```

4.27 main/hbasfp0.m.F File Reference

Functions/Subroutines

- program hbasfp0_v2

4.27.1 Function/Subroutine Documentation

4.27.1.1 program hbasfp0_v2 ( )

Definition at line 1 of file hbasfp0.m.F.

Here is the call graph for this function:

4.28 hbasfp0.m.F

```
      program hbasfp0_v2
c-- Generates orthonormal optimal product basis and required radial integrals in each MT.
c input files
c  GWinput : input data for GW
c  LMTO    : fundamental data for crystal
c  PHICV   : radial functions Valence and Core
c
c output files
c  BASFP//ibas :: product basis for each ibas
c  PPBRD_V2_//ibas :: radial <ppb> integrals. Note indexing of ppbrd
c
c The main part of this routine is in the subroutine basnfp_v2
      use m_rgwinf_v3,only:rgwinf_v3,
     & alat,nclass,natom,nspin,nl,nnv,nnc,nrx, cutbase,lcutmx,nindxc,
     & nindxv,occv,unoccv,occc,unoccc,iclass
      use m_keyvalue,only: getkeyvalue
      use m_anf,only: ibasf,laf,anfcond !may2015takao
      implicit none
      real(8):: qbas(3,3),ginv(3,3)
      integer(4)::
     l ifphiv(2),ifphic(2), iphiv(2),iphivd(2),iphic(2),iphi(2),iphidot(2),
     l ifev(2),ifevf(2),ibas,ibas1,ic,icx,ifaln,ifinin,iflmto,ifphi,
     l ii,ir,irad,isp,ix,lmx,lmx2,n,nbas,ncoremx,l,nn,icore,ifianf,nphi,nradmx,nsp,iopen,maxnn,iclose
      integer(4),allocatable:: lcutmxa(:)
      character(12) :: aaa
      integer(4),allocatable::nrofi(:), nocc(:,:),nunocc(:,:),nindx(:,:)
      logical :: ptest=.false. !See ptest in hvccfp0.f
      real(8),allocatable :: bb(:),zz(:), phic(:,:)
      integer(4) :: ndat
      integer(4),allocatable:: ncindx(:,:),lcindx(:,:),
     &           nrad(:), nindx_r(:,:), lindx_r(:,:),
     &           nc_max(:,:),ncore(:)
      real(8),allocatable:: phitoto(:,:,:,:,:), aa(:),rr(:,:),phitotr(:,:,:,:,:)
      character*11 :: ffaln
      integer(4)::incwfin,ret
      integer(4),allocatable:: idid(:)
      logical :: checkdid ,anfexist
      integer(4):: iread, idummy
c-----------------------------------------------------------------------
      ifinin=-99999 !dummy
      write(6,'(a)') ' --- Input normal(=0); coremode(=3);'//
     & ' ptest(=4); Excore(=5); for core-valence Ex(=6);'//
     & ' val-val Ex(7);  normal+<rho_spin|B> (8); version(-9999) ?'
      call readin5(ix,iread,idummy)
      call headver('hbasfp0',ix)
      if(ix==3) then
        write(6,*)'  ### coremode; Product basis for SEXcore ### '
        incwfin = -2
      elseif(ix==0) then
        write(6,*)'  ### usual mode use occ and unocc for core ### '
        incwfin = 0
      elseif(ix==4) then
        write(6,*)
     & ' ### ptest mode. now special for Q0P. GWIN_V2 is neglected ### '
        write(6,*) '  See basnfp.f of ptest section.'
        incwfin = 0
      elseif(ix==5) then
        write(6,*)
     &  '  ### calculate core exchange energy ### ix==5'
        incwfin = 0
      elseif(ix==6) then
        write(6,*)
     &  '  ### calculate p-basis for core-valence Ex ix==6'
        write(6,*) ' occ=1:unocc=0 for all core'
        incwfin = -3
      elseif(ix==7) then
        write(6,*)
     &  '  ### calculate p-basis for val-val Ex ix==7'
        write(6,*) ' occ=0:unocc=0 for all core'
        incwfin = -4
      elseif(ix==8) then !May2005
        write(6,"('  ### usual mode use occ and unocc for core',
     &            ' and <rho_spin |B(I)> ### ')")
        incwfin = 0
      else
        write(6,*)' hbasfp: input is out of range'
        call rx( ' hbasfp: input is out of range')
      endif

!! read data in m_rgwinf_v3
!! Output are allocated and data are setted as above.
      iflmto = iopen('LMTO',1,0,0)
      if (iflmto <= 0) call rx( 'unit file for LMTO <= 0')
      call rgwinf_v3(iflmto,ifinin,incwfin) ! readin inputs. See use use m_rgwinf_v3,only: ... at the
 begining.
      iflmto= iclose('LMTO')
      nsp=nspin
      write(6,*)'end of rgwinf'
!! readin lcutmxa ------------
      call getkeyvalue("GWinput","<PRODUCT_BASIS>",unit=ifinin,status=ret)
      allocate(lcutmxa(1:natom))
      do
        read(ifinin,*,err=980) aaa
        if(aaa=='lcutmx(atom)') then
          read(ifinin,*) lcutmxa(1:natom)
c          write(6, '(" lcutmxa=",20i3)' ) lcutmxa(1:natom)
          goto 990
        endif
      enddo
  980 continue
      lcutmxa=lcutmx
  990 continue
      close(ifinin)

      if(ix==8) then
         write(6,*)' Enfoece lcutmx=0 for all atoms'
         lcutmxa=0
      endif

      write(6,"(' lcutmxa=',$)")
      write(6,'(20i3)') lcutmxa(1:natom)
      lmx        = 2*(nl-1)
      lmx2       = (lmx+1)**2
      nn         = maxnn(nindxv,nindxc,nl,nclass)
      nphi       = nrx*nl*nn*nclass


c -optimal orthonormal product basis
c> reindex nocc,nunocc,nindx
! For valence  from GWIN_V2
! occv   : occ    switch
! unoccv : unocc  switch
! nindexv: n index
!-----------------------------
! For core  from GWIN_V2
! occc   : occ  switch
! unoccc : unocc switch
! nindexc: n index
!-----------------------------
! For valence+core
! nocc
! nunocc
! nindx
      allocate( nocc(nl*nn,nclass), nunocc(nl*nn,nclass), nindx(nl,nclass) )
      call reindx(occv,unoccv,nindxv,   occc,unoccc,nindxc,
     d              nl,nn,nnv,nnc,nclass,
     o              nocc,nunocc,nindx)
      write(6,*)' --- end of reindx ---'

c-----------
c read PHIVC  and reserve it to phitot
c----------
      ifphi  = iopen('PHIVC', 0,-1,0)     ! PHIV+PHIC augmentation wave and core
      read(ifphi) nbas, nradmx, ncoremx
      allocate(  ncindx(ncoremx,nbas),
     &           lcindx(ncoremx,nbas),
     &           nrad(nbas),
     &           nindx_r(1:nradmx,1:nbas),
     &           lindx_r(1:nradmx,1:nbas),
     &        aa(nbas),bb(nbas),zz(nbas), rr(nrx,nbas), nrofi(nbas) ,
     &        phitoto(nrx,0:nl-1,nn,nbas,nsp),
     &        phitotr(nrx,0:nl-1,nn,nbas,nsp),
     &        nc_max(0:nl-1,nbas),ncore(nbas) )
      read(ifphi) nrad(1:nbas)
      read(ifphi) nindx_r(1:nradmx,1:nbas),lindx_r(1:nradmx,1:nbas)
      nc_max=0
      do ibas=1,nbas
        write(6,*)' --- read PHIVC of ibas=',ibas
        ic = ibas
        read(ifphi) ncore(ic), ncoremx                            !core
        read(ifphi) ncindx(1:ncoremx,ibas),lcindx(1:ncoremx,ibas) !core
        read(ifphi) icx,zz(ic),nrofi(ic),aa(ic),bb(ic)
        if(ic/=icx) then
          write(6,*) 'ic icx=',ic,icx
          call rx( 'hbasfp0: ic/=icx')
        endif
        read(ifphi) rr(1:nrofi(ic),ic)
        do isp = 1, nsp
          write(6,*)'          ---  isp nrad ncore(ic)=',isp, nrad(ic),ncore(ic)
          do icore = 1, ncore(ic)
            l =  lcindx(icore,ic)
            n =  ncindx(icore,ic)
            read(ifphi) phitoto(1:nrofi(ic),l,n, ic,isp)   !core orthogonal
            phitotr(1:nrofi(ic),l,n, ic,isp)=              !core raw= core orthgonal
     &      phitoto(1:nrofi(ic),l,n, ic,isp)               !
            if(n>nc_max(l,ic)) nc_max(l,ic)=n
          enddo
          do irad = 1, nrad(ic)
            l = lindx_r(irad,ic)
            n = nindx_r(irad,ic) + nc_max(l,ic)
            read(ifphi) phitoto(1:nrofi(ic),l,n, ic,isp) !valence orthogonal
            read(ifphi) phitotr(1:nrofi(ic),l,n, ic,isp) !valence raw
          enddo
        enddo
      enddo
c-----------

!! check write
      ffaln ='PHIV.chk'
      ifaln = iopen(ffaln,1,-1,0)
      do ibas = 1,nbas
        ic = ibas
        do irad = 1, nrad(ic)
          l = lindx_r(irad,ic)
          n = nindx_r(irad,ic) + nc_max(l,ic)
          write(ifaln,"(a,5i5)")'------- ibas l n =',ibas,l,n
          do ir=1,nrofi(ic)
            write(ifaln,"(3d24.15)")rr(ir,ic), phitotr(ir,l,n,ic,1:nsp)
          enddo
        enddo
      enddo
      ifaln = iclose(ffaln)

!!  excore mode ---------
      if(ix==5 ) then
        call excore(nrx,nl,nnc,nclass,nsp,natom,
     &   phitotr(1:nrx,0:nl-1,1:nnc,1:nclass,1:nsp),   !core
     &   nindxc,iclass,
     &    aa,bb,nrofi,rr)
        goto 998
      endif

!! antiferro or not.
!! For AF case, we have laf=.true. and we have data set for 'call anfsig', stored in m_anf.
      call anfcond()
      if(laf) then
!!      Check iclass =ibas ; CLASS file contains true classs information.
c        allocate(idid(natom))
        write(6,*) '--- Antiferro mode --- '
        do ibas=1,natom
          if(iclass(ibas)/=ibas) call rx( ' iclass(ibas)/=ibas: ')
        enddo
        ii=0
        do ic=1,nclass
          ibas=ic
          if( ibasf(ibas)>0 ) then
            phitotr(:,:,:,ibasf(ibas), :)=phitotr(:,:,:,ibas, :)
            write(6,"(a,2i4)")
     &       '  radial functions: phi(ibasf)=phi(ibas): ibasf ibas=',ibasf(ibas),ibas
          endif
        enddo
c        if( sum (idid(1:ii)) /= natom*(natom+1)/2)
c     &    call rx( 'hbasfp0:sum (idid(1:ii)) /= n(n+1)/2')
c        write(6,*)' end of anf section...'
      endif

!! override cutbase to make epsPP_lmfh safer. may2013takao
      if(ix==4) then
         write(6,*)' !!! set tolerance for PB to be 1d-6 ---'
         cutbase=1d-6
      endif

      do ic = 1,nclass
        call basnfp_v2(nocc(1,ic),nunocc(1,ic),nindx(1,ic), ! Product Basis functions
     &    nl,nn,nrx, nrofi(ic),rr(1,ic),aa(ic),bb(ic),ic,
     &    phitoto,phitotr,nsp,nclass,
     i    cutbase, lcutmxa(ic),ix,iread,alat
     i  ,nc_max(0,ic) )
      end do
      if(ix==0) call rx0( ' OK! hbasfp0 ix=0 normal mode ')
      if(ix==3) call rx0( ' OK! hbasfp0 ix=3 core mode ')
      if(ix==4) call rx0( ' OK! hbasfp0 ix=4 ptest mode  ')
      if(ix==6) call rx0( ' OK! hbasfp0 ix=6 Exx core-val mode  ')
      if(ix==7) call rx0( ' OK! hbasfp0 ix=7 Exx val-val mode  ')
      if(ix==8) call rx0( ' OK! hbasfp0 ix=8 normal(ix==0) + <B|spin den>. Enforce lcutmx=0.')
 998  if(ix==5) call rx0( ' OK! hbasfp0 ix=5 ex core mode  ')
      end


c      logical function checkdid (idid,ii, ibas)
c      integer(4):: idid(ii),ix
c      checkdid=.true.
c      do ix=1,ii
c        if(idid(ix)==ibas) return
c      enddo
c      checkdid=.false.
c      end
```

4.29 main/hsfp0.sc.m.F File Reference

Functions/Subroutines

- program hsfp0_sc
- subroutine zsecsym (zsec, ntq, nq, nband, nbandmx, nspinmx, eibzsym, ngrp, tiii, q, is)

4.29.1 Function/Subroutine Documentation

4.29.1.1 program hsfp0_sc ( )

Definition at line 1 of file hsfp0.sc.m.F.

Here is the call graph for this function:

4.29.1.2 subroutine zsecsym (complex(8), dimension(ntq,ntq,nq), intent(inout) *zsec*, integer, intent(in) *ntq*, integer, intent(in) *nq*, integer, intent(in) *nband*, integer, dimension(nq,nspinmx), intent(in) *nbandmx*, integer, intent(in) *nspinmx*, integer, dimension(ngrp,-1:1,nq), intent(in) *eibzsym*, integer, intent(in) *ngrp*, logical, intent(in) *tiii*, real(8), dimension(3,nq), intent(in) *q*, integer, intent(in) *is*)

Definition at line 1228 of file hsfp0.sc.m.F.

Here is the caller graph for this function:

4.30 hsfp0.sc.m.F

```
00001       program hsfp0_sc
00002 !> Calculates the self-energy \Sigma in GW approximation,
00003 !!  including Off-diagonal components.
00004 !!  (hsfp0.F is for diagonal part only).
00005 !! ----------------------------------------
00006 !!    SEx(q,itp,itpp) = <psi(q,itp) |SEx| psi(q,itpp)>
00007 !!    SEc(q,itp,itpp) = <psi(q,itp) |SEc| psi(q,itpp)>
00008 !!    Here SEc(r,r';w) = (i/2pi) < [w'=-inf,inf] G(r,r';w+w') Wc(r,r';w') >
00009 !!
00010 !! ----------------------------------------
00011 !! See papers;
00012 !! [1]T. Kotani and M. van Schilfgaarde, Quasiparticle self-consistent GW method:
00013 !!     A basis for the independent-particle approximation, Phys. Rev. B, vol. 76, no. 16,
00014 !!     p. 165106[24pages], Oct. 2007.
00015 !! [2]T. Kotani, Quasiparticle Self-Consistent GW Method Based on the Augmented Plane-Wave
00016 !!     and Muffin-Tin Orbital Method, J. Phys. Soc. Jpn., vol. 83, no. 9, p. 094711 [11 Pages], Sep. 2014.
00017 !!
00018 !! EIBZ symmetrization;
00019 !! See [3] C. Friedrich, S. Bl?gel, and A. Schindlmayr,
00020 !!   Efficient implementation of the GW approximation within the all-electron FLAPW method,
00021 !!   Physical Review B, vol. 81, no. 12, Mar. 2010.
00022 !!
00023 !! Usage: This routine is called from a script for QSGW, ecalj/fpgw/exec/gwsc.
00024 !! which calls is as "echo 2|../exec/hsfp0_sc >lsc" when mode=2 (three times in the gwsc).
00025 !!
00026 !! mode= 1: exchange    mode SEx, the exchange part of the self-energy
00027 !! mode= 2: correlation mode SEc, the correlated part of the self-energy
00028 !! mode= 3: core exchange mode SEXcore
00029 !! xxx mode= 4: plot spectrum function ---See manual ---> this is performed by echo 4|hsfp0
00030 !!
00031 !! iSigMode parameter which determines approximation for self-energy is given by GWinput file as iSigMode.
00032 !!    iSigMode==0 SE_nn'(ef)+image integr:delta_nn'(SE_nn(e_n)-SE_nn(ef))
00033 !!    iSigMode==1 SE_nn'(ef)+delta_nn'(SE_nn(e_n)-SE_nn(ef))
00034 !!       xxx not support this mode now ... iSigMode==2 SE_nn'((e_n+e_n')/2)
00035 !!    iSigMode==3 (SE_nn'(e_n)+SE_nn'(e_n'))/2  <--- this is mainly used
00036 !!     iSigMode==5 delta_nn' SE_nn(e_n)
00037 !!     Output file contain hermitean part of SE for energies to be real
00038 !!    (for example, hermitean conjunction of SE_nn'(e_n) means SE_n'n(e_n')^* )
00039 !!
00040 !!     History: We learned so much from LMTO-ASA codeds developed by F.Aryasetiawan.
00041 !! ----------------------------------------
00042       use m_readefermi,only: readefermi,ef
00043       use m_readqg,only: readqg,readngmx
00044       use m_readeigen,only: init_readeigen,init_readeigen2,readeval,lowesteval
00045       use m_read_bzdata,only: read_bzdata,
00046      & nqbz,nqibz,nqbzw,nteti,ntetf
00047      & ,n1,n2,n3,qbas,ginv,qbz,wbz,qibz,wibz,qbzw,idtetf,ib1bz,idteti
00048      & ,nstar,irk,nstbz,ngrp2=>ngrp
00049       use m_genallcf_v3,only: genallcf_v3,
00050      & nclass,natom,nspin,nl,nn, ngrp,
00051      & nlmto,nlnmx, nctot,niw,!nw_input=>nw,
00052      & alat, delta,deltaw,esmr,symgrp,clabl,iclass, !diw,dw,
     & invg, il,in,im,nlnm,
     & plat, pos,z,ecore, symgg, konf,nlnx, iantiferro
      use m_keyvalue,only: getkeyvalue

!! Base data to generate matrix elements zmel*. Used in "call get_zmelt".
      use m_rdpp,only: rdpp,    !"call rdpp" generate following data.
     & nblocha,lx,nx,ppbrd,mdimx,nbloch,cgr
!! Generate matrix element for "call get_zmelt".
      use m_zmel,only:  ! folloiwng data set are stored in this module in the main routin,
                        ! and used when call get_zmelt, get_zmelt2.
     & nband,itq,ngcmx,ngpmx,
     & miat,tiat,shtvg, ntq, ppbir
!! antiferro condition. only laf is used, after 'call anfcond()'
      use m_anf,only: anfcond,
     &   laf
!! subroutine only
      use m_sxcfsc,only: sxcf_fal3_scz
!! MPI
      use m_mpi,only:
     &   mpi__initialize,mpi__real8send,mpi__real8recv,mpi__send_iv,mpi__recv_iv,mpi__sxcf_rankdivider,
     &   mpi__finalize,mpi__root,mpi__broadcast,mpi__rank,mpi__size,mpi__allreducesum,
     &   mpi__consoleout,
     &   mpi__barrier

      implicit none
!! ---------------------------------------------------------------------------------
!!     real(8),parameter :: ua  = 1d0 ! constant in w(0)exp(-ua^2*w'^2) to take care of peak around w'=0
c----------------------------------------
!!!   test switches to calculate the self-energy based on an another separation of \Sigma.
!!!     \Sigma = \Sigma_{sx} + \Sigma_{coh} + \Sigma_{img axis} + \Sigma_{pole} by Hedin PR(1965)A785
!!!     I found COH term has inevitably poor accuracy.
      logical ::tetra, tetra_hsfp0,
     & screen = .false.,        ! \Sigma_{sx} for mode 1 and
! \Sigma_{img axis} + \Sigma_{pole} for mode 2
     & cohtest= .false.         ! \Sigma_{coh}. mode swich is not required.
c     &  , tetra  = .false.  ! test switch for tetrahedron method test.
c     ! tetra=T is only effective for exchange=T case.
c     ! Tetrahedron mehod for correlation is a bit
! difficult and I gave up for a while.
! If you want to calculate with tetra=T for exchange, you
! have to uncomment tetra related part in
! sxcf.f, and a part calling sxcf in this routine. Note wtet wtetef!
! They sometimes cause array destruction if you run tetra=T without comment them.

c      real(8) :: shtw
      integer::
     & ixc,iopen,ifhbed, nprecb,mrecb,mrece,nlmtot,nqbzt, !nband,
     & ibas,ibasx,nxx,ifqpnt,ifwd,
     & nprecx,mrecl,nblochpmx2,nwp,niwt, nqnum,nblochpmx, !mdimx,nbloch
     & noccxv,maxocc,noccx,ifvcfpout,iqall,iaf, !ntq, !ifrcw,ifrcwi,
     & i,k,nspinmx, nq,is,ip,iq,idxk,ifoutsex,iclose,nq0i,ig,
     & mxkp,nqibzxx,ntet,nene,iqi, ix,iw,
     & nlnx4,invr,ivsum, ifoutsec, !niwx,
     & ifsec(2)
     & ,ifxc(2),ifsex(2), ifphiv(2),ifphic(2),ifec,ifexsp(2),
     & ifsex2(2),ifsec2(2),     !out S_nn'
     & ifsecomg(2),ndble=8
      real(8) :: pi,tpia,vol,voltot,rs,alpha,
     & qfermi,efx,valn,efnew,edummy,efz,qm,xsex,egex,edummyd(1),
     & zfac1,zfac2,dscdw1,dscdw2,dscdw,zfac
      logical :: lqall,laff,lntq
      real(8),allocatable    :: q(:,:)

      integer,allocatable ::
     & ngvecp(:,:), ngvecc(:,:),iqib(:),
     & kount(:,:)
      real(8),allocatable:: vxcfp(:,:,:),
     & wqt(:),q0i(:,:),
     & eqt(:),
     & ppbrdx(:,:,:,:,:,:,:),
     & eq(:),
     & eqx(:,:,:),eqx0(:,:,:),ekc(:),coh(:,:)
      complex(8),allocatable:: zsec(:,:,:)
c
      logical :: legas
      real(8) :: rydberg,hartree
      real(8):: qreal(3), ntot,nocctotg2,tripl,xxx(3,3)
      logical ::nocore

c     space group information
      integer,allocatable :: iclasst(:), invgx(:)
c     tetra
      real(8),allocatable :: qz(:,:),qbzxx(:),wbzxx(:),wtet(:,:,:,:),
     & eband(:,:,:), ene(:)
      integer,allocatable ::idtetx(:,:),idtet(:,:),ipq(:)
     & ,iene(:,:,:),ibzx(:)
      integer ::ib,iqx,igp,iii,ivsumxxx,isx,iflegas, iqpntnum
```

```
c
      real(8),allocatable   :: eex1(:,:,:),exsp1(:,:,:),qqex1(:,:,:,:)
      integer,allocatable:: nspex(:,:),ieord(:),itex1(:,:,:)
      real(8)    :: qqex(1:3), eex,exsp,eee, exwgt,deltax0
      integer :: itmx,ipex,itpex,itex,nspexmx,nnex,isig,iex,ifexspx
     & ,ifexspxx ,ifefsm, nq0ix,ifemesh,nz
      character(3)  :: charnum3
      character(12) :: filenameex
      logical :: exspwrite=.false.
      character*8 xt

      integer :: isigmode,ifinin ,idummy

      real(8),allocatable:: omega(:)
      real(8)   :: ebmx(2)
      integer:: nbmx(2)

      real(8):: volwgt

      integer:: incwfin
      real(8),allocatable::freqx(:),freqw(:),wwx(:)

      integer::  ngpn1,mrecg,ngcn1
      real(8)   :: wgtq0p,quu(3)

      character(2):: soflag
      integer:: ifianf

      integer:: ifpomat,nkpo,nnmx,nomx,ikpo,no
      real(8):: q_r(3)
      real(8),allocatable:: qrr(:,:)
      integer,allocatable:: nnr(:),nor(:)

      logical :: allq0i
      integer:: nw_i
      logical:: exonly
      real(8):: wex
!! newaniso mode
c      logical:: newaniso
      real(8),allocatable:: vcousq(:),dmlx(:,:),epinvq0i(:,:),wklm(:),vcoud(:)
      complex(8),allocatable:: zcousq(:,:)
      integer:: ifvcoud,lxklm,ifidmlx

      integer,allocatable:: irkip_all(:,:,:,:),irkip(:,:,:,:)

      integer,allocatable:: nrkip_all(:,:,:,:),nrkip(:,:,:,:)
      integer,allocatable:: neibz(:),nwgt(:,:),ngrpt(:),igx(:,:,:),igxt(:,:,:),eibzsym(:,:,:)
      integer:: iqxend,iqxini
      integer:: l2nl,igrp,kx,kr
      logical :: iprintx,tiii,timereversal, eibz4sig,tiiiout

      logical :: selectqp=.false.,diagonly=.false.
      integer:: ret,dest,nnn
      character(128) :: ixcc
      real(8):: eftrue,esmref   !jan2013
      real(4):: time_red1,time_red2
      integer:: timevalues(8) ,ibz

      integer::irot !,nn_
      real(8),allocatable:: wgt0(:,:)
      logical:: exchange
      real(8):: exx
      real(8),allocatable:: freq_r(:)
      integer:: ififr,ifile_handle,nwxx,ifih

      integer:: verbose,iband,isp,iqq
      integer,allocatable:: nbandmx(:,:)

      integer:: ificlass,ifiq0p,ntqxx,nq_r,nband_r
      logical:: hermitianw
      integer:: nw
      real(8)::dwdummy
c---------------------------------------
      call mpi__initialize()    ! MIZUHO-IR
      call date_and_time(values=timevalues)
      write(6,'(a,9i5)')'dateandtime1=',mpi__rank,timevalues(1:8)
!TIME0_0000
!TIME0_0010
      hartree=2d0*rydberg()
      hermitianw=.true.
      if(cohtest) then          !currently not used (may need fixing if necessary)
        screen = .true.
        ixc = 2; nz=0
        open(671,file='COH')
      elseif(mpi__root) then
        write(6,*) ' --- Choose modes below ------------'
        write(6,*) '  Sx(1) Sc(2) ScoreX(3) '
```

```
          write(6,*) '  [option --- (+ QPNT.{number} ?)] '
          write(6,*) ' Add 1000, eg, 1001 is diagonal only mode for one-shot Z=1'
          write(6,*) ' --- Put number above ! ------------'
          call readin5(ixc,nz,idummy)
          write(6,*) ixc
        endif
        call mpi__broadcast(ixc)
        call mpi__broadcast(nz)
        if(mpi__root) call headver('hsfp0_sc',ixc)
        write(ixcc,"('.mode=',i4.4)")ixc

        if(ixc>1000) then         !selected QP
          ixc=mod(ixc,1000)
          selectqp=.true.
          diagonly=.true.
          hermitianw=.false.
          write(6,*) "--- Diagonal-only mode. jobsw=5; see description at the top of sxcf_fal2.sc.F."
          write(6,*) "--- This is the same as one-shot calculaiton with iSigMode5 in GWinput."
        endif

        call mpi__consoleout('hsfp0_sc'//trim(ixcc))
        write(6,*) ' ixc nz=',ixc, nz
        if(ixc==0) call rx( ' --- ixc=0 --- Choose computational mode!')

!! ===  readin BZDATA. See gwsrc/rwbzdata.f ===
!! See use m_read_bzdata,only: at the top of this routine
        call read_bzdata()
        write(6,*)' nqbz =',nqbz
        write(6,*)' nqibz ngrp=',nqibz,ngrp2
        call pshprt(60)

!! === readin GWIN and LMTO, then allocate and set datas. ===
!! See use m_genallcf_v3,only: at the top of this routine
c      nwin   = 0                !Readin nw from NW file
c      efin=-999d0               !not readin EFERMI
        if(ixc==3) then; incwfin= -2 !core exchange mode
        else           ; incwfin= -1 !use 7th colmn for core at the end section of GWIN
        endif
        call genallcf_v3(incwfin) ! module m_genallcf_v3. See use m_genallcf in this rouitine
        if(ngrp/= ngrp2) call rx( 'ngrp inconsistent: BZDATA and LMTO GWIN_V2')
        esmref=esmr

!! iSigMode
        call readd_isigma_en(ifinin,isigmode) !reading self-energy mode parameter from file 'GWinput'
        if(diagonly) isigmode=5

!! Get maximums
        call getnemx8(nbmx,ebmx)  !Get maximums takao 18June03
!!     nbmx1 ebmx1: to set how many bands of <i|sigma|j>  do you calculate.
!!     nbmx2 ebmx2: to restrict num of bands of G to calculate G \times W
!! ebmx2 nbmx2 are not used. For safe, strange number is supplied here.
        nbmx(2)=9999999
        ebmx(2)=1d10
        write(6,"('  nbmx ebmx from GWinput=',i8,d13.5)") nbmx(1),ebmx(1)

!!Caution!  WE ASSUME iclass(iatom)= iatom (because of historical reason)
        if (nclass /= natom ) call rx( ' hsfp0: nclass /= natom ')
        write(6,*)' hsfp0_sc: end of genallcf_v3'
        call pshprt(30)
        pi   = 4d0*datan(1d0)
        tpia = 2d0*pi/alat
c      call dinv33(plat,1,xxx,vol)
c      voltot = dabs(vol)*(alat**3)
        voltot = abs(alat**3*tripl(plat,plat(1,2),plat(1,3)))
c      shtw = 0d0
        tetra= tetra_hsfp0()
!! if(esmr<1d-5) shtw=0.01d0 ! Ferdi's shift to avoid resonance effect(maybe), I used this until sep2012

c$$$!! ef is taken as rs for the empty-sphere test case of legas=T case
c$$$!! HOMOGENIOUS GAS code. Usually not used. Need fixing if necessary.
c$$$!! Keep this just as a memo.
c$$$      legas = .false.
c$$$      if(.false.) then
c$$$        INQUIRE (FILE = 'LEGAS', EXIST = legas)
c$$$        if(legas) then          !!! test for electron gas case.
c$$$          write(6,*)' find LEGAS. legas =',legas
c$$$          iflegas = 2101
c$$$          open (iflegas,file='LEGAS')
c$$$          read(iflegas,*)rs
c$$$          close(iflegas)
c$$$          alpha = (9*pi/4d0)**(1d0/3d0)
c$$$          qfermi = alpha/rs
c$$$          efx  = qfermi**2
c$$$          valn = efx**1.5d0*voltot/3d0/pi**2
c$$$          write (6,*)'  #### egas test mode  legas=T #### given rs =',rs
c$$$          write (6,*)' egas  Exact Fermi momentum  qf  =', qfermi
c$$$          write (6,*)' egas  Exact Fermi energy    Ef  =', efx
```

```
c$$$          if(tetra) call rx( 'legas You have to give ef of  tetrahedron')
c$$$        endif
c$$$      endif
c$$$!!
      if(ixc==1) then
        exchange=.true.
        write(6,*) ' --- Exchange mode --- '
        if(mpi__root) then
          ifxc(1)  = iopen('XCU'//xt(nz),1,-1,0)
          ifsex(1) = iopen('SEXU'//xt(nz),1,-1,0)
          ifsex2(1)= iopen('SEX2U',0,-1,0) !out SEX_nn'
          if (nspin == 2) then
            ifxc(2)  = iopen('XCD'//xt(nz),1,-1,0)
            ifsex(2) = iopen('SEXD'//xt(nz),1,-1,0)
            ifsex2(2)= iopen('SEX2D',0,-1,0) !out SEX_nn'
          endif
        endif
c         INQUIRE (FILE = 'EXspTEST', EXIST = exspwrite)
c         if(exspwrite) then
c            write(6,*)'--- Find EXspTEST ExspectrumWrite=',exspwrite
c            write(6,*)'--- esmr is chosen to be 2d0 Ry'
c            esmr= 2d0
c            do is=1,nspin
c               ifexsp(is)  = iopen('EXSP.'//char(48+is),1,-1,0)
c            enddo
c         endif
      elseif(ixc==2) then
        exchange=.false.
        write(6,*) ' --- Correlation mode --- '
        if(cohtest) write(6,*) ' COH calculation mode. Results in COH'
        if(mpi__root) then
          ifsec(1)   = iopen('SECU'//xt(nz),1,-1,0) ! output files
          ifsec2(1)= iopen('SEC2U',0,-1,0) !out SEC_nn'
          if (nspin == 2)
     .     ifsec(2)   = iopen('SECD'//xt(nz),1,-1,0)
          ifsec2(2)= iopen('SEC2D',0,-1,0) !out SEC_nn'
        endif
      elseif(ixc==3) then
        exchange=.true.
        esmr=0d0
        write(6,*) ' --- CORE Exchange mode --- '
        if(mpi__root) then
          ifsex(1)   = iopen('SEXcoreU'//xt(nz),1,-1,0)
          ifsex2(1)= iopen('SEXcore2U',0,-1,0) !out SEXcore_nn'
          if (nspin == 2) then
            ifsex(2)   = iopen('SEXcoreD'//xt(nz),1,-1,0)
            ifsex2(2)= iopen('SEXcore2D',0,-1,0) !out SEXcore_nn'
          endif
        endif
!! spectrum funciton mode, we do not use ixc==4
c     elseif(ixc==4) then
c     write(6,*) ' --- Spectrum function Sigma(\omega) mode --- '
c     exchange=.false.
c     ifsecomg(1) = iopen('SEComgU'//xt(nz),1,-1,0) ! output files
c     if (nspin == 2)
c     .  ifsecomg(2) = iopen('SEComgD'//xt(nz),1,-1,0)
      else
        call rx( ' hsfp0: Need input (std input) 1(Sx) 2(Sc) or 3(ScoreX)!')
      endif

c---  Neglect core is NoCore exists ------------
c     inquire(file='NoCore',exist=nocore)
c     if(nocore) nctot=0

      write(6, *) ' --- computational conditions --- '
      write(6,'("    deltaw  =",f13.6)') deltaw
c     write(6,'("    ua      =",f13.6)') ua
      write(6,'("    esmr    =",f13.6)') esmr
      write(6,'("    alat voltot =",2f13.6)') alat, voltot

!! read dimensions of wc,b,hb
      ifhbed = ifile_handle()   !  ifhbed = iopen('hbe.d',1,0,0)
! ifile_handle() search unused file handle
      open(ifhbed,file='hbe.d',status='old')
      read (ifhbed,*) nprecb,mrecb,mrece,nlmtot,nqbzt, nband,mrecg
      close(ifhbed)             !isx = iclose ('hbe.d')
      if (nprecb == 4) call rx( 'hsfp0: b,hb in single precision')
!!
      call init_readeigen(ginv,nspin,nband,mrece) !initialization of readEigen
! required for readeigen readchpi readgeig.

!!  === Get space group information ===
!! True class information in order to determine the space group,
!! because the class in the generated GW file is dummy. (iclass(ibas)=ibas should be kept).
      ificlass=ifile_handle()
      open (ificlass,file='CLASS')
      allocate(iclasst(natom),invgx(ngrp))
     & ,miat(natom,ngrp),tiat(3,natom,ngrp),shtvg(3,ngrp))
      write(6,*)'  --- Readingin CLASS info ---'
      do ibas = 1,natom
        read(ificlass,*) ibasx, iclasst(ibas)
        write(6, "(2i10)") ibasx, iclasst(ibas)
      enddo
      close(ificlass)
!!  Get space-group transformation information. See header of mptauouf.
      call mptauof(symgg,ngrp,plat,natom,pos,iclasst
     o ,miat,tiat,invgx,shtvg ) !note: miat,tiat,shtvg are defined in m_zmel.
      if(verbose()>=40) write (*,*)' hsfp0.sc.m.F: end of mptauof'

!! ====  Get array size to call rdpp can call rdpp to generate base data for get_zmel ====
      call getsrdpp2( nclass,nl,nxx)
      call readngmx('QGpsi',ngpmx)
      call readngmx('QGcou',ngcmx)
      write(6,*)' max number of G for QGpsi and QGcou: ngcmx ngpmx=',ngcmx,ngpmx
      allocate(ngvecp(3,ngpmx),ngvecc(3,ngcmx))
      call readqg('QGpsi',qibz(1:3,1),ginv, quu,ngpn1, ngvecp)
      call readqg('QGcou',qibz(1:3,1),ginv, quu,ngcn1, ngvecc)
      deallocate(ngvecp,ngvecc)
      write(6,*) ' end of read QGcou'
!!  ppbrd = radial integrals
!!  cgr   = rotated cg coeffecients.
      call rdpp(nxx, nl, ngrp, nn, nclass, nspin, symgg,qbas)
!  output: nblocha, lx, nx,  ppbrd , mdimx, nbloch, cgr are stored in m_rdpp.
      call pshprt(60)

!! Readin WV.d
      if(.not.exchange.or.(exchange.and.screen)) then !screen means screened exchange case
        ifwd=ifile_handle()     ! ifwd = iopen('WV.d',1,-1,0)
!direct access files WVR and WVI which include W-V.
        open(ifwd,file='WV.d')
        read (ifwd,*) nprecx,mrecl,nblochpmx,nwp,niwt, nqnum, nw_i
        write(6,"(' Readin WV.d =', 10i8)") nprecx,mrecl,nblochpmx,nwp,niwt, nqnum, nw_i
        close(ifwd)             !ifwd =iclose('WV.d')
        call checkeq(nprecx,ndble)
        nw = nwp-1
        if(niwt /= niw) call rx( 'hsfp0_sc: wrong niw')

!! Energy mesh; along real axis. Read 'freq_r'
!! NOTE nw_i=nw for non-timereversal case.
!!      nw_i=0 for time-reversal case.
!!  NOTE: We assume freq_r(i) == -freq_r(-i) in this code. feb2006
!!  NOTE: this program assumes freq_r(iw)=freq_r(-iw). freq_r(iw <0) is redundant.
c        write(6,'("    niw nw dw   =",2i6,f13.6)') niw,nw,dw
        ififr=ifile_handle()
        open(unit=ififr,file='freq_r')
        read(ififr,*)nwxx
        if(nwxx/= nw+1) call rx( ' freq_r nw /=nw')
        allocate(freq_r(nw_i:nw)) !freq_r(0)=0d0
        do iw= nw_i,nw
          read(ififr,*) freq_r(iw)
        enddo
        close(ififr)
        if(nw_i/=0) then
          if(nw/= -nw_i)        call rx( "sxcf_fal3_scz: nw/=-nw_i")
          if(freq_r(0)/=0d0)    call rx( "sxcf_fal3_scz: freq_r(0)/=0")
          if( sum(abs( freq_r(1:nw)+freq_r(-1:-nw:-1)))/=0)
     &      call rx( "sxcf_fal3_scz: freq_r /= -freq_r")
        endif
      endif

!! efermi by tetrahedron. this can be overwritten
c      ifief=ifile_handle()
c      open(ifief,file='EFERMI')
c      read(ifief,*) ef
c      close(ifief)
      call readefermi()

      if(tetra) goto 201        !tetra is experimental.  usually =F.

!!== Determine Fermi energy ef for given valn (legas case), or corresponding charge given by z and konf.==
!!    When esmr is negative, esmr is geven automatically by efsimplef.
c      write(6,"(a,f12.6)")' --- READIN ef from EFERMI. ef=',ef
      legas=.false.
      call efsimplef2a(nspin,wibz,qibz,ginv,
     i nband,nqibz
     i ,konf,z,nl,natom,iclass,nclass
     i ,valn, legas, esmref,    !!! valn is input for legas=T, output otherwise.
     i qbz,nqbz                 ! index_qbz, n_index_qbz,
     o ,efnew)
      if(ixc/=3) ef = efnew
      eftrue = efnew

!! ==== check total ele number =====
      ntot  = nocctotg2(nspin, ef,esmr, qbz,wbz, nband,nqbz)
      write(6,*)' ef    =',ef
      write(6,*)' esmr  =',esmr
      write(6,*)' valn  =',valn
      write(6,*)' ntot  =',ntot

!! == Core-exchange case. ef means just below the valence eigenvalue (to take only core in sxcf).==
      if(ixc==3) then
        ef = lowesteval() -1d-3 !lowesteigen(nspin,nband,qbz,nqbz) - 1d-3 !lowesteb was
        call getkeyvalue("GWinput","EXonly",wex,default=0d0)
        if(wex==0d0) then
          exonly=.false.
        else
          exonly=.true.
          write(6,*)' exonly=T ecore shift: ecore---> ecore-100'
          ecore = ecore-100.0
        endif
        write(6,"(a)")' CoreEx mode: We change ef as ef=lowesteval-1d-3, slightly below the bottom of
 valence.'
        write(6,"(a,f13.5,i5,i5)")' CoreEx mode: ef nspin nctot=',ef,nspin,nctot
        do ix=1,nctot
           write(6,"(i4,x,d13.5,x,d13.5)") ix,(ecore(ix,is),is=1,nspin)
        enddo
c        if(maxval(ecore(:,1:nspin))>ef) then !ef is bottom of valence.
c          call rx( 'hsfp0 ixc=3: ecore>evalence. ')
c        endif
      endif
 201  continue

      call init_readeigen2(mrecb,nlmto,mrecg) !initialize m_readeigen

!! Read q-points and states
      nspinmx = nspin
      if(selectqp .and. mpi__root) then
        call getkeyvalue("GWinput","<QPNT>",unit=ifqpnt,status=ret)
        lqall      = .false.
        laff        = .false.
        call readx(ifqpnt,10)
        read (ifqpnt,*) iqall,iaf
        if (iqall == 1) lqall = .true.
        if (iaf   == 1)   laff = .true.
        call readx(ifqpnt,100)
        if (lqall) then         !all q-points case
          nq         = nqibz
          allocate(q(3,nq))
          call dcopy(3*nqibz,qibz,1,q,1)
        else
          call readx(ifqpnt,100)
          read (ifqpnt,*) nq
          allocate(q(3,nq))
          do       k = 1,nq
            read (ifqpnt,*) i,q(1,k),q(2,k),q(3,k)
          enddo
        endif
        nspinmx = nspin
        if (laff) nspinmx =1
        close(ifqpnt)
      else
!     q-points. bzcase()=1
        nq = nqibz
        allocate(q(3,nq))
        q(:,1:nq) = qibz(:,1:nq) !call dcopy   (3*nqibz,qibz,1,q,1)
      endif
!!
      call mpi__broadcast(nq)
      if(mpi__root) then
        do dest=1,mpi__size-1
          call mpi__real8send(q,3*nq,dest)
        enddo
      else
        call mpi__real8recv(q,3*nq,0)
      endif
!! antiferro case. Only calculate up spin
      call anfcond()
      if(laf) nspinmx=1
      call mpi__broadcast(nspinmx)


!! Determine ntq.  See also in sxcf_fal.sc.F ntq should be common for all ixc modes.
!! FIX NTQ during iteration by the file NTQ 15jun2015
!!
!! Determine nbandmx. Moved from sxcf_fal2.sc.F.
!!!! count number of band to calculate.
!! I think it it better to determine nbandmx in a manner within LDA
!! (need to care degeneracy...).
      allocate(nbandmx(nq,nspinmx))
      if(mpi__root) then
        inquire(file='NTQXX',exist=lntq)

        ifih = ifile_handle()
        open(ifih,file='NTQXX')
!!   Get ntq
        if(lntq) then
          read(ifih,*) nband_r,nq_r,ntq
          if(nband_r/=nband.or.nq_r/=nq) then
            rewind ifih
            lntq=.false.
          endif
        endif
        if(.not.lntq) then
          ntq=0
          allocate(eqt(nband))
          do is = 1,nspin
            do ip = 1,nq
              call readeval(qibz(1,ip),is, eqt)
              do iband=1,nband
                ntq = max(iband,ntq)
                if(eqt(iband)-eftrue>ebmx(1)) exit
              enddo
            enddo
          enddo
          ntq = min(ntq, nbmx(1))
          deallocate(eqt)
          write(ifih,"(3i10)") nband,nq,ntq
        endif
!!   Get ntqxx(iq,isp) and nbandmx
        allocate(eqt(nband))
        do is = 1,nspinmx
          do ip = 1,nq
            call readeval(qibz(1,ip),is, eqt)
            if(lntq) then
              read(ifih,*) ntqxx   ! ntqxx = ntq !jun2016
            else
              ntqxx = 0
              do i = 1,ntq
                if(eqt(i)-eftrue<ebmx(1)) ntqxx =ntqxx  + 1
              enddo
              ntqxx = min(ntqxx, nbmx(1))
              write(ifih,"(i10)") ntqxx
            endif
            if(ntqxx<nband) then ! redudce ntqxx when band tops are degenerated.
              do i=ntqxx,1,-1
                if(eqt(i+1)-eqt(i)<1d-2) then !1d-2 is a tol to check degeneracy.
                  ntqxx=i-1
                else
                  exit
                endif
              enddo
            endif
            nbandmx(ip,is) = ntqxx !number of bands to be calculated
          enddo
        enddo
        deallocate(eqt)
        close(ifih)
      endif
      call mpi__broadcast(ntq)
!!
      do is=1,nspinmx
        if(mpi__root) then
          print *,'is nbandmx(:,is)=',is,nbandmx(:,is)
          do dest=1,mpi__size-1
            call mpi__send_iv(nbandmx(1:nq,is),dest)
          enddo
        else
          call mpi__recv_iv(nbandmx(1:nq,is),0)
        endif
      enddo

!! trivial case of itq itq(i)=i
      allocate (itq(ntq))
      do i = 1, ntq
         itq(i) = i !itq is used also in hsfp0.m.F
      enddo
      do iq=1,nq
         write(6,'(" Target iq q=",i6,3f9.4)')iq,q(:,iq)
      enddo

!! read LDA eigenvalues
c      allocate(omega(ntq))
      allocate(eqx(ntq,nq,nspin),eqx0(ntq,nq,nspin),eqt(nband))
      do is = 1,nspin
        do ip = 1,nq
          call readeval(q(1,ip),is,eqt)
          eqx0(1:ntq,ip,is) = eqt(itq(1:ntq))
          eqx(1:ntq,ip,is) = rydberg()*(eqt(itq(1:ntq))- eftrue)
        enddo
      enddo
      deallocate(eqt)

      write (6,*)' ***'
      write (6,6700) nspin,nq,ntq
 6700 format (1x,3i4,'  nspin  nq  ntq')
      write (6,6501) is,nbloch,ngpn1,ngcn1,nqbz,nqibz,ef,deltaw,alat,ef,esmr
 6501 format (' spin =',i2,'   nbloch ngp ngc=',3i4
     & ,'  nqbz =',i6,'  nqibz =',i6,'   ef=', f10.4,' Rydberg'
     & ,/,d23.16,' <= deltaw(Hartree)'
     & ,/,d23.16,' <= alat'
     & ,/,d23.16,' <= ef '
     & ,/,d23.16,' <= esmr')
c      call winfo(6,nspin,nq,ntq,is,nbloch,ngpn1,ngcn1,nqbz,nqibz,ef,deltaw,alat,esmr)
!!-------------------------
!!     LDA exchange-correlation
!!-------------------------
      if(ixc==1) then
        allocate(  vxcfp(ntq,nq,nspin) )
        call rsexx(nspin,itq,q,ntq,nq, ginv, vxcfp) !add ginv july2011
        if(mpi__root) then
          do is = 1,nspinmx
            write (ifxc(is),*) '==================================='
            write (ifxc(is),"(' LDA exchange-correlation : is=',i3)")is
            write (ifxc(is),*) '==================================='
            call winfo(ifxc(is),nspin,nq,ntq,is,nbloch
     &       ,ngpn1,ngcn1,nqbz,nqibz,ef,deltaw,alat,esmr)
            write (ifxc(is),*)' ***'
            write (ifxc(is),"(a)") ' jband   iq ispin
     &qvec
     &eigen-Ef (in eV)
     &LDA XC (in eV)'
            ifoutsex = ifxc(is)
            write(6,*)
            do ip = 1,nq
              do i  = 1,ntq
                write(ifoutsex,"(3i5,3d24.16,3x,d24.16,3x,d24.16)")
     &           itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &           vxcfp(i,ip,is)
                if(eqx(i,ip,is) <1d20.and.vxcfp(i,ip,is)/=0d0) then !takao june2009. See lmf2gw (evl_d=1d20; in Ry.. but eqx is in eV. no problem for inequality).
                  write(6,"(' j iq isp=' i3,i4,i2,'  q=',3f8.4,
     &'  eig=',f10.4,'  Sxc(LDA)=',f10.4)")
     &             itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &             vxcfp(i,ip,is)
                endif
              end do
            end do
            if(is==1) isx = iclose('XCU'//xt(nz))
            if(is==2) isx = iclose('XCD'//xt(nz))
          enddo                 !     end of spin-loop
        endif                   !MPI__root
        deallocate(vxcfp)
      endif

!!  Offset Gamma point Q0P
      write(6,*) 'reading QOP'
      ifiq0p=ifile_handle()
      open (ifiq0p,file='Q0P')
      read (ifiq0p,"(i5)") nq0i
      if(.not.exchange) call checkeq(nqibz+nq0i-1, nqnum)
      write(6,*) ' *** nqibz nq0i_total=', nqibz,nq0i
      allocate( wqt(1:nq0i),q0i(1:3,1:nq0i) )
c     read (101,"(d24.16,3x, 3d24.16)" )( wqt(i),q0i(1:3,i),i=1,nq0i)
      nq0ix = nq0i
      do i=1,nq0i
        read (ifiq0p,* ) wqt(i),q0i(1:3,i)
        if(wqt(i)==0d0 ) nq0ix = i-1
      enddo
      nq0i = nq0ix              ! New nq0i July 2001
      write(6,*) ' Used k number in Q0P =', nq0i
      write(6,"(i3,f14.6,2x, 3f14.6)" )(i, wqt(i),q0i(1:3,i),i=1,nq0i)
      close(ifiq0p)
      allocate( wgt0(nq0i,ngrp) )
      call getkeyvalue("GWinput","allq0i",allq0i,default=.false.) !S.F.Jan06
      call q0iwgt3(allq0i,symgg,ngrp,wqt,q0i,nq0i, !S.F.Jan06
     o wgt0)                    ! added allq0i argument
      if (nq0i/=0 ) write(6,*) ' *** tot num of q near 0   =', 1/wgt0(1,1)
      write(6,"('  sum(wgt0) from Q0P=',d14.6)")sum(wgt0)
c$$$      if(bzcase()==2) then
c$$$         wgt0= wgt0*wgtq0p()/dble(nqbz)
c$$$         write(6,"('bzcase=2:  sum(wgt0_modified )=',d14.6)")sum(wgt0)
c$$$      endif

!! Pointer to optimal product basis
c      allocate(imdim(natom))
```

```
c      call indxmdm (nblocha,nclass,iclass,natom,
c     o imdim )                  !in m_zmel
      if(niw/=0) then
!! Generate gaussian frequencies x between (0,1) and w=(1-x)/x
        allocate(freqx(niw),freqw(niw),wwx(niw)) !,expa(niw))
        call freq01x(niw,     !ua,
     o   freqx,freqw,wwx)       !,expa)
      endif

c$$$!!  ------ write energy mesh for check ----------
c$$$      ifemesh = iopen('emesh.hsfp0'//xt(nz),1,-1,0)
c$$$      deltax0 = 0d0
c$$$      if(MPI__root) then
c$$$        call writeemesh(ifemesh,freqw,niw,freq,nw,deltax0)
c$$$      endif

!! === readin Vcoud and EPSwklm for newaniso()=T ===
      ifidmlx = iopen('EPSwklm',0,0,0)
      read(ifidmlx) nq0ix,lxklm
      if(nq0i/=nq0ix) then
        write(6,*)'nq0i from EPSwklm /= nq0i',nq0i,nq0ix
        call rx( 'nq0i from EPSwklm /= nq0i')
      endif
      allocate( dmlx(nq0i,9))
      allocate( epinvq0i(nq0i,nq0i) )
      allocate( wklm((lxklm+1)**2))
      read(ifidmlx) dmlx, epinvq0i
      read(ifidmlx) wklm
      ifidmlx = iclose('EPSwklm')

c----tetra block is experimental. unused usually. ----------------
      if(tetra) then
c     --- get tetrahedron
c     mxkp = n1*n2*n3
c     allocate( qbzxx(3*mxkp),wbzxx(mxkp),ipq(mxkp) )
c     call bzmesh (plat,qbasmc,n1,n2,n3,w(igrp),ngrp,ipq,
c     .               qbzxx,wbzxx,nqibzxx,mxkp)
c     allocate(idtetx(0:4,mxkp*6))
c     call tetirr(qbasmc,n1,n2,n3,ipq,nqibz,ntet,
c     .              idtetx)
c     allocate(idtet(0:4,ntet))
c     idtet(0:4,1:ntet) = idtetx(0:4,1:ntet)
c     deallocate(idtetx,qbzxx,wbzxx,ipq)
c
c     nene = ntq*nq*nspin ! for energy points.
c     if(exchange) nene=0
c     allocate(wtet(nband,nspin,nqibz,0:3*nene),
c     &    eband(nband,nspin,nqibz), qz(3,nqibz),nstar(nqibz),
c     &    iene(3*ntq,nq,nspin), ene(0:3*nene) ) ! pointer for
        allocate(wtet(nband,nspin,nqibz,0:0),
     &   eband(nband,nspin,nqibz), qz(3,nqibz) ) ! pointer for
        call dcopy(3*nqibz,qibz,1,qz,1)
        do  is    = 1,nspin     !Readin eband
          do  iqi = 1,nqibz
c     iq       = idxk (qz(1:3,iqi),qbz,nqbz)
c     call rwdd1 (ifev(is), iq, nband, eband(:,is,iqi))
            call readeval(qz(1:3,iqi),is, eband(:,is,iqi))
          enddo
        enddo
c     wtet(nband,nsp,nqibz,iene) where
c     the energy pointer as iene(itp,ip,ispin) corresponding its energy value.
c     ene(0) = ef
c     if(.not.exchange) then
c     ix =0
c     do is = 1,nspin
c     do ip = 1,nq
c     do i  = 1,ntq
c     do iw = -1,1
c     ix  = ix+1
c     iene(3*i+iw-1,ip,is) = ix
c     ene(ix) = eqx0(i,ip,is) + 2.d0*(dble(iw)-shtw)*deltaw
c     enddo
c     enddo
c     enddo
c     enddo
c     endif
c     do ix = 0,3*nene
c     ene(ix) = ene(ix)-1d-15  ! to avoid coincidence
c     call bzints2(n1,n2,n3,eband,wtet(:,:,:,ix),nqibz,nband,nband,
c     .                nspin,edummy,edummy,edummy,1,ene(ix),2,ntet,idtet)
c     enddo
        volwgt = (3d0 - nspin) / ntetf ! ntetf was =6*n1*n2*n3
        call bzints2x(volwgt,eband,wtet(:,:,:,0),nqibz,nband,nband,
     .   nspin,edummy,edummy,edummyd,1,ef,2,nteti,idteti)
        ntot= sum(wtet)
c        if(legas) then
c          write(6,"(' tetra=T ef ntot nexact ratio=',15f12.6)") ef,ntot
```

```
c     &     , ef**1.5d0/3d0/pi**2*voltot, ef**1.5d0 /3d0/pi**2*voltot/ntot
c        else
c          write(6,"(' tetra=T ef nvalence)=',15f12.6)") ef,ntot
c        endif
        write(6,"(' tetra=T ef nvalence)=',15f12.6)") ef,ntot
        if(nspin==1) wtet = wtet/2d0
        do iqi = 1,nqibz
          wtet(:,:,iqi,:) = wtet(:,:,iqi,:)/nstar(iqi)
        enddo
        deallocate( eband, qz, ene ) ! pointer for
c -- ibzx denote the index of k{FBZ for given k{1BZ.
        allocate(ibzx(nqbz))
        call invkibzx(irk,nqibz,ngrp,nqbz,
     o   ibzx)
      else
        allocate(wtet(1,1,1,1), iene(1,1,1)) !dummy
      endif
c ---- end of tetra section -------------------------------------------
c      iii=ivsumxxx(irk,nqibz*ngrp)
c      write(6,*) " sum of nonzero iirk=",iii, nqbz


!!------------------------------------------------------------
!!     calculate the the self-energy SEx(ip) or SEc(ip)
!!------------------------------------------------------------
!! eibz4sig() is EIBZ symmetrization or not...
      if(eibz4sig()) then
        allocate(nwgt(nqbz,1:nq),igx(ngrp*2,nqbz,nq))
        allocate(igxt(ngrp*2,nqbz,nq), eibzsym(ngrp,-1:1,nq))
        iqxini=1
        iqxend=nq
c         write(6,"('TimeRevesal switch = ',l1)") timereversal()
c         call eibzgen(nq,symgg,ngrp,q(:,iqxini:iqxend),
c     &        iqxini,iqxend,qbz,nqbz,timereversal(),ginv,iprintx,
c     o        nwgt,igx,igxt,eibzsym,tiii)
!! Check timereversal is required for symmetrization operation or not. If tiii=timereversal=F is enforced,
!! the symmetrization procedure in x0kf_v4h becomes a little time-consuming.
        tiii=.false.            !Enforce no time reversal. time reversal not yet...
        write(6,*)'NOTE:TimeReversal not yet implemented in hsfp0.sc.m.F'
        write(6,"('=== goto eibzgen === used timereversal=',l1)")tiii
        iprintx=.false.
        if(mpi__root) iprintx=.true.
        call eibzgen(nq,symgg,ngrp,q(:,iqxini:iqxend),
     &   iqxini,iqxend,qbz,nqbz,tiii,ginv,iprintx,
     o   nwgt,igx,igxt,eibzsym,tiiiout)
c     call PBindex(natom,lx,l2nl,nx) !all input. this returns requied index stored in arrays in m_pbindex.
! PBindex: index for product basis.  We will unify this system; still similar is used in ppbafp_v2.
c     call readqgcou() !no input. Read QGcou and store date into variables.
c     call Spacegrouprot(symgg,ngrp,plat,natom,pos) ! all inputs.
C         do iq=iqxini,iqxini
C         do ibz=1,200
C           if(nwgt(ibz,iq)/=0) then
C             write(6,"('yyy1: ',i8,2x,25(i3,i2))") ibz,(igx(i,ibz,iq),igxt(i,ibz,iq),i=1,nwgt(ibz,iq))
C           endif
C         enddo
C         enddo
      endif

!! == irkip control paralellization  ==
!! We have to distribute non-zero irkip into processes (nrank).
!! When irkip(nqibz,ngrp,nq,nspinmx)/=0, we expect grain-size
!! for each job of (iqibz,igrp,iq,isp) is almost the same.
!! Our pupose is to calculate zsec(itp,itpp,iq).
!! Thus we need to set up communicator (grouping) MPI__COMM_iqisp(iq,isp) to do all_reduce.
!! (for given zsec(iq,isp), we take sum on zsec for (iqibz,igrp) by all_reduce.)
!! ---
!! NOTE: in future, we will further extend irkip for itp and itpp
      allocate(irkip_all(nspinmx,nqibz,ngrp,nq)) !this is global
      allocate(nrkip_all(nspinmx,nqibz,ngrp,nq)) !this is global
      allocate(nrkip(nspinmx,nqibz,ngrp,nq)) !this is global
      if(eibz4sig()) then
        nrkip_all=0
        irkip_all=0
        is=1                    ! not spin dependent
        do iqq=1,nq
c     irkip_all(is,:,:,iqq)=irk
          do kx=1,nqibz
            do igrp=1,ngrp
              kr = irk(kx,igrp) !ip_all(is,kx,igrp,iqq) !kr is index for qbz (for example, nonzero # of kr
 is 64 for 4x4x4)
              if(kr==0) cycle
              if(nwgt(kr,iqq)/=0) then
                irkip_all(is,kx,igrp,iqq)= irk(kx,igrp)
                nrkip_all(is,kx,igrp,iqq)= nwgt(kr,iqq)
              endif
c     write(6,*)' iqq kr irk =',iqq,kr,irkip_all(is,kx,igrp,iqq),nrkip_all(is,kx,igrp,iqq)
            enddo
          enddo
        enddo
C          do iqq=1,nq
C             write(6,"('iq=',i4,' # of EIBZ: Used(TimeR 1 or -1)=',i3,'=',i3,'+',i3)")
C     &           iqq,sum(eibzsym(:,:,iqq)),sum(eibzsym(:,1,iqq)),sum(eibzsym(:,-1,iqq))
C             write(6,"('eibz: iqq sum(nrkip_all)=nqbz  ',i3,3f11.5,3i8)")
C     &           iqq,q(:,iqq),sum(nrkip_all(is,:,:,iqq)),nqbz
C             do kx=1,nqibz
C                do igrp=1,ngrp
C                   kr = irkip_all(is,kx,igrp,iqq) !kr is index for qbz
C                   if(kr/=0) write(6,"('      ',i8,3f11.5,i8,2x,25(i4,i2))")
C     &                  kr,qbz(:,kr),nrkip_all(is,kx,igrp,iqq)
C     &                  ,(igx(i,kr,iqq),igxt(i,kr,iqq),i=1,nwgt(kr,iqq))
C                enddo
C             enddo
C!     !   Probably partial group symmetrization is enough. But it may not reduce computational time so
 much.
C         enddo
        if(nspinmx==2) then
          irkip_all(2,:,:,:)=irkip_all(1,:,:,:)
          nrkip_all(2,:,:,:)=nrkip_all(1,:,:,:)
        endif
      else                      ! not eibz4sig
        do is = 1,nspinmx
          do iqq=1,nq
            irkip_all(is,:,:,iqq)=irk
          enddo
        enddo
      endif

!! -- ppb= <Phi(SLn,r) Phi(SL'n',r) B(S,i,Rr)>
      allocate( ppbir(nlnmx*nlnmx*mdimx*nclass,ngrp,nspin))
      do irot = 1,ngrp
        do isp = 1,nspin
          call ppbafp_v2(irot,ngrp,isp,nspin,
     i     il,in,im,nlnm,       !w(i_mnl),
     i     nl,nn,nclass,nlnmx,
     i     mdimx,lx,nx,nxx,     !Bloch wave
     i     cgr, nl-1,           !rotated CG
     i     ppbrd,               !radial integrals
     o     ppbir(:,irot,isp))   !this is in m_zmel
        enddo
      enddo

!! MPI RankDivider for iqibz and irot cycle in sxcf.
!!     nrkip is weight correspoinding to irkip for a node.
      allocate(irkip(nspinmx,nqibz,ngrp,nq)) !local
      call mpi__sxcf_rankdivider(irkip,irkip_all,nspinmx,nqibz,ngrp,nq) ! MIZUHO-IR
      nrkip = nrkip_all         ! we don't need to change this for MPI case.
! It just need to distribute non-zero irkip.
!!
      nlnx4    = nlnx**4        ! niwx     = max0 (nw+1,niw) !nw --->nw+1 feb2006
      allocate( kount(nqibz,nq),zsec(ntq,ntq,nq), coh(ntq,nq) )
!TIME1_0010 "main:before2000loop"
      do 2000 is = 1,nspinmx
!TIME0_0020
        if(mpi__root) then
          if(exchange) then
            write(ifsex2(is)) nspin, nq, ntq,nqbz,nqibz, n1,n2,n3
            write(ifsex(is),*) '======================================='
            write(ifsex(is),"('Self-energy exchange SEx(q,t): is=',i3)") is
            write(ifsex(is),*) '======================================='
            call winfo(ifsex(is),nspin,nq,ntq,is,nbloch,ngpn1,
     &       ngcn1,nqbz,nqibz,ef,deltaw,alat,esmr)
            write (ifsex(is),*)' *** '
            write (ifsex(is),"(a)") ' jband   iq ispin                  '//
     &       '           qvec            eigen-Ef (in eV)           exchange (in eV)'
          elseif(ixc==2) then
            write(ifsec2(is)) nspin, nq, ntq ,nqbz,nqibz  ,n1,n2,n3
            write(ifsec(is),*) '=========================================='
            write(ifsec(is),"('Self-energy correlated SEc(qt,w): is=',i3)") is
            write(ifsec(is),*) '=========================================='
            call winfo(ifsec(is),nspin,nq,ntq,is,nbloch,ngpn1,
     &       ngcn1,nqbz,nqibz,ef,deltaw,alat,esmr)
            write (ifsec(is),*)' *** '
            write (ifsec(is),"(a)") ' jband   iq ispin                  '//
     &       '           qvec            eigen-Ef (in eV)           '//
     &       'Re(Sc) 3-points (in eV)                      '//
     &       '           In(Sc) 3-points (in eV)                Zfactor(=1)'
          endif
        endif
        zsec  = 0d0
        coh   = 0d0
        kount = 0
        if(ixc==3.and.nctot==0) goto 2001 !make dummy SEXcore
!! dummy to overlaid -check bounds sep2014
        if(size(ecore)==0) then
```

```
          deallocate(ecore)
          allocate(ecore(1,2))
        endif

!!== ip loop to spedify external q ==
c        do 1001 ip = 1,nq
c          if(sum(irkip(is,:,:,ip))==0) cycle
        call sxcf_fal3_scz(kount,q,itq,ntq,ef,esmr,
     i   nspin,is,
     i   qbas,ginv,qibz,qbz ,wbz, nstbz,
     i   irkip(is,:,:,:),nrkip(is,:,:,:),
     i   freq_r,nw_i,nw, freqx,wwx,
     i   dwdummy,
     i   ecore(:,is),
     d   nlmto,nqibz,nqbz,nctot,
     d   nbloch,ngrp,niw,nq,
     i   nblochpmx, ngpmx,ngcmx,
     i   wgt0,nq0i,q0i,symgg,alat,
     i   nband,                 !shtvg,
     i   ifvcfpout,
     i   exchange, screen, cohtest, ifexsp(is),
     i   nbmx,ebmx,
     i   wklm,lxklm,
     i   eftrue,
     i   jobsw = isigmode, nbandmx=nbandmx(1:nq,is), !nbandmx is input mar2015
     i   hermitianw=hermitianw,
     o   zsec=zsec)
c 1001   continue
!TIME1_0020 "main:endofsxcf_fal3_scz"
c        call date_and_time(values=timevalues)
c        write(6,'(a,9i5)')'dateandtime2=',MPI__rank,timevalues(1:8)
c        call cpu_time(time_red1)

!! CAUITION! Allreduce wait all cpu jobs done here.
!! Before nov2013, MPI__sxcf_rankdivider was stpid---> half of cores assigned for isp=2
!! was just waiting here!
c      call MPI__AllreduceMax( nbandmx(:,is), nq ) ! MIZUHO-IR
c      call cpu_time(time_red2)
c      write(6,*) MPI__rank,'time(MPI__AllreduceMax)=',time_red2-time_red1

c$$$!!  electron gas bare exchange (exact)
c$$$        if (legas.and.exchange) then
c$$$          efz=(ntot*3*pi**2/voltot)**(2d0/3d0) ! ef is calculated from ntot.
c$$$          pi         = 4.d0*datan(1.d0)
c$$$          tpia       = 2.d0*pi/alat
c$$$          qfermi= dsqrt(efz)
c$$$          alpha = (9*pi/4d0)**(1d0/3d0)
c$$$          write (6,*)' --- exact electron gas bare exchange --- '
c$$$          write (6,*)' density parameter rs= ', alpha/qfermi
c$$$          write (6,*)' kf= ',qfermi
c$$$          do      ip = 1,nq
c$$$            qreal =  tpia*q(1:3,ip)
c$$$            qm    = dsqrt ( sum(qreal**2) )
c$$$            xsex  = hartree * egex (qm,efz)
c$$$            write (6,*)
c$$$            write (6,"(' True qm-ef Sx=',2f14.6,' q/qf=',f14.6)")
c$$$     &       rydberg()*(qm**2-efz), xsex, qm/qfermi
c$$$            write (6,"(' Num  qm-ef Sx=',2f14.6)")
c$$$     &       eqx(1,ip,is),        hartree*dreal(zsec(1,1,ip)) !sf 21May02
c$$$            write (6,"(' === diff     =',2f14.6)")
c$$$     &       rydberg()*(qm**2-efz)-eqx(1,ip,is)
c$$$     &       , xsex - hartree*dreal(zsec(1,1,ip)) !sf 21May02
c$$$            write (661,"(' qm True qm-ef Sx=',3f14.6)")
c$$$     &       qm,rydberg()*(qm**2-efz), xsex
c$$$            write (662,"(' qm Num  qm-ef Sx=',3f14.6)")
c$$$     &       qm,eqx(1,ip,is),     hartree*dreal(zsec(1,1,ip)) !sf 21May02
c$$$ccc   write (ifsex(is),6600) qreal(1),qreal(2),qreal(3),xsex
c$$$ccc   write (6,6600) qreal(1),qreal(2),qreal(3),xsex
c$$$ccc   6600   format (' qreal =',3f8.4,'   SEx(q) =',d13.5)
c$$$            write (663,"(2f14.6)") qm/qfermi, qfermi
c$$$          end do
c$$$        endif
 2001   continue

!! eibz4sig symmetrization. MPI__AllreduceSum in zsecsym.
        if(eibz4sig()) then
!TIME0_0030
          call zsecsym(zsec,ntq,nq,nband,nbandmx,nspinmx, eibzsym,ngrp,tiii,q,is)
!TIME1_0030 'zsecsym'
        endif
!TIME0_0040
        call mpi__allreducesum( zsec,ntq*ntq*nq )
!TIME1_0040 'MPI__AllreduceSumzsec'

        if(mpi__root) then
          if(exchange) then
            ifoutsex=ifsex(is)
            write(6,*)
            do ip = 1,nq
              do i  = 1,ntq
                write(ifoutsex,"(3i5,3d24.16,3x,d24.16,3x,d24.16)")
     &           itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &           hartree*dreal(zsec(i,i,ip)) !sf 21May02
                if( eqx(i,ip,is)<1d20.and.abs(zsec(i,i,ip))/=0d0 ) then !takao june2009
                  write(6,"(' j iq isp=' i3,i4,i2,'  q=',3f8.4,' eig=',f10.4,'  Sx=',f10.4)")
     &             itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &             hartree*dreal(zsec(i,i,ip)) !sf 21May02
                endif
              end do
              write(ifsex2(is)) is, q(1:3,ip), zsec(1:ntq,1:ntq,ip) !SEC_nn' out
            end do
          elseif(ixc==2) then
            ifoutsec=ifsec(is)
            do ip = 1,nq
              do i  = 1,ntq
                if( eqx(i,ip,is)<1d20.and.abs(zsec(i,i,ip))/=0d0 ) then !takao june2009
                  write(6,"(' j iq isp=' i3,i4,i2,'  q=',3f8.4,'  eig=',f8.4,'  Re(Sc) =',f8.4,'  Img(Sc) =',f8.4 )")
     &             itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &             hartree*dreal(zsec(i,i,ip)),
     &             hartree*dimag(zsec(i,i,ip))
                endif
                write(ifoutsec,"(3i5,3d24.16,3x,d24.16,3x,d24.16, 3x,d24.16)")
     &           itq(i),ip,is, q(1:3,ip), eqx(i,ip,is),
     &           hartree*dreal(zsec(i,i,ip)),
     &           hartree*dimag(zsec(i,i,ip))
              end do
              write(ifsec2(is)) is, q(1:3,ip), zsec(1:ntq,1:ntq,ip) !SEC_nn' out
            end do
          endif                 !ixc
        endif                   !MPI__root

 2000 continue                  !end of spin-loop

c$$$!!   --- EXspectrum -----------------------------------------------------
c$$$c     This section is similar with efsimplef.f
c$$$      if(sum(ifexsp(1:nspin))/=0) then
c$$$         do is = 1,nspin
c$$$            write(6,*)' --- Goto ExSpectrum section --- is=',is
c$$$            rewind (ifexsp(is))
c$$$            itmx = 0
c$$$            do
c$$$               read(ifexsp(is),*,end=1215)ipex,itpex,itex,qqex(1:3), eex,exsp
c$$$               if(itex>itmx) itmx=itex
c$$$            enddo
c$$$ 1215       continue
c$$$            nspexmx = itmx*(nqbz+nq0i*ngrp) !Get marimum value of the number of the ex spectrum
c$$$c
c$$$            allocate( eex1(nspexmx,ntq,nq), exsp1(nspexmx,ntq,nq),
c$$$     &           nspex(ntq,nq) ,
c$$$     &           itex1(nspexmx,ntq,nq),
c$$$     &           qqex1(3,nspexmx,ntq,nq) )
c$$$            write(6,*)' nspexmx =',nspexmx
c$$$c
c$$$            rewind (ifexsp(is))
c$$$            nspex = 0
c$$$            do
c$$$               read(ifexsp(is),*,end=1216) ipex,itpex,itex,qqex(1:3),eex,exsp
c$$$               nspex(itpex,ipex) = nspex(itpex,ipex)+1
c$$$               iex  = nspex(itpex,ipex)
c$$$               eex1  (iex,itpex,ipex) = eex
c$$$               exsp1 (iex,itpex,ipex) = exsp
c$$$               itex1 (iex,itpex,ipex) = itex
c$$$               qqex1(:,iex,itpex,ipex)= qqex
c$$$            enddo
c$$$ 1216       continue            !Get eex1(1:nspex) exsp1(1:nspex) for itp ip.
c$$$            write(6,*)' nspex(1 1)=',nspex(1,1)
c$$$c
c$$$            do ipex = 1,nq
c$$$               do itpex=1,ntq
c$$$                  write(6,*)' is itq ip =',is,itq,ip
c$$$                  nnex = nspex(itpex,ipex)
c$$$                  allocate( ieord(1:nnex) )
c$$$                  call sortea( eex1(1:nnex,itpex,ipex),ieord, nnex,isig)
c$$$                  eex1 (1:nnex,itpex,ipex)  = eex1  (ieord(1:nnex),itpex,ipex)
c$$$                  exsp1 (1:nnex,itpex,ipex) = exsp1 (ieord(1:nnex),itpex,ipex)
c$$$                  itex1 (1:nnex,itpex,ipex) = itex1 (ieord(1:nnex),itpex,ipex)
c$$$                  qqex1(:,1:nnex,itpex,ipex)= qqex1 (:,ieord(1:nnex),itpex,ipex)
c$$$
c$$$                  filenameex = 'EXSP'//charnum3(ipex)//charnum3(itpex)
c$$$     &                 //'.'//char(48+is)
c$$$                  ifexspx=4111
c$$$                  open(ifexspx,file=filenameex)
c$$$
```

```
c$$$                  filenameex = 'EXSS'//charnum3(ipex)//charnum3(itpex)
c$$$     &                 //'.'//char(48+is)
c$$$                  ifexspxx=4112
c$$$                  open(ifexspxx,file=filenameex)
c$$$
c$$$                  do i=1,nnex
c$$$                     write(ifexspx, "(2d14.6, i4, 3f14.6)")
c$$$     &                    eex1  (i,itpex,ipex), exsp1 (i,itpex,ipex),
c$$$     &                    itex1 (i,itpex,ipex), qqex1 (1:3,i,itpex,ipex)
c$$$                  enddo
c$$$c
c$$$                  eee  =-1d99
c$$$                  exwgt= 0d0
c$$$                  do i=1,nnex
c$$$                     if(eex1(i,itpex,ipex) > eee+1d-4 .or. i==nnex) then
c$$$                        if(i/=1) write(ifexspxx, "(2d23.15)")
c$$$     &                       eee, exwgt*hartree
c$$$                        eee  = eex1(i,itpex,ipex)
c$$$                        exwgt= exsp1 (i,itpex,ipex)
c$$$                     else
c$$$                        exwgt= exwgt + exsp1 (i,itpex,ipex)
c$$$                     endif
c$$$                  enddo
c$$$c
c$$$                  deallocate( ieord )
c$$$                  close(ifexspx)
c$$$                  close(ifexspxx)
c$$$               enddo
c$$$            enddo
c$$$            deallocate( eex1, exsp1, nspex, itex1, qqex1 )
c$$$         enddo
c$$$         write(6,*)' End of ExSpectrum section ---'
c$$$      endif
c      isx = iclose ('wc.d')
c      isx = iclose ('wci.d')
c      isx = iclose ('hbe.d')
      call cputid(0)
      write(6,*) '--- end of hsfp0_sc --- irank=',mpi__rank
      call flush(6)
      call mpi__finalize
!TIME1_0000 "main:totalofhsfp0_sc"
!TIMESHOW
      if(ixc==1 ) call rx0( ' OK! hsfp0_sc: Exchange mode')
      if(ixc==2 ) call rx0( ' OK! hsfp0_sc: Correlation mode')
      if(ixc==3 ) call rx0( ' OK! hsfp0_sc: Core-exchange mode')
      end program hsfp0_sc



      subroutine zsecsym(zsec,ntq,nq,nband,nbandmx,nspinmx, eibzsym,ngrp,tiii,q,is)
!! --- symmetrize zsec for eibz4sig mode. -----------------
!! Read a file lmfgw_kdivider, which contains info for vxc and evec (they are in separated files in MPI)
!!
c      use m_mpi,only: MPI__AllreduceSum
      use m_readeigen,only: readeval
      implicit none
      complex(8),intent(inout)::zsec(ntq,ntq,nq)
      integer,intent(in)::ntq,nq,nspinmx,nband,nbandmx(nq,nspinmx),is
      integer,intent(in):: ngrp,eibzsym(ngrp,-1:1,nq)
      logical,intent(in):: tiii !time reversal switch
      real(8),intent(in):: q(3,nq)

      complex(8),allocatable::zsect(:,:)
      integer:: ifile_handle,iqq
      integer:: procid,nrankv,ifvxc_,ifevec_,ifiproc,iqqxx,
     & isp,ixx,ixxx,nqixx,nspxx,ispxx,iqbz,i,igrp,iq
      character*256:: extn,ext
      character*256,allocatable:: extp(:)
      integer,allocatable:: ifevec__(:),ifvxc__(:),iprocq(:,:)

      integer:: nsym,nhdim,it,nblk,iband,napw,ldim,ierr,ispx,nbsize,nbsizemx
     & ,iblk1,iblk2,ii1,ii2,ie1,ie2,ne1,ne2,iqxx, ndimhx, nspx,nnnx
      integer,allocatable::iblki(:),iblke(:)
      complex(8),allocatable:: evec(:,:),evec_inv(:,:),evecrot(:,:),rmatjj(:,:,:)
      real(8),allocatable::evaliq(:)
      real(8)::tolry=1d-4,qqqx(3),qtarget(3),tolq=1d-8
      complex(8),allocatable:: ovl(:,:)
      integer::nev,j
!TIME0_0100
      write(6,*)'zsecsym:'
      allocate( zsect(ntq,ntq))
!! === readin lmfgw_kdivider, and get extensions === apr2013
      ifiproc=ifile_handle()
      open(unit=ifiproc,file='lmfgw_kdivider',status='old')
      read(ifiproc,*) ext
      read(ifiproc,*) nqixx, nspxx, nrankv
      if(allocated(iprocq)) deallocate(iprocq)
      allocate(iprocq(nqixx,nspxx))
      do isp=1,nspxx
        do iqq=1,nqixx
          read(ifiproc,*) iqqxx, ispxx, ixxx
          if(iqqxx/=iqq) call rx( 'iqqxx/=iqq')
          if(ispxx/=isp) call rx( 'ispxx/=isp')
          iprocq(iqq,isp) = ixxx
          write(6,"('iqq isp irank=',i8,i2,i6)") iqq,isp, iprocq(iqq,isp)
        enddo
      enddo
      close(ifiproc)
!! for multiple files.
c      if(allocated(extp)) deallocate(extp,ifvxc__,ifevec__)
      allocate(extp(0:nrankv-1),ifvxc__(0:nrankv-1),ifevec__(0:nrankv-1))
      extp(0) = trim(ext)
      write(6,"('  0 ext= ',a,a)") trim(extp(0)),' ----------'
      do procid=1,nrankv-1
        write(extn,"(i10)") procid
        extp(procid)=trim(adjustl(ext))//'_'//trim(adjustl(extn))
        write(6,"(i3,' ext= ',a,a)") procid,trim(extp(procid)),' ----------'
      enddo
      do procid=0,nrankv-1
        ifvxc__(procid) = ifile_handle()
        open( ifvxc__(procid), file='vxc'//extp(procid),form='unformatted')
        ifevec__(procid)= ifile_handle()
        open( ifevec__(procid), file='evec'//extp(procid),form='unformatted')
      enddo
      ifvxc_ = ifvxc__(0)       !0 is root
      ifevec_= ifevec__(0)
      read(ifevec_)  ndimhx, nspx,nnnx
      read(ifvxc_)              !skip ndimh, nsp,nnn
      allocate(evaliq(nband),iblki(nband),iblke(nband))
!TIME1_0100 "zsecsym:endof_allocate_zsect"
!TIME0_0110
      iqq=0                     !iqq is to read multiple vxc.* evec.*
      do 3020 iq=1,nq           !nq means iq for which we will calculate sigma
        iqq=iqq+1
        do 3030 ispx=1,nspinmx  !ispx loop is to find isx=is
          ifvxc_  = ifvxc__(iprocq(iqq,ispx))
          ifevec_ = ifevec__(iprocq(iqq,ispx))
          if(ispx==is) then
!this if-block is due to evec and v_xc file-->they shall be divieded into spin files.
            read(ifvxc_)  nhdim,ldim
            read(ifvxc_)
            allocate( evec(nhdim,nhdim),evecrot(nhdim,nhdim))
            read(ifevec_) qqqx(1:3), evec(1:nhdim,1:nhdim),nev !nev number of true bands nov2015
            zsect = 0d0
          else                  !skip isx/=is. Need to get access sequential files evec and v_xc.
            read(ifvxc_)
            read(ifvxc_)
            read(ifevec_)
            cycle
          endif
          do i=1,nnnx           !nq     !qqqx from evec v_xc.
            if(sum(abs(qqqx-q(:,i)))<tolq) then
              iqxx=i
              goto 3011
            endif
          enddo
          deallocate(evec,evecrot)
          call rx( 'hsfp0_sc: bug:qqqx can not find ...')
 3011     continue
          if(tiii) call rx( 'timereversal is not yet implemented')

!! evec_inv(ib1,iww)= \sum_ib2  ovlinv(ib1,ib2)*dconjg(evec(iww,ib2))  nov2015, we introduce nev. iww is for PMT basis. ib for band index.
!! This is for converting rotated evec (=evecrot(ib)) in the representation of original evec(ib).
          allocate(ovl(nev,nev))
c          print *,'nnnnnnnnn zsecsym: nband=',nhdim,nband,nev
          do i=1,nev
          do j=1,nev
c            write(6,*)'evec orth=',i,j,sum(dconjg(evec(:,i)*evec(:,j)))
            ovl(i,j)=sum(dconjg(evec(:,i))*evec(:,j))
          enddo
          enddo
          call matcinv(nev,ovl) !ovl --> ovlinv
          allocate(evec_inv(nev,nhdim))
          evec_inv = matmul(ovl(1:nev,1:nev),dconjg(transpose(evec(:,1:nev)))) !note ovl means ovlinv
          deallocate(ovl)
c          evec_inv = evec
c          call matcinv(nhdim,evec_inv)
          call readeval(q(:,iqxx), is, evaliq)
          nsym = sum(eibzsym(:,:,iqxx))
          do it=1,1             !no-time reversal yet !it=1,-1,-2 !c.f. x0kf_v4h
            do igrp=1,ngrp      !A-rotator
              if( eibzsym(igrp,it,iqxx)==0) cycle
              nblk=0
              iblki=0
              iblke=0
              iblki(1)=1
!!  degeneracy divider for evaliq. See How to apply EIBZ to
!! Is this procedure really make speed up so much?
              tolry= 0.2d0      !Degeneracy tol. if tolry is large,
!! larger tolry is safer, although a little inefficient.
!! If tolry is too small to divide degenerated values to different blocks --> then we have wrong results.
!(NOTE that Hamiltonian can be not so symmetric in some reasons)
              nbsizemx=0
              do iband=2,nbandmx(iqxx,is)
! nbandmx is the number of bands for which we calculate self-energy.
! We assume nbandmx(iqxx,is) is well separated for degeneracy.
                if(evaliq(iband) > evaliq(iband-1)+tolry
     &           .or.iband==nbandmx(iqxx,is)) then
                  nblk=nblk+1
                  if(nblk>=2) iblki(nblk)=iblke(nblk-1)+1
                  if(iband==nbandmx(iqxx,is)) then
                    iblke(nblk)=iband
                  else
                    iblke(nblk)=iband-1
                  endif
                  nbsize = iblke(nblk)- iblki(nblk)+1
                  if( nbsize>nbsizemx ) nbsizemx = nbsize
                endif
              enddo             ! iband
!! rotation of evec. Generate evecrot. (Within degenerated block, evec are mapped).e
              allocate(rmatjj(nbsizemx,nbsizemx,nblk))
              napw=nhdim-ldim
              do iblk1=1,nblk
                ii1=iblki(iblk1)
                ie1=iblke(iblk1)
                ne1=ie1-ii1+1
                call rotwvigg(igrp,q(:,iqxx),q(:,iqxx),nhdim,
     &           napw,ne1,evec(:,ii1:ie1),evecrot(:,ii1:ie1),ierr )
                rmatjj(1:ne1,1:ne1,iblk1) =
     &           matmul(evec_inv(ii1:ie1,:),evecrot(:,ii1:ie1))
              enddo             ! iblk1
              do iblk1=1,nblk
                do iblk2=1,nblk
                  ii1=iblki(iblk1)
                  ie1=iblke(iblk1)
                  ne1=ie1-ii1+1
                  ii2=iblki(iblk2)
                  ie2=iblke(iblk2)
                  ne2=ie2-ii2+1
                  zsect(ii1:ie1,ii2:ie2)= zsect(ii1:ie1,ii2:ie2)
     &             + matmul( dconjg(transpose(rmatjj(1:ne1,1:ne1,iblk1))),
     &             matmul(zsec(ii1:ie1,ii2:ie2,iqxx),
     &             rmatjj(1:ne2,1:ne2,iblk2)) )
                enddo           ! iblk2
              enddo             ! iblk1
              deallocate(rmatjj)
            enddo               ! igrp
          enddo                 ! it
          deallocate(evec, evec_inv, evecrot)
          zsec(:,:,iqxx) = zsect(:,:)/dble(nsym)
c          call MPI__AllreduceSum( zsec(:,:,iqxx),ntq*ntq ) ! MIZUHO-IR
 3030   continue                ! ispx
 3020 continue                  ! iq
      do procid=0,nrankv-1
        close(ifvxc__(procid) )
        close(ifevec__(procid))
      enddo
      deallocate(iblki,iblke,evaliq)
      deallocate(zsect,extp,ifevec__,ifvxc__,iprocq)
!TIME1_0110 "sub_zsecsym"
      end subroutine zsecsym
```

4.31 main/hvccfp0.m.F File Reference

Functions/Subroutines

- program hvccfp0
- subroutine checkagree (a, b, char)
- subroutine mkradmatch (p, nxdim, rdmatch)
- subroutine phimatch (p, pd, p1, p1d, p2, p2d, s, t)
- subroutine pmatorth (oo, oon, pmat, no, nn, pomat)
- subroutine diagcvh (hh, ngb, eb, zz)

- subroutine zgesvdnn2 (no, nn, nnmx, epsmx, pmat, nnn)
- subroutine mkb0 (q, lxx, lx, nxx, nx, aa, bb, nrr, nrx, rprodx, alat, bas, nbas, nbloch, b0mat)

4.31.1 Function/Subroutine Documentation

4.31.1.1 subroutine checkagree (real(8), dimension(3) *a*, real(8), dimension(3) *b*, character∗(∗) *char*)

Definition at line 1286 of file hvccfp0.m.F.

4.31.1.2 subroutine diagcvh (complex(8), dimension(ngb,ngb) *hh*, integer(4) *ngb*, real(8), dimension(ngb) *eb*, complex(8), dimension(ngb,ngb) *zz*)

Definition at line 1412 of file hvccfp0.m.F.

4.31.1.3 program hvccfp0 ( )

Definition at line 1 of file hvccfp0.m.F.

Here is the call graph for this function:

4.31.1.4 subroutine mkb0 (real(8), dimension(3) *q*, integer(4) *lxx*, integer(4), dimension(nbas) *lx*, integer(4) *nxx*, integer(4), dimension(0:lxx,nbas) *nx*, real(8), dimension(nbas) *aa*, real(8), dimension(nbas) *bb*, integer(4), dimension(nbas) *nrr*, integer(4) *nrx*, real(8), dimension(nrx,nxx,0:lxx,nbas) *rprodx*, real(8) *alat*, real(8), dimension(3,nbas) *bas*, integer(4) *nbas*, integer(4) *nbloch*, complex(8), dimension(nbloch) *b0mat*)

Definition at line 1467 of file hvccfp0.m.F.

Here is the caller graph for this function:

4.31.1.5 subroutine mkradmatch (real(8), dimension(1:2, 1:nxdim) *p*, integer(4) *nxdim*, real(8), dimension(1:nxdim,1:nxdim) *rdmatch*)

Definition at line 1296 of file hvccfp0.m.F.

Here is the call graph for this function:

Here is the caller graph for this function:

4.31.1.6 subroutine phimatch (real(8) *p*, real(8) *pd*, real(8) *p1*, real(8) *p1d*, real(8) *p2*, real(8) *p2d*, real(8) *s*, real(8) *t*)

Definition at line 1365 of file hvccfp0.m.F.

Here is the caller graph for this function:

4.31.1.7 subroutine pmatorth (complex(8), dimension(no,no) *oo,* complex(8), dimension(nn,nn) *oon,* complex(8), dimension(no,nn) *pmat,* integer(4) *no,* integer(4) *nn,* complex(8), dimension(nn,no) *pomat*)

Definition at line 1388 of file hvccfp0.m.F.

Here is the caller graph for this function:

4.31.1.8 subroutine zgesvdnn2 (integer(4) *no,* integer(4) *nn,* integer(4) *nnmx,* real(8) *epsmx,* complex(8), dimension(no,nn) *pmat,* integer(4) *nnn*)

Definition at line 1429 of file hvccfp0.m.F.

4.32 hvccfp0.m.F

```
      program hvccfp0
c- Coulomb matrix. <f_i | v| f_j>_q. ------------------------
c input files
c    HVCCIN      : some inputs by hbg0.
c    PLN         : plane wave expansion data by nbg0.
c    BASFP//atom : product basis by hbasfp0. ic=iatom should be kept!
c output
c    VCCFP : the coulomb matrix vcoul(nblochpmx,nblochpmx) for all qibz.
c-------------------------------------------------------------
c int
c    strx: structure constant for e=0 (means 1/|r-r’| )
c
      use m_readqg,only: readqg,readngmx
      use m_keyvalue,only: getkeyvalue
      use m_mpi,only: mpi__hx0fp0_rankdivider2,mpi__task,mpi__initialize,mpi__finalize,mpi__root,
     &   mpi__broadcast,mpi__dblecomplexsend,mpi__dblecomplexrecv,mpi__rank,mpi__size,
     &   mpi__ranktab,mpi__consoleout,mpi__iend,mpi__iini,mpi__getrange

      implicit none
      integer(4) :: ifvcfpout,iopen,ifhvccfp,is, nqbz, nbas,lmxcg,
     &  nband, ifplane,ngpmx, ngcmx, nblochpmx, nbloch,
     &  ibas,ic,lxx,nxx,nrx,l,n,k,isx,kdummy, iclose,
     &  nkdmx,nkqmx,lmax,nkdest,nkrest,ngp,ngc,nlxx,i,lnjcg,lnxcg,
     &  nkd,nkq ,ibas1,ibas2,nlx1,nlx2, nqibz,iqibz
      real(8) :: alat, plat(3,3),qlat(3,3),q(3),p(3),voltot,
     &  pi,fpi,tripl,alat0,epsx,
     &  tol,as,tpiba,qb0(3,3),vol0,rdist0,qdist0,radd,qadd,
     &  a0,awald,alat1,tol1,r0,q0,awald0,qg(3),   absqg2,aaa,aaa12
      integer(4),allocatable :: jcg(:),indxcg(:),
     &  lx(:),kmx(:),nblocha(:),nr(:),ificrb(:),
     &  nx(:,:),ngvecp(:,:),ngvecc(:,:),ngvecci(:,:,:),iqibzx(:)
      real(8),allocatable :: qbz(:,:),qibz(:,:),bas(:,:),rmax(:),
     &  cg(:),rprodx(:,:,:,:),dlv(:,:),qlv(:,:),work(:),ngcn(:),
     &  rojb(:,:,:), sgbb(:,:,:,:),aa(:),bb(:),rofit(:),phi(:),psi(:),
     &  wqt(:), q0i(:,:)
      complex(8) ,allocatable :: vcoul(:,:),geig(:,:),strx(:,:,:,:),
     &  sgpb(:,:,:,:),sgpp(:,:,:,:),
     &  fouvb(:,:,:,:),fouvp(:,:,:,:),vcoul0(:,:),
     &  s(:,:),sd(:,:),rojp(:,:,:) , vcoulnn(:,:)
      character*7,allocatable :: filename(:)
      character(20) :: xxt

      complex(8):: phasep,img=(0d0,1d0)
      integer(4)::ir,ig1,n1,n2

      complex(8),allocatable :: hh(:,:),oox(:,:),ooxi(:,:),oo(:,:),zz(:,:),zzr(:)
      real(8),allocatable    :: eb(:)

      complex(8),allocatable :: matp(:),matp2(:)
      complex(8) :: xxx,trwv
      integer(4) :: ngb,nev,nmx,iqx,ipl1,ipl2,nq0i,igx1,igx2
      logical checkeig
      logical:: besseltest=.false. !test
      real(8) :: sss1,sss2,dnorm
c

      complex(8),allocatable:: gbvec(:), ppovl(:,:), b0mat(:)

      integer(4) ::igc,igc0,ifgb0vec,ifgb0vec1,ix, iy

      integer(4) :: iqxini, iqxend,imode
      logical :: allochk=.false. !paralellx0=.true.,

      complex(8),allocatable:: hh1(:,:),oo1(:,:)
      integer(4)::  nqnumc,ifiqgc !bzcase,
c      character(5)  :: charnum5
c      integer(4),allocatable:: iqok(:)
      real(8):: qqq(3),qpgcut_cou       !,qq(3)

      integer(4),allocatable:: ngvecc0(:,:)
      integer(4):: ngc0

      real(8):: ginv(3,3),quu(3)

c---
      real(8),allocatable :: rkpr(:,:,:),rkmr(:,:,:),rofi(:,:)
      real(8):: eee,eees, q_org(3),screenfac
      integer(4):: ifvcfporg,nqbz_in,nblochpmx_in
      complex(8),allocatable:: vcoul_org(:,:)

      logical :: smbasis,debug=.false.,smbb
      integer(4)   :: ifprodmt,nl_r,lx_,nxx_r,nxdim,ibl1,nn,no,ngbnew,
     &  nmatch,ifpmatch,nmatch_q,ifpmatch_q,m,ifpomat,nbln,ibln,ngb_in,nnr,igc2
      character(3) :: charnum3
      character(5) :: charnum5
      character(11):: filenamep
      integer(4),allocatable:: nx_r(:), ibl(:,:,:,:)
     &  ,imatcho(:),imatchn(:),imatcho_q(:),imatchn_q(:)
      real(8),allocatable:: prodmt(:,:,:,:),rdmatch(:,:,:,:)
      complex(8),allocatable:: ppmt(:,:,:,:),pmat(:,:),pomat(:,:),oon(:,:)
      complex(8):: pval,pslo,phasex
      real(8)::absqq,qqx(3), epsmx,aaaa
      integer(4):: nnmx ,ngcnn,ngbo
cki      integer(4):: is_mix0vec ,ifgb0vec_a,ifgb0vec_b
      integer(4):: ifgb0vec_a,ifgb0vec_b , ifvcoud,idummy,nq0iadd
      logical:: is_mix0vec,wvcc !,newaniso
      character(128):: vcoudfile
      real(8),allocatable:: wqfac(:),qbzwww(:,:)
      integer:: ifiwqfac,iqbz,iqbzx,nnn,ixyz
      character(128) :: ixcc
!!------------------------------------------------------------
      call mpi__initialize()
      pi  = 4d0*datan(1d0)
      fpi = 4d0*pi
      if(mpi__root) write(6,"(' mode=0,3,202 (0 and 3 give the same results for given bas)' )")
c      call readin5(imode,iqxini,iqxend)
      if( mpi__root ) then
         read(5,*) imode
      end if
      call mpi__broadcast(imode)
      write(ixcc,"('.mode=',i4.4)")imode
      call mpi__consoleout('hvccfp0'//trim(ixcc))
      call headver('hvccfp0: start',imode)
      call cputid(0)
      if(imode==202 ) then
        write(6,*)' hvccfp0: imode=',imode
c      elseif(imode==101) then
c        write(6,*)' hvccfp0: imode=',imode
c        write(6,*)' remove_r0c is effective'
c        write(6,*)' Generate VCCFP = VCCFP.ORG - new_VCCFP'
c        ifvcfporg = iopen( "VCCFP.ORG",0,-1,0)
c      elseif(imode==102) then
c        write(6,*)' hvccfp0: imode=',imode
c        write(6,*)' remove_r0c is effective'
      elseif(imode==0) then
      elseif(imode==3) then
      else
Cstop2rx 2013.08.09 kino        stop 'hvccfp0: now hvccfp0 support just normal mode=0 3 202 101'
        call rx( 'hvccfp0: now hvccfp0 support just normal mode=0 3 202 101')
      endif
c      if(iqxini< 2) paralellx0=.false.
c      if(paralellx0) then
c        write(6,"(' PARALELL.X0 mode: iqxini iqxend=',2i3)")
c     &  iqxini, iqxend
c      endif

C --- q, nqbz, alat, qlat, nbas, bas
```

```
      ifhvccfp = iopen('HVCCIN',0,-1,0)
      read (ifhvccfp) alat, plat,qlat, nqbz, nbas, nband
      if(allochk)
     & write(*,*) 'allocate(qbz(3,nqbz),bas(3,nbas),rmax(nbas))'
      allocate(qbz(3,nqbz),bas(3,nbas),rmax(nbas))
      read(ifhvccfp) qbz, bas,rmax
      read(ifhvccfp) nqibz
      if(allochk)
     &  write(*,*)'allocate(qibz(3,nqibz),iqibzx(nqbz))'
      allocate(qibz(3,nqibz),iqibzx(nqbz))
      read(ifhvccfp) qibz(1:3,1:nqibz)
      voltot = abs(alat**3*tripl(plat,plat(1,2),plat(1,3)))
      write(6,*)' voltot=',voltot
      write(6,*)
      write(6,"(i4,3f13.6)")(i,qibz(1:3,i),i=1,nqibz)
c$$$!! Use instead of HVCCIN
c$$$      call read_bzdata()
c$$$      nwin   = 0                !Readin nw from NW file
c$$$      incwfin= 0                !use ForX0 for core in GWIN
c$$$      efin =  0d0               !readin EFERMI
c$$$      call genallcf_v3(nwin,efin,incwfin) !in module m_genallcf_v3
c$$$      call dinv33x (plat,qlat)
c$$$      allocate(rmax(nbas))
c$$$      voltot = abs(alat**3*tripl(plat,plat(1,2),plat(1,3)))
c$$$      write(6,*)' voltot=',voltot
c$$$      write(6,*)
c$$$      write(6,"(i4,3f13.6)")(i,qibz(1:3,i),i=1,nqibz)
      is = iclose('HVCCIN')

c$$$!!! 9dec2012
c$$$      ifiwqfac = iopen('WQFAC',0,-1,0)
c$$$      read(ifiwqfac) nnn
c$$$      write(6,*)'nnn nqbz=',nnn,nqbz
c$$$c      if(nnn/=nqbz) stop 'hvccfp0_sc: readin nnn WQFAC/= nqbz'
c$$$      allocate( wqfac(nnn),qbzwww(3,nnn) )
c$$$      read(ifiwqfac) wqfac,qbzwww
c$$$      ifiwqfac = iclose('WQFAC')

c$$$c-----------------------------------------------------
c$$$      iqibzx =0
c$$$      do iqibz=1,nqibz
c$$$        do iq =1,nqbz
c$$$          if( sum(abs(qibz(:,iqibz)-qbz(:,iq)))<1d-8 ) then
c$$$            iqibzx(iq)=iqibz !iqibzx is the index for iqx in bz
c$$$            goto 119
c$$$          endif
c$$$        enddo
c$$$        stop " hvccfp: cannot find iqx"
c$$$  119 enddo
c$$$c --- Readin PLN. plane wave contributions 2000 May
c$$$      ifplane = iopen('PLN',0,-1,0)
c$$$      read (ifplane) ngpmx, ngcmx
c$$$c q+G vector
c$$$      if(allochk)
c$$$     & write(*,*) 'allocate( ngcn(nqbz), ngvecci(3,ngcmx,nqibz))'
c$$$      allocate( ngcn(nqbz), ngvecci(3,ngcmx,nqibz))
c$$$      do iq=1, nqbz
c$$$        read(ifplane) ngp, ngc
c$$$        if(allochk)write(*,*)'allocate( geig, ngvecp, ngvecc)'
c$$$        allocate( geig(ngp,nband), ngvecp(3,ngp), ngvecc(3,ngc))
c$$$        read(ifplane) ngvecp, ngvecc, geig
c$$$        if(iqibzx(iq) /=0) then
c$$$          iqibz = iqibzx(iq)
c$$$          ngcn(iqibz) = ngc
c$$$          ngvecci(1:3,1:ngc,iqibz) = ngvecc(1:3,1:ngc)
c$$$        endif
c$$$        if(allochk)write(*,*) 'deallocate( geig, ngvecp, ngvecc)'
c$$$        deallocate( geig, ngvecp, ngvecc)
c$$$      enddo
c-----------------------------------------------------

c q+G vector
c$$$      ifiqgc = 1302
c$$$      open(ifiqgc, file='QGcou',form='unformatted')
c$$$      read(ifiqgc ) nqnumc, ngcmx, QpGcut_Cou
c$$$      allocate( ngcn(nqbz), ngvecci(3,ngcmx,nqibz), ngvecc(3,ngcmx),iqok(nqibz))
c$$$      iqok=1
c$$$      do iq=1, nqnumc
c$$$        read (ifiqgc) qqq, ngc
c$$$        read (ifiqgc) ngvecc(1:3,1:ngc)
c$$$        do iqibz=1,nqibz
c$$$          if( sum(abs(qibz(:,iqibz)-qqq))<1d-8 ) then
c$$$            ngcn(iqibz) = ngc
c$$$            ngvecci(1:3,1:ngc,iqibz) = ngvecc(1:3,1:ngc)
c$$$            iqok(iqibz)=0
c$$$            exit
c$$$          endif
```

```
c$$$        enddo
c$$$        if(sum(iqok)==0) exit
c$$$      enddo
c$$$      if(sum(iqok)/=0) stop 'hvccfp0: iqok/=0;wrong QGcou?'
c$$$      deallocate(ngvecc,iqok)
c$$$      close(ifiqgc)


      call readngmx('QGcou',ngcmx)
      allocate(ngvecc(3,ngcmx))
c        allocate( ngcn(nqibz), ngvecci(3,ngcmx,nqibz),ngveccc0(3,ngcmx))
c        do iqibz = 1,nqibz
c          call readqg('QGcou',qibz(:,iqibz), ngcn(iqibz),ngvecci(1,1,iqibz))
c        enddo
c        call readqg('QGcou',(/0d0,0d0,0d0/), ngc0, ngvecc0(1,1))
c        call releaseqg('QGcou')



c --- Readin BASFP//atom. The product basis functions.
      if(allochk)
     &  write(*,*)'allocte(lx,kmx,nblocha,nr,aa,bb,filename,ificrb'
      allocate(lx(nbas),kmx(nbas),nblocha(nbas),
     &         nr(nbas),aa(nbas),bb(nbas),filename(nbas),
     &         ificrb(nbas) )

      do ibas = 1,nbas
        ic = ibas !
        filename(ibas)= 'BASFP'//char( 48+ic/10 )//char( 48+mod(ic,10))
        ificrb(ibas)  = iopen( filename(ibas),1,3,0)
        read(ificrb(ibas),"(4i6,2d24.16)")
     &    lx(ibas), kmx(ibas), nblocha(ibas), nr(ibas),aa(ibas),bb(ibas)
      enddo
      lxx = maxval(lx)
      if(allochk) write(*,*) 'allocate( nx(0:lxx,nbas) )'
      allocate( nx(0:lxx,nbas) )
      do ibas = 1,nbas
        read(ificrb(ibas),"(i5)") nx(0:lx(ibas),ibas)
      enddo
      nxx = maxval(nx)
      nrx = maxval(nr)
      if(allochk) write(*,*) 'allocate( rprodx(nrx,nxx,0:lxx,nbas) )'
      allocate( rprodx(nrx,nxx,0:lxx,nbas) )

      do ibas = 1,nbas
        do l = 0, lx(ibas)
          do n = 1, nx(l,ibas)
            read(ificrb(ibas),"(3i5)"   ) k, kdummy,kdummy
            read(ificrb(ibas),"(d23.15)") (rprodx(i,n,l,ibas),i=1,nr(ibas))
cccccccccccccc
c      write(660+ibas,*)
c      write(660+ibas,'(" *** nlibas=",3i3)')  n,l,ibas
c      do i=1,nr(ibas)
c       write(660+ibas,'(2d16.8)') bb(ibas)*( exp(aa(ibas)*(i-1))- 1d0),
c     & rprodx(i,n,l,ibas)
c      enddo
cccccccccccccc
          enddo
        enddo
c      isx = iclose(filename(ibas))
      enddo


ccccccccccccccccccccccccccccccccccccccccccccccc
cccc TEST cccccccccccccccc
c      open(117, file='xin')
c      do i=1,nr(1)
c        read(117,"(d24.16)") rprodx(i,1,0,1)
c      enddo
cccccccccccccccccccccccccccccccccccccccccccccccc





ccccccccccccccccccccccccccccccccccccccccccccc
c TEST cccccccccccccccccccccccccccccccccccccc
      if(besseltest) then
        write(6,*)
        write(6,*)
        write(6,*) ' *** TEST case ***  rprodx is given by Bessel.'
ccc test G, corresponding <q+G|v|q+G> should be exact. e.g. ig1=1 and  ig1=35 for iqx=2
ccc You can change these values for tests. cccccccccccc
        iqx  = 2
        igx1 = 1
        igx2 = 35
c
```

```
        write(6,"(' iqx=',i3,' ig1 ig2=',2i3)") iqx,igx1,igx2
        write(6,"(a)")
     & ' <q+G|v|q+G> for the corresponding iqx ig1 ig2 should be exact!'
        write(6,"(a)") ' See fort.196'
        write(6,"(a)")
     & ' Errors will be from the radial function integrals !!!'
        write(6,"(a)") ' You can slso so similar test from hbasfp0.'
        write(6,"(a)") ' See test1 in basnfp0.'
c
        if(allochk) write(*,*) 'deallocate(rprodx,nx)'
        deallocate(rprodx,nx)
        tpiba=8.d0*datan(1.d0)/alat
        lx = 4
        nr = nr(1)
        aa = aa(1)
        bb = bb(1)
        lxx = maxval(lx)
        if(allochk) write(*,*)'allocate( nx(0:lxx,nbas) )'
        allocate( nx(0:lxx,nbas) )
        kmx= 1
        nx = 2
        nxx = maxval(nx)
        nblocha= nxx *(lxx+1)**2
        nrx = maxval(nr)
        if(allochk) write(*,*)'allocate(rprodx,rofi ,phi,psi) '
        allocate(rprodx(nrx,nxx,0:lxx,nbas),rofit(nrx)
     &  ,phi(0:lxx),psi(0:lxx))
        rofit(1) = 0d0
        do ir   = 1, nrx
          rofit(ir) = bb(1)*( exp(aa(1)*(ir-1)) - 1d0)
        enddo
        do n = 1, nxx
          if(n==1) ig1 = igx1
          if(n==2) ig1 = igx2
          qg(1:3) =
     &    tpiba * (qibz(1:3,iqx)+ matmul(qlat, ngvecci(1:3,ig1,iqx)))
          absqg2  = sum(qg(1:3)**2)
c
          do ir =1,nrx
            call bessl(absqg2*rofit(ir)**2,lxx,phi,psi)
            do ibas=1,nbas
              do l = 0, lx(ibas)
                rprodx(ir,n,l,ibas) = phi(l)* rofit(ir) **(l +1 )
              enddo
            enddo
          enddo
        enddo
c --- orthogonalized rprodx.
        do ibas=1,nbas
          do l = 0, lx(ibas)
            rprodx(1:nr(ibas),1,l,ibas)=
     &     rprodx(1:nr(ibas),1,l,ibas)
     &   + rprodx(1:nr(ibas),2,l,ibas)
            n = 1
            call gintxx(rprodx(1,n,l,ibas),rprodx(1,n,l,ibas)
     &    ,aa(ibas),bb(ibas),nr(ibas), aaa )
            aaa = 1d0/sqrt(aaa)
            rprodx(1:nr(ibas),n,l,ibas)= aaa*rprodx(1:nr(ibas),n,l,ibas)
            if(nxx==1) cycle
            n1=1
            n2=2
            call gintxx(rprodx(1,n1,l,ibas),rprodx(1,n2,l,ibas)
     &    ,aa(ibas),bb(ibas),nr(ibas), aaa12 )
            rprodx(1:nr(ibas),n2,l,ibas) = rprodx(1:nr(ibas),n2,l,ibas)
     &      - aaa12*rprodx(1:nr(ibas),n1,l,ibas)
            n = 2
            call gintxx(rprodx(1,n,l,ibas),rprodx(1,n,l,ibas)
     &    ,aa(ibas),bb(ibas),nr(ibas), aaa )
            aaa = 1d0/sqrt(aaa)
            rprodx(1:nr(ibas),n,l,ibas)= aaa*rprodx(1:nr(ibas),n,l,ibas)
          enddo
        enddo
      endif
cccc TEST end ccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccc


      nbloch    = sum(nblocha)
      nblochpmx = nbloch + ngcmx

c --- CG coefficienets. <LM3|lm1 lm2>
c inxcg = lm1(lm1-1)/2 + lm2 (lm1>lm2)
c Injcg = indxcg(inxcg) to indxcg(inxcg)-1
c cg(inxcg)  : = <lm3|lm1 lm2>
c jcg(lnjcg) : = lm3
      lmxcg = lxx

      call scg_sizechk(lmxcg,lnjcg,lnxcg) !(lmax,c,cindx,js)
      write(6,*)'scg_sizechk= ',lnjcg,lnxcg
c      if (lmxcg .le. 6) then
c        lnjcg = 6500
c        lnxcg = 1300
c      else if (lmxcg .le. 8) then
c        lnjcg = 22700
c        lnxcg = 3400
c      else if (lmxcg .le. 10) then
c        lnjcg = 62200
c        lnxcg = 7400
c      else
c        call rxi('setcg: cannot handle lmxcg=',lmxcg)
c      endif
c      if(allochk)
c     &  write(*,*) 'allocate(cg(lnjcg),jcg(lnjcg),indxcg(lnxcg))'
      allocate(cg(lnjcg),jcg(lnjcg),indxcg(lnxcg))
      call scg(lmxcg,cg,indxcg,jcg)
      if(allochk) write(6,*)' end of scg: cg coefficients generated.'


      call minv33(qlat,ginv)

c --- Get real-space vectors and reciprocal-space vectors for Ewald sum.
C defaults values for ewald sum
c      call lattc(awald0,tol,alat,alat,plat0,gx,gy,gz,gam,plat,qlat,
c     .   lmxst,vol,awald,w(odlv),nkd,w(oqlv),nkq,nkdmx,nkqmx,w(owork))
c- taken from lattc.f

c default values ok?
      awald0 = 2d0   !See p_lat_0
      tol    = 1d-9
      nkdmx  = 800
      nkqmx  = 800
      lmax   = 2*lxx  !lxx or lmax=6 ???

      vol0= abs(tripl(plat,plat(1,2),plat(1,3)))
      as   = awald0
c      alat0= alat
      alat1= alat
c      if(alat1.le.0.5d0) alat1=alat
      tpiba=8.d0*datan(1.d0)/alat
      call cross_x(plat(1,2),plat(1,3),qb0)
      call cross_x(plat(1,3),plat(1,1),qb0(1,2))
      call cross_x(plat(1,1),plat(1,2),qb0(1,3))
      qb0(1:3,1:3) = qb0(1:3,1:3)/vol0

      rdist0=vol0**(1.d0/3.d0)
      qdist0=1.d0/rdist0
      radd=.7*rdist0
      qadd=.7*qdist0
      a0=as/rdist0
      awald=a0/alat
cccccccccccccccccccccccccccccccccccccc
c takao
c      tol1= tol*alat**(lmax+1) *0.01
cccccccccccccccccccccccccccccccccccccccc
      tol1= tol*alat**(lmax+1)
      if(allochk) write(*,*) 'allocate(dlv, qlv, work) '
      allocate(dlv(3,nkdmx), qlv(3,nkqmx), work(max0(nkdmx,nkqmx)) )
      call lctoff(a0,vol0,lmax,tol1,r0,q0)
      nkdest =4.18879*(r0+radd)**3/vol0+.5
      nkrest =4.18879*(q0+qadd)**3*vol0+.5
      write(6,340) as,tol,lmax,awald,vol0,alat1,nkdest,nkrest
  340 format(/' lattc:  as=',f6.3,'   tol=',1p,e8.2,'   lmax=',i2,
     .  '   awald=',0p,f7.4,'   v0=',f10.3/' alat1=',f9.5,
     .  '   estimates:   nkd',i6,'   nkr',i6)
      call lgen(plat,r0+radd,nkd,nkdmx,dlv,work)
      write(6,342) r0,r0*alat,radd,nkd
  342 format('  r0=',f9.4,'   rc=',f9.4,'   radd=',f9.4,'   nkd=', i7)
      call lgen(qb0,q0+qadd,nkq,nkqmx,qlv,work)
      write(6,341) q0,q0*tpiba,qadd,nkq
  341 format('  q0=',f9.4,'   qc=',f9.4,'   qadd=',f9.4,'   nkr=', i7)
      if(allochk) write(*,*) 'deallocate(work)'
      deallocate(work)

C... readin r0c
c      if(newaniso()) then
        eee=screenfac() !takao feb2012
c      elseif(imode==101.or.imode==102) then
c        eee = eees()
c      else
c        eee=0d0
c      endif

!! for eps_lmf and epsPP_lmf mode,
!! even the small eee=1d-4 can affect to dielectric function near q=0 when its values is large as
```

```
      one-hundred or more.
!! Thus we set eee=0d0 to avoid this.
      if(imode==202) then !
         eee=0d0
      endif

      write(6,"(' Coulomb is exp(sqrt(-eee)*r)/r. eee=',d13.6,d13.6)")eee

C--- bessel and hankel for the expansion of exp(-r/r_0)/r.
c bessel and hankel is renomarized so that its behaves as r^l and r^{-l-1} near r=0.
c  rkpr means r^l*r for e=0 (r0c =infinity) case
      allocate(rkpr(nrx,0:lxx,nbas),rkmr(nrx,0:lxx,nbas),rofi(nrx,nbas))
      do ibas=1,nbas
        call genjh(eee,nr(ibas),aa(ibas),bb(ibas),lx(ibas), nrx,lxx,
     o     rofi(1,ibas), rkpr(1,0,ibas), rkmr(1,0,ibas))
      enddo

C--- onsite integrals <j(e=0)|B> and <B|v(onsite)|B>
cc      if(allochkw) write(*,*) ' allocate  rojb, sgbb '
c      allocate( rojb(nxx, 0:lxx, nbas), sgbb(nxx,  nxx,  0:lxx, nbas))
c      do ibas = 1,nbas
c        call mkjb( lxx, lx(ibas),nxx, nx(0:lxx,ibas),
c     i                 aa(ibas),bb(ibas), nr(ibas), nrx,
c     i                 rprodx(1,1,0,ibas),
c     o        rojb(1,0,ibas), sgbb(1,1,0,ibas))
c      enddo
      allocate( rojb(nxx, 0:lxx, nbas), sgbb(nxx,  nxx,  0:lxx, nbas))
      do ibas = 1,nbas
        call mkjb_4( lxx, lx(ibas),nxx, nx(0:lxx,ibas),
     i                 aa(ibas),bb(ibas), nr(ibas), nrx,
     i                 rprodx(1,1,0,ibas),
     i     rofi(1,ibas), rkpr(1,0,ibas), rkmr(1,0,ibas),
     o        rojb(1,0,ibas), sgbb(1,1,0,ibas))
      enddo

c----------------
C--- coulomb matrix for each q = qibz
c----------------
      nlxx= (lxx+1)**2
c      ngb = nbloch + ngcn(1)
      allocate(ngvecc0(3,ngcmx))
      call readqg('QGcou',(/0d0,0d0,0d0/),ginv,  quu,ngc0, ngvecc0)
      deallocate(ngvecc0)
      ngb = nbloch + ngc0
      if(allochk) write(*,*) 'allocate( vcoul)'
      allocate( vcoul(nblochpmx,nblochpmx) )
c      if(imode==101) allocate( vcoul_org(nblochpmx,nblochpmx) )

      vcoul  = 0d0

C... q near zero
      write(6,*) '--- readin Q0P -------'
      open (101,file='Q0P')
      read (101,*) nq0i,idummy,nq0iadd
      if(allochk) write(*,*)'allocate( wqt(1:nq0i),q0i(1:3,1:nq0i) )'
      allocate( wqt(1:nq0i+nq0iadd),q0i(1:3,1:nq0i+nq0iadd) )
      read (101,*)( wqt(i),q0i(1:3,i),ixyz,i=1,nq0i+nq0iadd)
      write (6,"(d13.5,3x, 3d13.5)" )( wqt(i),q0i(1:3,i),i=1,nq0i+nq0iadd)
      close(101)
      write(6,*) ' *** goto do iq nqibz+nq0iadd nq0i=',nqibz,nq0i+nq0iadd

C --- Check PARALELL.X0
c      INQUIRE (FILE = 'PARALELL.X0', EXIST = paralellx0)
c$$$      wvcc=.true.
c$$$      if(newaniso()) wvcc=.false.
      wvcc=.false.
      write(6,'(a)') " Mix0vec.XXX is not empty only when"
     &   //" the corresponding q is in Q0P with zero weight."
c      if(paralellx0) then
c        if(wvcc) ifvcfpout = iopen( "VCCFP." //xxt(iqxini,iqxend),0,-1,0)
c        ifgb0vec = iopen ( "Mix0vec."//xxt(iqxini,iqxend),1,3,0)
c        ifgb0vec1 = iopen ( "Mix0vec1."//xxt(iqxini,iqxend),1,3,0)
c      else
        iqxend = nqibz + nq0i+nq0iadd
        if(wvcc) ifvcfpout = iopen('VCCFP',0,-1,0)
        ifgb0vec = iopen( "Mix0vec",1,3,0)
        ifgb0vec1 = iopen( "Mix0vec1",1,3,0)
c      endif

      if(imode==202) then
        iqxini= nqibz + 1
c      elseif(paralellx0) then
c     &  !skip
c$$$      elseif(bzcase()==1) then
c$$$!        iqxini = 2
c$$$        iqxini = 1 !oct2005
      else
```

```
       iqxini = 1
     endif
!!
c      if(newaniso().and.imode==0) then
     if(imode==0) then
       iqxini=1
c        iqxend=nqibz ! comment out at 18nov2012
     endif
     write(6,*)'iqxini iqxend=',iqxini,iqxend
c qibz loop
c      epsx =  0.01d0
c      if(bzcase()==1) then
     if(abs(sum(qibz(:,1)**2))/=0d0) call rx( 'hvccfp0: sum(q**2)==0d0')
c      endif
     if(wvcc) write(ifvcfpout) nqbz, nblochpmx
c      if(imode==101) then
c        read(ifvcfporg) nqbz_in, nblochpmx_in
c        if(nqbz /= nqbz_in) stop 'nqbz /= nqbz_in VCCFP.ORG'
c        if(nblochpmx /= nblochpmx_in)
c     &     stop 'nblochpmx /= nblochpmx_in VCCFP.ORG'
c      endif

C... Readin PRODMT into prodmt. oct2005
     smbb = smbasis()
     write(6,*) ' smooth mixed basis=',smbb
     if(smbasis()) then
       allocate( prodmt(2,nxx,0:lxx,nbas))
       allocate( nx_r(0:lxx))
       do ibas =1,nbas
         filenamep = 'PRODMT_'//charnum3(ibas)
         ifprodmt  = iopen(filenamep,0,-1,0)
         read(ifprodmt) nl_r
         if( 2*(nl_r-1) /= lxx ) then
           write(6,*) 2*(nl_r-1),lxx
           call rx( '2*nl_r-1 /= lxx ')
         endif
         read(ifprodmt) nxx_r
         write(6,"(' nxx =',100i3)")nxx_r
         if(nxx_r>nxx) call rx( 'nxx_r>nxx')
         read(ifprodmt) nx_r(0:lxx)
         write(6,"(' nx_r=',100i3)") nx_r(0:lxx)
         lx_ = lx(ibas)
         if(sum(abs(nx(0:lx_,ibas)-nx_r(0:lx_))) /=0) then
           write(6,*)' debug: nx  =',nx(0:lx_,ibas)
           write(6,*)' debug: nx_r=',nx_r(0:lx_)
           call rx( 'nx /=nx_r')
         endif
         read(ifprodmt) prodmt(1:2, 1:nxx_r, 0:lxx, ibas)
         write(6,*)' sumcheck prodmt=',sum(abs(prodmt(:,:,:,ibas)))
         isx = iclose(filenamep)
       enddo

C... Check write for radial part of the product basis
       if(.false.) then
         do ibas= 1,1 !1,nbas
           do l   =  0,lx(ibas)
             open(1011,file='ProdOld_ibas'//charnum3(ibas)//'_l'//charnum3(l))
c         open(2011,file='ProdNew_ibas'//charnum3(ibas)//'_l'//charnum3(l))
             nxdim = nx(l,ibas)
             do ix=1,nxdim
               write(1011,"(' -- -- -- ',3i3,' --- ' )") ix,l,ibas
c         write(2011,"(' -- -- -- ',3i3,' --- ' )") ix,l,ibas
               do ir =1,nr(ibas)
                 write(1011,"(d13.5,2x,2d18.8)")
    &   rofi(ir,ibas), rprodx(ir,ix,l,ibas)
    &   , rprodx(ir,ix,l,ibas) /rofi(ir,ibas)
c         write(2011,"(d13.5,2x,2d18.8)")
c     &   rofi(ir,ibas), sum(rprodx(ir,1:nxdim,l,ibas)*rdmatch(1:nxdim,ix,l,ibas))
c     &   , sum(rprodx(ir,1:nxdim,l,ibas)*rdmatch(1:nxdim,ix,l,ibas))/rofi(ir,ibas)
               enddo
             enddo
             close(1011)
c         close(2011)
           enddo
         enddo
c       stop 'text end'
       endif
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
!
       allocate( rdmatch(nxx,nxx,0:lxx,nbas) )
       do ibas= 1, nbas
         do l   = 0, lx(ibas)
           nxdim = nx(l,ibas)
           if(nxdim<=1)write(6,*)'hvccfp0:smbasis case error nxdim <=1'
!       pval  = prodmt(1, 1:nxdim,  l,ibas)
!       pslo  = prodmt(2, 1:nxdim,  l,ibas)
!       prod(r, inew) = \sum_iold rrmat(inew,iold) * prod(r,iold)
```

```
            write(6,"('goto mkradmatch ibas lnxdim =',3i4)")ibas,l,nxdim
            call mkradmatch(prodmt(1:2, 1:nxdim, l,ibas), nxdim,
     o          rdmatch(1:nxdim,1:nxdim,l,ibas) )
          enddo
        enddo



! index (mx,nx,lx,ibas) ordering: taken from voul_4
        allocate(ibl(-lxx:lxx,nxx,0:lxx,nbas))
        ibl1 = 0
        ibl=999999
        do ibas= 1, nbas
          do l   = 0, lx(ibas)
            do n   = 1, nx(l,ibas)
              do m   = -l, l
                ibl1  = ibl1 + 1
                ibl(m,n,l,ibas) = ibl1
!       write(6,*)ibl1,n,l,m,lmbl(ibl1)
              enddo
            enddo
          enddo
        enddo
        if(ibl1/= nbloch) then
          write(6,*)' ibl1 nbloch',ibl1, nbloch
Cstop2rx 2013.08.09 kino          stop ' hvccfp0:smbasis mode  error ibl1/= nbloch'
          call rx( ' hvccfp0:smbasis mode  error ibl1/= nbloch')
        endif
! index (mx,nx,lx,ibas) ordering
ctttt
        nnr = 2 ! =2 new
        ! =0 equivalence with original mixed basis
        write(6,*)' sss:nbas lx=',nbas,lx(1:nbas)

        nbln=0
        do ibas= 1, nbas
          do l   = 0, lx(ibas)
            write(6,"('sss: nx=',3i4)") ibas,l,nx(l,ibas)
            if(nx(l,ibas)<=0) cycle
Cstop2rx 2013.08.09 kino            if(nx(l,ibas)==1) stop 'nx(l,ibas) =1'
            if(nx(l,ibas)==1) call rx( 'nx(l,ibas) =1')
ccccccccccccccccd
ctttt
c         nnr = 2 ! =2 new
c               ! =0 equivalence with original mixed basis
c         if(l<=3) nnr=0
cccccccccccccccccc
            nbln = nbln + (2*l+1)*(nx(l,ibas)-nnr)
          enddo
        enddo
        allocate( pmat(nbloch+ngcmx, nbln+ngcmx) )
        pmat=0d0
        ibln = 0
        do ibas= 1, nbas
          do l   = 0, lx(ibas)
ccccccccccccccccccc
ctttt
c         nnr = 2 ! =2 new
c               ! =0 equivalence with original mixed basis
c         if(l<=3) nnr=0
cccccccccccccccccc
            do nn  = nnr+1, nx(l,ibas) !nn=1 and nn=2 corresponds to non-zero val sol
              do m   = -l, l
                ibln = ibln +1
                nxdim = nx(l,ibas)
                pmat( ibl(m,1:nxdim,l,ibas), ibln)
     &    =  rdmatch(1:nxdim, nn, l,ibas)
ctttt
c          pmat( ibl(m,nn,l,ibas), ibln)
c     &    =  1d0
cccccccccccccccccccccccccccccccccccccccccccccccccc
              enddo
            enddo
          enddo
        enddo
C... Store matting matrix (imatchn,imatcho,pmatch)
        ifpomat   = iopen('POmat',0,-1,0)
c       write(6,*)'ttt= sumchk pmat(b)=',sum(abs(pmat(1:nbloch, 1:nbln)))
      endif
!! === open file Vcoud ===
!! This contains E(\nu,I), given in PRB81,125102

!! == main loop for iqx ==
      call mpi__getrange( mpi__iini, mpi__iend, iqxini, iqxend )
      do 1001 iqx = mpi__iini, mpi__iend ! q=(0,0,0) is omitted!
c$$$      do 1001 iqx = iqxini, iqxend ! q in IBZ. avoid q=0 case for iqx=1
        write(6,"('#### do 1001 start iqx=',i5)")iqx
```

```
       vcoudfile='Vcoud.'//charnum5(iqx)  !this is closed at the end of do 1001
       ifvcoud = iopen(trim(vcoudfile),0,-1,0)
       if(iqx > nqibz) then  !       iq = 1
         q  = q0i(:,iqx-nqibz)
c          qq = 0d0
       else                  !       iq = iqx
         q  = qibz(:,iqx)
c          qq = q
       endif
cccccccccccccccccccccccc
c        if(imode==202) then !for iqx>nqibz
c           qq=q
c        endif
cccccccccccccccccccccccc

c$$$        if(.not. newaniso() ) then !this is for fe_epsPP_lmfh_chipm feb2012
c$$$          if(sum(q**2)<1d-12) q=(/1d-4,0d0,0d0/) !takao oct2006
c$$$        endif
!! ==== q+G vector ====
       call readqg('QGcou',q,ginv,  quu,ngc, ngvecc ) !qq-->q
       ngb = nbloch + ngc  !it was ngcnn(iq)
       write(6,'(" iqx q ngc =",i5,3f10.4,i5)') iqx,q,ngc

c$$$        if(newaniso()) then
c$$$          continue
c$$$        elseif(bzcase()==1.and.iqx==1) then
c$$$          goto 1101
c$$$        endif

c        ngc = ngcn(iq)
c        ngvecc(1:3,1:ngc) = ngvecci(1:3,1:ngc,iq)
c        write(6,*)' iq ngc=',iq, ngc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  q test
c       q=(/ 0.09d0,0.09d0,0.09d0/)
c      q = q+(/ 0.01d0,0.01d0,0.01d0/)
c        q=q/4
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc

C--- strxq structure factor.
       if(allochk) write(6,*)' goto strxq'
       if(allochk) write(*,*) 'allocate( strx(nlxx,nbas,nlxx,nbas))'
       allocate( strx(nlxx,nbas,nlxx,nbas))
       do ibas1 =1,nbas
         do ibas2 =1,nbas
           p = bas(:,ibas2)-bas(:,ibas1)
           phasep =exp(img*2*pi*sum(q*p))
           nlx1 = (lx(ibas1)+1)**2
           nlx2 = (lx(ibas2)+1)**2
           if(allochk) write(*,*) 'allocate( s(nlx1,nlx2))'
           allocate( s(nlx1,nlx2),sd(nlx1,nlx2)) !kino add sd----but sd is dummy
c          call strxq(1,0d0,q,p,nlx1,nlx2,nlx1,alat,voltot,
           call strxq(1,eee,q,p,nlx1,nlx2,nlx1,alat,voltot,
    i        awald,nkd,nkq,dlv,qlv,
    i        cg,indxcg,jcg,
    o        s,sd)
           strx(1:nlx1,ibas1,1:nlx2,ibas2) = fpi*s      !!! *phasep
           if(allochk) write(*,*)'deallocate( s )'
           deallocate( s,sd )
         enddo
       enddo
ccccccccccccccccccccccccccc
c        strx=0d0
cccccccccccccccccccccccccccc

C--- onsite integrals <j(e=0)|P^(q+G)_L> and <B|v(onsite)|B>
c$$$      if(.true.) then !==New version without sgpp and fouvp allocation June2004=====
       if(allochk) write(*,*)'allocate(rojp,sgpb,fouvb)'
       allocate(  rojp(ngc,      nlxx, nbas),
    &           sgpb(ngc, nxx, nlxx, nbas),
    &          fouvb(ngc, nxx, nlxx, nbas))
c     &           sgpp(ngc, ngc, nlxx, nbas),
c     &          fouvp(ngc, ngc, nlxx, nbas) )
       do ibas = 1,nbas
         if(allochk) write(6,*)' --- goto mkjp_4',ibas
         call mkjp_4(q,ngc, ngvecc, alat, qlat,
    i      lxx, lx(ibas),nxx, nx(0:lxx,ibas),
    i      bas(1,ibas),aa(ibas),bb(ibas),rmax(ibas),
    i      nr(ibas), nrx, rprodx(1,1,0,ibas),
    i   eee, rofi(1,ibas), rkpr(1,0,ibas), rkmr(1,0,ibas),
    o      rojp(1,1,ibas),  sgpb(1,1,1,ibas),
    o      fouvb(1,1,1,ibas))
c          call mkjp3(q,ngc, ngvecc, alat, qlat,
c     i      lxx, lx(ibas),nxx, nx(0:lxx,ibas),
c     i      bas(1,ibas),aa(ibas),bb(ibas),rmax(ibas),
c     i      nr(ibas), nrx, rprodx(1,1,0,ibas),
c     o      rojp(1,1,ibas),  sgpb(1,1,1,ibas),
c     o      fouvb(1,1,1,ibas))
       enddo

C--- the Coulomb matrix
       if(allochk) write(6,*)' goto vcoulq_4'
       call vcoulq_4(q, nbloch, ngc,
    i                  nbas, lx,lxx, nx,nxx,
    i                  alat, qlat, voltot, ngvecc,
    i       strx, rojp,rojb, sgbb,sgpb, fouvb, !sgpp,fouvp,
    i       nblochpmx, bas,rmax,
    i    eee, aa,bb,nr,nrx,rkpr,rkmr,rofi,
    o          vcoul)
c       call vcoulq2(q, nbloch, ngc,
c     i                  nbas, lx,lxx, nx,nxx,
c     i                  alat, qlat, voltot, ngvecc,
c     i       strx, rojp,rojb, sgbb,sgpb, fouvb, !sgpp,fouvp,
c     i       nblochpmx, bas,rmax,
c     o          vcoul)
       if(allochk) write(6,*)' end of vcoulq_4'
       deallocate( strx, rojp,sgpb,fouvb)

c$$$      else !===old version (allocation of sgpp and fouvp are required) ====
c$$$
c$$$       if(allochk) write(*,*) 'allocate(rojp,sgpb,sgpp,fouvb,fouvp)'
c$$$       allocate( rojp(ngc,      nlxx, nbas),
c$$$     &            sgpb(ngc, nxx, nlxx, nbas),
c$$$     &           fouvb(ngc, nxx, nlxx, nbas),
c$$$     &            sgpp(ngc, ngc, nlxx, nbas),
c$$$     &           fouvp(ngc, ngc, nlxx, nbas) )
c$$$       do ibas = 1,nbas
c$$$          write(6,*)' xxx goto mkjp',ibas
c$$$          call mkjp2(q,ngc, ngvecc, alat, qlat,
c$$$     i      lxx, lx(ibas),nxx, nx(0:lxx,ibas),
c$$$     i      bas(1,ibas),aa(ibas),bb(ibas),rmax(ibas),
c$$$     i      nr(ibas), nrx, rprodx(1,1,0,ibas),
c$$$     o      rojp(1,1,ibas),  sgpb(1,1,1,ibas),
c$$$     o      fouvb(1,1,1,ibas),
c$$$     o      sgpp(1,1,1,ibas),fouvp(1,1,1,ibas) )
c$$$       enddo
c$$$c--- the Coulomb matrix
c$$$       write(6,*)' goto vcoulq'
c$$$       call vcoulq(q, nbloch, ngc,
c$$$     i                  nbas, lx,lxx, nx,nxx,
c$$$     i                  alat, qlat, voltot, ngvecc,
c$$$     i          strx, rojp,rojb, sgbb,sgpb,sgpp, fouvb,fouvp, nblochpmx,
c$$$     o          vcoul)
c$$$       if(allochk)
c$$$     &   write(*,*)'deallocate(strx, rojp,sgpb,sgpp, fouvb,fouvp)'
c$$$       deallocate( strx, rojp,sgpb,sgpp, fouvb,fouvp)
c$$$
c$$$      endif !==============================================================

c----check write
       trwv = 0d0
       do i = 1,nbloch
         trwv = trwv + vcoul(i,i)
       enddo
       write(6,'(" vcoul trwi=",i6,2d22.14)') iqx,trwv
       write(6,'("### sum vcoul(1:ngb,      1:ngb) ",2d22.14,2x,d22.14)')
    & sum(vcoul(1:ngb,1:ngb)), sum(abs(vcoul(1:ngb,1:ngb)))
       write(6,'("### sum vcoul(1:nbloch,1:nbloch) ",2d22.14,2x,d22.14)')
    & sum(vcoul(1:nbloch,1:nbloch)),sum(abs(vcoul(1:nbloch,1:nbloch)))
       write(6,*)
ccccccccccccccccccccccccccccccc
c      vcoul(:, nbloch+1:ngb)=0d0
c      vcoul(nbloch+1:ngb,:)=0d0
ccccccccccccccccccccccccccccccc

 1101   continue
       ngbo=ngb
C... Generate ppmt mattix oct2005 .......................
       if(smbasis()) then
         allocate( ppmt(2,(lxx+1)**2,nbas,ngc) )
         ppmt = 0d0
         call mkppmt(alat,plat,qlat, q,
    i    ngc, ngvecc,
    i    rmax, nbas,  bas, lx, lxx,
    o    ppmt) ! ppmt contains value and slove of e(i q+G r) at MT boundaries.
         ! ppmt(2,lmxaa,nbas)
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        write(6,*) 'lxx ppmtsum=',lxx, sum(abs(ppmt))
         write(6,*) 'nbln ngc',nbln,ngc
ccccccccccccccccccccccccccccccccccccccccccccccccccccc

C... Matching matrix pmtch. ppmt and prodmt
         pmat(:, nbln+1:nbln+ngc)=0d0
c        write(6,*) 'sss nbln ngc',nbln,ngc
```

```
          do igc=1,ngc
c          write(6,*) 'igc=',igc
            pmat(nbloch+igc, nbln+igc) = 1d0
            do ibas= 1, nbas
              do l  =  0, lx(ibas)
                do m  = -l, l
c           write(6,*) 'ibas l m=',ibas,l,m
                  pval= ppmt(1, l**2 + l+1 +m, ibas,igc)
                  pslo= ppmt(2, l**2 + l+1 +m, ibas,igc)
                  do n = 1,nx(l,ibas)
                    if(n==1.and.debug) write(6,"('ttt2: ')")
                    pmat(ibl(m,n,l,ibas), nbln+igc)
     &         =  rdmatch(n,1,l,ibas) * pval
     &         +  rdmatch(n,2,l,ibas) * pslo
                    if(debug.and.abs(pmat(ibl(m,n,l,ibas), nbln+igc))/=0d0)
     &        write(6,"('ttt2: i1 i2 pmat=',2i5,2d13.5)")
     &        ibl(m,n,l,ibas), nbln+igc, pmat(ibl(m,n,l,ibas), nbln+igc)
                  enddo
                enddo
              enddo
            enddo
          enddo
          deallocate(ppmt)
          nn = nbln  +ngc ! number for new smooth mixed basis.
          no = nbloch+ngc ! number for original size of mixed basis.
          if(debug) write(6,*) 'end of pmat'

C... oo(no,no). The original overlap matrix.
          allocate( pomat(nn,no) )
          allocate( ppovl(ngc,ngc),oo(no,no))
          call mkppovl2(alat,plat,qlat,
     i          ngc,  ngvecc,
     i          ngc,  ngvecc,
     i          nbas, rmax, bas,
     o          ppovl)
          oo = 0d0
          do ipl1 = 1,nbloch
            oo(ipl1,ipl1) = 1d0
          enddo
          do ix= 1,ngc
            do iy= 1,ngc
              oo(nbloch+ix, nbloch+iy) = ppovl(ix,iy)
            enddo
          enddo
          if(debug) write(6,*) 'end of oo'


C... oon(nn,nn) is the overlap matrix with new basis
          allocate(oon(nn,nn))
          oon = matmul( dconjg(transpose(pmat(1:no,1:nn)))
     &               ,matmul(oo,pmat(1:no,1:nn)) )


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Reduction of pmat by SVD ... not meaningful
c      nnmx = 1000000
c      epsmx= 1D20
cc      write(6,*)' sumchk pmat=',sum(abs(pmat(1:no,1:nn)))
c      call  zgesvdnn2(nn,nn, nnmx,epsmx,
c     o   oon, ! pmat is reduced to pmat(1:no,1:nnn) by SVD.
c     o   ngcnn)
cc      call  zgesvdnn2(no,ngc, nnmx,epsmx,
cc     i   pmat(1:no,nbln+1:nbln+ngc), ! pmat is reduced to pmat(1:no,1:nnn) by SVD.
cc     o   ngcnn)
cc      nn= nbln+ngcnn
c      write(6,*)' svd ngc ngcnn=',ngc, ngcnn
c      stop 'test end xxxxxx'
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

ccccccccccccccccccccccccccc
          if(.false.) then
            open(3011,file='oontest'//charnum5(iqx))
            do ix=nbln+1,nn
              igc=ix-nbln
              qqx(1:3) = (q(1:3)+ matmul(qlat, ngvecc(1:3,igc)))
              absqq  =  sqrt(sum(qqx(1:3)**2))
c           absqg2x(ix) =sum( (2*pi/alat *q0i(1:3,nq0i))**2)
              do iy=nbln+ 1,nn
                igc2=iy-nbln
                if(ix==iy) then
                  write(3011,"('on : ',2i8,3i3,2x,3i3,f13.5,3x,2f20.10)")
     &       ix,iy, ngvecc(1:3,igc),ngvecc(1:3,igc2),absqq, oon(ix,iy)
                else
                  write(3011,"('off:', 2i8,3i3, 2f20.10)")ix,iy,
     &       ngvecc(1:3,igc)-ngvecc(1:3,igc2),  oon(ix,iy)
                endif
              enddo
            enddo
            close(3011)
          endif
ccccccccccccccccccccccccccccccc


C... Generat pomat
!    zmelt_new(K, ij) = \sum_I pomat(K,I)* zmelt(I, ij)
!    means <psi_i psi_j | K> where |K> denote new mixed basis.
!  See sxcf_fal2 and x0kf.
!   Be carefull its transpose procedure---it is a little confusing...
          call pmatorth(oo,oon, pmat(1:no,1:nn), no, nn,
     o    pomat)

ccccccccccccccccccccccccccccccccccccccccccccc
ctttt
c        pomat=0d0
c        do ix= 1,ngb
c          pomat(ix,ix)=1d0
c        enddo
c        do ix= 1,ngb
c        do iy= 1,ngb
c          if(pmat(ix,iy)/=0d0 )
c     &   write(6,"(' ttt: pmat=',2i3,2d13.6)")
c     &   ix,iy,pmat(ix,iy)
c        enddo
c        enddo
c        write(6,"(' ttt:sumchk=',2d13.6,2i4)")
c     &     sum(pomat(:,:)), no,nn
cccccccccccccccccccccccccccccccccccccccccccccc

          if( iqx <= nqibz ) deallocate(oon)
          deallocate(ppovl,oo)
C... Store matching matrix
          write(ifpomat) q,nn,no,iqx
          write(ifpomat) pomat
          deallocate(pomat)
        endif

c$$$        if(newaniso()) then
c$$$          continue
c$$$        elseif(bzcase()==1.and.iqx==1)then
c$$$          cycle
c$$$        endif

!! == Write out VCCFP ==
        if(debug) write(6,*) 'write out vcoul'
        if(smbasis()) then
          ngb= nn
          allocate(vcoulnn(ngb,ngb))
          vcoulnn= matmul(transpose(dconjg(pmat(1:no,1:nn)))
     &               ,matmul(vcoul(1:no,1:no),pmat(1:no,1:nn)))
          vcoul(1:ngb,1:ngb)= vcoulnn
          deallocate(vcoulnn)
        endif
        if(wvcc) then
          write(ifvcfpout) ngb
          write(ifvcfpout) vcoul(1:ngb,1:ngb),q
        endif
        write(6,"(' ngc ngb/ngbo=',6i6)") ngc,ngb,ngbo

c Mix0vec ---------------------------------
!! diagonalize the Coulomb matrix
        if(.true.) then
c        if( iqx > nqibz .or. iqx==1) then !feb2012 add iqx==1 for newansio()=T
          if(allochk) write(*,*) 'allocate( ppovl(ngc,ngc))'
          allocate( oo(ngb,ngb) )
          allocate( ppovl(ngc,ngc) )
          call mkppovl2(alat,plat,qlat,
     &           ngc,  ngvecc,
     &           ngc,  ngvecc,
     &           nbas, rmax, bas,
     o           ppovl)
          if(smbasis()) then
            oo = oon
            deallocate(oon)
          else
            oo = 0d0
            do ipl1=1,nbloch
              oo(ipl1,ipl1) = 1d0
            enddo
            do ix=1,ngc
              do iy=1,ngc
                oo(nbloch+ix, nbloch+iy) = ppovl(ix,iy)
              enddo
            enddo
          endif
```

```

         allocate( oox(ngb,ngb) )
         oox = oo
         write(6,*)' --- goto eigen check1 --- '
         allocate(  vcoul0(ngb,ngb) )
         vcoul0 = vcoul(1:ngb,1:ngb)
         if(allochk)
     &    write(*,*) 'allocate(hh(ngb,ngb),oo(ngb,ngb),oox,zz,eb,zzr)'
         allocate(hh(ngb,ngb),zz(ngb,ngb),eb(ngb),zzr(ngb))
         hh  = - vcoul0
c          nmx = 15
         nmx = ngb
         call diagcv(oo,hh,zz,ngb, eb,nmx,1d99,nev)
         do ipl1=1,nev
           if(ipl1==11) write(6,*)' ... '
           if(ipl1>10.and.ipl1<nev-5) cycle
           write(6,'(i4,d23.16)')ipl1,-eb(ipl1)
         enddo
         write(6,"(' nev ngv q=',2i5,3f10.6)")nev,ngb,q

c$$$!! Modify -eb
c$$$          if(iqx<=nqibz) then
c$$$             do iqbz=1,nqbz     !! check
c$$$                if(sum(abs(qbzwww(:,iqbz)-q))<1d-6) then
c$$$                   iqbzx=iqbz
c$$$                   goto 888
c$$$                endif
c$$$             enddo
c$$$             stop 'hvccfp0:sum(abs(qbzwww(:,iq)-qbz(:,iq)))>1d-6'
c$$$ 888         continue
c$$$             if(abs((-eb(1)+eb(2))/eb(2))<1d-2) then
c$$$!! Center. touching case. Respect smoothness when we change n1n2n3 division.
c$$$                eb(1)=eb(1)*wqfac(iqbzx)
c$$$                eb(2)=eb(2)*wqfac(iqbzx)
c$$$             else
c$$$                eb(1)=eb(1)*wqfac(iqbzx)
c$$$             endif
c$$$          endif

!     ! === save zz === apr2012takao
c          if( newaniso().and.iqx==1 ) then
c            if(sum(q**2)>1d-10) then
c               stop ' hvccfp0: sanity check. |q(iqx)| /= 0'
c            endif
         write(ifvcoud) ngb
         write(ifvcoud) q
         write(ifvcoud) -eb
         write(ifvcoud) zz

c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$        write(6,*)' dddddddddddddddd q=',q
c$$$        do ix=1,ngb
c$$$        do iy=1,ngb
c$$$          aaaa=  sum(  dconjg(zz(1:ngb,ix))*matmul( oox,zz(1:ngb,iy))  )
c$$$           if(ix==iy .and.  abs(aaaa-1d0) >1d-8 ) then
c$$$            write(*,*)' dddd zcousum check',ix,iy,aaaa
c$$$           endif
c$$$           if(ix/=iy .and.  abs(aaaa) >1d-8 ) then
c$$$            write(*,*)' dddd zcousum check',ix,iy,aaaa
c$$$           endif
c$$$        enddo
c$$$        enddo
c$$$ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



         write(6,*)
         write(6,'(" eig0 must be equal to the largest =", 2d24.16)')
     &       sum(  dconjg(zz(1:ngb,1))*matmul( vcoul0,zz(1:ngb,1))  )
         write(6,'(" zz norm check=",d24.16)')
     &    sum( dconjg(zz(1:ngb,1))*matmul(oox,zz(1:ngb,1)) )
         write(6,*)
c          write(6,'(" --- vcoul(exact  no eee)=",d14.6," absq2=",d24.16)')
c     &    fpi*voltot/(sum(tpiba**2*q(1:3)**2))
c     &              , (sum(tpiba**2*q(1:3)**2))
         write(6,'(" --- vcoul(exact)=",d14.6," absq2=",d24.16)')
     &    fpi*voltot/(sum(tpiba**2*q(1:3)**2)-eee)
     &              , (sum(tpiba**2*q(1:3)**2)-eee)
         write(6,'(" --- vcoul(cal ) =",2d14.6)')
     &    sum( dconjg(zz(1:ngb,1))*matmul( vcoul0,zz(1:ngb,1)) )*voltot
cccccccccccccccccccccccccccccccccccccccccc
c          do igc=1,ngb
c          qqx(1:3) = (q(1:3)+ matmul(qlat, ngvecc(1:3,igc)))
c          write(6,'(" --- vcoul(exact) xxx =",d14.6," absq2=",d24.16)')
c     &    fpi*voltot/(sum(tpiba**2*(qqx(1:3)**2)-eee))
c     &              , (sum(tpiba**2*(qqx(1:3)**2)-eee))
c          write(6,'(" --- vcoul(cal ) xxx =",2d14.6)')
```

```
c     &    sum( dconjg(zz(1:ngb,igc))*matmul( vcoul0,zz(1:ngb,igc)) )*voltot
c          enddo
cccccccccccccccccccccccccccccccccccccccccc
          deallocate(vcoul0)

          if( iqx-nqibz>=1 ) then
             if( wqt(iqx-nqibz)==0d0) then ! MIZUHO-IR

C --- To get the vector <Mixed basis| q=0> --------------
cki            if(is_mix0vec()==0) then     !used original befor oct2006
            if(.not.is_mix0vec()) then     !used original befor oct2006
! See switch.F ---> this is not used now.
              ifgb0vec_a =ifgb0vec1
              ifgb0vec_b =ifgb0vec
cki            elseif(is_mix0vec()==1) then !oct2006 new case
            else
! ismix0vec=1 is to avoid problem at BZ boundary when is_mix0vec()=0.
              ifgb0vec_a =ifgb0vec
              ifgb0vec_b =ifgb0vec1
            endif
c1... Case1 to write ifgb0vec -------------------------------------
            write(6,*)' voltot=',voltot
            if(ngc==0) then
              continue
            else
              do igc=1,ngc
                if( sum(abs( ngvecc(1:3,igc) ))==0 ) then
                  igc0=igc
                  exit
                endif
              enddo
              write(6,*)' igc0=',igc0,ngvecc(1:3,igc0)
              zzr(nbloch+1:nbloch+ngc) = ppovl(1:ngc,igc0)
            endif

            allocate( gbvec(ngb), b0mat(nbloch) )
            write(6,*)' goto mkb0'

C ... get a vector <Product Basis| q+0>
            call mkb0( q, lxx,lx,nxx,nx, aa,bb,nr,nrx,rprodx,
     i        alat,bas,nbas,nbloch,
     o        b0mat)
            zzr(1:nbloch) = b0mat(1:nbloch)
cccccccccccccccccccccccccccc
c           do igc=1,ngb
c             write(6,"('ssss: ',i5,2d14.6)") igc, zzr(igc)
c           enddo
cccccccccccccccccccccccccccc
            allocate(ooxi(ngb,ngb))
            ooxi=oox
            call matcinv(ngb,ooxi)
            gbvec = matmul(ooxi, zzr)

cccccccccccccc
c           do igc=1,ngb
c             write(6,"('ssss: ',i5,2d14.6)") igc, gbvec(igc)
c           enddo
ccccccccccccccc
            deallocate(ooxi)
            dnorm = sqrt( sum(dconjg(gbvec)*zzr) )
! remove /dnorm at 14June2008. See main/hx0fp0.
! dnorm corresponds to volume (or sum of MT volume if no IPW).
c            gbvec = gbvec /dnorm
c            zzr   = zzr   /dnorm
! Not dnorm=1 at 14June2008. See main/hx0fp0.
c            dnorm=1
            write(ifgb0vec_a,"(3d24.16,2i10,d24.16)") q, ngb,igc0,dnorm
            write(ifgb0vec_a,"(4d24.16)") (gbvec(i),zzr(i),i=1,ngb)
            deallocate( gbvec, b0mat)
c1-----------------------------------------------------

c2... --- Case2 to write ifgb0vec c2 is problematic at BZ boundary...------
            dnorm  = 1d0
            zzr(:) = matmul(oox, zz(:,1))
            igc0 = 999999 !dummy now
c phasex ---just to clean. this is irrelevant
            phasex =1d0
            do i=1,ngb
              if(abs(zz(i,1)) > 1d-3) phasex = abs(zz(i,1))/zz(i,1)
            enddo
            do i=1,ngb
              zz(i,1)= phasex * zz(i,1)
              zzr(i) = phasex * zzr(i)
            enddo
            write (ifgb0vec_b,"(3d24.16,2i10,d24.16)") q, ngb,igc0,dnorm
            write (ifgb0vec_b,"(4d24.16)") (zz(i,1),zzr(i),i=1,ngb)
          endif
          endif ! MIZUHO-IR
          if(allochk) !bugfix ---this was in inside or above if 7Feb2006
     &    write(*,*)'deallocate(hh,oo,zz,eb,oox,zzr)'
          deallocate(hh,oo,zz,eb,oox,zzr)
          deallocate(ppovl)
c2---------------------
        endif
        idummy=iclose(trim(vcoudfile))
 1001 continue
      deallocate(ngvecc)
      call cputid(0)
      call flush(6)
      call mpi__finalize
      if(imode==202) call rx0( ' OK! hvccfp0 imode=202 only for Q0P')
      if(imode==0) call rx0( ' OK! hvccfp0 imode=0')
      if(imode==3) call rx0( ' OK! hvccfp0 imode=3')
      end

      subroutine checkagree(a,b,char)
      real(8):: a(3),b(3)
      character*(*) :: char
      if(sum(abs(a-b))>1d-6) then
        write(6,*)' Error in checkagree:',char
Cstop2rx 2013.08.09 kino        stop ' Error in checkagree:'
        call rx( ' Error in checkagree:')
      endif
      end

      subroutine mkradmatch( p, nxdim,
     o          rdmatch)
C- make rdmatch
C-----------------------------------------------------
Ci  p(1,i): phi     at mt for i-th basis
Ci  p(2,i): dphi/dr at mt for i-th basis
Co rdmatch(nxdim,nxdim)
C-------
Cr    phinew_j(r) =sum_i phi_i(r)* rdmatch (i,j)
Cr     phinew_1(rmt)    =1      phinew_2(rmt)   =0
Cr   d phinew_1(rmt)/dr =0    d phinew_2(rmt)/dr=1
Cr for k >=3
Cr     phinew_k(rmt)    =0
Cr   d phinew_k(rmt)/dr =0
C-----------------------------------------------------
      implicit none
      integer(4):: nxdim,lbas,i,i1,i2,ix
      real(8):: p(1:2, 1:nxdim), rdmatch(1:nxdim,1:nxdim)
      real(8):: pd,p1,p1d,p2,p2d,s,t, eps=1d-3,delta
Cr                                          old     new
c      write(6,"('mkradmatch: nxdim=',i4)") nxdim
      if(nxdim <=0) return
Cstop2rx 2013.08.09 kino      if(nxdim ==1) stop 'mkradmatch err nxdim==1'
      if(nxdim ==1) call rx( 'mkradmatch err nxdim==1')
      rdmatch=0d0
C... pivot--- get better set of phi for augmentation
      do
        i1= nxdim
        i2= nxdim-1
        p1 = p(1, i1)
        p2 = p(1, i2)
        p1d= p(2, i1)
        p2d= p(2, i2)
        write(6,"('mkradmatch: i1 p1 p1d=',i3,2d13.6)") i1,p1,p1d
        write(6,"('mkradmatch: i2 p2 p2d=',i3,2d13.6)") i2,p2,p2d
        delta = p1*p2d-p2*p1d
        if(abs(delta) <eps*p1*p2) then
          if(i2==1) then
            write(6,"(' i1 i2=',2i5,2d13.6)") i1,i2,p1d/p1,p2d/p2
Cstop2rx 2013.08.09 kino            stop'mkradmatch: err poor linear dep'
            call rx( 'mkradmatch: err poor linear dep')
          endif
          i2=i2-1
        endif
        exit
      enddo
C...
      call phimatch(1d0,0d0,  p1,p1d,p2,p2d, s,t)
      rdmatch(i1, 1)=  s
      rdmatch(i2, 1)=  t
      write(6,"('mkradmatch: 1 0    st=',2d13.5)") s,t
      call phimatch(0d0,1d0,  p1,p1d,p2,p2d, s,t)
      rdmatch(i1, 2)=  s
      rdmatch(i2, 2)=  t
      write(6,"('mkradmatch: 0 1    st=',2d13.5)") s,t

      ix=2
      do i= 1,nxdim
        if(i==i1.or.i==i2) cycle
```

```
          ix=ix+1
c          write(6,"('mkradmatch: i p pd=',i3,2d13.5)") i,p(1,i),p(2,i)
          call phimatch(p(1,i),p(2,i),  p1,p1d,p2,p2d, s,t)
          rdmatch(i,  ix)=  1d0
          rdmatch(i1, ix)=  -s
          rdmatch(i2, ix)=  -t
          write(6,"('mkradmatch: ix st=',i3,2d13.5)") ix,s,t
        enddo
        end

        subroutine phimatch(p,pd, p1,p1d,p2,p2d, s,t)
C --- match for given p and pd
c   phi = s phi1 + t phi2 !slope and value are at MT
c     p  = s p1  + t p2
c     pd = s pd1 + t pd2
        implicit none
        real(8):: matinv(2,2),p,pd,p1,p1d,p2,p2d,s,t,delta,ddd1,ddd2
        delta = p1*p2d-p2*p1d
        matinv(1,1) = 1/delta *  p2d
        matinv(1,2) = 1/delta * (-p2)
        matinv(2,1) = 1/delta * (-p1d)
        matinv(2,2) = 1/delta *  p1
        s = matinv(1,1) *p  + matinv(1,2) *pd
        t = matinv(2,1) *p  + matinv(2,2) *pd
C... check
        ddd1 = abs(s*p1  + t*p2   -  p )
Cstop2rx 2013.08.09 kino      if(  ddd1 >1d-8 ) stop 'phimatch: ddd1 err'
        if(  ddd1 >1d-8 ) call rx( 'phimatch: ddd1 err')
        ddd2 = abs(s*p1d + t*p2d  -  pd)
Cstop2rx 2013.08.09 kino      if(  ddd2 >1d-8 ) stop 'phimatch: ddd2 err'
        if(  ddd2 >1d-8 ) call rx( 'phimatch: ddd2 err')
        end

        subroutine pmatorth(oo,oon,pmat,no,nn, pomat)
C get conversion matrix from old mixed basis(no) to augmented mixed basis(nn).
C pmatorth contains
c   oo^{-1}_IJ
        implicit none
        integer(4):: no,nn,io,in,i
        complex(8):: pmat(no,nn),pomat(nn,no),oo(no,no),oon(nn,nn)
        complex(8),allocatable:: ooninv(:,:)
        real(8),allocatable:: eb(:)
        allocate(ooninv(nn,nn))
        ooninv = oon
        call matcinv(nn,ooninv) !generate ooninv
c        pomat = matmul(ooninv, matmul(dconjg(transpose(pmat)),oo))
        pomat = transpose(matmul( oo, matmul(pmat,ooninv)))
        deallocate(ooninv)
        end
c        allocate(pp(nn,nn),ppin(nn,nn),eb(nn),zz(nn,nn),zze(nn,nn))
c        ppin = pp
c        call diagcvh(ppin,nn,eb,zz)
c        do i=1,nn
c          zze(:,i) =  zz(:,i)* sqrt(eb(i))
c        enddo
c        pomat = matmul(pmat, matmul(zze,dconjg(transpose(zz))))

        subroutine diagcvh(hh,ngb,eb,zz)
        implicit none
        integer(4):: nmx,nev,i,ngb
        complex(8):: hh(ngb,ngb),oo(ngb,ngb),zz(ngb,ngb)
        real(8):: eb(ngb)
        nmx=ngb
        oo = 0d0
        do i=1,ngb
          oo(i,i) = 1d0
        enddo
        call diagcv(oo,hh,zz,ngb, eb,nmx,1d99,nev)
        write(6,*)' diagcvv: ngb,nev=',ngb,nev
        do i=1,nev
          write(6,'(i4,d23.16)')i, eb(i)
        enddo
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine zgesvdnn2(no,nn, nnmx,epsmx,
       i   pmat,
       o   nnn)
c pmat(no,nn) ---> pmat(no,nnn)
Cio input          pmat(no,nn)
Cio output reduced pmat(no,nnn)
        implicit none
        integer(4):: lwork,info,nn,no,nnn,nnmx,i
        complex(8)::  pmat(no,nn),uu(no,no),vt(nn,nn)
        real(8):: ss(nn),epsmx
        real(8),allocatable:: rwork(:)
        complex(8),allocatable:: work(:),vtt(:,:),pmatx(:,:)
c        write(6,*)' sumchk pmat=',sum(abs(pmat(1:no,1:nn)))
```

```
      lwork=4*no
      allocate(work(lwork),rwork(5*no),pmatx(no,nn))
      pmatx =pmat
      call zgesvd('A','A',no,nn,pmat,no,ss,uu,no,vt,nn,work,lwork,rwork,info)
      nnn=-999
      do i=1,nn
        write(6,"(' i ss=',i4,' ', d13.5 )")i,ss(i) !    write(6,"(' i ss=',i4,'  ', d13.5,' ss0*ss=',d13.5
 )")i,SS(i),ss(i)*ss0(ngb-i+1)
!         vtt(i,:)=ss(i)*vt(i,:)
        if(nnn==-999.and.ss(i)<epsmx) nnn = i-1
      enddo
c      write(6,*) 'nnn=',nnn
Cstop2rx 2013.08.09 kino      if(nnn==0) stop 'strange: nnn=0'
      if(nnn==0) call rx( 'strange: nnn=0')
      if(nnn>nnmx) nnn=nnmx
      pmat=pmatx
c      pmat(:,1:nnn) = uu(:,1:nnn)
!      write(6,"('sumcheck zzz  zzz-uu*s*vt=',d13.5,d13.5)")
!     &  sum(abs(zw0bk)), sum(abs(zw0bk - matmul(uu,vtt)))
!      if(abs(sum(abs(zw0bk - matmul(uu,vtt))))>1d-8*sum(abs(zw0bk)))
!     &  stop 'sumcheck zzz  zzz-uu*s*vt= error'
!      deallocate(vtt)
      end


c---------------------------------------------------------------------
      subroutine mkb0( q, lxx,lx,nxx,nx, aa,bb, nrr,nrx,rprodx,
     i        alat,bas,nbas,nbloch,
     o        b0mat)
C--make the matrix elementes < B_q | exp(iq r)>
      use m_lldata,only: ll
      implicit none
      integer(4) :: nlx,l,n,m,nr,ir,lm,ibl1,ibas,nrx,nbloch

      integer(4) :: nbas,lxx, lx(nbas), nxx, nx(0:lxx,nbas),nrr(nbas)
      real(8)    :: rprodx(nrx,nxx,0:lxx,nbas),aa(nbas),bb(nbas),
     &   phi(0:lxx),psi(0:lxx), bas(3,nbas),
     &   alat,
     &   pi,fpi,tpiba,qg1(3),q(3),absqg,r2s,a,b
c
      complex(8) :: b0mat(nbloch),img=(0d0,1d0) ,phase
c
      integer(4),allocatable:: ibasbl(:), nbl(:), lbl(:), lmbl(:)
      real(8),allocatable :: ajr(:,:),rofi(:),rob0(:,:,:)
      real(8),allocatable::cy(:),yl(:)
      complex(8),allocatable :: pjyl(:,:)
c$$$#ifdef COMMONLL
c$$$      integer(4) ll(51**2)
c$$$      common/llblock/ll
c$$$#else
c$$$      integer(4) ll
c$$$#endif

c-----
      write(6,*)'mkb0:'
      pi   = 4d0*datan(1d0)
      fpi  = 4*pi
      nlx  = (lxx+1)**2
c
      tpiba = 2*pi/alat
      qg1(1:3) = tpiba * q(1:3)
      absqg    = sqrt(sum(qg1(1:3)**2))
c
      allocate(ajr(1:nrx,0:lxx), pjyl(nlx,nbas),rofi(nrx),
     &  ibasbl(nbloch), nbl(nbloch), lbl(nbloch), lmbl(nbloch),
     &  cy(nlx),yl(nlx),rob0(nxx,0:lxx,nbas))
c
      call sylmnc(cy,lxx)
      call sylm( qg1/absqg,yl,lxx,r2s) !spherical factor Y( q+G )
c
      do ibas = 1,nbas
        a = aa(ibas)
        b = bb(ibas)
        nr= nrr(ibas)
        rofi(1)    = 0d0
        do ir      = 1, nr
          rofi(ir) = b*( exp(a*(ir-1)) - 1d0)
          call bessl(absqg**2*rofi(ir)**2,lx(ibas),phi,psi)
          do l  = 0,lx(ibas)
c ... bessel function
            ajr(ir,l) = phi(l)* rofi(ir) **(l +1 )
            ! ajr = j_l(sqrt(e) r) * r / (sqrt(e))**l
          enddo
        enddo

c ... Coefficients for j_l yl  on MT  in the expantion of of exp(i q r).
        phase = exp( img*sum(qg1(1:3)*bas(1:3,ibas))*alat  )
```

```
          do lm = 1,(lx(ibas)+1)**2
            l = ll(lm)
            pjyl(lm,ibas) = fpi *img**l *cy(lm)*yl(lm) *phase  *absqg**l
          enddo
c ... rob0
          do l = 0,lx(ibas)
            do n = 1,nx(l,ibas)
              call gintxx( ajr(1,l), rprodx(1,n,l,ibas), a,b,nr,
     o                 rob0(n,l,ibas) )
            enddo
          enddo
        enddo

c ... index (mx,nx,lx,ibas) order.
        ibl1 = 0
        do ibas= 1, nbas
          do l   = 0, lx(ibas) ! write(6,'(" l ibas nx =",3i5)') l,nx(l,ibas),ibas
            do n   = 1, nx(l,ibas)
              do m   = -l, l
                ibl1  = ibl1 + 1
                ibasbl(ibl1) = ibas
                nbl(ibl1) = n
                lbl(ibl1) = l
                lmbl(ibl1) = l**2 + l+1 +m ! write(6,*)ibl1,n,l,m,lmbl(ibl1)
              enddo
            enddo
          enddo
        enddo
c ... pjyl * rob0
        do ibl1= 1, nbloch
          ibas= ibasbl(ibl1)
          n   = nbl(ibl1)
          l   = lbl(ibl1)
          lm  = lmbl(ibl1)
          b0mat(ibl1) = pjyl(lm,ibas) * rob0(n,l,ibas)
        enddo
        deallocate(ajr, pjyl,rofi,
     &  ibasbl, nbl, lbl, lmbl,
     &  cy,yl,rob0)
        end
```

4.33 main/hx0fp0.m.F File Reference

Functions/Subroutines

- program hx0fp0
- real ∗8 function eclda_bh (rs)
- real ∗8 function eclda_pz (rs)
- subroutine wecqw (ifcor,
- subroutine getsqovlp (q, ngc, ngb, sqovlp)
- subroutine tr_chkwrite (tagname, zw, iw, freqq, nblochpmx, nbloch, ngb, iq)
- complex(8) function, dimension(1) matcinvf (a)
- subroutine diagno00 (nbloch, wpvc, eval)

4.33.1 Function/Subroutine Documentation

4.33.1.1 subroutine diagno00 (integer *nbloch,* complex(8), dimension(nbloch,nbloch) *wpvc,* real(8), dimension(nbloch) *eval*)

Definition at line 2372 of file hx0fp0.m.F.

4.33.1.2 real∗8 function eclda_bh (real(8) *rs*)

Definition at line 2204 of file hx0fp0.m.F.

Here is the caller graph for this function:

4.33.1.3 real∗8 function eclda_pz (real(8) *rs*)

Definition at line 2213 of file hx0fp0.m.F.

Here is the caller graph for this function:

4.33.1.4 subroutine getsqovlp (real(8), dimension(3) *q,* integer *ngc,* integer *ngb,* complex(8), dimension(ngb,ngb) *sqovlp*)

Definition at line 2254 of file hx0fp0.m.F.

Here is the call graph for this function:

4.33.1.5 program hx0fp0 ( )

Definition at line 5 of file hx0fp0.m.F.

Here is the call graph for this function:

4.33.1.6 complex(8) function, dimension(1) matcinvf (complex(8), dimension(:,:) *a*)

Definition at line 2347 of file hx0fp0.m.F.

4.33.1.7 subroutine tr_chkwrite (character∗(∗) *tagname,* complex(8), dimension(nblochpmx,nblochpmx) *zw,* integer *iw,* real(8) *freqq,* integer *nblochpmx,* integer *nbloch,* integer *ngb,* integer *iq*)

Definition at line 2308 of file hx0fp0.m.F.

4.33.1.8 subroutine wecqw (*ifcor*)

Definition at line 2223 of file hx0fp0.m.F.

Here is the caller graph for this function:

4.34 hx0fp0.m.F

```
00001 !!  Calculate x0, \epsilon, spin susceptibility.
00002 !!
00003 !! eps_lmf_cphipm mode is now commented out; you may need to recover this if necessary
00004 !! (only epsPP_lmf_chipm mode works).
00005       program hx0fp0
00006       use m_readefermi,only: readefermi,ef
00007       use m_readqg,only: readqg,readngmx
00008       use m_readeigen,only: readeval,init_readeigen,init_readeigen2
00009       use m_read_bzdata,only: read_bzdata,
00010      & ngrp2=>ngrp,nqbz,nqibz,nqbzw,nteti,ntetf,n1,n2,n3,qbas,ginv,
00011      & dq_,qbz,wbz,qibz,wibz,qbzw,
00012      & idtetf,ib1bz,idteti,
00013      & nstar,irk,nstbz
```

```fortran
      use m_genallcf_v3,only: genallcf_v3,
     & nclass,natom,nspin,nl,nn, ngrp,
     & nlmto,nlnmx, nctot,niw, !nw_input=>nw,
     & alat, delta,deltaw,esmr,symgrp,clabl,iclass, !diw,dw,
     & invg, il,in,im,nlnm,
     & plat, pos,ecore, symgg
      use m_keyvalue,only: getkeyvalue
      use m_pbindex,only: pbindex !,norbt,l_tbl,k_tbl,ibas_tbl,offset_tbl,offset_rev_tbl
      use m_readqgcou,only: readqgcou
      use m_mpi,only: mpi__hx0fp0_rankdivider2,mpi__task,mpi__initialize,mpi__finalize,mpi__root,
     & mpi__broadcast,mpi__dblecomplexsend,mpi__dblecomplexrecv,mpi__rank,mpi__size,
     & mpi__ranktab,mpi__consoleout,mpi__barrier
!! Base data to generate matrix elements zmel*. Used in "call get_zmelt".
      use m_rdpp,only: rdpp, !NOTE: "call rdpp" generate following data.
     & nblocha,lx,nx,ppbrd,mdimx,nbloch,cgr
!! Generate matrix element for "call get_zmelt".
      use m_zmel,only:       !NOTE: these data set are stored in this module, and used when
     & nband,itq,ngcmx,ngpmx, ppovlz,
     & ppbir,shtvg, miat,tiat , ntq
!! frequency
      use m_freq,only: getfreq, !NOTE: call getfreq generate following data.
     & frhis,freq_r,freq_i, nwhis,nw_i,nw,npm,wiw !, frhis0,nwhis0 !output of getfreq
!! tetwt
      use m_tetwt,only: tetdeallocate,gettetwt, !followings are output of
 'L871:call gettetwt')
     &  whw,ihw,nhw,jhw,ibjb,nbnbx,nhwtot,n1b,n2b,nbnb
!! w0 and w0i (head part at Gamma point)
      use m_w0w0i,only: w0w0i,
     & w0,w0i

      use m_lldata,only: ll
      implicit none
!! ------------------------------------------------
!! We calculate chi0 by the follwoing three steps.
!!  gettetwt: tetrahedron weights
!!  x0kf_v4h: Accumlate Im part of the Lindhard function. Im(chi0) or Im(chi0^+-)
!!  dpsion5: calculate real part by the Hilbert transformation from the Im part
!!  eibz means extented irreducible brillowin zone scheme by C.Friedlich. (not so efficient in cases).
!!------------------------------------------------

cccccc this may be wrong or correct ccccccccc
cr be careful for the indexing...
cr      a routine idxlnmc(nindxv,nindxc,...  in index.f
cr      specifies the order of the(core wave)+(argumentation wave) in each mt.
cr      the total number of the wave are mnl(ic)= mnlc(ic) + mnlv(ic).
cr      the indexing starts with core first and then valence on top of core
cr      so n-index in "in" for valence electron is different from "inv".
cccccccccccccccccccccccccccccccccccccccccccccccccc
      real(8):: q(3),  qgbin(3),qx(3)
      real(8):: ua=1d0 ! this is a dummy.
      integer:: ifrb(2),ifcb(2),ifrhb(2),ifchb(2) !,ifev(2)
      integer:: ndble=8
      integer:: nword
      real(8),allocatable:: vxcfp(:,:),
     &  wqt(:), wgt0(:,:),q0i(:,:)
      integer,allocatable :: ngvecpb(:,:,:),ngveccb(:,:) !,ngveccB(:,:,:)
     &           , ngvecp(:,:), ngvecc(:,:), !,ngpn(:),ngcni(:),iqib(:),
     &   ifppb(:)   !ongveccBr(:,:,:),nx(:,:),nblocha(:),lx(:),
      complex(8),allocatable:: geigb(:,:,:,:) ,geig(:,:),vcoul(:,:),
     &  zw(:,:),zw0(:,:),
     &  zxq(:,:,:),zxqi(:,:,:)
      real(8),allocatable :: eqt(:), !ppbrd (:,:,:,:,:,:,:),cgr(:,:,:,:),
     &  ppbrdx(:,:,:,:,:,:,:),aaa(:,:),symope(:,:),
     &  ppb(:,:),pdb(:,:),dpb(:,:),ddb(:,:),
     &  qbze(:,:),qibze(:,:)  !,ecore(:,:)  freqr(:),freqi(:) !rw(:,:),cw(:,:) --->zw
      complex(8),allocatable :: trwv(:),trwv2(:),rcxq(:,:,:,:)
c     & ,rcxqmean(:,:,:,:),rcxqmeanc(:,:,:,:) !now rcxqmean is treated as a case of rcxq(nmbas,nmbas)

!  tetrahedron method
      logical :: tetra !,tmpwwk=.true.! If tmpwwk=.true., this use a temporary file tmp.wwk
      ! so as to reduce the memory usage.
      complex(8) :: fff,img=(0d0,1d0)
      complex(8),allocatable :: wwk(:,:,:)
      integer,allocatable ::
     &         noccxvv(:) !n1b(:,:,:),n2b(:,:,:),nbnb(:,:),nbnbtt(:,:),
      real(8) ::qbzx(3),anfvec(3)
      logical :: debug
      integer,allocatable:: ibasf(:)
      real(8),allocatable :: transaf(:,:)
      logical :: realomega, imagomega
      complex(8),allocatable:: epsi(:,:),gbvec(:),zzr(:,:),x0mean(:,:,:),zzr0(:)
      complex(8) :: epxxx,vcmean, vcmmmm
      complex(8),allocatable:: vcmmm(:)
      character*11 fileps
      character*11 fileps23
      character*16 filepsnolfc
      character*11  filele
      character(5) :: charnum5
      character(20):: xxt

      real(8) :: emin, emax,emin2,emax2
      real(8) :: omg2max,omg1max,wemax
      real(8), allocatable :: freqr2(:)  , ekxxx(:,:,:)

      logical::imagonly=.false.,realonly=.false. !,readgwinput
      integer::iopen,maxocc2,iclose,
     & ixc,iqxini,iqxend,iqxendx,
     &   ifhbe,
     &   nprecb,mrecb,mrece,nlmtot,nqbzt,!nband,
     &   nq0i,i,nq0ix,neps,ngrpmx,mxx,nqbze,nqibze,ini,ix,ngrpx !ngcmx,
     &   ,nblochpmx,ndummy1,ndummy2,ifcphi,is,nwp, !ifvcfpout,,mdimx,nbloch
     &   ifepscond,nxx !,ifvxcpout,ifgb0vec
     &   ,nw0,iw,ifinin,iw0,ifwwk,noccxv,noccx
     &   ,nprecx,mrecl,ifwd,ifrcwi,ifrcw,nspinmx,ifianf,ibas
     &   ,ibas1,irot,iq,ngb,iqixc2,ifepsdatnolfc,ifepsdat,ngbin,igc0dummy
     &   ,kx,isf,kqxx,kp,job,noccxvx(2)=-9999,nwmax  !,ifev1,ifev2 nbnbx,nhwtot,
     &   ,ihis,jhwtot,ik,ibib,ib1,ib2,ichkhis,ihww,j,imode
c     &   ,ngpmx !,  ifchipmlog

      real(8):: dum1,dum2,dum3,wqtsum,epsrng,dnorm,
     & dwry,dwh,omg_c,omg2

      integer:: incwfin,  verbose

      integer:: ngc,mrecg !bzcase,
      real(8):: quu(3), deltaq(3)!,qq(3) !,qqq(3)=0d0
      logical:: omitqbz=.false., noq0p

      logical,allocatable :: iwgt(:,:,:,:)
      complex(8),allocatable:: wgt(:,:,:)

      real(8),allocatable:: qbz2(:,:)
      logical :: qbzreg !if true, we use off-gamma mesh.
      integer:: nbcut,nbcut2

      integer,allocatable:: nstibz(:) !Nov2004 Miyake's tote
      real(8),allocatable:: ecqw(:,:) !,wiw(:)
      real(8) :: erpaqw, trpvqw, trlogqw,rydberg,hartree
     &   ,pi,efz,qfermi,alpha,rs,voltot,ecelgas,efx,valn
      integer:: iqbz,iqindx,iflegas,nmx
     &   ,ifcor,nqitot,isx,ntot,ieclog,iww,iqq,ieceig,ecorr_on=-1
      real(8) :: eclda_bh,eclda_pz,wk4ec,faca
      real(8),allocatable::    evall(:)
      complex(8),allocatable:: ovlpc(:,:),evecc(:,:)
      integer:: nev !,  ifdpin

      real(8),allocatable:: ecut(:),ecuts(:) ,totexc(:), trpv(:),trlog(:)
      integer:: necut,iecut

      integer:: ifv,lxx,ibasx,ilmx,ilm_r,nx_r,lb,nb,mb
      integer,allocatable:: nxx_r(:)
      real(8),allocatable:: svec(:,:),spinvec(:,:),consvec(:,:),cvec(:,:)
      character*3:: charnum3
      character*4:: charnum4
      complex(8),allocatable:: jcoup(:,:), mcm(:,:,:)
      real(8)::chg1,chg2,spinmom,schi=1d0
c$$$#ifdef COMMONLL
c$$$      integer::ll(51**2)
c$$$      common/llblock/ll
c$$$#else
c$$$      integer :: ll
c$$$      external ll
c$$$#endif
      complex(8),allocatable:: ovlp(:,:),evec(:,:),ovlpi(:,:)
      real(8),allocatable::eval(:)
      integer:: new,nmxx,ii,iy,ipl1,ixx

      complex(8),allocatable :: ppovl(:,:),oo(:,:),x0meanx(:,:),x0inv(:,:),ppovlzinv(:,:)
      real(8)::qxx(3),ssm
! svd. not used now
      real(8),allocatable::ss(:),rwork(:),ss0(:)
      complex(8),allocatable:: uu(:,:),vt(:,:),work(:),zw0bk(:,:),ddd(:,:)
     & ,vtt(:,:),zzz(:,:),sqsvec(:),ooo(:,:),ppo(:,:) !,sqovlp(:,:),sqovlpi(:,:)
      integer::lwork,info,imin,ifzxq
      complex(8)::x0mx
      complex(8),allocatable:: uu0(:,:),vt0(:,:)

      logical ::  chipm=.false.,nolfco=.false. !sergeyv only ngczero=.false.,
     &  ,epsmode=.false.,normalm=.false., eiqr=.false.
      integer::  ife, idum4 !ifchipmn,ifchipm,
      real(8):: qs,qt,ww,muu, ddq(3)
      character*11 ::ttt
      integer:: nnmx,nomx

```

```fortran
! Feb2006 time-reversal=off case
      logical :: timereversal, testtimer,onceww
      integer:: jpm,ncc
      real(8):: frr

      integer:: ipm,nrecoff

      real(8),allocatable:: ebb(:)
      logical :: evaltest !for a debug test
      character*300:: aline
      integer:: istat,nmbas,imb,imb1,imb2,nmbas_in
      integer,allocatable:: imbas(:), imbas_s(:),iibas(:)
!...
      complex(8),allocatable:: am1(:),am2(:),mmat(:,:),
     &     x0mat(:,:),x0matinv(:,:),eiqrm(:)
      integer:: ifchipmn_mat, ifchipm_fmat !,ifchipm_mat
      integer::ifstoner,ifx,i1
      real(8):: istoner,zz1,zz2,zz3,zz4,istoner0,jzero2,dumm1,dumm2
      complex(8):: trr,trr0,trr1     , zzzx(4,4), zzzy(4,4),trrx,mmatx(4,4),denom(4,4)
      real(8),allocatable:: eee(:),mmnorm(:),
     &    asvec(:,:),ssv(:,:),sproj(:,:),sprojx(:,:), momsite(:)
      real(8):: eex(4),eey(4),qvv(3)
!!
c      logical :: newaniso,newaniso2,newanisox !,z1offd
      integer :: ngb0,ifvcoud,idummy,ifepstinv,igb1,igb2,ngb_in,nmbas1,nmbas2,iq0,ifisk,iqx,ig,nmbas1x,
ifiss,iq0x
      complex(8),allocatable:: zcousq(:,:),epstinv(:,:),epstilde(:,:),zcousqrsum(:,:,:),zcousqr(:,:)
      real(8),allocatable:: vcousq(:)
      real(8):: fourpi,sqfourpi,tpioa,absq,vcou1,vcou1sq

!! Eq.(40) in PRB81 125102
c      complex(8),allocatable::sk(:,:,:),sks(:,:,:),ski(:,:,:),sksi(:,:,:),
c     &  w_k(:,:,:),w_ks(:,:,:),w_ki(:,:,:),w_ksi(:,:,:), llw(:,:), llwi(:,:),
      complex(8),allocatable::sk(:),sks(:),ski(:),sksi(:),
     &  w_k(:),w_ks(:),w_ki(:), w_ksi(:), s_vc(:),vw_k(:),vw_ks(:)
      complex(8),allocatable:: llw(:,:), llwi(:,:),aaamat(:,:)
      integer:: lxklm,nlxklm,ifrcwx,iq0xx,ircw,nini,nend,iwxx,nw_ixxx,nwxxx,niwxxx,iwx,icc1,icc2
      complex(8):: vc1vc2
      integer,allocatable:: neibz(:),nwgt(:,:),ngrpt(:),igx(:,:,:),igxt(:,:,:),eibzsym(:,:,:)

      real(8),allocatable:: aik(:,:,:,:)
      integer,allocatable:: aiktimer(:,:)
      integer:: l2nl
      logical:: eibz4x0,tiii,iprintx,symmetrize,eibzmode
      real(8):: qread(3),imagweight

      character(128):: vcoudfile
      integer:: src,dest
      logical:: crpa,lqall
      integer,allocatable :: iclasst(:), invgx(:)
      integer:: ificlass,ifile_handle,k
      complex(8),allocatable:: ppovl_(:,:)

      logical:: readw0w0itest=.false.

      real(8)::ebmx
      integer:: nbmx,mtet(3)
      real(8),allocatable:: ekxx1(:,:),ekxx2(:,:)

!! ------------------------------------------------------------------
      call mpi__initialize()
      call mpi__consoleout('hx0fp0')
      call cputid(0)
      hartree  = 2d0*rydberg()
      pi       = 4d0*datan(1d0)
      fourpi   = 4d0*pi
      sqfourpi = sqrt(fourpi)
!! computational mode select
c takao keeps only the sergey mode.
      write(6,"(a)") '--- Type numbers #1 #2 #3 [#2 and #3 are options] ---'
      write(6,"(a)") ' #1:run mode'
      write(6,"(a)") '    11  : normal    Sergey'
      write(6,"(a)") '    202 : epsNoLFC  Sergey'
      write(6,"(a)") '    203 : eps       Sergey'
      write(6,"(a)") '    222 : chi^+- NoLFC Sergey'
      write(6,"(a)") '    223 : chi^+- Sergey'
      write(6,"(a)") '    12  : total energy Miyake Sergey'
      write(6,"(a)") '    -9999: just show version num'
c      write(6,"(a)") ' #2=iqxini   #3=iqxend' '
c      write(6,"(a)") '  10222 : <e^{iqr}|chi^+-|e^{iqr}> nolfc'
      write(6,"(a)")  '-------------------------------------------------------'
      if( MPI__root ) then
         read(5,*) ixc
      endif
      call MPI__Broadcast(ixc)
      call headver('hx0fp0',ixc)
      call cputid(0)
      crpa=.false.
.or..or.      if(ixc<=6ixc==22ixc==23) then
        write(6,*)’these modes are removed now’
        call rx( ’these modes are not supported’)
! Sergey (Hilbert-transformation) modes
      elseif(ixc==11) then; write(6,*) " OK ixc=11 normal mode "
        normalm=.true.
      elseif(ixc==111) then; write(6,*) " OK ixc=111 normal mode. fullband"
        normalm=.true.
      elseif(ixc==10011) then; write(6,*) " OK ixc=10011 crpa mode "
        normalm=.true.
        crpa=.true.
!     -- eps mode NoLFC
      elseif(ixc==202) then
        write(6,*) " OK ixc=202  sergey’s eps mode only nolfc "
        realonly=.true.
c        iepsmode=202
        omitqbz=.true.
!     -- eps mode with LFC
      elseif(ixc==203) then
        write(6,*) " ok ixc=203 sergey’s eps mode with LFC "
        realonly=.true.
c        iepsmode=203
        omitqbz=.true.
! Total energy modes
      elseif(ixc==12) then
        write(6,*) " ixc=12 Miyake’s total energy sergey--->need to fix this mode"
        call rx( " ixc=12 miyake’s total energy Sergey--->need to fix this mode")
        imagonly=.true.
        ecorr_on=901
!     -- chipm mode NoLFC
      elseif(ixc==222) then
        write(6,*) " OK ixc=222    chipm sergey’s "
        realonly=.true.
        omitqbz=.true.
        eiqr =.false. ! .true. aug2012
!     -- chipm mode NoLFC
c      elseif(ixc==10222) then
c        write(6,*) " ok ixc=10222  <q|chipm_0|q> sergey"
c        sergeyv=.true.
c        realonly=.true.
c        omitqbz=.true.
c        eiqr =.true.
!     -- eps mode with LFC
      elseif(ixc==223) then
        write(6,*) " ixc=223  chipm with lfc sergey’s -->commented out not. need to fix this mode if necessary."
        call rx( " ixc=223  chipm with LFC sergey’s -->commented out not. need to fix this mode if necessary.")
        realonly=.true.
        omitqbz=.true.
        eiqr =.true.
      else
        call rx( ’ hx0fp0: mode ixc is not appropriate’)
      endif

.or..or..or.      if(ixc==202ixc==203ixc==222ixc==223) then
        epsmode = .true.
.or.        if(mod(ixc,200)==22mod(ixc,200)==23) chipm =.true.
        if(mod(ixc,10)==2)                       nolfco=.true.
      endif

C ... files for RPA correlation energy mode.
      if(ecorr_on > 0) then
        ieclog = 8155
        if(ecorr_on==901) then
          ieceig=8156
          open(ieceig,file=’rpa_eigen.chk’)
          close(ieceig,status=’delete’)
        endif
        open(ieclog, file=’ecorr.chk’)
      endif
!! ====newaniso2====
c$$$      newaniso2=.false.
c$$$      if(newaniso()) then
c$$$        newaniso2=.true.
c$$$      endif

!! naraga says this cause a stop in ifort --->why???
c      write(6,*)’Timereversal=’,Timereversal()

!! Readin BZDATA. See m_read_bzdata in gwsrc/rwbzdata.f
      call read_BZDATA()

!! read bzdata; See use m_read_bzdata,only:
!! Use off-regular mesh for qbzreg()=F See hx0fp0.m.sc.F also.
!! This must be consistent with qg4gw.F-mkqg.F
```

```
.not.      if(qbzreg()) then
         deltaq= qbas(:,1)/n1 + qbas(:,2)/n2 +qbas(:,3)/n3
         do i=1,nqbz
            qbz(:,i) = qbz(:,i) - deltaq/2d0
            write(6,"('i qbz=',i3,3f8.4)") i,qbz(:,i)
         enddo
      endif
      write(6,"(' nqbz nqibz ngrp=',3i5)") nqbz,nqibz,ngrp

C --- Use regular mesh even for bzcase==2 and qbzreg()=T
! A little confusing...
c      ddq = 0d0
c      if(bzcase()==2) ddq= dq_
c      do iq = 1, nqbz
c        qbz(1:3,iq) = qbz(1:3,iq) + ddq
c        ! This new qbz is regular mesh, which are identical in the both bzcase.
c      enddo
.not.c      if(qbzreg()) then ! off-regular mesh case
c        do i=1,nqbz
c          qbz(:,i) = qbz(:,i) - dq_
c        enddo
c      endif
      if(MPI__root) then
         do i=1,nqbz
.or.            if(i<10i>nqbz-10) write(6,"('i qbz=',i8,3f8.4)") i,qbz(:,i)
.and.            if(i==10nqbz>18) write(6,"('... ')")
         enddo
         write(6,*)' nqbz nqibz =',nqbz,nqibz
      endif

c$$$!!- oct2005 not implimented cases.
.and.c$$$      if(smbasis()chipm) then
c$$$        write(6,*)' smbasis=T & chipm=T is not implimented yet.'//
c$$$     &       ' Supply consistent MixSpin for smbasis!'//
c$$$     &       ' MixSpin should be converted at the end of hvccfp0.'
c$$$        call rx( ' smbasis=T & chipm=T is not implimented yet.')
c$$$      endif

c      call getkeyvalue("gwinput","scaledgapx0",sciss,default=1d0)
c      write(6,"(' ScaledGapX0=',f5.3)") sciss

!! === Readin by genallcf ===
!! See "use m_genallcf_v3" at the begining of this routine
!! We set basic data.

c$$$      if(epsmode) then
c$$$        nwin = -999
c$$$      else
c$$$        nwin = 0      !Readin nw from NW file
c$$$      endif
      incwfin= 0  !use ForX0 for core in GWIN
c      efin =  0d0 !readin EFERMI
cc--- EFERMI
c      ifief=ifile_handle()
c      open(ifief,file='EFERMI')
c      read(ifief,*) ef
c      close(ifief)
      call readefermi()
      write(6,"(a,f12.6)")' --- READIN ef from EFERMI. ef=',ef
      call genallcf_v3(incwfin) !in module m_genallcf_v3
      if(ngrp/= ngrp2) call rx( 'ngrp inconsistent: BZDATA and LMTO GWIN_V2')
      tpioa=2d0*pi/alat

.and.      if(chipmnspin==1) call rx( 'chipm mode is for nspin=2')
      debug=.false.; if(verbose()>=100) debug=.true.
      if(debug) write(6,*)' end of genallc'
c      write(6,"(' ncore=',i4)") ncore
c      write(6,*) 'nw_input delta=',nw_input,delta

!!!! WE ASSUME iclass(iatom)= iatom !!!!!!!!!!!!!!!!!!!!!!!!!
!!!!  We assume nclass = natom.     !!!!!!!!!!!!!!!!!!!!!!!!!
      if(nclass /= natom) call rx( ' nclass /= natom ')

!! --- tetra or not
      if(delta <= 0d0) then
c        tetra =  .true.
        delta = -delta
        write(6,*)' hx0fp0: tetrahedron mode delta=',delta
      else
c        tetra = .false. ! switch for tetrahedron method for dielectric functions
        call rx(' hx0fp0: only tetra=T support')
      endif

!! --- read dimensions of h,hb
      ifhbe  = iopen('hbe.d',1,0,0)
      read (ifhbe,*) nprecb,mrecb,mrece,nlmtot,nqbzt,nband,mrecg !warn nband is in m_zmel
      is = iclose('hbe.d')
      if(nlmto/=nlmtot) call rx('hx0fp0: nlmto/=nlmtot in hbe.d')
c      if(nqbz /=nqbzt ) call rx('hx0fp0: nqbz /=nqbzt  in hbe.d')

!! --- Readin Offset Gamma --------
      if(debug) write(6,*) 'reading QOP'
      open (101,file='Q0P')
      read (101,"(i5)") nq0i
      write(6,*) ' ### nqibz nq0i=', nqibz,nq0i
      allocate( wqt(1:nq0i),q0i(1:3,1:nq0i) )
      do i=1,nq0i
        read (101, * ) wqt(i),q0i(1:3,i)
      enddo
      nq0ix = nq0i
      do i=1,nq0i
        if(wqt(i)==0d0 ) then
          nq0ix = i-1
          exit
        endif
      enddo
      neps = nq0i - nq0ix  ! number of zero weight q0p which are used for ixc=2 or 3 mode.
      write( 6,*) ' num of zero weight q0p=',neps
      write(6,"(i3,f14.6,2x, 3f14.6)" )(i, wqt(i),q0i(1:3,i),i=1,nq0i)
      close(101)
.not.c$$$      if(newaniso2) then
c$$$        wqtsum = sum(abs(wqt(1:nq0i)))
c$$$        call getkeyvalue("gwinput","testnoq0p",noq0p,default=.false.)
.and..and..not.c$$$        if(normalmabs(wqtsum-1d0) >1d-10(noq0p))
c$$$     &  call rx( ' wqtsum of Q0P /=1 ')
c$$$      endif

C --- readin by rdpp ; Radial integrals ppbrd and plane wave part
      call getsrdpp2( nclass,nl,nxx)
      call readngmx('QGpsi',ngpmx)
      call readngmx('QGcou',ngcmx)
      write(6,*)' ngcmx ngpmx=',ngcmx,ngpmx
! qibze(3,nqbze) qbze(3,nqibze)
      nqbze  = nqbz *(1 + nq0i)
      nqibze = nqibz + nq0i
      allocate( qbze(3, nqbze), qibze(3, nqibze))
      qbze(:,1:nqbz)   = qbz(:,1:nqbz)
      qibze(:,1:nqibz) = qibz(:,1:nqibz)
      do i = 1,nq0i
        qibze(:,nqibz+i)  = q0i(:,i)
        ini = nqbz*(1 + i -1)
        do ix=1,nqbz
          qbze (:,ini+ix)   = q0i(:,i) + qbze(:,ix)
          if( abs(qbze(1,ini+ix)+0.1d0)+abs(qbze(2,ini+ix)+0.1d0)<1d-6 ) then
            write(6,"('aaaaaa qbze=',i8,3f18.14,2x,3f14.10)") ini+ix,qbze(:,ini+ix),q0i(:,i)
          endif
        enddo
      enddo
      ngrpx = 1
      l2nl=2*(nl-1)
      allocate(symope(3,3))
      symope(1:3,1) = (/1d0,0d0,0d0/)
      symope(1:3,2) = (/0d0,1d0,0d0/)
      symope(1:3,3) = (/0d0,0d0,1d0/)
      ificlass=ifile_handle()
      open (ificlass,file='CLASS')
      allocate(iclasst(natom),invgx(ngrp)
     & ,miat(natom,ngrp),tiat(3,natom,ngrp),shtvg(3,ngrp))
      write(6,*)'  --- Readingin CLASS info ---'
      do ibas = 1,natom
        read(ificlass,*) ibasx, iclasst(ibas)
        write(6, "(2i10)") ibasx, iclasst(ibas)
      enddo
      close(ificlass)
!! Get space-group transformation information. See header of mptaouof.
      call mptauof(symope,ngrpx,plat,natom,pos,iclasst
     o ,miat,tiat,invgx,shtvg ) !note: miat,tiat,shtvg are defined in m_zmel.
      if(verbose()>=40) write (*,*)' hsfp0.sc.m.F: end of mptauof'
!!  ppbrd = radial integrals,  cgr = rotated cg coeffecients.
      call rdpp(nxx, nl, ngrpx, nn, nclass, nspin, symope,qbas)
      ntq = nband
      allocate(itq(ntq)) !itq=i for i=1,ntq. a dummy. c.f. hsfp0.sc.F
      do i=1,ntq
        itq(i)=i
      enddo
!! Pointer to optimal product basis
c      allocate(imdim(natom))
c      call indxmdm (nblocha,nclass,iclass,natom,
c     o imdim )                  !use in m_zmel
.not.c      if(smbasis()) nblochpmx = nbloch + ngcmx
      nblochpmx = nbloch + ngcmx
      allocate(ngveccB(3,ngcmx)) ! work arry

!! ... for legas test (not used so often. To compare homogeneos electron gas).
```

```
c      legas = .false.
c      INQUIRE (FILE = 'LEGAS', EXIST = legas)
!!
      iqxend = nqibz + nq0i
      write(6,*) ' nqibz nqibze=',nqibz,nqibze

!! Initialization of readEigen !readin m_hamindex
ccccccccccccccccccccccccccccccc
      ginv=transpose(plat)
ccccccccccccccccccccccccccccccc
      call init_readeigen(ginv,nspin,nband,mrece)!EVU EVD are read in init_readeigen
      call init_readeigen2(mrecb,nlmto,mrecg)
      if(verbose()>50) print *,'eeee exit of init_readeigen2'
!! We get frhis,freq_r,freq_i, nwhis,nw,npm,wiw  by getfreq
      call findemaxmin(nband,qbze,nqbze,nspin, emax,emin)
      if (nctot > 0) Emin=minval(ecore(:,1:nspin))
      omg2max = (Emax-Emin)*.5d0+.2d0
             ! (in Hartree) covers all relevant omega, +.2 for margin
      if(MPI__root) write(6,"(' emin emax omega2max=',3f13.5)") emin, emax, omg2max
      realomega = .true.
      imagomega = .true.
      tetra     = .true.
      if(imagonly) then !WVI only for imagonly for ixc==12
        realomega =.false.
        imagomega =.true.
      endif
      if(realonly) then !epsPP noLFC mode for ixc==13
        realomega =.true.
        imagomega =.false.
      endif
!! getfreq returun date given at " use m_freq,only:".
      lqall=.false.
.not.      if(ixc==11) then
        lqall=.true.
      endif
.not.      if(epsmode) call getwemax(lqall,wemax) !wemax is to determine nw !real axis divisions
      call getfreq(epsmode,realomega,imagomega,tetra,omg2max,wemax,niw,ua,MPI__root)
      if(MPI__root) write(6,"(' nw=',i5)") nw
      nwp = nw+1
.not.      if(imagomega) niw=1
!! ... get eigenvector corresponds to exp(iqr) (q is almost zero).
      if(epsmode) then !iepsmode/=0) then ;  write(6,*) ' read in Mix0vec'
        allocate(epsi(nw_i:nw,neps)) !5July2005 nwp should be used after it is defined!
      endif

!! Miyake tote mode Nov2004. Need fixing.
      if(ecorr_on>0) then  !it was bzcase()==2 Was it bug?
        allocate(nstibz(nqibz))
        do iq=1,nqibz
          iqbz = iqindx(qibz(:,iq),ginv,qbz,nqbz)
          nstibz(iq) = nstbz(iqbz)
c          write(6,"(' iq qibz nstibz=',i5,3f9.4,i5)")iq,qibz(:,iq),nstibz(iq)
        enddo
      endif

!! tetra init
c      call getkeyvalue("gwinput","tmpwwk",tmpwwk,default=.false.)
c      if(tetra) then
c        allocate( !wgt(nband+nctot,nband,nqbz), !noccxvv(nw+niw),
c     &    nbnbtt(nqbz,npm), ekxx1(nband,nqbz), ekxx2(nband,nqbz)) !!! nband=nlmto
c        if(tmpwwk)  ifwwk = iopen('tmp.wwk',0,-1,0)
c      endif ;  if(debug) write(6,*)' xxx1:'
      noccxv     = maxocc2 (nspin,ef, nband, qbze,nqbze) ! maximum no. occupied valence states
      if(noccxv>nband) call rx( 'hx0fp0: all the bands filled! too large Ef')
      noccx      = noccxv + nctot

C allocate( ppb(nlnmx*nlnmx*mdimx*nclass,nspin) )
c$$$C ... This is just to get nblochpmx
c$$$      if(smbasis()) then
c$$$        call getngbpomat(nqibz+nq0i,  nnmx,nomx)
c$$$        nblochpmx = nnmx
c$$$      endif

      nprecx = ndble  !We use double precision arrays only.
      mrecl  = nprecx*2*nblochpmx*nblochpmx/nword()
      if (MPI__root) then
        ifwd   = iopen('WV.d',1,-1,0)
        write (ifwd,"(1x,10i14)") nprecx,mrecl,nblochpmx,nwp,niw,nqibz + nq0i-1,nw_i
        ifwd = iclose('WV.d'); ifwd=0
      endif
      allocate(  zw(nblochpmx,nblochpmx) )
      nspinmx = nspin

!!... these are used x0k
      call getkeyvalue("gwinput","nbcutlow",nbcut, default=0 )
      call getkeyvalue("gwinput","nbcutlowto",nbcut2, default=0 )
      write(6,"(' nbcut nbcutlowto=',2i5)") nbcut,nbcut2
```

```
!! -- ppb= <Phi(SLn,r) Phi(SL'n',r) B(S,i,R(r))>
!! This is general for rotated CG coefficient; but hx0fp0 mode is only for  ngrpx=1 (not rotated).
!! Compare usage in hsfp0 modes.
      irot=1
      allocate( ppbir(nlnmx*nlnmx*mdimx*nclass,irot,nspin))
      do is = 1,nspin
        call ppbafp_v2 (irot,ngrpx,is,nspin,
     i   il,in,im,nlnm,
     i   nl,nn,nclass,nlnmx,
     i   mdimx,lx,nx,nxx,       !Bloch wave
     i   cgr, nl-1,             !rotated CG
     i   ppbrd,                 !radial integrals
     o   ppbir(:,irot,is))      ! this is in m_zmel, used to generate <phi|phi B>
      enddo
      if(debug)write(6,*) ' end of ppbafp_v2'

!! Set iqxini
      if(omitqbz) then
        iqxini= nqibz + 1
      else
        iqxini= 1
      endif

!! check write 1st part for Ec mode to ecorr.chk Nov2004
      if(ecorr_on>0) then
!!!!!!!!!!!!! this path is under developing. !!!!!!!!!!!
        call rx(' ! hx0fp0: need to fix this path. check subroutine getwk and so on in this path')
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
.and..not.        if(ecorr_on >0 (imagomega) )
.and..not.     &   call rx( ' hx0fp0: ecorr_on  (imagomega)')
        write(ieclog, "('   iq                q                       wk')")
        do iqq = iqxini,iqxend
          call getwk(iqq, wibz, wqt,nqbz,nqibz,nstibz,nq0i, wk4ec)
          write(ieclog,"(i5,3x,3f12.8, f15.5)") iqq, q, wk4ec
        enddo
        write(ieclog,*)
        write(ieclog,"('   iw omega(Ry)       wiw')")
        do iww=1,niw
          write(ieclog,"(i5, f10.5, f10.5)") iww,2d0*freq_i(iww),wiw(iww)
        enddo
        write(ieclog,*)
        write(ieclog,"(' Note:IntWgt=wk*wiw.',
     &'  Ec =\sum_{k,iw) IntWgt(k,iw)*ecqw(k,iw)')")
        close(ieclog)
        open(ieclog,file="ecorr.chk",access='append')

        call getkeyvalue("gwinput","necut_p",necut, default=1 )
        allocate(totexc(necut),trpv(necut),trlog(necut))
        totexc = 0d0
        trpv   = 0d0
        trlog  = 0d0
      else
        necut=1
      endif
!!
      allocate(ecut(necut),ecuts(necut))
      call getkeyvalue("gwinput","ecut_p" ,ecut, necut,default=(/1d10/) )
      call getkeyvalue("gwinput","ecuts_p",ecuts,necut,default=(/1d10/) )
!!
      if( chipm ) then
        nmbas=natom
        allocate(imbas(nmbas),imbas_s(nmbas))
        istat=-9999 ! istat=-9999 means noumber of readin arguments is returened in istat.
        call getkeyvalue("gwinput","magatom",
     &        imbas,nmbas,status=istat)
        nmbas = istat
        write(6,*)
        write(6,"('Readin MagAtom nmbas =',i3,' imbas= ',10i3)") nmbas,imbas(1:nmbas)
        imbas_s(1:nmbas) = imbas(1:nmbas)
        imbas(1:nmbas)   = abs(imbas(1:nmbas))
        allocate(jcoup(nw_i:nw,neps) )
        allocate( svec(nbloch,nmbas) )  !sep2006
        svec=0d0
        allocate( cvec(nbloch,nmbas),momsite(nmbas),
     &    mmnorm(nmbas))              !May2007
        cvec=0d0
        do imb=1,nmbas
          ibas= imbas(imb)
          ifv = iopen ('MixSpin.'//charnum3(ibas),1,3,0)
          read(ifv,*) ibasx,lxx
          allocate(nxx_r(0:lxx))
          do i=0,lxx
            read(ifv,*) nxx_r(i)  !   write(6,"(2i5,d13.6)") nxx_r(i)
          enddo
          allocate(spinvec((lxx+1)**2,maxval(nxx_r)))
          allocate(consvec((lxx+1)**2,maxval(nxx_r)))
          spinvec=0d0
          do ilmx = 1, (lxx+1)**2
            lb = ll(ilmx )       !  write(6,*)' lb=',lb,lxx,ilmx
            do ixx = 1, nxx_r(lb)  !  write(6,*)' nn=',nn,nxx_r(lb)
              if(ilmx==1) then
                read(ifv,*) ilm_r, nx_r, spinvec(ilmx,ixx),chg1,chg2
     &       ,consvec(ilmx,ixx)
              else
                read(ifv,*) ilm_r, nx_r, spinvec(ilmx,ixx),dumm1,dumm2
     &       ,consvec(ilmx,ixx)
              endif
!           write(6,"(2i5,d13.6)") ilmx, ixx, spinvec(ilmx,ixx)
            enddo
          enddo
!! Calculate ChiPM. So sign of omega should be correct.
          if(imb==1) then !determine spin direction with respect to ibas=imbas(imb=1)
            spinmom=(chg1-chg2)
            schi=1d0
            if(spinmom<0d0) then
              schi  = -1d0   ! This affects to dpsion. Obtained results
                             ! should be the same in both mode.
            endif
          endif
!!  ReOrdering of spinvec in natom ordering...
          i=0
          if(ibas>1) i= sum(nblocha(1:ibas-1))
          do lb  = 0, lx (ibas)
            do nb  = 1, nx (lb,ibas)
              do mb  = -lb, lb
                i = i+1
                ilmx = lb**2+ lb+ mb +1
                svec(i,imb) = spinvec(ilmx,nb)
                cvec(i,imb) = consvec(ilmx,nb)
                write(6,"(' i lb mb svec svec**2=',3i4,2d13.5)")
     &      i,lb,mb,svec(i,imb),svec(i,imb)**2
              enddo
            enddo
          enddo
          deallocate(nxx_r,spinvec,consvec)
          close(ifv)
          mmnorm (imb) = sqrt(sum(svec(:,imb)**2))
          momsite(imb) = chg1-chg2
c        write(6,"(' svecsum=',e23.15)") sum(svec(:,imb)**2
c         write(ifchipmlog,"(2e23.15,' ! mmom mmnorm')")momsite(imb),mmnorm(imb)
          write(6,"( 'mmom mmnorm= ',2f14.10)")  momsite(imb),mmnorm(imb)
        enddo
      endif

! I assume 1 is for majority for eiqr case.
c      if(ix==10222) then
cc        schi=1d0 !1d0 means Majority is isp=1. If Majority is isp=2, use schi=-1d0.
cc        allocate(jcoup(nw_i:nw,neps))
c        mmnorm=1d0
c      endif
c
c! nmbas_in is for rcxqmean
.and..and.c      if(chipm  nolfco) then ! ix/=10222) then
c        nmbas_in = nmbas
c      else
c        nmbas_in = 1
c      endif
.and.c      if(epsmodenolfco) then
c        allocate( rcxqmean(nwhis,npm,nmbas_in,nmbas_in))
c        if(debug) write(6,"('fff:',3i5)") nwhis,npm,nmbas_in
c      else
c        allocate( rcxqmean(1,1,1,1)) !dummy
c      endif
c
c      if(chipm) allocate(eiqrm(nmbas))
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cctakao
c$$$      allocate( x0meanx(nmbas,nmbas) )
c$$$      allocate( x0mat(nmbas,nmbas),x0matinv(nmbas,nmbas) )
c$$$      do 1101 iq = iqxini,iqxend ! q=(0,0,0) is omitted!
c$$$         if(iq==iqxini+2) exit
c$$$         q = qibze(:,iq)
c$$$         write(6,*)'aaaaaaaaaa q=',q
c$$$          read(ifgb0vec,*) qgbin(1:3),ngbin,igc0,dnorm
c$$$          if(sum(abs(q))==0d0)then
c$$$            if(sum(qgbin**2) >1d-7)stop'qgbin=0 xxx See hvccfp0'
c$$$          elseif(sum(abs(qgbin(1:3)-q)) >1d-8)then
c$$$            stop'qgbin inconsistent'
c$$$          endif
c$$$          write(6,"(' --- Readin Mix0vec: ',3d13.6,2i5,d18.8)")
c$$$     &      qgbin(1:3),ngbin,igc0,dnorm
c$$$c          if(ngb/=ngbin) stop 'hx0fp0: ngb/=ngbgin'
```

```
c$$$          ngb=ngbin
c$$$          write(6,"(' ngb nwp niw=',3i8)")ngb,nwp,niw
c$$$          nmbas_in=1
c$$$          allocate( gbvec(ngb),zzr(ngb,1),x0mean(nw_i:nw,1,1))
c$$$          x0mean=0d0
c$$$          do i=1,ngb
c$$$            read(ifgb0vec,"(4d24.15)") zz1,zz2,zz3,zz4
c$$$            gbvec(i)= dcmplx(zz1,zz2)
c$$$            zzr(i,1)= dcmplx(zz3,zz4)
c$$$          enddo
c$$$          write(6,"(' normchk=',255e23.15)") sum( dconjg(gbvec)*zzr(:,1) )
c$$$     &        ,sum(abs(gbvec(:))), sum(abs(zzr(:,1)))
c$$$            allocate(eiqrm(nmbas))
c$$$            do imb=1,nmbas
c$$$            eiqrm(imb)= sum( dconjg(gbvec(1:nbloch))*svec(1:nbloch,imb) )
c$$$            write(6,"(' <eiqr|m> =',255e23.15)") eiqrm(imb)
c$$$            if( imbas_s(imb)<-1) eiqrm(imb)= -eiqrm(imb)
c$$$            enddo
c$$$            write(6,"('<eiqr|m>:Set \pm in GWinput(for stuggard chi)')")
c$$$              iqixc2 = iq- (nqibz+nq0ix)
c$$$              ifx = iopen ('StonerNLFC.dat',1,3,0)
c$$$              read(ifx,*) jzero2
c$$$              ifx= iclose('StonerNLFC.dat')
c$$$
c$$$              ifchipm2=iopen(
c$$$     &                 'ChiPM'//charnum4(iqixc2)//'.nolfc.mat',1,3,0)
c$$$              do iw=1,10
c$$$              read(ifchipm2,
c$$$     &        '(36x,2x,20x,2x,255e23.15)') x0meanx(:,:)
c$$$              write(6,'("xxx x0mat=",255d13.5)') x0meanx
c$$$              x0matinv=x0meanx
c$$$              call matcinv(nmbas,x0matinv)
c$$$              do i=1,nmbas
c$$$                x0matinv(i,i)= x0matinv(i,i) - jzero2 ! (chipm_0^+-)^-1 - I
c$$$              enddo
c$$$              x0mat = x0matinv
c$$$              do i=1,nmbas
c$$$              x0mat(i,i) = x0mat(i,i)+ img*1d-30 ! to avoid inversion error.
c$$$              enddo
c$$$
c$$$              call matcinv(nmbas,x0mat) !this is full x0_+-
c$$$              trr = sum( eiqrm*matmul(x0mat,dconjg(eiqrm)) ) !*mmnorm
c$$$              write(6,
c$$$     &        '("ttt",3f12.8,2x,f10.5,2x,2e23.15,2x,2e23.15)') q, 2*schi*frr, trr,1d0/trr
c$$$              enddo
c$$$            ifx=iclose(ifchipm2)
c$$$ 1101  continue
c$$$          stop 'xxxxxxxxxxxxxxxxxxxxxxx'
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!! nov2016 moved from tetwt5 --> here
      call getkeyvalue("gwinput","nband_chi0",nbmx, default=nband )
      call getkeyvalue("gwinput","emax_chi0", ebmx, default=1d10  )
      mtet=(/1,1,1/)
      call getkeyvalue("gwinput","multitet",mtet,3,default=(/1,1,1/))
      ! multitet=T ==> micro tetrahedron method (divided-tetrahedron). Not used so much now...
      allocate(ekxx1(nband,nqbz),ekxx2(nband,nqbz))


!! -- EIBZ mode for nolfco -----------------------
      eibzmode=eibz4x0()
!! If eibzmode=T, it is efficient but can slightly break crystal symmetry.

!! This is because band connectivity is judged by just from band ordering in tetrahedron weitht tetwt5.
c      if(nolfco) then
c          eibzmode = .false.
c      endif
c!! ---------------------------------------------

!! === Use of symmetry. EIBZ procedure PRB81,125102 ===
!!  For rotation of zcousq.  See readeigen.F rotwv.F ppbafp.fal.F(for index of product basis).
      if(eibzmode) then
!! commentout block inversion Use iqxendx=iqxend because of full inversion
        call cputid(0)
        write(6,*)' ---goto eibzmode block ---'
        iqxendx=iqxend
        if(epsmode) iqxendx=iqxend
        allocate( nwgt(nqbz,iqxini:iqxendx), !qeibz(3,nqbz,iqxini:nqibz),neibz(iqxini:nqibz),
     &   igx(ngrp*2,nqbz,iqxini:iqxendx),igxt(ngrp*2,nqbz,iqxini:iqxendx),
     &   eibzsym(ngrp,-1:1,iqxini:iqxendx))
!! Check timereversal is required for symmetrization operation or not. If tiii=timereversal=F is enforced,
!! the symmetrization procedure in x0kf_v4h becomes a little time-consuming.
        write(6,*)
        write(6,"('=== Goto eibzgen === TimeRevesal switch =',l1)")timereversal()
        if(MPI__root) iprintx=.true.
        call eibzgen(nqibz,symgg,ngrp,qibze(:,iqxini:iqxend),iqxini,iqxendx,qbz,nqbz,
     i   timereversal(),ginv,iprintx,
     o   nwgt,igx,igxt,eibzsym,tiii)
        write(6,"('Used timeRevesal for EIBZ = ',l1)") tiii
        call cputid(0)
! PBindex: index for product basis.  We will unify this system; still similar is used in ppbafp_v2.
        call PBindex(natom,lx,l2nl,nx) !all input. Returns requied index stored in arrays in m_pbindex.
        call cputid(0)
        call readqgcou()        !no input. Read QGcou and store date into variables.
c     call Spacegrouprot(symgg,ngrp,plat,natom,pos) ! all inputs.
      else                      !dummy allocation to overlaid -check bound !sep2014
        iqxendx=iqxend
        allocate( nwgt(1,iqxini:iqxendx),igx(1,1,iqxini:iqxendx)
     &   ,igxt(1,1,iqxini:iqxendx), eibzsym(1,1,iqxini:iqxendx)) !dummy
        nwgt=1
      endif

      allocate( llw(nw_i:nw,nq0i), llwI(niw,nq0i) )
!! == Calculate x0(q,iw) and W == main loop 1001 for iq.
!! NOTE: iq=1 (q=0,0,0) write 'EPS0inv', which is used for iq>nqibz for ixc=11 mode
!! Thus it is necessary to do iq=1 in advance to performom iq >nqibz.
!! (or need to modify do 1001 loop).
!! iq>nqibz for ixc=11 is not time-consuming.
      call MPI__hx0fp0_rankdivider2(iqxini,iqxend)

!! ------------------------------------------------------------
!! === loop over iq ==========================================
!! ------------------------------------------------------------
      do 1001 iq = iqxini,iqxend  ! NOTE: q=(0,0,0) is omitted when iqxini=2
.not.        if(  MPI__task(iq) ) cycle
.or.        if(ixc==101normalm) then
          ifrcwi = iopen('WVI.'//charnum5(iq),0,-1,mrecl)
        endif
        if (normalm) then
          ifrcw  = iopen('WVR.'//charnum5(iq),0,-1,mrecl)
        endif
        call cputid (0)

        q = qibze(:,iq)
        call readqg('QGcou', q, ginv,  quu,ngc,ngveccB)

!! Caution : confusing point
!!  ngc by QGcou is shown at the bottom of lqg4gw.
!!  ngc read from PPOVL are given by rdata4gw---> ngc(iq>nqibz )=ngc for q=0
!!
.and.c        if( newaniso2iq==1 ) then ! *sanity check
        if( iq==1 ) then ! *sanity check
          if(sum(q**2)>1d-10) then
            call rx( ' hx0fp0: sanity check. |q(iqx)| /= 0')
          endif
        endif

!! ==== readin Coulomb matrix ====
        ngb = nbloch + ngc !ngb is readin from vcoul 25jan2006
        write(6,*)
        write(6,"('===== do 1001: iq q=',i7,3f9.4,' ========')")iq,q !qq
        write(6,"('  nbloch ngb ngc=',3i10)") nbloch,ngb,ngc

!! === readin diagonalized Coulomb interaction ===
!! zcousq: E(\nu,I), given in PRB81,125102; vcousq: sqrt(v), as well.
.and..not.c        if(newaniso2(chipm)) then
.not.        if((chipm)) then
          vcoudfile='Vcoud.'//charnum5(iq)  !this is closed at the end of do 1001.  iq was iqqv
          ifvcoud = iopen(trim(vcoudfile),0,-1,0)
          read(ifvcoud) ngb0
          if( ngb0/=ngb ) call rx( 'hx0fp0.m.f:ngb0/=ngb')
          read(ifvcoud) qvv
          if(sum(abs(qvv-q))>1d-10) then
            write(6,*)'qvv =',qvv
            call rx( 'hx0fp0: qvv/=0 hvcc is not consistent')
          endif
          if(allocated(zcousq)) deallocate( zcousq,vcousq )
          allocate( zcousq(ngb0,ngb0),vcousq(ngb0))
          read(ifvcoud) vcousq
          read(ifvcoud) zcousq
          idummy=iclose(trim(vcoudfile))
          vcousq=sqrt(vcousq)
          if(allocated(zzr)) deallocate(zzr)
          allocate(zzr(1,1)) !dummy
          zzr=0d0
        endif

.and..and.        if(chipm  nolfco) then ! ix/=10222) then
          nmbas_in = nmbas
        elseif(nolfco) then
          nmbas_in = 1
        else
          nmbas_in = ngb
        endif
        nmbas1 = nmbas_in
        nmbas2 = nmbas1

!! ==== set up for epsilon mode =====
        if(epsmode) then
          iqixc2 = iq- (nqibz+nq0ix)
.not..and.          if((chipm)nolfco) then
            allocate( x0mean(nw_i:nw,1,1) )
            x0mean=0d0
          endif
.and.!! zzr is only for chipmnolfco mode
.and.          if(chipm nolfco) then
            allocate(zzr(ngb,nmbas),x0mean(nw_i:nw,nmbas,nmbas))
            x0mean=0d0
            zzr   =0d0
            zzr(1:nbloch,1:nmbas) = svec(1:nbloch,1:nmbas)
          endif
!! ... Open ChiPM*.nolfc_mat
.and.          if( wqt(iq-nqibz)==0d0chipm ) then
            ifchipmn_mat=iopen('ChiPM'//charnum4(iqixc2)//'.nlfc.mat',1,3,0)
            write(ifchipmn_mat,"(255i5)") nmbas
            write(ifchipmn_mat,"(255i5)") imbas(1:nmbas)
            write(ifchipmn_mat,"(255e23.15)") momsite(1:nmbas)
            write(ifchipmn_mat,"(255e23.15)")  mmnorm(1:nmbas)
c            write(ifchipmn_mat,"(255e23.15)") eiqrm(1:nmbas)!if necessary, fix code to give eiqrm.
 takaoAug2012
            write(ifchipmn_mat,"( ' Here was eiqrm: If needed, need to fix hx0fp0')")
.not.            if(nolfco) then
              ifchipm_fmat=iopen('ChiPM'//charnum4(iqixc2)//'.fmat',0,3,0)
              write(ifchipm_fmat) nbloch, natom,nmbas, iqxini,iqxend, nw_i,nw
              write(ifchipm_fmat) imbas(1:nmbas),momsite(1:nmbas),mmnorm(1:nmbas)
              write(ifchipm_fmat) nblocha(1:natom),svec(1:nbloch,1:nmbas)
              write(ifchipm_fmat) zzr0(1:nbloch) !zzr(1:nbloch,1)
            endif
.and..not.          elseif(wqt(iq-nqibz)==0d0(chipm)) then
!! ... Open EPS* file
            filepsnolfc ='EPS'//charnum4(iqixc2)//'.nlfc.dat'
            ifepsdatnolfc = iopen ( filepsnolfc,1,3,0)
            write(ifepsdatnolfc,"(a)")' q(1:3)   w(Ry)   eps    epsi  --- NO LFC'
.not.            if(nolfco) then
              fileps = 'EPS'//charnum4(iqixc2)//'.dat'
              ifepsdat = iopen ( fileps,1,3,0)
              write(ifepsdat,"(a)") ' q(1:3)   w(Ry)   eps  epsi --- LFC included. '
            endif
          endif
        endif

.and.        if(epsmodenolfco) then !iepsmode==202) then
c          rcxqmean=0d0
        else
          write(6,*) "rcxq alloc ngb nwhis npm ---",ngb,nwhis,npm
          allocate( rcxq(ngb,ngb,nwhis,npm) )
        endif

!! === zmelt conversion on different basis. ppovlz is used in get_zmelt2 in m_zmel (called in x0kf_v4h).
.and.        if(chipmnolfco) then
          if(allocated(ppovlz)) deallocate(ppovlz)
          allocate(ppovlz(ngb,nmbas1))
          ppovlz= zzr
.and.        elseif(nolfco  nmbas1==1) then !for <e^iqr|x0|e^iqr>
c          if(allocated(ppovlzinv)) deallocate(ppovlzinv)
          if(allocated(ppovlz)) deallocate(ppovlz)
          if(allocated(ppovl)) deallocate(ppovl)
          allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb)) !,   ppovlzinv(ngb,ngb))
          call readppovl0(q,ngc,ppovl)
          ppovlz(1:nbloch,:) = zcousq(1:nbloch,:)
          ppovlz(nbloch+1:nbloch+ngc,:) = matmul(ppovl,zcousq(nbloch+1:nbloch+ngc,:))
          write(6,*)'nnnnn',nbloch+ngc,ngb
        else                    !may2013  this removes O^-1 factor from zmelt
c          if(allocated(ppovlzinv)) deallocate(ppovlzinv)
          if(allocated(ppovlz)) deallocate(ppovlz)
          if(allocated(ppovl)) deallocate(ppovl)
          allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb)) !,   ppovlzinv(ngb,ngb))
          call readppovl0(q,ngc,ppovl)
          allocate(ppovl_(ngb,ngb))
          ppovl_=0d0
          do i=1,nbloch
            ppovl_(i,i)=1d0
          enddo
          ppovl_(nbloch+1:nbloch+ngc,nbloch+1:nbloch+ngc)=ppovl
.not.          if(eibz4x0()) then !sep2014 added for eibz4x0=F
            ppovl_= matmul(ppovl_,zcousq)
          endif
          ppovlz = ppovl_
          deallocate(ppovl_,ppovl)
        endif

```

```
!! takao apr2012
        if(nolfco) then
          if(allocated(rcxq)) deallocate(rcxq)
          if(allocated(zxq) ) deallocate(zxq)
          if(allocated(zxqi) ) deallocate(zxqi)
          allocate( rcxq(nmbas1,nmbas2,nwhis,npm) )
          allocate( zxq (nmbas1,nmbas2,nw_i:nw), zxqi (nmbas1,nmbas2,niw))
        else
          allocate( zw0(ngb,ngb), zxq (ngb,ngb,nw_i:nw), zxqi(ngb,ngb,niw) )
        endif
        zxq=0d0;  zxqi=0d0;  rcxq = 0d0
!! ---------------------------------------------------------------
!! === loop over spin=== ======================================
!! ---------------------------------------------------------------
        do 1003 is = 1,nspinmx
          write(6,"(' ##### ',2i4,' out of nqibz+n0qi nsp=',2i4,' ##### ')")iq, is, nqibz + nq0i,nspin
          if(debug) write(6,*)' niw nw=',niw,nw
!! ==== spin chi_charge or chi_+- ====
          isf=is
          if(chipm) then
            write(6,*)" chi_+- mode ixc=",ixc
            if(is==1) isf=2
            if(is==2) isf=1
            rcxq=0d0
          endif

c!! Tetrahedron weitht for zero section
c          call gettetwt(q,iq,is,isf,nwgt(:,iq),frhis0,nwhis0,npm)

!! Tetrahedron weight.
!! output
!!     nbnbx
!!     ihw(ibjb,kx): omega index, to specify the section of the histogram.
!!     nhw(ibjb,kx): the number of histogram sections
!!     jhw(ibjb,kx): pointer to whw
!!     whw( jhw(ibjb,kx) ) \to whw( jhw(ibjb,kx) + nhw(ibjb),kx)-1 ), where ibjb=ibjb(ib,jb,kx)
!!     : histogram weights for given ib,jb,kx for histogram sections
!!     from ihw(ibjb,kx) to ihw(ibjb,kx)+nhw(ibjb,kx)-1.
c            write(6,*) ' --- goto x0kf_v4hz ---- newaniso= ',newaniso2
!! input
!!     ekxx1 for   rk,is
!!     ekxx2 for q+rk,isf
          do kx = 1, nqbz
            call readeval(qbz(:,kx),   is,  ekxx1(1:nband, kx) )
            call readeval(q+qbz(:,kx), isf, ekxx2(1:nband, kx) )
          enddo
          call gettetwt(q,iq,is,isf,nwgt(:,iq),frhis,nwhis,npm,
     i     qbas,ginv, ef, nqibz, nband,ekxx1,ekxx2, nctot,ecore,
     i     nqbz,qbz,nqbzw,qbzw,  ntetf,idtetf,ib1bz,
     i     nbmx,ebmx,mtet,eibzmode) !nov2016

c$$$c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$        jpm=1
c$$$        do k=1,nqbz
c$$$c           nkqmin= 999999
c$$$c           do ibib = 1, nbnb(k,jpm)
c$$$c              nkqmin = min(n2b(ibib,k,jpm),nkqmin)
c$$$c           enddo
c$$$           do ibib = 1, nbnb(k,jpm) !---  ibib loop
c$$$c             print *,' k ibib=',k,ibib
c$$$c              it =  n1b(ibib,k,jpm)    !valence
c$$$c              itp = n2b(ibib,k,jpm) - itps + 1 !val
c$$$c              if(n1b(ibib,k,jpm)==n2b(ibib,k,jpm)) then
c$$$              do iw = ihw(ibib,k,jpm),ihw(ibib,k,jpm)+nhw(ibib,k,jpm)-1 !iiww=iw+ihw(ibib,k)-1
c$$$c                 if(iw<20) then
c$$$                 imagweight = whw(jhw(ibib,k,jpm)+iw-ihw(ibib,k,jpm))
c$$$c                 write(*,'("eeee ",4i5,2x,d13.5,x,d13.5)') k, n1b(ibib,k,jpm),n2b(ibib,k,jpm), iw,
 imagweight
c$$$c                 endif
c$$$              enddo            ! iw
c$$$c              endif
c$$$           enddo
c$$$        enddo
c$$$c        stop 'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv'
c$$$c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccc


!! == x0kf_v4hz is the main routine to accumalte imaginary part of x0 ==
          call cputid(0)
          if(npm==1) then
            ncc=0
          else
            ncc=nctot
          endif
          call x0kf_v4hz(npm,ncc,
     i        ihw,nhw,jhw,whw,nhwtot,  ! tetwt5
     i        n1b,n2b,nbnbx,nbnb,    ! use whw by tetwt5 ,
     i              q,
     i              nspin,is,isf, !symmetrize, !
     i              qbas,ginv,  qbz,wbz,
     d            nlmto,nqbz,nctot, !noccx,noccxv,
     d            nbloch,  nwhis, !nlnmx,mdimx,
     i    iq,ngb,ngc,ngpmx,ngcmx,  !ngb/=ngc+nbloch for smbasis()=T oct2005
     i    nqbze,nband,nqibz,
     o    rcxq,   !  rcxq is the accumulating variable for spins
     i    nolfco,zzr,nmbas_in, zcousq,   !ppovl,nmbas1,nmbas2, is removed ppovlz,
     i    chipm,eibzmode, !z1offd,!for nolfco Add nmbas Sep2006
     i    nwgt(:,iq),igx(:,:,iq),igxt(:,:,iq),ngrp, eibzsym(:,:,iq),crpa)
!! ------Question, Apr2015takao. -------------------
!! ???  we may need
!! ???  "if(is==nspinmx.or.chipm) then" for chipm mode.
!! ???  really OK ??? Need check more... Compare with old code...
!! ---------------------------------------------------
!kino 2014-08-19 add

!! == Symmetrizer for crystal symmetry (and also for spin)
!!    Symmetrize and convert to Enu basis by dconjg(tranpsoce(zcousq)*rcxq8zcousq if eibzmode
.or.          if (is==nspinmxchipm) then ! Apr2015. Takao think ".or.chipm" is required for chipm mode
                                         ! Because rcxq is calculated for each is, symmetrized and its contribution
                                         ! is added to zxq in dpsion5.
            call x0kf_v4hz_symmetrize(npm, !ncc,
c     i       ihw,nhw,jhw,whw,nhwtot, ! tetwt5
c     i       n1b,n2b,nbnbx,nbnb, ! use whw by tetwt5 ,
     i       q,
     i       nspin,is,isf, !symmetrize, !
     i       qbas,ginv,  !qbz,wbz,
c     i       nblocha,           !nlnm,nlnmv,nlnmc,iclass,
c     i              ppb(1,is),
c     i              icore,ncore,
c     d       nlmto,nqbz,nctot,  !noccx,noccxv,
c     d       natom,             !nl,nclass,natom,nnc,
     d       nbloch,  nwhis,    ! nlnmx,mdimx,
     i       iq,ngb,ngc,ngpmx,ngcmx, !ngb/=ngc+nbloch for smbasis()=T oct2005
     i       nqbze,nband,nqibz,
     o       rcxq,              !  rcxq is the accumulating variable for spins
     i       nolfco,zzr,nmbas_in, zcousq, !ppovl,nmbas1,nmbas2, is removed ppovlz,
     i       chipm,eibzmode,    !z1offd,!for nolfco Add nmbas Sep2006
     i       ngrp, eibzsym(:,:,iq))
          endif
          call tetdeallocate() !deallocate(ihw,nhw,jhw, whw,ibjb,n1b,n2b)
          iecut=1
          if(debug) write(6,"(a)") ' --- goto dpsion5 --- '
.or.          if(is==nspinmxchipm) then
            write(6,"('  nmbas1,nmbas2=',2i10)") nmbas1,nmbas2
            call dpsion5(frhis,nwhis, freq_r, nw, freq_i,niw, realomega, imagomega,
     i       rcxq, npm,nw_i, nmbas1,nmbas2, ! rcxq is alterd---used as work
     o       zxq, zxqi,
     i       chipm, schi,is,  ecut(iecut),ecuts(iecut))
.and.           if(nolfcoepsmode) then
              do iw=nw_i,nw
                x0mean(iw,:,:)=zxq(:,:,iw)
              enddo
            endif
            write(6,*)' --- end of dpsion5 ----',sum(abs(zxq)),sum(abs(zxqi))
          endif
 1003   continue  !end of spin loop =====
        if(allocated(rcxq) ) deallocate(rcxq)

!! ===  RealOmega ===================================
        if (realomega) then
.or.          if(chipm) then !ixc==22ixc==23) then
            if (nspin==1) call rx( 'chipm modes are for nspin==2')
.and..and..not..or.          elseif(epsmodenolfco(chipm)) then !ixc==2iepsmode==202) then
            if (nspin==1) x0mean= 2d0*x0mean  !if paramagnetic, multiply x0 by 2
            if (nspin==1) zxq = 2d0*zxq       !if paramagnetic, multiply x0 by 2
          else
            if (nspin == 1) zxq = 2d0*zxq   !if paramagnetic, multiply x0 by 2
          endif

c         write (ifxd,"(1x,3f10.4)") q(1),q(2),q(3)
c         write (ifrx) rxq,cxq
          if(epsmode) then
            if(nolfco) then
              ttt='without LFC'
            else
              ttt='with LFC'
            endif
            if(chipm) then
              write(6,*) '--- chi0_{+-}}^{-1}      --- '//ttt
            else
              write(6,*) '--- dielectric constant --- '//ttt
              write(6, *)" trace check for w-v"
            endif
          endif

!! prepare for iq0.
          iq0 = iq - nqibz
          if(allocated(epstilde)) deallocate(epstilde,epstinv)
          allocate(epstilde(ngb,ngb),epstinv(ngb,ngb))

!! === iw loop for real axiw ===
          do 1015 iw  = nw_i,nw  !Feb2006. Before it was 1:nwp (nwp=nw+1).
            !  So freq_r(iw-1) is shifted to freq_r(iw).
            frr= dsign(freq_r(abs(iw)),dble(iw))
.not..or.            if(epsmode) then  !if(ixc==1sergeyv) then
              imode = 1
!! === wcf: W= (1-v zxq)^{-1} v ===
.and.c              if(newaniso2iq<=nqibz) then !for mmmw
              if(iq<=nqibz) then !for mmmw
                if(iq==1) then
                  ix=1
                  zw0(:,1)=0d0
                  zw0(1,:)=0d0
                else
                  ix=0
                endif
!!  Eqs.(37),(38) in PRB81 125102
                do igb1=ix+1,ngb
                  do igb2=ix+1,ngb
                    epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
                    if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
                  enddo
                enddo
                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
c                write(*,"(’rrr: ’,i5,2x,f10.4,2x,10(d12.5,x,d12.5,2x))")
 iw,freq_r(iw),(epstinv(igb,igb),igb=1,5)
                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
c                write(*,"(’ggg: ’,i5,2x,f10.4,2x,10(d12.5,x,d12.5))")
 iw,freq_r(iw),(epstinv(igb,igb),igb=1,5)

c$$$ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$cmmm direct inversion vs. block inversion
c$$$                if(iq>nqibz) then
c$$$c direct inversion
c$$$                ix=0
c$$$                do igb1=ix+1,ngb
c$$$                do igb2=ix+1,ngb
c$$$                  epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
c$$$                  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
c$$$                enddo
c$$$                enddo
c$$$                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
c$$$                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
c$$$                do igb1=1+ix,ngb
c$$$                do igb2=1+ix,ngb
c$$$                  zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
c$$$                  if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
c$$$                enddo
c$$$                enddo
c$$$c              write(*,"(’mmmmzp99x  ’,i3,10(2d13.5,2x))") iw,zw0(1,1),zw0(2:10:3,1),zw0(63:70:3,1)
c$$$              write(*,"(’mmmmzp99x  ’,i3,10(2d13.5,2x))")
 iw,1d0/epstinv(1,1),zw0(2:10:3,1),zw0(63:70:3,1)
c$$$c             write(*,"(’mmmmzp99x  ’,i3,10(2d13.5,2x))") iw,zw0(1,1),zw0(1,2:10:3),zw0(1,63:70:3)
c$$$c block inversion
c$$$                ix=1
c$$$                do igb1=ix+1,ngb
c$$$                do igb2=ix+1,ngb
c$$$                  epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
c$$$                  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
c$$$                enddo
c$$$                enddo
c$$$                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
c$$$                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
c$$$                absq=sqrt(sum(q**2*tpioa**2))
c$$$                sk(  1:ngb)= zxq(1,1:ngb,iw)
c$$$                sks( 1:ngb)= zxq(1:ngb,1,iw)
c$$$                w_k(1) =0d0
c$$$                w_ks(1)=0d0
c$$$                w_k( 2:ngb)=
 vcousq(2:ngb)*vcousq(1)*matmul(vcousq(1)*sk(2:ngb)*vcousq(2:ngb),epstinv(2:ngb,2:ngb))
c$$$                w_ks(2:ngb)=
 vcousq(2:ngb)*vcousq(1)*matmul(epstinv(2:ngb,2:ngb),vcousq(1)*sks(2:ngb)*vcousq(2:ngb))
c$$$                llw(iw,iq0)=
c$$$     &             1d0
c$$$     &            -vcousq(1)*sk(1)*vcousq(1) ! sk(1,1,iw)=sks(1,1,iw)=H of Eq.(40).
c$$$     &            -vcousq(1)*vcousq(1)* sum( vcousq(2:ngb)*sk(2:ngb) *
 matmul(epstinv(2:ngb,2:ngb),sks(2:ngb)*vcousq(2:ngb)))
c$$$                write(*,"(’mmmmzwp99x ’,i3,10(2d13.5,2x))") iw,llw(iw,iq0),
 !(1d0/llw(iw,iq0)-1d0)*vcousq(1)**2,
c$$$c     &                  w_k(2:10:3)/llw(iw,iq0), w_k(63:70:3)/llw(iw,iq0)
```

```
c$$$     &                  w_ks(2:10:3)/llw(iw,iq0), w_ks(63:70:3)/llw(iw,iq0)
c$$$                write(*,"('mmmmzwp99x ')")
c$$$                endif
c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
                do igb1=1+ix,ngb
                  do igb2=1+ix,ngb
                    zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
                    if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
                  enddo
                enddo
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cmmmm
c       if(iq>nqibz) then
c         write(*,"('mmmmz99x ',i3,10(2d13.5,2x))") iw,zw0(1,1)+vcousq(1)**2,zw0(2:10:3,1),zw0(63:70:3,1)
c       endif
.or.c       if(iq==1iq>nqibz) then
c                write(*,"('mmmz0  ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(1,2:10:3,iw),zxq(1,63:70:3,iw)
c                write(*,"('mmmz0* ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(2:10:3,1,iw),zxq(63:70:3,1,iw)
c         write(*,"('mmmmz99x ',i3,10(2d13.5,2x))") iw,zw0(1,1)+vcousq(1)**2,zw0(1,2:10:3),zw0(1,63:70:3)
c       write(*,"('mmmzx  ',2i3,10(2d13.5,2x))") iq,iw,zxq(2,1,iw),zxq(2,2:10:3,iw),zxq(2,63:70:3,iw)
c       write(*,"('mmmzx  ',2i3,10(2d13.5,2x))") iq,iw,zxq(3,1,iw),zxq(3,2:10:3,iw),zxq(3,63:70:3,iw)
c       write(*,"('mmmzxs ',2i3,10(2d13.5,2x))") iq,iw,zxq(1,1,iw),zxq(2:10:3,1,iw),zxq(63:70:3,1,iw)
c       write(*,"('mmmzxs ',2i3,10(2d13.5,2x))") iq,iw,zxq(1,2,iw),zxq(2:10:3,2,iw),zxq(63:70:3,2,iw)
c       write(*,"('mmmmzee',2i3,10(2d13.5,2x))")iq,iw,epstilde(2,2),epstilde(2,2:10:3),epstilde(2,63:70:3)
c       write(*,"('mmmmzee',2i3,10(2d13.5,2x))")iq,iw,epstilde(3,2),epstilde(3,2:10:3),epstilde(3,63:70:3)
c       endif
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
              endif

!! for iq>nqibz
.and.c              if(newaniso2iq>nqibz) then
              if(iq>nqibz) then
!! Full inversion to calculalte eps with LFC.
                ix=0
                vcou1 = fourpi/sum(q**2*tpioa**2) ! --> vcousq(1)**2!  !fourpi/sum(q**2*tpioa**2-eee)
                do igb1=ix+1,ngb
                  do igb2=ix+1,ngb
.and.                   if(igb1==1igb2==1) then
                      epstilde(igb1,igb2)= 1d0 - vcou1*zxq(1,1,iw)
                      cycle
                    endif
                    epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
                    if(igb1==igb2) then
                      epstilde(igb1,igb2)=1d0 + epstilde(igb1,igb2)
                    endif
                  enddo
                enddo
                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
                llw(iw,iq0)= 1d0/epstinv(1,1)
                write(6,"('iq iw_real eps(withLFC) eps(woLFC) ',2i5,d13.6,x,d13.6,2x,d13.6,x,d13.6)")
     &           iq,iw,llw(iw,iq0),1d0-vcou1*zxq(1,1,iw)
c$$$c                read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$c                write(6,*)'sumcheck epstinv=',sum(abs(epstinv(2:ngb,2:ngb)))
c$$$c                if(iw/=iwx) stop 'hx0fp0: iw/=iwx'  !sanity check
c$$$c                sk  (1:ngb)= zxq(1,1:ngb,iw)
c$$$c                sks (1:ngb)= zxq(2,1:ngb,iw) ! WARNING: zxq(2,1:ngb,iw) contains zxq(1:ngb,1,iw).
c$$$                                             ! A little confusing. See nolfco=T case in x0kf_v4h.F.
c$$$                                             !
c$$$cc                 sk(  1:ngb)= zxq(1,1:ngb,iw)
c$$$cc                 sks( 1:ngb)= zxq(1:ngb,1,iw)
c$$$                vcou1= fourpi/sum(q**2*tpioa**2) !test --> vcousq(1)**2! !fourpi/sum(q**2*tpioa**2-eee)
c$$$                vcou1sq= sqrt(vcou1)   ! only vcousq(1) should be replaced.
c$$$c                write(ifiss)iw,iq0,ngb,q
c$$$c                write(ifiss)vcou1,vcou1sq,vcousq(2:ngb),sk(1:ngb),sks(1:ngb)
c$$$c                w_k(1) =0d0
c$$$c                w_ks(1)=0d0
c$$$c                w_k( 2:ngb)= vcou1sq*matmul( sk(2:ngb)*vcousq(2:ngb), epstinv(2:ngb,2:ngb) )
c$$$c                w_ks(2:ngb)= vcou1sq*matmul( epstinv(2:ngb,2:ngb), sks(2:ngb)*vcousq(2:ngb))
c$$$cmmm epsPP mode - vcou1sq*sum( sk(2:ngb) * w_ks(2:ngb)*vcousq(2:ngb) )
c$$$c                llw(iw,iq0)=  1d0 -vcou1*sk(1) !- vcou1sq*sum( sk(2:ngb) * w_ks(2:ngb)*vcousq(2:ngb) )
c$$$
c$$$                llw(iw,iq0)=  1d0 - vcou1*zxq(1,1,iw) !- vcou1sq*sum( sk(2:ngb) * w_ks(2:ngb)*vcousq(2:ngb) )
c$$$                write(6,*) 'epsPP iq iw',iq,iw, 1d0 - fourpi* zxq(1,1,iw)/sum(q**2*tpioa**2)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                write(*,"('mmmw0  ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(1,2:10:3,iw),zxq(1,63:70:3,iw)
c                write(*,"('mmmw0* ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(2,2:10:3,iw),zxq(2,63:70:3,iw)
c                write(*,"('mmmmw99x ',i3,10(2d13.5,2x))") iw,fourpi/sum(q**2*tpioa**2)/llw(iw,iq0),
c     &                  w_k(2:10:3)/llw(iw,iq0),w_k(63:70:3)/llw(iw,iq0)
c                write(*,"('mmmmw99x ',i3,10(2d13.5,2x))") iw,llw(iw,iq0),
c     &                  w_ks(2:10:3)/llw(iw,iq0),w_ks(63:70:3)/llw(iw,iq0)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                write(6,*) 'uuuu iq iw llw vc x0mean',iq,iw,fourpi/sum(q**2*tpioa**2),sk(1)
c                write(ifisk) iw,iq0,q
c                write(ifisk) vcousq(2:ngb)*w_k( 2:ngb),vcousq(2:ngb)*w_ks( 2:ngb)
```

```
              endif

.not.c$$$              if(newaniso2) then               ! Original mode
c$$$                call rx( 'not checked here')
c$$$c               call wcf( ngb, vcoul, zxq(1,1,iw), imode, zw0)
c$$$              endif

c$$$!!... a debug mode
c$$$                write(6,"('hhh --- EigenValues for Im( W) --------')")
c$$$                allocate(ebb(ngb))
c$$$                call diagcvh2( (zw0-transpose(dconjg(zw0)))/2d0/img, ngb, ebb)
c$$$                do ii=1,ngb
.and.c$$$                  if( abs(ebb(ii))>1d-8  ebb(ii)>0) then
c$$$                    write(6, "('hhhIWq : iw ii eb=',2i4,d13.5)") iw, ii, ebb(ii)
c$$$                  else
c$$$                    write(6, "('hhhIWqxxx : iw ii eb=',2i4,d13.5)") iw, ii, ebb(ii)
c$$$                  endif
c$$$                enddo
c$$$                deallocate(ebb)

.and.c              if(newaniso2iq>nqibz) then
              if(iq>nqibz) then
c                zw(1:ngb,1:ngb) = 0d0
c                write(ifrcw, rec=((iq-iqxini)*(nw-nw_i+1)+ iw-nw_i+1 ) ) zw   !  WP = vsc-v
              else
                zw(1:ngb,1:ngb) = zw0
c                write(ifrcw, rec=((iq-iqxini)*(nw-nw_i+1)+ iw-nw_i+1 ) ) zw   !  WP = vsc-v
                write(ifrcw, rec= iw-nw_i+1) zw   !  WP = vsc-v
                call tr_chkwrite("freq_r iq iw realomg trwv=", zw, iw, frr,nblochpmx, nbloch,ngb,iq)
              endif
!! epsmode
.and..not..and.           elseif(epsmode(chipm)) then !ixc/=23) then ! No LFC (local field correction).
 It's better to use echo 4| hbasfp0.
               if(debug)write(6,*) 'xxx2 epsmode iq,iw=',iq,iw
c              write(6,*)'ppppp sumcheck zxq=',sum(abs(zxq)),sum(abs(zzr)),sum(abs(vcoul)),sum(abs(gbvec))
c              if(newaniso2) then
!! there is difference of two vcmean below since we use (sligthy) screened Coulomb (screenfac() in
 switch.F)
!!   NOTE that we use vcoul with screening (screenfac() is used in hvccfp0.F
c               vcmean = fourpi/sum(q**2*tpioa**2) !aug2012
                vcmean=vcousq(1)**2
                epsi(iw,iqixc2)= 1d0/(1d0 - vcmean*zxq(1,1,iw))
                write(6,'(" iq iw omega eps epsi nolfc=",2i6,f8.3,2e23.15,3x, 2e23.15,
     &          " vcmean x0mean =", 2e23.15,3x, 2e23.15)') iqixc2,iw,2*frr,
     &          1d0/epsi(iw,iqixc2),epsi(iw,iqixc2),vcmean, zxq(1,1,iw)!x0mean(iw,1,1)
                write(ifepsdatnolfc,'(3f12.8,2x,d12.4,2e23.15,2x,2e23.15)')
     &          q, 2*frr, 1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
.not.                if(nolfco) then
                  ix=0
                  do igb1=ix+1,ngb
                  do igb2=ix+1,ngb
.and.                    if(igb1==1igb2==1) then
                      epstilde(igb1,igb2)= -vcmean*zxq(igb1,igb2,iw) !aug2012
                    else
                      epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
                    endif
                    if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
                  enddo
                  enddo
                  epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
                  call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
                  epsi(iw,iqixc2)= epstinv(1,1)
                  write(6,'( " iq iw omega eps epsi  wlfc="
     &           ,2i6,f8.3,2e23.15,3x, 2e23.15)')
     &            iqixc2,iw,2*frr,1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
                  write(6,*)
                  write(ifepsdat,'(3f12.8,2x,d12.4,2e23.15,2x,2e23.15)')
     &            q, 2*frr,1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
                endif
c$$$              else
c$$$                 write(6,*)'not support epsmode and newaniso=F mode now.'
c$$$c$$$                vcmean= sum( dconjg(gbvec) * matmul(vcoul,gbvec) )
.not.c$$$c$$$                if(nolfco) then
c$$$c$$$                 x0mean(iw,1,1) = sum( dconjg(zzr(:,1))* matmul(zxq(:,:,iw),zzr(:,1)))
c$$$c$$$                endif
c$$$c$$$                epsi(iw,iqixc2) = 1d0/(1- vcmean * x0mean(iw,1,1))
c$$$c$$$                write(6,'(" iq iw omega eps epsi nolfc=",2i6,f8.3,2e23.15,3x, 2e23.15,
c$$$c$$$     &          " vcmean x0mean =", 2e23.15,3x, 2e23.15)') iqixc2,iw,2*frr,
c$$$c$$$     &          1d0/epsi(iw,iqixc2),epsi(iw,iqixc2),vcmean,x0mean(iw,1,1)
c$$$c$$$                write(ifepsdatnolfc,'(3f12.8,2x,d12.4,2e23.15,2x,2e23.15)')
c$$$c$$$     &          q, 2*frr, 1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
.not.c$$$c$$$                if(nolfco) then ! With LFC
c$$$c$$$                  imode=2
c$$$c$$$                  call wcf( ngb, vcoul, zxq(1,1,iw), imode,
c$$$c$$$     &                zw0) !  write(6,"('ssschk1=',3d13.5)") sum(abs(zw0)) sum(abs(gbvec))
c$$$c$$$                  epsi(iw,iqixc2)= sum( dconjg(gbvec) * matmul(zw0,zzr(:,1)) )
c$$$c$$$                  write(6,'( " iq iw omega eps epsi  wlfc="
c$$$c$$$     &            ,2i6,f8.3,2e23.15,3x, 2e23.15)')
c$$$c$$$     &            iqixc2,iw,2*frr,1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
c$$$c$$$                  write(6,*)
c$$$c$$$                  write(ifepsdat,'(3f12.8,2x,d12.4,2e23.15,2x,2e23.15)')
c$$$c$$$     &            q, 2*frr,1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
c$$$c$$$                endif
c$$$              endif
C --- ChiPM mode
.and.            elseif(epsmodechipm) then
              allocate( x0meanx(nmbas,nmbas) )
              if(nolfco) then  ! ChiPM mode without LFC
c$$$                if(legas) then
c$$$                   call rx( ' hx0fp0.m.F need to implement thigs here if required')
c$$$! --- three lines below may work for test purpose for legas. But not sure.
c$$$c                  vcmean= sum( dconjg(gbvec) * matmul(vcoul,gbvec) )
c$$$c                  write(ifchipmn,'(3f12.8,2x,f8.5,2x,2e23.15)')
c$$$c     &            q, 2*schi*frr, 1d0-vcmean*2*x0mean(iw,1,1)
 !4*pi*alat**2/sum(q**2)/4d0/pi**2*x0mean(iw)
c$$$                else
c$$$                  x0meanx = x0mean(iw,:,:)/2d0  !in Ry unit.
c$$$                endif
                x0meanx = x0mean(iw,:,:)/2d0 !in Ry unit.
              else
C ... ChiPM mode with LFC... NoLFC part
                zxq(1:ngb,1:ngb,iw) = zxq(1:ngb,1:ngb,iw)/2d0  ! in Ry.
                do imb1=1,nmbas
                  do imb2=1,nmbas
                    x0meanx(imb1,imb2)=
     &              sum( svec(1:nbloch,imb1)*
     &              matmul(zxq(1:nbloch,1:nbloch,iw),svec(1:nbloch,imb2)))  !/ mmnorm**2  I removed mmnorm
 may2007
                  enddo
                enddo
!       x0meanx= <m|chi^+-(\omega)|m>/<m|m>**2
              endif
              do imb1=1,nmbas
                do imb2=1,nmbas
                  x0meanx(imb1,imb2) =
     &            x0meanx(imb1,imb2)/mmnorm(imb1)/mmnorm(imb2)
                enddo
              enddo
              write(ifchipmn_mat,'(3f12.8,2x,f20.15,2x,255e23.15)')q, 2*schi*frr, x0meanx(:,:)
.not.              if(nolfco) write(ifchipm_fmat) q, 2*schi*frr, zxq(1:nbloch,1:nbloch,iw)

c! These lines commented by "c! ' are histories ---> For Takao's memo. Maybe not so useful for others.
c! ! for NoLFC, Get I from q=0, and calculate Tr(Chipm)
c!               allocate( x0mat(nmbas,nmbas),x0matinv(nmbas,nmbas) )
c!               ifx = iopen ('StonerNLFC.dat',1,3,0)
c!               if(iw==0 .and. sum(q**2) <1d-13) then
c!                 x0mat = x0meanx
c!                 x0mat(:,:)= x0mat +transpose(dconjg(x0mat))
c!                 x0matinv= 0.5d0*x0mat
c!                 call matcinv(nmbas,x0matinv)
c!                 write(6,*) ' q=',q
c!                 write(6,*) ' nmbas ifx=',nmbas,ifx
c!                 write(6,*) ' x0matinv=',x0matinv
c!                 allocate(evall(nmbas))
c!                 call diagno00(nmbas,x0matinv,evall)
c! ! Note that x0matinv at omega=0 is negative definite matrix (by definition).
c!                 do i1=1,nmbas
c!                   write(6,'(" eval(iw=0)=",i5,f15.5)') i1, -evall(i1)
c!                 enddo
c!                 jzero2 = minval(-evall)
c!                 deallocate(evall)
c!                 write(ifx,"(e23.15)")  jzero2
c!                 do imb=1,nmbas !temporary
c!                   write(ifx,"(e23.15,' ! tttt temporary... U_mm in eV')")
c!      &            rydberg()*jzero2*mmnorm(imb)**2/momsite(imb)**2
c!                 enddo
c!               elseif(iw==0) then
c!                 read(ifx,*,end=1013,err=1013) jzero2
c!                 goto 1014
c!  1013           continue
c!                 stop " i/o error StonerNLFC.dat"
c!  1014           continue
c!               endif
c!               ifx= iclose('StonerNLFC.dat')
c!               if(onceww(6)) write(6,*)' i/o end: StonerNLFC.dat'
c! !
c!               x0matinv = x0meanx
c!               call matcinv(nmbas,x0matinv)
c!               do i=1,nmbas
c!                 x0matinv(i,i)= x0matinv(i,i) + jzero2 ! (chipm_0^+-)^-1 + I
c!               enddo
c!               x0mat= x0matinv
c!               do i=1,nmbas
c!                 x0mat(i,i) = x0mat(i,i)+ img*1d-30 ! to avoid inversion error.
```

```
c!               enddo
c!               call matcinv(nmbas,x0mat) !this is full x0_+-
c!               trr = sum( eiqrm*matmul(x0mat,dconjg(eiqrm)) )
c!               write(ifchipmn,
c!      &        '(3f12.8,2x,f20.15,2x,2e23.15,2x,2e23.15)') q, 2*schi*frr, trr,1d0/trr
c!               deallocate( x0mat,x0matinv)
c!
c! C--- With LFC ! save or read Istoner
c!               if(.not.nolfco) then
c!                 zzz = zxq(1:nbloch,1:nbloch,iw)
c!                 ifstoner = iopen ('Stoner.dat',1,3,0)
c!                 if( sum(q**2) < 1d-10 .and. iw==0 ) then
c!                   call diagno00(nbloch,zzz,ss0)
c!                   ! zzz is negative definite at omegw=0 if the ground state is stable.
c!                   ! minval(ss0) is for the largest negative value (softest mode).
c!                   Istoner = -1d0/minval(ss0)
c!                   do ii= 1,nbloch
c!                     if(verbose()>50.or.iw<=2) then
c!                       write(6,"(' eig chi^0_+- =',
c!      &                     i4,d13.5,256d13.5 )" ) ii, ss0(ii)
c!                     endif
c!                   enddo
c!
c! cxxxx thisa SVD procedure is not used now.
c! c!      SVD of chi^-1: !now only look for lowest eigenvalue problem... So rather eigenvalue problem instead of SVD
c! c                    write(6,"(a,i5)")' ----SVD: chiinv --- iw=',iw
c! c                    zxq(1:nbloch,1:nbloch,iw)=zzz
c! c                    call zgesvdnn(
c! c     i              nbloch, zxq(1:nbloch,1:nbloch,iw),
c! c     o              SS0,UU0,VT0)
c! c                    Istoner = -sum(UU0(:,1)*VT0(1,:))/ss0(1)
c! c!                    do ii= 1,nbloch
c! c!                      write(ifstoner,'(4e23.15)') UU0(ii,1),VT0(1,ii)
c! c!                    enddo
c!                   write(ifstoner,"(e23.15)")
c!      &             Istoner
c!                   do imb=1,nmbas
c!                     write(ifstoner,"(e23.15,'!tttt temporary U_mm in eV')")
c!      &            Istoner*rydberg()*mmnorm(imb)**2/momsite(imb)**2
c!                   enddo
c!                 elseif(iw==0) then
c!                   read(ifstoner,*) Istoner
c!                 endif
c!                 ifstoner = iclose('Stoner.dat')
c! C...  <eqir| 1/(1 + I chi^0_+-) | eiqr>
c!                 mmat = + Istoner * zzz
c!                 do i = 1, nbloch
c!                   mmat(i,i) = mmat(i,i) + 1d0
c!                 enddo
c! c                trr0 = sum( dconjg(zzr(1:nbloch,1))*
c! c     &                     matmul( mmat,zzr(1:nbloch,1) )  )
c! c                write(6,"(' <eiqr| 1 + I chi0^+-|eiqr> =',255e23.15)") trr0
c!                 do i=1,nbloch
c!                   mmat(i,i) = mmat(i,i)+ img*1d-30 ! to avoid inversion error.
c!                 enddo
c!
c! c$$$c prtest for NiO with 4 bloch basis
c! c$$$              zzzx = mmat   !matmul(sproj,matmul(mmat,sproj))
c! c$$$                  call zgesvdnn(
c! c$$$     i            nbloch, zzzx,
c! c$$$     o            eex,UU0,VT0)
c! c$$$! projected denominator
c! c$$$              denom = matmul( sproj,matmul(Istoner * zzz,sproj))
c! c$$$              do i = 1, nbloch
c! c$$$                denom(i,i) = denom(i,i) + 1d0
c! c$$$              enddo
c! c$$$!
c! c$$$              zzzx=denom
c! c$$$                  call zgesvdnn(
c! c$$$     i            nbloch, zzzx,
c! c$$$     o            eey,UU0,VT0)
c! c$$$              write(ifchipm2,
c! c$$$     &        '(3f12.8,2x,f20.15,2x,4f11.5,3x,4f11.5)') q, 2*schi*frr, eex,eey
c! c$$$!
c! c$$$              zzzy=denom
c! c$$$              do i=1,nbloch
c! c$$$                zzzy(i,i) = zzzy(i,i)+ img*1d-30 ! to avoid inversion error.
c! c$$$              enddo
c! c$$$              call matcinv(nbloch, zzzy)
c! c$$$Ctest --- another inversion procedure  ! zzzy is the inverse of denom
c! c$$$c             VT = dconjg(transpose(UU0))
c! c$$$c             UU = dconjg(transpose(VT0))
c! c$$$c             zzzy=0d0
c! c$$$c             do i=1,nbloch
c! c$$$c               do ix=1,nbloch
c! c$$$c               do iy=1,nbloch
```

```
c! c$$$c                zzzy(ix,iy) = zzzy(ix,iy) + UU(ix,i)*VT(i,iy)/eey(i)
c! c$$$c                enddo
c! c$$$c                enddo
c! c$$$c              enddo
c! c$$$c              zzzx = matmul(denom,zzzy)
c! c$$$c              do i=1,nbloch
c! c$$$c              do j=1,nbloch
c! c$$$c                 write(6,"('zzzx=',2i5,2d13.6)")i,j,zzzx(i,j)
c! c$$$c              enddo
c! c$$$c              enddo
c! c$$$              zzzx = matmul( sproj,matmul(zzz,sproj) )
c! c$$$              mmatx = matmul(zzzx, zzzy)
c! c$$$              trrx  = sum( dconjg(zzr(1:nbloch,1)) *
c! c$$$     &                    matmul(mmatx,zzr(1:nbloch,1)) )
c! c$$$cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c!                 call matcinv(nbloch, mmat)
c!                 mmat = matmul(zzz,mmat)
c! c--- prtest I found that This makes the difference at high energy part!!! Nov-9-2006
c! c---            mmat = matmul( sproj,matmul(mmat,sproj) )
c!                 trr  = sum( dconjg(zzr(1:nbloch,1)) *
c!      &                     matmul(mmat,zzr(1:nbloch,1)) )
c!                 write(ifchipm,
c!      &         '(3f12.8,2x,f20.15,2x,2e23.15,2x,4e23.15)')
c!      &              q,   2*schi*frr,  trr, 1d0/trr
c!               endif
              deallocate(x0meanx)
            endif
c            write(6,*)'tttt aaa iw=',iw
 1015     continue
c          write(6,*)'tttt  end of  do 1015 loop'
c          if(newaniso2) then
c            if(allocated(sk)) deallocate(sk)!,sks,w_k,w_ks)
c          endif


c
c          if(chipm.and.(.not.nolfco))
c     &          deallocate(sqovlp,sqovlpi,uu0,vt0,ss0,mmat,zzz)
c$$$          if( ixc==5.or.ixc==6 ) then
c$$$            jpm=1
c$$$c            nwmax = nw
c$$$c            if(ixc==5) nwmax =nw
c$$$            allocate(trwv(nw_i:nw),trwv2(nw_i:nw))
c$$$            do iw = nw_i,nw !max ! trace check
c$$$              trwv(iw) = zxq(6,7,iw)
c$$$              trwv2(iw) = 0d0
c$$$              do i = 1,ngb
c$$$                trwv2(iw) = trwv2(iw) + zxq(i,i,iw)
c$$$              enddo
c$$$            enddo
c$$$            do iw= nw_i,nw-1
c$$$              if(ixc==5)
c$$$     &         write(6,"('iq iw[min_max]=',2i5,2f7.4,' trwv by wwk*h= ',
c$$$     &         12d13.5)") iq, iw, freq_r(iw), freq_r(iw+1),
c$$$     &         (trwv2(iw)+trwv2(iw+1))/2d0*(freq_r(iw)-freq_r(iw+1)),
c$$$     &         (trwv(iw)+trwv(iw+1))  /2d0*(freq_r(iw)-freq_r(iw+1))
c$$$              !weight for the histgram range. by tetwt5
c$$$              if(ixc==6)
c$$$     &         write(6,"('iq iw[min_max]=',2i5,2f7.4,' trwv by whw  = ',
c$$$     &         12d13.5)") iq,iw, freq_r(iw), freq_r(iw+1),
c$$$     &            trwv2(iw),trwv(iw) !weight for the histgram range. by tetwt5
c$$$            enddo
c$$$            deallocate(trwv,trwv2)
c$$$          endif
          if( allocated(zzr)   ) deallocate(zzr)
          if( allocated(x0mean)) deallocate(x0mean)
          if( allocated(gbvec) ) deallocate(gbvec)
        endif


c ... Close files for epsmode
        if(epsmode) then !iepsmode/=0) then      ! only calculate iq>nqibz
          if(chipm) then
            ifchipmn_mat=iclose('ChiPM'//charnum4(iqixc2)//'.nlfc.mat')
            if(.not.nolfco) then
              ifchipm_fmat=iclose( 'ChiPM'//charnum4(iqixc2)//'.fmat')
            endif
          else
            filepsnolfc ='EPS'//charnum4(iqixc2)//'.nolfc.dat'
            ifepsdatnolfc = iclose( filepsnolfc)
            if(.not.nolfco) then
              fileps = 'EPS'//charnum4(iqixc2)//'.dat'
              ifepsdat  = iclose(fileps)
            endif
          endif
        endif
c --- realomega end ==============================



c --- imagomega ==================================
        if (imagomega) then
          write(6,*)' goto imag omega'
          if (nspin == 1) zxqi = 2d0*zxqi   ! if paramagnetic, multiply x0 by 2
          if (ecorr_on>0)then !ixc==101.or.(sergeyv.and.imagonly)) then
            imode=0
          else
            imode=1
          endif

!! === iw loop for imag axiw ===
          do 1016 iw  = 1,niw
c            if(newaniso2 .and. iq<=nqibz ) then
            if( iq<=nqibz ) then
!!  Eqs.(37),(38) in PRB81 125102
              if(iq==1) then
                ix=1
                zw0(:,1)=0d0
                zw0(1,:)=0d0
              else
                ix=0
              endif
!!  Eqs.(37),(38) in PRB81 125102
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  epstilde(igb1,igb2)= -vcousq(igb1)*zxqi(igb1,igb2,iw)*vcousq(igb2)
                  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
                enddo
              enddo
              epstinv=epstilde
              call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
                  if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
                enddo
              enddo
c              if(iq==1) write(ifepstinv) epstinv(ix+1:ngb,ix+1:ngb),iq,iw
            endif
c            if(newaniso2.and.iq>nqibz) then
            if(iq>nqibz) then
!! Full inversion to calculalte eps with LFC.
              ix=0
              vcou1 = fourpi/sum(q**2*tpioa**2) ! --> vcousq(1)**2!  !fourpi/sum(q**2*tpioa**2-eee)
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  if(igb1==1.and.igb2==1) then
                    epstilde(igb1,igb2)= 1d0 - vcou1*zxqi(1,1,iw)
                    cycle
                  endif
                  epstilde(igb1,igb2)= -vcousq(igb1)*zxqi(igb1,igb2,iw)*vcousq(igb2)
                  if(igb1==igb2) then
                    epstilde(igb1,igb2)=1d0 + epstilde(igb1,igb2)
                  endif
                enddo
              enddo
              epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
              call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
              llwi(iw,iq0)= 1d0/epstinv(1,1)
              write(6,*) 'iq iw_img  eps(withLFC) eps(woLFC)',iq,iw,llwi(iw,iq0),1d0-vcou1*zxqi(1,1,iw)

c$$$c              read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$c              if(iw/=iwx) then
c$$$c                write(6,*)'iw iwx=',iw,iwx
c$$$c                stop 'hx0fp0: iw/=iwx' !sanity check
c$$$c              endif
c$$$cmmm3
c$$$cc                ski(1:ngb)= zxqi(1,1:ngb,iw)
c$$$cc                sksi(1:ngb)= zxqi(1:ngb,1,iw)
c$$$c              ski(1:ngb)= zxqi(1,1:ngb,iw)
c$$$c              sksi(1:ngb)= zxqi(2,1:ngb,iw)
c$$$              vcou1  = fourpi/sum(q**2*tpioa**2) ! test-->vcousq(1)**2 !fourpi/sum(q**2*tpioa**2-eee)
c$$$              vcou1sq= sqrt(vcou1)
c$$$c              write(ifiss) iw,iq0,ngb,q
c$$$c              write(ifiss) vcou1,vcou1sq,vcousq(2:ngb),ski(1:ngb),sksi(1:ngb)
c$$$c              w_ki(1) = 0d0
c$$$c              w_ksi(1)= 0d0
c$$$c              w_ki( 2:ngb)= vcou1sq*matmul( ski(2:ngb)*vcousq(2:ngb), epstinv(2:ngb,2:ngb) )
c$$$c              w_ksi(2:ngb)= vcou1sq*matmul( epstinv(2:ngb,2:ngb), sksi(2:ngb)*vcousq(2:ngb))
c$$$cmmm epspp mode ---> no - vcou1sq*sum( ski(2:ngb) * w_ksi(2:ngb)*vcousq(2:ngb) )
c$$$c              llwi(iw,iq0)=  1d0 -vcou1*ski(1) !- vcou1sq*sum( skI(2:ngb) * w_ksI(2:ngb)*vcousq(2:ngb) )
c$$$
c$$$              llwi(iw,iq0)=  1d0 - vcou1*zxqi(1,1,iw)  !- vcou1sq*sum( skI(2:ngb) *
 w_ksI(2:ngb)*vcousq(2:ngb) )
c$$$              write(6,*) 'iq iw llwI',iq,iw,llwi(iw,iq0)
c$$$c              write(ifisk) iw,iq0,q
c$$$c              write(ifisk) vcousq(2:ngb)*w_ki(2:ngb),vcousq(2:ngb)*w_ksi( 2:ngb)
            endif

c$$$            if(.not.newaniso2) then               ! original mode
c$$$               call rx( 'not checked here')
c$$$c              call wcf( ngb, vcoul,zxqi(1,1,iw),imode,  zw0)
c$$$            endif

c            if(newaniso2.and.iq>nqibz) then
            if(iq>nqibz) then
c              zw(1:ngb,1:ngb) = 0d0 ! zw(nblochpmx,nblochpmx)
c              write(ifrcwi, rec=(iq-iqxini)*niw + iw)  zw    !  WP = vsc-v
            else
              zw(1:ngb,1:ngb) = zw0 ! zw(nblochpmx,nblochpmx)
c              write(ifrcwi, rec=(iq-iqxini)*niw + iw)  zw    !  WP = vsc-v
              write(ifrcwi, rec=iw)  zw    !  WP = vsc-v
              call tr_chkwrite("freq_i iq iw imgomg trwv=",zw,iw,freq_i(iw),nblochpmx,nbloch,ngb
,iq)
            endif


!! --- Miyake's total energy branch !Nov2004. not maintained now... need to fix this maybe(2012takao)
 -----------
            if(.false.) then
c            if(ecorr_on>0 .and. (.not.newaniso2)) then !I did not modified this for newaniso2 2012takao
              if (debug) write(6,*)'ip,ix=',iq,iw,'  niw=',niw
              call getwk(iq, wibz, wqt,nqbz,nqibz,nstibz,nq0i, wk4ec)
              call ecorq2(vcoul, zw0, ngb, iq,iw,ieceig,
     o        erpaqw, trpvqw, trlogqw)
c --- integration along imaginary axis.
! omit k and basis index for simplicity
! wint = -(i/4pi) < [w'=-inf,inf] Q(w') >
!
! When w' ==> iw', w' is now real,
!   wint =  (1/2pi) < [w'=0,inf] Q(iw') >
!
! transform: x = 1/(1+w')
! this leads to a denser mesh in w' around 0 for equal mesh x
! which is desirable since Q is peaked around w'=0
!    wint =  (1/2pi) < [x=0,1] Q(iw') / x^2 >
              faca   =  wk4ec* wiw(iw)
              trpv(iecut)    = trpv(iecut)  + faca* trpvqw
              trlog(iecut)   = trlog(iecut) + faca* trlogqw
              totexc(iecut)  = totexc(iecut)+ faca* erpaqw !  = trpv+ trlog
c            ecqw(iq,iw) = erpaqw
              if(iw==1) then
                write(ieclog,*)
              endif
              if(iw==1.and.iq==iqxini) then
                write(ieclog,
     &         "('   iq   iw   omega/i(Ry)        IntWgt',
     &         '    trpvqw(eV)      ecqw(eV)   ecqw*IntWgt',
     &         ' :  ecut   ecuts')")
              endif
              write(ieclog,"( 2i5,3f14.6,3f14.6,2f8.3)")
     &       iq,iw, 2d0*freq_i(iw), faca, trpvqw*hartree, erpaqw*hartree,
     &       faca*erpaqw*hartree, ecut(iecut),ecuts(iecut)
              close(ieclog)
              open(ieclog,file="ecorr.chk",access='append')
ccccccccccccccccccccccccccccccccccccccccccccccccc
c           allocate( ovlpc(ngb,ngb),evall(ngb),
c     &          evecc(ngb,ngb))
c           evall=0d0
c           ovlpc=0d0
c           do i=1,ngb
c             ovlpc(i,i)=1d0
c           enddo
c           nmx=ngb
cc1           call diagcv(ovlpc,zw0/2d0+transpose(dconjg(zw0))/2d0,evecc,ngb, evall,nmx,1d99, nev)
c           call diagcv(ovlpc,zw0,evecc,ngb, evall,nmx,1d99, nev)
c           write(6,"('ngb nev=',2i5)") ngb,nev
c           write(6,"('chk eigen of zw0 Max Min=',2d13.6)")maxval(evall),minval(evall)
c           do i=1,3
c             write(6,*) i, evall(i)
c           enddo
c           do i=ngb-3,ngb
c            write(6,*) i, evall(i)
c           enddo
c           deallocate( ovlpc,evall,evecc)
ccccccccccccccccccccccccccccccccccccccccccccccc
            endif
 1016     continue
c          if(newaniso2) then
            deallocate(epstinv)
            if(allocated(epstilde)) deallocate(epstilde)
            if(allocated(epstilde)) deallocate(epstilde)
c          endif
c$$$          enddo
        endif
c... imagomega end ===============================
        if(allocated(vcoul)) deallocate(vcoul)
        if(allocated(zw0)) deallocate(zw0)
        if(allocated(zxq )) deallocate(zxq)
        if(allocated(zxqi)) deallocate(zxqi)
        if    (ixc==101.or.normalm) then
          ifrcwi = iclose('WVI.'//charnum5(iq))
        endif
        if (normalm) then
          ifrcw  = iclose('WVR.'//charnum5(iq))
        endif
 1001 continue
!! == end of loop 1001 for q point ==
      call mpi__barrier()


!! === Recieve llw and llwI at node 0, where q=0(iq=1) is calculated. ===
      if(mpi__size/=1) then
        do iq=nqibz+1,iqxend
          iq0 = iq - nqibz
c     write(6,*)' iq iq0 mpi_rank mpi_ranktab(iq)=',iq, iq0,mpi__rank,mpi__ranktab(iq),mpi__root,nw,nw_i,
niw
          if(mpi__ranktab(iq)/=0) then !jan2012
            if(mpi__ranktab(iq) == mpi__rank) then
              dest=0
              call mpi__dblecomplexsend(llw(nw_i,iq0),(nw-nw_i+1),dest)
              call mpi__dblecomplexsend(llwi(1,iq0),niw,dest)
            elseif(mpi__root) then
c     write(6,*)' mpi_recv iq from',iq,mpi__ranktab(iq),nw,nw_i,niw
              src=mpi__ranktab(iq)
              call mpi__dblecomplexrecv(llw(nw_i,iq0),(nw-nw_i+1),src)
              call mpi__dblecomplexrecv(llwi(1,iq0),niw,src)
c     do i=nw_i,nw
c     write(6,*)'recivxxx',i,llw(i,iq0)
c     enddo
c     write(6,*)' recv llw sum=',sum(abs(llw(:,iq0))),nw,nw_i
c     write(6,*)' recv llwI sum=',sum(abs(llwi(:,iq0))),niw
            endif
          endif
        enddo
      endif

c$$$      deallocate( llw, llwi )
c$$$!! == generate llw and llwI ==
c$$$      ifiss=iopen('SkSks',0,-1,0)
c$$$      ifepstinv = iopen('EPS0inv',0,0,0)
c$$$      allocate( llw(nw_i:nw,nq0i), llwi(niw,nq0i) )
c$$$      read(ifepstinv) ngb
c$$$      do 1501 iq0=1,nq0i
c$$$        rewind ifepstinv
c$$$        read(ifepstinv) ngb
c$$$        allocate(vcousq(2:ngb),sk(ngb),sks(ngb),w_k(ngb),w_ks(ngb))
c$$$        do iw=nw_i,nw
c$$$          read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$          read(ifiss) iwx,iq0x,ngb,q
c$$$          if(iw/=iwx) stop 'hx0fp0:1501 iw/=iwx'
c$$$          read(ifiss)vcou1,vcou1sq,vcousq(2:ngb),sk(1:ngb),sks(1:ngb)
c$$$          w_k(1) =0d0
c$$$          w_ks(1)=0d0
c$$$          w_k( 2:ngb)= vcou1sq*matmul( sk(2:ngb)*vcousq(2:ngb), epstinv(2:ngb,2:ngb) )
c$$$          w_ks(2:ngb)= vcou1sq*matmul( epstinv(2:ngb,2:ngb), sks(2:ngb)*vcousq(2:ngb))
c$$$          !! epsPP mode - vcou1sq*sum( sk(2:ngb) * w_ks(2:ngb)*vcousq(2:ngb) )
c$$$          llw(iw,iq0)=  1d0 -vcou1*sk(1) !- vcou1sq*sum( sk(2:ngb) * w_ks(2:ngb)*vcousq(2:ngb) )
c$$$          write(6,*) 'epsPP iq iw',iq,iw, 1d0 - fourpi* sk(1)/sum(q**2*tpioa**2)
c$$$        enddo
c$$$        deallocate(vcousq,sk,sks,w_k,w_ks)
c$$$        allocate(vcousq(2:ngb),ski(ngb),sksi(ngb),w_ki(ngb),w_ksi(ngb))
c$$$        do iw=1,niw
c$$$          read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$          read(ifiss) iwx,iq0x,ngb,q
c$$$          if(iw/=iwx) stop 'hx0fp0:1501 iw/=iwx'
c$$$          read(ifiss) vcou1,vcou1sq,vcousq(2:ngb),ski(1:ngb),sksi(1:ngb)
c$$$          w_ki(1) = 0d0
c$$$          w_ksi(1)= 0d0
c$$$          w_ki( 2:ngb)= vcou1sq*matmul( ski(2:ngb)*vcousq(2:ngb), epstinv(2:ngb,2:ngb) )
c$$$          w_ksi(2:ngb)= vcou1sq*matmul( epstinv(2:ngb,2:ngb), sksi(2:ngb)*vcousq(2:ngb))
c$$$          !! epsPP mode ---> no - vcou1sq*sum( skI(2:ngb) * w_ksI(2:ngb)*vcousq(2:ngb) )
c$$$          llwi(iw,iq0)=  1d0 -vcou1*ski(1) !- vcou1sq*sum( skI(2:ngb) * w_ksI(2:ngb)*vcousq(2:ngb) )
c$$$          write(6,*) 'iq iw llwI',iq,iw,llwi(iw,iq0)
c$$$        enddo
c$$$        deallocate(vcousq,ski,sksi,w_ki,w_ksi)
c$$$ 1501   continue
```

```

!! == W(0) divergent part and W(0) non-analytic constant part.==
!!   Note that this is only for q=0 -->iq=1
c      if(newaniso2.and.ixc==11.and.mpi__rank==0) then
      if((ixc==11.or.ixc==10011.or.ixc==111).and.mpi__rank==0) then
!! get w0 and w0i (diagonal element at Gamma point
!! This return w0, and w0i
        call w0w0i(llw,llwi,nw_i,nw,nq0i,niw,q0i)
!! === w0,w0i are stored to zw for q=0 ===
!! === w_ks*wk are stored to zw for iq >nqibz ===
        do iq = 1,1             !iq=1 only 4pi/k**2 /eps part only ! iq = iqxini,iqxend
          q = qibze(:,iq)
c          if(iq>nqibz) then
c            iq0 = iq - nqibz
c            read(ifisk) ngb,nw_ixxx,nwxxx,niwxxx
c            allocate(vw_k(ngb),vw_ks(ngb))
c          endif
          do ircw=1,2
            if    (ircw==1) then;  nini=nw_i;   nend=nw;
              ifrcwx = iopen('WVR.'//charnum5(iq),0,-1,mrecl)
            elseif(ircw==2) then;  nini=1;     nend=niw;
              ifrcwx = iopen('WVI.'//charnum5(iq),0,-1,mrecl)
            endif
            do iw=nini,nend
c             if(iq<=nqibz) read(ifrcwx, rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 ) ) zw !(1:ngb,1:ngb)
              read(ifrcwx, rec= iw-nini+1) zw !(1:ngb,1:ngb)
c              if( iq==1 ) then
              if(ircw==1) zw(1,1) = w0(iw)
              if(ircw==2) zw(1,1) = w0i(iw)

cccccccccccccccccccccccccccccccccccccccccccc
c              if(ircw==1) zw(1,1) = 0d0
c              if(ircw==2) zw(1,1) = 0d0
              if(ircw==1) then
                 write(6,"('ffffrrr:', f13.6,2x,f13.6,x,f13.6)") hartree*freq_r(iw),w0(iw)
              endif
ccccccccccccccccccccccccccccccccccccccccccccc


c$$$cmmm3 cccccccccccccccccccccccccccccccccccccccccc
c$$$          elseif( iq>nqibz ) then !-->In future, we store sperical average of zw below to zw(at q=0)===
c$$$            write(6,*)'ddd skip readin ifisk dddddddd'
c$$$            read(ifisk) iwxx,iq0xx,qxx
c$$$            if(iwxx /=iw) stop 'iwxx/=iw'
c$$$           if(iq0xx /=iq-nqibz) stop 'iq0xx /=iq'
c$$$            if(sum(abs(qibze(:,iq)-qxx))>1d-8) stop 'sum(abs(qq-qxx))>1d-8'
c$$$            read(ifisk) vw_k(2:ngb),vw_ks(2:ngb)
c$$$            zw=0d0
c$$$            do igb1=1+1,ngb
c$$$            do igb2=1+1,ngb
c$$$              vc1vc2 = vw_ks(igb1)*vw_k(igb2)
c$$$              if(ircw==1) zw(igb1,igb2)=vc1vc2/llw(iw,iq0)
c$$$              if(ircw==2) zw(igb1,igb2)=vc1vc2/llwi(iw,iq0)
c$$$            enddo
c$$$            enddo
c$$$ccccccccccccccccccccccccccccccccccccccccccccccc
c              endif
c          if(iq==1.or.iq>nqibz) write(ifrcwx,rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 ) ) zw !(1:ngb,1:ngb)
c             write(ifrcwx,rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 ) ) zw !(1:ngb,1:ngb)
              write(ifrcwx,rec= iw-nini+1 ) zw !(1:ngb,1:ngb)
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccmmm3
c          if(iq<=nqibz) then
c             zw=0d0
c             write(ifrcwx, rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 )  ) zw !(1:ngb,1:ngb)
c          endif
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cooo  check write
c          if(mod(iw,3)==1) then
c           do igb1=1,ngb,23
c           do igb2=1,ngb,23
c             if(ircw==1) write(*,"('zzzwr:',4i4,2d13.5)")iq,iw,igb1,igb2,zw(igb1,igb2)
c             if(ircw==2) write(*,"('zzzwi:',4i4,2d13.5)")iq,iw,igb1,igb2,zw(igb1,igb2)
c           enddo
c           enddo
c          endif
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
            enddo
            if    (ircw==1) then
              ifrcwx = iclose('WVR.'//charnum5(iq))
            elseif(ircw==2) then
              ifrcwx = iclose('WVI.'//charnum5(iq))
            endif
          enddo
        enddo
      endif
```

```fortran


c$$$!! --- legas mode is not working now. Need fixing... voltot ntot are not given.
c$$$      if(epsmode.and.legas) then
c$$$        call rx( ' LEGAS mode is not maintained well. Need some fixing.')
c$$$        voltot=0d0
c$$$        ntot=0d0
c$$$        write(6,*)' Find LEGAS. legas =',legas
c$$$        iflegas = 2101
c$$$        open (iflegas,file='LEGAS')
c$$$        read(iflegas,*)rs
c$$$        close(iflegas)
c$$$        alpha  = (9*pi/4d0)**(1d0/3d0)
c$$$        qfermi = alpha/rs
c$$$        efx  = qfermi**2
c$$$        valn = efx**1.5d0*voltot/3d0/pi**2
c$$$        write (6,*)'  #### egas test mode  legas=T #### given rs =',rs
c$$$        write (6,*)'     Exact Fermi momentum  qf  =', qfermi
c$$$        write (6,*)'     Exact Fermi energy    Ef  =', efx
c$$$        do iq = iqxini,iqxend ! q=(0,0,0) is omitted!
c$$$          if(iq<=nqibz) cycle
c$$$          write(6,*)' iq=',iq
c$$$          iqixc2 = iq- (nqibz+nq0ix)
c$$$          filele ='EPSEG'//charnum4(iqixc2)//'.dat'
c$$$          ife = iopen( filele,1,3,0)
c$$$          write(ife,"(a)")
c$$$     &          ' q(1:3)   w(Ry)   eps    epsi  --- NO LFC'
c$$$          q = qibze(:,iq)
c$$$          qt= sqrt(sum(qibze(1:,iq)**2))*2d0*pi/alat
c$$$          qs= qt/qfermi
c$$$          write(6,"(' qs qfermi=',2d13.5)"    ) qs,qfermi
c$$$          write(6,"(' q-q^2/2 q+q^2=',2d13.5)") qs-qs**2/2d0,qs+qs**2/2d0
c$$$          do iw  = nw_i,nw
c$$$            ww  = freq_r(iw)
c$$$            muu = ww/qfermi**2
c$$$            if(     qs<2d0 .and. muu < qs-qs**2/2d0) then
c$$$              x0mx= -img*qfermi/(4*pi*qs)*2*muu
c$$$            elseif( qs<2d0 .and. muu < qs+qs**2/2d0) then
c$$$              x0mx= -img*qfermi/(4*pi*qs)*( 1d0-(muu/qs-.5d0*qs)**2 )
c$$$            else
c$$$              x0mx=0d0
c$$$            endif
c$$$            vcmmmm= 4*pi/qt**2
c$$$            epsi(iw,iqixc2) = 1d0/(1- vcmmmm * x0mx)
c$$$c            epsi(iw,iqixc2) = 1d0/(1- vcmmm(iq) * x0meanx)
c$$$            write(ife,'(3f12.8,2x,d12.4,2e23.15,2x,2e23.15)')
c$$$     &        q, 2*ww,1d0/epsi(iw,iqixc2),epsi(iw,iqixc2)
c$$$          enddo
c$$$        enddo
c$$$        write(6,*)' ----------legas end--------'
c$$$      endif

!! Write TEECOR ecorr_on mode
      if(imagomega.and.ecorr_on>0) then
        hartree=2d0*rydberg()
        ifcor   = iopen('TEECORR2',1,-1,0) ! output files
        do iecut=1,necut
          write(6,"( ' RPA Ec =' 3f23.15,'   ecut ecuts (Ry)=',2d12.4)")
     &   totexc(iecut)*hartree,trpv(iecut)*hartree, trlog(iecut)*hartree
     &    ,ecut(iecut),ecuts(iecut)
          write(ifcor,*) '============================'
          write(ifcor,*) 'Correlation energy Erpa (eV)'
          write(ifcor,*) '============================'
          write(ifcor,*)' ### '
          write(ifcor,"(5e23.15)")
     &     totexc(iecut)*hartree,trpv(iecut)*hartree,trlog(iecut)*hartree
     &    ,ecut(iecut),ecuts(iecut)
        enddo
c... output ecqw !    write(ifcor,*)'### ecqw(q,w) ###'
        write(ifcor,*)' nqibz =',nqibz
        write(ifcor,*)' nq0i  =',nq0i
        write(ifcor,*)' niw   =',niw
        write(ifcor,*)' --- See details of Ec in ecor.chk ---'
c        nqitot = nqibz + nq0i
c        call wecqw(ifcor,
c     d           nqibz,nqbz,nq0i,nqitot,niw,
c     o           wibz,wqt,wx,freqx,ecqw)
c... Write electron gas correlation energy
c$$$        legas = .false.
c$$$        INQUIRE (file = 'LEGAS', exist = legas)
c$$$        if(legas) then !!! test for electron gas case.
c$$$          call rx( ' LEGAS mode is not maintained well. Need some fixing.')
c$$$          voltot=0d0
c$$$          ntot=0d0
c$$$          write(6,*)' find LEGAS. legas =',legas
c$$$          iflegas = 2101
c$$$          open (iflegas,file='LEGAS')
```

```
c$$$          read(iflegas,*)rs
c$$$          close(iflegas)
c$$$          alpha = (9*pi/4d0)**(1d0/3d0)
c$$$          qfermi = alpha/rs
c$$$          efx  = qfermi**2
c$$$          valn = efx**1.5d0*voltot/3d0/pi**2
c$$$          write (6,*)'  #### egas test mode  legas=T #### given rs =',rs
c$$$          write (6,*)' egas  Exact Fermi momentum  qf  =', qfermi
c$$$          write (6,*)' egas  Exact Fermi energy    Ef  =', efx
c$$$          if(tetra) call rx( 'legas You have to give ef of  tetrahedron')
c$$$          efz=(ntot*3*pi**2/voltot)**(2d0/3d0) ! ef is calculated from ntot.
c$$$          qfermi= dsqrt(efz)
c$$$          alpha = (9*pi/4d0)**(1d0/3d0)
c$$$          rs    = alpha/qfermi
c$$$          write (ifcor,*)' --- electron gas ---'
c$$$          write (ifcor,*)' density parameter rs= ', rs
c$$$          write (ifcor,*)' kf= ',qfermi
c$$$          write (ifcor,*)' ### Barth-Hedin formula'
c$$$          ecelgas = eclda_bh(rs) * hartree * ntot
c$$$          write (ifcor,*)ecelgas
c$$$          write (ifcor,*)' ### Perdew-Zunger formula'
c$$$          ecelgas = eclda_pz(rs) * hartree * ntot
c$$$          write (ifcor,*)ecelgas
c$$$          write (ifcor,*)' ### Gell-Mann and Brueckner formula'
c$$$          ecelgas = (-0.0311d0 * dlog(rs) -0.048d0) * hartree * ntot
c$$$          write (ifcor,*)ecelgas
c$$$        endif
      endif
      call cputid(0)
      call mpi__finalize
      if(ixc==11)   call rx0( ' OK! hx0fp0 mode=11     read <Q0P> normal sergeyv')
      if(ixc==111)  call rx0( ' OK! hx0fp0 mode=111    normal sergeyv')
      if(ixc==10011)call rx0( ' OK! hx0fp0 mode=10011  crpa normal sergeyv')
      if(ixc==12)   call rx0( ' OK! hx0fp0 mode=12  Ecor sergeyv mode')
      if(ixc==101)  call rx0( ' OK! hx0fp0 mode=101 Ecor ')
      if(ixc==202)  call rx0( ' OK! hx0fp0 mode=202 sergeyv epsPP NoLFC')
      if(ixc==203)  call rx0( ' OK! hx0fp0 mode=203 sergeyv eps LFC ')
      if(ixc==222)  call rx0( ' OK! hx0fp0 mode=222 chi+- NoLFC sergeyv')
      if(ixc==223)  call rx0( ' OK! hx0fp0 mode=223 chi+- LFC sergeyv')
      end

c--------------------------------------------------------------------
      real*8 function eclda_bh(rs)
      real(8) :: rs,cp,rp,z
      cp       = 0.0504d0*0.5d0 ! 0.5 changes unit from Ry to Hartree
      rp       = 30.d0
      z        = rs / rp
      eclda_bh = -cp * ( (1.d0+z**3)*dlog(1.d0+1.d0/z)
     .                 + 0.5d0*z - z**2 - 0.33333333d0 )
      end
c--------------------------------------------------------------------
      real*8 function eclda_pz(rs)
      real(8) :: rs
      if (rs.ge.1.d0) then
        eclda_pz = -0.1423d0 / (1.d0 + 1.0529d0*dsqrt(rs) + 0.334d0*rs)
      else
        eclda_pz = -0.0480d0 + 0.0311d0*dlog(rs) - 0.0116d0 * rs
     .         + 0.0020d0*rs*dlog(rs)
      endif
      end
c--------------------------------------------------------------------
      subroutine wecqw(ifcor,
     d                 nqibz,nqbz,nq0i,nqitot,niw,
     o                 wibz,wqt,wx,freqx,ecqw)

      implicit double precision (a-h,o-z)
      dimension   wibz(nqibz),wqt(nq0i),wx(niw),
     .            freqx(niw),ecqw(nqitot,niw)
      real(8):: rydberg
      write(ifcor,*)'### ecqw(q,w) ###'
      write(ifcor,*)'nqibz =',nqibz
      write(ifcor,*)'nq0i  =',nq0i
      write(ifcor,*)'niw   =',niw
      do ip = 2,nqitot
        if (ip <= nqibz) then
          wk = wibz(ip)*0.5d0 ! 0.5 for the normalization of wibz
        else
c          wk = wqt(ip-nqibz)*wibz(1)*0.5d0 ! 0.5 for the normalization of wibz
          wk = wqt(ip-nqibz)* 1d0/dble(nqbz)
        endif
        write(ifcor,*)'### iq,wq = ',ip,wk
        sume=0d0
        do ix = 1,niw
          write(ifcor,*)freqx(ix),ecqw(ip,ix),wx(ix)
          sume=sume+  wx(ix)/(freqx(ix)*freqx(ix)) * ecqw(ip,ix)
        enddo
        write(ifcor,*) '  sum ecqw*wx=', wk*sume*2d0*rydberg()
```

```
! end of ip-loop
      enddo
      return
      end
c-------------------------------------------------------------------
      subroutine getsqovlp(q,ngc,ngb,sqovlp)
!! == Get sqrt of ppovl ==
      implicit none
      real(8)::q(3)
      integer:: ngc,ngb,nbloch,i,nmxx,ix,iy,nev
      complex(8):: sqovlp(ngb,ngb)
      complex(8),allocatable:: ooo(:,:),ppo(:,:),sqovlpi(:,:),ppovl(:,:)
      complex(8),allocatable:: ovlp(:,:),evec(:,:)
      real(8),allocatable:: eval(:)
      nbloch = ngb-ngc
      if(ngc==0) goto 888

      allocate(ppovl(1:ngc,1:ngc))
      call readppovl0(q,ngc,ppovl)
      allocate(ooo(ngc,ngc),ppo(ngc,ngc),evec(ngc,ngc),eval(ngc))
      ooo= 0d0
      do ix=1,ngc
        ooo(ix,ix)=1d0
      enddo
      ppo = ppovl
      deallocate(ppovl)
      nmxx = ngc
      evec = 0d0
      eval = 0d0
      call diagcv(ooo, ppo,
     &     evec, ngc, eval, nmxx, 1d99, nev)
      write(6,*)' diagcv overlap ngc nev=',ngc,nev
      deallocate(ooo,ppo)
c
 888  continue
      sqovlp=0d0
      do i=1,nbloch
        sqovlp(i,i)=1d0
      enddo
      do i=1,ngc
        if(eval(i)<0d0) then
          call rx( 'getsqovlp:  eval(i) <0d0')
        endif
        do ix=1,ngc;  do iy=1,ngc
          sqovlp(ix+nbloch,iy+nbloch)=
     &     sqovlp(ix+nbloch,iy+nbloch)
     &      + evec(ix,i)* sqrt(eval(i))* dconjg(evec(iy,i))
        enddo ;      enddo
      enddo
      if(allocated(evec)) deallocate(evec)
      if(allocated(eval)) deallocate(eval)
      write(6,*)' end of getsqovlp'
c         sqovlpi = sqovlp
c         call matcinv(ngb,sqovlp)     !  inverse
c         ovlpi=ovlp
c         deallocate(ppovl,ovlp)
      end

c------------------------------------------------------------------
      subroutine tr_chkwrite(tagname,zw,iw,freqq,nblochpmx,nbloch,ngb,iq)
!! == check write for zw, no output == !!
      implicit none
      integer:: nblochpmx,nbloch,ngb,iw,i,iq
      complex(8):: zw(nblochpmx,nblochpmx),trwv,trwv2
      real(8):: freqq
c      logical :: smbasis
      character*(*)::tagname
      trwv=0d0
c      if(.not.smbasis()) then
        do i = 1,nbloch
          trwv = trwv + zw(i,i)
        enddo
c      endif
      trwv2 = 0d0
      do i = 1,ngb
        trwv2 = trwv2 + zw(i,i)
      enddo !  write(6,'(" realomg trwv=",2i6,4d22.14)') iq,iw,trwv(iw),trwv2(iw)
      write(6,'(a,f10.6,2i5,4d20.12)')trim(adjustl(tagname)),freqq,iq,iw,trwv,trwv2
c      do i = 1,ngb
c        write(6,'("iii i=",i4,a,f10.4,2i5,4d22.14)')i,tagname,freqq,iq,iw,zw(i,i)
c      enddo
      end

c------------------------------------------------------------------
c      subroutine test_xxx(tagname,zw,iw,freqq,nblochpmx,nbloch,ngb,iq)
c      implicit none
c      integer:: nblochpmx,nbloch,ngb,iw,i,iq
```

```
c      complex(8):: zw(nblochpmx,nblochpmx),trwv,trwv2
c      real(8):: freqq
c      logical :: smbasis
c      character*(*)::tagname
c      trwv2 = 0d0
c      forall( i = 1:ngb)
c        trwv2 = trwv2 + zw(i,i)
c      end forall
c      end
c--------------------------------------------------------------------

      function matcinvf(a) result(b)
!!== Test routine for Inversion ==
      implicit none
      integer :: info,n,n2(2)
      integer,allocatable :: ipiv(:)
      complex(8):: a(:,:), b(1)
      complex(8),allocatable:: work(:)
      n2= shape(a)
      n=n2(1)
      call zcopy(n,b,1,a,1)
      call zgetrf(n,n,a,n,ipiv,info)
      if(info/=0) then
        write(6,*)' matcinv: zegtrf info=',info
        call rx( ' matcinv: zegtrf ')
      endif
      allocate(work(n*n))
      call zgetri(n,a,n,ipiv,work,n*n,info)
      deallocate(work)
      if(info/=0) then
        write(6,*)'matcinv: zegtri info=',info
        call rx( 'matcinv: zegtri ')
      endif
      end

c--------------------------------------------------------------------
      subroutine diagno00(nbloch,wpvc,eval)
!! == ontain eigenvalue only for input complex matrix wpvc(nbloch,nbloch)
      implicit none
      integer:: nbloch,nmx,nev,i
      complex(8),allocatable:: ovlpc(:,:),evecc(:,:),wpvcc(:,:)
      real(8)::emx,eval(nbloch)
      complex(8):: wpvc(nbloch,nbloch)
      allocate( ovlpc(nbloch,nbloch),evecc(nbloch,nbloch),wpvcc(nbloch,nbloch))
      wpvcc= wpvc
      ovlpc= 0d0
      do i=1,nbloch
        ovlpc(i,i)=1d0
      enddo
      eval=0d0
      nev  = nbloch
      nmx  = nbloch
      call diagcv(ovlpc,wpvcc, evecc, nbloch, eval, nmx, 1d99, nev)
      deallocate(ovlpc,evecc,wpvcc)
      end

```

4.35 main/hx0fp0.sc.m.F File Reference

Functions/Subroutines

- program hx0fp0_sc
- subroutine tr_chkwrite (tagname, zw, iw, freqq, nblochpmx, nbloch, ngb, iq)

4.35.1 Function/Subroutine Documentation

4.35.1.1 program hx0fp0_sc ( )

Definition at line 1 of file hx0fp0.sc.m.F.

Here is the call graph for this function:

4.35.1.2 subroutine tr_chkwrite (character∗(∗) *tagname*, complex(8), dimension(nblochpmx,nblochpmx) *zw*, integer *iw*, real(8) *freqq*, integer *nblochpmx*, integer *nbloch*, integer *ngb*, integer *iq*)

Definition at line 1343 of file hx0fp0.sc.m.F.

Here is the caller graph for this function:

4.36 hx0fp0.sc.m.F

```
      program hx0fp0_sc
!!  Calculate W-V for QSGW mode.
!! We calculate chi0 by the follwoing three steps.
!!  tetwt5: tetrahedron weights
!!  x0kf_v4h: Accumlate Im part of the Lindhard function. Im(chi0) or Im(chi0^+-)
!!  dpsion5: calculate real part by the Hilbert transformation from the Im part

c      use m_readeps,only: read_eps, epsinv, w_mu, llmat2=>llmat,deallocate_eps
!!

      use m_readefermi,only: readefermi,ef
      use m_readqg,only: readngmx,readqg
      use m_readeigen,only: init_readeigen,init_readeigen2,readeval
      use m_read_bzdata,only: read_bzdata, !<--- 'call read_bzdata' sets up following data.
     &   ngrp2=>ngrp,nqbz,nqibz,n1,n2,n3,qbas,ginv,
     &   dq_,qbz,wbz,qibz,wibz,
     &     ntetf,idtetf,ib1bz, qbzw,nqbzw !for tetrahedron
c     &     idteti, nstar,irk,nstbz
      use m_genallcf_v3,only: genallcf_v3,
     &     nclass,natom,nspin,nl,nn,ngrp,
     &     nlmto,nlnmx, nctot,niw, !nw_input=>nw,
     &     alat, delta,deltaw,esmr,symgrp,clabl,iclass, !diw,dw,
     &     invg, il, in, im, nlnm,
     &     plat, pos, ecore, symgg

      use m_keyvalue,only: getkeyvalue
      use m_pbindex,only: pbindex !,norbt,l_tbl,k_tbl,ibas_tbl,offset_tbl,offset_rev_tbl
      use m_readqgcou,only: readqgcou

!! Base data to generate matrix elements zmel*. Used in "call get_zmelt".
      use m_rdpp,only: rdpp,    !"call rdpp" generate following data.
     & nblocha,lx,nx,ppbrd,mdimx,nbloch,cgr
!! Generate matrix element for "call get_zmelt".
      use m_zmel,only:    !these data set are stored in this module, and used when
     & nband,itq,ngcmx,ngpmx,    ppovlz,
     & ppbir,shtvg, miat,tiat , ntq
!! frequency
      use m_freq,only: getfreq,
     &   frhis,freq_r,freq_i, nwhis,nw_i,nw,npm !output of getfreq
!! antiferro
c      use m_anf,only: anfcond,
c     & laf,ibasf !,ldima,pos,natom
!! tetwt
      use m_tetwt,only: tetdeallocate,gettetwt, !followings are output of 'L871:call gettetwt')
     &  whw,ihw,nhw,jhw,ibjb,nbnbx,nhwtot,n1b,n2b,nbnb
!! w0 and w0i (head part at Gamma point)
      use m_w0w0i,only: w0w0i,
     & w0,w0i,llmat

!! MPI
      use m_mpi,only: mpi__hx0fp0_rankdivider2q,mpi__hx0fp0_rankdivider2s,
     &   mpi__qtask,mpi__initializeqspbm,mpi__finalize,mpi__root,
     &   mpi__broadcast,mpi__dblecomplexsendq,mpi__dblecomplexrecvq,mpi__rank,mpi__size,
     &   mpi__qranktab,mpi__consoleout,mpi__ss,mpi__se, mpi__allreducesums,
     &   mpi__barrier, mpi__rankq,mpi__rootq,mpi__roots
!! q0p
      use m_readq0p,only: readq0p,
     & wqt,q0i,nq0i ,nq0iadd,ixyz

      implicit none
      integer,allocatable:: nwgt(:,:)
      integer::iopen,maxocc2,iclose, ixc,iqxini,iqxend,
     &     ifhbe,  nprecb,mrecb,mrece,nlmtot,nqbzt,!nband,
     &     i,nq0ix,ngrpmx,mxx,nqbze,nqibze,ini,ix,ngrpx !ngcmx,
     &     ,nblochpmx,ndummy1,ndummy2,ifcphi,is,nwp, !ifvcfpout,,mdimx,nbloch
     &     ifepscond,nxx,ifvxcpout,ifgb0vec
     &     ,nw0,iw,ifinin,iw0,noccxv,noccx
     &     ,nprecx,mrecl,ifwd,ifrcwi,ifrcw,nspinmx,ifianf,ibas
     &     ,ibas1,irot,iq,ngb,iqixc2,ifepsdatnolfc,ifepsdat,ngbin,igc0
     &     ,kx,isf,kqxx,kp,job,nwmax !,ifev1,ifev2 !,nhwtot
     &     ,ihis,ik,ibib,ib1,ib2,ichkhis,ihww,j,imode
     &     ,  ifchipmlog ,   nw_w,nwmin  ! ,ngpmx
      real(8):: dum1,dum2,dum3,wqtsum,epsrng,dnorm, dwry,dwh,omg2, q(3),  qgbin(3),qx(3)
      real(8):: ua=1d0          ! this is a dummy.
      integer:: ifrb(2),ifcb(2),ifrhb(2),ifchb(2), ndble=8, nword
      integer,allocatable :: ngveccb(:,:), iqib(:),ifppb(:) !,lx(:) ngvecc(:,:),
      complex(8),allocatable:: geigb(:,:,:,:) ,geig(:,:),vcoul(:,:),
     &     zw(:,:),zw0(:,:), zxq(:,:,:),zxqi(:,:,:)
      real(8),allocatable :: eqt(:), !ppbrd (:,:,:,:,:,:,:),cgr(:,:,:,:)
     &     ppbrdx(:,:,:,:,:,:,:),aaa(:,:),symope(:,:),
     &     ppb(:,:),pdb(:,:),dpb(:,:),ddb(:,:), qbze(:,:),qibze(:,:) !,ecore(:,:)
c     &  freqr(:),freqi(:) !rw(:,:),cw(:,:) --->zw
      complex(8),allocatable :: rcxq(:,:,:,:)
      complex(8) :: fff,img=(0d0,1d0)
      complex(8),allocatable :: wwk(:,:,:)
      real(8) ::qbzx(3)
      logical :: debug
c      integer,allocatable:: ibasf(:)
      logical :: realomega, imagomega
      complex(8),allocatable:: zzr(:,:),ppovl(:,:),ppovlzinv(:,:) !,ppovlz(:,:)
      complex(8) :: epxxx,vcmean
      character*9 fileps
      character*15 filepsnolfc
c      logical :: paralellx0=.true. !, hist
      character(5) :: charnum5
      character(20):: xxt
      real(8) :: emin, emax      ,emax2,emin2
c     integer :: iSigma_en  !sf..21May02  !iSigma_en is integer
                                !parameter stored in GWIN_V2
                                !which determines approximation for  self-energy.
                                !Self-energy should be made hermitian for energies to be real
cxxx  !iSigma_en==0 SE_nn'(ef)+img integral:delta_nn'([SE_nn(e_n)+c.c.]/2-SE_nn(ef))
cxxx  !iSigma_en==1 SE_nn'(ef)+delta_nn'([SE_nn(e_n)+c.c.]/2-SE_nn(ef))
                                !iSigma_en==2 [SE_nn'((e_n+e_n')/2)+h.c.]/2
                                !iSigma_en==3 [(SE_nn'(e_n)+SE_nn'(e_n'))/2+h.c.]/2
      real(8) :: omg2max,omg1max,wemax
      logical::imagonly=.false. , noq0p !,readgwinput
      integer::nwin, incwfin, verbose,nbcut,nbcut2,ifpomat,nnmx,ikpo,nn_,noo,iqxxx,nomx
c      real(8)::efin
      logical :: nolfco=.false.
      integer:: isp1,isp2, ngc,mrecg ! bzcase,
      real(8)::  quu(3),deltaq(3),qqq(3)=0d0 !
      complex(8),allocatable:: wgt(:,:,:)
      real(8),allocatable:: qbz2(:,:)
      logical :: qbzreg
!    logical ::smbasis !smbasis will be implemented in m_zmel.f which generates <phi|phi M>
      real(8):: q_r(3)
      complex(8),allocatable:: pomat(:,:)
      logical   :: timereversal,onceww
      integer :: jpm,ncc
      real(8) :: frr !, sciss
      integer :: ngb0,ifvcoud,idummy,igb1,igb2,ngb_in,nmbas1,nmbas2,iq0,ifisk,iqx,ig,nmbas1x !ifepstinv,
      complex(8),allocatable:: zcousq(:,:),epstinv(:,:),epstilde(:,:),zcousqrsum(:,:,:),zcousqr(:,:),eemat(
:,:),zcousq0(:,:)
      real(8),allocatable:: vcousq(:),vcousq0(:),vcoudummy(:)
      real(8):: fourpi,sqfourpi,tpioa,absq,vcou1,vcou1sq
!! Eq.(40) in PRB81 125102
c     complex(8),allocatable::sk(:,:,:),sks(:,:,:),skI(:,:,:),sksI(:,:,:),
c     &  w_k(:,:,:),w_ks(:,:,:),w_kI(:,:,:),w_ksI(:,:,:), llw(:,:), llwI(:,:),
      complex(8),allocatable::sk(:,:,:),sks(:,:,:),ski(:,:,:),sksi(:,:,:),
     &     w_k(:),w_ks(:),w_ki(:), w_ksi(:)
      complex(8),allocatable:: llw(:,:), llwi(:,:),aaamat(:,:)
      integer:: lxklm,nlxklm,ifidmlx,ifrcwx,iq0xx,ircw,nini,nend,iwxx,nw_ixxx,nwxxx,niwxxx,iwx,icc1,icc2
      complex(8):: vc1vc2
      integer,allocatable:: neibz(:),ngrpt(:),igx(:,:,:),igxt(:,:,:),eibzsym(:,:,:)
      real(8),allocatable:: aik(:,:,:,:)
      integer,allocatable:: aiktimer(:,:)
      integer:: l2nl, nmbas_in , iqxendx,imb2 !iqqv,
      logical:: eibz4x0,tiii,iprintx,chipm=.false.,iqinit,localfieldcorrectionllw
      real(8)::qvv(3),ecut,ecuts,hartree,rydberg,pi
      character(128):: vcoudfile
      integer :: iqeibz
      complex(8):: epslfc, axxx(10)
      integer:: src,dest
      integer:: ifw0w0i
      logical :: symmetrize,eibzmode
      real(8):: schi=-9999 !dummy
      integer:: i_reduction_npm, i_reduction_nwhis,  i_reduction_nmbas2
      logical:: crpa
      integer,allocatable :: iclasst(:), invgx(:)
      integer:: ibasx,ificlass,ifile_handle,ifiq0p
      complex(8),allocatable:: ppovl_(:,:)
      logical:: tetra ,readw0w0itest=.false.
      integer::nw_ixx,nwxx

      logical:: w4pmode
      complex(8),allocatable:: wmu(:,:),wmuk(:,:)
      integer:: ifw4p,ngbq0,igb
      real(8):: qv(3,3)

      real(8)::ebmx
      integer:: nbmx,mtet(3)
      real(8),allocatable:: ekxx1(:,:),ekxx2(:,:)

c      integer:: ifief
c      real(8):: ef
!-------------------------------------------------------------------------
!TIME0_1001 ProgAll
!TIME0_11001 readbzdata
      call mpi__initializeqspbm()
      call mpi__consoleout('hx0fp0_sc')
      call cputid(0)
      allocate( zzr(1,1)) !dummy
      hartree= 2d0*rydberg()
      pi     = 4d0*datan(1d0)
      fourpi = 4d0*pi
      sqfourpi=sqrt(fourpi)
      write(6,*) ' --- hx0fp0_sc Choose omodes below ----------------'
      write(6,*) '  ixc= 11,10011,or 1011 '
      write(6,*) ' --- Put number above ! -----------------'
      if( mpi__root ) then
         read(5,*) ixc !c     call readin5(ixc,iqxini,iqxend)
      end if
      call mpi__broadcast(ixc)
      crpa=.false.
      if(ixc==0) call rx( ' --- ixc=0 --- Choose computational mode!')
      call headver('hx0fp0_sc',ixc)
c      call getkeyvalue("GWinput","ScaledGapX0",sciss,default=1d0)
c      write(6,"(' ScaledGapX0=',f8.3)") sciss
      if(ixc==11) then
         write(6,*) " OK ixc=11 normal mode "
      elseif(ixc==10011) then
         write(6,*) " OK ixc=10011 crpa mode "
         crpa=.true.
      elseif(ixc==1011) then
         write(6,*) 'OK ixc=1011 Add W0W0I part at q=0'
      else
         write(6,*)'we only allow ixc==11. given ixc=',ixc
         call rx( 'error:we only allow ixc==11.')
      endif
!! newaniso2 is now fixed to be .true.
      call getkeyvalue("GWinput","ecut_p" ,ecut, default=1d10 )
      call getkeyvalue("GWinput","ecuts_p",ecuts,default=1d10 )
c    Prof.Nagara says this cause a stop in ifort --->why???
c    write(6,*)'Timereversal=',Timereversal()

!! Readin BZDATA. See m_read_bzdata in gwsrc/rwbzdata.f
      call read_bzdata()

!TIME1_11001 "readbzdata"
!TIME0_12001 Q0P
!! Use regular mesh even for bzcase==2 and qbzreg()=T
!!      off-regular mesh for bzcase==1 and qbzreg()=F
c      if( ( bzcase()==2.and.qbzreg() )       .or.
c     &     ( bzcase()==1.and.(.not.qbzreg()))      ) then
!! this mechanism for qbzreg=F is too complicated. We may need to modify difinition of qbz for qbzreg=F.
      if(.not.qbzreg()) then ! set off-gamma mesh
         deltaq= qbas(:,1)/n1 + qbas(:,2)/n2 +qbas(:,3)/n3
         do i=1,nqbz
            qbz(:,i) = qbz(:,i) - deltaq/2d0
            write(6,"('i qbz=',i3,3f8.4)") i,qbz(:,i)
         enddo
      endif
      write(6,"(' nqbz nqibz ngrp=',3i5)") nqbz,nqibz,ngrp
!! === Readin by genallcf ===
!! See "use m_genallcf_v3" at the begining of this routine
!! We set basic data.
c      nwin   = 0                !Readin nw from NW file
      incwfin= 0                !use ForX0 for core in GWIN
c      efin =  0d0               !readin EFERMI
c--- EFERMI
      call readefermi()
      call genallcf_v3(incwfin) !in module m_genallcf_v3
      if(ngrp/= ngrp2) call rx( 'ngrp inconsistent: BZDATA and LMTO GWIN_V2')
c      nw_input = nw ;
c      write(6,*) 'nw delta=',nw_input,delta
      debug=.false.
      if(verbose()>=100) debug=.true.
      if(debug) write(6,*)' end of genallc'
      tpioa=2d0*pi/alat
!!!!  WE ASSUME iclass(iatom)= iatom,  nclass = natom.  !!!!!!!!!!!!!!!!!!!!!!!!!
      if(nclass /= natom) call rx( ' hx0fp0_sc: nclass /= natom ')
```

```
!! --- tetra or not
c      if(delta <= 0d0) then
      tetra =  .true.
      delta = -delta
      write(6,*)' hx0fp0.sc: tetrahedron mode delta=',delta
c      else
c         tetra = .false. ! switch for tetrahedron method for dielectric functions
c      endif
!! --- read dimensions of h,hb
      ifhbe      = iopen('hbe.d',1,0,0)
      read (ifhbe,*) nprecb,mrecb,mrece,nlmtot,nqbzt,nband,mrecg
      if(nlmto/=nlmtot) call rx( ' hx0fp0: nlmto/=nlmtot in hbe.d')
      if(nqbz /=nqbzt ) call rx( ' hx0fp0: nqbz /=nqbzt  in hbe.d')

!! --- Readin Offset Gamma --------
      call readq0p()
      write(6,"(' ### nqibz nq0i nq0iadd=', 3i5)")nqibz,nq0i,nq0iadd

c$$$      if(.not.newaniso2) then
c$$$         wqtsum = sum(abs(wqt(1:nq0i)))
c$$$         call getkeyvalue("GWinput","TestNoQ0P",noq0p,default=.false.)
c$$$      endif

!TIME1_12001 "Q0P"
!TIME0_13001 mptauof
      call getsrdpp2(nclass,nl,nxx)
      call readngmx('QGpsi',ngpmx)
      call readngmx('QGcou',ngcmx)
      write(6,*)' ngcmx ngpmx=',ngcmx,ngpmx
      nqbze  = nqbz *(1 + nq0i+nq0iadd)
      nqibze = nqibz + nq0i+nq0iadd
      allocate( qbze(3, nqbze), qibze(3, nqibze))
      call dcopy(3*nqbz, qbz,  1, qbze,1)
      call dcopy(3*nqibz,qibz, 1, qibze,1)
      do i = 1,nq0i+nq0iadd
         qibze(:,nqibz+i)  = q0i(:,i)
         ini = nqbz*(1 + i -1)
         do ix=1,nqbz
            qbze(:,ini+ix)   = q0i(:,i) + qbze(:,ix)
         enddo
      enddo
!! ---------------- dummy ngrpx=1 -------------------
      ngrpx = 1
      l2nl=2*(nl-1)
      allocate(symope(3,3))
      symope(1:3,1) = (/1d0,0d0,0d0/)
      symope(1:3,2) = (/0d0,1d0,0d0/)
      symope(1:3,3) = (/0d0,0d0,1d0/)
!!  dummy. Get space-group transformation information. See header of mptaouof.
      ificlass=ifile_handle()
      open (ificlass,file='CLASS')
      allocate(iclasst(natom),invgx(ngrp)
     & ,miat(natom,ngrp),tiat(3,natom,ngrp),shtvg(3,ngrp))
      write(6,*)'  --- Readingin CLASS info ---'
      do ibas = 1,natom
        read(ificlass,*) ibasx, iclasst(ibas)
        write(6, "(2i10)") ibasx, iclasst(ibas)
      enddo
      close(ificlass)
      call mptauof(symope,ngrpx,plat,natom,pos,iclasst
     o ,miat,tiat,invgx,shtvg ) !note: miat,tiat,shtvg are defined in m_zmel.
      if(verbose()>=40) write (*,*)' hsfp0.sc.m.F: end of mptauof'
!!  call rdpp gives ppbrd = radial integrals and cgr = rotated cg coeffecients.
      call rdpp(nxx, nl, ngrpx, nn, nclass, nspin, symope,qbas)
      ntq=nband
      allocate(itq(ntq))
      do i=1,ntq
        itq(i)=i
      enddo
!! Pointer to optimal product basis
c      allocate(imdim(natom))
c      call indxmdm (nblocha,nclass,iclass,natom,
c     o imdim )                  !in m_zmel
      nblochpmx = nbloch + ngcmx
      allocate(ngveccb(3,ngcmx))
      iqxend = nqibz + nq0i + nq0iadd
      write(6,*) ' nqibz nqibze=',nqibz,nqibze
!TIME1_13001 "mptauof"
!TIME0_14001 init_readeigen
!!... initialization of readEigen !readin m_hamindex
      call init_readeigen(ginv,nspin,nband,mrece)!EVU EVD are read in init_readeigen
      call init_readeigen2(mrecb,nlmto,mrecg)
c     --- ecore ---
c     allocate(ecore(nctot,nspin)) !core energies
c     do  is = 1,nspin
c     if (nctot .gt. 0) then
c     call catch1 (w(iecore),is,nctot,2,ecore(:,is)) !core energies
```

```
c     write(6,*)' ecore is=',is,ecore(:,is)
c     endif
c     enddo

c     --- set realomega, imagomega tetra nw niw nwp ifgb0vec ---------------------
!     nwp, freq_r,  frhis(1:nwhis+1)
c     if   ( ixc==1 ) then !old imagw = 2 case
c     realomega =.true.
c     imagomega =.true.
c     stop 'hsfp0sc: ixc==1 is not implimented'
cccccccccccccccccccfaleev 21May02,  use only ixc=1,11 modes ccccccccc
c     elseif( ixc==2.or.ixc==3 ) then
c     realomega =.true.
c     imagomega =.false.
c     niw = 0
c     ifepscond = 2102
c     open (ifepscond,file='EPScond')
c     read (ifepscond,*) epsrng, dwry !epsrng dw in Ry
c     dw = dwry/2d0
c     close(ifepscond)
c     if(dw==0d0) then
c     nw = 1
c     else
c     nw = (epsrng/2d0 - 1d-10)/(dw/2d0) + 2 !epsrng/2d0 corresponds to in a.u.
c     endif
c     allocate(epsi(nw,neps))
c     elseif(ixc==4.or.ixc==5.or.ixc==6) then
c     ! ... These are test modes.
c     ! ixc=4 tetrahedren weight test. tetwt5.vs.tetwt5. Write tethis.chk
c     ! ixc=5 Spectrum function (Img part) along the Real axis with tetwt4
c     ! ixc=6 Spectrum function (Img part) along the Real axis with tetwt5. Histgram method.
c     realomega = .true.
c     imagomega = .false.
c     tetra     = .true.
c     niw = 0
c     ! ---  For tetwt5 ---  the tetrahedron weight for spectrum function (imaginary part)
c     !   Histogram bins are specified by freq_r(1:nwp)
c     !     nwp=nw+1; frhis(1)=0
c     !     The 1st   bin  is     [frhis(1),  frhis(2)]   ...
c     !     The last  bin  is     [frhis(nw), frhis(nwp)].
c
c     ! ... These parameters specifies a test histgram bins;Sergey's mesh just for test modes.
c     nw0 = 200    !100    800
c     dwh = 0.01d0 !0.02d0 0.0025d0 !in hartree
c     ! ...
c     call findemaxmin(ifev,nband,nqbz,nspin,emax,emin)
c     if (nctot .gt. 0) Emin = minval(ecore)
c     omg2max = (Emax-Emin)*.5+.2d0    !(in Hartree) covers  all relevant omega, +.2 for margin
c     omg1max = dwh*(nw0-1)
c     nwp = int(sqrt(omg2max*(2*nw0-1d0)/dwh-(nw0**2-3*nw0+1d0)))+1 ! + 1 for margin
c     nw  = nwp-1
c     write(6,*) Emax,Emin,nw0,nw  ! nwp is new max number in frequency array
c     write(6,'(a32,2i7,2d15.3)')'hx0fp1: nw0,nw,omg1max,omg2max='
c     &              , nw0,nw,  omg1max,omg2max
c     if (nw <= nw0) stop 'hx0fp0:ixc==[456] nw2 <= nw'
c     allocate(freq_r(nwp))
c     do iw=1,nwp  !This is a test mesh by Sergey.Faleev
c     if(iw<=nw0) then;  freq_r(iw)=dwh*(iw-1)
c     else;  freq_r(iw)=dwh*(iw**2+nw0**2-3*nw0+1)/(2*nw0-1d0)
c     endif
c     enddo !freq_r(iw) is linear for iw<=nw and quadratic for nw<iw<=nw2
c     !freq_r(iw) chosen in such a way that it is continues with
c!!!  nw nwp=nw+1 freq_r(1:nwp) are used after here.
c     allocate(frhis(nwp))
c     frhis=freq_r(1:nwp)
c     nwhis=nw

!! We get frhis,freq_r,freq_i, nwhis,nw,npm  by getfreq
      realomega = .true.
      imagomega = .true.
      tetra     = .true.
      call findemaxmin(nband,qbze,nqbze,nspin, emax,emin)
      if(.not.qbzreg()) then
         allocate(qbz2(3,nqbz))
         do iq=1,nqbz
            qbz2(:,iq)=qbz(:,iq)+dq_
         enddo
         call findemaxmin(nband,qbz2,nqbz,nspin ,emax2,emin2)
         emax=max(emax,emax2)
         emin=min(emin,emin2)
         deallocate(qbz2)
      endif
      if (nctot > 0) emin=minval(ecore(:,1:nspin))
      omg2max = (emax-emin)*.5d0+.2d0
             ! (in Hartree) covers all relevant omega, +.2 for margin
      if(mpi__root) write(6,"(' emin emax omega2max=',3f13.5)") emin, emax, omg2max
      call getwemax(.true.,wemax) !wemax is to determine nw !real axis divisions
      if(mpi__root) write(6,"(' wemax=  ',f13.4)") wemax
      call getfreq(.false.,realomega,imagomega,tetra,omg2max,wemax,niw,ua,mpi__root)
      nwp = nw+1

!! We first accumulate Imaginary parts. Then do K-K transformation to get real part.
      noccxv = maxocc2(nspin,ef, nband, qbze,nqbze)
        !max no. of occupied valence states
      if(noccxv>nband) call rx( 'hx0fp0_sc: all the bands filled! too large Ef')
      noccx  = noccxv + nctot
      nprecx = ndble        !We use double precision arrays only.
      mrecl  = nprecx*2*nblochpmx*nblochpmx/nword()
      if(mpi__root)then
        ifwd   = iopen('WV.d',1,-1,0)
        write (ifwd,"(1x,10i14)") !"(1x,i3,i8,i5,5i4)")
     &  nprecx,mrecl,nblochpmx,nwp,niw,nqibz + nq0i-1,nw_i
        ifwd = iclose('WV.d'); ifwd=0
      endif
      allocate(  zw(nblochpmx,nblochpmx) )
      nspinmx = nspin
!TIME1_14001 "init_readeigen"
!TIME0_15001 ppbafp_v2

!!... these are used x0k
      call getkeyvalue("GWinput","nbcutlow",nbcut, default=0 )
      call getkeyvalue("GWinput","nbcutlowto",nbcut2, default=0 )
      write(6,"(' nbcut nbcutlowto=',2i5)") nbcut,nbcut2
!! -- ppb= <Phi(SLn,r) Phi(SL'n',r) B(S,i,Rr)>
!! This is general for rotated CG coefficient; but hx0fp0 mode is only for  ngrpx=1 (not rotated).
!! Compare usage in hsfp0 modes.
      irot=1
      allocate( ppbir(nlnmx*nlnmx*mdimx*nclass,irot,nspin))
      do is = 1,nspin
        call ppbafp_v2(irot,ngrpx,is,nspin,
     i   il,in,im,nlnm,         !w(i_mnl),
     i   nl,nn,nclass,nlnmx,
     i   mdimx,lx,nx,nxx,       !Bloch wave
     i   cgr, nl-1,             !rotated CG
     i   ppbrd,                 !radial integrals
     o   ppbir(:,irot,is))      !this is in m_zmel
      enddo
      if(debug) write(6,*) ' end of ppbafp_v2'
!TIME1_15001 "ppbafp_v2"
!TIME0_16001 readqgcou
      call getkeyvalue("GWinput","nbcutlow",nbcut, default=0 )
      call getkeyvalue("GWinput","nbcutlowto",nbcut2, default=0 )
      write(6,"(' nbcut nbcutlowto=',2i5)") nbcut,nbcut2
      iqxini=1 !for newaniso
      eibzmode = eibz4x0()

!! nov2016 moved from tetwt5 --> here
      call getkeyvalue("GWinput","nband_chi0",nbmx, default=nband )
      call getkeyvalue("GWinput","emax_chi0", ebmx, default=1d10  )
      mtet=(/1,1,1/)
      call getkeyvalue("GWinput","multitet",mtet,3,default=(/1,1,1/))
      ! multitet=T ==> micro tetrahedron method (divided-tetrahedron). Not used so much now...
      allocate(ekxx1(nband,nqbz),ekxx2(nband,nqbz))

!! === Use of symmetry. EIBZ procedure PRB81,125102 ===
!!  For rotation of zcousq.  See readeigen.F rotwv.F ppbafp.fal.F(for index of product basis).
      if(eibzmode) then
!! commentout block inversion Use iqxendx=iqxend because of full inversion
         iqxendx=iqxend
         allocate( nwgt(nqbz,iqxini:iqxendx), !qeibz(3,nqbz,iqxini:nqibz),neibz(iqxini:nqibz),
     &        igx(ngrp*2,nqbz,iqxini:iqxendx),igxt(ngrp*2,nqbz,iqxini:iqxendx),
     &        eibzsym(ngrp,-1:1,iqxini:iqxendx))
         iprintx=.false.

         write(6,*)
         write(6,"('=== Goto eibzgen === TimeRevesal switch =',l1)")timereversal()
         if(mpi__root) iprintx=.true.
         call eibzgen(nqibz,symgg,ngrp,qibze(:,iqxini:iqxend),iqxini,iqxendx,qbz,nqbz,
     i        timereversal(),ginv,iprintx,
     o        nwgt,igx,igxt,eibzsym,tiii)
         write(6,"('Used timeReversal for EIBZ = ',l1)") tiii
         call cputid(0)
c$$$
c$$$         write(6,"('TimeRevesal switch = ',l1)") timereversal()
c$$$         call
 eibzgen(nqibz,symgg,ngrp,qibze(:,iqxini:iqxend),iqxini,iqxendx,qbz,nqbz,timereversal(),ginv,iprintx,
c$$$     o        nwgt,igx,igxt,eibzsym)
c$$$!! Check timereversal is required for symmetrization operation or not. If not tiii=timereversal=F is
 used.
c$$$!! this is because the symmetrization is a little time-consuming.
c$$$         tiii=timereversal()
c$$$         if(minval(igxt)==1) tiii=.false.
c$$$         iprintx=.true.
c$$$cccccccccccccccccccc
```

```
c$$$c     tiii=.true.
c$$$cccccccccccccccccccc
c$$$         write(6,"('=== goto eibzgen === used timereversal=',l1)")tiii
c$$$         call eibzgen(nqibz,symgg,ngrp,qibze(:,iqxini:iqxend),iqxini,iqxendx,qbz,nqbz,tiii,ginv,iprintx,
c$$$     o        nwgt,igx,igxt,eibzsym)

!All input. this returns required index stored in arrays in m_pbindex.
         call pbindex(natom,lx,l2nl,nx)
                          ! PBindex: index for product basis.  We will unify this system; still similar is used in ppbafp_v2.
         call readqgcou() ! no input. Read QGcou and store date into variables.
!!  call Spacegrouprot(symgg,ngrp,plat,natom,pos) ! all inputs.
      else !dummy allocation to overlaid -check bound !sep2014
         iqxendx=iqxend
         allocate( nwgt(1,iqxini:iqxendx),igx(1,1,iqxini:iqxendx)
     &    ,igxt(1,1,iqxini:iqxendx), eibzsym(1,1,iqxini:iqxendx)) !dummy
      endif

      allocate( llw(nw_i:nw,nq0i), llwi(niw,nq0i) )
      llw=1d99
      llwi=1d99
      if(ixc==1011)then !ixc==11 is a debug mode to test contrib. at \Gamma point.
         goto 1191
      endif
!! for w4phonon. all nodes have wmu array.
      w4pmode=.false.
      if(sum(ixyz)/=0) w4pmode=.true.
      if(w4pmode) then
        allocate( wmuk(2:nblochpmx,3))
        wmuk=1d99
      endif

!! rank divider
      call mpi__hx0fp0_rankdivider2q(iqxini,iqxend)
      call mpi__hx0fp0_rankdivider2s(nspinmx)

!! == Calculate x0(q,iw) and W == main loop 1001 for iq.
!! NOTE: iq=1 (q=0,0,0) write 'EPS0inv', which is used for iq>nqibz for ixc=11 mode
!! Thus it is necessary to do iq=1 in advance to performom iq >nqibz.
!! (or need to modify do 1001 loop).
!! ---------------------------------------------------------------
!! === do 1001 loop over iq ============================================
!! ---------------------------------------------------------------
      iqinit=.true.
      write(6,'("irank=",i5," allocated(MPI__qtask)=",L5)')mpi__rank,allocated(mpi__qtask)
      do iq = iqxini,iqxend
        if(mpi__qtask(iq)) write(6,'("irank iq=",i5,i5)') mpi__rank,iq
      enddo

!! Get ngbq0 (for q=0) and broadcast for w4p
      if( mpi__root.and. w4pmode ) then
        q = (/0d0,0d0,0d0/)
        call readqg('QGcou', q, ginv,  quu,ngc,ngveccb)
        ngbq0 = nbloch+ngc
      endif
      call mpi__broadcast(ngbq0)

!TIME1_16001 "readqgcou"
!TIME0_170001 do1001
      do 1001 iq = iqxini,iqxend
        if( .not. mpi__qtask(iq) ) cycle
        if (mpi__roots) then
          ifrcwi = iopen('WVI.'//charnum5(iq),0,-1,mrecl)
          ifrcw  = iopen('WVR.'//charnum5(iq),0,-1,mrecl)
        endif
!!
        call cputid(0)
        q = qibze(:,iq)
        call readqg('QGcou', q, ginv,  quu,ngc,ngveccb) ! q was qq

!! Caution : confusing point
!!  ngc by QGcou is shown at the bottom of lqg4gw.
!!  ngc read from PPOVL are given by rdata4gw.
!!  Note that  ngc(iq>nqibz )=ngc (q=0), because when it is generated in mkqg.F
!!
c         if( newaniso2.and.iq==1 ) then ! *sanity check
        if( iq==1 ) then        ! *sanity check
          if(sum(q**2)>1d-10) call rx( ' hx0fp0.sc: sanity check. |q(iqx)| /= 0')
        endif

!! ==== readin Coulomb matrix ====
        ngb = nbloch + ngc
        write(6,"('do 1001: iq q=',i5,3f9.4)")iq,q
        write(6,*)'nbloch ngb ngc=',nbloch,ngb,ngc

!! === readin diagonalized Coulomb interaction ===
!! zcousq: E(\nu,I), given in PRB81,125102; vcousq: sqrt(v), as well.
c         if(newaniso2) then
        vcoudfile='Vcoud.'//charnum5(iq) ! iq was iqqv this is closed at the end of do 1001
        ifvcoud = iopen(trim(vcoudfile),0,-1,0)
        read(ifvcoud) ngb0
        read(ifvcoud) qvv
        if(sum(abs(qvv-q))>1d-10) then
          write(6,*)'qvv =',qvv
          call rx( 'hx0fp0: qvv/=0 hvcc is not consistent')
        endif
        if(allocated(zcousq)) deallocate( zcousq,vcousq )
        allocate( zcousq(ngb0,ngb0),vcousq(ngb0))
        read(ifvcoud) vcousq
        read(ifvcoud) zcousq
        idummy=iclose(trim(vcoudfile))
        vcousq=sqrt(vcousq)

c         if(newaniso2.and. iq>nqibz.and.(.not.localfieldcorrectionllw())  ) then
        if(iq>nqibz.and.(.not.localfieldcorrectionllw())  ) then
          if( ngb0/=ngb ) then
            call rx( 'hx0fp0.m.f:ngb0/=ngb')
          endif
          nolfco =.true.
          nmbas_in = 1
c         elseif(newaniso2) then !.and.iq==1) then
        else
          nolfco = .false.
          nmbas_in = ngb
        endif
        nmbas1 = nmbas_in
        nmbas2 = nmbas1

!! newaniso=T case. Used in get_zmelt in m_zmel called in x0kf_v4hz
        if(allocated(ppovlz)) deallocate(ppovlz)
        if(allocated(ppovlzinv)) deallocate(ppovlzinv)
        if(allocated(ppovl)) deallocate(ppovl)
        allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb),   ppovlzinv(ngb,ngb))
        call readppovl0(q,ngc,ppovl) !q was qq
c         ppovlz(1:nbloch,:) = zcousq(1:nbloch,:)
c         ppovlz(nbloch+1:nbloch+ngc,:)
c     &     = matmul(ppovl,zcousq(nbloch+1:nbloch+ngc,:))
        allocate(ppovl_(ngb,ngb))
        ppovl_=0d0
        do i=1,nbloch
          ppovl_(i,i)=1d0
        enddo
        ppovl_(nbloch+1:nbloch+ngc,nbloch+1:nbloch+ngc)=ppovl
        if(.not.eibz4x0()) then !sep2014 added for eibz4x0=F
          ppovl_= matmul(ppovl_,zcousq)
        endif
        ppovlz = ppovl_
        deallocate(ppovl_,ppovl)

c$$$  if(ixc==11) then
c$$$  write(6,*)" xxx2: memsize 8*ngb*ngb*nwhis=", 8*ngb*ngb*nwhis,' ngb nwhis=',ngb,nwhis
c$$$  allocate( rcxq(ngb,ngb,nwhis,npm) )
c$$$  rcxq=(0d0,0d0)
c$$$  else
c$$$  if(onceww(2)) write(6,*)" xxx2:allocate zxq zxqi memsize 16*ngb*ngb*(nwp+niw)=",
c$$$  &    16*ngb*ngb*(1+nwp+niw),' ngb nwp niw=',ngb,nwp,niw
c$$$  allocate(
c$$$  &    zxq (ngb,ngb,nw_i:nw),   !,nwp) feb2006
c$$$  &    zxqi(ngb,ngb,niw))
c$$$  zxq=0d0; zxqi=0d0
c$$$  endif

        allocate( rcxq(nmbas1,nmbas2,nwhis,npm) )
        allocate( zw0(ngb,ngb) ) !, zxq (ngb,ngb,nw_i:nw), zxqi(ngb,ngb,niw) )
        rcxq = 0d0

!! ---------------------------------------------------------------
!! === loop over spin=== ========================================
!! ---------------------------------------------------------------
!TIME0_180001 Do1003
!         do 1003 is = 1,nspinmx
        do 1003 is = mpi__ss,mpi__se
          write(6,"(' ### ',2i4,' out of nqibz+n0qi+nq0iadd nsp=',2i4,' ### ')")
     &     iq, is, nqibz + nq0i+nq0iadd, nspin
          if(debug) write(6,*)' niw nw=',niw,nw
          isf = is

!! Tetrahedron weight.
!! output
!!     nbnbx
!!     ihw(ibjb,kx): omega index, to specify the section of the histogram.
!!     nhw(ibjb,kx): the number of histogram sections
!!     jhw(ibjb,kx): pointer to whw
!!     whw( jhw(ibjb,kx) ) \to whw( jhw(ibjb,kx) + nhw(ibjb),kx)-1 ), where ibjb=ibjb(ib,jb,kx)
!!     : histogram weights for given ib,jb,kx for histogram sections
!!     from ihw(ibjb,kx) to ihw(ibjb,kx)+nhw(ibjb,kx)-1.
c            write(6,*) ' --- goto x0kf_v4hz ---- newaniso= ',newaniso2
!! input
!!     ekxx1 for   rk,is
!!     ekxx2 for q+rk,isf
          do kx = 1, nqbz
            call readeval(qbz(:,kx),   is,  ekxx1(1:nband, kx) )
            call readeval(q+qbz(:,kx), isf, ekxx2(1:nband, kx) )
          enddo
          call gettetwt(q,iq,is,isf,nwgt(:,iq),frhis,nwhis,npm,
     i     qbas,ginv, ef, nqibz, nband,ekxx1,ekxx2, nctot,ecore,
     i     nqbz,qbz,nqbzw,qbzw,  ntetf,idtetf,ib1bz,
     i     nbmx,ebmx,mtet,eibzmode) !nov2016

!! == x0kf_v4hz is the main routine to accumalte imaginary part of x0 ==
          iqeibz=iq
          if(npm==1) then
            ncc=0
          else
            ncc=nctot
          endif
          call x0kf_v4hz(npm,ncc,
     i     ihw,nhw,jhw,whw,nhwtot, ! tetwt5
     i     n1b,n2b,nbnbx,nbnb,  ! use whw by tetwt5 ,
     i     q,
     i     nspin,is,isf, !symmetrize, !
     i     qbas,ginv,  qbz,wbz,
     d     nlmto,nqbz,nctot,    !noccx,noccxv,
     d     nbloch,  nwhis,      !nlnmx,mdimx,
     i     iq,ngb,ngc,ngpmx,ngcmx, !ngb/=ngc+nbloch for smbasis()=T oct2005
     i     nqbze,nband,nqibz,
     o     rcxq,                ! rcxq is the accumulating variable for spins
     i     nolfco,zzr,nmbas_in, zcousq, !ppovl,nmbas2 is removed.,nmbas1 ppovlz,
     i     chipm,eibzmode,      !z1offd,
     i     nwgt(:,iqeibz),igx(:,:,iqeibz),igxt(:,:,iqeibz),ngrp, eibzsym(:,:,iqeibz),crpa)
          write(6,*)' end of x0kf_v4h sum rcxq=',sum(abs(rcxq))
          call tetdeallocate() !deallocate(ihw,nhw,jhw, whw,ibjb )
c          if(tetra)  deallocate( n1b,n2b)
 1003   continue;write(6,*) 'end of spin-loop nwp=',nwp !end of spin-loop
!TIME1_180001 "Do1003"
c===========end of spin loop===========================================

!! symmetrize and convert to Enu basis by dconjg(tranpsoce(zcousq)*rcxq8zcousq if eibzmode
!TIME0_190001 x0kf_sym
        if(eibzmode)  then
          is=1                  ! dummy
          call x0kf_v4hz_symmetrize(npm,!ncc,
c     i     ihw,nhw,jhw,whw,nhwtot, ! tetwt5
c     i     n1b,n2b,nbnbx,nbnb,  ! use whw by tetwt5 ,
     i     q,
     i     nspin,is,isf, !symmetrize, !
     i     qbas,ginv,  !qbz,wbz,
c     i              nblocha,!nlnm,nlnmv,nlnmc,iclass,
c     i              ppb(1,is),
c     i              icore,ncore,
c     d     nlmto,nqbz,nctot,    !noccx,noccxv,
c     d              natom, !nl,nclass,natom,nnc,
     d     nbloch,  nwhis,      !nlnmx,mdimx,
     i     iq,ngb,ngc,ngpmx,ngcmx, !ngb/=ngc+nbloch for smbasis()=T oct2005
     i     nqbze,nband,nqibz,
     o     rcxq,                ! rcxq is the accumulating variable for spins
     i     nolfco,zzr,nmbas_in, zcousq, !ppovl,nmbas2 is removed.,nmbas1 ppovlz,
     i     chipm,eibzmode,      !z1offd,
     i     ngrp, eibzsym(:,:,iqeibz))
        endif

!! reduction rcxq in the S-axis
        write(6,*) 'MPI__AllreduceSumS start'
        do i_reduction_npm=1,npm
          do i_reduction_nwhis=1,nwhis
            do i_reduction_nmbas2=1,nmbas2
              call mpi__allreducesums(
     .         rcxq(1,i_reduction_nmbas2,i_reduction_nwhis,i_reduction_npm), nmbas1)
            enddo
          enddo
        enddo
        write(6,*) 'MPI__AllreduceSumS end'
!TIME1_190001 "x0kf_sym"
!TIME0_200001 "HilbertTransformation"
!! --- Hilbert transform.  Genrerate Real part from Imaginary part. ======
        if(allocated(zxq) ) deallocate(zxq,zxqi)
        allocate(zxq(nmbas1,nmbas2,nw_i:nw), zxqi(nmbas1,nmbas2,niw))
        write(6,'("goto dpsion5: nwhis nw_i niw nw_w nmbas1 nmbas2=",6i5)') nwhis,nw_i,nw,niw,nmbas1,nmbas2
        write(6,*)' -------- nmbas1,nmbas2=', nmbas1,nmbas2
        call dpsion5(frhis,nwhis, freq_r, nw, freq_i,niw, realomega, imagomega,
     i   rcxq, npm,nw_i, nmbas1,nmbas2, ! rcxq is alterd---used as work
     o   zxq, zxqi,
     i   chipm, schi,is,  ecut,ecuts)
        write(6,*)' --- end of dpsion5 ----',sum(abs(zxq)),sum(abs(zxqi))
        if(allocated(rcxq) ) deallocate(rcxq)
!TIME1_200001 "HilbertTransformation"

!! ===  RealOmega ===
        if (realomega) then
!TIME0_210001 ralloc
          if (nspin == 1) zxq = 2d0*zxq !if paramagnetic, multiply x0 by 2
          nwmax = nw
          nwmin = nw_i
!! prepare for iq0.
          iq0 = iq - nqibz
c               if(newaniso2) then
c$$$                 if( iq==1 ) then
c$$$                    write(6,*)'open EPS0inv mpi=',MPI__rank
c$$$                   ifepstinv = iopen('EPS0inv',0,-1,0)
c$$$                   write(ifepstinv) ngb
c$$$                 endif
c$$$
c$$$             if(iqinit) then
c$$$               allocate( sk(ngb,nwmin:nwmax,nq0i),  sks(ngb,nwmin:nwmax,nq0i) )
c$$$               allocate( skI(ngb,niw,nq0i), sksI(ngb,niw,nq0i))
c$$$               iqinit=.false.
c$$$             endif
          allocate(epstilde(ngb,ngb))
          allocate(epstinv(ngb,ngb))
c               endif
!KINO                write(6,*)'kino: nwmin,nwmax,ngb=',nwmin,nwmax,ngb
          write(6, *)" === trace check for W-V === nwmin nwmax=",nwmin,nwmax
!TIME1_210001 "ralloc"
!TIME0_2200011 do1015
          do 1015 iw  = nwmin,nwmax
            frr= dsign(freq_r(abs(iw)),dble(iw))
            imode = 1
c                  if(newaniso2.and.iq<=nqibz) then !for mmmw
            if(iq<=nqibz) then  !for mmmw
              if(iq==1) then
                ix=1
                zw0(:,1)=0d0
                zw0(1,:)=0d0
              else
                ix=0
              endif

!!  Eqs.(37),(38) in PRB81 125102 (Friedlich)
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
                  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
                enddo
              enddo
              epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
              call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))

!! w4p writing eps
              if(iw==0.and.w4pmode) then
                !static epstinv is saved. For q=0 epstilde (mu=1 skipped). For q/=0 full matrix inversion.
                             !(ix=1 is set for q=0)
                ifw4p = ifile_handle()
                open(ifw4p,file='W4PHONON.'//charnum5(iq),form='unformatted')
                write(ifw4p) iq,q,ngb,ix !ix=0, or ix=1 for q=0 (iq=1)
                write(ifw4p) epstinv(ix+1:ngb,ix+1:ngb)
                close(ifw4p)
              endif
!TIME0_3000011 zweqzw0

c$$$  cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c$$$  cmmm direct inversion vs. block inversion
c$$$  if(iq>nqibz) then
c$$$  c direct inversion
c$$$  ix=0
c$$$  do igb1=ix+1,ngb
c$$$  do igb2=ix+1,ngb
c$$$  epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
c$$$  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
c$$$  enddo
c$$$  enddo
c$$$  epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
c$$$  call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
c$$$  do igb1=1+ix,ngb
c$$$  do igb2=1+ix,ngb
c$$$  zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
c$$$  if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
c$$$  enddo
```

```
c$$$  enddo
c$$$  c              write(6,"('mmmmzp99x  ',i3,10(2d13.5,2x))") iw,zw0(1,1),zw0(2:10:3,1),zw0(63:70:3,1)
c$$$  write(6,"('mmmmzp99x  ',i3,10(2d13.5,2x))") iw,1d0/epstinv(1,1),zw0(2:10:3,1),zw0(63:70:3,1)
c$$$  c             write(6,"('mmmmzp99x  ',i3,10(2d13.5,2x))") iw,zw0(1,1),zw0(1,2:10:3),zw0(1,63:70:3)
c$$$  c block inversion
c$$$  ix=1
c$$$  do igb1=ix+1,ngb
c$$$  do igb2=ix+1,ngb
c$$$  epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
c$$$  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
c$$$  enddo
c$$$  enddo
c$$$  epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
c$$$  call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
c$$$  absq=sqrt(sum(q**2*tpioa**2))
c$$$  sk(  1:ngb)= zxq(1,1:ngb,iw)
c$$$  sks( 1:ngb)= zxq(1:ngb,1,iw)
c$$$  w_k(1) =0d0
c$$$  w_ks(1)=0d0
c$$$  w_k( 2:ngb)= vcousq(2:ngb)*vcousq(1)*matmul(vcousq(1)*sk(2:ngb)*vcousq(2:ngb),epstinv(2:ngb,2:ngb))
c$$$  w_ks(2:ngb)= vcousq(2:ngb)*vcousq(1)*matmul(epstinv(2:ngb,2:ngb),vcousq(1)*sks(2:ngb)*vcousq(2:ngb))
c$$$  llw(iw,iq0)=
c$$$  &             1d0
c$$$  &            -vcousq(1)*sk(1)*vcousq(1) ! sk(1,1,iw)=sks(1,1,iw)=H of Eq.(40).
c$$$  &            -vcousq(1)*vcousq(1)* sum( vcousq(2:ngb)*sk(2:ngb) *
     matmul(epstinv(2:ngb,2:ngb),sks(2:ngb)*vcousq(2:ngb)))
c$$$  write(6,"('mmmmzwp99x ',i3,10(2d13.5,2x))") iw,llw(iw,iq0), !(1d0/llw(iw,iq0)-1d0)*vcousq(1)**2,
c$$$  c     &                 w_k(2:10:3)/llw(iw,iq0), w_k(63:70:3)/llw(iw,iq0)
c$$$  &                 w_ks(2:10:3)/llw(iw,iq0), w_ks(63:70:3)/llw(iw,iq0)
c$$$  write(6,"('mmmmzwp99x ')")
c$$$  endif
c$$$  ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
              do igb1=1+ix,ngb
                do igb2=1+ix,ngb
                  zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
                  if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
                enddo
              enddo
c$$$                    if(iq==1) write(ifepstinv) epstinv(ix+1:ngb,ix+1:ngb),iq,iw
              zw(1:ngb,1:ngb) = zw0
!TIME1_3000011 "zweqzw0"
!TIME0_3100011 tr_chkwrite
              if (mpi__roots)then
                write(ifrcw, rec= iw-nw_i+1 ) zw !  WP = vsc-v
              endif
              call tr_chkwrite("freq_r iq iw realomg trwv=", zw, iw, frr,nblochpmx, nbloch,ngb,
      iq)
!TIME1_3100011  "tr_chkwrite"
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     if(iq>nqibz) then
c     write(6,"('mmmmz99x ',i3,10(2d13.5,2x))") iw,zw0(1,1)+vcousq(1)**2,zw0(2:10:3,1),zw0(63:70:3,1)
c     endif
c     if(iq==1.or.iq>nqibz) then
c     write(6,"('mmmz0  ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(1,2:10:3,iw),zxq(1,63:70:3,iw)
c     write(6,"('mmmz0* ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(2:10:3,1,iw),zxq(63:70:3,1,iw)
c     write(6,"('mmmmz99x ',i3,10(2d13.5,2x))") iw,zw0(1,1)+vcousq(1)**2,zw0(1,2:10:3),zw0(1,63:70:3)
c     write(6,"('mmmzx  ',2i3,10(2d13.5,2x))") iq,iw,zxq(2,1,iw),zxq(2,2:10:3,iw),zxq(2,63:70:3,iw)
c     write(6,"('mmmzx  ',2i3,10(2d13.5,2x))") iq,iw,zxq(3,1,iw),zxq(3,2:10:3,iw),zxq(3,63:70:3,iw)
c     write(6,"('mmmzxs ',2i3,10(2d13.5,2x))") iq,iw,zxq(1,1,iw),zxq(2:10:3,1,iw),zxq(63:70:3,1,iw)
c     write(6,"('mmmzxs ',2i3,10(2d13.5,2x))") iq,iw,zxq(1,2,iw),zxq(2:10:3,2,iw),zxq(63:70:3,2,iw)
c     write(6,"('mmmmzee',2i3,10(2d13.5,2x))")iq,iw,epstilde(2,2),epstilde(2,2:10:3),epstilde(2,63:70:3)
c     write(6,"('mmmmzee',2i3,10(2d13.5,2x))")iq,iw,epstilde(3,2),epstilde(3,2:10:3),epstilde(3,63:70:3)
c     endif
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
            endif

c                 if(newaniso2.and.iq>nqibz) then
            if(iq>nqibz) then
!! Full inversion to calculalte eps with LFC.
              vcou1 = fourpi/sum(q**2*tpioa**2) ! --> vcousq(1)**2!  !fourpi/sum(q**2*tpioa**2-eee)
              if(localfieldcorrectionllw()) then
                ix=0
                do igb1=ix+1,ngb
                  do igb2=ix+1,ngb
                    if(igb1==1.and.igb2==1) then
                      epstilde(igb1,igb2)= 1d0 - vcou1*zxq(1,1,iw)
                      cycle
                    endif
                    epstilde(igb1,igb2)= -vcousq(igb1)*zxq(igb1,igb2,iw)*vcousq(igb2)
                    if(igb1==igb2) then
                      epstilde(igb1,igb2)=1d0 + epstilde(igb1,igb2)
                    endif
                  enddo
                enddo
c !TIME0
                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
                if(iq0<=nq0i) llw(iw,iq0)= 1d0/epstinv(1,1)
c !TIME1 "end of matcinv_epstinv" !this gives wrong message, probably
c         because of a bug of !TIME1 processing for MPI mode.

!! Wing elements calculation july2016
!! We need check nqb is the same as that of q=0
                if(ixyz(iq0)/=0.and.iw==0) then
                   if(ngb/=ngbq0) then
                      write(6,*)q,iq0,ngb,ngbq0
                      call rx('hx0p0_sc: ngb/=ngbq0')
                   endif
                   wmuk(2:ngb,ixyz(iq0))=epstinv(1,2:ngb)/epstinv(1,1) !this is dot(q(:)*w_mu(:,igb)). See PRB125102(2016) eq.(36)
                endif
              else
c commentout block inversion
c$$$                     sk  (1:ngb,iw,iq0)= zxq(1,1:ngb,iw)
c$$$                     sks (1:ngb,iw,iq0)= zxq(1:ngb,1,iw)
c$$$c                     sks (1:ngb,iw,iq0)= zxq(2,1:ngb,iw) !nmbas1=2 see z1stcol in x0kf_v4h.
c$$$                     vcou1 = fourpi/sum(q**2*tpioa**2) ! --> vcousq(1)**2! !fourpi/sum(q**2*tpioa**2-eee)
                if(iq0<=nq0i) llw(iw,iq0)= 1d0 - vcou1*zxq(1,1,iw)
              endif
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cmmmm
c     write(6,"('mmmw0  ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(1,2:10:3,iw),zxq(1,63:70:3,iw)
c     write(6,"('mmmw0* ',i3,10(2d13.5,2x))") iw,zxq(1,1,iw),zxq(2,2:10:3,iw),zxq(2,63:70:3,iw)
c     write(6,"('mmmmw99x ',i3,10(2d13.5,2x))") iw,fourpi/sum(q**2*tpioa**2)/llw(iw,iq0),
c     &                   w_k(2:10:3)/llw(iw,iq0),w_k(63:70:3)/llw(iw,iq0)
c     write(6,"('mmmmw99x ',i3,10(2d13.5,2x))") iw,llw(iw,iq0),
c     &                   w_ks(2:10:3)/llw(iw,iq0),w_ks(63:70:3)/llw(iw,iq0)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!TIME0_3200011 writeiqiwreal
              if(iq0<=nq0i) write(6,"('iq iw_R omg(iw) eps(wFC) eps(woLFC) ',2i5,x,10(d13.6,2x,d13.6,x,d13.6,2x,d13.6,x,d13.6))")
     &         iq,iw,freq_r(iw),llw(iw,iq0),1d0-vcou1*zxq(1,1,iw)

!TIME1_3200011 "writeiqiwreal"
            endif

c$$$                  if(.not.newaniso2) then ! Original mode
c$$$                     call rx( 'not checked here')
c$$$c     call wcf( ngb, vcoul, zxq(1,1,iw), imode, zw0)
c$$$                  endif

c$$$  !!... a debug mode
c$$$  write(6,"('hhh --- EigenValues for Im( W) --------')")
c$$$  allocate(ebb(ngb))
c$$$  call diagcvh2( (zw0-transpose(dconjg(zw0)))/2d0/img, ngb, ebb)
c$$$  do ii=1,ngb
c$$$  if( abs(ebb(ii))>1d-8 .and. ebb(ii)>0) then
c$$$  write(6, "('hhhIWq : iw ii eb=',2i4,d13.5)") iw, ii, ebb(ii)
c$$$  else
c$$$  write(6, "('hhhIWqxxx : iw ii eb=',2i4,d13.5)") iw, ii, ebb(ii)
c$$$  endif
c$$$  enddo
c$$$  deallocate(ebb)

c     if(newaniso2.and.iq>nqibz) then
c     c                zw(1:ngb,1:ngb) = 0d0
c     c                write(ifrcw, rec=((iq-iqxini)*(nw-nw_i+1)+ iw-nw_i+1 ) ) zw   !  WP = vsc-v
c     else
c     zw(1:ngb,1:ngb) = zw0
c     c                write(ifrcw, rec=((iq-iqxini)*(nw-nw_i+1)+ iw-nw_i+1 ) ) zw   !  WP = vsc-v
c     write(ifrcw, rec= iw-nw_i+1 ) zw   !  WP = vsc-v
c     call tr_chkwrite("freq_r iq iw realomg trwv=", zw, iw, frr,nblochpmx, nbloch,ngb,iq)
c     endif
 1015     continue              !iw
!TIME1_2200011 "do1015"

c     if(newaniso2) then
c     if(allocated(sk)) deallocate(sk,sks,w_k,w_ks)
c     endif
          if( allocated(zzr) ) deallocate(zzr)
        endif
!! === RealOmega end ===

!! === ImagOmega ===
!TIME0_230001 imagomega
        if (imagomega) then
          write(6,*)' goto imag omega'
          if (nspin == 1) zxqi = 2d0*zxqi ! if paramagnetic, multiply x0 by 2
          imode=1
          do 1016 iw  = 1,niw
c                  if( newaniso2 .and. iq<=nqibz ) then
            if( iq<=nqibz ) then
!!  Eqs.(37),(38) in PRB81 125102
```

```
              if(iq==1) then
                ix=1
                zw0(:,1)=0d0
                zw0(1,:)=0d0
              else
                ix=0
              endif
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  epstilde(igb1,igb2)= -vcousq(igb1)*zxqi(igb1,igb2,iw)*vcousq(igb2)
                  if(igb1==igb2) epstilde(igb1,igb2)=1+epstilde(igb1,igb2)
                enddo
              enddo
              epstinv=epstilde
              call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
              do igb1=ix+1,ngb
                do igb2=ix+1,ngb
                  zw0(igb1,igb2)= vcousq(igb1)*epstinv(igb1,igb2)*vcousq(igb2)
                  if(igb1==igb2) zw0(igb1,igb2)= zw0(igb1,igb2)-vcousq(igb1)*vcousq(igb2)
                enddo
              enddo
c$$$                     if(iq==1) write(ifepstinv) epstinv(ix+1:ngb,ix+1:ngb),iq,iw

              zw(1:ngb,1:ngb) = zw0 ! zw(nblochpmx,nblochpmx)
              if (mpi__roots) then
                write(ifrcwi, rec= iw)  zw !  WP = vsc-v
              endif
              call tr_chkwrite("freq_i iq iw imgomg trwv=",zw,iw,freq_i(iw),nblochpmx,nbloch,ngb
,iq)
            endif

c                  if( newaniso2.and.iq>nqibz) then
            if(iq>nqibz) then
!! Full inversion to calculalte eps with LFC.
              vcou1 = fourpi/sum(q**2*tpioa**2) ! --> vcousq(1)**2!  !fourpi/sum(q**2*tpioa**2-eee)
              if(localfieldcorrectionllw()) then
                ix=0
                do igb1=ix+1,ngb
                  do igb2=ix+1,ngb
                    if(igb1==1.and.igb2==1) then
                      epstilde(igb1,igb2)= 1d0 - vcou1*zxqi(1,1,iw)
                      cycle
                    endif
                    epstilde(igb1,igb2)= -vcousq(igb1)*zxqi(igb1,igb2,iw)*vcousq(igb2)
                    if(igb1==igb2) then
                      epstilde(igb1,igb2)=1d0 + epstilde(igb1,igb2)
                    endif
                  enddo
                enddo
                epstinv(ix+1:ngb,ix+1:ngb)=epstilde(ix+1:ngb,ix+1:ngb)
                call matcinv(ngb-ix,epstinv(ix+1:ngb,ix+1:ngb))
                if(iq0<=nq0i) llwi(iw,iq0)= 1d0/epstinv(1,1)
              else
c commentout block inversion
c$$$                     skI  (1:ngb,iw,iq0)= zxqi(1,1:ngb,iw)
c$$$c                      sksI (1:ngb,iw,iq0)= zxqi(2,1:ngb,iw) !nmbas1=2 see z1stcol in x0kf_v4h.
c$$$                     sksI (1:ngb,iw,iq0)= zxqi(1:ngb,1,iw) !nmbas1=2 see z1stcol in x0kf_v4h.
c$$$                     vcou1  = fourpi/sum(q**2*tpioa**2) ! test-->vcousq(1)**2
 !fourpi/sum(q**2*tpioa**2-eee)
c$$$                     vcou1sq= sqrt(vcou1)
c$$$!! llwI without LFC. LFC contribution is added in
                if(iq0<=nq0i) llwi(iw,iq0)=  1d0 -vcou1*zxqi(1,1,iw) !- vcou1sq*sum( skI(2:ngb) *
 w_ksI(2:ngb)*vcousq(2:ngb) )
              endif
              if(iq0<=nq0i) write(6,"('iq iw_img eps(wLFC) eps(noLFC)',i4,i4,2f10.4,2x,2f10.4)")
     &         iq,iw,llwi(iw,iq0),1d0-vcou1*zxqi(1,1,iw)
            endif

 1016     continue
c               if(newaniso2) then
c$$$                    if(iq==1) ifepstinv = iclose('EPS0inv') !iq==1 close write mode.
          deallocate(epstinv)
          if(allocated(epstilde)) deallocate(epstilde)
c               endif
        endif
!! === ImagOmega end ===
!TIME1_230001 "imagomega"

c     1002 continue  ! end of frequency block-loop
        if(allocated(vcoul)) deallocate(vcoul)
        if(allocated(zw0)) deallocate(zw0)
        if(allocated(zxq )) deallocate(zxq)
        if(allocated(zxqi)) deallocate(zxqi)

        if (mpi__roots) then
          ifrcwi = iclose('WVI.'//charnum5(iq))
          ifrcw  = iclose('WVR.'//charnum5(iq))
        endif
!!
 1001 continue
!TIME1_170001 "do1001"
c============end of loop over q point =================================
c=====================================================================
      call mpi__barrier()

!TIME0_24001 w0mpi
!! === Recieve llw and llwI at node 0, where q=0(iq=1) is calculated. ===
      if(mpi__size/=1) then
        do iq=nqibz+1,iqxend
          iq0 = iq - nqibz
c     write(6,*)' iq iq0 mpi_rank mpi_ranktab(iq)=',iq,iq0,MPI__rank,MPI__ranktab(iq),MPI__root,nw,nw_i,niw
          if(mpi__qranktab(iq)/=0) then !jan2012
            if(mpi__qranktab(iq) == mpi__rankq) then
c     write(6,*)' mpi_send iq from',iq,MPI__ranktab(iq)
c     write(6,*)' send llw sum=',sum(abs(llw(:,iq0))),nw,nw_i
c     do i=nw_i,nw
c     write(6,*)'sendxxx',i,llw(i,iq0)
c     enddo
c     write(6,*)' send llwI sum=',sum(abs(llwI(:,iq0))),niw
              dest=0
              if(iq0<=nq0i) then
                 call mpi__dblecomplexsendq(llw(nw_i,iq0),(nw-nw_i+1),dest)
                 call mpi__dblecomplexsendq(llwi(1,iq0),niw,dest)
              endif
              if(ixyz(iq0)/=0) then
                call mpi__dblecomplexsendq(wmuk(2:ngbq0,ixyz(iq0)),ngbq0-1,dest)
              endif
            elseif(mpi__rootq) then
c     write(6,*)' mpi_recv iq from',iq,MPI__ranktab(iq),nw,nw_i,niw
              src=mpi__qranktab(iq)
              if(iq0<=nq0i) then
                 call mpi__dblecomplexrecvq(llw(nw_i,iq0),(nw-nw_i+1),src)
                 call mpi__dblecomplexrecvq(llwi(1,iq0),niw,src)
              endif
              if(ixyz(iq0)/=0) then
                call mpi__dblecomplexrecvq(wmuk(2:ngbq0,ixyz(iq0)),ngbq0-1,src)
              endif
c     do i=nw_i,nw
c     write(6,*)'recivxxx',i,llw(i,iq0)
c     enddo
c     write(6,*)' recv llw sum=',sum(abs(llw(:,iq0))),nw,nw_i
c     write(6,*)' recv llwI sum=',sum(abs(llwI(:,iq0))),niw
            endif
          endif
        enddo
      endif
!TIME1_24001 "w0mpi"

c commentout block inversion
c$$$!! Add LFC (local field correction) to llw and llwI
c$$$         if(newaniso2 .and. MPI__rank == 0 ) then ! only on root node
c$$$            iq=1 !for q=0
c$$$            vcoudfile='Vcoud.'//charnum5(iq)
c$$$            ifvcoud = iopen(trim(vcoudfile),0,-1,0)
c$$$            read(ifvcoud) ngb0
c$$$            read(ifvcoud) qvv
c$$$            if(sum(abs(qvv))>1d-10) then
c$$$               write(6,*)'qvv =',qvv
c$$$               stop 'hx0fp0: qvv/=0 hvcc is not consistent'
c$$$            endif
c$$$            if(allocated(zcousq0)) deallocate( zcousq0,vcousq0 )
c$$$            allocate( zcousq0(ngb0,ngb0),vcousq0(ngb0))
c$$$            read(ifvcoud) vcousq0
c$$$            read(ifvcoud) zcousq0
c$$$            idummy=iclose(trim(vcoudfile))
c$$$            vcousq=sqrt(vcousq)
c$$$            allocate(epstinv(ngb0,ngb0),w_k(ngb0),w_ks(ngb0),w_kI(ngb0),w_ksI(ngb0),eemat(ngb0,ngb0))
c$$$
c$$$            do iq0=1,nq0i
c$$$              iq = iq0 + nqibz
c$$$              q = qibze(:,iq)
c$$$
c$$$              vcoudfile='Vcoud.'//charnum5(iq)
c$$$              ifvcoud = iopen(trim(vcoudfile),0,-1,0)
c$$$              read(ifvcoud) ngb
c$$$              read(ifvcoud) qvv
c$$$              if(sum(abs(qvv-q))>1d-10) then
c$$$               write(6,*)'qvv =',qvv
c$$$               stop 'hx0fp0: qvv/=0 hvcc is not consistent'
c$$$              endif
c$$$              if(allocated(zcousq)) deallocate(zcousq)
c$$$              if(allocated(vcousq)) deallocate(vcousq)
c$$$              allocate( zcousq(ngb0,ngb0),vcousq(ngb0))
```

```
c$$$              read(ifvcoud) vcousq
c$$$              read(ifvcoud) zcousq
c$$$              idummy=iclose(trim(vcoudfile))
c$$$              vcousq=sqrt(vcousq)
c$$$
c$$$              ifepstinv = iopen('EPS0inv',0,0,0)
c$$$              read(ifepstinv) ngb
c$$$
c$$$               ngc=ngb-nbloch
c$$$               if(allocated(ppovlz)) deallocate(ppovlz)
c$$$               if(allocated(ppovl)) deallocate(ppovl)
c$$$               allocate(ppovl(ngc,ngc),ppovlz(ngb,ngb))
c$$$               call readppovl0(q,ngc,ppovl) !q was qq
c$$$               ppovlz(1:nbloch,:) = zcousq(1:nbloch,:)
c$$$               ppovlz(nbloch+1:nbloch+ngc,:) = matmul(ppovl,zcousq(nbloch+1:nbloch+ngc,:))
c$$$
c$$$!  eemat: Z\mu_i(\bfk=0)^* <i|j> Z\nu_j(\bfk)
c$$$               eemat =matmul(transpose(dconjg(zcousq0)),matmul(ppovlz,zcousq))
c$$$               vcou1  = fourpi/sum(q**2*tpioa**2) ! test-->vcousq(1)**2 !fourpi/sum(q**2*tpioa**2-eee)
c$$$               vcou1sq = vcou1**.5
c$$$               write(6,*)
c$$$
c$$$              do iw=nwmin,nwmax
c$$$                read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$                epstinv(2:ngb,2:ngb) = matmul( transpose(dconjg(eemat(2:ngb,2:ngb))),
c$$$     &                            matmul(epstinv(2:ngb,2:ngb),eemat(2:ngb,2:ngb)) )
c$$$                if(iw/=iwx) then
c$$$                write(6,*)'iw iwx=',iw,iwx
c$$$                stop 'hx0fp0_sc: iw/=iwx'
c$$$                endif
c$$$                w_k(2:ngb) = vcou1sq*matmul( epstinv(2:ngb,2:ngb), sk(2:ngb,iw,iq0)*vcousq(2:ngb))
c$$$                epslfc = -vcou1sq*sum( sks(2:ngb,iw,iq0) * w_k(2:ngb) *vcousq(2:ngb) )
c$$$                llw(iw,iq0) = llw(iw,iq0)  + epslfc
c$$$                write(6,"('eps(on real) iq iw',2i4,2f9.3,2x,2f9.3)") iq0,iw, llw(iw,iq0)-epslfc,llw(iw,iq0)
c$$$              enddo
c$$$              do iw=1,niw
c$$$                read(ifepstinv) epstinv(2:ngb,2:ngb),iqx,iwx
c$$$                if(iw/=iwx) then
c$$$                 write(6,*)'iw iwx=',iw,iwx
c$$$                 stop 'hx0fp0_sc: iw/=iwx'
c$$$                endif
c$$$                w_kI(2:ngb)= vcou1sq*matmul( epstinv(2:ngb,2:ngb), skI(2:ngb,iw,iq0)*vcousq(2:ngb))
c$$$                epslfc=- vcou1sq*sum( sksI(2:ngb,iw,iq0)* w_kI(2:ngb)*vcousq(2:ngb) )
c$$$                llwI(iw,iq0)= llwI(iw,iq0)+epslfc
c$$$                write(6,"('eps(on img ) iq iw',2i4,2f9.3,2x,2f9.3)")iq0,iw, llwI(iw,iq0)-epslfc,llwI(iw,iq0)
c$$$              enddo
c$$$              ifepstinv = iclose('EPS0inv')
c$$$           enddo
c$$$         endif


!! == W(0) divergent part and W(0) non-analytic constant part.==
 1191 continue
!TIME0_40001 WVRI
c         if(newaniso2 .and. MPI__rank == 0 ) then ! MIZUHO-IR only on root node
      if(mpi__rank == 0 ) then  ! MIZUHO-IR only on root node


!! ix=1011 is a special mode to overwrite llw and llwI for test purpose
!! A file W0W0I is  generated by call w0w0i, but usually unused at anywhere.
        if(ixc==1011) then
          ifw0w0i = ifile_handle('W0W0I')
          open(ifw0w0i,form='unformatted')
          read(ifw0w0i) nw_ixx,nwxx,niw,nq0ix
          write(6,*)'w0w0i: n=',nw_ixx,nwxx,niw,nq0ix
          if(nq0i/=nq0ix)  call rx('nq0i/=nq0ix')
          if(nw_i/=nw_ixx) call rx(nw_i/=nw_ixx)
          if(nw/=nwxx) call rx(nw/=nwxx)
          read(ifw0w0i) llw(nw_i:nw,1:nq0i)
          read(ifw0w0i) llwi(1:niw,1:nq0i)
c            read(ifw0w0i) w0(nw_i:nw)
c            read(ifw0w0i) w0i(1:niw)
          close(ifw0w0i)
        endif

!! get w0 and w0i (diagonal element at Gamma point)
!! This return w0 and w0i. (llw and llwi are input)
!! Outputs w0,w0i,llmat. See use m_w0w0i at the begining of this routine.
        call w0w0i(llw,llwi,nw_i,nw,nq0i,niw,q0i)  !all inputs. get effective W0,W0i, and L(omega=0) matrix.

!! Finalize w4phonon
!! wmuk(ix)= matmul(wmu,qv) ==> wmu= matmul(wmuk,qvinv)
        if(w4pmode) then
          do i=1,3
```

```
            qv(:,i)= tpioa*q0i(:,ixyz(i))
            qv(:,i)= qv(:,i)/sqrt(sum(qv(:,i)**2))
          enddo
          call matinv(3,qv)
          allocate( wmu(2:ngbq0,3) )
          do igb=2,ngbq0
            wmu(igb,:) =matmul(wmuk(igb,:),qv)
          enddo
          ifw4p = ifile_handle()
          open(ifw4p,file='W4PHONON.HeadWing',form='unformatted')
          write(ifw4p) llmat(1:3,1:3),ngbq0 !for q~0
          write(ifw4p) wmu(2:ngbq0,1:3) !for q~0
          close(ifw4p)
cccccccccccccccccccc
c          write(6,*)'nqbq0=',ngbq0
c          write(6,*)'llmat=',llmat
c          write(6,*)'wmu sum=',sum(abs(wmu(2:ngbq0,1:3)))
c          write(6,*)'wmuksum=',sum(abs(wmuk(2:ngbq0,1:3)))
c          call rx(' test end xxxxxxxxxxxx')
ccccccccccccccccccccc
          deallocate(wmu,wmuk)
        endif

!! Read WVR and WVI at Gamma point, and give correct W(0) (averaged in the Gamma cell, where
!! Gamma cell) is the micro cell of BZ including Gamma point).

c     write(6,*)'sumcheck w0,w0i=',sum(abs(w0)),sum(abs(w0i))
!! === w0,w0i are stored to zw for q=0 ===
!! === w_ks*wk are stored to zw for iq >nqibz ===
! We assume iq=1 is for rank=0
        do iq = 1,1             !iq=1 only 4pi/k**2 /eps part only ! iq = iqxini,iqxend
c               if( .not. MPI__task(iq) ) cycle
          q = qibze(:,iq)
          do ircw=1,2
            if    (ircw==1) then
              nini=nw_i
              nend=nw
              ifrcwx = iopen('WVR.'//charnum5(iq),0,-1,mrecl)
            elseif(ircw==2) then;  nini=1;      nend=niw;
              ifrcwx = iopen('WVI.'//charnum5(iq),0,-1,mrecl)
            endif
            do iw=nini,nend
c     if(iq<=nqibz) read(ifrcwx, rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 ) ) zw !(1:ngb,1:ngb)
              read(ifrcwx, rec= iw-nini+1 ) zw !(1:ngb,1:ngb)
              if( iq==1 ) then
                if(ircw==1) zw(1,1) = w0(iw)
                if(ircw==2) zw(1,1) = w0i(iw)
              endif
c     write(ifrcwx,rec=((iq-iqxini)*(nend-nini+1)+ iw-nini+1 ) ) zw !(1:ngb,1:ngb)
              write(ifrcwx,rec=iw-nini+1) zw !(1:ngb,1:ngb)
            enddo
            if    (ircw==1) then
              ifrcwx = iclose('WVR.'//charnum5(iq))
            elseif(ircw==2) then
              ifrcwx = iclose('WVI.'//charnum5(iq))
            endif
          enddo
        end do
      endif
      is = iclose('hbe.d')
!TIME1_40001 "WVRI"
!TIME1_1001 "ProgAll"
!TIMESHOW
      call cputid(0)
      write(6,*) '--- end of hx0fp0_sc --- irank=',mpi__rank
      call flush(6)
      call mpi__finalize
      if(ixc==11) call rx0( ' OK! hx0fp0_sc ixc=11 Sergey F. mode')
      if(ixc==1011) call rx0( ' OK! hx0fp0_sc ixc=1011 W0W0Ionly')
      end program hx0fp0_sc


C===================================================================
      subroutine tr_chkwrite(tagname,zw,iw,freqq,nblochpmx,nbloch,ngb,iq)
      implicit none
      integer:: nblochpmx,nbloch,ngb,iw,i,iq
      complex(8):: zw(nblochpmx,nblochpmx),trwv,trwv2
      real(8):: freqq
      character*(*)::tagname
      trwv=0d0
      do i = 1,nbloch
        trwv = trwv + zw(i,i)
      enddo
      trwv2 = 0d0
      do i = 1,ngb
         trwv2 = trwv2 + zw(i,i)
      enddo                     !  write(6,'(" realomg trwv=",2i6,4d22.14)') iq,iw,trwv(iw),trwv2(iw)
```

```
01357       write(6,'(a,f10.4,2i5,4d22.14)')tagname,freqq,iq,iw,trwv,trwv2
01358 c     do i = 1,ngb
01359 c     write(6,'("iii i=",i4,a,f10.4,2i5,4d22.14)')i,tagname,freqq,iq,iw,zw(i,i)
01360 c     enddo
01361       end
01362
01363
```

4.37 main/qg4gw.m.F File Reference

Functions/Subroutines

- program qg4gw

4.37.1 Function/Subroutine Documentation

4.37.1.1 program qg4gw ( )

Definition at line 1 of file qg4gw.m.F.

Here is the call graph for this function:

4.38 qg4gw.m.F

```
00001       program qg4gw
00002 !> Generate required q+G vectors and so on for GW calculations.
00003 !! input file
00004 !!   LATTC: contains these lattice informations;
00005 !!    alat       : lattice constant in a.u.
00006 !!    QpGcut_psi : maxmum of |q+G| in a.u. in the expansion of the eigenfunction.
00007 !!    QpGcut_Cou : maxmum of |q+G| in a.u. in the expansion of the Coulomb matrix.
00008 !!    plat(1:3,1): 1st primitive translation vector in the unit of alat
00009 !!    plat(1:3,2): 2nd primitive translation vector
00010 !!    plat(1:3,3): 3rd primitive translation vector
00011 !!   SYMOPS file : include point group operation. See sample.
00012 !!
00013 !! outtput files:
00014 !!   QGpsi: q and G vector for the eigenfunction
00015 !!   QGcou: q and G vector for the Coulomb matrix
00016 !!   Q0P  : offset Gamma point around \Gamma points
00017 !!   EPSwklm : offset Gamma method.
00018 !! and so on.
00019 !!ccc   Qmtet: q vectors for devided-tetrahedron.
00020 !! --------------------------
00021 !! For exampl,e QGpsi is written in the following manner. See mkqg2 in mkqg.F
00022 !!     open(ifiqg, file='QGpsi',)
00023 !!      write(ifiqg ) nqnum,ngpmx,QpGcut_psi,nqbz,nqi,imx,nqibz
00024 !!      allocate( ngvecprev(-imx:imx,-imx:imx,-imx:imx) ) !inverse mapping table
00025 !!      ngvecprev=9999
00026 !!      ngveccrev=9999
00027 !!      do iq = 1, nqnum
00028 !!         q = qq(1:3,iq)
00029 !!         write (ifiqg) q, ngp, irr(iq) ! irr=1 for irreducible points
00030 !!         do ig = 1,ngp
00031 !!            nnn3 = ngvecp(1:3, ig)
00032 !!            ngvecprev( nnn3(1), nnn3(2),nnn3(3)) = ig
00033 !!         enddo
00034 !!         write (ifiqg)  ngvecp,ngvecprev !ngvecprev is added on mar2012takao
00035 !!         do ig = 1,ngc
00036 !!            nnn3 = ngvecc(1:3, ig)
00037 !!            ngveccrev( nnn3(1), nnn3(2),nnn3(3)) = ig
00038 !!         enddo
00039 !!       enddo
00040 !!     close(ifiqg)
00041 !! -----------------------------------------------------
00042 !! True q (in a.u. in Cartesian coordinate) is given by
00043 !!    q(1:3)     = 2*pi/alat * q(1:3)
00044 !! True q+G is given by
00045 !!    qplusG(1:3,igp) = 2*pi/alat * (q + matmul(qlat * ngvec(1:3,igp))), for igp=1,ngp
00046 !! ----------------------------------------------------------------------
```

```
      use m_keyvalue,only: getkeyvalue
      implicit none
      integer(4) ::n1q,n2q,n3q,ifiqg,ifiqgc,ifigw0,ngrp,ifi,i,ig,iq0pin,idummy
      real(8) ::  alat,qpgcut_psi, qpgcut_cou,dummy ,plat(3,3)
      real(8) :: volum,q0(3),qlat0(3,3),qpgx2,a1,a2,pi,unit !,QpGx1
      real(8),allocatable :: symops(:,:,:)
      character(len=150):: recrdxxx
      character(len=10) :: keyw1='unit_2pioa',keyw2
      logical ::unit2=.false. !  readgwinput,
      integer(4)::nnn(3),ret
      integer(4):: verbose,q0pchoice,wgtq0p    !,normcheck !version,
      logical:: gausssmear,keepeigen,core_orth,ldummy, lnq0iadd=.false. !keepppovl,
      integer(4):: iq0pinxxx ,ifile_handle,n1,n2,n3
      integer:: gammacellctrl=0
      pi= 4d0* atan(1d0)
      call cputid(0)
      write(6,*)' qg4gw: Generate Q0P->1; Readin Q0P->2; band mode->3; SW(chipm)->4'
      write(6,*)'        Generate Q0P->101(old offset Gamma)'
      write(6,*)'        Generate Q0P and Q0P for xyz ->201 '
      read (5,*) iq0pin
      call headver('qg4gw',iq0pin)
      write(6,*) ' mode iq0pin = ',iq0pin
      if(iq0pin==-100.or.iq0pin==1.or.iq0pin==2.or.iq0pin==3.or.iq0pin==101) then
        iq0pinxxx=iq0pin
      elseif(iq0pin==10002) then
         iq0pinxxx=2
         gammacellctrl=1 !Gammacell skip mode
      elseif(iq0pin==20002) then
         iq0pinxxx=2
         gammacellctrl=2 !Gammacell only mode
      elseif(iq0pin==4) then
         iq0pinxxx=2
      elseif(iq0pin==201) then
        iq0pinxxx=1
        lnq0iadd=.true.
      else
        call rx( 'Not allowed iq0pin')
      endif
c this is moved to gwinit.m.F march2016
c!! Generate templeta of GWinput for iq0pin=-100
c      if(iq0pin==-100) then
c         call conv2gwinput()
c         call rx0( ' OK! qg4gw mode=-100 to generate GWinput')
c      endif
      idummy=q0pchoice()
      write(6,"(' q0pchoice() = ',i4)")  q0pchoice()

      ifi=ifile_handle()
      open (ifi, file='LATTC')
      read(ifi,*) alat
      read(ifi,*) plat(1:3,1)
      read(ifi,*) plat(1:3,2)
      read(ifi,*) plat(1:3,3)
      read(ifi,*) !dummy
      close(ifi)
!! --- readin SYMOPS. point group operations. r'=matmul(symops(:,:),r) for any ig.
      ifi=ifile_handle()
      open (ifi, file='SYMOPS')
      read(ifi,*) ngrp
      write(6,*) ' SYMOPS ngrp=',ngrp
      allocate(symops(3,3,ngrp))
      do ig = 1,ngrp
        read(ifi,*)
        do i=1,3
          read(ifi,*) symops(i,1:3,ig)
        enddo
      enddo
      close(ifi)
!! --- check write
      write(6,*) ' --- primitive vectors ---'
      write(6,"(' unit(a.u.) alat  =',f13.6 )") alat
      write(6,"(' primitive_1 =',3f13.6)") plat(1:3,1)
      write(6,"(' primitive_2 =',3f13.6)") plat(1:3,2)
      write(6,"(' primitive_3 =',3f13.6)") plat(1:3,3)
      write(6,*) ' --- point group operations --- '
      do ig = 1,ngrp
        print *, ' ig=',ig
        do i=1,3
          write(6,"(3f14.6)") symops(i,1:3,ig)
        enddo
      enddo
!! --- Readin GWinput
      call getkeyvalue("GWinput", "n1n2n3", nnn,3)
      n1q=nnn(1); n2q=nnn(2); n3q = nnn(3)
      call getkeyvalue("GWinput", "QpGcut_psi",qpgx2)
      call getkeyvalue("GWinput", "QpGcut_cou",qpgcut_cou)
      call getkeyvalue("GWinput", "unit_2pioa",unit2)
      if(unit2) then
        unit = 2d0*pi/alat
        qpgx2     = qpgx2      *unit
        qpgcut_cou= qpgcut_cou *unit
      endif
      qpgcut_psi = qpgx2
      write(6,"(' --- k points for GW from GWinput =',3i3)") nnn(1:3)
      write(6,"(' ---  |k+G| < QpG(psi) QpG(Cou)=',2d13.6)") qpgcut_psi, qpgcut_cou
      ifiqg  = ifile_handle()
      open(ifiqg ,file='QGpsi',form='unformatted')
      ifiqgc = ifile_handle()
      open(ifiqgc,file='QGcou',form='unformatted')
      if(iq0pin==4) then
         qpgcut_psi=0d0
         qpgcut_cou=0d0
      endif
!!
      call mkqg2(alat,plat,symops,ngrp,nnn,iq0pinxxx,
     &     qpgcut_psi, qpgcut_cou, ifiqg, ifiqgc, gammacellctrl,lnq0iadd)
      write(6,*) ' OK! End of qg4gw '
      if(iq0pin ==1)     call rx0( ' OK! qg4gw mode=1 normal mode')
      if(iq0pin ==2)     call rx0( ' OK! qg4gw mode=2 Readin Q0P mode')
      if(iq0pin ==10002) call rx0( ' OK! qg4gw mode=10002 Readin Q0P. GammaCell skipped.')
      if(iq0pin ==20002) call rx0( ' OK! qg4gw mode=20002 Readin Q0P. GammaCell Only.')
      if(iq0pin ==3)     call rx0( ' OK! qg4gw mode=3 band-plot mode')
      if(iq0pin ==4)     call rx0( ' OK! qg4gw mode=4 Readin Q0P mode. Set ngp=ngc=0')
      end
```

4.39 Wannier/genMLWF File Reference

Variables

- if [$#-ne 3][$2!="-np"]
- then echo An example of usage
- then $echo_run echo!Perform job_band in advance exit fi source $nfpgw run_arg $echo_run echo rm f SYML BNDS ln s syml
- then mv sigm $MATERIAL sigm $MATERIAL bakup ln s f sigm sigm $MATERIAL $echo_run echo sigm is used sigm $MATERIAL is softlink to it fi else $echo_run echo Neither sigm nor sigm $MATERIAL exists
- run_arg $argin $NO_MPI $nfpgw lmfgw llmfgw00 $MATERIAL argin =1

4.39.1 Variable Documentation

4.39.1.1 run_arg $argin $MPI_SIZE $nfpgw hx0fp0 lx0_10011 argin =1

Definition at line 52 of file genMLWF.

4.39.1.2 then mv sigm $MATERIAL sigm $MATERIAL bakup ln s f sigm sigm $MATERIAL $echo_run echo sigm is used sigm $MATERIAL is softlink to it fi else $echo_run echo Neither sigm nor sigm $MATERIAL exists

Initial value:

```
==> LDA '
fi
```

```
run_arg '---' $NO_MPI   $nfpgw /lmfa  llmfa $MATERIAL # if lmfa is not yet.
run_arg '---' $MPI_SIZE $nfpgw /lmf-MPIK llmf_start $MATERIAL
rm -f ewindow.${MATERIAL}* qbyl.${MATERIAL}* eigze*.${MATERIAL}* # remove temporaly files.


argin=0
```

Definition at line 42 of file genMLWF.

4.39.1.3 then cat UUq0U ∗UUq0U rm f UUq0U ∗fi if

Definition at line 9 of file genMLWF.

4.39.1.4 then $echo_run echo ! Perform job_band in advance exit fi source $nfpgw run_arg $echo_run echo rm f SYML BNDS ln s syml

Initial value:

```
{MATERIAL} SYML
ln -s bnds.${MATERIAL} BNDS


if [ -e sigm ]
```

Definition at line 31 of file genMLWF.

4.39.1.5 then echo An example of usage

Definition at line 19 of file genMLWF.

4.40 genMLWF

```
00001 #!/bin/bash
00002 # ------------------------------------------------------------------------
00003 # generate MLWF.
00004 # NOTE: Wannier is generated before wanplot (wanplot is only to make *.xsf file for plot).
00005 #       After wanplot, we goto calculate <wan wan |W |wan wan>
00006 # For cray, set machine="cray"
00007 #-------------------------------------------------------------------------
00008 ### all input arguments are processed ###
00009 if [ $# -ne 3 ] || [ $2 != "-np" ] ; then
00010     echo "An example of usage: genMLWF cu -np 4"
00011     echo "Do job_band_* in advance to genMLWF to get superposition of Wannier band plot!"
00012     exit 101
00013 fi
00014 nfpgw='dirname $0'
00015 MATERIAL=$1
00016 MPI_SIZE=$3
00017 NO_MPI=0
00018 ### end of processing input arguments ###
00019 if [ ! -e bnds.$1 ];then
00020     $echo_run echo "!!! Perform job_band in advance!"
00021     exit
00022 fi
00023
00024 ### Read funcitons run_arg and run_arg_tee defined in a file run_arg ###
00025 source $nfpgw/run_arg
00026
00027
00028 ######## start here ##########
00029 $echo_run echo "### START genMLWF: MPI size= " $MPI_SIZE, "MATERIAL= "$MATERIAL
00030 rm -f SYML BNDS
00031 ln -s syml.${MATERIAL} SYML
00032 ln -s bnds.${MATERIAL} BNDS
00033 ## Make softlink from sigm --> simg.$MATERIAL.
00034 ## If sigm and sigm.$MATERIAL coexist, sigm.$MATERIAL is moved to sigm.$MATERIAL.backup in advance.
00035 if [ -e sigm ]; then
00036     if [ -e sigm.$MATERIAL ]; then
00037         mv sigm.$MATERIAL sigm.$MATERIAL.bakup
```

```
        ln -s -f sigm sigm.$MATERIAL
        $echo_run echo '--- sigm is used. sigm.$MATERIAL is softlink to it  ---'
    fi
else
    $echo_run echo '--- Neither sigm nor sigm.$MATERIAL exists. ==> LDA '
fi

######## lmf part ########################################
run_arg '---' $NO_MPI   $nfpgw /lmfa  llmfa $MATERIAL # if lmfa is not yet.
run_arg '---' $MPI_SIZE $nfpgw /lmf-MPIK llmf_start $MATERIAL
rm -f ewindow.${MATERIAL}* qbyl.${MATERIAL}* eigze*.${MATERIAL}* # remove temporaly files.

##### preparation of required inputs for GW (mainly prepare required eigenfuncitons) ######
argin=0; run_arg $argin $NO_MPI   $nfpgw /lmfgw      llmfgw00 $MATERIAL
argin=1; run_arg $argin $NO_MPI   $nfpgw /qg4gw      lqg4gw   #Generate requied q+G vectors.
argin=1; run_arg $argin $MPI_SIZE $nfpgw /lmfgw-MPIK llmfgw01 $MATERIAL
run_arg  '---' $NO_MPI   $nfpgw /lmf2gw     llmf2gw  #reform data for gw

##### GW related part (up to preparation of MPB) ###########
argin=0; run_arg $argin $NO_MPI   $nfpgw /rdata4gw_v2 lrdata4gw_v2
if [ -e ANFcond ];then # This is for ANFcond. Unused recently
    #  cp EVU EVD
    $echo_run echo "Not maintained recently"
    exit 10
fi
argin=1; run_arg $argin $NO_MPI $nfpgw /heftet leftet # A file EFERMI for hx0fp0
#argin=1; run_arg $argin $NO_MPI $nfpgw /hchknw lchknw # A file NW, containing nw for given QPNT (probably only for one-shot GW).
argin=0; run_arg $argin $NO_MPI $nfpgw /hbasfp0 lbas  # Product basis generation

##### maxloc start here #########################
argin=1 ;run_arg $argin $NO_MPI $nfpgw   /hmaxloc lmaxloc1     # b-vector BBVEC
argin=1 ;run_arg $argin $MPI_SIZE $nfpgw /hpsig_MPI lpsig_MPI # PSIG* =<Psi|Gaussian>.
# Gather all PSIG* into a file. (U meand UP isp=1, D means Down spin isp=2)
cat PSIGU.* >PSIGU
rm -f PSIGU.*
if [ -e PSIGD.0000 ]; then
    cat PSIGD.* >PSIGD
    rm -f PSIGD.*
fi

argin=2 ;run_arg $argin $MPI_SIZE $nfpgw /huumat_MPI luumat2   # UU (UUmatrix <u_k,i|u_k+b,j>) matrix are caltulated.
# Gather all UU*.* into a file UUU/UUD.
cat UUU.* >UUU
rm -f UUU.*
if [ -e UUD.0000 ]; then
    cat UUD.* >UUD
    rm -f UUD.*
fi
# -- Main part of Wannier (Both of Souza's and Marzari's and procedures sucessively).
argin=2; run_arg $argin $NO_MPI $nfpgw /hmaxloc lmaxloc2  #(band plot data are generated.)


############## Wannier function plot. *.xsf for Xcrysden. ############
run_arg '---' $NO_MPI $nfpgw  /wanplot lwanplot


### Here on, we calculate W (v and W-v) for Wannier.###########
# -- UUmatrix for Q0P (offset Gamma point) are required calculation v and W at the limit of q \to 0.
argin=3; run_arg $argin $MPI_SIZE $nfpgw /huumat_MPI luumat3
# Gather all UU*.* into a file UU*, PSIG* as well. (U meand UP isp=1, D means Down spin isp=2)
if [ -e UUq0U.0000 ]; then
    cat UUq0U.* > UUq0U
    rm -f UUq0U.*
fi
if [ -e UUq0D.0000 ]; then
    cat UUq0D.* > UUq0D
    rm -f UUq0D.*
fi

### pkm4crpa file mode for crpa ###
argin=10011; run_arg $argin 1 $nfpgw /hwmatK_MPI lpkm4crpa

### Main part of v, W-v for Wanniers. ##########################
argin=0;  run_arg $argin $MPI_SIZE $nfpgw /hvccfp0 lvcc        # Coulomb matrix v
argin=1;  run_arg $argin $MPI_SIZE $nfpgw /hwmatK_MPI lwmatK1  # Matrix elements of v for Wannier
argin=111; run_arg $argin $MPI_SIZE $nfpgw /hx0fp0 lx0_111     # Screened Coulomb W minus v, W-v
argin=2;  run_arg $argin $MPI_SIZE $nfpgw /hwmatK_MPI lwmatK2  # Matrix element of W-v
#$nfpgw/Cal_W.py

#### crpa
argin=10011; run_arg $argin $MPI_SIZE $nfpgw /hx0fp0 lx0_10011    # cRPA Screened Coulomb W minus v, W-v
argin=100;  run_arg $argin $MPI_SIZE $nfpgw /hwmatK_MPI lwmatK2crpa # Matrix element of W-v
#$nfpgw/Cal_W.py

```

```
$echo_run echo "OK! It’s finished well."
exit 0
```

4.41 Wannier/hmaxloc.F File Reference

Functions/Subroutines

- program hmaxloc
- subroutine chk_amnkweight (qbz, iko_ix, iko_fx, amnk, nqbz, nwf, nband, nlmto)

  read dimensions of wc,b,hb
- subroutine chk_cnkweight (qbz, iko_ix, iko_fx, cnk, nqbz, nwf, nband, nlmto)
- subroutine chk_umn (cnk, umnk, qbz, iko_ix, iko_fx, iko_i, iko_f, nwf, nqbz, nband, nlmto)

4.41.1 Function/Subroutine Documentation

4.41.1.1 subroutine chk_amnkweight (real(8), dimension(3,nqbz) *qbz*, *iko_ix*, *iko_fx*, complex(8), dimension(iko_ix:iko_fx,nwf,nqbz) *amnk*, *nqbz*, *nwf*, *nband*, *nlmto*)

read dimensions of wc,b,hb

Definition at line 1165 of file hmaxloc.F.

4.41.1.2 subroutine chk_cnkweight (real(8), dimension(3,nqbz) *qbz*, *iko_ix*, *iko_fx*, complex(8), dimension(iko_ix:iko_fx,nwf,nqbz) *cnk*, *nqbz*, *nwf*, *nband*, *nlmto*)

Definition at line 1234 of file hmaxloc.F.

Here is the caller graph for this function:

4.41.1.3 subroutine chk_umn (complex(8), dimension(iko_ix:iko_fx,nwf,nqbz) *cnk*, complex(8), dimension(nwf,nwf,nqbz) *umnk*, real(8), dimension(3,nqbz) *qbz*, *iko_ix*, *iko_fx*, integer(4), dimension(nqbz) *iko_i*, integer(4), dimension(nqbz) *iko_f*, *nwf*, *nqbz*, *nband*, *nlmto*)

Definition at line 1303 of file hmaxloc.F.

Here is the call graph for this function:

4.41.1.4 program hmaxloc ( )

Definition at line 1 of file hmaxloc.F.

Here is the call graph for this function:

4.42 hmaxloc.F

```
      program hmaxloc
c-------------------------------------------------------------
c construct maximally localized Wannier functions
c
c References
c [1] N. Marzari and D.Vanderbilt, PRB56,12847(1997)
c [2] I. Souza, N. Marzari and D.Vanderbilt, PRB65,035109(2002)
c
c mode 1:  determine parameters for <u(m,k)|u(n,k+b> (uu-matrix)
c mode 2:  main part
c    Step 1: choose Hilbert space  (Ref.[2])
c    Step 2: maximally localize Wannier functions (Ref.[1])
c    Step 3: construct effective Hamiltonian and interpolate bands (Ref.[2])
c
cm Oct 2008 Takashi Miyake, updated
cm Aug 2007 Takashi Miyake, berry connection in the Wannier gauge
c  May 2004 Takashi Miyake, from hwmat.f
c------------------------------------------------------------
      use m_readqg,only: readngmx,readqg
      use m_readeigen,only: init_readeigen,init_readeigen2,readeval
      use m_read_bzdata,only: read_bzdata,
     &     ngrp2=>ngrp,nqbz,nqibz,nqbzw,nteti,ntetf,n1,n2,n3,qbas,ginv, !qbasmc,
     &     dq_,qbz,wbz,qibz,wibz,qbzw,
     &     idtetf,ib1bz,idteti,
     &     nstar,irk,nstbz
      use m_genallcf_v3,only: genallcf_v3,
     &     nclass,natom,nspin,nl,nn,ngrp,
     &     nlmto,nlnmx, nctot,niw, !nw_input=>nw,ef
     &     alat,delta,deltaw,esmr,symgrp,clabl,iclass,!,diw,dw
     &     invg, il, in, im, nlnm,
     &     plat, pos, ecore, symgg , konf,z,
     &     spid
      use m_read_worb,only: s_read_worb, s_cal_worb,
     &  nwf, nclass_mlwf, cbas_mlwf, nbasclass_mlwf,
     &   classname_mlwf, iclassin,
     &  iphi, iphidot, nphi, nphix
      use m_keyvalue,only: getkeyvalue
      implicit none
c------------------------------------
      real(8),allocatable:: r0g(:,:), wphi(:,:)
      real(8)    :: esmr2,shtw
      integer :: iclass2
      integer(4)::
     &   ixc,iopen,ifhbed, nprecb,mrecb,mrece,nlmtot,nqbzt, nband,
     &   ibas,ibasx,ngpmx,nxx,ngcmx,nbloch,ifqpnt,ifwd,ifbb,
     &   nprecx,mrecl,nblochpmx2,nwt,niwt, nqnum,mdimx,nblochpmx,
     &   ifrcw,ifrcwi,  noccxv,maxocc2,noccx,ifvcfpout,iqall,iaf,ntq,
     &   i,j,k,nspinmx, nq,is,ip,iq,idxk,ifoutsex,iclose,nq0i,ig,
     &   mxkp,nqibzxx,ntet,nene,iqi, ix,iw,
     &   nlnx4,niwx,irot,invr,invrot,ivsum, ifoutsec,ntqx,
     &    ifmlw(2),ifmlwe(2) !,ifcphi
     &   ,ifxc(2),ifsex(2), ifphiv(2),ifphic(2),ifec,ifexsp(2),
     &   ifsecomg(2),ifexx,ifwand,ndble=8
      real(8) :: pi,tpia,vol,voltot,rs,alpha,
     & qfermi,efx,valn,efnew,edummy,efz,qm,xsex,egex,
     & zfac1,zfac2,dscdw1,dscdw2,dscdw,zfac,ef2=1d99,exx,exxq,exxelgas
      logical lqall,laf

      integer(4),allocatable :: itq(:)
      real(8),allocatable    :: q(:,:)

c takao
      integer(4),allocatable :: ngvecpb(:,:,:),!ngveccB(:,:,:),
     &  ngvecp(:,:), ngvecc(:,:),iqib(:), !,ngpn(:)ngcni(:)
     &  kount(:,:), nx(:,:),nblocha(:),lx(:) !ngveccBr(:,:,:)
      real(8),allocatable:: vxcfp(:,:,:),
     &  wqt(:), wgt0(:,:),q0i(:,:),
     &  ppbrd(:,:,:,:,:,:,:),cgr(:,:,:,:),eqt(:),
     &  ppbrdx(:,:,:,:,:,:,:),aaa(:,:), !symope(:,:,:)=symgg, ! qibz(:,:),
     &  ppb(:), eq(:), !,pdb(:),dpb(:),ddb(:)
     &  eqx(:,:,:),eqx0(:,:,:),ekc(:),coh(:,:)
     &         , rw_w(:,:,:,:,:),cw_w(:,:,:,:,:),
     &           rw_iw(:,:,:,:,:),cw_iw(:,:,:,:,:)
      complex(8),allocatable:: geigb(:,:,:,:)
c
      logical :: screen, exchange, cohtest, legas, tote
      real(8) ::  rydberg,hartree
      real(8):: qreal(3), ntot,nocctotg2,tripl!,xxx(3,3)
      real(8):: qlat(3,3)
      logical ::nocore

c space group infermation
      integer(4),allocatable :: iclasst(:), invgx(:), miat(:,:)
      real(8),allocatable    :: tiat(:,:,:),shtvg(:,:)
```

```

c
      real(8),allocatable   :: eex1(:,:,:),exsp1(:,:,:),qqex1(:,:,:,:)
      integer(4),allocatable:: nspex(:,:),ieord(:),itex1(:,:,:)
      real(8)    :: qqex(1:3), eex,exsp,eee, exwgt,deltax0
      integer(4) :: itmx,ipex,itpex,itex,nspexmx,nnex,isig,iex,ifexspx
     & ,ifexspxx ,ifefsm, nq0ix,ifemesh,nz
      character(3)  :: charnum3,sss
      character(12) :: filenameex
      logical :: exspwrite=.false.
      character*8 xt


      integer(4)::nqbze,ini,nq0it,idummy
      real(8),allocatable:: qbze(:,:)

      real(8)   :: ebmx
      integer(4):: nbmx

      real(8):: volwgt

      integer(4)::nwin, incwfin
      real(8)::efin,ddw
      integer(4),allocatable::imdim(:)
      real(8),allocatable::freqx(:),freqw(:),wwx(:),expa(:)

      logical:: gausssmear !readgwinput,
      integer(4)::ret
      character*(150):: ddd


      integer(4):: bzcase,  ngpn1,mrecg,verbose,ngcn1,nwxx
      real(8)   :: wgtq0p,quu(3)

      integer(4):: iii,isx,ivsumxxx

c for maxloc
      real(8)   :: wbb(12),wbbsum,bb(3,12),
     c             eomin,eomax,eimin,eimax,
     c             qwf0(3),dqwf0(3),qks(3),q0(3)
      complex(8),allocatable:: uumat(:,:,:,:),evecc(:,:),eveccs(:,:),
     c                         amnk(:,:,:),cnk(:,:,:),umnk(:,:,:)
      real(8),allocatable:: ku(:,:),kbu(:,:,:),eunk(:,:),eval(:),evals(:),
     c                      eks(:),rt(:,:),rt8(:,:,:),qbz0(:,:)
      integer(4):: nbb,isc,ifq0p,
     c             nox,iko_ix,iko_fx,
     c             noxs(2),iko_ixs(2),iko_fxs(2),
     c             ieo_swt,iei_swt,itin_i,itin_f,itout_i,itout_f,
     c             nbbelow,nbabove
      integer(4),allocatable:: ikbidx(:,:)
      integer(4),allocatable:: iki_i(:),iki_f(:),
     c                         ikbi_i(:,:),ikbi_f(:,:),
     c                         iko_i(:),iko_f(:),
     c                         ikbo_i(:,:),ikbo_f(:,:)
      logical :: leout,lein,lbin,lq0p,lsyml,lbnds
      logical :: debug=.false.
!
      integer(4):: nlinex,ntmp
      parameter(nlinex=100)
      integer(4)::nline,np(nlinex)
      real(8):: qi(3,nlinex),qf(3,nlinex)
c step 1
      complex(8),allocatable:: cnq0(:,:),
     c                         upu(:,:,:,:),cnk2(:,:,:),
     c                         zmn(:,:)
      complex(8):: ctmp
      real(8),allocatable:: omgik(:)
      real(8)   :: omgi,omgiold,conv1,alpha1,domgi,qtmp(3)
      integer(4):: nsc1,ndz,nin,ifhoev,ifuu0,ifpsig
c step 2
      complex(8),allocatable:: mmn(:,:,:,:),mmn0(:,:,:,:),
     c                         rmn(:,:),smn(:,:),amn(:,:),
     c                         tmn(:,:),dwmn(:,:)
      real(8),allocatable:: rn(:,:),qn(:)
      real(8)   :: omgd,omgod,omgdod,omgidod,omgdodold,domgdod,
     c             conv2,alpha2
      integer(4):: nsc2,ibb,ii,ij,ik
      logical :: lrmn,lmmn
c step 3
      complex(8),allocatable:: hrotk(:,:,:),hrotr(:,:,:),hrotkp(:,:)
     c                       , hrotkps(:,:)
      real(8):: e1,e2,rcut
      integer(4):: iband,ifbnd,iftb,ifsh,nsh,nsh1,nsh2
      logical :: lsh
      real(8),allocatable :: rws(:,:),drws(:)
      integer(4),allocatable:: irws(:)
      integer(4):: nrws,ifham
```

```
c ixc=3
      character(20)::filename
      complex(8),allocatable::  hrotrcut(:,:,:)
      integer:: ifh
      real(8):: heps ,r_v

      real(8)::qold(3)
      real(8),allocatable:: xq(:),eval1(:,:),eval2(:,:),eval3(:,:)

      integer::npin
      real(8):: qiin(3),qfin(3)

      integer(4),allocatable::
     &  m_indx(:),n_indx(:),l_indx(:),ibas_indx(:),ibasiwf(:)
      integer:: ifoc,iwf,ldim2,ixx,ifile_handle

      real(8):: enwfmax,qxx(3),eeee,enwfmaxi, ef
      integer:: inii
      logical:: leauto,leinauto

ccccccccccccccccccccccccccccccccccccccccccxxxxxxx
c      open(1107,file='xxx1')
c      open(1108,file='xxx2')
cccccccccccccccccccccccccccccccccccccccccccc

c---------------------------------------
      hartree=2d0*rydberg()

      iii=verbose()
      write(6,*)' verbose=',iii

c mode switch. --------------
      write(6,*) ' --- Choose omodes below ----------------'
      write(6,*) '  bb vectors (1) or Wannier fn. (2) or TB Hamiltonian (3)'
      write(6,*) ' --- Put number above ! -----------------'
      call readin5(ixc,nz,idummy)
      write(6,*) ' ixc=',ixc
      if(ixc<1.or.ixc>3) call rx(' --- ixc=0 --- Choose computational mode!')

c---  readin BZDATA. See gwsrc/rwbzdata.f
c--------readin data set when you call read_BZDATA ---------------
c       integer(4)::ngrp,nqbz,nqibz,nqbzw,nteti,ntetf
ccccc    ! &   ,n_index_qbz
c       integer(4):: n1,n2,n3
c       real(8):: qbas(3,3),ginv(3,3),qbasmc(3,3)
c       real(8),allocatable:: qbz(:,:),wbz(:),qibz(:,:)
c     &    ,wibz(:),qbzw(:,:)
c       integer(4),allocatable:: idtetf(:,:),ib1bz(:),idteti(:,:)
c     &    ,nstar(:),irk(:,:),nstbz(:)          !,index_qbz(:,:,:)
c-----------------------------------------------------------------
      call read_bzdata()
      write(6,*)' nqibz ngrp=',nqibz,ngrp
      write(6,*)' nqbz  =',nqbz
c      write(6,*)  qbz
c      write(6,*)' irk=',irk
c      write(6,*)' #### idtetf: ####'
c      write(6,*) idtetf

c set up work array
c      call wkinit (iwksize)
      call pshprt(60)

C--- readin GWIN and LMTO, then allocate and set datas.
c      nwin =-999    !not readin NW file
c      efin =-999d0  !not readin EFERMI
c      efin = 0d0    !readin EFERMI
c      call readefermi()
      incwfin= -1  !use 7th colmn for core at the end section of GWIN
      call genallcf_v3(incwfin) !in module m_genallcf_v3
      if(ngrp/= ngrp2) stop 'ngrp inconsistent: BZDATA and LMTO GWIN_V2'
c---  These are allocated and setted.
c      integer(4)::  nclass,natom,nspin,nl,nn,nnv,nnc, ngrp,
c     o  nlmto,nlnx,nlnxv,nlnxc,nlnmx,nlnmxv,nlnmxc, nctot,niw, !not readin nw
c      real(8) :: alat,ef, diw,dw,delta,deltaw,esmr
c      character(120):: symgrp
c      character(6),allocatable :: clabl(:)
c      integer(4),allocatable:: iclass(:)
c     &  ,nindxv(:,:),nindxc(:,:),ncwf(:,:,:) ,
c     o    invg(:), il(:,:), in(:,:), im(:,:),   ilnm(:),  nlnm(:),
c     o    ilv(:),inv(:),imv(:),  ilnmv(:), nlnmv(:),
c     o    ilc(:),inc(:),imc(:),  ilnmc(:), nlnmc(:),
c     o    nindx(:,:),konf(:,:),icore(:,:),ncore(:),
c     &    occv(:,:,:),unoccv(:,:,:)
c     &   ,occc(:,:,:),unoccc(:,:,:),
c     o    nocc(:,:,:),nunocc(:,:,:)
c      real(8), allocatable::
```

```
c     o  plat(:,:),pos(:,:),z(:),  ecore(:,:),  symgg(:,:,:) ! symgg=w(igrp),freq(:)
c-----------------------------------------------------------------------

ccccccccccccccccccccccccccccccccccccccccccccc
      do i=1,natom
       print *,'  iatom, spid= ',i,spid(i)
      enddo
ccccccccccccccccccccccccccccccccccccccccccccc




c--- Get maximums takao 18June03
      call getnemx(nbmx,ebmx,8,.true.) !8+1 th line of GWIN0

c-------------------------------------------------------------------
c      if (nclass > mxclass) stop ' hsfp0: increase mxclass'
c!!!! WE ASSUME iclass(iatom)= iatom !!!!!!!!!!!!!!!!!!!!!!!!!
      if (nclass /= natom ) stop ' hsfp0: nclass /= natom ' ! We assume nclass = natom.
      write(6,*)' hsfp0: end of genallcf2'
c
      call pshprt(30)
      pi   = 4d0*datan(1d0)
      tpia = 2d0*pi/alat

c      call dinv33(plat,1,xxx,vol)
c      call dinv33(plat,1,qlat,vol)
c      voltot = dabs(vol)*(alat**3)
      call minv33tp(plat,qlat)
      voltot = abs(alat**3*tripl(plat,plat(1,2),plat(1,3)))

      ifmlw(1) = iopen('MLWU',0,-1,0)
      ifmlwe(1)= iopen('MLWEU',0,-1,0)
      if (nspin == 2) then
         ifmlw(2) = iopen('MLWD',0,-1,0)
         ifmlwe(2)= iopen('MLWED',0,-1,0)
      endif

c>> read dimensions of wc,b,hb
      ifhbed     = iopen('hbe.d',1,0,0)
      read (ifhbed,*) nprecb,mrecb,mrece,nlmtot,nqbzt, nband,mrecg
      if (nprecb == 4) stop 'hsfp0: b,hb in single precision'

      call init_readeigen(ginv,nspin,nband,mrece) !initialization of readEigen

c --- get space group information ---------------------------------
c true class information in order to determine the space group -----------
c because the class in the generated GW file is dummy.(iclass(ibas)=ibas should be kept).
      open (102,file='CLASS')
      allocate(iclasst(natom),invgx(ngrp)
     &         ,miat(natom,ngrp),tiat(3,natom,ngrp),shtvg(3,ngrp))
      write(6,*)'  --- Readingin CLASS info ---'
      do ibas = 1,natom
        read(102,*) ibasx, iclasst(ibas)
        write(6, "(2i10)") ibasx, iclasst(ibas)
      enddo

c Get space-group transformation information. See header of mptaouof.
      call mptauof(symgg,ngrp,plat,natom,pos,iclasst
     o              ,miat,tiat,invgx,shtvg )
c        write (*,*)  'tiat=', tiat(1:3,1:natom,invr),invr

c-----------------------------------------------
      call pshprt(60)

c... Readin eigen functions
c      ifev(1)   = iopen('EVU', 0,0,mrece)
c      if (nspin==2) ifev(2) = iopen('EVD', 0,0,mrece)

! read EF from 'BNDS' if exists
      lbnds=.false.
      inquire(file='BNDS',exist=lbnds)
      if (lbnds) then
        write(*,*)'Read EF from BNDS'
        ifh=ifile_handle()
        open(ifh,file='BNDS',status='old')
        read(ifh,*)ntmp,ef
        close(ifh)
      else ! lbnds
        call rx('you have to perform job_band in advance')
      endif
c$$$c --- determine Fermi energy ef for given valn (legas case) or corresponding charge given by z and
 konf.
c$$$! When esmr is negative, esmr is geven automatically by efsimplef.
c$$$        write(*,*)'Calculate EF in efsimplef2a'
c$$$        legas = .false.
c$$$        call efsimplef2a(nspin,wibz,qibz,ginv,
c$$$     i        nband,nqibz
c$$$     i       ,konf,z,nl,natom,iclass,nclass
c$$$     i       ,valn, legas, esmr,  !!! valn is input for legas=T, output otherwise.
c$$$c
c$$$     i        qbz,nqbz !index_qbz, n_index_qbz,
c$$$     o       ,efnew)
c$$$c
c$$$c        write(6,*)' end of efsimple'
c$$$        ef = efnew
c$$$      endif ! lbnds
c- check total ele number -------
      ntot  = nocctotg2(nspin, ef,esmr, qbz,wbz, nband,nqbz) !wbz
      write(6,*)' ef    =',ef
      write(6,*)' esmr  =',esmr
      write(6,*)' valn  =',valn
      write(6,*)' ntot  =',ntot

c      ifcphi  = iopen('CPHI',0,0,mrecb)

      call init_readeigen2(mrecb,nlmto,mrecg) !initialize m_readeigen

!c QPNT data
ctm, 080222
! read QPNT from 'SYML' if exists
      lsyml=.false.
      inquire(file='SYML',exist=lsyml)
      if (lsyml) then
        write(*,*)'Read k points for bands from SYML'
        lqall      = .false.
        laf        = .false.
        open(99,file='SYML',status='old')
        nline=0
        do i = 1,nlinex
          read(99,*,err=551,end=552)npin,qiin,qfin
          if (npin==0) exit
          nline = nline+1
          np(nline)=npin
          qi(1:3,nline)=qiin
          qf(1:3,nline)=qfin
 551      continue
        enddo
 552    continue
        if (nline.eq.nlinex) call rx('hmaxloc: too many lines in SYML')
        close(99)
        nq = 0
        do i = 1,nline
           nq = nq + np(i)
        enddo ! i
        allocate(q(3,nq),xq(nq))
        iq = 0
        xq=0d0
        qold=q(:,1)
        do i = 1,nline
        do j = 0,np(i)-1
          iq = iq + 1
          q(:,iq) = qi(:,i) + (qf(:,i)-qi(:,i))*dble(j)/dble(np(i)-1)
          if(iq>1) then
            xq(iq)= xq(iq-1) + dsqrt( sum((q(:,iq)-qold)**2) )
          endif
          qold=q(:,iq)
        enddo ! j
        enddo ! i
      else ! lsyml
        write(*,*)'Read k points for bands from GWinput'
        call getkeyvalue("GWinput","<QPNT>",unit=ifqpnt,status=ret)
        write(6,*)' ifqpnt ret=',ifqpnt,ret
c
        lqall      = .false.
        laf        = .false.
        call readx(ifqpnt,10)
        read (ifqpnt,*) iqall,iaf
        if (iqall == 1) lqall = .true.
        if (iaf   == 1)   laf = .true.
        call readx(ifqpnt,100)
ctm 040622
        read (ifqpnt,*)
        read (ifqpnt,*)

        if (lqall) then !all q-points case
          nq         = nqibz
          allocate(q(3,nq))
          call dcopy(3*nqibz,qibz,1,q,1)
        else
          call readx(ifqpnt,100)
          read (ifqpnt,*) nq
          allocate(q(3,nq))
          do       k = 1,nq
          read (ifqpnt,*) i,q(1,k),q(2,k),q(3,k)
          write(6,'(i3,3f13.6)') i,q(1,k),q(2,k),q(3,k)
          enddo
        endif ! lqall
        close(ifqpnt)
        allocate(xq(nq))
        xq=0d0
      endif ! syml
c
      nspinmx = nspin
      if (laf) nspinmx =1
c------------
c input parameters specific to MAXLOC
      call s_read_worb()


      do iclass2=1,nclass_mlwf
        write(*,*)'output:',iclassin(iclass2), nwf
     & ,trim(classname_mlwf(iclass2)),cbas_mlwf(1:nbasclass_mlwf(iclass2),iclass2)
      enddo

      call s_cal_worb()

      allocate (r0g(nphix,nwf), wphi(nphix,nwf))

      r0g = 2d0
      wphi = 1d0



      call wan_input(leout,lein,lbin,ieo_swt,iei_swt,
     &    eomin,eomax,itout_i,itout_f,nbbelow,nbabove,
     &    eimin,eimax,itin_i,itin_f,
     &    nsc1,nsc2,conv1,conv2,alpha1,alpha2,rcut)
c

cskino
      r_v=rcut
      call getkeyvalue("GWinput",'wan_tbcut_rcut',heps,default=r_v)
      call getkeyvalue("GWinput",'wan_tbcut_heps',heps,default=0.0d0)
      write(*,*) 'mloc.heps ', heps
cekino


cc --- read LDA eigenvalues
      ntq = nwf
cc     ntp0=ntq
c      allocate(eqx(ntq,nq,nspin),eqx0(ntq,nq,nspin),eqt(nband))
c      do      is = 1,nspin
c      do      ip = 1,nq
cc        iq       = idxk (q(1,ip),qbze,nqbze)
cc        call rwdd1   (ifev(is), iq, nband, eqt) !direct access read b,hb and e(q,t)
c        call readeval(q(1,ip),is,eqt)
cc        write(6,*)' eqt=',eqt
c        eqx0(1:ntq,ip,is) = eqt(itq(1:ntq))
c        eqx (1:ntq,ip,is) = rydberg()*(eqt(itq(1:ntq))- ef)
c      enddo
c      enddo
c      deallocate(eqt)

c --- info
      call winfo(6,nspin,nq,ntq,is,nbloch
     &    ,0,0,nqbz,nqibz,ef,deltaw,alat,esmr)

c
      iii=ivsumxxx(irk,nqibz*ngrp)
      write(6,*) " sum of nonzero iirk=",iii, nqbz

c-------------------------------------------------------------
c debug:
c      allocate(eqt(nband))
c      do ip = 1,nqbz
c        call readeval(qbz(1,ip),1,eqt)
c        write(80,"('***',3f10.5)")qbz(:,ip)
c        do is=1,nband
c           write(80,"(i5,f12.6)")is,eqt(is)
c        enddo
c      enddo

c Rt vectors
      allocate (rt(3,nqbz),rt8(3,8,nqbz),qbz0(3,nqbz))
c      write(6,"(a,9f9.4)")'qbas=',qbas
c      write(6,"(a,9f9.4)")'plat=',plat
      call getrt(qbz,qbas,plat,n1,n2,n3,nqbz,
     o           rt,rt8,qbz0)

c b vectors
```

```
      call getbb(plat,alat,n1,n2,n3,
     o           nbb,wbb,wbbsum,bb)

c index for k and k+bb
      allocate (ku(3,nqbz),kbu(3,nbb,nqbz),ikbidx(nbb,nqbz))

      call kbbindx(qbz,ginv,bb,
     d             nqbz,nbb,
     o             ikbidx,ku,kbu)


      allocate (iko_i(nqbz),iko_f(nqbz),
     &          iki_i(nqbz),iki_f(nqbz),
     &          ikbo_i(nbb,nqbz),ikbo_f(nbb,nqbz),
     &          ikbi_i(nbb,nqbz),ikbi_f(nbb,nqbz))

!! takao list eigen -----
      enwfmax =-1d9
      enwfmaxi=1d9
      allocate(eqt(1:nband))
      do is = 1,nspin
      do iq = 1,nqbz
         qxx = qbz(:,iq)
         call readeval(qxx,is,eqt)
         ini=1
         do i=1,nband
c           write(6,*)'eqeq',eqt(i),eomin,eqt(nwf)
           if (eqt(i)>eomin) then
             inii=i
             exit
           endif
         enddo
         eeee= (eqt(nwf+inii-1)-ef)*rydberg()
         write(6,"('elist: q iq is nwfi nwfe e(nwf)= ',3f9.4,i5,i2,2i5,f10.3)") qxx,iq,is,inii,nwf+inii-1,
eeee
         if (enwfmax < eeee) enwfmax  = eeee
         if (enwfmaxi >eeee) enwfmaxi = eeee
      enddo
      enddo
      deallocate(eqt)
      write(6,"('elist max enwf enwfmaxi=',2f13.5)") enwfmax,enwfmaxi
      call getkeyvalue("GWinput","wan_out_emax_auto",leauto,default=.false.)
      if(leauto) then
         eomax= enwfmax + 1d-4
         write(6,*)
         write(6,"(' WE USE wan_out_emax_auto on ==> +1d-3 ==> eomax=',3f13.5)") eomax
      endif
      call getkeyvalue("GWinput","wan_in_emax_auto",leinauto,default=.false.)
      if(leinauto) then
         eimax= enwfmaxi + 1d-4
         write(6,*)
         write(6,"(' WE USE  wan_in_emax_auto on ==> +1d-3 ==> eimax=',3f13.5)") eimax
      endif

c      stop 'qqqqqqqqqqqqqqqqqq'

!! ixc = 1 ----------------
      if (ixc.eq.1) then
         do is = 1,nspin
         call ewindow(is,ieo_swt,iei_swt,itout_i,itout_f,itin_i,itin_f,
     i                eomin,eomax,eimin,eimax,ef,qbz,ikbidx,
     i                nbbelow,nbabove,
     d                nqbz,nbb,nband,nwf,nspin,
     o                iko_i,iko_f,iki_i,iki_f,
     o                ikbo_i,ikbo_f,ikbi_i,ikbi_f,
     o                iko_ixs(is),iko_fxs(is),noxs(is),
     o                leout,lein)
         enddo

c write bb vectors to 'BBVEC'
         call writebb(ifbb,wbb(1:nbb),bb(1:3,1:nbb),
     i                ikbidx,ku,kbu,
     i                iko_ixs,iko_fxs,noxs,
     d                nspin,nqbz,nbb)

ctm, 060923 !!!
         ifwand = iopen('wan.d',1,-1,0)
         iko_ix = iko_ixs(1)
         iko_fx = iko_fxs(1)
         if (nspin.eq.2) then
            if (iko_ixs(2).lt.iko_ix) iko_ix = iko_ixs(2)
            if (iko_fxs(2).gt.iko_fx) iko_fx = iko_fxs(2)
         endif
         write(ifwand,*)nqbz,nwf,iko_ix,iko_fx
         write(ifwand,*)nspin
         do is = 1,nspin
           write(ifwand,*)nqbz,nwf,iko_ixs(is),iko_fxs(is)
         enddo
         isx = iclose('wan.d')
         call rx0('hmaxloc: ixc=1 ok')
      endif

!! loop over spin -----------------------
      do 1000 is = 1,nspin
      write(*,*)'is =',is,'  out of',nspin
c energy window
      call ewindow(is,ieo_swt,iei_swt,itout_i,itout_f,itin_i,itin_f,
     i             eomin,eomax,eimin,eimax,ef,qbz,ikbidx,
     i             nbbelow,nbabove,
     d             nqbz,nbb,nband,nwf,nspin,
     o             iko_i,iko_f,iki_i,iki_f,
     o             ikbo_i,ikbo_f,ikbi_i,ikbi_f,
     o             iko_ix,iko_fx,nox,
     o             leout,lein)
!      call chk_ewindow(ifbb,is,nspin,nqbz,nbb,iko_ix,iko_fx)

cccccccccccccccccccccc
c      do iq=1,nqbz
c      write(6,"('iiii: iq and internal window region=',3i5)")iq,iki_i(iq),iki_f(iq)
c      enddo
cccccccccccccccccccccc

c read uu-matrix
      allocate (uumat(iko_ix:iko_fx,iko_ix:iko_fx,nbb,nqbz))
      call readuu(is,iko_ix,iko_fx,ikbidx,
     d            nqbz,nbb,
     o            uumat)
      call chkuu(is,iko_ix,iko_fx,ikbidx,uumat,
     d           nqbz,nbb)

!! step 1  -- choose Hilbert space -- determine cnk
      write(*,*)'Step 1: Hilbert space branch'
      write(6,*)' iko_ix iko_fx=',iko_ix,iko_fx
      allocate (amnk(iko_ix:iko_fx,nwf,nqbz),
     &          upu(iko_ix:iko_fx,iko_ix:iko_fx,nbb,nqbz),
     &          cnk(iko_ix:iko_fx,nwf,nqbz),
     &          cnk2(iko_ix:iko_fx,nwf,nqbz),
     &          omgik(nqbz))
! amnk appered in Eq.22 in Ref.II. <psi|Gaussian>
      call init_unkg(is,qbz,ginv,ef,lein,
     i              iko_ix,iko_fx,iko_i,iko_f,
     i              iki_i,iki_f,
     d              nwf,nband,nqbz,
     o              amnk,cnk)
!      call chk_amnkweight(qbz,iko_ix,iko_fx,amnk,
!     &     nqbz,nwf,nband,nlmto)
!      call chk_cnkweight(qbz,iko_ix,iko_fx,cnk,
!     &     nqbz,nwf,nband,nlmto)
      do isc = 1,nsc1
         do iq = 1,nqbz
            call dimz(lein,iko_i(iq),iko_f(iq),iki_i(iq),iki_f(iq),
     o                ndz,nin)
            if (nwf.gt.nin) then
              if (ndz.lt.1) call rx('ndz < 1')
c (1-2) <u_mk | P_k+b | u_nk>
               call getupu(isc,
     i                    uumat(:,:,:,iq),cnk,
     i                    lein,alpha1,iq,ikbidx(:,iq),
     i                    iko_ix,iko_fx,
     i                    iko_i(iq),iko_f(iq),
     i                    iki_i(iq),iki_f(iq),
     i                    ikbo_i(:,iq),ikbo_f(:,iq),
     i                    ikbi_i(:,iq),ikbi_f(:,iq),
     d                    nwf,nbb,nqbz,
     u                    upu(:,:,:,iq))
c (1-3) Zmn(k) > phi,eval
               allocate (zmn(ndz,ndz),evecc(ndz,ndz),eval(ndz))
               call getzmn(upu(:,:,:,iq),wbb,lein,
     i                    iko_ix,iko_fx,
     i                    iko_i(iq),iko_f(iq),
     i                    iki_i(iq),iki_f(iq),
     d                    nwf,nbb,nqbz,ndz,
     o                    zmn)

               call chk_hm(zmn,ndz)
               call diag_hm(zmn,ndz,eval,evecc)
               call new_cnk(cnk(:,:,iq),evecc,iq,
     i                     iko_ix,iko_fx,
     i                     iko_i(iq),iko_f(iq),
     i                     iki_i(iq),iki_f(iq),
     d                     nwf,ndz,
     o                     cnk2(:,:,iq))
c (1-3) w_I(k)  eq.(18)
               call chk_eval(wbb,eval,nbb,ndz)
            call get_omgik(wbb,eval,
     i                     iko_i(iq),iko_f(iq),
     i                     iki_i(iq),iki_f(iq),
     d                     nbb,nwf,ndz,
     o                     omgik(iq))
            deallocate (zmn,evecc,eval)
         else
            omgik(iq) = 0d0
            cnk2(:,:,iq) = cnk(:,:,iq)
c end if (ndz>1)
         endif
c end of iq-loop
      enddo
c (1-5) w_I(k) > Omaga_I  eq.(11)
      omgi = sum(omgik(:)*wbz(:))
c (1-6) check self-consistency
      write(*,"('#SC-loop, conv.',i5,d13.5)")isc,omgi
      if (isc.ge.2) then
         domgi = dabs((omgiold - omgi) / omgiold)
         if (domgi .lt. conv1) then
            write(*,*) 'step1: converged!'
            goto 810
         endif
      endif
c update
      omgiold = omgi
      cnk     = cnk2
c end of self-consistent loop
   enddo
   write(*,*)'step1: not converged'
 810  continue
   deallocate(upu,cnk2)

c      call chk_cnkweight(qbz,iko_ix,iko_fx,cnk,
c     &     nqbz,nwf,nband,nlmto)

!! NOTE: cnk is the final results of step 1
!!   cnk(iko_ix:iko_fx,nwf,nqbz)
!!   cnk(iko_i(iq):iko_f(iq),nwf,iq) gives nwf-dimentional space.
!!   step 1 (minimization of Omega_I)


!! === step 2 -- localize Wannier fn. ================
   write(*,*)'Step 2: Wannier fn. branch'

   allocate (mmn(nwf,nwf,nbb,nqbz),mmn0(nwf,nwf,nbb,nqbz),
  &          umnk(nwf,nwf,nqbz),
  &          rmn(nwf,nwf),amn(nwf,nwf),smn(nwf,nwf),
  &          rn(3,nwf),qn(nwf),tmn(nwf,nwf),dwmn(nwf,nwf),
  &          eunk(nwf,nqbz))

!! (2-0) construct initlal u~ from u
!! eunk(= e~) of {H~}_mn): eigenvalue within the nwf-dimentional Hilbert space
   call diag_unk(is,qbz,
  i              iko_ix,iko_fx,iko_i,iko_f,
  d              nband,nwf,nqbz,
  u              cnk,
  o              eunk)

!      call chk_cnkweight(qbz,iko_ix,iko_fx,cnk,
!     &     nqbz,nwf,nband,nlmto)
!
! check ortho-normality of u~'s
!      call chk_cnk(cnk,
!     i              iko_ix,iko_fx,iko_i,iko_f,
!     d              nband,nwf,nqbz)
!
! check: eunk vs. KS energy
!      call chk_eunk(is,qbz,eunk,ef,
!     d               nqbz,nband,nwf)

!! (2-1) initial: uumat -> M_mn(0) Eq.58 in Ref.[1]
   call init_mmn(cnk,uumat,ikbidx,
  i              iko_ix,iko_fx,iko_i,iko_f,ikbo_i,ikbo_f,
  d              nwf,nqbz,nbb,
  o              mmn0)

!! (2-2) initial U
!!    umnk= U(m,n) = ( A S^{-1/2} )_mn. See Eq.23 in Ref.II.
   call init_umnk(amnk,cnk,
  i               iko_ix,iko_fx,iko_i,iko_f,
  d               nwf,nqbz,
  o               umnk)

!      call chk_umn(cnk,umnk,qbz,
!     i              iko_ix,iko_fx,iko_i,iko_f,
!     d              nwf,nqbz,nband,nlmto)
```

```

      call updt_mmn(umnk,mmn0,ikbidx,
     d              nwf,nqbz,nbb,
     u              mmn)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c         do i=1,nbb
c            write(1106+is,"(a,i4,13f13.5)")'bbbb',i,bb(1:3,i),wbb(i)
c         enddo
c         do i=1,nqbz
c            write(1106+is,"(a,i4,13f13.5)")'www',i,wbz(i)
c         enddo
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      do isc = 1, nsc2
cccccccccccccccccccccccc
c      mmn=mmn+(0d0,1d-8)
ccccccccccccccccccccccc

c <r_n> ([1] eq.31)
         call get_rn(mmn,bb,wbb,wbz,
     d               nwf,nqbz,nbb,
     o               rn)
ccccccccccccccccccccccccccccccccccccccccccccc
c         do i=1,nwf
c            write(1106+is,"(a,3f13.5)")'rrrrrn',rn(1:3,i)
c         enddo
ccccccccccccccccccccccccccccccccccccccccccccccccc

         do iq = 1,nqbz
            dwmn = (0d0,0d0)
            do ibb = 1,nbb
cccccccccccccccccccccccccccccccccccccccccccccc
c      do i = 1,nwf
c      do j = 1,nwf
c         write(1106+is,"(a,4i5,2f13.3)")' mmmmm ',i,j,ibb,iq,mmn(i,j,ibb,iq)+(0d0,0.0001)
c      enddo
c      enddo
c      do i = 1,nwf
c      do j = 1,nwf
c         write(1106+is,"(a,4i5,2f13.3)")' nnnnnn ',i,j,ibb,iq,mmn0(i,j,ibb,iq)+(0d0,0.0001)
c      enddo
c      enddo
ccccccccccccccccccccccccccccccccccccccccccccccccccc

c (2-3) A[R] matrix
               call getrmn(mmn(:,:,ibb,iq),
     d                     nwf,
     o                     rmn)
               call getamn(rmn,
     d                     nwf,
     o                     amn)

c (2-4) S[T] matrix
               call gettmn(rn,mmn(:,:,ibb,iq),bb(:,ibb),
     d                     nwf,
     o                     qn,tmn)
               call getsmn(tmn,
     d                     nwf,
     o                     smn)

cccccccccccccccccccccccccccccccccc
c                smn=0d0
c                amn=0d0
cccccccccccccccccccccccccccccccccc

c DW(k) ([1] eq.57)
               dwmn(:,:) =  dwmn(:,:)
     &       + wbb(ibb) * (amn(:,:) - smn(:,:)) * alpha2 / wbbsum

c end of ibb-loop
            enddo

ccccccccccccccccccccccccc
c            dwmn=0d0
ccccccccccccccccccccccccc
c (2-5) DW(k) -> U(k) ([1] eq.60)
            call updt_uk(dwmn,
     d                   nwf,
     u                   umnk(:,:,iq))
c            call chk_um(umnk(:,:,iq),nwf)

ccccccccccccccccccccccccccccccccccccccccccccccccccc
c      do i = 1,nwf
```

```
c      do j = 1,nwf
c         write(1106+is,"(a,3i5,2f13.3")")' zzzzz',iq,i,j,umnk(i,j,iq)
c      enddo
c      enddo
ccccccccccccccccccccccccccccccccccccccccccccccccccc

c end of iq-loop
         enddo



c update Mmn ([1] eq.61)
         call updt_mmn(umnk,mmn0,ikbidx,
     d                 nwf,nqbz,nbb,
     u                 mmn)

c (2-6) Omeg_I, Omega_D and Omega_OD ([1] eq.34,35,36)
         call getomg(mmn,rn,bb,wbb,wbz,
     d               nwf,nqbz,nbb,
     o               omgi,omgd,omgod,omgdod,omgidod)

c check self-consistency
c         write(*,*)'#SC-loop, conv.',isc,omgdod
c         write(*,950)'Omg: I, OD, D ',omgi,omgod,omgd
         write(*,"('#SC-loop, conv.',i6,e13.5,' Omg:_I,_OD,_D= ',3f17.10)")
     &   isc,omgdod,omgi,omgod,omgd
         if (isc.ge.2) then
            domgdod = dabs((omgdodold - omgdod) / omgdodold)
            if (domgdod .lt. conv2) then
               write(*,*) 'step2: converged!'
               goto 820
            endif
         endif
         omgdodold = omgdod

c end of self-consistent loop
      enddo
      write(*,*)'step2: not converged'
 820  continue

!      call chk_dnk(is,eunk,qbz,
!     i             umnk,cnk,
!     i             iko_ix,iko_fx,iko_i,iko_f,
!     d             nband,nwf,nqbz)
!
!      call chk_umn(cnk,umnk,qbz,
!     i             iko_ix,iko_fx,iko_i,iko_f,
!     d             nwf,nqbz,nband,nlmto)

c output
      write(*,*)"---------- wlaxloc isp =",is
      call wmaxloc(ifmlw(is),ifmlwe(is),
     i            qbz,umnk,cnk,eunk,
     i            iko_ix,iko_fx,iko_i,iko_f,
     d            nwf,nqbz,nband,nlmto, is)
      call writeomg(is,mmn,rn,bb,wbb,wbz,tpia,
     d            nwf,nqbz,nbb)
c 070824
      call getkeyvalue("GWinput","wan_write_rmn",lrmn,default=.false.)
      if (lrmn)
     & call writermn(is,mmn,bb,wbb,qbz,qbz0,wbz,rt,
     d             nwf,nqbz,nbb,n1,n2,n3)
c 070830
      call getkeyvalue("GWinput","wan_write_mmn",lmmn,default=.false.)
      if (lmmn)
     & call writemmn(is,mmn,bb,wbb,qbz,wbz,rt,
     d             nwf,nqbz,nbb,n1,n2,n3)

      deallocate(uumat,amnk,omgik,mmn,mmn0,
     &           rmn,amn,smn,rn,qn,tmn,dwmn)


!! step 3 -- reduced Hamiltonian ------------------------------
      write(*,*)'Step 3: reduced Hamiltonian branch'
c open file
      if (is .eq. 1) then
         ifbnd = iopen('bnds.maxloc.up',1,-1,0)
         iftb  = iopen('bnds.tb.up',1,-1,0)
      else
         ifbnd = iopen('bnds.maxloc.dn',1,-1,0)
         iftb  = iopen('bnds.tb.dn',1,-1,0)
      endif
      write(ifbnd,*)nq
      write(ifbnd,*)nwf
      write(iftb,*)nq
      write(iftb,*)nwf
c allocate
```

```fortran
      if(allocated(hrotk)) deallocate(hrotk,hrotkp,evecc,eval)
      allocate (hrotk(nwf,nwf,nqbz), ! hrotr(nwf,nwf,nqbz),
     o          hrotkp(nwf,nwf),evecc(nwf,nwf),eval(nwf))
c for small Hamiltonian
      call getkeyvalue("GWinput","wan_small_ham",lsh,default=.false.)
      if (lsh) then
         call getkeyvalue("GWinput","wan_nsh1",nsh1, default=1 )
         call getkeyvalue("GWinput","wan_nsh2",nsh2, default=2 )
         write(*,*)'SmallHam on',nsh1,nsh2
         nsh = nsh2 - nsh1 + 1
         if (is .eq. 1) then
            ifsh = iopen('bnds.sh.up',1,-1,0)
         else
            ifsh = iopen('bnds.sh.dn',1,-1,0)
         endif
         write(ifsh,*)nq
         write(ifsh,*)nsh
         allocate (hrotkps(nsh,nsh),eveccs(nsh,nsh),evals(nsh))
      endif
c (3-1) ~H(k) -> Hrot(k): note eunk is eigenvalues in the basis of cnk
      call rot_hmnk(umnk,eunk,
     d              nwf,nqbz,
     o              hrotk) !rotated Hamiltonian in MLW basis.
c (3-2) Hrot_mn(R)
      if(allocated(irws)) deallocate(irws,rws,drws)
      allocate(irws(n1*n2*n3*8),rws(3,n1*n2*n3*8),drws(n1*n2*n3*8))
      call wigner_seitz(alat,plat,n1,n2,n3,nrws,rws,irws,drws)
      if(allocated(hrotr)) deallocate(hrotr)
      allocate(hrotr(nwf,nwf,nrws)) !real space Hamiltonian in Wannier funciton basis
c      write(*,*) 'xxxxxxxxxx1'
      if (ixc.eq.2) then
         call get_hrotr_ws(hrotk,qbz,wbz,
     i        rws,irws,drws,
     d        nwf,nqbz,nrws,
     o        hrotr)
c     skino
c     write hrotr and *rws
         if (is .eq. 1) then
            ifh = iopen('hrotr.up',1,-1,0)
         else
            ifh = iopen('hrotr.dn',1,-1,0)
         endif

         call write_hrotr(ifh, hrotr,
     i        rws,irws,drws,
     d        nwf,nrws )

         close (ifh)
c     ekino
c     skino
      else if (ixc.eq.3) then
         if (is .eq. 1) then
            filename='hrotr.up'
         else
            filename = 'hrotr.dn'
         endif
         call read_hrotr(filename,nwf,nrws,
     o        hrotr)
         if (is .eq. 1) then
            ifh = iopen('hrotr.cut.up',1,-1,0)
         else
            ifh = iopen('hrotr.cut.dn',1,-1,0)
         endif
         allocate(hrotrcut(nwf,nwf,nrws))
         call make_hrotrcut( hrotr,
     i        rws,irws,drws,
     i        rcut,heps,
     d        nwf,nrws,
     o        hrotrcut )
         call write_hrotr(ifh, hrotrcut,
     i        rws,irws,drws,
     d        nwf,nrws )
         close (ifh)
         deallocate(hrotrcut)
c     ekino
      endif
c      write(*,*) 'xxxxxxxxxx2'

!! ---------------------------------------------------------
!! k-point mesh
      call get_nqbze(nqbz,nqbze)
      allocate(qbze(3,nqbze))
      call get_qbze(qbz,nqbz,
     o              qbze,nqbze)
      write(ifmlw(is))nqbze,nwf
      write(ifmlwe(is))nqbze,nwf
      do iq = 1,nqbze
```

```
c         write(*,*)'goto get_hrotkp_ws iq=',iq,nqbze
          call get_hrotkp_ws(hrotr,rws,drws,irws,qbze(:,iq), !july2014 qbz->qbze
     d                     nwf,nqbz,nrws,
     o                     hrotkp)
          call diag_hm(hrotkp,nwf,eval,evecc)
          call wmaxloc_diag(ifmlw(is),ifmlwe(is),
     i                 iq,qbze(1:3,iq),umnk,cnk,eunk,evecc,eval,
     i                 iko_ix,iko_fx,iko_i,iko_f,
     d                 nwf,nqbz)
      enddo
c      write(6,*)'eeeeeeeee'
      deallocate(qbze)
ccc          write(*,990)'iq =',iq,qbz(1:3,iq)
cc          if (iq.le.nqbz) then
cc          do iband = 1,nwf
cc             e1 = (eval(iband)   -ef)*rydberg()
cc             e2 = (eunk(iband,iq)-ef)*rydberg()


!! --------------------------------------------------------------
c --- Readin nlam index
      ifoc = iopen('@MNLA_CPHI',1,0,0)
      ldim2 = nlmto
      read(ifoc,*)
      if(allocated(m_indx)) deallocate(m_indx,n_indx,l_indx,ibas_indx,ibasiwf)
      allocate(m_indx(ldim2),n_indx(ldim2),l_indx(ldim2),ibas_indx(ldim2))
      do ix =1,ldim2
        read(ifoc,*)m_indx(ix),n_indx(ix),l_indx(ix),ibas_indx(ix),ixx
        if(ixx/=ix) call rx('failed to readin @MNLA_CPHI')
      enddo
      ix = iclose('@MNLA_CPHI')
      allocate(ibasiwf(nwf))
      do iwf=1,nwf
        ibasiwf(iwf) = ibas_indx(iphi(1,iwf))
      enddo


!! write HrotRS
      ifh=ifile_handle()
      if(is==1) open(ifh,file='HrotRS.up',form='unformatted')
      if(is==2) open(ifh,file='HrotRS.dn',form='unformatted')
      write(ifh)alat,plat,natom
      write(ifh)pos
      write(ifh)ef
      write(ifh)nwf,nrws,n1,n2,n3
      write(ifh) irws,rws,hrotr, ibasiwf
      close(ifh)

      ifh = ifile_handle()

      call write_hopping_output(is, ifh, hrotr,
     &                  rws,irws,alat,plat,qlat,pos,natom,
     &                  ibasiwf, nwf,nrws,spid , m_indx, l_indx,
     &                  nphix, iphi, ldim2)

      close(ifh)


!! other k-points
      write(ifbnd,*)ef,' ef'
      write(iftb,*)ef,' ef'
      if (lsh) write(ifsh,*)ef,' ef'
      allocate(eval1(nwf,nq),eval3(nwf,nq))
      if(lsh) allocate(eval2(nwf,nq))
      do iq = 1,nq
c         write(6,*)' got get_hrotkp_ws iq =',iq
c (3-3) Hrot_mn(k')
          call get_hrotkp_ws(hrotr,rws,drws,irws,q(:,iq),
     d                     nwf,nqbz,nrws,
     o                     hrotkp)
c (3-4) diagonalize
          call diag_hm(hrotkp,nwf,eval,evecc)
          eval1(1:nwf,iq)=eval
c (3-4) diagonalize  -- Small Hamiltonian --
          if (lsh) then
             hrotkps(1:nsh,1:nsh) = hrotkp(nsh1:nsh2,nsh1:nsh2)
             call diag_hm(hrotkps,nsh,evals,eveccs)
             write(ifsh,*)'iq =',iq
             write(ifsh,990)q(1:3,iq)
             eval2(1:nsh,iq)= evals(1:nsh)
          endif                 ! lsh
c (3-3) Hrot_mn(k')  -- Tight-binding ---
          call get_hrotkp_tb_ws(rcut,plat,alat,
     i         hrotr,rws,drws,irws,q(:,iq),  ibasiwf,pos,natom,
     d         nwf,nqbz,nrws,
     o         hrotkp)
c     (3-4) diagonalize -- Tight-binding --
```

```
            call diag_hm(hrotkp,nwf,eval,evecc)
            eval3(1:nwf,iq)=eval
        enddo
!      ! write eval july2014takao
        do iband = 1,nwf
           do iq = 1,nq
              write(ifbnd,"(i5,3f13.5,'  ',f13.6,f13.6,i5,' !eee! x eval-ef(ev) iband' )")
     &           iq,q(1:3,iq),  xq(iq),(eval1(iband,iq)-ef)*rydberg(),iband
              write(iftb,"(i5,3f13.5,'  ',f13.6,f13.6,i5,' !eee! x eval-ef(ev) iband' )")
     &           iq,q(1:3,iq),  xq(iq),(eval3(iband,iq)-ef)*rydberg(),iband
           enddo
           write(ifbnd,*)
           write(iftb,*)
        enddo
        deallocate(eval1,eval3)

        if(lsh) then
           do iband = 1,nsh
              do iq = 1,nq
                 write(ifsh,"(i5,3f13.5,'  ',f13.6,f13.6,i5,' !eee! x eval-ef(ev) iband' )")
     &              iq,q(1:3,iq),  xq(iq),(eval2(iband,iq)-ef)*rydberg(),iband
              enddo
           enddo
        endif
        call writeham(ifham,is,ef,alat,plat,pos,qbz,wbz,rws,irws,hrotk,nspin,natom,nwf,nqbz,nrws)
        deallocate(cnk,umnk,eunk,hrotk,hrotr,hrotkp,evecc,eval,irws,rws,drws,
     &           ibasiwf,m_indx,n_indx,l_indx,ibas_indx)
        if (lsh) deallocate(hrotkps,eveccs,evals)
        close(ifbnd)
c end of loop over spin
 1000 continue
 950  format(a14,3f23.16)
 990  format(3f12.6)
      call cputid(0)
      call rx0('hmaxloc: ixc=2 ok')
      end

c-----------------------------------------------------------------------
      subroutine chk_amnkweight(qbz,iko_ix,iko_fx,amnk,
     &     nqbz,nwf,nband,nlmto)
      use m_readqg
      use m_readeigen
      implicit real*8(a-h,o-z)

      complex(8) :: amnk(iko_ix:iko_fx,nwf,nqbz)
      complex(8),allocatable:: cphi1(:,:),cphi2(:,:)
      real(8) :: qbz(3,nqbz),q(3),quu(3)
      real(8),allocatable:: wbas(:,:)
      integer(4),allocatable::
     &  m_indx(:),n_indx(:),l_indx(:),ibas_indx(:)

c --- Readin nlam index
      ifoc = iopen('@MNLA_CPHI',1,0,0)
      ldim2 = nlmto
      read(ifoc,*)
      allocate(m_indx(ldim2),n_indx(ldim2),l_indx(ldim2),ibas_indx(ldim2))
      do ix =1,ldim2
        read(ifoc,*)m_indx(ix),n_indx(ix),l_indx(ix),ibas_indx(ix),ixx
        if(ixx/=ix) call rx('failed to readin @MNLA_CPHI')
      enddo

      nbas = ibas_indx(nlmto)
      allocate(cphi1(nlmto,nband),cphi2(nlmto,nwf),wbas(nbas,nwf))
      wbas = 0d0
      cphi2=0d0
      do iq = 1,nqbz
      q = qbz(:,iq)
      call readcphi(q,nlmto,1,quu,cphi1)

       do iwf=1,nwf
         do ib=iko_ix,iko_fx
            cphi2(:,iwf) = cphi2(:,iwf) + cphi1(:,ib)*amnk(ib,iwf,iq)
         enddo
      enddo

      enddo ! iq

      do iwf=1,nwf
         do ia=1,nlmto
            ibas = ibas_indx(ia)
            wbas(ibas,iwf) = wbas(ibas,iwf) +
     &                 conjg(cphi2(ia,iwf))*cphi2(ia,iwf)
         enddo ! ia
      enddo ! iwf
      wbas = wbas / dble(nqbz**2)

      write(*,*)'*** ibas,iwf,wbas'
      do iwf=1,nwf
      do ibas=1,nbas
         write(*,*)ibas,iwf,wbas(ibas,iwf)
      enddo
      write(*,*)
      enddo
      write(*,*)'*** ibas,wbas'
      do ibas=1,nbas
         w = 0d0
         do iwf=1,nwf
            w = w + wbas(ibas,iwf)
         enddo
         write(*,*)ibas,w
      enddo

      deallocate(cphi1,cphi2,wbas,m_indx,l_indx,n_indx,ibas_indx)
      ix = iclose('@MNLA_CPHI')

      end
c-----------------------------------------------------------------------
      subroutine chk_cnkweight(qbz,iko_ix,iko_fx,cnk,
     &     nqbz,nwf,nband,nlmto)
      use m_readqg
      use m_readeigen
      implicit real*8(a-h,o-z)

      complex(8) :: cnk(iko_ix:iko_fx,nwf,nqbz)
      complex(8),allocatable:: cphi1(:,:),cphi2(:,:)
      real(8) :: qbz(3,nqbz),q(3),quu(3)
      real(8),allocatable:: wbas(:,:)
      integer(4),allocatable::
     &  m_indx(:),n_indx(:),l_indx(:),ibas_indx(:)

c --- Readin nlam index
      ifoc = iopen('@MNLA_CPHI',1,0,0)
      ldim2 = nlmto
      read(ifoc,*)
      allocate(m_indx(ldim2),n_indx(ldim2),l_indx(ldim2),ibas_indx(ldim2))
      do ix =1,ldim2
        read(ifoc,*)m_indx(ix),n_indx(ix),l_indx(ix),ibas_indx(ix),ixx
        if(ixx/=ix) call rx('failed to readin @MNLA_CPHI')
      enddo

      nbas = ibas_indx(nlmto)
      allocate(cphi1(nlmto,nband),cphi2(nlmto,nwf),wbas(nbas,nwf))
      wbas = 0d0
      cphi2=0d0

      do iq = 1,nqbz
      q = qbz(:,iq)
      call readcphi(q,nlmto,1,quu,cphi1)

      do iwf=1,nwf
         do ib=iko_ix,iko_fx
            cphi2(:,iwf) = cphi2(:,iwf) + cphi1(:,ib)*cnk(ib,iwf,iq)
         enddo
      enddo

      enddo ! iq

      do iwf=1,nwf
         do ia=1,nlmto
            ibas = ibas_indx(ia)
            wbas(ibas,iwf) = wbas(ibas,iwf) +
     &                 conjg(cphi2(ia,iwf))*cphi2(ia,iwf)
         enddo ! ia
      enddo ! iwf
      wbas = wbas / dble(nqbz*nqbz)

      write(*,*)'*** ibas,iwf,wbas'
      do iwf=1,nwf
      do ibas=1,nbas
         write(*,*)ibas,iwf,wbas(ibas,iwf)
      enddo
      write(*,*)
      enddo

      write(*,*)'*** ibas,wbas'
      do ibas=1,nbas
         w = 0d0
         do iwf=1,nwf
            w = w + wbas(ibas,iwf)
         enddo
         write(*,*)ibas,w
      enddo
      deallocate(cphi1,cphi2,wbas,m_indx,l_indx,n_indx,ibas_indx)
      ix = iclose('@MNLA_CPHI')
      end
c-----------------------------------------------------------------------
      subroutine chk_umn(cnk,umnk,qbz,
     i                  iko_ix,iko_fx,iko_i,iko_f,
     d                  nwf,nqbz,nband,nlmto)
      use m_readqg
      use m_readeigen

      implicit real*8(a-h,o-z)

      complex(8) :: cnk(iko_ix:iko_fx,nwf,nqbz),
     &              dnk(iko_ix:iko_fx,nwf,nqbz),
     &              umnk(nwf,nwf,nqbz)
      real(8) :: qbz(3,nqbz)
      integer(4) :: iko_i(nqbz),iko_f(nqbz)

      dnk = (0d0,0d0)
      do iq = 1,nqbz
         do imp = iko_i(iq),iko_f(iq)
         do in = 1,nwf
            do im = 1,nwf
               dnk(imp,in,iq) = dnk(imp,in,iq)
     &                 + umnk(im,in,iq) * cnk(imp,im,iq)
            enddo ! im
         enddo ! in
         enddo ! imp
      enddo ! iq

      call chk_cnkweight(qbz,iko_ix,iko_fx,dnk,
     &     nqbz,nwf,nband,nlmto)

      end
```

4.43 /home/takao/ecalj/lm7K/run_arg File Reference

4.44 run_arg

```
### bash subroutine used in gwsc and so on. ###
### See fpgw/exec/gwsc for usage
# T.Kotani Jan.2015
# SeungWoo Jang Sep.2014
echo_run=""                    # standard
serial_run=""                  # standard
#echo_run="aprun"                                      # cray
#serial_run="aprun"                                    # cray
function run_arg
{
    local argin=$1
    local MPI_SIZE=$2
    local nfpgw=$3
    local command=$4
    local output=$5
    local TARGET=${@:6:($#-2)}
    local mpi_run="mpirun -np $MPI_SIZE"                          # standard
    #local pi_run="aprun -n $LSB_PROCS -d $LSB_CPUS -N $LSB_PPN"  # cray
    $echo_run echo -n 'OK! --> Start'
    $echo_run echo $argin > _IN_
    if [ $MPI_SIZE == '0' ]; then
        $echo_run echo " echo $argin | $nfpgw$command $TARGET > $output "
        $serial_run $nfpgw$command  $TARGET < _IN_ > $output
    else
        $echo_run echo " echo $argin | mpirun -np $MPI_SIZE $nfpgw$command $TARGET > $output "
        $mpi_run $nfpgw$command $TARGET < _IN_ > $output
    fi
    if [ $? != 0 ]; then
        $echo_run echo Error in $command input_arg=$argin. See OutputFile=$output
        exit 10
    fi
}

echo "NOTE: Use run_arg defined in $nfpgw/run_arg"
```