progress

1.1.0

Generated by Doxygen 1.8.17

Chapter 1

Testing the Progress library

1.1 Testing program for the progress library

1.1.1 To run the tests:

Go into the build folder and type:

```
  make test
```

To run the tests in verbose mode:

```
  make test ARGS="-V"
```

1.1.2 To run a single test:

To run a test on its own (in build) we just need to type:

```
/qmd-progress/build/main  <test_name>
```

, where "test_name" is the name of the test we want to run. Right now the keywords (test_name) we can pass are the following:

- density : Tests the diagonalization routine to build the density.
- sp2_short : Tests the first version of sp2
- sp2_alg1 : Algorithm 1 for sp2
- sp2_alg2 : Algorithm 2 for sp2
- sp2_alg2_ellpack : Algorithm 2 for sp2 with ellpack
- sp2_alg1_seq : See sp2_mod.F90 source file
- sp2_alg2_seq : See sp2_mod.F90 source file
- deorthogonalize_dense: See nonortho.F90 source file
- orthogonalize_dense: See nonortho.F90 source file
- buildzdiag: See genz_mod.F90 source file

1.1.3 To add a test:

- add the corresponding name of the test in /progress/tests/CMakeLists.txt
- add the corresponding keyword and test in /progress/tests/src/main.F90
- Copy any file that is necessary to run (data) in /progress/tests/tests_data/
- reconfigure and recompile

Chapter 2

Todo List

Module **prg_dos_mod**

Add LDOS.

Subprogram **prg_pulaycomponent_mod::prg_pulaycomponent0** **(rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)**

M and bml_type will have to be removed from the input parameter.

Subprogram **prg_pulaycomponent_mod::prg_pulaycomponentt** **(rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)**

M and bml_type will have to be removed from the input parameter.

Module **prg_pulaymixer_mod**

add the density matrix mixer.

Module **prg_response_mod**

Add the response scf

Change name response_SP2 to dm_prt_response

Change name response_rs to rs_prt_response

Subprogram **prg_response_mod::prg_pert_from_file** **(prt_bml, norb)**

Add read perturbation from file

Subprogram **prg_system_mod::prg_parse_system** **(system, filename, extin)**

Integrate this loop in the loop for building the split.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Modules Index

4.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 5

Data Type Index

5.1 Data Types List

Here are the data types with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Module Documentation

7.1 (LATTE related routines)

7.2 (PROGRESS related routines)

Modules

- module prg_charges_mod

  A module to compute the Mulliken charges of a chemical system.
- module prg_chebyshev_mod

  Module to obtain the density matrix by applying a Chebyshev polynomial expansion.
- module prg_densitymatrix_mod

  Module to obtain the density matrix by diagonalizing an orthogonalized Hamiltonian.
- module prg_extras_mod

  Extra routines.
- module prg_genz_mod

  To produce a matrix Z which is needed to orthogonalize H.
- module prg_graph_mod

  The graph module.
- module prg_graphsolver_mod

  Module for graph-based solvers.
- module prg_graphsp2parser_mod

  Graph partitioning SP2 parser.
- module prg_homolumo_mod

  The homolumo module.
- module prg_initmatrices_mod

  Initialization module.
- module prg_kernelparser_mod

  Some general parsing functions.
- module prg_nonortho_mod

  Module to prg_orthogonalize and prg_deorthogonalize any operator.
- module prg_openfiles_mod

  Module to handle input output files for the PROGRESS lib.
- module prg_parallel_mod

  The parallel module.
- module prg_progress_mod

  The progress module.
- module prg_ptable_mod

  Periodic table of elements.
- module prg_pulaycomponent_mod

  Produces a matrix to get the Pulay Component of the forces.
- module prg_pulaymixer_mod

  Pulay mixer mode.
- module prg_quantumdynamics_mod

  A module to add in common quantum dynamical operations.
- module prg_response_mod

  Module to compute the density matrix response and related quantities.
- module prg_sp2_fermi_mod

  The SP2 Fermi module.
- module prg_sp2_mod

  The SP2 module.
- module prg_sp2parser_mod

  SP2 parser.

- module prg_syrotation_mod

  A module to rotate the coordinates of a sybsystem in chemical systems.
- module prg_system_mod

  A module to read and handle chemical systems.
- module prg_timer_mod

  The timer module.
- module prg_xlbo_mod

  A module to perform XLBO integration.
- module prg_xlbokernel_mod

  Pre-conditioned O(N) calculation of the kernel for XL-BOMD.
- module prg_xlkernel_mod

  Add name.

7.2.1 Detailed Description

7.3 (EXTERNAL related routines)

7.4 (High-level codes using PROGRESS/LATTE modules)

Chapter 8

Module Documentation

8.1 prg_charges_mod Module Reference

A module to compute the Mulliken charges of a chemical system.

Functions/Subroutines

- subroutine, public prg_get_charges (rho_bml, over_bml, hindex, charges, numel, spindex, mdimin, threshold)

  Constructs the charges from the density matrix.
- subroutine, public prg_get_hscf (ham0_bml, over_bml, ham_bml, spindex, hindex, hubbardu, charges, coulomb_pot_r, coulomb_pot_k, mdimin, threshold)

  Constructs the SCF Hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i;$, where U is the Hubbard parameter for every atom i. V is the coulombic potential for every atom i.

Variables

- integer, parameter dp = kind(1.0d0)

8.1.1 Detailed Description

A module to compute the Mulliken charges of a chemical system.

This module contains routines that compute properties related to charges.

8.1.2 Function/Subroutine Documentation

8.1.2.1 prg_get_charges()

```
subroutine, public prg_charges_mod::prg_get_charges (
            type(bml_matrix_t), intent(inout) rho_bml,
            type(bml_matrix_t), intent(inout) over_bml,
            integer, dimension(:,:), intent(in) hindex,
            real(dp), dimension(:), intent(inout), allocatable charges,
            real(dp), dimension(:), intent(in) numel,
            integer, dimension(:), intent(in) spindex,
            integer, intent(in) mdimin,
            real(dp), intent(in) threshold )
```

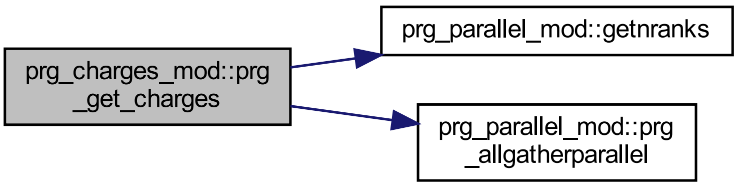
Constructs the charges from the density matrix.

Parameters

rho_bml	Density matrix in bml format.
over_bml	Overlap matrix in bml format.
hindex	Start and end index for every atom in the system.
charges	Output parameter that gives the vectorized charges.
threshold	Threshold value for matrix elements.

Definition at line 31 of file prg_charges_mod.F90.

Here is the call graph for this function:


prg_charges_mod::prg_get_charges
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel

8.1.2.2 prg_get_hscf()

```
subroutine, public prg_charges_mod::prg_get_hscf (
             type(bml_matrix_t), intent(in) ham0_bml,
             type(bml_matrix_t), intent(in) over_bml,
             type(bml_matrix_t), intent(inout) ham_bml,
             integer, dimension(:), intent(in) spindex,
             integer, dimension(:,:), intent(in) hindex,
             real(dp), dimension(:), intent(in) hubbardu,
             real(dp), dimension(:), intent(in) charges,
             real(dp), dimension(:), intent(in) coulomb_pot_r,
             real(dp), dimension(:), intent(in) coulomb_pot_k,
             integer, intent(in) mdimin,
             real(dp), intent(in) threshold )
```

Constructs the SCF Hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i;$, where U is the Hubbard parameter for every atom i. V is the coulombic potential for every atom i.

Parameters

ham_bml	Hamiltonian in bml format.
over_bml	Overlap in bml format.
hindex	Start and end index for every atom in the system.
hubbardu	Hubbard parameter for every atom.
charges	Charges for every atom.

Parameters

coulomb_pot↩_r	Coulombic potential (r contribution)
coulomb_pot↩_k	Coulombic potential (k contribution)
mdim	Maximum nonzeroes elements per row for every row.
threshold	Threshold value for matrix elements.

Definition at line 100 of file prg_charges_mod.F90.

8.1.3 Variable Documentation

8.1.3.1 dp

```
integer, parameter prg_charges_mod::dp = kind(1.0d0)  [private]
```

Definition at line 17 of file prg_charges_mod.F90.

8.2 prg_chebyshev_mod Module Reference

Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

Data Types

- type chebdata_type

  General Cheb solver type.

Functions/Subroutines

- subroutine, public prg_parse_cheb (chebdata, filename)

  Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:
- subroutine, public prg_build_density_cheb (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, jon, verbose)

  Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.
- subroutine, public prg_build_density_cheb_fermi (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, getef, fermitol, jon, npts, trkfunc, verbose)

  Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.
- real(dp) function jackson (ncoeffs, i, jon)

  Evaluates the Jackson Kernel Coefficients.
- subroutine prg_get_chebcoeffs (npts, kbt, ef, ncoeffs, coeffs, emin, emax)

  Gets the coefficients of the Chebyshev expansion.

- subroutine prg_get_chebcoeffs_fermi_bs (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)

  Gets the coefficients of the Chebyshev expansion with Ef computation.
- subroutine prg_get_chebcoeffs_fermi_nt (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)

  Gets the coefficients of the Chebyshev expansion with Ef computation.
- real(dp) function tr (r, x)

  Chebyshev polynomial obtained by recursion.
- real(dp) function fermi (e, ef, kbt)

  Gives the Fermi distribution value for energy e.
- real(dp) function absmaxderivative (func, de)

  Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter dp = kind(1.0d0)
- real(dp), parameter pi = 3.14159265358979323846264338327950_dp

8.2.1 Detailed Description

Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

See Amparo Gil 2007 **[Amparo2007]** , See Silver et al **[Silver1996]** , See Weisse et al **[Weisse2006]**

8.2.2 Function/Subroutine Documentation

8.2.2.1 absmaxderivative()

```
real(dp) function prg_chebyshev_mod::absmaxderivative (
             real(dp), dimension(:), intent(in) func,
             real(dp), intent(in) de ) [private]
```

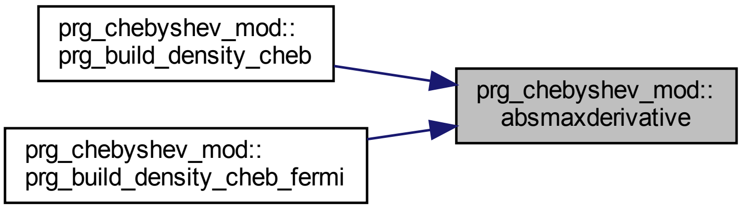
Gets the absolute maximum of the derivative of a function.

Parameters

func.	
de	Energy step.

Definition at line 802 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



8.2.2.2 fermi()

```
real(dp) function prg_chebyshev_mod::fermi (
            real(dp), intent(in) e,
            real(dp), intent(in) ef,
            real(dp), intent(in) kbt )  [private]
```

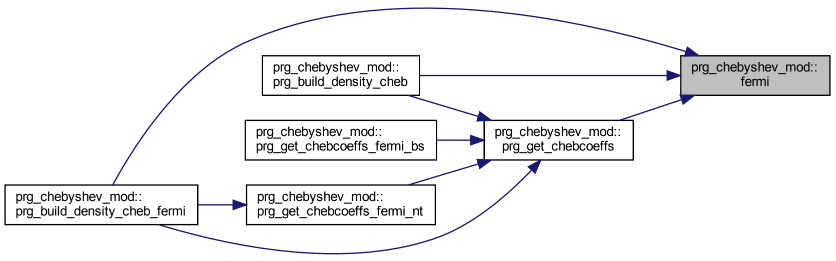
Gives the Fermi distribution value for energy e.

Parameters

e	Energy.
ef	Fermi energy.

Definition at line 790 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



8.2.2.3 jackson()

```
real(dp) function prg_chebyshev_mod::jackson (
            integer, intent(in) ncoeffs,
            integer, intent(in) i,
            logical, intent(in) jon )  [private]
```

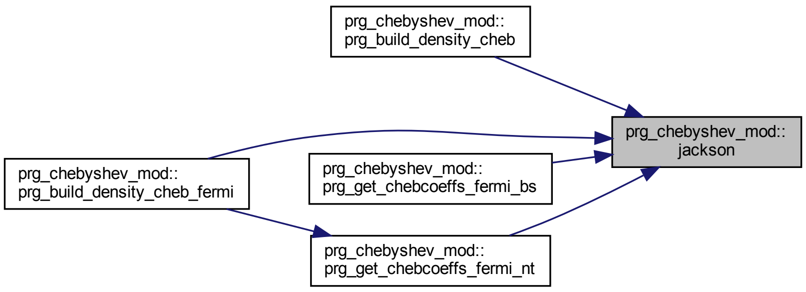
Evaluates the Jackson Kernel Coefficients.

Parameters

ncoeffs	Number of Chebyshev polynomial.
i	Coefficient number i.

Definition at line 532 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



8.2.2.4 prg_build_density_cheb()

```
subroutine, public prg_chebyshev_mod::prg_build_density_cheb (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) athr,
            real(dp), intent(in) threshold,
            integer, intent(in) ncoeffs,
            real(dp), intent(in) kbt,
            real(dp), intent(in) ef,
            real(dp), intent(in) bndfil,
            logical, intent(in) jon,
            integer, intent(in) verbose )
```

Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.

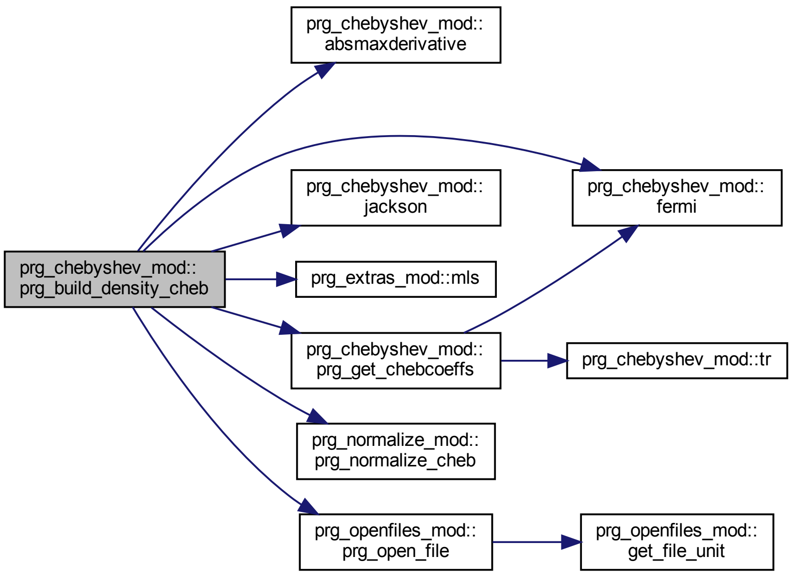
$\rho_{n+1} = b_{n+1}T_{n+1} + \rho_n$ Where, T_n is the nth Chebyshev polynomial and b_n is the nth coefficient of the expansion for the Fermi function. In the sparse version (when ellpack is used) the threshold can be varied linearly with the polynomial degree. The function is the following: $Thresh(n) = Thresh_0[a_{thr}(n-1) + (1 - a_{thr})]$

Parameters

ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix.
athr	Threshold linear increasing constant.
threshold	Threshold for sparse matrix algebra.
ncoeffs	Number of Chebyshev coefficients.
kbt	Electronic temperature in the energy units of the Hamiltonian.
ef	Fermi level in the energy units of the Hamiltonian.
bndfil	Band filing factor.
verbose	Verbosity level.

Definition at line 143 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:


prg_chebyshev_mod:: absmaxderivative
prg_chebyshev_mod:: jackson
prg_chebyshev_mod:: fermi
prg_chebyshev_mod:: prg_build_density_cheb
prg_extras_mod::mls
prg_chebyshev_mod:: prg_get_chebcoeffs
prg_chebyshev_mod::tr
prg_normalize_mod:: prg_normalize_cheb
prg_openfiles_mod:: prg_open_file
prg_openfiles_mod:: get_file_unit

8.2.2.5 prg_build_density_cheb_fermi()

```
subroutine, public prg_chebyshev_mod::prg_build_density_cheb_fermi (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) athr,
            real(dp), intent(in) threshold,
            integer, intent(in) ncoeffs,
            real(dp), intent(in) kbt,
            real(dp), intent(out) ef,
            real(dp), intent(in) bndfil,
            logical, intent(in) getef,
            real(dp) fermitol,
            logical, intent(in) jon,
            integer npts,
            logical, intent(in) trkfunc,
            integer, intent(in) verbose )
```

Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.

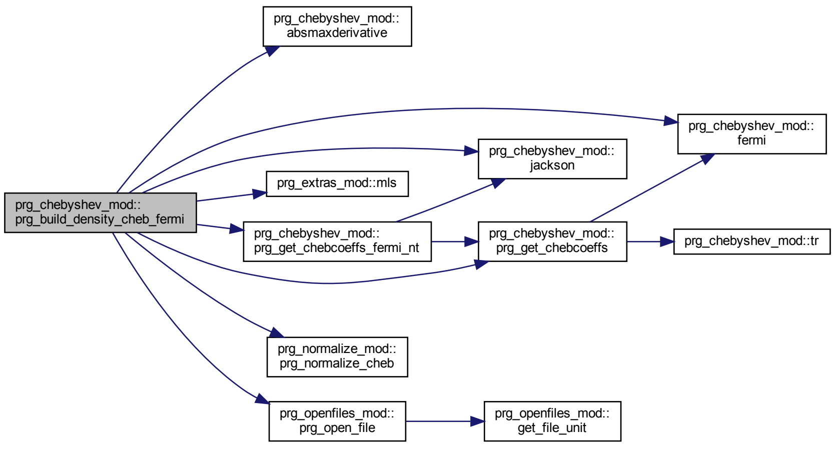
$\rho_{n+1} = b_{n+1}T_{n+1} + \rho_n$ Where, T_n is the nth Chebyshev polynomial and b_n is the nth coefficient of the expansion for the Fermi function. In the sparse version (when ellpack is used) the threshold can be varied linearly with the polynomial degree. The function is the following: $Thresh(n) = Thresh_0[a_{thr}(n-1) + (1 - a_{thr})]$

Parameters

ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix.
athr	Threshold linear increasing constant.
threshold	Threshold for sparse matrix algebra.
ncoeffs	Number of Chebyshev coefficients.
kbt	Electronic temperature in the energy units of the Hamiltonian.
ef	Fermi level in the energy units of the Hamiltonian.
bndfil	Band filing factor.
npts	Number of energy points to compute the coefficients
verbose	Verbosity level.

Definition at line 309 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:


prg_chebyshev_mod:: absmaxderivative
prg_chebyshev_mod:: fermi
prg_chebyshev_mod:: jackson
prg_extras_mod::mls
prg_chebyshev_mod:: prg_build_density_cheb_fermi
prg_chebyshev_mod:: prg_get_chebcoeffs_fermi_nt
prg_chebyshev_mod:: prg_get_chebcoeffs
prg_chebyshev_mod::tr
prg_normalize_mod:: prg_normalize_cheb
prg_openfiles_mod:: prg_open_file
prg_openfiles_mod:: get_file_unit

8.2.2.6 prg_get_chebcoeffs()

```
subroutine prg_chebyshev_mod::prg_get_chebcoeffs (
             integer, intent(in) npts,
             real(dp), intent(in) kbt,
             real(dp), intent(in) ef,
             integer, intent(in) ncoeffs,
             real(dp), dimension(:), intent(inout) coeffs,
             real(dp), intent(in) emin,
             real(dp), intent(in) emax )  [private]
```

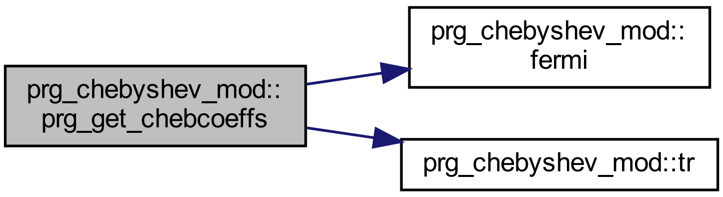
Gets the coefficients of the Chebyshev expansion.

Parameters

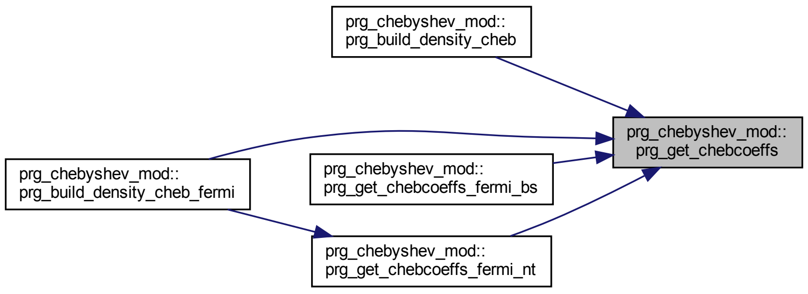
npts	Number of points for discretization.
kbt	Electronic temperature.
ef	Fermi level.
ncoeffs	Number of Chebyshev coefficients.
coeffs	Output vector for the Chebyshev coefficients.
emin	lowest boundary for the eigenvalues of H.
emax	highest boundary for the eigenvalues of H.

Definition at line 568 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.2.7 prg_get_chebcoeffs_fermi_bs()

```
subroutine prg_chebyshev_mod::prg_get_chebcoeffs_fermi_bs (
             integer, intent(in) npts,
             real(dp), intent(in) kbt,
             real(dp), intent(inout) ef,
             real(dp), dimension(:), intent(in) tracesT,
             integer, intent(in) ncoeffs,
             real(dp), dimension(:), intent(inout) coeffs,
             real(dp), intent(in) emin,
             real(dp), intent(in) emax,
             real(dp), intent(in) bndfil,
             integer, intent(in) norb,
             real(dp), intent(in) tol,
             logical, intent(in) jon,
             integer, intent(in) verbose )  [private]
```

Gets the coefficients of the Chebyshev expansion with Ef computation.

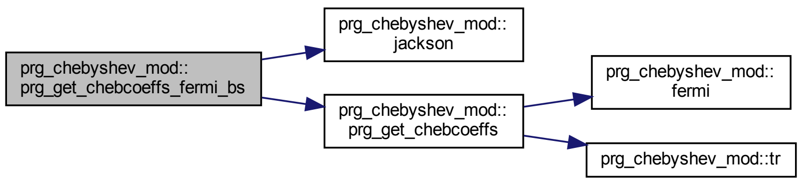
In this case we are applying the bisection method to find the root.

Parameters

npts	Number of points for the discretization.
kbt	Electronic temperature.
ef	Fermi level.
tracesT	Input traces for matrix polynomials.
ncoeffs	Number of Chebyshev coefficients.
coeffs	Output vector for the Chebyshev coefficients.
emin	lowest boundary for the eigenvalues of H.
emax	highest boundary for the eigenvalues of H.
tol	Tolerance for the bisection method.
verbose	Verbosity level.

Definition at line 620 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



8.2.2.8 prg_get_chebcoeffs_fermi_nt()

```
subroutine prg_chebyshev_mod::prg_get_chebcoeffs_fermi_nt (
             integer, intent(in) npts,
             real(dp), intent(in) kbt,
             real(dp), intent(inout) ef,
             real(dp), dimension(:), intent(in) tracesT,
             integer, intent(in) ncoeffs,
             real(dp), dimension(:), intent(inout) coeffs,
             real(dp), intent(in) emin,
             real(dp), intent(in) emax,
             real(dp), intent(in) bndfil,
             integer, intent(in) norb,
             real(dp), intent(in) tol,
             logical, intent(in) jon,
             integer, intent(in) verbose )  [private]
```

Gets the coefficients of the Chebyshev expansion with Ef computation.

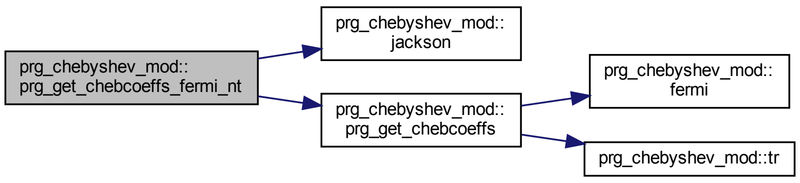
In this case the Newton-Raphson method is applied to find the root.

Parameters

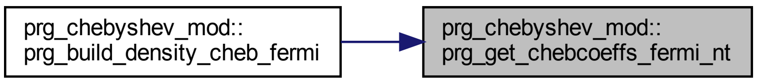
npst	Number of points for the discretization.
kbt	Electronic temperature.
ef	Fermi level.
tracesT	Input traces for matrix polynomials.
ncoeffs	Number of Chebyshev coefficients.
coeffs	Output vector for the Chebyshev coefficients.
emin	lowest boundary for the eigenvalues of H.
emax	highest boundary for the eigenvalues of H.
bndfil	Band filing factor.
norb	Number of orbitals.
tol	Tolerance for NR method.
verbose	Verbosity level.

Definition at line 697 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.2.9 prg_parse_cheb()

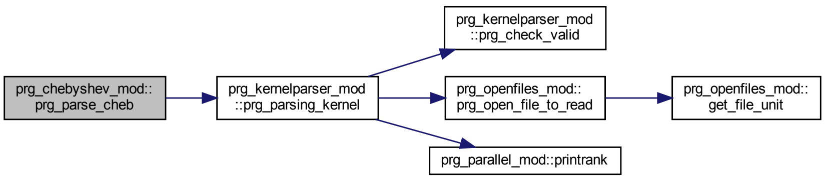
```
subroutine, public prg_chebyshev_mod::prg_parse_cheb (
              type(chebdata_type), intent(inout) chebdata,
              character(len=*) filename )
```

Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.
- Add the keyword (character type) in the keyvector_re vector.
- Add a default value (real type) in the valvector_re.
- Define a new variable and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.

Definition at line 54 of file prg_chebyshev_mod.F90.

Here is the call graph for this function:



8.2.2.10 tr()

```
real(dp) function prg_chebyshev_mod::tr (
            integer, intent(in) r,
            real(dp), intent(in) x )  [private]
```

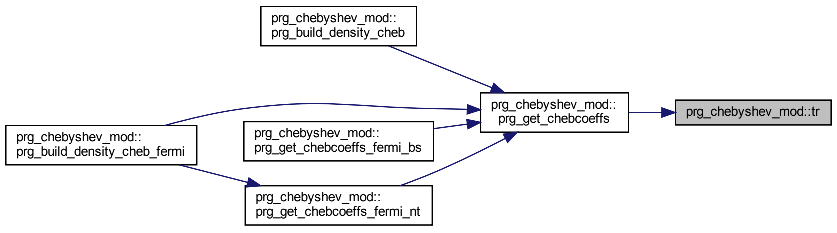
Chebyshev polynomial obtained by recursion.

Parameters

r	rth polynomial.
x	argument the evaluate the polynomial.

Definition at line 777 of file prg_chebyshev_mod.F90.

Here is the caller graph for this function:



8.2.3 Variable Documentation

8.2.3.1 dp

```
integer, parameter prg_chebyshev_mod::dp = kind(1.0d0)  [private]
```

Definition at line 23 of file prg_chebyshev_mod.F90.

8.2.3.2 pi

```
real(dp), parameter prg_chebyshev_mod::pi = 3.14159265358979323846264338327950_dp  [private]
```

Definition at line 24 of file prg_chebyshev_mod.F90.

8.3 prg_densitymatrix_mod Module Reference

Module to obtain the density matrix by diagonalizing an orthogonalized Hamiltonian.

Functions/Subroutines

- subroutine, public prg_build_density_t0 (ham_bml, rho_bml, threshold, bndfil, eigenvalues_out)

  Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.
- subroutine, public prg_build_density_t (ham_bml, rho_bml, threshold, bndfil, kbt, ef, eigenvalues_out)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public prg_build_density_t_fulldata (ham_bml, rho_bml, threshold, bndfil, kbt, ef, eigenvalues_↩ out, evects_bml, fvals)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public prg_build_density_t_fermi (ham_bml, rho_bml, threshold, kbt, ef, verbose)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.
- subroutine, public prg_build_atomic_density (rhoat_bml, numel, hindex, spindex, norb, bml_type)

  Builds the atomic density matrix. $\rho_{ii} = mathcalZ_{ii}$ Where, $mathcalZ_{ii}$ is the number of electrons for orbital i.
- subroutine, public prg_get_flevel (eigenvalues, kbt, bndfil, tol, Ef, err)

  Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.
- subroutine, public prg_get_flevel_nt (eigenvalues, kbt, bndfil, tol, ef, err, verbose)

  Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Newton-Raphson method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.
- subroutine, public prg_get_eigenvalues (ham_bml, eigenvalues, verbose)

  Gets the eigenvalues of the Orthogonalized Hamiltonian.
- subroutine, public prg_check_idempotency (mat_bml, threshold, idempotency)

  To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.
- real(dp) function fermi (e, ef, kbt)

  Gives the Fermi distribution value for energy e.

Variables

- integer, parameter dp = kind(1.0d0)

8.3.1 Detailed Description

Module to obtain the density matrix by diagonalizing an orthogonalized Hamiltonian.

8.3.2 Function/Subroutine Documentation

8.3.2.1 fermi()

```
real(dp) function prg_densitymatrix_mod::fermi (
            real(dp), intent(in) e,
            real(dp), intent(in) ef,
            real(dp), intent(in) kbt )  [private]
```

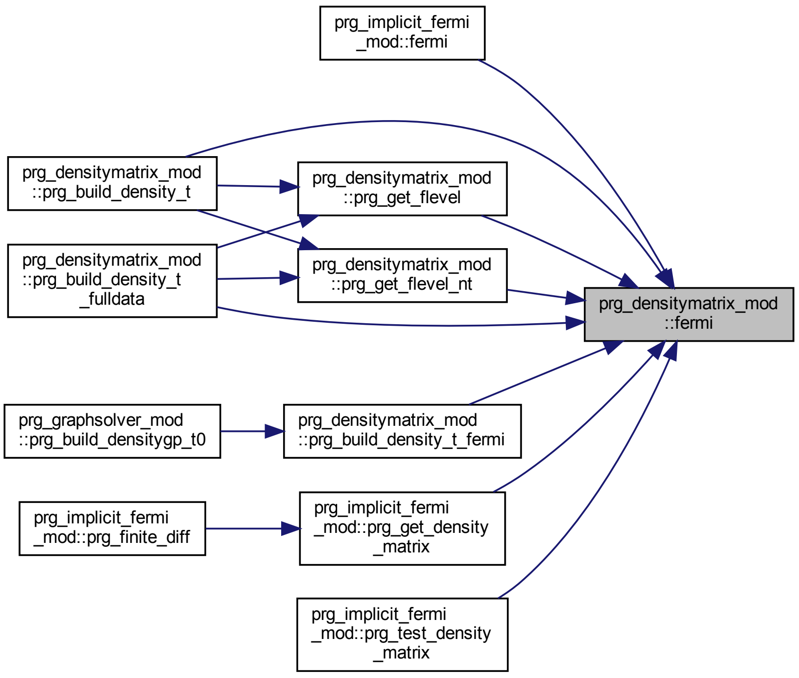
Gives the Fermi distribution value for energy e.

Parameters

e	Energy.
ef	Fermi energy.

Definition at line 628 of file prg_densitymatrix_mod.F90.

Here is the caller graph for this function:


prg_implicit_fermi _mod::fermi
prg_densitymatrix_mod ::prg_build_density_t
prg_densitymatrix_mod ::prg_get_flevel
prg_densitymatrix_mod ::prg_build_density_t _fulldata
prg_densitymatrix_mod ::prg_get_flevel_nt
prg_densitymatrix_mod ::fermi
prg_graphsolver_mod ::prg_build_densitygp_t0
prg_densitymatrix_mod ::prg_build_density_t_fermi
prg_implicit_fermi _mod::prg_finite_diff
prg_implicit_fermi _mod::prg_get_density _matrix
prg_implicit_fermi _mod::prg_test_density _matrix

8.3.2.2 prg_build_atomic_density()

```
subroutine, public prg_densitymatrix_mod::prg_build_atomic_density (
            type(bml_matrix_t), intent(inout) rhoat_bml,
            real(dp), dimension(:), intent(in) numel,
            integer, dimension(:,:), intent(in) hindex,
            integer, dimension(:), intent(in) spindex,
            integer, intent(in) norb,
            character(len=*), intent(in) bml_type )
```

Builds the atomic density matrix. $\rho_{ii} = mathcalZ_{ii}$ Where, $mathcalZ_{ii}$ is the number of electrons for orbital i.

Parameters

rhoat	Output atomic diagonal density matrix,
hindex	Start and end index for every atom in the system.
numel	Number of electrons per specie. It runs over the specie index.
spindex	Specie index.
norbs	Number of orbitals.

Definition at line 341 of file prg_densitymatrix_mod.F90.

8.3.2.3 prg_build_density_t()

```
subroutine, public prg_densitymatrix_mod::prg_build_density_t (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(in) bndfil,
            real(dp), intent(in) kbt,
            real(dp), intent(inout) ef,
            real(dp), dimension(:), intent(out), optional, allocatable eigenvalues_out )
```

Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^{\dagger}$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.

Parameters

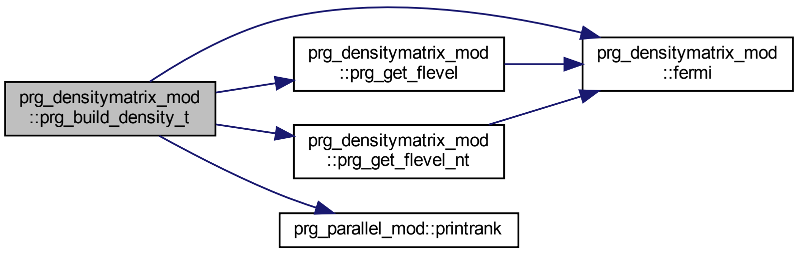
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.
kbt	Electronic temperature.
ef	Fermi level.
eigenvalues_out	Output the eigenvalues.

Warning

This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preorthogonalized.

Definition at line 118 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_build_density_t
prg_densitymatrix_mod ::prg_get_flevel
prg_densitymatrix_mod ::prg_get_flevel_nt
prg_densitymatrix_mod ::fermi
prg_parallel_mod::printrank

8.3.2.4 prg_build_density_t0()

```
subroutine, public prg_densitymatrix_mod::prg_build_density_t0 (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(8), intent(in) threshold,
            real(8), intent(in) bndfil,
            real(dp), dimension(:), intent(out), optional, allocatable eigenvalues_out )
```

Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.

Parameters

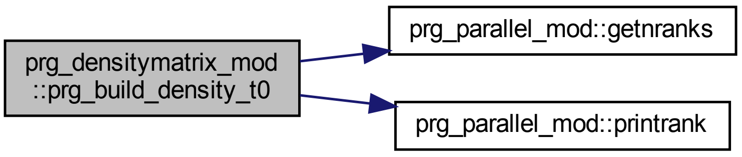
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix.
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.
eigenvalues_out	Output the eigenvalues.

Warning

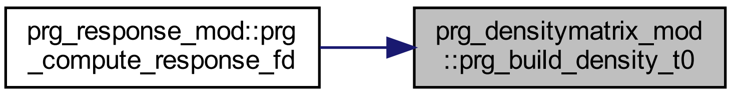
This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preprg_↩ orthogonalized.

Definition at line 35 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_build_density_t0
prg_parallel_mod::getnranks
prg_parallel_mod::printrank

Here is the caller graph for this function:


prg_response_mod::prg _compute_response_fd
prg_densitymatrix_mod ::prg_build_density_t0

8.3.2.5 prg_build_density_t_fermi()

```
subroutine, public prg_densitymatrix_mod::prg_build_density_t_fermi (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(in) kbt,
            real(dp), intent(in) ef,
            integer, intent(in), optional verbose )
```

Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.

Parameters

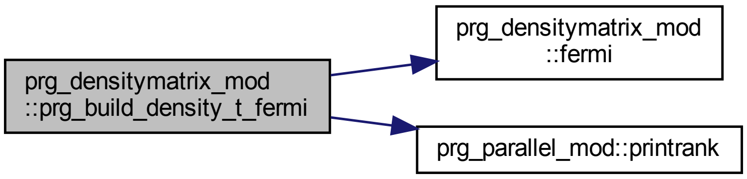
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.

Warning

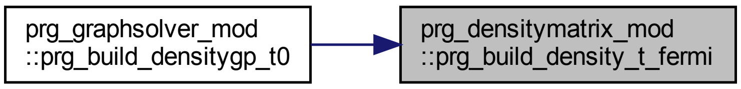
This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preorthogonalized.

Definition at line 279 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_build_density_t_fermi
prg_densitymatrix_mod ::fermi
prg_parallel_mod::printrank

Here is the caller graph for this function:


prg_graphsolver_mod ::prg_build_densitygp_t0
prg_densitymatrix_mod ::prg_build_density_t_fermi

8.3.2.6 prg_build_density_t_fulldata()

```
subroutine, public prg_densitymatrix_mod::prg_build_density_t_fulldata (
             type(bml_matrix_t), intent(in) ham_bml,
             type(bml_matrix_t), intent(inout) rho_bml,
             real(dp), intent(in) threshold,
             real(dp), intent(in) bndfil,
             real(dp), intent(in) kbt,
             real(dp), intent(inout) ef,
             real(dp), dimension(:), intent(inout), allocatable eigenvalues_out,
             type(bml_matrix_t), intent(inout) evects_bml,
             real(dp), dimension(:), intent(inout), allocatable fvals )
```

Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.

Parameters

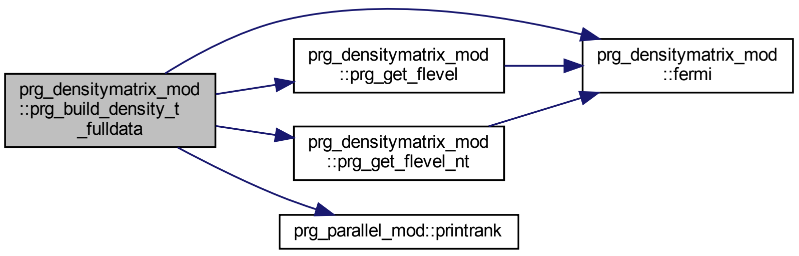
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.
kbt	Electronic temperature.
ef	Fermi level.
eigenvalues_out	Output the eigenvalues.
evects_bml	Output the eigenvectors.
fvals	Output the occupancies.

Warning

This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preorthogonalized.

Definition at line 203 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_build_density_t _fulldata
prg_densitymatrix_mod ::prg_get_flevel
prg_densitymatrix_mod ::prg_get_flevel_nt
prg_densitymatrix_mod ::fermi
prg_parallel_mod::printrank

8.3.2.7 prg_check_idempotency()

```
subroutine, public prg_densitymatrix_mod::prg_check_idempotency (
            type(bml_matrix_t), intent(in) mat_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(out) idempotency )
```

To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.

Parameters

mat_bml	Some bml matrix
idempotency	(Output value of the idempotency error)

Definition at line 604 of file prg_densitymatrix_mod.F90.

8.3.2.8 prg_get_eigenvalues()

```
subroutine, public prg_densitymatrix_mod::prg_get_eigenvalues (
            type(bml_matrix_t), intent(in) ham_bml,
            real(dp), dimension(:), intent(inout), allocatable eigenvalues,
            integer, intent(in) verbose )
```

Gets the eigenvalues of the Orthogonalized Hamiltonian.

Parameters

ham_bml	Input Orthogonalized Hamiltonian matrix.
eigenvalues	Output eigenvalues of the system.
verbose	Verbosity level.

Definition at line 559 of file prg_densitymatrix_mod.F90.

8.3.2.9 prg_get_flevel()

```
subroutine, public prg_densitymatrix_mod::prg_get_flevel (
            real(dp), dimension(:), intent(in) eigenvalues,
            real(dp), intent(in) kbt,
            real(dp), intent(in) bndfil,
            real(dp) tol,
            real(dp), intent(inout) Ef,
            logical, intent(inout) err )
```

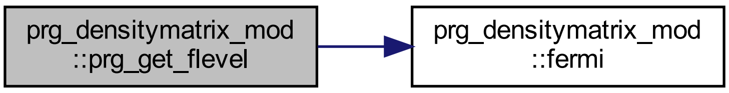
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.

Parameters

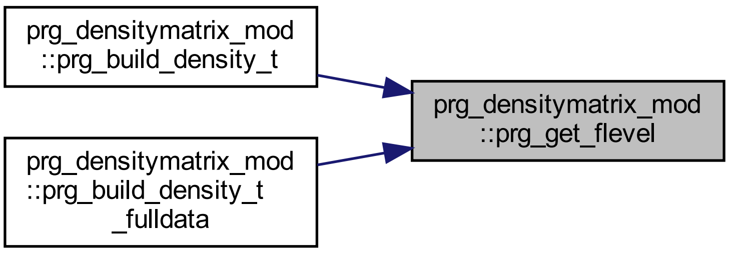
eigenvalues	Eigenvalues of the system ($\{\epsilon_k\}$).
kbt	Temperature times the Boltzmann's constant (k_bT).
bndfil	Filing factor ($N_{el}/(2*N_{orbs})$).
tol	Tolerance for the bisection method.
Ef	Fermi level (μ).
err	Error logical variable

Definition at line 405 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_get_flevel
prg_densitymatrix_mod ::fermi

Here is the caller graph for this function:


prg_densitymatrix_mod ::prg_build_density_t
prg_densitymatrix_mod ::prg_build_density_t _fulldata
prg_densitymatrix_mod ::prg_get_flevel

8.3.2.10 prg_get_flevel_nt()

```
subroutine, public prg_densitymatrix_mod::prg_get_flevel_nt (
            real(dp), dimension(:), intent(in) eigenvalues,
            real(dp), intent(in) kbt,
            real(dp), intent(in) bndfil,
            real(dp), intent(in) tol,
            real(dp), intent(inout) ef,
            logical, intent(inout) err,
            integer, intent(in), optional verbose )
```

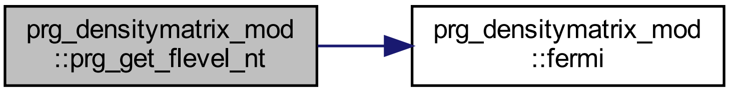
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Newton-Raphson method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.

Parameters

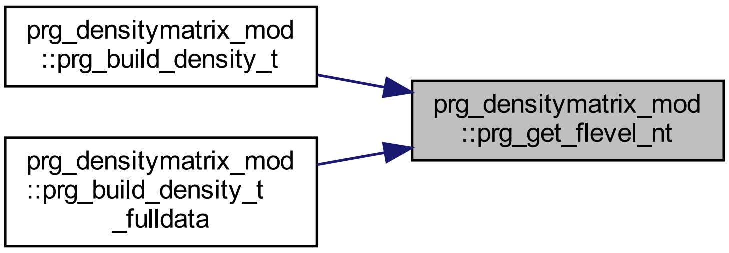
eigenvalues	Eigenvalues of the system ($\{\epsilon_k\}$).
kbt	Temperature times the Boltzmann's constant (k_bT).
bndfil	Filing factor ($N_{el}/(2 * N_{orbs})$).
tol	Tolerance for the bisection method.
Ef	Fermi level (μ).
err	Error logical variable

Definition at line 477 of file prg_densitymatrix_mod.F90.

Here is the call graph for this function:


prg_densitymatrix_mod ::prg_get_flevel_nt
prg_densitymatrix_mod ::fermi

Here is the caller graph for this function:


prg_densitymatrix_mod ::prg_build_density_t
prg_densitymatrix_mod ::prg_build_density_t _fulldata
prg_densitymatrix_mod ::prg_get_flevel_nt

8.3.3 Variable Documentation

8.3.3.1 dp

```
integer, parameter prg_densitymatrix_mod::dp = kind(1.0d0)  [private]
```

Definition at line 14 of file prg_densitymatrix_mod.F90.

8.4 prg_dos_mod Module Reference

A module to compute the Density of state (DOS) and IDOS.

Functions/Subroutines

- subroutine, public prg_write_tdos (eigenvals, gamma, npts, emin, emax, filename)

  Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ *Where* $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates.$
- real(dp) function lorentz (energy, eigenvals, loads, Gamma)

  Lorentzian Function.

Variables

- integer, parameter dp = kind(1.0d0)

8.4.1 Detailed Description

A module to compute the Density of state (DOS) and IDOS.

This module will be used to compute DOS and IDOS.

Todo Add LDOS.

8.4.2 Function/Subroutine Documentation

8.4.2.1 lorentz()

```
real(dp) function prg_dos_mod::lorentz (
            real(dp), intent(in) energy,
            real(dp), dimension(:), intent(in) eigenvals,
            real(dp), dimension(:), intent(in) loads,
            real(dp), intent(in) Gamma )  [private]
```

Lorentzian Function.

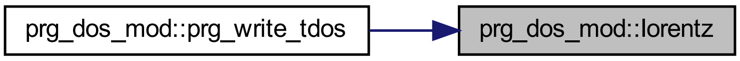
Computes: $L(\epsilon) = \sum_k \frac{\omega(k)\Gamma}{2\pi} \frac{1}{(\epsilon-\epsilon_k)^2+(\Gamma/2)^2}$

Parameters

energy	Energy point.
eigenvals	Eigenvalues of the system.
Gamma	Lorentz function broadening.

Definition at line 77 of file prg_dos_mod.F90.

Here is the caller graph for this function:



8.4.2.2 prg_write_tdos()

```
subroutine, public prg_dos_mod::prg_write_tdos (
             real(dp), dimension(:), intent(in) eigenvals,
             real(dp), intent(in) gamma,
             integer, intent(in) npts,
             real(dp), intent(in) emin,
             real(dp), intent(in) emax,
             character(len=*), intent(in) filename )
```

Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates$.

Note

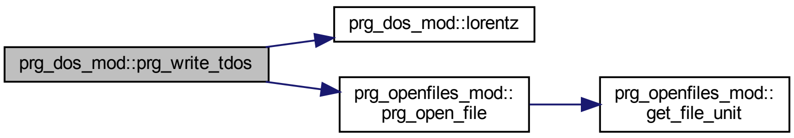
DOS is NOT shifted respect to Ef.

Parameters

eigenvals	Eigenvalues of the system. \para gamma Lorentzian width.
npts	Number of energy points.
emin	Minimum energy value.
emax	Maximum energy value.
filename	Filename to write the DOS.

Definition at line 35 of file prg_dos_mod.F90.

Here is the call graph for this function:


prg_dos_mod::prg_write_tdos
prg_dos_mod::lorentz
prg_openfiles_mod::prg_open_file
prg_openfiles_mod::get_file_unit

8.4.3 Variable Documentation

8.4.3.1 dp

```
integer, parameter prg_dos_mod::dp = kind(1.0d0)  [private]
```

Definition at line 17 of file prg_dos_mod.F90.

8.5 prg_ewald_mod Module Reference

Functions/Subroutines

- subroutine, public ewald_real_space_single_latte (COULOMBV, I, RXYZ, Box, Nr_elem, DELTAQ, J, U, Element_Pointer, Nr_atoms, COULACC, HDIM, Max_Nr_Neigh)

    Find Coulomb potential on site I from single charge at site J.
- subroutine, public ewald_real_space_single (COULOMBV, FCOUL, I, RX, RY, RZ, LBox, DELTAQ, J, U, Element_Type, Nr_atoms, COULACC, TIMERATIO, HDIM, Max_Nr_Neigh)
- subroutine, public ewald_real_space_matrix_latte (E, RXYZ, Box, U, Element_Pointer, Nr_atoms, COULACC, nebcoul, totnebcoul, HDIM, Max_Nr_Neigh, Nr_Elem)
- subroutine, public ewald_real_space_latte (COULOMBV, I, RXYZ, Box, DELTAQ, U, Element_Pointer, Nr_↩ atoms, COULACC, nebcoul, totnebcoul, HDIM, Max_Nr_Neigh, Nr_Elem)
- subroutine, public ewald_real_space_test (COULOMBV, I, RX, RY, RZ, LBox, DELTAQ, U, Element_Type, Nr_atoms, COULACC, nnRx, nnRy, nnRz, nrnnlist, nnType, Max_Nr_Neigh)
- subroutine, public ewald_real_space (COULOMBV, FCOUL, I, RX, RY, RZ, LBox, DELTAQ, U, Element_Type, Nr_atoms, COULACC, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, HDIM, Max_Nr_Neigh)
- subroutine, public ewald_k_space_latte (COULOMBV, RXYZ, Box, DELTAQ, Nr_atoms, COULACC, Max_↩ Nr_Neigh)
- subroutine, public ewald_k_space_matrix_latte (E, RXYZ, Box, Nr_atoms, COULACC, Max_Nr_Neigh, nebcoul, totnebcoul)
- subroutine, public ewald_k_space_latte_single (COULOMBV, J, RXYZ, Box, DELTAQ, Nr_atoms, COULA↩ CC)
- subroutine, public ewald_k_space_test (COULOMBV, RX, RY, RZ, LBox, DELTAQ, Nr_atoms, COULACC, Max_Nr_Neigh)

Variables

- integer, parameter dp = kind(1.0d0)

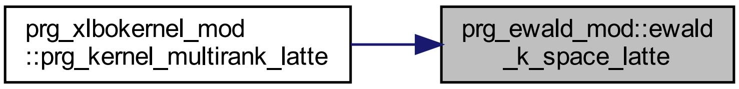
8.5.1 Function/Subroutine Documentation

8.5.1.1 ewald_k_space_latte()

```
subroutine, public prg_ewald_mod::ewald_k_space_latte (
            real(prec), dimension(nr_atoms), intent(out) COULOMBV,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            integer, intent(in) Max_Nr_Neigh )
```

Definition at line 745 of file prg_ewald_mod.F90.

Here is the caller graph for this function:

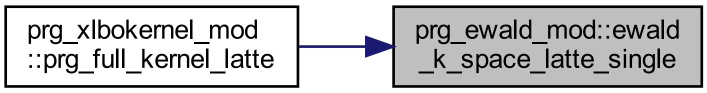

prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_ewald_mod::ewald _k_space_latte

8.5.1.2 ewald_k_space_latte_single()

```
subroutine, public prg_ewald_mod::ewald_k_space_latte_single (
            real(prec), dimension(nr_atoms), intent(out) COULOMBV,
            integer, intent(in) J,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC )
```

Definition at line 999 of file prg_ewald_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_full_kernel_latte
prg_ewald_mod::ewald _k_space_latte_single

8.5.1.3 ewald_k_space_matrix_latte()

```
subroutine, public prg_ewald_mod::ewald_k_space_matrix_latte (
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) E,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            integer, intent(in) Max_Nr_Neigh,
            integer, dimension(4,max_nr_neigh,nr_atoms), intent(in) nebcoul,
            integer, dimension(nr_atoms), intent(in) totnebcoul )
```

Definition at line 871 of file prg_ewald_mod.F90.

8.5.1.4 ewald_k_space_test()

```
subroutine, public prg_ewald_mod::ewald_k_space_test (
            real(prec), dimension(nr_atoms), intent(out) COULOMBV,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            integer, intent(in) Max_Nr_Neigh )
```

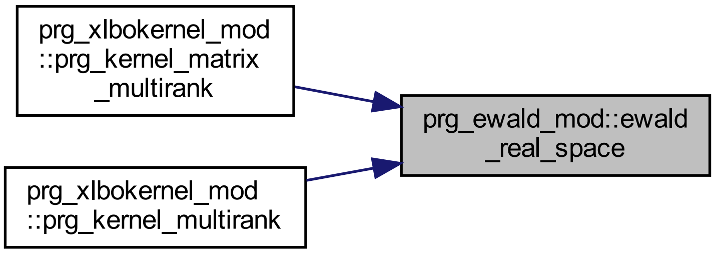
Definition at line 1097 of file prg_ewald_mod.F90.

8.5.1.5 ewald_real_space()

```
subroutine, public prg_ewald_mod::ewald_real_space (
            real(prec), intent(out) COULOMBV,
            real(prec), dimension(3), intent(out) FCOUL,
            integer, intent(in) I,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            real(prec), dimension(nr_atoms), intent(in) U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            real(prec), intent(in) TIMERATIO,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
            integer, dimension(nr_atoms), intent(in) nrnnlist,
            integer, dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh )
```

Definition at line 623 of file prg_ewald_mod.F90.

Here is the caller graph for this function:

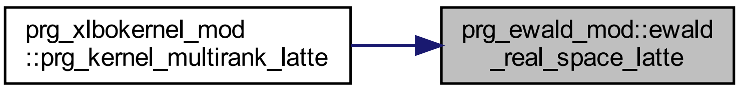

prg_xlbokernel_mod ::prg_kernel_matrix _multirank
prg_xlbokernel_mod ::prg_kernel_multirank
prg_ewald_mod::ewald _real_space

8.5.1.6 ewald_real_space_latte()

```
subroutine, public prg_ewald_mod::ewald_real_space_latte (
              real(prec), intent(out) COULOMBV,
              integer, intent(in) I,
              real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
              real(prec), dimension(3,3), intent(in) Box,
              real(prec), dimension(nr_atoms), intent(in) DELTAQ,
              real(prec), dimension(nr_elem), intent(in) U,
              integer, dimension(nr_atoms), intent(in) Element_Pointer,
              integer, intent(in) Nr_atoms,
              real(prec), intent(in) COULACC,
              integer, dimension(4,max_nr_neigh,nr_atoms), intent(in) nebcoul,
              integer, dimension(nr_atoms), intent(in) totnebcoul,
              integer, intent(in) HDIM,
              integer, intent(in) Max_Nr_Neigh,
              integer, intent(in) Nr_Elem )
```

Definition at line 389 of file prg_ewald_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_ewald_mod::ewald _real_space_latte

8.5.1.7 ewald_real_space_matrix_latte()

```
subroutine, public prg_ewald_mod::ewald_real_space_matrix_latte (
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) E,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            real(prec), dimension(nr_elem), intent(in) U,
            integer, dimension(nr_atoms), intent(in) Element_Pointer,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            integer, dimension(4,max_nr_neigh,nr_atoms), intent(in) nebcoul,
            integer, dimension(nr_atoms), intent(in) totnebcoul,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh,
            integer, intent(in) Nr_Elem )
```

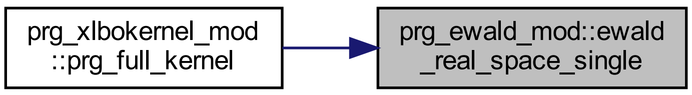
Definition at line 265 of file prg_ewald_mod.F90.

8.5.1.8 ewald_real_space_single()

```
subroutine, public prg_ewald_mod::ewald_real_space_single (
            real(prec), intent(out) COULOMBV,
            real(prec), dimension(3), intent(out) FCOUL,
            integer, intent(in) I,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            integer, intent(in) J,
            real(prec), dimension(nr_atoms), intent(in) U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            real(prec), intent(in) TIMERATIO,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh )
```

Definition at line 142 of file prg_ewald_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_full_kernel
prg_ewald_mod::ewald _real_space_single

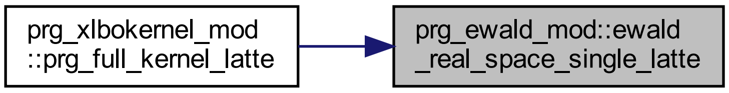
8.5.1.9 ewald_real_space_single_latte()

```
subroutine, public prg_ewald_mod::ewald_real_space_single_latte (
            real(prec), intent(out) COULOMBV,
            integer, intent(in) I,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            integer, intent(in) Nr_elem,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            integer, intent(in) J,
            real(prec), dimension(nr_elem), intent(in) U,
            integer, dimension(nr_atoms), intent(in) Element_Pointer,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh )
```

Find Coulomb potential on site I from single charge at site J.

Definition at line 30 of file prg_ewald_mod.F90.

Here is the caller graph for this function:



8.5.1.10 ewald_real_space_test()

```
subroutine, public prg_ewald_mod::ewald_real_space_test (
            real(prec), intent(out) COULOMBV,
            integer, intent(in) I,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) DELTAQ,
            real(prec), dimension(nr_atoms), intent(in) U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer, intent(in) Nr_atoms,
            real(prec), intent(in) COULACC,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
            integer, dimension(nr_atoms), intent(in) nrnnlist,
            integer, dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
            integer, intent(in) Max_Nr_Neigh )
```

Definition at line 506 of file prg_ewald_mod.F90.

8.5.2 Variable Documentation

8.5.2.1 dp

```
integer, parameter prg_ewald_mod::dp = kind(1.0d0)  [private]
```

Definition at line 12 of file prg_ewald_mod.F90.

8.6 prg_extras_mod Module Reference

Extra routines.

Data Types

- interface prg_memory_consumption
- interface to_string

Functions/Subroutines

- character(len=:) function, allocatable to_string_integer (i)

    Convert integer to string.
- character(len=:) function, allocatable to_string_long_long (i)

    Convert integer to string.
- character(len=:) function, allocatable to_string_double (x)

    Convert double to string.
- subroutine, public prg_print_matrix (matname, amat, i1, i2, j1, j2)

    To write a dense matrix to screen.
- real(dp) function, public mls ()

    To get the actual time in milliseconds.
- subroutine, public prg_delta (x, s, nn, dta)

    Delta function $||X^\wedge tSX - I||$.
- subroutine, public prg_get_mem (procname, tag)

    Get proc memory.
- subroutine prg_twonorm (a, nn, norm2)

    Gets the norm2 of a square matrix.
- real(dp) function, public prg_norm2 (a)

    Gets the norm2 of a vector.

Variables

- integer, parameter dp = kind(1.0d0)

8.6.1 Detailed Description

Extra routines.

A module to add any extra routine considered necessary but which is NOT essential for any other PROGRESS routine.

8.6.2 Function/Subroutine Documentation

8.6.2.1 mls()

```
real(dp) function, public prg_extras_mod::mls
```

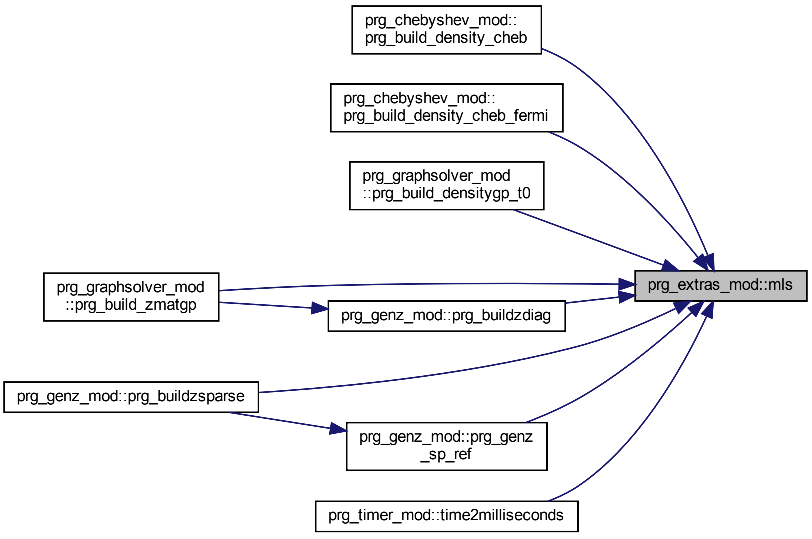
To get the actual time in milliseconds.

Parameters

mls	Output value with the machine time in milliseconds.

Definition at line 140 of file prg_extras_mod.F90.

Here is the caller graph for this function:



8.6.2.2 prg_delta()

```
subroutine, public prg_extras_mod::prg_delta (
            real(dp), dimension(nn,nn) x,
            real(dp), dimension(nn,nn) s,
            integer nn,
            real(dp) dta )
```

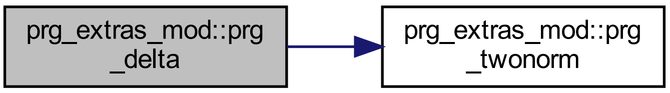
Delta function ||X^tSX - I||.

Parameters

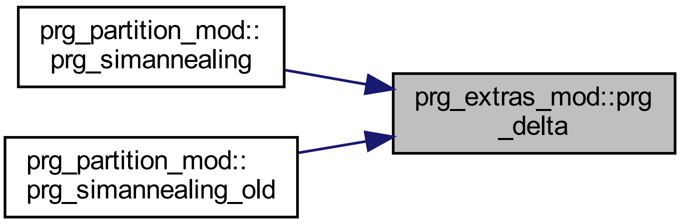
x	input matrix.
s	overlap matrix.
dta	Delta output value.

Definition at line 156 of file prg_extras_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:


prg_partition_mod::prg_simannealing
prg_partition_mod::prg_simannealing_old
prg_extras_mod::prg_delta

8.6.2.3 prg_get_mem()

```
subroutine, public prg_extras_mod::prg_get_mem (
             character(*), intent(in) procname,
             character(*), intent(in) tag )
```

Get proc memory.

Parameters

procname	Process name to get the mem usage.
tag	Tag to pprint the processor mem usage.

Definition at line 192 of file prg_extras_mod.F90.

8.6.2.4 prg_norm2()

```
real(dp) function, public prg_extras_mod::prg_norm2 (
            real(dp), dimension(:), intent(in) a )
```

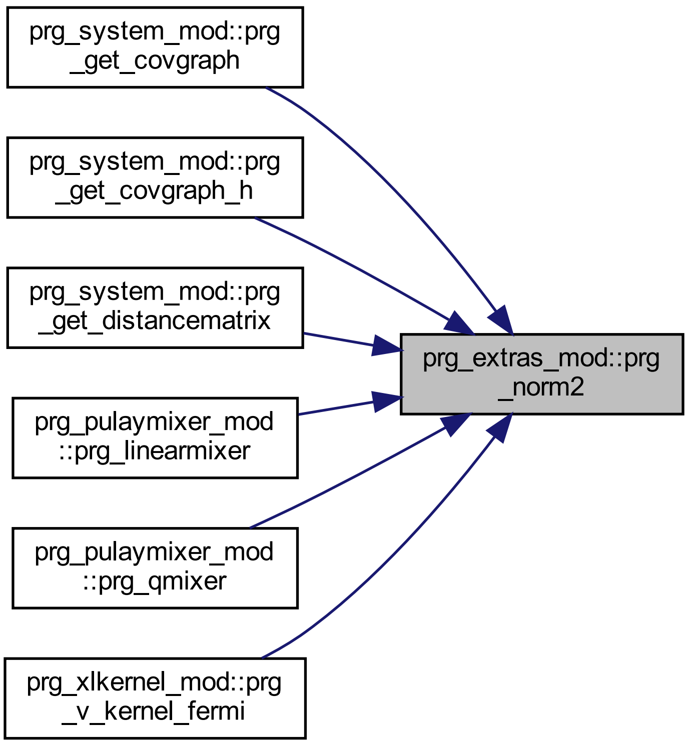
Gets the norm2 of a vector.

Parameters

a	Vector.

Definition at line 241 of file prg_extras_mod.F90.

Here is the caller graph for this function:


prg_system_mod::prg_get_covgraph
prg_system_mod::prg_get_covgraph_h
prg_system_mod::prg_get_distancematrix
prg_pulaymixer_mod::prg_linearmixer
prg_pulaymixer_mod::prg_qmixer
prg_xlkernel_mod::prg_v_kernel_fermi
prg_extras_mod::prg_norm2

8.6.2.5 prg_print_matrix()

```
subroutine, public prg_extras_mod::prg_print_matrix (
             character(len=*) matname,
             real(dp), dimension(:,:), intent(in) amat,
             integer, intent(in) i1,
             integer, intent(in) i2,
             integer, intent(in) j1,
             integer, intent(in) j2 )
```

To write a dense matrix to screen.

Parameters

matname	Matrix name.
amat	Matrix to be printed.
i1	Print from row i1.
i2	Print up to from row i2.
j1	Print from column j1.
j2	Print up to column j2.

Definition at line 101 of file prg_extras_mod.F90.

8.6.2.6 prg_twonorm()

```
subroutine prg_extras_mod::prg_twonorm (
             real(dp), dimension(nn,nn) a,
             integer nn,
             real(dp) norm2 )  [private]
```

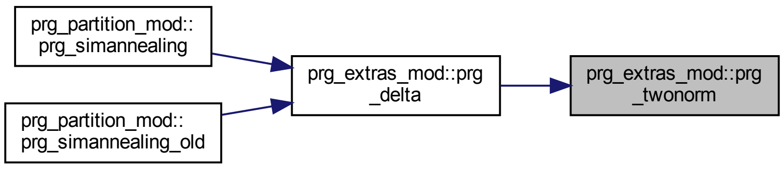
Gets the norm2 of a square matrix.

Parameters

a	Square matrix.
nn	Matrix size.
norm2	Two-norm of matrix a.

Definition at line 216 of file prg_extras_mod.F90.

Here is the caller graph for this function:



8.6.2.7 to_string_double()

```
character(len=:)  function, allocatable prg_extras_mod::to_string_double (
             double precision, intent(in) x )  [private]
```

Convert double to string.

Parameters

x	The double

Returns

The string

Definition at line 81 of file prg_extras_mod.F90.

8.6.2.8 to_string_integer()

```
character(len=:)  function, allocatable prg_extras_mod::to_string_integer (
             integer, intent(in) i )  [private]
```

Convert integer to string.

Parameters

i	The integer

Returns

The string

Definition at line 47 of file prg_extras_mod.F90.

8.6.2.9 to_string_long_long()

```
character(len=:)  function, allocatable prg_extras_mod::to_string_long_long (
             integer(kind=c_long_long), intent(in) i )  [private]
```

Convert integer to string.

Parameters

i	The integer

Returns

The string

Definition at line 63 of file prg_extras_mod.F90.

8.6.3 Variable Documentation

8.6.3.1 dp

```
integer, parameter prg_extras_mod::dp = kind(1.0d0)  [private]
```

Definition at line 31 of file prg_extras_mod.F90.

8.7 prg_genz_mod Module Reference

To produce a matrix Z which is needed to orthogonalize H.

Data Types

- type genzspinp

    Input for the genz driver. This type controlls all the variables that are needed by genz.

Functions/Subroutines

- subroutine, public prg_parse_zsp (input, filename)

  The parser for genz solver.
- subroutine, public prg_init_zspmat (igenz, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type, bml_element_type)

  Initiates the matrices for the Xl integration of Z.
- subroutine, public prg_buildzdiag (smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)

  Usual subroutine involving diagonalization. $Z = U\sqrt{s}U^\dagger$, where U = eigenvectors and s = eigenvalues. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.
- subroutine, public prg_buildzsparse (smat_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3↩ _bml, zk4_bml, zk5_bml, zk6_bml, nfirst, nrefi, nreff, thresholdi, thresholdf, integration, verbose)

  Inverse factorization using Niklasson's algorithm.
- subroutine, public prg_genz_sp_initialz0 (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)

  Initial estimation of Z.
- subroutine, public prg_genz_sp_initial_zmat (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)

  Initial estimation of Z.
- subroutine prg_genz_sp_int (zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)

  Inverse factorization using Niklasson's algorithm.
- subroutine, public prg_genz_sp_ref (smat_bml, zmat_bml, nref, norb, bml_type, threshold)

  Iterative refinement.

Variables

- integer, parameter dp = kind(1.0d0)

8.7.1 Detailed Description

To produce a matrix Z which is needed to orthogonalize H.

$H_{orth} = Z^\dagger HZ$ See Negre 2016 [2]

8.7.2 Function/Subroutine Documentation

8.7.2.1 prg_buildzdiag()

```
subroutine, public prg_genz_mod::prg_buildzdiag (
            type(bml_matrix_t), intent(inout) smat_bml,
            type(bml_matrix_t) zmat_bml,
            real(dp) threshold,
            integer, intent(in) mdimin,
            character(len=*), intent(in) bml_type,
            integer, intent(in), optional verbose )
```

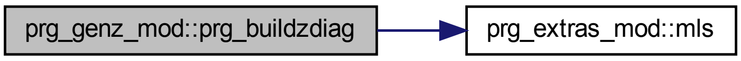
Usual subroutine involving diagonalization. $Z = U\sqrt{s}U^\dagger$, where U = eigenvectors and s = eigenvalues. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.

Parameters

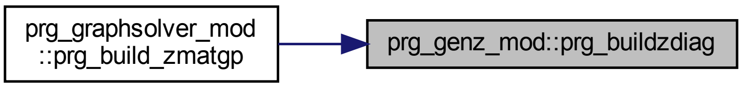
smat_bml	Overlap matrix in bml format.
zmat_bml	Congruence transform in bml format.
threshold	Threshold value to use, in this case, only in the backtransformation to ellpack format.
mdim	Maximun nonzero to use, in this case, only in the backtransformation to ellpack format.
bml_type	the bml type we are passing.

Definition at line 175 of file prg_genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:


prg_graphsolver_mod ::prg_build_zmatgp
prg_genz_mod::prg_buildzdiag

8.7.2.2 prg_buildzsparse()

```
subroutine, public prg_genz_mod::prg_buildzsparse (
             type(bml_matrix_t) smat_bml,
             type(bml_matrix_t) zmat_bml,
             integer igenz,
             integer mdim,
             character(20) bml_type,
             type(bml_matrix_t) zk1_bml,
             type(bml_matrix_t) zk2_bml,
             type(bml_matrix_t) zk3_bml,
             type(bml_matrix_t) zk4_bml,
             type(bml_matrix_t) zk5_bml,
             type(bml_matrix_t) zk6_bml,
             integer nfirst,
             integer nrefi,
             integer nreff,
             real(dp) thresholdi,
             real(dp) thresholdf,
             logical integration,
             integer verbose )
```

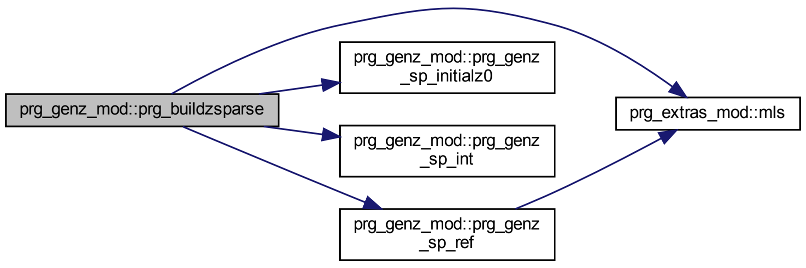
Inverse factorization using Niklasson's algorithm.

Parameters

smat_bml	overlap matrix
zmat_bml	congruence transform to be updated or computed. (bml format)
igenz	counter to keep track of the calls to this subroutine.
mdim	dimension of the maxnonzero per row.
zk1_bml-zk6_bml	history of the past congruence transforms.
nfirst	first pre septs with nrefi and thresholdi.
nrefi	number of refinement iterations for the firsts "nfirst" steps.
nreff	number of refinement iterations for the rest of the steps.
integration	if we want to apply xl integration scheme for z (default is always .true.)
verbose	to print extra information.

Definition at line 276 of file prg_genz_mod.F90.

Here is the call graph for this function:


prg_genz_mod::prg_buildzsparse
prg_genz_mod::prg_genz_sp_initialz0
prg_genz_mod::prg_genz_sp_int
prg_genz_mod::prg_genz_sp_ref
prg_extras_mod::mls

8.7.2.3 prg_genz_sp_initial_zmat()

```
subroutine, public prg_genz_mod::prg_genz_sp_initial_zmat (
              type(bml_matrix_t), intent(in) smat_bml,
              type(bml_matrix_t) zmat_bml,
              integer norb,
              integer mdim,
              character(20) bml_type_f,
              real(dp), intent(in) threshold )
```

Initial estimation of Z.

Note

Most of the operations are done in pure dense format. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.

Parameters

smat_bml	Overlap matrix in bml format.
zmat_bml	Congruence transform in bml format.
norb	Congruence transform in bml format.
mdim	Congruence transform in bml format.
bml_↩ type_f	The bml final type of zmat_bml.
threshold	Threshold value to use, in this case, only in the backtransformation to ellpack format.

Definition at line 479 of file prg_genz_mod.F90.

8.7.2.4 prg_genz_sp_initialz0()

```
subroutine, public prg_genz_mod::prg_genz_sp_initialz0 (
            type(bml_matrix_t), intent(in) smat_bml,
            type(bml_matrix_t) zmat_bml,
            integer norb,
            integer mdim,
            character(20) bml_type_f,
            real(dp) threshold )
```

Initial estimation of Z.

Note

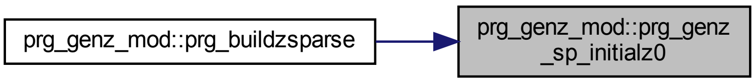
Most of the operations are done in pure dense format. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.

Parameters

smat_bml	Overlap matrix in bml format.
zmat_bml	Congruence transform in bml format.
norb	Congruence transform in bml format.
mdim	Congruence transform in bml format.
bml_↩ type_f	The bml final type of zmat_bml.
threshold	Threshold value to use, in this case, only in the backtransformation to ellpack format.

Definition at line 339 of file prg_genz_mod.F90.

Here is the caller graph for this function:


prg_genz_mod::prg_buildzsparse
prg_genz_mod::prg_genz _sp_initialz0

8.7.2.5 prg_genz_sp_int()

```
subroutine prg_genz_mod::prg_genz_sp_int (
             type(bml_matrix_t) zmat_bml,
             type(bml_matrix_t) zk1_bml,
             type(bml_matrix_t) zk2_bml,
             type(bml_matrix_t) zk3_bml,
             type(bml_matrix_t) zk4_bml,
             type(bml_matrix_t) zk5_bml,
             type(bml_matrix_t) zk6_bml,
             integer igenz,
             integer norb,
             character(20) bml_type,
             real(dp) threshold )  [private]
```

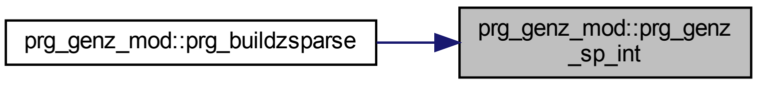
Inverse factorization using Niklasson's algorithm.

Parameters

smat_bml	overlap matrix
zmat_bml	congruence transform to be updated or computed. (bml format)
mdim	dimension of the maxnonzero per row.
zk1_bml-zk6_bml	history of the past congruence transforms.
igenz	counter to keep track of the calls to this subroutine.
norb	Congruence transform in bml format.
bml_type_f	The bml final type of zmat_bml.
threshold	Threshold value to use.

Definition at line 636 of file prg_genz_mod.F90.

Here is the caller graph for this function:



8.7.2.6 prg_genz_sp_ref()

```
subroutine, public prg_genz_mod::prg_genz_sp_ref (
            type(bml_matrix_t), intent(in) smat_bml,
            type(bml_matrix_t), intent(inout) zmat_bml,
            integer, intent(in) nref,
            integer, intent(inout) norb,
            character(20), intent(in) bml_type,
            real(dp), intent(in) threshold )
```

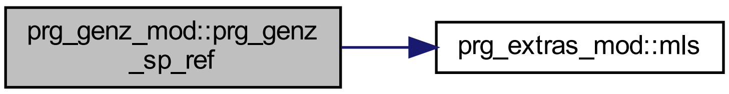
Iterative refinement.

Parameters

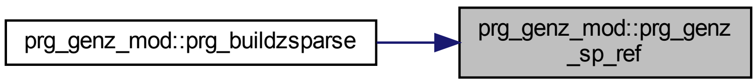
smat_bml	overlap matrix
zmat_bml	congruence transform to be updated or computed. (bml format)
nref	Number of refinement iterations.
bml_↩ type_f	The bml final type of zmat_bml.
threshold	Threshold value to use.
verbose	to print extra information.

Definition at line 720 of file prg_genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.2.7 prg_init_zspmat()

```
subroutine, public prg_genz_mod::prg_init_zspmat (
             integer igenz,
             type(bml_matrix_t) zk1_bml,
             type(bml_matrix_t) zk2_bml,
             type(bml_matrix_t) zk3_bml,
             type(bml_matrix_t) zk4_bml,
             type(bml_matrix_t) zk5_bml,
             type(bml_matrix_t) zk6_bml,
             integer norb,
             character(20) bml_type,
             character(20), optional bml_element_type )
```

Initiates the matrices for the Xl integration of Z.

Parameters

self	input zsp variables
zk1_bml-zk6_bml	history record of the previous Z matrices.
norb	number of orbitals.
bml_type	the bml format we are passing.

Definition at line 135 of file prg_genz_mod.F90.

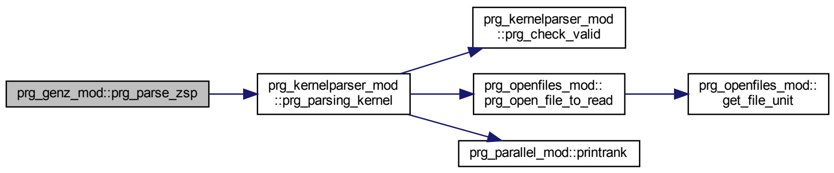
8.7.2.8 prg_parse_zsp()

```
subroutine, public prg_genz_mod::prg_parse_zsp (
             type(genzspinp), intent(inout) input,
             character(len=*) filename )
```

The parser for genz solver.

Definition at line 68 of file prg_genz_mod.F90.

Here is the call graph for this function:


prg_genz_mod::prg_parse_zsp
prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

8.7.3 Variable Documentation

8.7.3.1 dp

```
integer, parameter prg_genz_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_genz_mod.F90.

8.8 prg_graph_mod Module Reference

The graph module.

Data Types

- type graph_partitioning_t

  Trace per iteration.
- type subgraph_t

  Subgraph type.

Functions/Subroutines

- subroutine, public prg_initsubgraph (sg, pnum, hsize)

  Initialize subgraph.
- subroutine, public prg_destroysubgraph (sg)

  Destroy subgraph.
- subroutine, public prg_initgraphpartitioning (gp, pname, np, nnodes, nnodes2)

  Initialize graph partitioning.
- subroutine, public prg_destroygraphpartitioning (gp)

  Destroy graph partitioning.
- subroutine, public prg_printgraphpartitioning (gp)

  Print graph partitioning structure data.
- subroutine, public prg_equalpartition (gp, nodesPerPart, nnodes)

Create equal graph partitions, based on number of rows/orbitals.
- subroutine, public prg_equalgrouppartition (gp, hindex, ngroup, nodesPerPart, nnodes)

  Create equal group graph partitions, based on number of atoms/groups.
- subroutine, public prg_filepartition (gp, partFile)

  Read graph partitions from a file, based on number of rows/orbitals.
- subroutine prg_readpart (gp, partFile)

  Read parts (core) from part file.
- subroutine, public prg_fnormgraph (gp)

  Accumulate trace norm across all subgraphs.

Variables

- integer, parameter dp = kind(1.0d0)

8.8.1 Detailed Description

The graph module.

8.8.2 Function/Subroutine Documentation

8.8.2.1 prg_destroygraphpartitioning()

```
subroutine, public prg_graph_mod::prg_destroygraphpartitioning (
            type (graph_partitioning_t), intent(inout) gp )
```

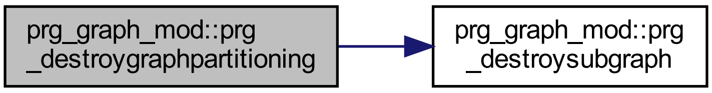
Destroy graph partitioning.

Parameters

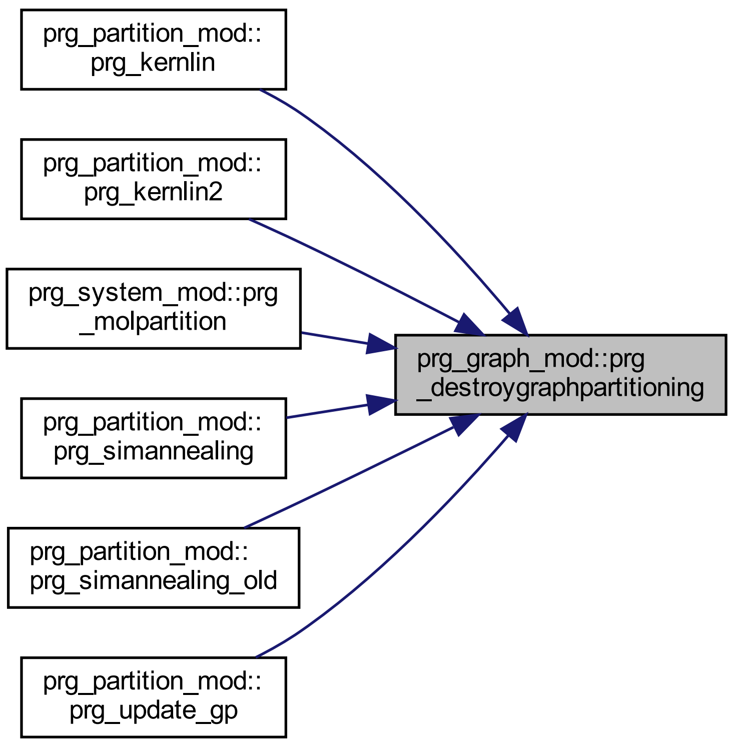
sg	Subgraph

Definition at line 263 of file prg_graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.2.2 prg_destroysubgraph()

```
subroutine, public prg_graph_mod::prg_destroysubgraph (
            type (subgraph_t), intent(inout) sg )
```

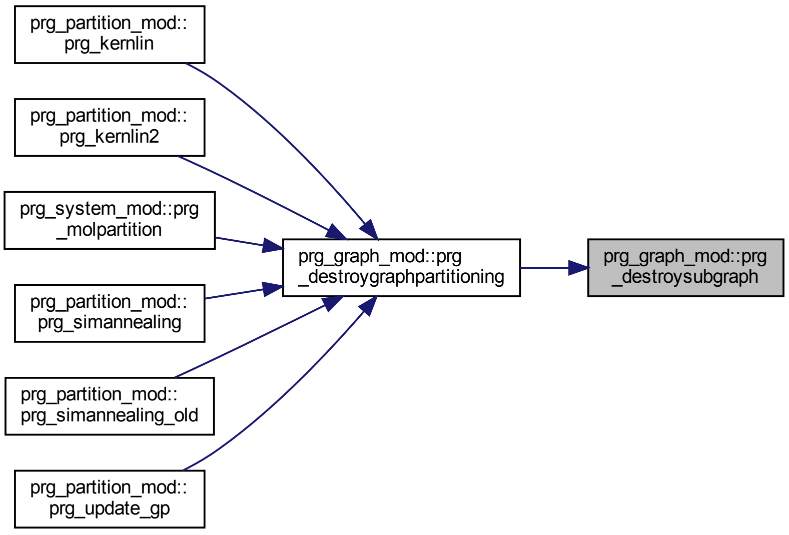
Destroy subgraph.

Parameters

sg	Subgraph

Definition at line 159 of file prg_graph_mod.F90.

Here is the caller graph for this function:



8.8.2.3 prg_equalgrouppartition()

```
subroutine, public prg_graph_mod::prg_equalgrouppartition (
              type (graph_partitioning_t), intent(inout) gp,
              integer, dimension(2,ngroup), intent(in) hindex,
              integer, intent(in) ngroup,
              integer, intent(in) nodesPerPart,
              integer, intent(in) nnodes )
```

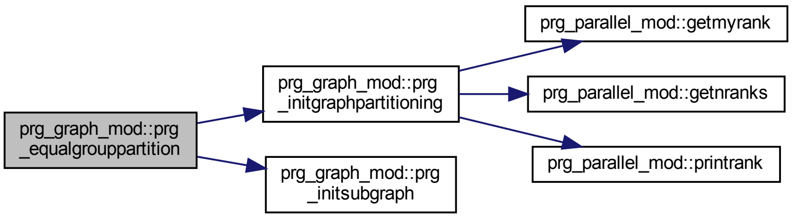
Create equal group graph partitions, based on number of atoms/groups.

Parameters

gp	Graph partitioning
hindex	Node indeces that represent ranges of atoms/groups
ngroup	Number of group nodes
nodesPerPart	Number of core nodes per partition
nnodes	Total nodes in Hamiltonian matrix

Definition at line 402 of file prg_graph_mod.F90.

Here is the call graph for this function:



8.8.2.4 prg_equalpartition()

```
subroutine, public prg_graph_mod::prg_equalpartition (
             type (graph_partitioning_t), intent(inout) gp,
             integer, intent(in) nodesPerPart,
             integer, intent(in) nnodes )
```

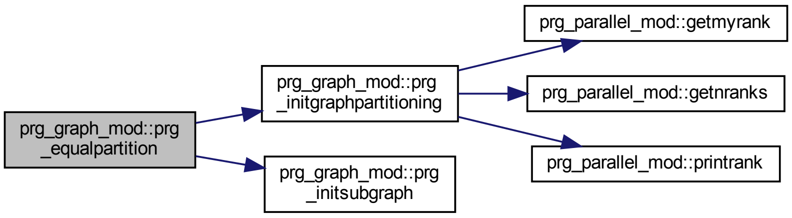
Create equal graph partitions, based on number of rows/orbitals.

Parameters

gp	Graph partitioning`
nodesPerPart	Number of core nodes per partition
nnodes	Total nodes in Hamiltonian matrix

Definition at line 355 of file prg_graph_mod.F90.

Here is the call graph for this function:



8.8.2.5 prg_filepartition()

```
subroutine, public prg_graph_mod::prg_filepartition (
             type (graph_partitioning_t), intent(inout) gp,
             character(len=*), intent(in) partFile )
```

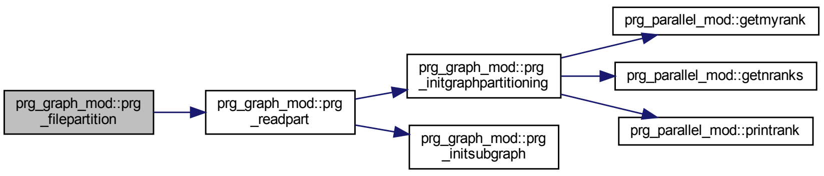
Read graph partitions from a file, based on number of rows/orbitals.

Parameters

partFile	File containing core nodes for each partition
gp	Graph partitioning

Definition at line 463 of file prg_graph_mod.F90.

Here is the call graph for this function:


prg_graph_mod::prg_filepartition
prg_graph_mod::prg_readpart
prg_graph_mod::prg_initgraphpartitioning
prg_graph_mod::prg_initsubgraph
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::printrank

8.8.2.6 prg_fnormgraph()

```
subroutine, public prg_graph_mod::prg_fnormgraph (
             type(graph_partitioning_t), intent(inout) gp )
```

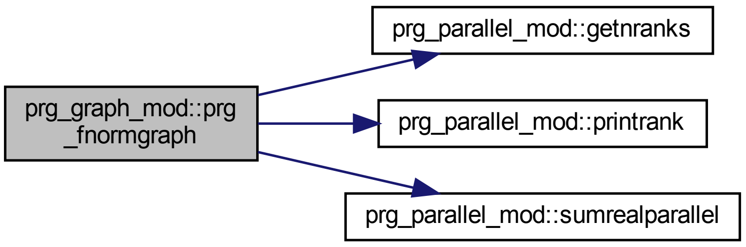
Accumulate trace norm across all subgraphs.

Parameters

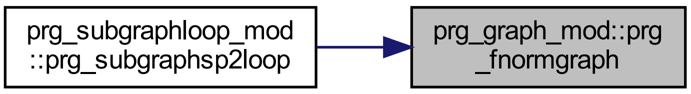
gp	Graph partitioning

Definition at line 516 of file prg_graph_mod.F90.

Here is the call graph for this function:


prg_graph_mod::prg
_fnormgraph
prg_parallel_mod::getnranks
prg_parallel_mod::printrank
prg_parallel_mod::sumrealparallel

Here is the caller graph for this function:



8.8.2.7 prg_initgraphpartitioning()

```
subroutine, public prg_graph_mod::prg_initgraphpartitioning (
            type (graph_partitioning_t), intent(inout) gp,
            character(len=*), intent(in) pname,
            integer, intent(in) np,
            integer, intent(in) nnodes,
            integer, intent(in) nnodes2 )
```

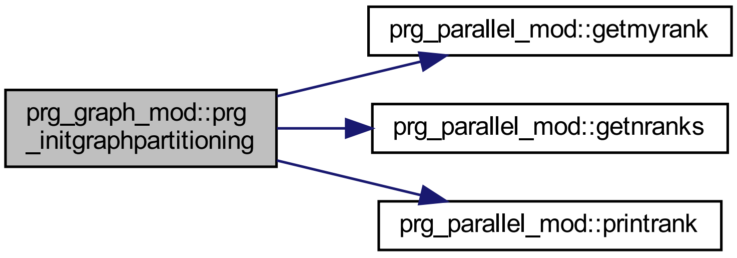
Initialize graph partitioning.

Parameters

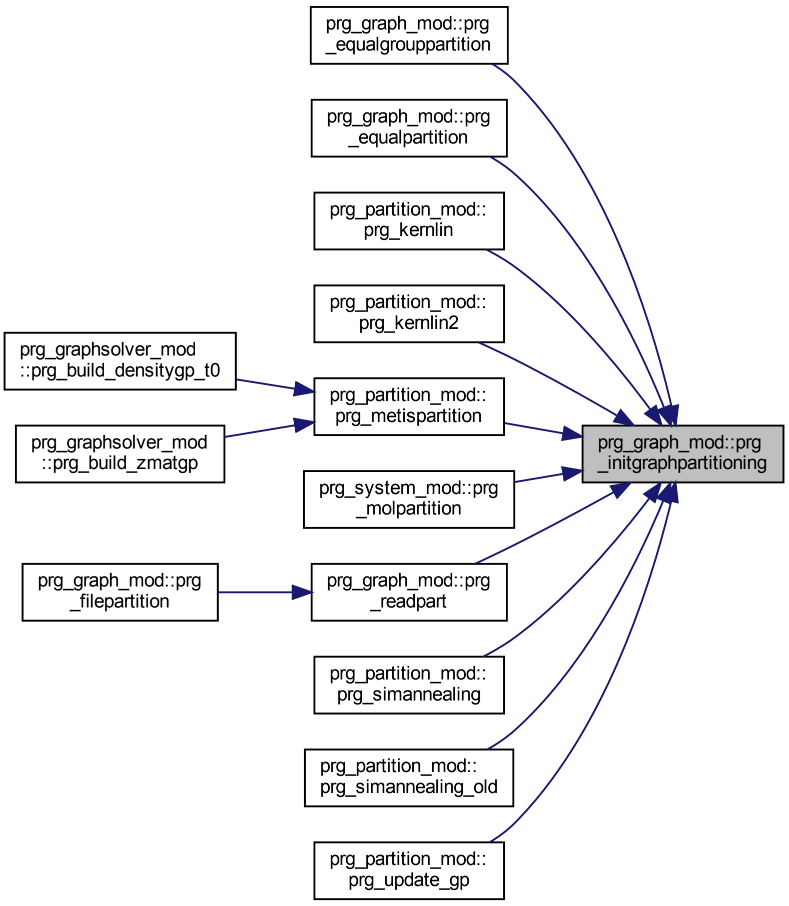
gp	Graph partitioning
pname	Partitioning name
np	Number of partitions
nnodes	Number of groups/nodes
nnodes2	Number of nodes

Definition at line 175 of file prg_graph_mod.F90.

Here is the call graph for this function:


prg_graph_mod::prg _initgraphpartitioning
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::printrank

Here is the caller graph for this function:


prg_graph_mod::prg _equalgrouppartition
prg_graph_mod::prg _equalpartition
prg_partition_mod:: prg_kernlin
prg_partition_mod:: prg_kernlin2
prg_graphsolver_mod ::prg_build_densitygp_t0
prg_partition_mod:: prg_metispartition
prg_graphsolver_mod ::prg_build_zmatgp
prg_graph_mod::prg _initgraphpartitioning
prg_system_mod::prg _molpartition
prg_graph_mod::prg _filepartition
prg_graph_mod::prg _readpart
prg_partition_mod:: prg_simannealing
prg_partition_mod:: prg_simannealing_old
prg_partition_mod:: prg_update_gp

8.8.2.8 prg_initsubgraph()

```
subroutine, public prg_graph_mod::prg_initsubgraph (
            type (subgraph_t), intent(inout) sg,
            integer, intent(in) pnum,
            integer, intent(in) hsize )
```

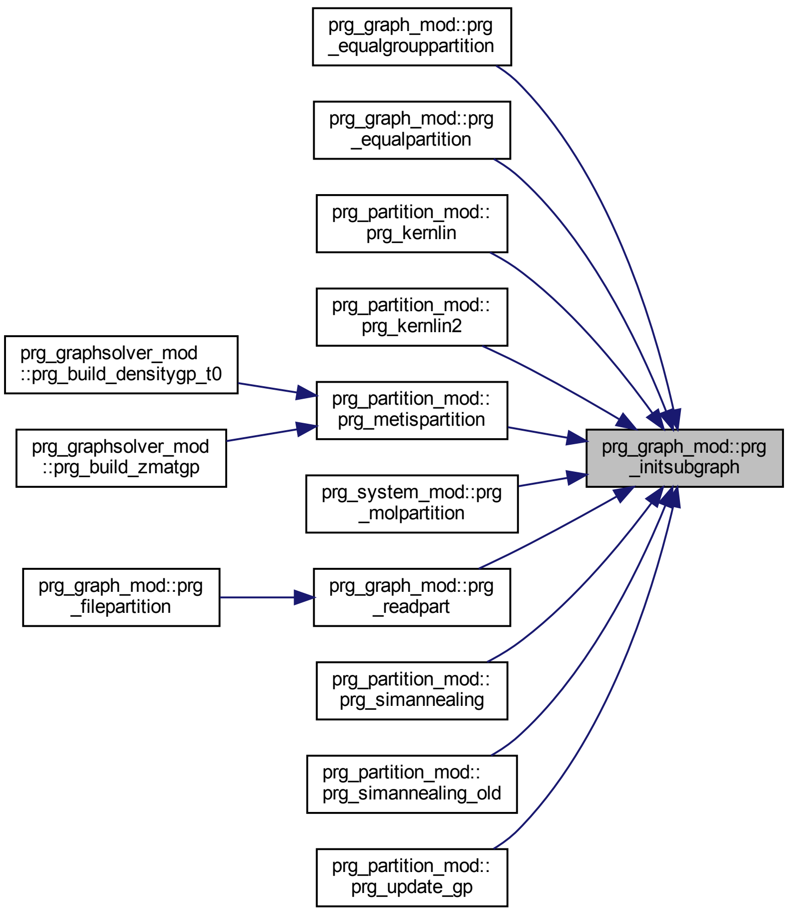
Initialize subgraph.

Parameters

sg	Subgraph
pnum	Part number
hsize	Size of full matrix

Definition at line 143 of file prg_graph_mod.F90.

Here is the caller graph for this function:


prg_graph_mod::prg_equalgrouppartition
prg_graph_mod::prg_equalpartition
prg_partition_mod::prg_kernlin
prg_partition_mod::prg_kernlin2
prg_graphsolver_mod::prg_build_densitygp_t0
prg_partition_mod::prg_metispartition
prg_graphsolver_mod::prg_build_zmatgp
prg_graph_mod::prg_initsubgraph
prg_system_mod::prg_molpartition
prg_graph_mod::prg_filepartition
prg_graph_mod::prg_readpart
prg_partition_mod::prg_simannealing
prg_partition_mod::prg_simannealing_old
prg_partition_mod::prg_update_gp

8.8.2.9 prg_printgraphpartitioning()

```
subroutine, public prg_graph_mod::prg_printgraphpartitioning (
            type (graph_partitioning_t), intent(in) gp )
```

Print graph partitioning structure data.

Parameters

gp	Graph partitioning

Definition at line 292 of file prg_graph_mod.F90.

8.8.2.10 prg_readpart()

```
subroutine prg_graph_mod::prg_readpart (
            type (graph_partitioning_t), intent(inout) gp,
            character(len=*), intent(in) partFile )  [private]
```

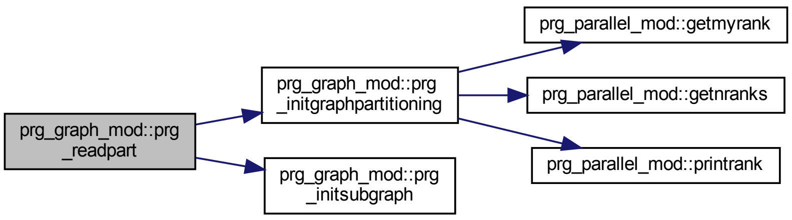
Read parts (core) from part file.

Parameters

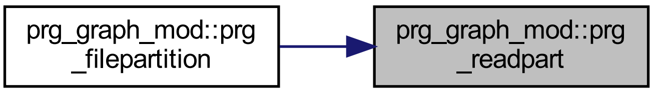
gp	Graph partitioning
partFile	Partition file

Definition at line 475 of file prg_graph_mod.F90.

Here is the call graph for this function:


prg_graph_mod::prg_readpart
prg_graph_mod::prg_initgraphpartitioning
prg_graph_mod::prg_initsubgraph
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::printrank

Here is the caller graph for this function:



8.8.3 Variable Documentation

8.8.3.1 dp

```
integer, parameter prg_graph_mod::dp = kind(1.0d0)  [private]
```

Definition at line 16 of file prg_graph_mod.F90.

8.9 prg_graphsolver_mod Module Reference

Module for graph-based solvers.

Functions/Subroutines

- subroutine, public prg_build_densitygp_t0 (ham_bml, g_bml, rho_bml, threshold, bndfil, Ef, nparts, verbose)

  Builds the density matrix from H_0 using a graph-based approach.
- subroutine, public prg_build_zmatgp (over_bml, g_bml, zmat_bml, threshold, nparts, verbose)

  Builds the inverse overlap factor matrix from S using a graph-based approach.

Variables

- integer, parameter dp = kind(1.0d0)

8.9.1 Detailed Description

Module for graph-based solvers.

8.9.2 Function/Subroutine Documentation

8.9.2.1 prg_build_densitygp_t0()

```
subroutine, public prg_graphsolver_mod::prg_build_densitygp_t0 (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(in) g_bml,
            type(bml_matrix_t), intent(out) rho_bml,
            real(dp) threshold,
            real(dp) bndfil,
            real(dp), intent(inout) Ef,
            integer, intent(inout) nparts,
            integer, intent(in), optional verbose )
```

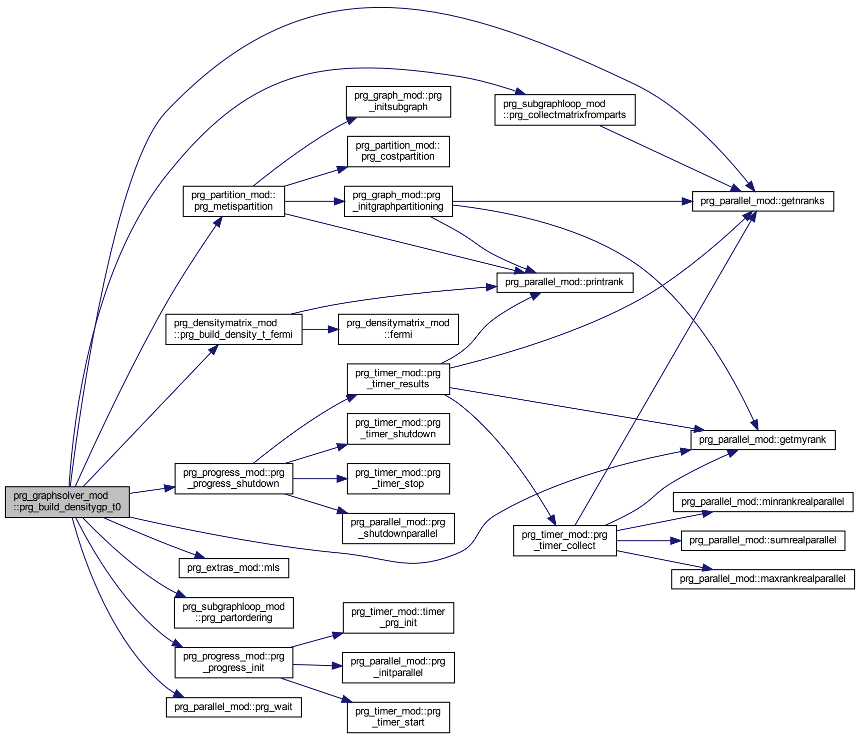
Builds the density matrix from H_0 using a graph-based approach.

Parameters

ham_bml	Input Orthogonalized Hamiltonian matrix.
g_bml	Matrix to extract the graph from.
rho_bml	Density matrix.
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.
Ef	Fermi level or chemical potential.
nparts	Number of parts to be used in graph partitioning.
verbose	Verbosity level.

Definition at line 41 of file prg_graphsolver_mod.F90.

Here is the call graph for this function:


prg_graphsolver_mod ::prg_build_densitygp_t0
prg_graph_mod::prg _initsubgraph
prg_subgraphloop_mod ::prg_collectmatrixfromparts
prg_partition_mod:: prg_costpartition
prg_partition_mod:: prg_metispartition
prg_graph_mod::prg _initgraphpartitioning
prg_parallel_mod::getnranks
prg_parallel_mod::printrank
prg_densitymatrix_mod ::prg_build_density_t_fermi
prg_densitymatrix_mod ::fermi
prg_timer_mod::prg _timer_results
prg_timer_mod::prg _timer_shutdown
prg_parallel_mod::getmyrank
prg_progress_mod::prg _progress_shutdown
prg_timer_mod::prg _timer_stop
prg_parallel_mod::minrankrealparallel
prg_parallel_mod::prg _shutdownparallel
prg_timer_mod::prg _timer_collect
prg_parallel_mod::sumrealparallel
prg_extras_mod::mls
prg_parallel_mod::maxrankrealparallel
prg_subgraphloop_mod ::prg_partordering
prg_timer_mod::timer _prg_init
prg_progress_mod::prg _progress_init
prg_parallel_mod::prg _initparallel
prg_parallel_mod::prg_wait
prg_timer_mod::prg _timer_start

8.9.2.2 prg_build_zmatgp()

```
subroutine, public prg_graphsolver_mod::prg_build_zmatgp (
              type(bml_matrix_t), intent(in) over_bml,
              type(bml_matrix_t), intent(in) g_bml,
              type(bml_matrix_t), intent(out) zmat_bml,
              real(dp) threshold,
              integer, intent(inout) nparts,
              integer, intent(in), optional verbose )
```

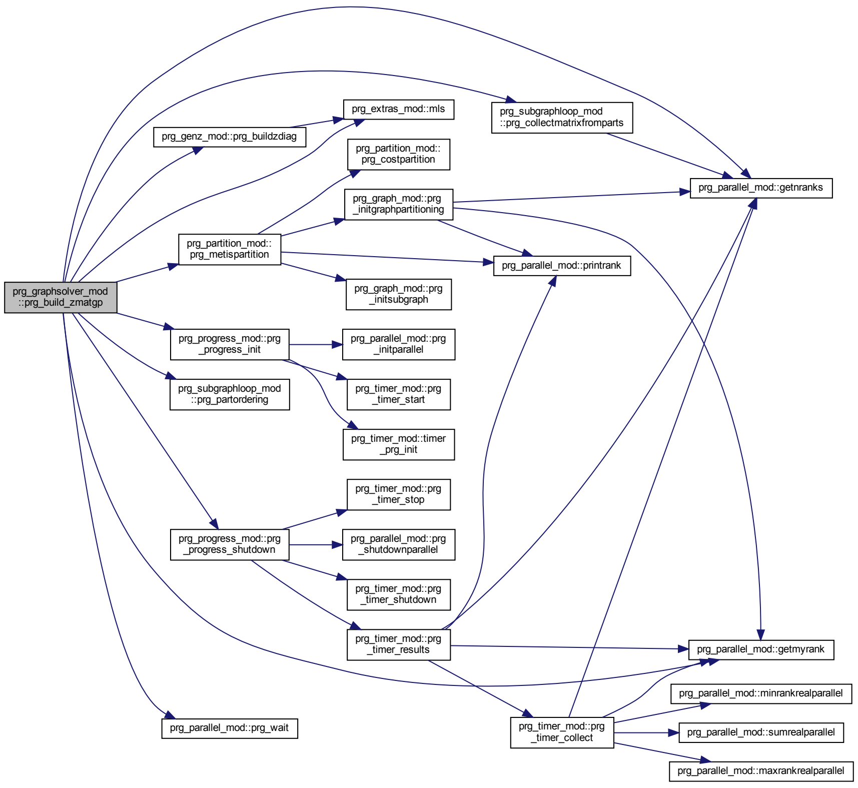
Builds the inverse overlap factor matrix from S using a graph-based approach.

Parameters

over_bml	Input Overlap matrix.
g_bml	Matrix to extract the graph from.
zmat_bml	Overlap matrix.
threshold	Threshold for sparse matrix algebra.
nparts	Number of parts to be used in graph partitioning.
verbose	Verbosity level.

Definition at line 183 of file prg_graphsolver_mod.F90.

Here is the call graph for this function:



8.9.3 Variable Documentation

8.9.3.1 dp

```
integer, parameter prg_graphsolver_mod::dp = kind(1.0d0)  [private]
```

Definition at line 23 of file prg_graphsolver_mod.F90.

8.10 prg_graphsp2parser_mod Module Reference

Graph partitioning SP2 parser.

Data Types

- type gsp2data_type
  General SP2 solver type.

Functions/Subroutines

- subroutine, public prg_parse_gsp2 (gsp2data, filename)
  The parser for SP2 solver.

Variables

- integer, parameter dp = kind(1.0d0)

8.10.1 Detailed Description

Graph partitioning SP2 parser.

This module is used to parse all the neccesary input variables for graph-based SP2 electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.
- Add the keyword (character type) in the keyvector_re vector.
- Add a default value (real type) in the valvector_re.
- Define a new variable and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.

8.10.2 Function/Subroutine Documentation

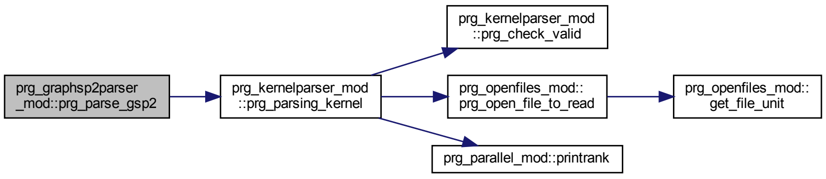
8.10.2.1 prg_parse_gsp2()

```
subroutine, public prg_graphsp2parser_mod::prg_parse_gsp2 (
            type(gsp2data_type), intent(inout) gsp2data,
            character(len=*) filename )
```

The parser for SP2 solver.

Definition at line 62 of file prg_graphsp2parser_mod.F90.

Here is the call graph for this function:


prg_graphsp2parser_mod::prg_parse_gsp2
prg_kernelparser_mod::prg_parsing_kernel
prg_kernelparser_mod::prg_check_valid
prg_openfiles_mod::prg_open_file_to_read
prg_openfiles_mod::get_file_unit
prg_parallel_mod::printrank

8.10.3 Variable Documentation

8.10.3.1 dp

```
integer, parameter prg_graphsp2parser_mod::dp = kind(1.0d0)  [private]
```

Definition at line 22 of file prg_graphsp2parser_mod.F90.

8.11 prg_homolumo_mod Module Reference

The homolumo module.

Functions/Subroutines

- subroutine, public prg_homolumogap (vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)
- subroutine, public prg_sp2sequence (pp, imax, mineval, maxeval, ehomo, elumo, errlimit, verbose)

Variables

- integer, parameter dp = kind(1.0d0)

8.11.1 Detailed Description

The homolumo module.

8.11.2 Function/Subroutine Documentation

8.11.2.1 prg_homolumogap()

```
subroutine, public prg_homolumo_mod::prg_homolumogap (
            real(dp), dimension(:), intent(in) vv,
            integer, intent(in) imax,
            integer, dimension(:), intent(in) pp,
            real(dp), intent(in) mineval,
            real(dp), intent(in) maxeval,
            real(dp), intent(inout) ehomo,
            real(dp), intent(inout) elumo,
            real(dp), intent(inout) egap,
            integer, intent(in), optional verbose )
```

Definition at line 24 of file prg_homolumo_mod.F90.

8.11.2.2 prg_sp2sequence()

```
subroutine, public prg_homolumo_mod::prg_sp2sequence (
            integer, dimension(:), intent(inout) pp,
            integer, intent(inout) imax,
            real(dp), intent(in) mineval,
            real(dp), intent(in) maxeval,
            real(dp), intent(in) ehomo,
            real(dp), intent(in) elumo,
            real(dp), intent(in) errlimit,
            integer, intent(in), optional verbose )
```

Definition at line 98 of file prg_homolumo_mod.F90.

8.11.3 Variable Documentation

8.11.3.1 dp

```
integer, parameter prg_homolumo_mod::dp = kind(1.0d0)  [private]
```

Definition at line 14 of file prg_homolumo_mod.F90.

8.12 prg_implicit_fermi_mod Module Reference

Functions/Subroutines

- subroutine, public prg_implicit_fermi_save_inverse (Inv_bml, h_bml, p_bml, nsteps, nocc, mu, beta, occErr↩ Limit, threshold, tol, SCF_IT, occiter, totns)

  Recursive Implicit Fermi Dirac for finite temperature.
- subroutine, public prg_implicit_fermi (h_bml, p_bml, nsteps, k, nocc, mu, beta, method, osteps, occErrLimit, threshold, tol)

  Recursive Implicit Fermi Dirac for finite temperature.
- subroutine, public prg_implicit_fermi_zero (h_bml, p_bml, nsteps, mu, method, threshold, tol)

  Recursive Implicit Fermi Dirac for zero temperature.
- subroutine, public prg_implicit_fermi_first_order_response (H0_bml, H1_bml, P0_bml, P1_bml, Inv_bml, nsteps, mu0, beta, nocc, threshold)

  Calculate first order density matrix response to perturbations using Implicit Fermi Dirac.
- subroutine, public prg_implicit_fermi_response (H0_bml, H1_bml, H2_bml, H3_bml, P0_bml, P1_bml, P2_↩ bml, P3_bml, nsteps, mu0, mu, beta, nocc, occ_tol, lin_tol, order, threshold)

  Calculate density matrix response to perturbations using Implicit Fermi Dirac.
- subroutine, public prg_finite_diff (H0_bml, H_list, mu0, mu_list, beta, order, lambda, h, threshold)

  Calculate density matrix response from perturbations in the Hamiltonian.
- subroutine prg_setup_linsys (p_bml, A_bml, b_bml, p2_bml, y_bml, aux_bml, aux1_bml, k, threshold)

  Set up linear system for Implicit Fermi Dirac.
- subroutine prg_newtonschulz (a_bml, ai_bml, r_bml, tmp_bml, d_bml, I_bml, tol, threshold, num_iter)

  Find the inverse of the matrix A with Newton-Schulz iteration.
- subroutine prg_pcg (A_bml, p_bml, p2_bml, d_bml, wtmp_bml, cg_tol, threshold)

  Solve the system AX = B with conjugate gradient.
- subroutine prg_conjgrad (A_bml, p_bml, p2_bml, tmp_bml, d_bml, w_bml, cg_tol, threshold)

  Solve the system AX = B with conjugate gradient.
- subroutine prg_get_density_matrix (ham_bml, p_bml, beta, mu, threshold)

  Calculate the density matrix with diagonalization.
- subroutine, public prg_test_density_matrix (ham_bml, p_bml, beta, mu, nocc, osteps, occErrLimit, threshold)

  Calculate the density matrix with diagonalization and converge chemical.
- real(dp) function fermi (e, mu, beta)

  Gives the Fermi distribution value for energy e.

Variables

- integer, parameter dp = kind(1.0d0)

8.12.1 Function/Subroutine Documentation

8.12.1.1 fermi()

```
real(dp) function prg_implicit_fermi_mod::fermi (
            real(dp), intent(in) e,
            real(dp), intent(in) mu,
            real(dp), intent(in) beta )  [private]
```

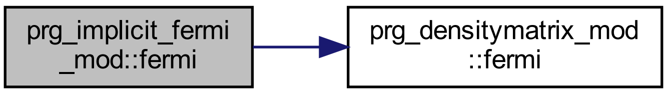
Gives the Fermi distribution value for energy e.

Parameters

e	Energy.
mu	Fermi energy.
beta	Inverse temperature

Definition at line 1193 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:



8.12.1.2 prg_conjgrad()

```
subroutine prg_implicit_fermi_mod::prg_conjgrad (
             type(bml_matrix_t), intent(in) A_bml,
             type(bml_matrix_t), intent(inout) p_bml,
             type(bml_matrix_t), intent(in) p2_bml,
             type(bml_matrix_t), intent(inout) tmp_bml,
             type(bml_matrix_t), intent(inout) d_bml,
             type(bml_matrix_t), intent(inout) w_bml,
             real(dp), intent(in) cg_tol,
             real(dp), intent(in) threshold )  [private]
```

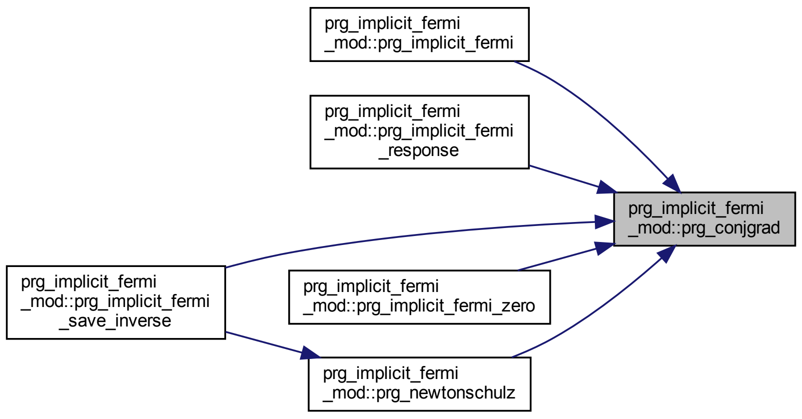
Solve the system AX = B with conjugate gradient.

Parameters

A_bml	Coefficient matrix A
p_bml	Output solution X
p2_bml	Right side matrix B
d_bml	Auxillary matrix
w_bml	Auxillary matrix
cg_tol	Convergence condition (OBS squared Frobenius norm of residual matrix)
threshold	Threshold for matrix algebra

Definition at line 1007 of file prg_implicit_fermi_mod.F90.

Here is the caller graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi
prg_implicit_fermi _mod::prg_implicit_fermi _response
prg_implicit_fermi _mod::prg_conjgrad
prg_implicit_fermi _mod::prg_implicit_fermi _save_inverse
prg_implicit_fermi _mod::prg_implicit_fermi_zero
prg_implicit_fermi _mod::prg_newtonschulz

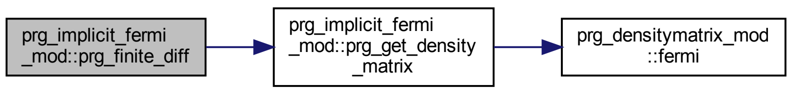
8.12.1.3 prg_finite_diff()

```
subroutine, public prg_implicit_fermi_mod::prg_finite_diff (
            type(bml_matrix_t), intent(in) H0_bml,
            type(bml_matrix_t), dimension(:), intent(in), allocatable H_list,
            real(dp), intent(in) mu0,
            real(dp), dimension(:), intent(in), allocatable mu_list,
            real(dp), intent(in) beta,
            integer, intent(in) order,
            real(dp), intent(in) lambda,
            real(dp), intent(in) h,
            real(dp), intent(in) threshold )
```

Calculate density matrix response from perturbations in the Hamiltonian.

Definition at line 727 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_finite_diff
prg_implicit_fermi _mod::prg_get_density _matrix
prg_densitymatrix_mod ::fermi

8.12.1.4 prg_get_density_matrix()

```
subroutine prg_implicit_fermi_mod::prg_get_density_matrix (
             type(bml_matrix_t), intent(in) ham_bml,
             type(bml_matrix_t), intent(inout) p_bml,
             real(dp), intent(in) beta,
             real(dp), intent(in) mu,
             real(dp), intent(in) threshold )  [private]
```

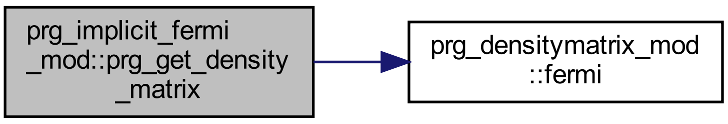
Calculate the density matrix with diagonalization.

Parameters

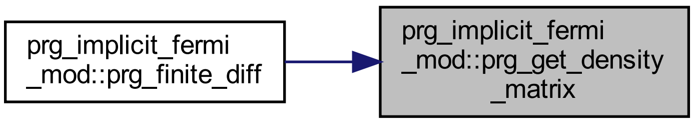
ham_bml	Input hamiltonian
p_bml	Output density matrix
beta	Inverse temperature
mu	Chemical potential
threshold	Threshold for matrix algebra

Definition at line 1057 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi_mod::prg_get_density_matrix
prg_densitymatrix_mod::fermi

Here is the caller graph for this function:


prg_implicit_fermi_mod::prg_finite_diff
prg_implicit_fermi_mod::prg_get_density_matrix

8.12.1.5 prg_implicit_fermi()

```
subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) p_bml,
            integer, intent(in) nsteps,
            integer, intent(in) k,
            real(dp), intent(in) nocc,
            real(dp), intent(inout) mu,
            real(dp), intent(in) beta,
            integer, intent(in) method,
            integer, intent(in) osteps,
            real(dp), intent(in) occErrLimit,
            real(dp), intent(in) threshold,
            real(dp), intent(in) tol )
```

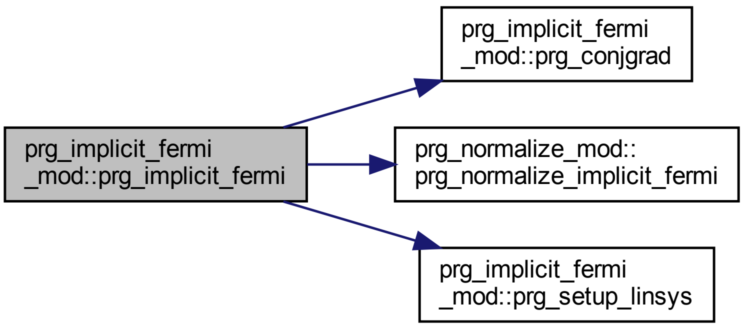
Recursive Implicit Fermi Dirac for finite temperature.

Parameters

h_bml	Input Hamiltonian matrix.
p_bml	Output density matrix.
nsteps	Number of recursion steps.
k	Expansion order
nocc	Number of occupied states.
mu	Shifted chemical potential
beta	Input inverse temperature.
method	0 - conjugate gradient, 1 - newton-schultz
osteps	Outer loop steps to converge chemical potential
occErrLimit	Occupation error limit.
threshold	Threshold for multiplication.
tol	Tolerance for linear system solver See

Definition at line 211 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi
prg_implicit_fermi _mod::prg_conjgrad
prg_normalize_mod:: prg_normalize_implicit_fermi
prg_implicit_fermi _mod::prg_setup_linsys

8.12.1.6 prg_implicit_fermi_first_order_response()

```
subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_first_order_response (
            type(bml_matrix_t), intent(in) H0_bml,
            type(bml_matrix_t), intent(in) H1_bml,
            type(bml_matrix_t), intent(inout) P0_bml,
            type(bml_matrix_t), intent(inout) P1_bml,
            type(bml_matrix_t), dimension(nsteps), intent(in) Inv_bml,
            integer, intent(in) nsteps,
            real(dp), intent(in) mu0,
            real(dp), intent(in) beta,
            real(dp), intent(in) nocc,
            real(dp), intent(in) threshold )
```

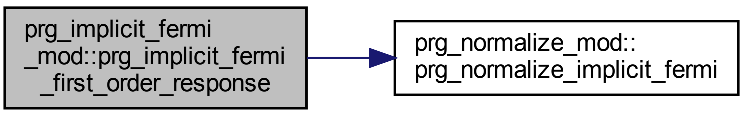
Calculate first order density matrix response to perturbations using Implicit Fermi Dirac.

Parameters

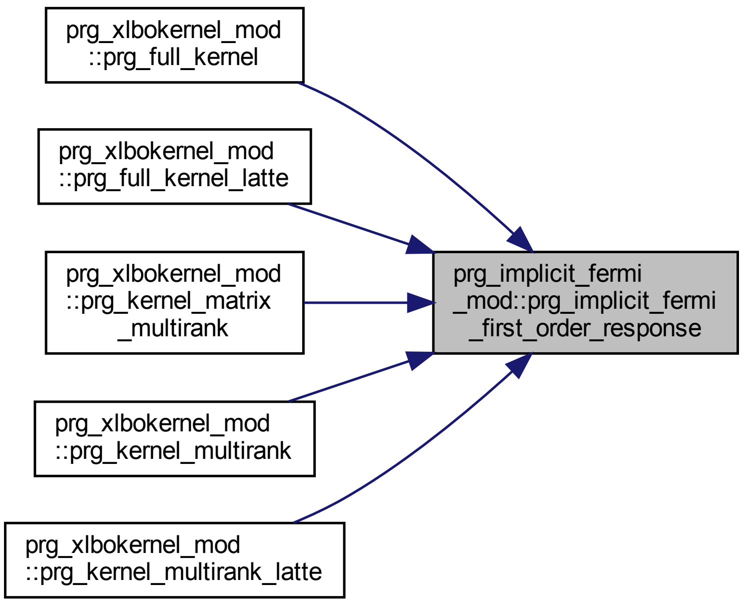
H0_bml	Input Hamiltonian matrix.
H1_bml	Input First order perturbation of H0.
P0_bml	Output density matrix.
P1_bml	Output First order density matrix response.
nsteps	Number of recursion steps.
mu0	Shifted chemical potential.
beta	Input inverse temperature.
nocc	Number of occupied states.
threshold	Threshold for matrix algebra. See

Definition at line 441 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi _first_order_response
prg_normalize_mod:: prg_normalize_implicit_fermi

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_full_kernel
prg_xlbokernel_mod ::prg_full_kernel_latte
prg_xlbokernel_mod ::prg_kernel_matrix _multirank
prg_xlbokernel_mod ::prg_kernel_multirank
prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_implicit_fermi _mod::prg_implicit_fermi _first_order_response

8.12.1.7 prg_implicit_fermi_response()

```
subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_response (
            type(bml_matrix_t), intent(in) H0_bml,
            type(bml_matrix_t), intent(in) H1_bml,
            type(bml_matrix_t), intent(in) H2_bml,
            type(bml_matrix_t), intent(in) H3_bml,
            type(bml_matrix_t), intent(inout) P0_bml,
            type(bml_matrix_t), intent(inout) P1_bml,
            type(bml_matrix_t), intent(inout) P2_bml,
            type(bml_matrix_t), intent(inout) P3_bml,
            integer, intent(in) nsteps,
            real(dp), intent(inout) mu0,
            real(dp), dimension(:), intent(inout), allocatable mu,
            real(dp), intent(in) beta,
            real(dp), intent(in) nocc,
            real(dp), intent(in) occ_tol,
            real(dp), intent(in) lin_tol,
            integer order,
            real(dp) threshold )
```

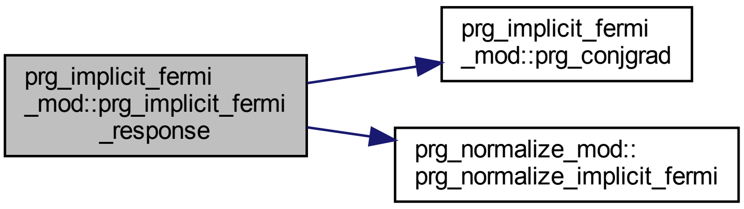
Calculate density matrix response to perturbations using Implicit Fermi Dirac.

Parameters

H0_bml	Input Hamiltonian matrix.
H1_bml,H2_bml,H3_bml	Input First to third order perturbations of H0.
P0_bml	Output density matrix.
P1_bml,P2_bml,P3_bml	Output First to third order density matrix response.
nsteps	Number of recursion steps.
mu0	Shifted chemical potential.
mu	Pre-allocated array of length order.
beta	Input inverse temperature.
nocc	Number of occupied states.
occ_tol	Occupation error tolerance.
lin_tol	Linear solver tolerance.
order	Calculate response up to this order.
threshold	Threshold for matrix algebra. See

Definition at line 549 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi _response
prg_implicit_fermi _mod::prg_conjgrad
prg_normalize_mod:: prg_normalize_implicit_fermi

8.12.1.8 prg_implicit_fermi_save_inverse()

```
subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_save_inverse (
            type(bml_matrix_t), dimension(nsteps), intent(inout) Inv_bml,
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) p_bml,
            integer, intent(in) nsteps,
            real(dp), intent(in) nocc,
            real(dp), intent(inout) mu,
            real(dp), intent(in) beta,
            real(dp), intent(in) occErrLimit,
            real(dp), intent(in) threshold,
            real(dp), intent(in) tol,
            integer, intent(in) SCF_IT,
            integer, intent(inout) occiter,
            integer, intent(inout) totns )
```

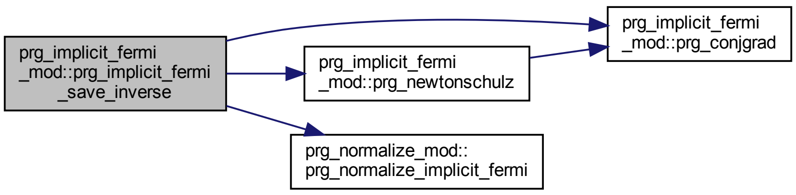
Recursive Implicit Fermi Dirac for finite temperature.

Parameters

Inv_bml	Inverses generated by algorithm.
h_bml	Input Hamiltonian matrix.
p_bml	Output density matrix.
nsteps	Number of recursion steps.
nocc	Number of occupied states.
mu	Shifted chemical potential
beta	Input inverse temperature.
occErrLimit	Occupation error limit.
threshold	Threshold for multiplication.
tol	Tolerance for linear system solver.
SCF_IT	The current SCF iteration.
occiter	Counts the total nr of DM calculations during MD. See

Definition at line 48 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi _save_inverse
prg_implicit_fermi _mod::prg_conjgrad
prg_implicit_fermi _mod::prg_newtonschulz
prg_normalize_mod:: prg_normalize_implicit_fermi

8.12.1.9 prg_implicit_fermi_zero()

```
subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_zero (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) p_bml,
            integer, intent(in) nsteps,
            real(dp), intent(in) mu,
            integer, intent(in) method,
            real(dp), intent(in) threshold,
            real(dp), intent(inout), optional tol )
```

Recursive Implicit Fermi Dirac for zero temperature.

Parameters

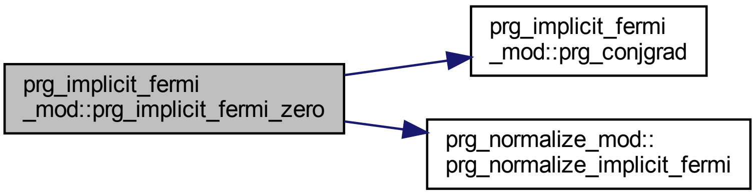
h_bml	Input Hamiltonian matrix.
p_bml	Output density matrix.
nsteps	Number of recursion steps.

Parameters

mu	Shifted chemical potential
beta	Input inverse temperature.
method	0 - conjugate gradient, 1 - newton-schultz
threshold	Threshold for multiplication.
tol	Tolerance for linear system solver

Definition at line 348 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi_zero
prg_implicit_fermi _mod::prg_conjgrad
prg_normalize_mod:: prg_normalize_implicit_fermi

8.12.1.10 prg_newtonschulz()

```
subroutine prg_implicit_fermi_mod::prg_newtonschulz (
             type(bml_matrix_t), intent(in) a_bml,
             type(bml_matrix_t), intent(inout) ai_bml,
             type(bml_matrix_t), intent(inout) r_bml,
             type(bml_matrix_t), intent(inout) tmp_bml,
             type(bml_matrix_t), intent(inout) d_bml,
             type(bml_matrix_t), intent(in) I_bml,
             real(dp), intent(in) tol,
             real(dp), intent(in) threshold,
             integer, intent(out) num_iter )  [private]
```

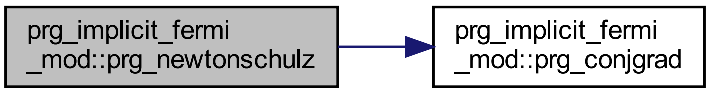
Find the inverse of the matrix A with Newton-Schulz iteration.

Parameters

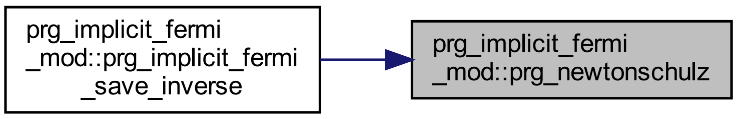
a_bml	Input matrix A
ai_bml	Input starting guess and output inverse
r_bml	Auxillary matrix
tmp_bml	Auxillary matrix
tol	Convergence criterion (Frobenius norm of residual matrix)
threshold	Threshold for matrix algebra

Definition at line 889 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_newtonschulz
prg_implicit_fermi _mod::prg_conjgrad

Here is the caller graph for this function:


prg_implicit_fermi _mod::prg_implicit_fermi _save_inverse
prg_implicit_fermi _mod::prg_newtonschulz

8.12.1.11 prg_pcg()

```
subroutine prg_implicit_fermi_mod::prg_pcg (
             type(bml_matrix_t), intent(in) A_bml,
             type(bml_matrix_t), intent(inout) p_bml,
             type(bml_matrix_t), intent(inout) p2_bml,
             type(bml_matrix_t), intent(inout) d_bml,
             type(bml_matrix_t), intent(inout) wtmp_bml,
             real(dp), intent(in) cg_tol,
             real(dp), intent(in) threshold )  [private]
```

Solve the system AX = B with conjugate gradient.

Parameters

A_bml	Coefficient matrix A
p_bml	Output solution X
p2_bml	Right side matrix B
d_bml	Auxillary matrix
w_bml	Auxillary matrix
cg_tol	Convergence condition (OBS squared Frobenius norm of residual matrix)
threshold	Threshold for matrix algebra

Definition at line 935 of file prg_implicit_fermi_mod.F90.

8.12.1.12 prg_setup_linsys()

```
subroutine prg_implicit_fermi_mod::prg_setup_linsys (
             type(bml_matrix_t), intent(in) p_bml,
             type(bml_matrix_t), intent(inout) A_bml,
             type(bml_matrix_t), intent(inout) b_bml,
             type(bml_matrix_t), intent(inout) p2_bml,
             type(bml_matrix_t), intent(inout) y_bml,
             type(bml_matrix_t), intent(inout) aux_bml,
             type(bml_matrix_t), intent(inout) aux1_bml,
             integer, intent(in) k,
             real(dp), intent(in) threshold )  [private]
```

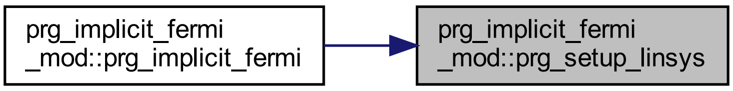
Set up linear system for Implicit Fermi Dirac.

Parameters

p_bml	Input X_i matrix.
p2_bml	Output X_i^k matrix.
A_bml	Output [X_i^k + (I - X_i)^k] matrix.
y_bml	Auxillary matrix.
aux_bml	Auxillary matrix
aux1_bml	Auxillary matrix.
k	Expansion order (an even number)
threshold	Threshold for multiplication. OBS this routine can be numerically unstable for k $>$ 4

Definition at line 845 of file prg_implicit_fermi_mod.F90.

Here is the caller graph for this function:



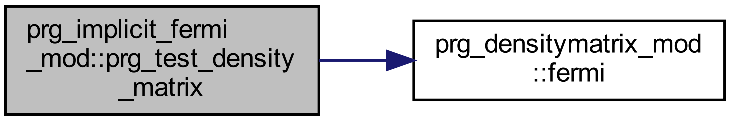
8.12.1.13 prg_test_density_matrix()

```
subroutine, public prg_implicit_fermi_mod::prg_test_density_matrix (
             type(bml_matrix_t), intent(in) ham_bml,
             type(bml_matrix_t), intent(inout) p_bml,
             real(dp), intent(in) beta,
             real(dp), intent(inout) mu,
             real(dp), intent(in) nocc,
             integer, intent(in) osteps,
             real(dp), intent(in) occErrLimit,
             real(dp), intent(in) threshold )
```

Calculate the density matrix with diagonalization and converge chemical.

Definition at line 1113 of file prg_implicit_fermi_mod.F90.

Here is the call graph for this function:


prg_implicit_fermi _mod::prg_test_density _matrix
prg_densitymatrix_mod ::fermi

8.12.2 Variable Documentation

8.12.2.1 dp

```
integer, parameter prg_implicit_fermi_mod::dp = kind(1.0d0)  [private]
```

Definition at line 20 of file prg_implicit_fermi_mod.F90.

8.13 prg_initmatrices_mod Module Reference

Initialization module.

Functions/Subroutines

- subroutine, public prg_init_hsmat (ham_bml, over_bml, bml_type, mdim, norb)

  Initialize Hamiltonian and Overlap Matrix.
- subroutine, public prg_init_pzmat (rho_bml, zmat_bml, bml_type, mdim, norb)

  Initialize Density matrix and Inverse square root Overlap.
- subroutine, public prg_init_ortho (orthoh_bml, orthop_bml, bml_type, mdim, norb)

  Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Variables

- integer, parameter dp = kind(1.0d0)

8.13.1 Detailed Description

Initialization module.

Routines in this module are used to initialize several matrices that will be used in the code.

8.13.2 Function/Subroutine Documentation

8.13.2.1 prg_init_hsmat()

```
subroutine, public prg_initmatrices_mod::prg_init_hsmat (
            type(bml_matrix_t), intent(inout) ham_bml,
            type(bml_matrix_t), intent(inout) over_bml,
            character(20) bml_type,
            integer, intent(inout) mdim,
            integer, intent(in) norb )
```

Initialize Hamiltonian and Overlap Matrix.

Allocation of the Hamiltonian and Overlap matrix into bml formats.

Parameters

ham_bml	Hamiltonian in bml format.
over_bml	Overlap in bml format.
threshold	Threshold value for matrix elements.
mdim	Max nonzero elements per row for every row see [1] .
norb	Total number of orbitals.

Definition at line 29 of file prg_initmatrices_mod.F90.

8.13.2.2 prg_init_ortho()

```
subroutine, public prg_initmatrices_mod::prg_init_ortho (
            type(bml_matrix_t), intent(inout) orthoh_bml,
            type(bml_matrix_t), intent(inout) orthop_bml,
            character(20) bml_type,
            integer, intent(inout) mdim,
            integer, intent(in) norb )
```

Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Allocation of the orthogonal Hamiltonian and Density matrix into bml formats.

Parameters

orthoh_bml	Orthogonal Hamiltonian in bml format.
orthop_bml	Orthogonal Density Matrix in bml format.
threshold	Threshold value for matrix elements.
mdim	Max nonzero elements per row for every row see [1] .
norb	Total number of orbitals.

Definition at line 73 of file prg_initmatrices_mod.F90.

8.13.2.3 prg_init_pzmat()

```
subroutine, public prg_initmatrices_mod::prg_init_pzmat (
            type(bml_matrix_t), intent(inout) rho_bml,
            type(bml_matrix_t), intent(inout) zmat_bml,
            character(20) bml_type,
            integer, intent(inout) mdim,
            integer, intent(in) norb )
```

Initialize Density matrix and Inverse square root Overlap.

Allocation of the Density matrix and Inverse square root Overlap matrix into bml formats.

Parameters

rho_bml	Density matrix in bml format.
zmat_bml	Inverse square root Overlap in bml format.
threshold	Threshold value for matrix elements.
mdim	Max nonzero elements per row for every row see [1] .
norb	Total number of orbitals.

Definition at line 51 of file prg_initmatrices_mod.F90.

8.13.3 Variable Documentation

8.13.3.1 dp

```
integer, parameter prg_initmatrices_mod::dp = kind(1.0d0)  [private]
```

Definition at line 14 of file prg_initmatrices_mod.F90.

8.14 prg_kernelparser_mod Module Reference

Some general parsing functions.

Functions/Subroutines

- subroutine, public prg_parsing_kernel (keyvector_char, valvector_char, keyvector_int, valvector_int, keyvector_re, valvector_re, keyvector_log, valvector_log, filename, startstop)

    The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.
- subroutine prg_check_valid (invalidc)

    Check for valid keywords (checks for an = sign)

Variables

- integer, parameter dp = kind(1.0d0)

8.14.1 Detailed Description

Some general parsing functions.

8.14.2 Function/Subroutine Documentation

8.14.2.1 prg_check_valid()

```
subroutine prg_kernelparser_mod::prg_check_valid (
             character(len=*), intent(in) invalidc )  [private]
```

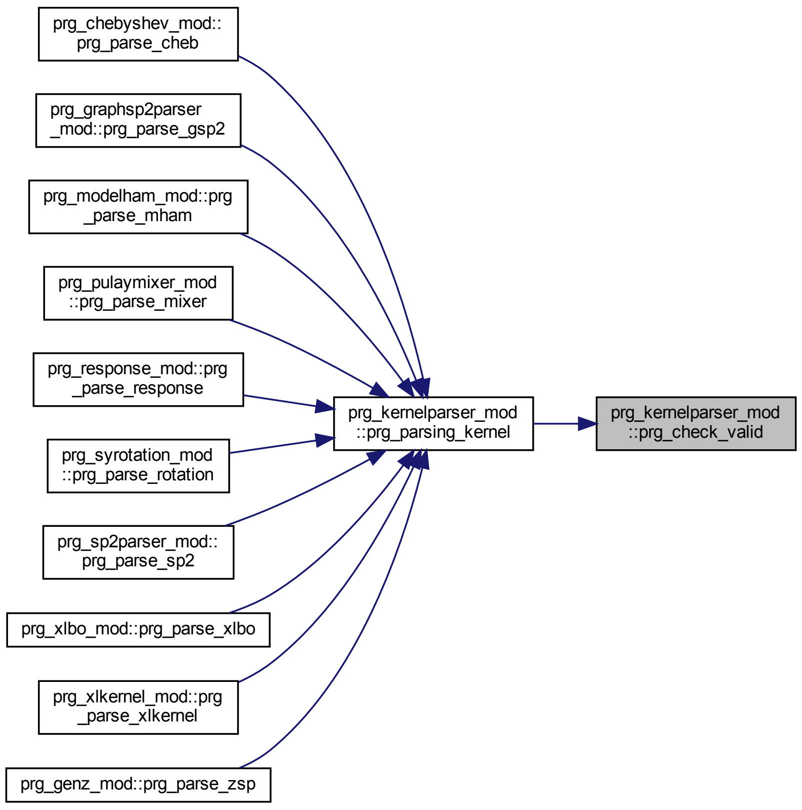
Check for valid keywords (checks for an = sign)

Parameters

invalidc	Keyword to check.

Definition at line 393 of file prg_kernelparser_mod.F90.

Here is the caller graph for this function:



8.14.2.2 prg_parsing_kernel()

```
subroutine, public prg_kernelparser_mod::prg_parsing_kernel (
             character(50), dimension(:)  keyvector_char,
             character(100), dimension(:)  valvector_char,
             character(50), dimension(:)  keyvector_int,
             integer, dimension(:)  valvector_int,
             character(50), dimension(:)  keyvector_re,
             real(dp), dimension(:)  valvector_re,
             character(50), dimension(:)  keyvector_log,
             logical, dimension(:)  valvector_log,
             character(len=*) filename,
             character(len=*), dimension(2), intent(in), optional startstop )
```

The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.

Note

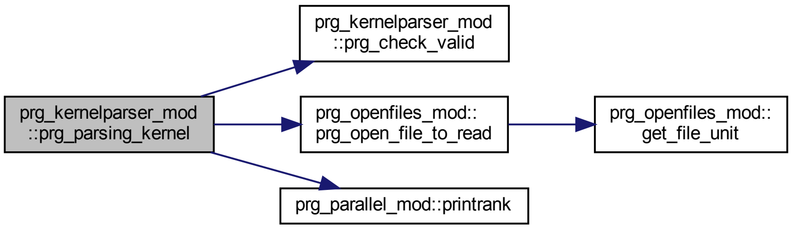
This parsing strategy can only parse a file of 500 lines by 500 words.

Warning

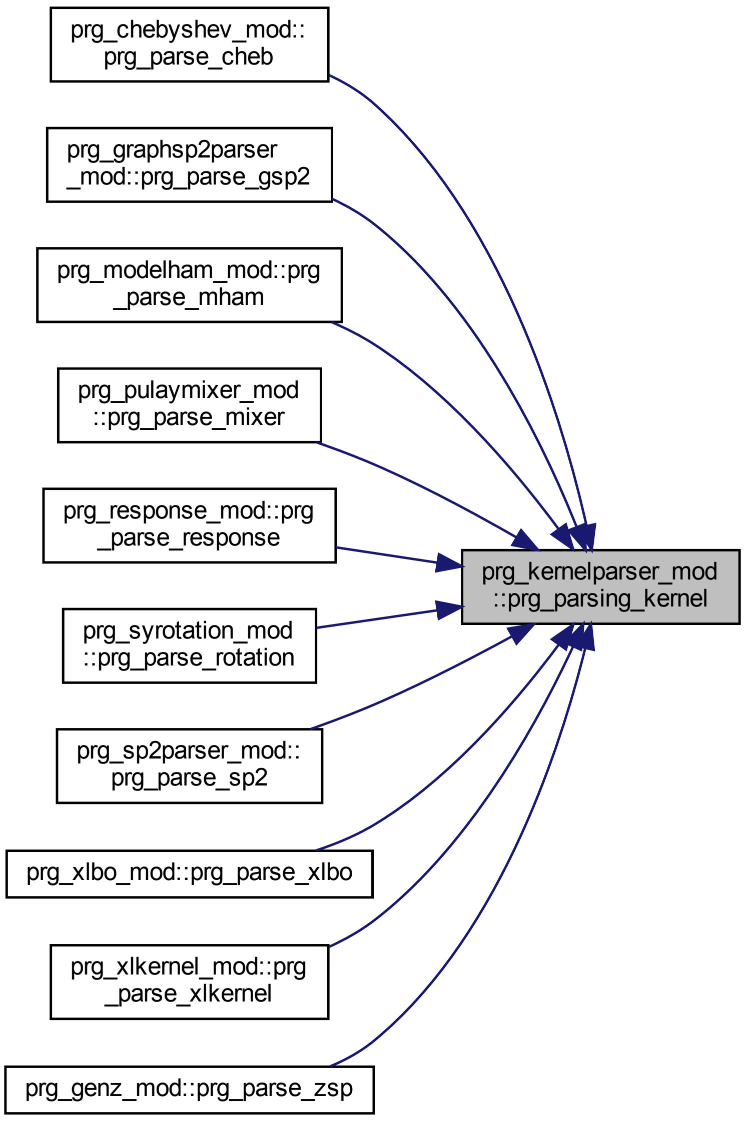
If the length of variable vect is changed, this could produce a segmentation fault.

Definition at line 30 of file prg_kernelparser_mod.F90.

Here is the call graph for this function:


prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

Here is the caller graph for this function:


prg_chebyshev_mod::prg_parse_cheb
prg_graphsp2parser_mod::prg_parse_gsp2
prg_modelham_mod::prg_parse_mham
prg_pulaymixer_mod::prg_parse_mixer
prg_response_mod::prg_parse_response
prg_syrotation_mod::prg_parse_rotation
prg_sp2parser_mod::prg_parse_sp2
prg_xlbo_mod::prg_parse_xlbo
prg_xlkernel_mod::prg_parse_xlkernel
prg_genz_mod::prg_parse_zsp
prg_kernelparser_mod::prg_parsing_kernel

8.14.3 Variable Documentation

8.14.3.1 dp

```
integer, parameter prg_kernelparser_mod::dp = kind(1.0d0)  [private]
```

Definition at line 13 of file prg_kernelparser_mod.F90.

8.15 prg_modelham_mod Module Reference

The prg_hamiltonian module.

Data Types

- type mham_type

  General ModelHam type.

Functions/Subroutines

- subroutine, public prg_parse_mham (mham, filename)

  Model Ham parse.
- subroutine, public prg_twolevel_model (ea, eb, dab, daiaj, dbibj, dec, rcoeff, reshuffle, seed, h_bml, verbose)

  Construct a two-level model Hamiltonian.

Variables

- integer, parameter dp = kind(1.0d0)

8.15.1 Detailed Description

The prg_hamiltonian module.

This module will create a model Hamiltonian for benchmarking purposes.

8.15.2 Function/Subroutine Documentation

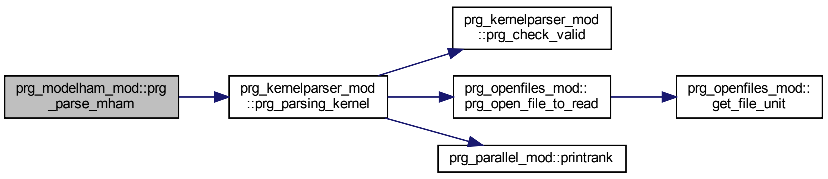
8.15.2.1 prg_parse_mham()

```
subroutine, public prg_modelham_mod::prg_parse_mham (
            type(mham_type), intent(inout) mham,
            character(len=*) filename )
```

Model Ham parse.

Definition at line 37 of file prg_modelham_mod.F90.

Here is the call graph for this function:


prg_modelham_mod::prg_parse_mham
prg_kernelparser_mod::prg_parsing_kernel
prg_kernelparser_mod::prg_check_valid
prg_openfiles_mod::prg_open_file_to_read
prg_openfiles_mod::get_file_unit
prg_parallel_mod::printrank

8.15.2.2 prg_twolevel_model()

```
subroutine, public prg_modelham_mod::prg_twolevel_model (
             real(dp), intent(in) ea,
             real(dp), intent(in) eb,
             real(dp), intent(in) dab,
             real(dp), intent(in) daiaj,
             real(dp), intent(in) dbibj,
             real(dp) dec,
             real(dp), intent(in) rcoeff,
             logical, intent(in) reshuffle,
             integer, intent(in) seed,
             type(bml_matrix_t), intent(inout) h_bml,
             integer, intent(in) verbose )
```

Construct a two-level model Hamiltonian.

Parameters

ea	First onsite energy
eb	Second onsite energy
dab	Onsite Hamiltonian element
daiaj	Intersite first level Hamiltonian elements
dbibj	Intersite second level Hamiltonian elements
dec	Decay constant
rcoeff	Random coefficient
reshuffle	If rows needs to be reshuffled
seed	Random seed
h_bml	Output hamiltonian matrix
verbose	Verbosity level

Definition at line 120 of file prg_modelham_mod.F90.

8.15.3 Variable Documentation

8.15.3.1 dp

```
integer, parameter prg_modelham_mod::dp = kind(1.0d0)  [private]
```

Definition at line 15 of file prg_modelham_mod.F90.

8.16 prg_nonortho_mod Module Reference

Module to prg_orthogonalize and prg_deorthogonalize any operator.

Functions/Subroutines

- subroutine, public prg_orthogonalize (A_bml, zmat_bml, orthoA_bml, threshold, bml_type, verbose)

  This routine performs: $A_{ortho} = Z^{\dagger} A Z$.
- subroutine, public prg_deorthogonalize (orthoA_bml, zmat_bml, a_bml, threshold, bml_type, verbose)

  This routine performs: $A = Z A_{ortho} Z^{\dagger}$.

Variables

- integer, parameter dp = kind(1.0d0)

8.16.1 Detailed Description

Module to prg_orthogonalize and prg_deorthogonalize any operator.

Typically the Hamiltonin needs to be prg_orthogonalized: $H_{\mathrm{ortho}} = Z^{\dagger} H Z$

Also, if the density matrix was obtained from the prg_orthogonalized Hamiltonian, it can be prg_deorthogonalized as: $\rho = Z \rho_{\mathrm{ortho}} Z^{\dagger}$

8.16.2 Function/Subroutine Documentation

8.16.2.1 prg_deorthogonalize()

```
subroutine, public prg_nonortho_mod::prg_deorthogonalize (
              type(bml_matrix_t), intent(in) orthoA_bml,
              type(bml_matrix_t), intent(in) zmat_bml,
              type(bml_matrix_t), intent(inout) a_bml,
              real(dp) threshold,
              character(len=*) bml_type,
              integer verbose )
```

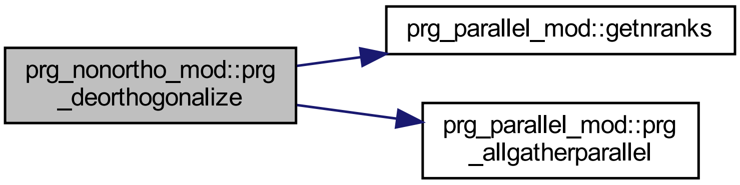
This routine performs: $A = Z A_{ortho} Z^{\dagger}$.

Parameters

orthoA_bml	Matrix to be prg_deorthogonalized.
zmat_bml	Congruence transform to be used.
A_bml	Matrix resulting from the prg_deorthogonalized in bml format.
threshold	Threshold value to be used in the matrix-matrix operations.
bml_type	bml format to be used.
verbose	Verbosity level.

Definition at line 82 of file prg_nonortho_mod.F90.

Here is the call graph for this function:


prg_nonortho_mod::prg_deorthogonalize
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel

8.16.2.2 prg_orthogonalize()

```
subroutine, public prg_nonortho_mod::prg_orthogonalize (
            type(bml_matrix_t), intent(inout) A_bml,
            type(bml_matrix_t), intent(inout) zmat_bml,
            type(bml_matrix_t), intent(inout) orthoA_bml,
            real(dp), intent(in) threshold,
            character(len=*), intent(in) bml_type,
            integer, intent(in) verbose )
```

This routine performs: $A_{ortho} = Z^{\dagger}AZ$.

Parameters

A_bml	Matrix to be prg_orthogonalized in bml format.
zmat_bml	Congruence transform to be used.
orthoA_bml	Matrix resulting from the orthogonalization.
threshold	Threshold value to be used in the matrix-matrix operations.
bml_type	bml format to be used.
verbose	Verbosity level.

Definition at line 36 of file prg_nonortho_mod.F90.

8.16.3 Variable Documentation

8.16.3.1 dp

```
integer, parameter prg_nonortho_mod::dp = kind(1.0d0)  [private]
```

Definition at line 19 of file prg_nonortho_mod.F90.

8.17 prg_normalize_mod Module Reference

The prg_normalize module.

Functions/Subroutines

- subroutine, public prg_normalize (h_bml)

  Normalize a Hamiltonian matrix prior to running the SP2 algorithm.
- subroutine, public prg_normalize_fermi (h_bml, h1, hN, mu)

  Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.
- subroutine, public prg_normalize_implicit_fermi (h_bml, cnst, mu)

  Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.
- subroutine, public prg_gershgorinreduction (gp)

  Determine gershgorin bounds across all parts, local and distributed.
- subroutine, public prg_normalize_cheb (h_bml, mu, emin, emax, alpha, scaledmu)

  Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

Variables

- integer, parameter dp = kind(1.0d0)

8.17.1 Detailed Description

The prg_normalize module.

8.17.2 Function/Subroutine Documentation

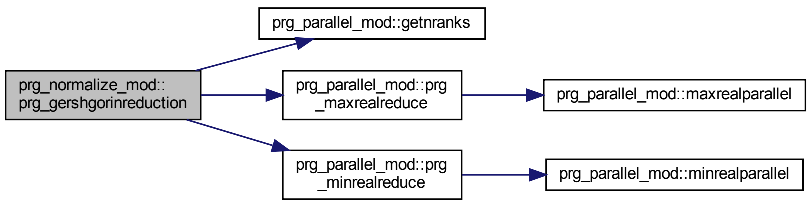
8.17.2.1 prg_gershgorinreduction()

```
subroutine, public prg_normalize_mod::prg_gershgorinreduction (
            type(graph_partitioning_t), intent(inout) gp )
```

Determine gershgorin bounds across all parts, local and distributed.

Definition at line 98 of file prg_normalize_mod.F90.

Here is the call graph for this function:



8.17.2.2 prg_normalize()

```
subroutine, public prg_normalize_mod::prg_normalize (
            type(bml_matrix_t), intent(inout) h_bml )
```

Normalize a Hamiltonian matrix prior to running the SP2 algorithm.

X0 = (e_max ∗ I - H) / (e_max - e_min)

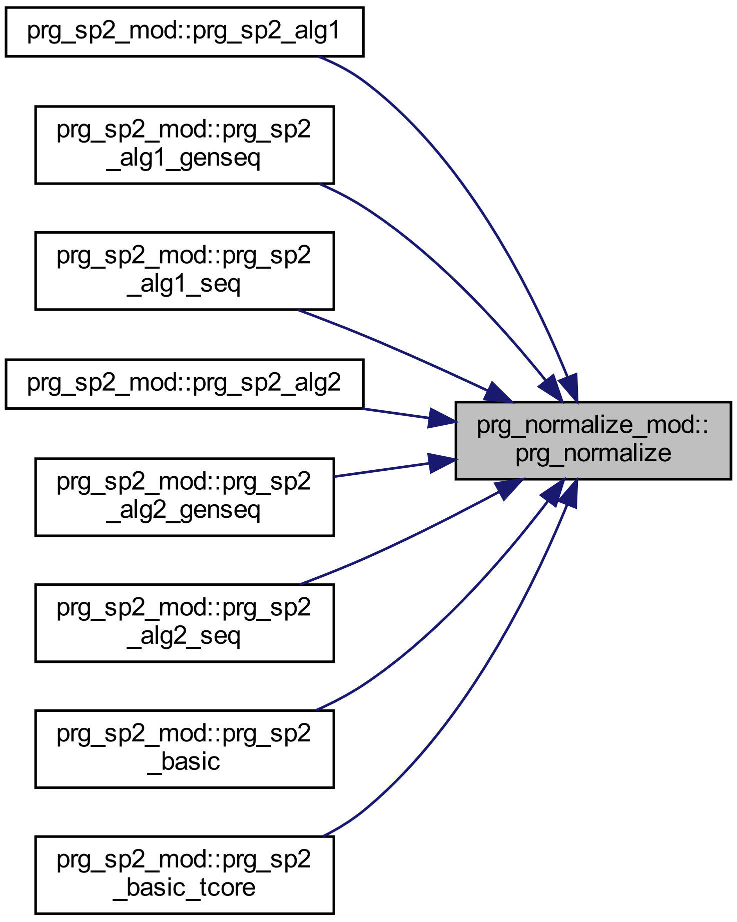
where e_max and e_min are obtained sing the Gershgorin circle theorem.

Parameters

h_bml	Input/Output Hamiltonian matrix

Definition at line 33 of file prg_normalize_mod.F90.

Here is the caller graph for this function:


prg_sp2_mod::prg_sp2_alg1
prg_sp2_mod::prg_sp2_alg1_genseq
prg_sp2_mod::prg_sp2_alg1_seq
prg_sp2_mod::prg_sp2_alg2
prg_sp2_mod::prg_sp2_alg2_genseq
prg_sp2_mod::prg_sp2_alg2_seq
prg_sp2_mod::prg_sp2_basic
prg_sp2_mod::prg_sp2_basic_tcore
prg_normalize_mod::prg_normalize

8.17.2.3 prg_normalize_cheb()

```
subroutine, public prg_normalize_mod::prg_normalize_cheb (
            type(bml_matrix_t), intent(inout) h_bml,
            real(dp), intent(in) mu,
            real(dp), intent(in) emin,
            real(dp), intent(in) emax,
            real(dp), intent(inout) alpha,
            real(dp), intent(inout) scaledmu )
```

Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

X0 = 2∗(H - e_min∗I) / (e_max - e_min) - I

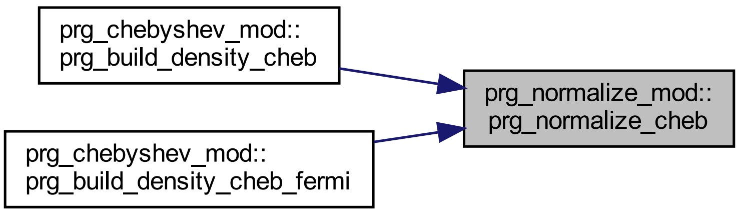
where e_max and e_min are obtained sing the Gershgorin circle theorem.

Parameters

h_bml	Input/Output Hamiltonian matrix

Definition at line 127 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



8.17.2.4 prg_normalize_fermi()

```
subroutine, public prg_normalize_mod::prg_normalize_fermi (
            type(bml_matrix_t), intent(inout) h_bml,
            real(dp), intent(in) h1,
            real(dp), intent(in) hN,
            real(dp), intent(in) mu )
```

Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.

X0 = ((hN-mu) ∗ I - H) / (hN - h1) or X0 = (hN∗I-H0-mu∗I)/(hN-h1)

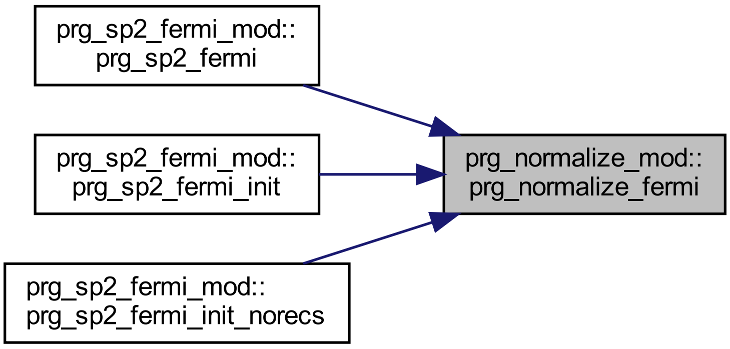
where h1 and hN are scaled Gershgorin bounds.

Parameters

H_bml	Hamiltonian matrix
h1	Scaled minimum Gershgorin bound.
hN	Scaled maximum Gershgorin bound.
mu	Chemical potential

Definition at line 60 of file prg_normalize_mod.F90.

Here is the caller graph for this function:



8.17.2.5 prg_normalize_implicit_fermi()

```
subroutine, public prg_normalize_mod::prg_normalize_implicit_fermi (
              type(bml_matrix_t), intent(inout) h_bml,
              real(dp), intent(in) cnst,
              real(dp), intent(in) mu )
```

Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.

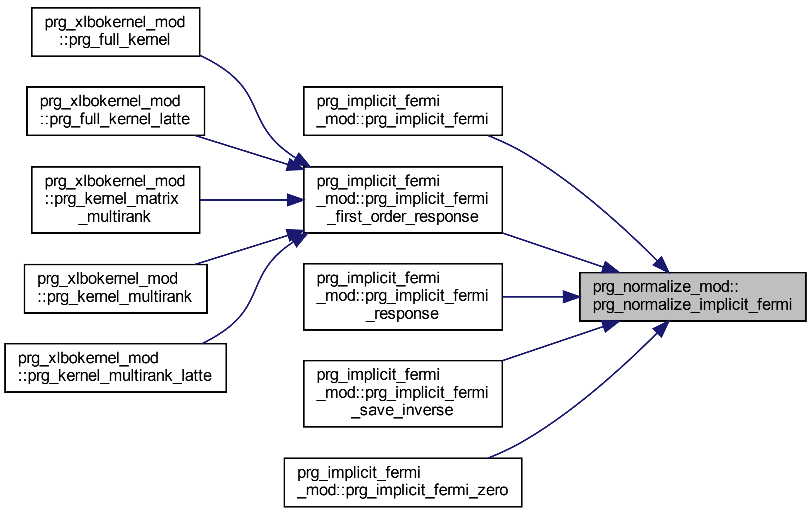
X0 = 0.5∗II - cnst∗(H0-mu0∗II) or X0 = (0.5 + cnst ∗ mu0)∗II -cnst∗ H0

Parameters

H_bml	Hamiltonian matrix
cnst	Constant based on beta and steps
mu	Chemical potential

Definition at line 84 of file prg_normalize_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_full_kernel
prg_xlbokernel_mod ::prg_full_kernel_latte
prg_xlbokernel_mod ::prg_kernel_matrix _multirank
prg_xlbokernel_mod ::prg_kernel_multirank
prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_implicit_fermi _mod::prg_implicit_fermi
prg_implicit_fermi _mod::prg_implicit_fermi _first_order_response
prg_implicit_fermi _mod::prg_implicit_fermi _response
prg_implicit_fermi _mod::prg_implicit_fermi _save_inverse
prg_implicit_fermi _mod::prg_implicit_fermi_zero
prg_normalize_mod:: prg_normalize_implicit_fermi

8.17.3 Variable Documentation

8.17.3.1 dp

```
integer, parameter prg_normalize_mod::dp = kind(1.0d0)  [private]
```

Definition at line 15 of file prg_normalize_mod.F90.

8.18 prg_openfiles_mod Module Reference

Module to handle input output files for the PROGRESS lib.

Functions/Subroutines

- integer function, public get_file_unit (io_max)

  Returns a unit number that is not in use.
- subroutine, public prg_open_file (io, name)

  Opens a file to write.
- subroutine, public prg_open_file_to_read (io, name)

  Opens a file to read.

8.18.1 Detailed Description

Module to handle input output files for the PROGRESS lib.

8.18.2 Function/Subroutine Documentation

8.18.2.1 get_file_unit()

```
integer function, public prg_openfiles_mod::get_file_unit (
            integer io_max )
```

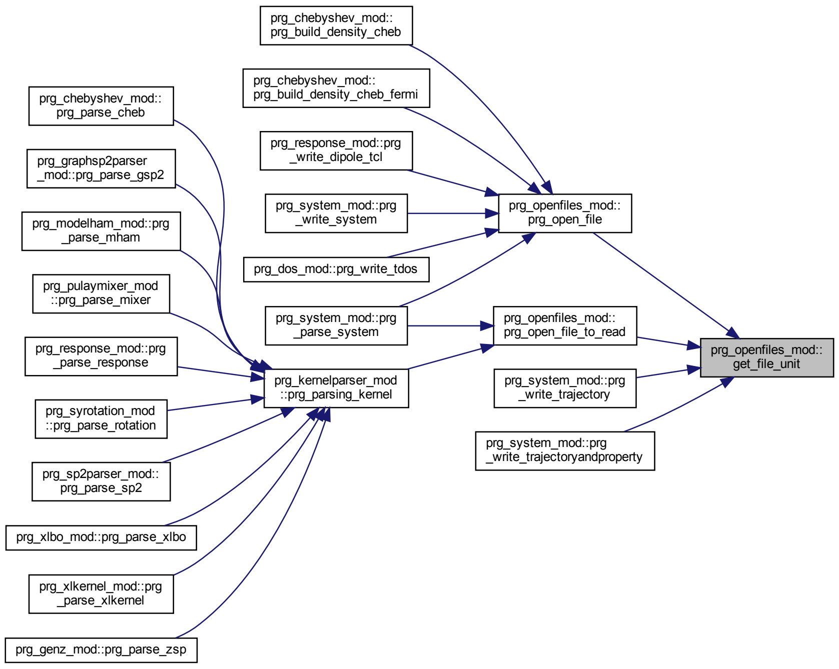
Returns a unit number that is not in use.

Parameters

io_max	Maximum units to search.
get_file_unit	Unit return to use for the file.

Definition at line 19 of file prg_openfiles_mod.F90.

Here is the caller graph for this function:


prg_chebyshev_mod::prg_build_density_cheb
prg_chebyshev_mod::prg_build_density_cheb_fermi
prg_chebyshev_mod::prg_parse_cheb
prg_response_mod::prg_write_dipole_tcl
prg_graphsp2parser_mod::prg_parse_gsp2
prg_system_mod::prg_write_system
prg_openfiles_mod::prg_open_file
prg_modelham_mod::prg_parse_mham
prg_dos_mod::prg_write_tdos
prg_pulaymixer_mod::prg_parse_mixer
prg_system_mod::prg_parse_system
prg_openfiles_mod::prg_open_file_to_read
prg_response_mod::prg_parse_response
prg_openfiles_mod::get_file_unit
prg_kernelparser_mod::prg_parsing_kernel
prg_system_mod::prg_write_trajectory
prg_syrotation_mod::prg_parse_rotation
prg_system_mod::prg_write_trajectoryandproperty
prg_sp2parser_mod::prg_parse_sp2
prg_xlbo_mod::prg_parse_xlbo
prg_xlkernel_mod::prg_parse_xlkernel
prg_genz_mod::prg_parse_zsp

8.18.2.2 prg_open_file()

```
subroutine, public prg_openfiles_mod::prg_open_file (
            integer io,
            character(len=*) name )
```

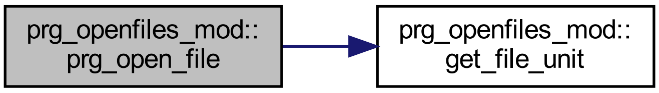
Opens a file to write.

Parameters

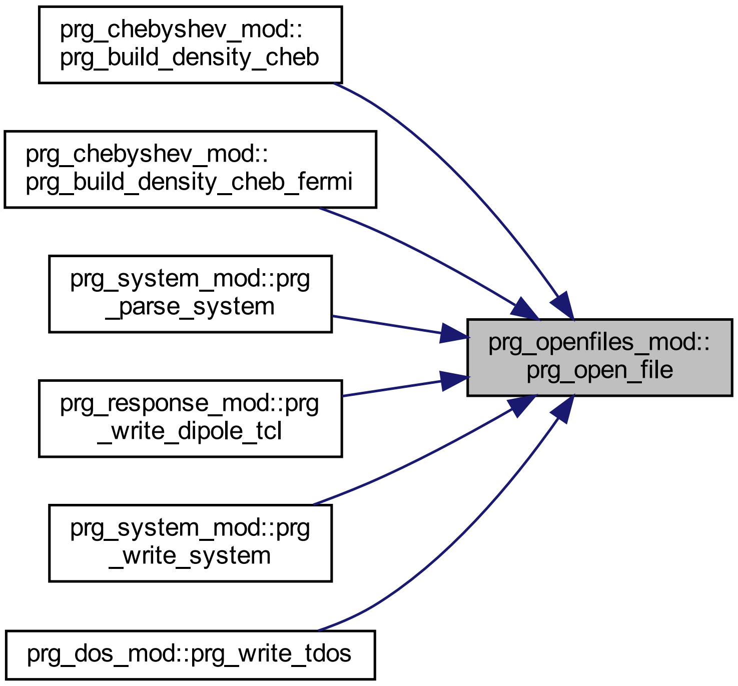
io	Unit for the file.
name	Name of the file.

Definition at line 38 of file prg_openfiles_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.2.3 prg_open_file_to_read()

```
subroutine, public prg_openfiles_mod::prg_open_file_to_read (
            integer io,
            character(len=*) name )
```

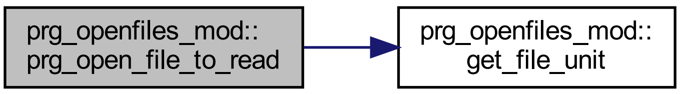
Opens a file to read.

Parameters

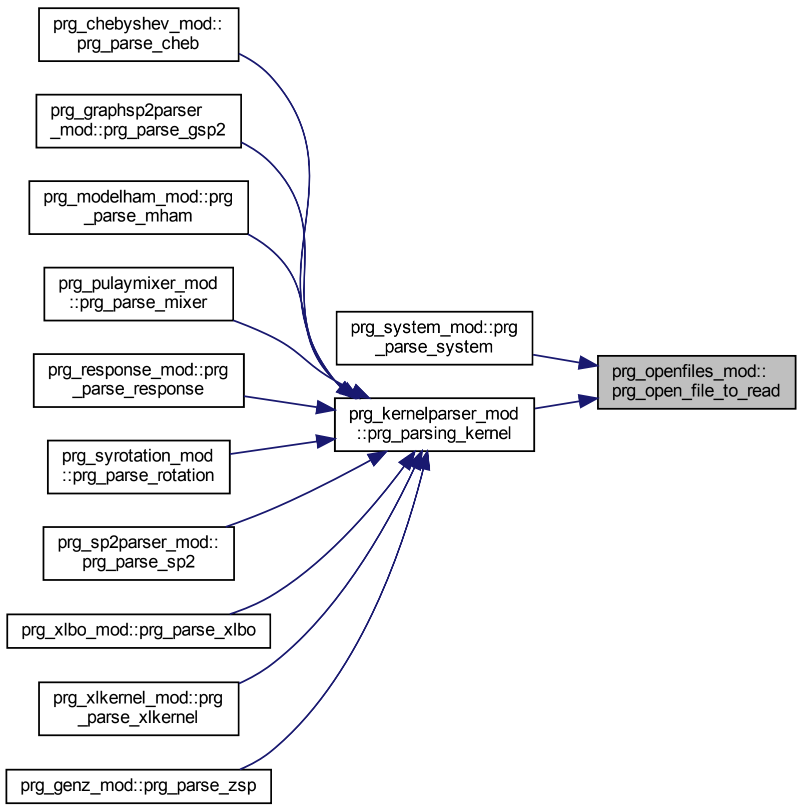
io	Unit for the file.
name	Name of the file.

Definition at line 54 of file prg_openfiles_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:


prg_chebyshev_mod::prg_parse_cheb
prg_graphsp2parser_mod::prg_parse_gsp2
prg_modelham_mod::prg_parse_mham
prg_pulaymixer_mod::prg_parse_mixer
prg_response_mod::prg_parse_response
prg_syrotation_mod::prg_parse_rotation
prg_sp2parser_mod::prg_parse_sp2
prg_xlbo_mod::prg_parse_xlbo
prg_xlkernel_mod::prg_parse_xlkernel
prg_genz_mod::prg_parse_zsp
prg_system_mod::prg_parse_system
prg_kernelparser_mod::prg_parsing_kernel
prg_openfiles_mod::prg_open_file_to_read

8.19 prg_parallel_mod Module Reference

The parallel module.

Data Types

- type rankreducedata_t

  Data structure for rection over MPI ranks.

Functions/Subroutines

- integer function, public getnranks ()
- integer function, public getmyrank ()
- integer function, public printrank ()
- subroutine, public prg_initparallel ()
- subroutine, public prg_shutdownparallel ()
- integer function saverequest (irequest)
- subroutine, public prg_barrierparallel ()
- subroutine, public sendreceiveparallel (sendBuf, sendLen, dest, recvBuf, recvLen, source, nreceived)

- subroutine, public isendparallel (sendBuf, sendLen, dest)
- subroutine, public sendparallel (sendBuf, sendLen, dest)
- subroutine, public prg_iprg_recvparallel (recvBuf, recvLen, rind)
- subroutine, public prg_recvparallel (recvBuf, recvLen)
- subroutine, public sumintparallel (sendBuf, recvBuf, icount)
- subroutine, public sumrealparallel (sendBuf, recvBuf, icount)
- subroutine, public maxintparallel (sendBuf, recvBuf, icount)
- subroutine, public maxrealparallel (sendBuf, recvBuf, icount)
- subroutine, public minintparallel (sendBuf, recvBuf, icount)
- subroutine, public minrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_minrealreduce (rvalue)
- subroutine, public prg_maxrealreduce (rvalue)
- subroutine, public prg_maxintreduce2 (value1, value2)
- subroutine, public prg_sumintreduce2 (value1, value2)
- subroutine, public prg_sumrealreduce (value1)
- subroutine, public prg_sumrealreduce2 (value1, value2)
- subroutine, public prg_sumrealreduce3 (value1, value2, value3)
- subroutine, public prg_sumrealreducen (valueVec, N)
- subroutine, public prg_sumintreducen (valueVec, N)
- subroutine, public minrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public maxrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_bcastparallel (buf, blen, root)
- subroutine, public allgatherrealparallel (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public allgatherintparallel (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public allgathervrealparallel (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public allgathervintparallel (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public prg_allsumrealreduceparallel (buf, buflen)
- subroutine, public prg_allsumintreduceparallel (buf, buflen)
- subroutine, public prg_allgatherparallel (a)
- subroutine, public prg_wait ()

Variables

- integer, parameter dp = kind(1.0d0)
- integer myrank
- integer nranks
- integer ierr
- integer reqcount
- integer, dimension(:), allocatable requestlist
- integer, dimension(:), allocatable rused

8.19.1 Detailed Description

The parallel module.

8.19.2 Function/Subroutine Documentation

8.19.2.1 allgatherintparallel()

```
subroutine, public prg_parallel_mod::allgatherintparallel (
             integer, dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             integer, dimension(*), intent(out) recvBuf,
             integer, intent(in) recvLen )
```

Definition at line 660 of file prg_parallel_mod.F90.

8.19.2.2 allgatherrealparallel()

```
subroutine, public prg_parallel_mod::allgatherrealparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, intent(in) recvLen )
```

Definition at line 644 of file prg_parallel_mod.F90.

8.19.2.3 allgathervintparallel()

```
subroutine, public prg_parallel_mod::allgathervintparallel (
             integer, dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             integer, dimension(*), intent(out) recvBuf,
             integer, dimension(*), intent(in) recvLen,
             integer, dimension(*), intent(in) recvDispl )
```

Definition at line 696 of file prg_parallel_mod.F90.

8.19.2.4 allgathervrealparallel()

```
subroutine, public prg_parallel_mod::allgathervrealparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, dimension(*), intent(in) recvLen,
             integer, dimension(*), intent(in) recvDispl )
```

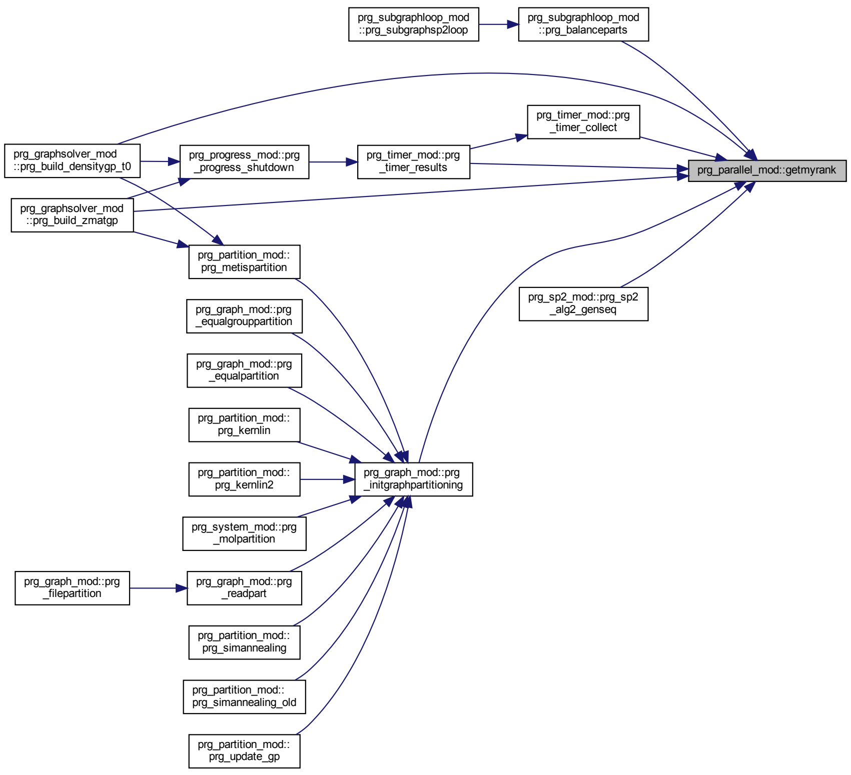
Definition at line 676 of file prg_parallel_mod.F90.

8.19.2.5 getmyrank()

```
integer function, public prg_parallel_mod::getmyrank
```

Definition at line 99 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

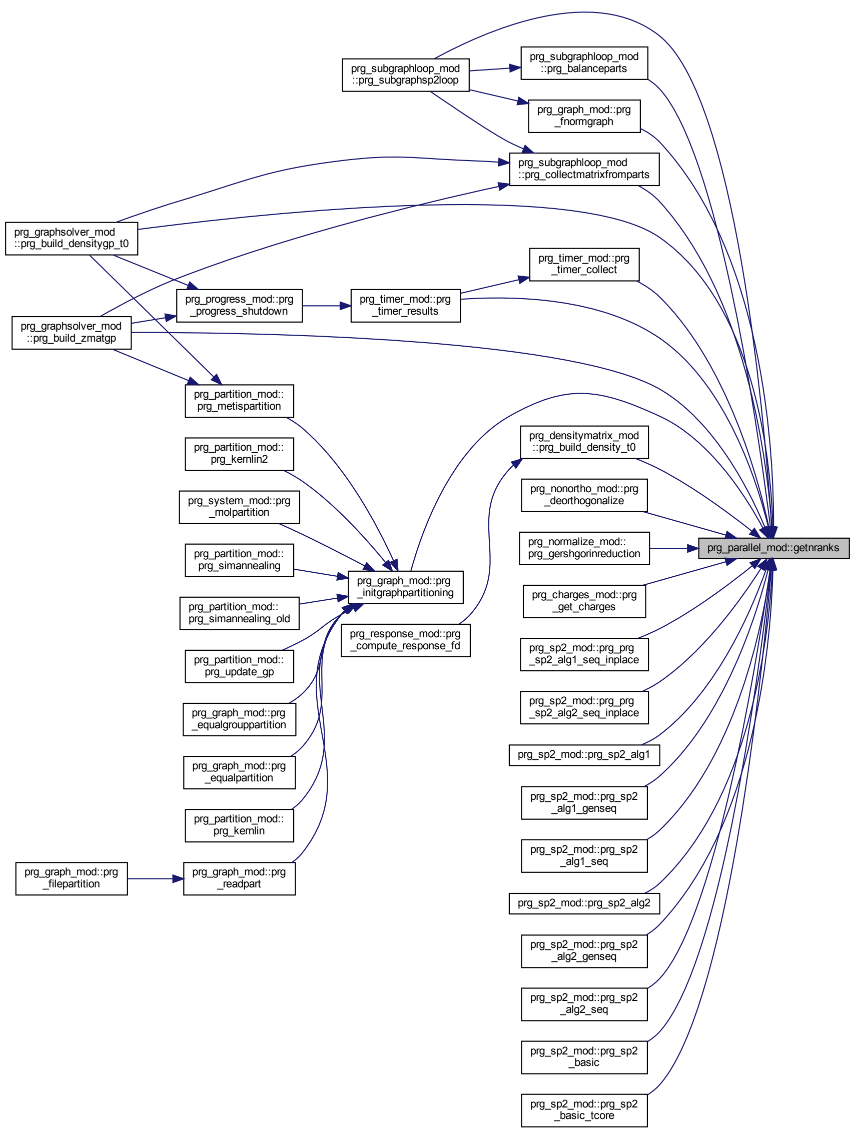

prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_subgraphloop_mod ::prg_balanceparts
prg_timer_mod::prg _timer_collect
prg_graphsolver_mod ::prg_build_densitygp_t0
prg_progress_mod::prg _progress_shutdown
prg_timer_mod::prg _timer_results
prg_parallel_mod::getmyrank
prg_graphsolver_mod ::prg_build_zmatgp
prg_partition_mod:: prg_metispartition
prg_sp2_mod::prg_sp2 _alg2_genseq
prg_graph_mod::prg _equalgrouppartition
prg_graph_mod::prg _equalpartition
prg_partition_mod:: prg_kernlin
prg_partition_mod:: prg_kernlin2
prg_graph_mod::prg _initgraphpartitioning
prg_system_mod::prg _molpartition
prg_graph_mod::prg _filepartition
prg_graph_mod::prg _readpart
prg_partition_mod:: prg_simannealing
prg_partition_mod:: prg_simannealing_old
prg_partition_mod:: prg_update_gp

8.19.2.6 getnranks()

```
integer function, public prg_parallel_mod::getnranks
```

Definition at line 88 of file prg_parallel_mod.F90.

Here is the caller graph for this function:


prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_subgraphloop_mod ::prg_balanceparts
prg_graph_mod::prg _fnormgraph
prg_subgraphloop_mod ::prg_collectmatrixfromparts
prg_graphsolver_mod ::prg_build_densitygp_t0
prg_timer_mod::prg _timer_collect
prg_progress_mod::prg _progress_shutdown
prg_timer_mod::prg _timer_results
prg_graphsolver_mod ::prg_build_zmatgp
prg_partition_mod:: prg_metispartition
prg_densitymatrix_mod ::prg_build_density_t0
prg_partition_mod:: prg_kernlin2
prg_nonortho_mod::prg _deorthogonalize
prg_system_mod::prg _molpartition
prg_normalize_mod:: prg_gershgorinreduction
prg_parallel_mod::getnranks
prg_partition_mod:: prg_simannealing
prg_graph_mod::prg _initgraphpartitioning
prg_charges_mod::prg _get_charges
prg_partition_mod:: prg_simannealing_old
prg_response_mod::prg _compute_response_fd
prg_sp2_mod::prg_prg _sp2_alg1_seq_inplace
prg_partition_mod:: prg_update_gp
prg_sp2_mod::prg_prg _sp2_alg2_seq_inplace
prg_graph_mod::prg _equalgrouppartition
prg_sp2_mod::prg_sp2_alg1
prg_graph_mod::prg _equalpartition
prg_sp2_mod::prg_sp2 _alg1_genseq
prg_partition_mod:: prg_kernlin
prg_sp2_mod::prg_sp2 _alg1_seq
prg_graph_mod::prg _filepartition
prg_graph_mod::prg _readpart
prg_sp2_mod::prg_sp2_alg2
prg_sp2_mod::prg_sp2 _alg2_genseq
prg_sp2_mod::prg_sp2 _alg2_seq
prg_sp2_mod::prg_sp2 _basic
prg_sp2_mod::prg_sp2 _basic_tcore

8.19.2.7 isendparallel()

```
subroutine, public prg_parallel_mod::isendparallel (
            real(dp), dimension(*), intent(in) sendBuf,
            integer, intent(in) sendLen,
            integer, intent(in) dest )
```

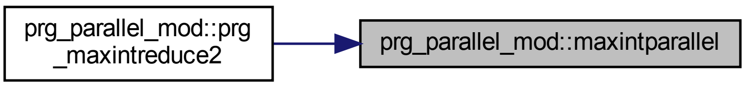
Definition at line 230 of file prg_parallel_mod.F90.

8.19.2.8 maxintparallel()

```
subroutine, public prg_parallel_mod::maxintparallel (
            integer, dimension(*), intent(in) sendBuf,
            integer, dimension(*), intent(out) recvBuf,
            integer, intent(in) icount )
```

Definition at line 337 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

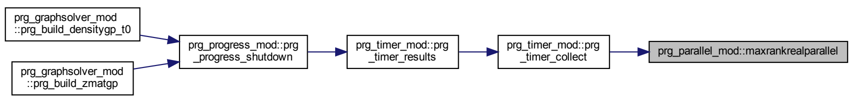


8.19.2.9 maxrankrealparallel()

```
subroutine, public prg_parallel_mod::maxrankrealparallel (
            type(rankreducedata_t), dimension(*), intent(in) sendBuf,
            type(rankreducedata_t), dimension(*), intent(out) recvBuf,
            integer, intent(in) icount )
```

Definition at line 607 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

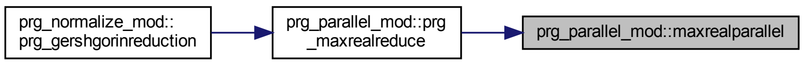


8.19.2.10 maxrealparallel()

```
subroutine, public prg_parallel_mod::maxrealparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, intent(in) icount )
```

Definition at line 358 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



8.19.2.11 minintparallel()

```
subroutine, public prg_parallel_mod::minintparallel (
             integer, dimension(*), intent(in) sendBuf,
             integer, dimension(*), intent(out) recvBuf,
             integer, intent(in) icount )
```

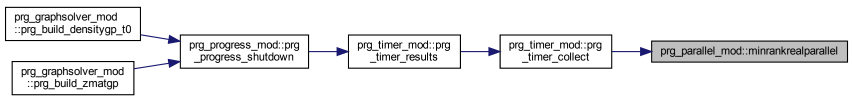
Definition at line 379 of file prg_parallel_mod.F90.

8.19.2.12 minrankrealparallel()

```
subroutine, public prg_parallel_mod::minrankrealparallel (
             type(rankreducedata_t), dimension(*), intent(in) sendBuf,
             type(rankreducedata_t), dimension(*), intent(out) recvBuf,
             integer, intent(in) icount )
```

Definition at line 584 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

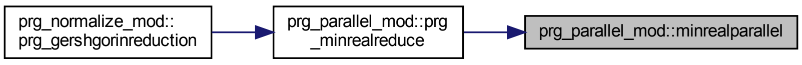


8.19.2.13 minrealparallel()

```
subroutine, public prg_parallel_mod::minrealparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, intent(in) icount )
```

Definition at line 400 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

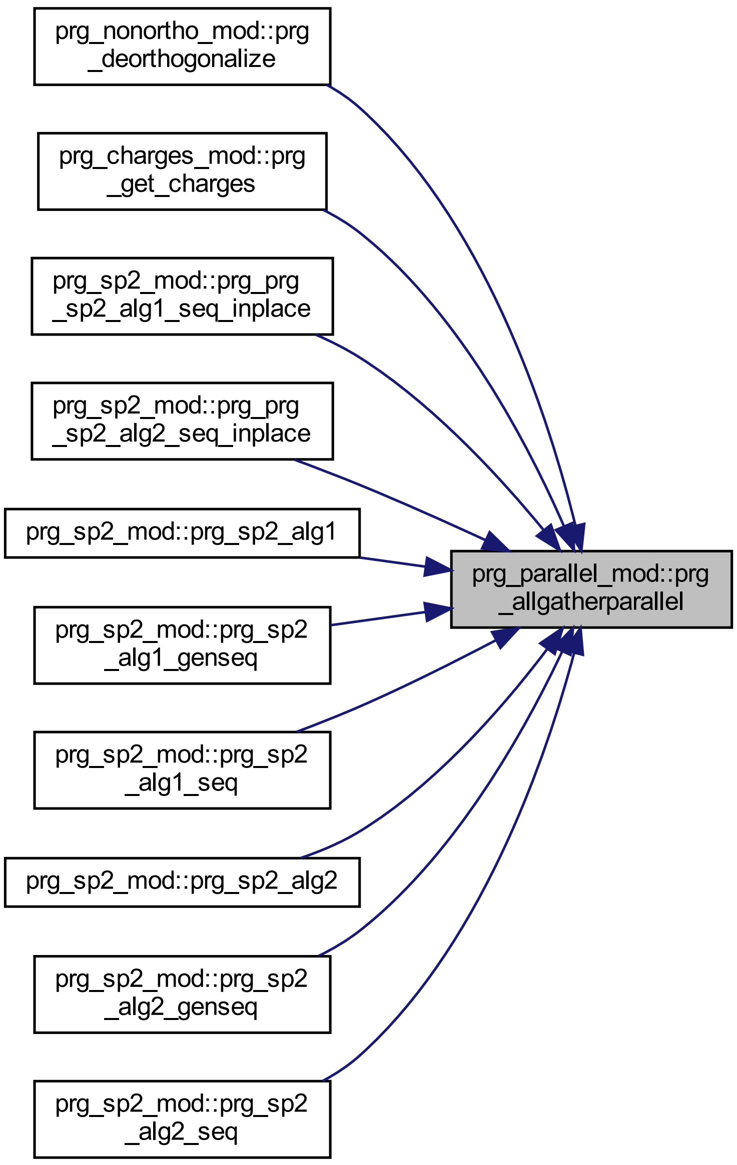


8.19.2.14 prg_allgatherparallel()

```
subroutine, public prg_parallel_mod::prg_allgatherparallel (
             type (bml_matrix_t), intent(inout) a )
```

Definition at line 744 of file prg_parallel_mod.F90.

Here is the caller graph for this function:


prg_nonortho_mod::prg_deorthogonalize
prg_charges_mod::prg_get_charges
prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace
prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace
prg_sp2_mod::prg_sp2_alg1
prg_sp2_mod::prg_sp2_alg1_genseq
prg_sp2_mod::prg_sp2_alg1_seq
prg_sp2_mod::prg_sp2_alg2
prg_sp2_mod::prg_sp2_alg2_genseq
prg_sp2_mod::prg_sp2_alg2_seq
prg_parallel_mod::prg_allgatherparallel

8.19.2.15 prg_allsumintreduceparallel()

```
subroutine, public prg_parallel_mod::prg_allsumintreduceparallel (
              integer, dimension(*), intent(inout) buf,
              integer, intent(in) buflen )
```

Definition at line 729 of file prg_parallel_mod.F90.

8.19.2.16 prg_allsumrealreduceparallel()

```
subroutine, public prg_parallel_mod::prg_allsumrealreduceparallel (
            real(dp), dimension(*), intent(inout) buf,
            integer, intent(in) buflen )
```

Definition at line 714 of file prg_parallel_mod.F90.

8.19.2.17 prg_barrierparallel()

```
subroutine, public prg_parallel_mod::prg_barrierparallel
```

Definition at line 196 of file prg_parallel_mod.F90.

8.19.2.18 prg_bcastparallel()

```
subroutine, public prg_parallel_mod::prg_bcastparallel (
            character, dimension(*), intent(in) buf,
            integer, intent(in) blen,
            integer, intent(in) root )
```

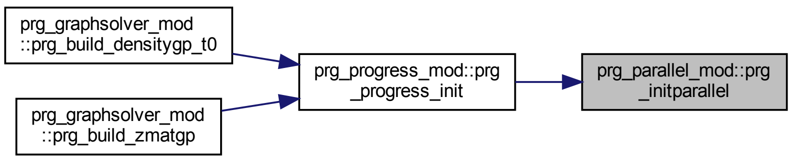
Definition at line 630 of file prg_parallel_mod.F90.

8.19.2.19 prg_initparallel()

```
subroutine, public prg_parallel_mod::prg_initparallel
```

Definition at line 127 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

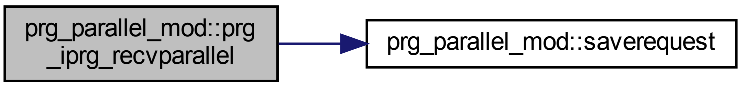


8.19.2.20 prg_iprg_recvparallel()

```
subroutine, public prg_parallel_mod::prg_iprg_recvparallel (
            real(dp), dimension(*) recvBuf,
            integer, intent(in) recvLen,
            integer rind )
```

Definition at line 261 of file prg_parallel_mod.F90.

Here is the call graph for this function:

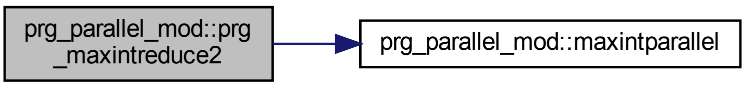


8.19.2.21 prg_maxintreduce2()

```
subroutine, public prg_parallel_mod::prg_maxintreduce2 (
            integer, intent(inout) value1,
            integer, intent(inout) value2 )
```

Definition at line 453 of file prg_parallel_mod.F90.

Here is the call graph for this function:

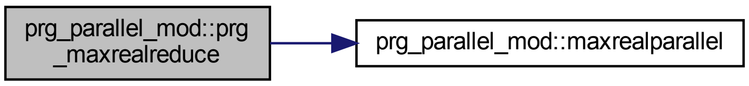


8.19.2.22 prg_maxrealreduce()

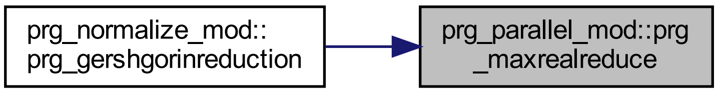
```
subroutine, public prg_parallel_mod::prg_maxrealreduce (
             real(dp), intent(inout) rvalue )
```

Definition at line 437 of file prg_parallel_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::prg_maxrealreduce
prg_parallel_mod::maxrealparallel

Here is the caller graph for this function:

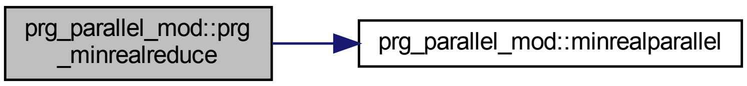

prg_normalize_mod::prg_gershgorinreduction
prg_parallel_mod::prg_maxrealreduce

8.19.2.23 prg_minrealreduce()

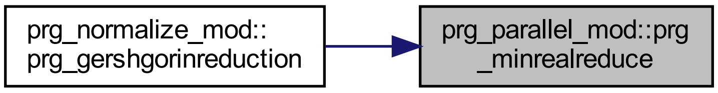
```
subroutine, public prg_parallel_mod::prg_minrealreduce (
             real(dp), intent(inout) rvalue )
```

Definition at line 421 of file prg_parallel_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::prg_minrealreduce
prg_parallel_mod::minrealparallel

Here is the caller graph for this function:



8.19.2.24 prg_recvparallel()

```
subroutine, public prg_parallel_mod::prg_recvparallel (
             real(dp), dimension(*) recvBuf,
             integer, intent(in) recvLen )
```

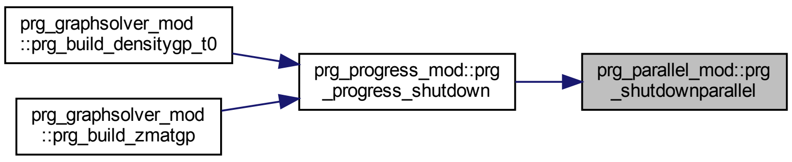
Definition at line 279 of file prg_parallel_mod.F90.

8.19.2.25 prg_shutdownparallel()

```
subroutine, public prg_parallel_mod::prg_shutdownparallel
```

Definition at line 154 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

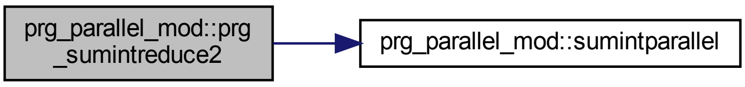


8.19.2.26 prg_sumintreduce2()

```
subroutine, public prg_parallel_mod::prg_sumintreduce2 (
             integer, intent(inout) value1,
             integer, intent(inout) value2 )
```

Definition at line 471 of file prg_parallel_mod.F90.

Here is the call graph for this function:

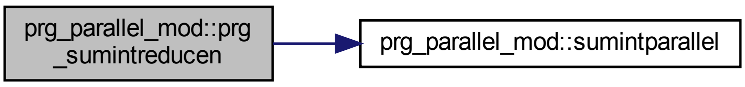

prg_parallel_mod::prg_sumintreduce2
prg_parallel_mod::sumintparallel

8.19.2.27 prg_sumintreducen()

```
subroutine, public prg_parallel_mod::prg_sumintreducen (
             integer, dimension(n), intent(inout) valueVec,
             integer, intent(in) N )
```

Definition at line 564 of file prg_parallel_mod.F90.

Here is the call graph for this function:

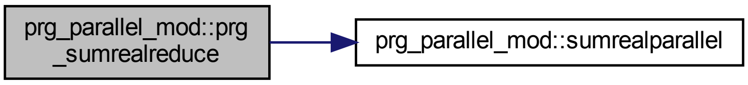

prg_parallel_mod::prg_sumintreducen
prg_parallel_mod::sumintparallel

8.19.2.28 prg_sumrealreduce()

```
subroutine, public prg_parallel_mod::prg_sumrealreduce (
             real(dp), intent(inout) value1 )
```

Definition at line 489 of file prg_parallel_mod.F90.

Here is the call graph for this function:

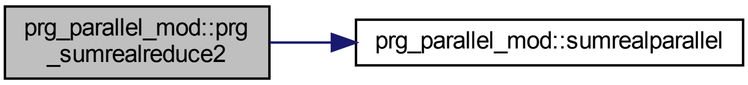


8.19.2.29 prg_sumrealreduce2()

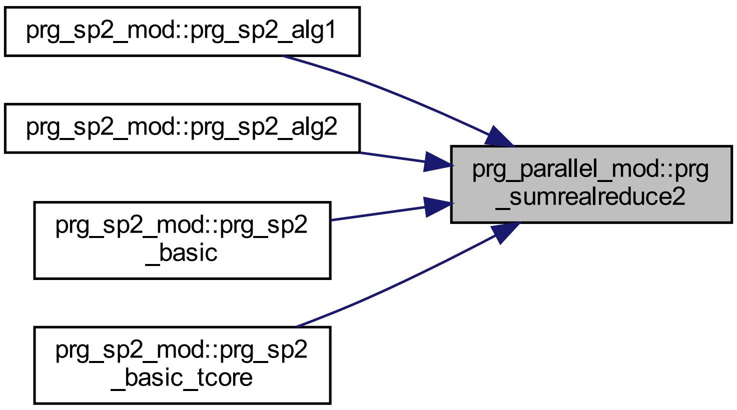
```
subroutine, public prg_parallel_mod::prg_sumrealreduce2 (
             real(dp), intent(inout) value1,
             real(dp), intent(inout) value2 )
```

Definition at line 505 of file prg_parallel_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

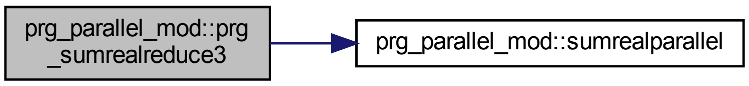

prg_sp2_mod::prg_sp2_alg1
prg_sp2_mod::prg_sp2_alg2
prg_sp2_mod::prg_sp2 _basic
prg_sp2_mod::prg_sp2 _basic_tcore
prg_parallel_mod::prg _sumrealreduce2

8.19.2.30 prg_sumrealreduce3()

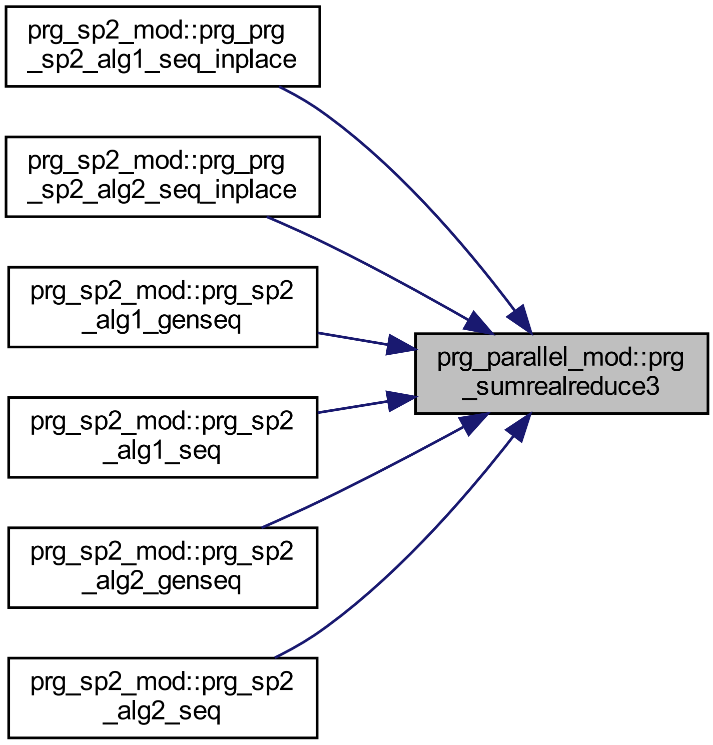
```
subroutine, public prg_parallel_mod::prg_sumrealreduce3 (
             real(dp), intent(inout) value1,
             real(dp), intent(inout) value2,
             real(dp), intent(inout) value3 )
```

Definition at line 523 of file prg_parallel_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::prg _sumrealreduce3
prg_parallel_mod::sumrealparallel

Here is the caller graph for this function:

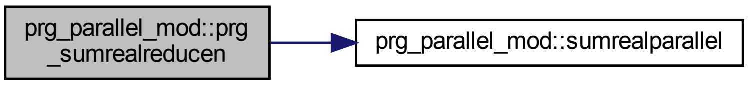

prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace
prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace
prg_sp2_mod::prg_sp2_alg1_genseq
prg_sp2_mod::prg_sp2_alg1_seq
prg_sp2_mod::prg_sp2_alg2_genseq
prg_sp2_mod::prg_sp2_alg2_seq
prg_parallel_mod::prg_sumrealreduce3

8.19.2.31 prg_sumrealreducen()

```
subroutine, public prg_parallel_mod::prg_sumrealreducen (
             real(dp), dimension(n), intent(inout) valueVec,
             integer, intent(in) N )
```

Definition at line 543 of file prg_parallel_mod.F90.

Here is the call graph for this function:

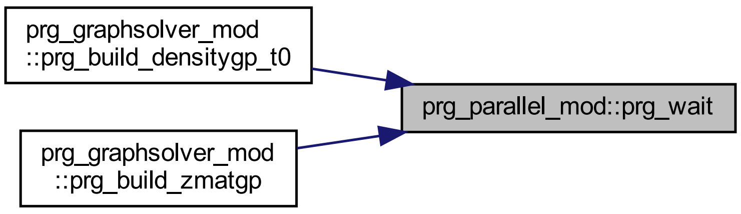

prg_parallel_mod::prg_sumrealreducen
prg_parallel_mod::sumrealparallel

8.19.2.32 prg_wait()

```
subroutine, public prg_parallel_mod::prg_wait
```

Definition at line 758 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

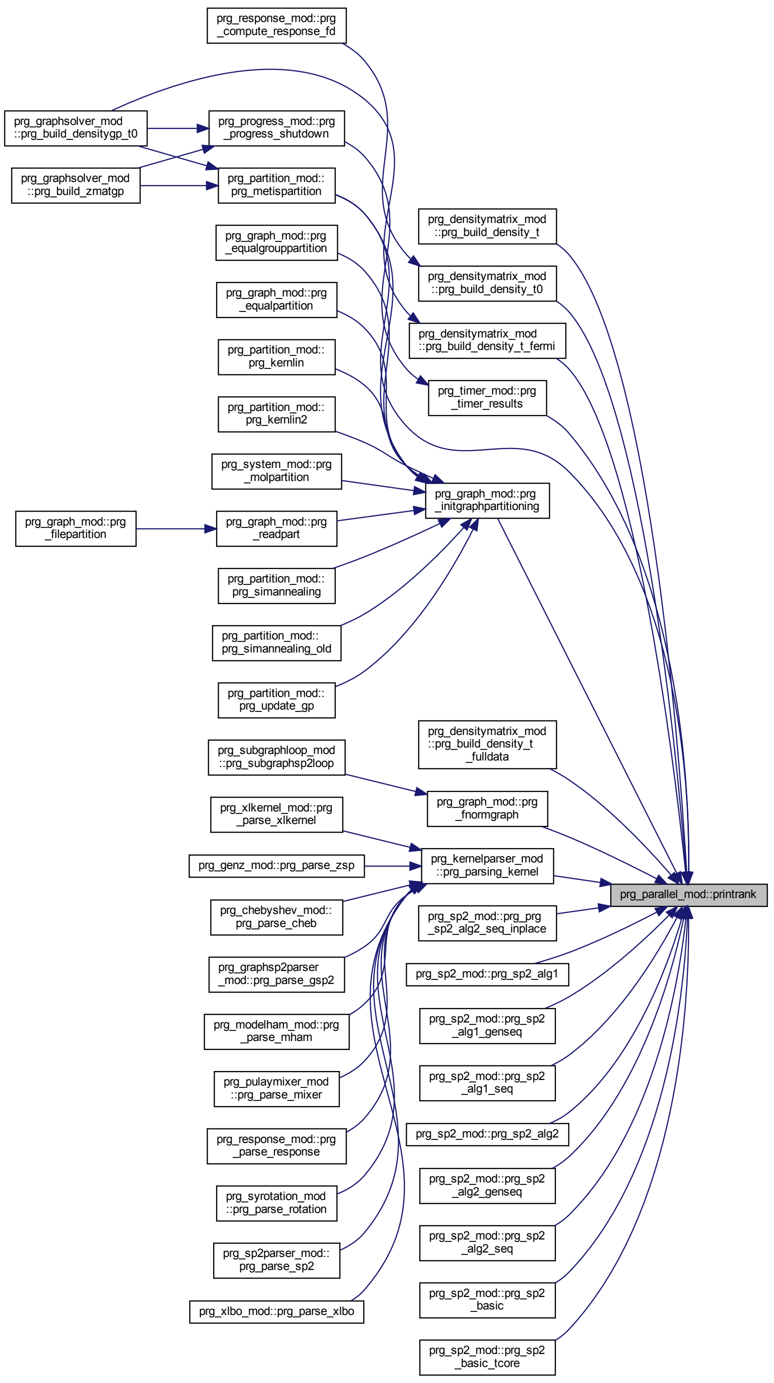

prg_graphsolver_mod ::prg_build_densitygp_t0
prg_parallel_mod::prg_wait
prg_graphsolver_mod ::prg_build_zmatgp

8.19.2.33 printrank()

```
integer function, public prg_parallel_mod::printrank
```

Definition at line 111 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

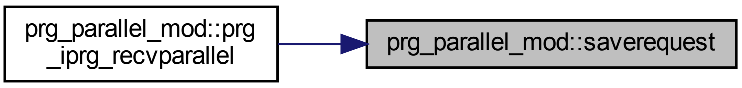

prg_response_mod::prg_compute_response_fd
prg_graphsolver_mod::prg_build_densitygp_t0
prg_progress_mod::prg_progress_shutdown
prg_graphsolver_mod::prg_build_zmatgp
prg_partition_mod::prg_metispartition
prg_densitymatrix_mod::prg_build_density_t
prg_graph_mod::prg_equalgrouppartition
prg_densitymatrix_mod::prg_build_density_t0
prg_graph_mod::prg_equalpartition
prg_densitymatrix_mod::prg_build_density_t_fermi
prg_partition_mod::prg_kernlin
prg_timer_mod::prg_timer_results
prg_partition_mod::prg_kernlin2
prg_system_mod::prg_molpartition
prg_graph_mod::prg_initgraphpartitioning
prg_graph_mod::prg_filepartition
prg_graph_mod::prg_readpart
prg_partition_mod::prg_simannealing
prg_partition_mod::prg_simannealing_old
prg_partition_mod::prg_update_gp
prg_densitymatrix_mod::prg_build_density_t_fulldata
prg_subgraphloop_mod::prg_subgraphsp2loop
prg_graph_mod::prg_fnormgraph
prg_xlkernel_mod::prg_parse_xlkernel
prg_kernelparser_mod::prg_parsing_kernel
prg_genz_mod::prg_parse_zsp
prg_parallel_mod::printrank
prg_chebyshev_mod::prg_parse_cheb
prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace
prg_graphsp2parser_mod::prg_parse_gsp2
prg_sp2_mod::prg_sp2_alg1
prg_modelham_mod::prg_parse_mham
prg_sp2_mod::prg_sp2_alg1_genseq
prg_pulaymixer_mod::prg_parse_mixer
prg_sp2_mod::prg_sp2_alg1_seq
prg_response_mod::prg_parse_response
prg_sp2_mod::prg_sp2_alg2
prg_syrotation_mod::prg_parse_rotation
prg_sp2_mod::prg_sp2_alg2_genseq
prg_sp2parser_mod::prg_parse_sp2
prg_sp2_mod::prg_sp2_alg2_seq
prg_xlbo_mod::prg_parse_xlbo
prg_sp2_mod::prg_sp2_basic
prg_sp2_mod::prg_sp2_basic_tcore

8.19.2.34 saverequest()

```
integer function prg_parallel_mod::saverequest (
            integer, intent(in) irequest )  [private]
```

Definition at line 170 of file prg_parallel_mod.F90.

Here is the caller graph for this function:



8.19.2.35 sendparallel()

```
subroutine, public prg_parallel_mod::sendparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             integer, intent(in) dest )
```

Definition at line 246 of file prg_parallel_mod.F90.

8.19.2.36 sendreceiveparallel()

```
subroutine, public prg_parallel_mod::sendreceiveparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             integer, intent(in) sendLen,
             integer, intent(in) dest,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, intent(in) recvLen,
             integer, intent(in) source,
             integer, intent(out) nreceived )
```

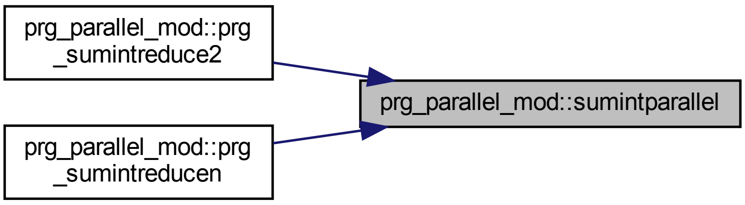
Definition at line 207 of file prg_parallel_mod.F90.

8.19.2.37 sumintparallel()

```
subroutine, public prg_parallel_mod::sumintparallel (
             integer, dimension(*), intent(in) sendBuf,
             integer, dimension(*) recvBuf,
             integer, intent(in) icount )
```

Definition at line 295 of file prg_parallel_mod.F90.

Here is the caller graph for this function:

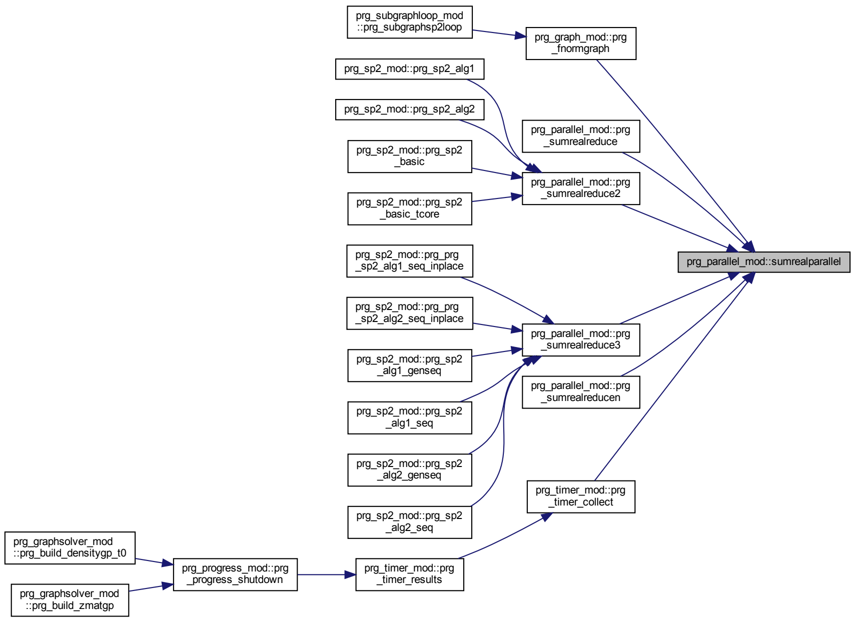

prg_parallel_mod::prg_sumintreduce2
prg_parallel_mod::prg_sumintreducen
prg_parallel_mod::sumintparallel

8.19.2.38 sumrealparallel()

```
subroutine, public prg_parallel_mod::sumrealparallel (
             real(dp), dimension(*), intent(in) sendBuf,
             real(dp), dimension(*), intent(out) recvBuf,
             integer, intent(in) icount )
```

Definition at line 316 of file prg_parallel_mod.F90.

Here is the caller graph for this function:


prg_subgraphloop_mod::prg_subgraphsp2loop
prg_graph_mod::prg_fnormgraph
prg_sp2_mod::prg_sp2_alg1
prg_sp2_mod::prg_sp2_alg2
prg_parallel_mod::prg_sumrealreduce
prg_sp2_mod::prg_sp2_basic
prg_parallel_mod::prg_sumrealreduce2
prg_sp2_mod::prg_sp2_basic_tcore
prg_parallel_mod::sumrealparallel
prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace
prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace
prg_parallel_mod::prg_sumrealreduce3
prg_sp2_mod::prg_sp2_alg1_genseq
prg_parallel_mod::prg_sumrealreducen
prg_sp2_mod::prg_sp2_alg1_seq
prg_sp2_mod::prg_sp2_alg2_genseq
prg_timer_mod::prg_timer_collect
prg_sp2_mod::prg_sp2_alg2_seq
prg_graphsolver_mod::prg_build_densitygp_t0
prg_progress_mod::prg_progress_shutdown
prg_timer_mod::prg_timer_results
prg_graphsolver_mod::prg_build_zmatgp

8.19.3 Variable Documentation

8.19.3.1 dp

```
integer, parameter prg_parallel_mod::dp = kind(1.0d0)  [private]
```

Definition at line 26 of file prg_parallel_mod.F90.

8.19.3.2 ierr

```
integer prg_parallel_mod::ierr  [private]
```

Definition at line 29 of file prg_parallel_mod.F90.

8.19.3.3 myrank

```
integer prg_parallel_mod::myrank  [private]
```

Definition at line 28 of file prg_parallel_mod.F90.

8.19.3.4 nranks

```
integer prg_parallel_mod::nranks  [private]
```

Definition at line 28 of file prg_parallel_mod.F90.

8.19.3.5 reqcount

```
integer prg_parallel_mod::reqcount  [private]
```

Definition at line 29 of file prg_parallel_mod.F90.

8.19.3.6 requestlist

```
integer, dimension(:), allocatable prg_parallel_mod::requestlist  [private]
```

Definition at line 30 of file prg_parallel_mod.F90.

8.19.3.7 rused

```
integer, dimension(:), allocatable prg_parallel_mod::rused  [private]
```

Definition at line 30 of file prg_parallel_mod.F90.

8.20 prg_partition_mod Module Reference

The partition module.

Functions/Subroutines

- subroutine, public prg_metispartition (gp, ngroups, nnodes, xadj, adjncy, nparts, part, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Create graph partitions minizing number of cut edges.
- subroutine, public prg_costpartition (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sum↩ Cubes, maxCH, smooth_maxCH, pnorm)

  Compute cost of a partition.
- subroutine, public update_prg_costpartition (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_↩ count, sumCubes, maxCH, smooth_maxCH, pnorm, node, new_part)

  Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.
- subroutine prg_accept_prob (it, prg_delta, r)

  Compute acceptance probability for simulated annealing.
- subroutine prg_costindex (cost, sumCubes, maxCH, smooth_maxCH, obj_fun)

  Choose objective function to work with.
- subroutine prg_rand_node (gp, node, seed)

  Pick a random node.
- subroutine, public prg_simannealing (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

  Graph partitioning based on Simulated Annealing.
- subroutine, public prg_kernlin (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, nconverg, seed)

  Graph partitioning based on inspired by Kernighan-Lin Review METiS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_↩ part),with prg_delta = change in obj_value Dequeue and allow hill climbing.
- subroutine, public prg_update_gp (gp, partNumber, core_count)
- subroutine prg_rand_shuffle (array, seed)

  Randomly shuffle array.
- subroutine, public prg_check_arrays (gp, core_count, CH_count, Halo_count)

  Error checking Checking that core_count, CH_count, Halo_count match.
- subroutine, public prg_kernlin_queue (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain Currently implementation is very slow.
- subroutine prg_find_best_move (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sum↩ Cubes, maxCH, smooth_maxCH, pnorm, best_node, best_part)

  For kerlin_queue to find (vertex, new_part) pair with highest gain.
- subroutine, public prg_kernlin2 (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sum↩ Cubes, maxCH, smooth_maxCH, pnorm)
- subroutine prg_get_largest_hedge_in_part (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_↩ count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)
- subroutine, public prg_simannealing_old (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

Variables

- integer, parameter dp = kind(1.0d0)
- integer, parameter metis_index_kind = METIS_INDEX_KIND

  From /usr/include/metis.h.
- integer, parameter metis_real_kind = kind(METIS_REAL_KIND)

  From /usr/include/metis.h.

8.20.1 Detailed Description

The partition module.

Contains different partitioning algorihms such as Metis, Simulated Annealing etc. Also contains optimization routines to improve upon existing partitioning, such as simulated annealing, etc.

8.20.2 Function/Subroutine Documentation

8.20.2.1 prg_accept_prob()

```
subroutine prg_partition_mod::prg_accept_prob (
             integer, intent(in) it,
             real(dp), intent(in) prg_delta,
             real, intent(inout) r )  [private]
```

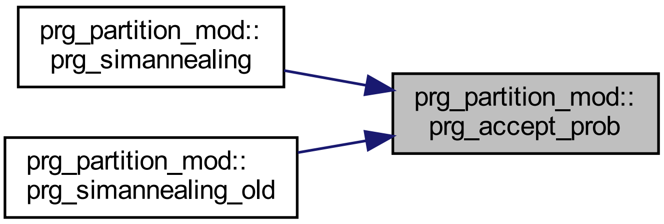
Compute acceptance probability for simulated annealing.

Parameters

it	iteration
prg_delta	(new_obj_value - old_obj_value)
r	acceptance probability

Definition at line 489 of file prg_partition_mod.F90.

Here is the caller graph for this function:



8.20.2.2 prg_check_arrays()

```
subroutine, public prg_partition_mod::prg_check_arrays (
             type (graph_partitioning_t), intent(inout) gp,
             integer, dimension(:), intent(inout), allocatable core_count,
             integer, dimension(:), intent(inout), allocatable CH_count,
             integer, dimension(:,:), intent(inout), allocatable Halo_count )
```

Error checking Checking that core_count, CH_count, Halo_count match.

Definition at line 1146 of file prg_partition_mod.F90.

8.20.2.3 prg_costindex()

```
subroutine prg_partition_mod::prg_costindex (
            real(dp), intent(inout) cost,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            integer, intent(inout) obj_fun )  [private]
```

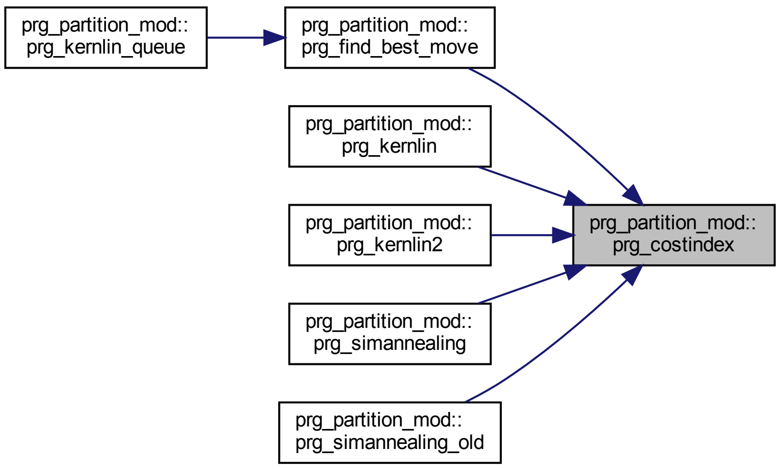
Choose objective function to work with.

Parameters

cost	output according to chosen obj_fun
sumCubes	Sum of cubes obj value
maxCH	maximum core-halo part size obective value
obj_fun	0=sumcubes, 1=maxCH

Definition at line 507 of file prg_partition_mod.F90.

Here is the caller graph for this function:


prg_partition_mod:: prg_kernlin_queue
prg_partition_mod:: prg_find_best_move
prg_partition_mod:: prg_kernlin
prg_partition_mod:: prg_kernlin2
prg_partition_mod:: prg_costindex
prg_partition_mod:: prg_simannealing
prg_partition_mod:: prg_simannealing_old

8.20.2.4 prg_costpartition()

```
subroutine, public prg_partition_mod::prg_costpartition (
             type (graph_partitioning_t), intent(inout) gp,
             integer, dimension(:), intent(inout), allocatable xadj,
             integer, dimension(:), intent(inout), allocatable adjncy,
             integer, dimension(:), intent(in), allocatable partNumber,
             integer, dimension(:), intent(inout), allocatable core_count,
             integer, dimension(:), intent(inout), allocatable CH_count,
             integer, dimension(:,:), intent(inout), allocatable Halo_count,
             real(dp), intent(inout) sumCubes,
             real(dp), intent(inout) maxCH,
             real(dp), intent(inout) smooth_maxCH,
             real(dp), intent(inout) pnorm )
```

Compute cost of a partition.

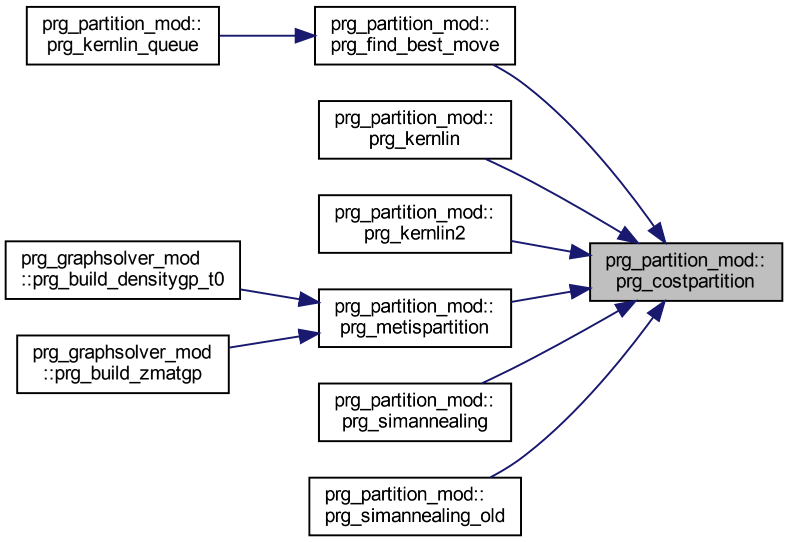
Parameters

gp	Graph partitioning
xadj	CSR array of graph nodes
adjncy	CSR array of graph neighbors
nparts	Number of Parts
partNumber	Partition vector
core_count	Array: number of core vertices in each part
CH_count	Array: number of core+halo vertices in each part
Halo_count	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i\ with k connections
sumCubes	Sum of cubes objective value
maxCh	maximum core-halo part size obective value

prg_initialize

Definition at line 327 of file prg_partition_mod.F90.

Here is the caller graph for this function:



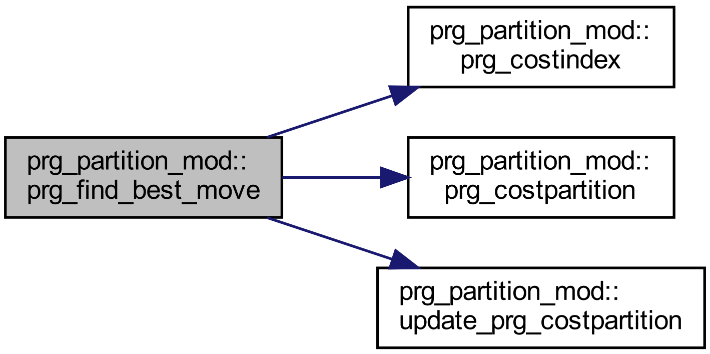
8.20.2.5 prg_find_best_move()

```
subroutine prg_partition_mod::prg_find_best_move (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm,
            integer, intent(inout) best_node,
            integer, intent(inout) best_part )  [private]
```

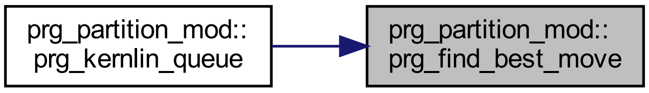
For kerlin_queue to find (vertex, new_part) pair with highest gain.

Definition at line 1209 of file prg_partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



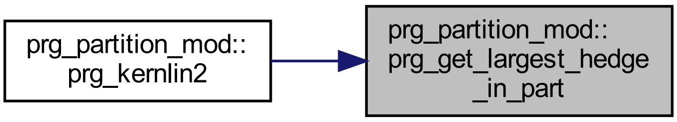
8.20.2.6 prg_get_largest_hedge_in_part()

```
subroutine prg_partition_mod::prg_get_largest_hedge_in_part (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm,
            integer, intent(inout) search_part,
            integer, intent(inout) largest_Hedge )  [private]
```

i can be viewed as a hyperedge for all hyperedges in search_part, pick the one with largest size

Definition at line 1421 of file prg_partition_mod.F90.

Here is the caller graph for this function:



8.20.2.7 prg_kernlin()

```
subroutine, public prg_partition_mod::prg_kernlin (
             type (graph_partitioning_t), intent(inout) gp,
             integer, dimension(:), intent(inout), allocatable xadj,
             integer, dimension(:), intent(inout), allocatable adjncy,
             integer, dimension(:), intent(inout), allocatable partNumber,
             integer, dimension(:), intent(inout), allocatable core_count,
             integer, dimension(:), intent(inout), allocatable CH_count,
             integer, dimension(:,:), intent(inout), allocatable Halo_count,
             real(dp), intent(inout) sumCubes,
             real(dp), intent(inout) maxCH,
             real(dp), intent(inout) smooth_maxCH,
             real(dp), intent(inout) pnorm,
             integer, intent(in) nconverg,
             integer, intent(inout) seed )
```

Graph partitioning based on inspired by Kernighan-Lin Review METiS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_part),with prg_delta = change in obj_value Dequeue and allow hill climbing.

Parameters

gp	Graph partitioning
xadj	CSR array of graph nodes
adjncy	CSR array of graph neighbors
nparts	Number of Parts
partNumber	Partition vector
core_count	Array: number of core vertices in each part
CH_count	Array: number of core+halo vertices in each part
Halo_count	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i\ with k connections
sumCubes	Sum of cubes objective value
maxCh	maximum core-halo part size obective value
nconverg	number of before convergence
seed	random number generator seed

Allocate arrays

Initialize variables

Initialize array of nodes

Randomize nodes

Compute current cost of partition

Choose objective function to minimize

iterate over the columns of the matrix, ie the hyperedges

KL iteration

let min_part be the smallest CH_part

Try and move free nodes to min_part

lock vertices (climb_counter) vertices have been accepted need to lock (climb_counter) vertices Last vertex to be moved is node_backup(climb_counter)

reset

If all vertices locked, go to next iteration

If empty parts exit, place a vertex in max_part there

Place j and it's neighbors that are in the max part into the empty part

Check Convergence

Check empty part exist move nodes from maxpart to empty part

move it neighbor in the max parts to the newpart

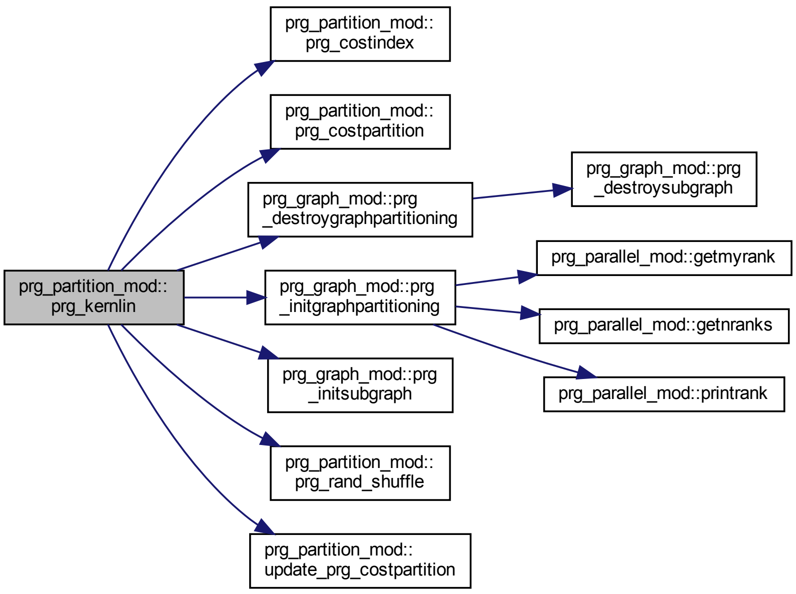
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 758 of file prg_partition_mod.F90.

Here is the call graph for this function:


prg_partition_mod::prg_costindex
prg_partition_mod::prg_costpartition
prg_graph_mod::prg_destroygraphpartitioning
prg_graph_mod::prg_destroysubgraph
prg_parallel_mod::getmyrank
prg_partition_mod::prg_kernlin
prg_graph_mod::prg_initgraphpartitioning
prg_parallel_mod::getnranks
prg_parallel_mod::printrank
prg_graph_mod::prg_initsubgraph
prg_partition_mod::prg_rand_shuffle
prg_partition_mod::update_prg_costpartition

8.20.2.8 prg_kernlin2()

```
subroutine, public prg_partition_mod::prg_kernlin2 (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm )
```

Allocate arrays

Pick hyperedge with largest size or random hyperedge with probability 0.5 We wiil change it to pick hyperedge with highest priority, where priority will be defined according to different metrics

Find part with smalles size (should be included in update_prg_costPartition

if current part is max, move to min_part then move subsets (neighbors)

Move hyperedges to minCH part

Try and move intersecting hyperedges

Move k number of vertices. k should be small i.e k <=20, k set in prg_Kernlin_queue Only use this for small systems

Check empty part exist move nodes from maxpart to empty part

move it neighbor in the max parts to the newpart

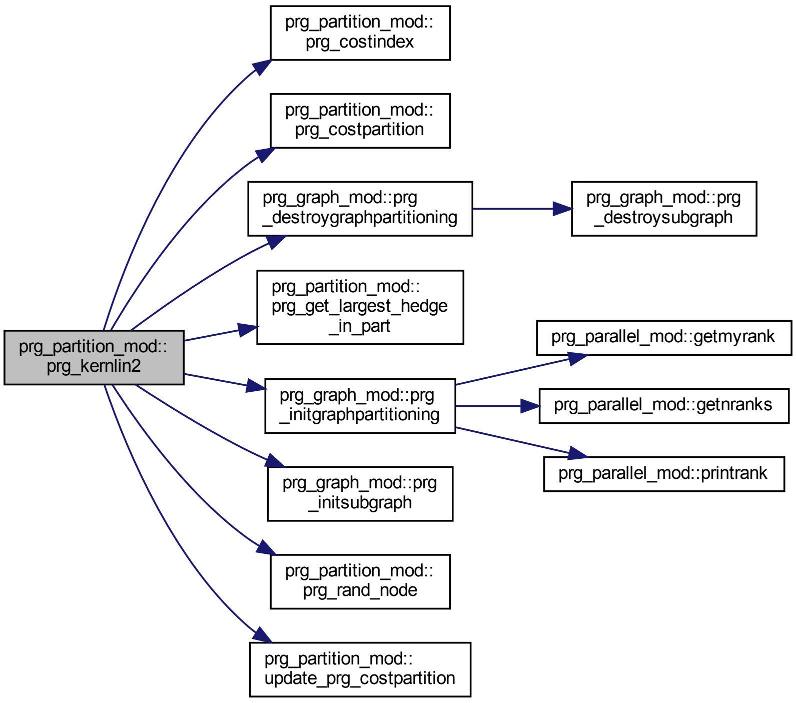
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 1257 of file prg_partition_mod.F90.

Here is the call graph for this function:



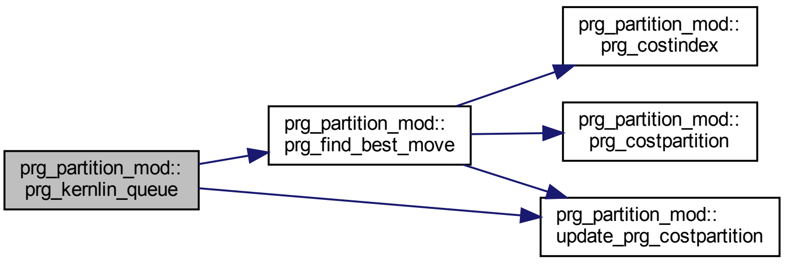
8.20.2.9 prg_kernlin_queue()

```
subroutine, public prg_partition_mod::prg_kernlin_queue (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm )
```

Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain Currently implementation is very slow.

Definition at line 1173 of file prg_partition_mod.F90.

Here is the call graph for this function:



8.20.2.10 prg_metispartition()

```
subroutine, public prg_partition_mod::prg_metispartition (
             type (graph_partitioning_t), intent(inout) gp,
             integer, intent(in) ngroups,
             integer, intent(in) nnodes,
             integer, dimension(:), intent(inout), allocatable xadj,
             integer, dimension(:), intent(inout), allocatable adjncy,
             integer, intent(inout) nparts,
             integer, dimension(:), intent(inout), allocatable part,
             integer, dimension(:), intent(inout), allocatable core_count,
             integer, dimension(:), intent(inout), allocatable CH_count,
             integer, dimension(:,:), intent(inout), allocatable Halo_count,
             real(dp), intent(inout) sumCubes,
             real(dp), intent(inout) maxCH,
             real(dp), intent(inout) smooth_maxCH,
             real(dp), intent(inout) pnorm )
```

Create graph partitions minizing number of cut edges.

Parameters

gp	Graph partitioning`
ngroups	Number of groups/nodes
nnodes	Number of nodes
xadj	CSR array of graph nodes
adjncy	CSR array of graph neighbors
nparts	Number of Parts
part	Partition vector
core_count	Array: number of core vertices in each part
CH_count	Array: number of core+halo vertices in each part
Halo_count	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i\ with k connections
sumCubes	Sum of cubes objective value
maxCh	maximum core-halo part size obective value

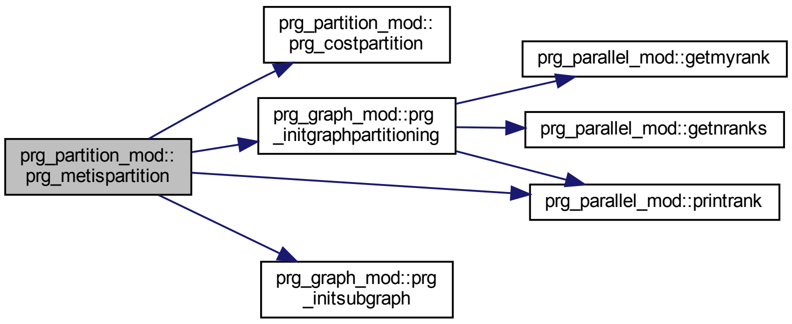
prg_initialize

Partition graph into nparts'

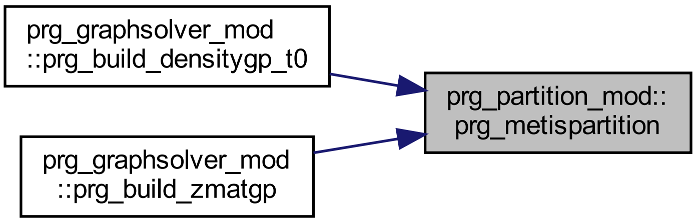
Compute cost of partition

Definition at line 217 of file prg_partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.11 prg_rand_node()

```
subroutine prg_partition_mod::prg_rand_node (
            type (graph_partitioning_t), intent(inout) gp,
            integer, intent(inout) node,
            integer, intent(inout) seed )  [private]
```

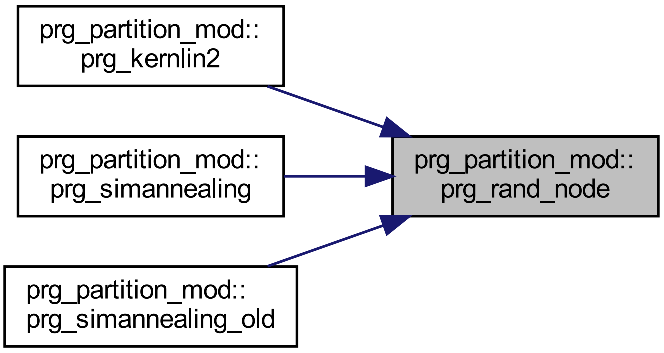
Pick a random node.

Parameters

gp	graph partitioning structure
node	output node
seed	random seed

Definition at line 527 of file prg_partition_mod.F90.

Here is the caller graph for this function:



8.20.2.12 prg_rand_shuffle()

```
subroutine prg_partition_mod::prg_rand_shuffle (
             integer, dimension(:), intent(inout) array,
             integer, intent(inout) seed )  [private]
```

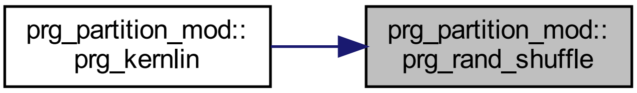
Randomly shuffle array.

Random seed

Shuffle array

Definition at line 1123 of file prg_partition_mod.F90.

Here is the caller graph for this function:



8.20.2.13 prg_simannealing()

```
subroutine, public prg_partition_mod::prg_simannealing (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm,
            integer, intent(in) niter,
            integer, intent(inout) seed )
```

Graph partitioning based on Simulated Annealing.

Parameters

gp	Graph partitioning
xadj	CSR array of graph nodes
adjncy	CSR array of graph neighbors
nparts	Number of Parts
partNumber	Partition vector
core_count	Array: number of core vertices in each part
CH_count	Array: number of core+halo vertices in each part
Halo_count	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i\ with k connections
sumCubes	Sum of cubes objective value
maxCh	maximum core-halo part size obective value
niter	Number of iterations
seed	Random seed

Compute current cost of partition

Choose objective function to minimize

Perform SA

Find part with smalles size (should be included in update_prg_costPartition

if part(node) == max_ch_part, try to move node and it's neighbors to min_ch_part else move neighbors to part(node)

Check empty part exist move nodes from maxpart to empty part

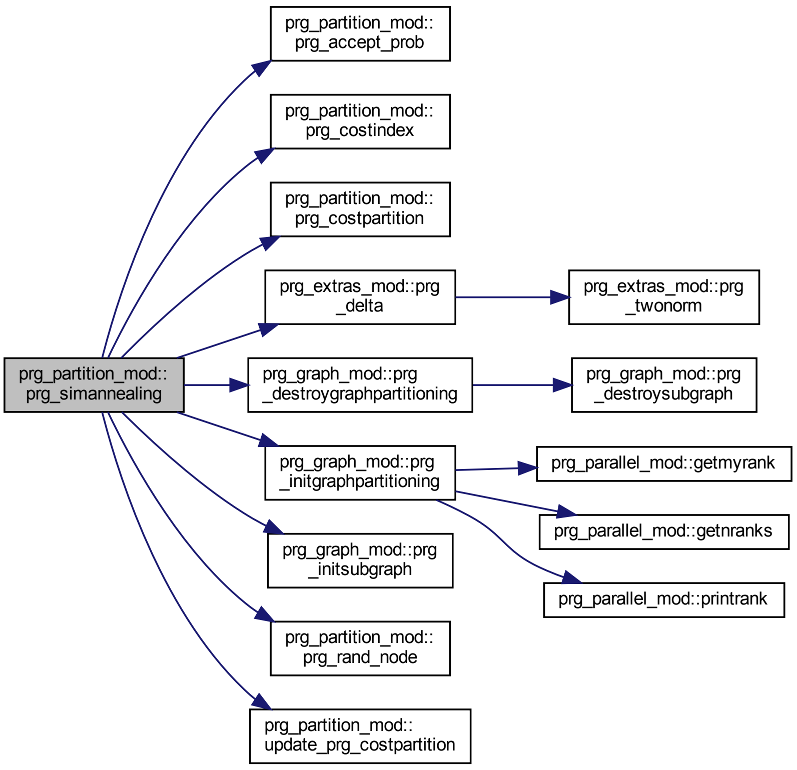
move it neighbor in the max parts to the newpart

Update graph structure

For debuging

Definition at line 552 of file prg_partition_mod.F90.

Here is the call graph for this function:



8.20.2.14 prg_simannealing_old()

```
subroutine, public prg_partition_mod::prg_simannealing_old (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm,
            integer, intent(in) niter,
            integer, intent(inout) seed )
```

Compute current cost of partition

Choose objective function to minimize

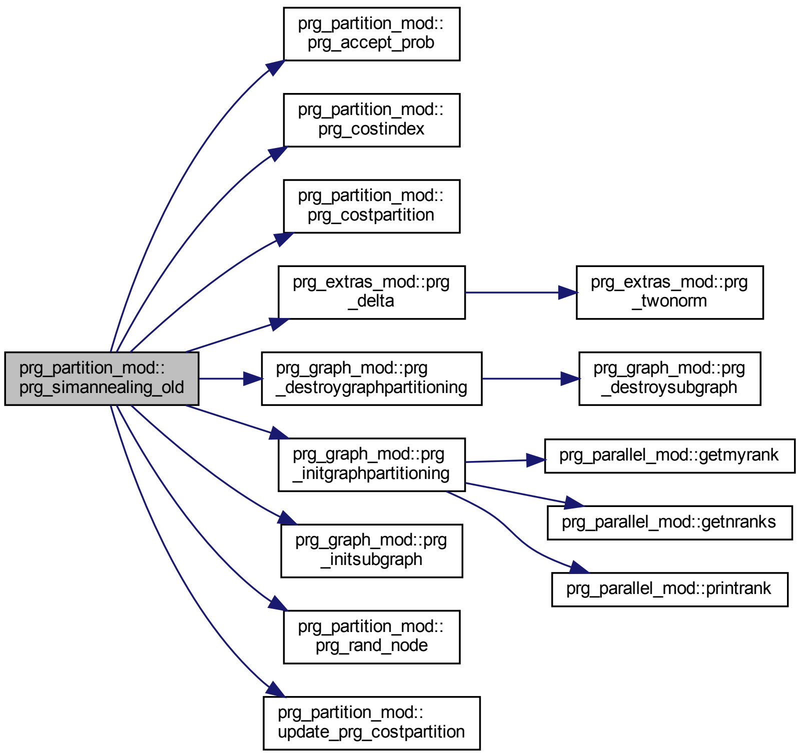
Perform SA

Update graph structure

For debuging

Definition at line 1454 of file prg_partition_mod.F90.

Here is the call graph for this function:



8.20.2.15 prg_update_gp()

```
subroutine, public prg_partition_mod::prg_update_gp (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count )
```

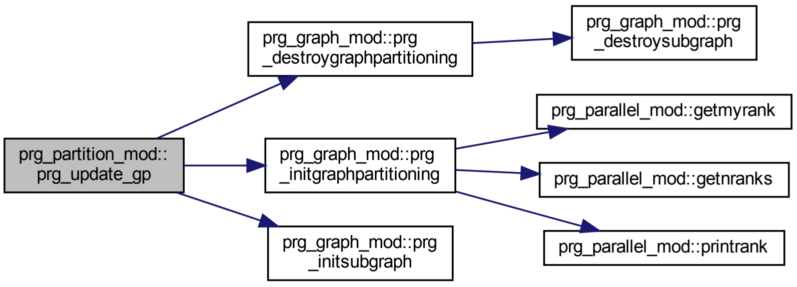
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 1082 of file prg_partition_mod.F90.

Here is the call graph for this function:



8.20.2.16 update_prg_costpartition()

```
subroutine, public prg_partition_mod::update_prg_costpartition (
            type (graph_partitioning_t), intent(inout) gp,
            integer, dimension(:), intent(inout), allocatable xadj,
            integer, dimension(:), intent(inout), allocatable adjncy,
            integer, dimension(:), intent(inout), allocatable partNumber,
            integer, dimension(:), intent(inout), allocatable core_count,
            integer, dimension(:), intent(inout), allocatable CH_count,
            integer, dimension(:,:), intent(inout), allocatable Halo_count,
            real(dp), intent(inout) sumCubes,
            real(dp), intent(inout) maxCH,
            real(dp), intent(inout) smooth_maxCH,
            real(dp), intent(inout) pnorm,
            integer, intent(in) node,
            integer, intent(in) new_part )
```

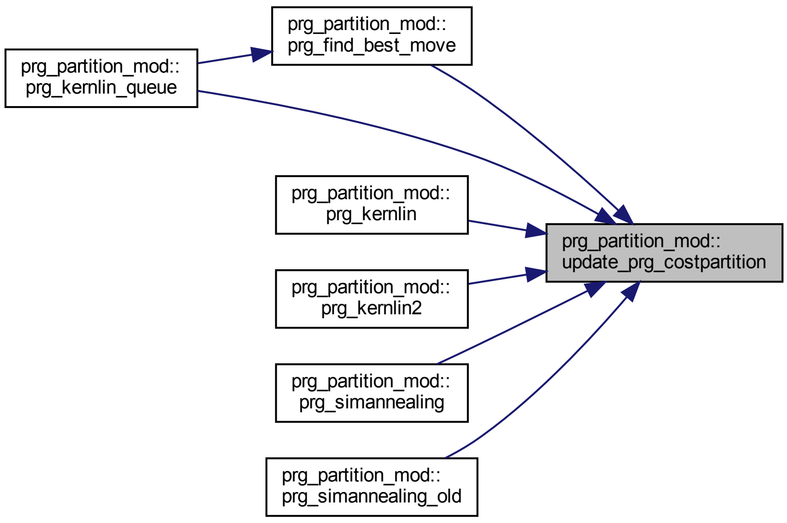
Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.

Parameters

gp	Graph partitioning
xadj	CSR array of graph nodes
adjncy	CSR array of 1043365660.0000000graph neighbors
nparts	Number of Parts
partNumber	Partition vector
core_count	Array: number of core vertices in each part
CH_count	Array: number of core+halo vertices in each part
Halo_count	2D Array of size nparts by totalNodes: Halo_count(i,j) = k, node j is a halo of part i\ with k connections
sumCubes	Sum of cubes objective value
maxCh	maximum core-halo part size obective value
node	Vertex that has moved to new_part
new_part	new part that node has moved to

Definition at line 401 of file prg_partition_mod.F90.

Here is the caller graph for this function:



8.20.3 Variable Documentation

8.20.3.1 dp

```
integer, parameter prg_partition_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_partition_mod.F90.

8.20.3.2 metis_index_kind

```
integer, parameter prg_partition_mod::metis_index_kind = METIS_INDEX_KIND  [private]
```

From /usr/include/metis.h.

IDXTYPEWIDTH = 32 --> metis_index_kind = 4 IDXTYPEWIDTH = 64 --> metis_index_kind = 8

Definition at line 24 of file prg_partition_mod.F90.

8.20.3.3 metis_real_kind

```
integer, parameter prg_partition_mod::metis_real_kind = kind(METIS_REAL_KIND)  [private]
```

From /usr/include/metis.h.

REALTYPEWIDTH = 32 --> metis_real_kind = kind(0e0) REALTYPEWIDTH = 64 --> metis_real_kind = kind(0d0)

Definition at line 30 of file prg_partition_mod.F90.

8.21 prg_progress_mod Module Reference

The progress module.

Functions/Subroutines

- subroutine, public prg_progress_init ()

  Initialize progress.
- subroutine, public prg_progress_shutdown ()

  Shutdown progress.

Variables

- integer, parameter dp = kind(1.0d0)

8.21.1 Detailed Description

The progress module.

8.21.2 Function/Subroutine Documentation

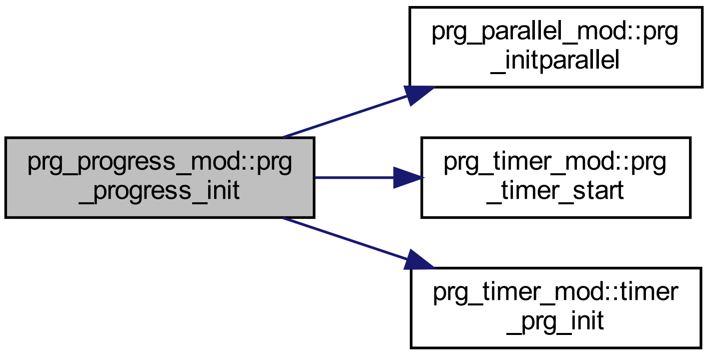
8.21.2.1 prg_progress_init()

`subroutine, public prg_progress_mod::prg_progress_init`

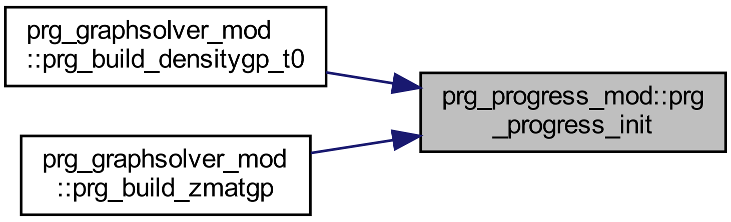
Initialize progress.

Definition at line 25 of file prg_progress_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::prg_initparallel
prg_progress_mod::prg_progress_init
prg_timer_mod::prg_timer_start
prg_timer_mod::timer_prg_init

Here is the caller graph for this function:


prg_graphsolver_mod::prg_build_densitygp_t0
prg_graphsolver_mod::prg_build_zmatgp
prg_progress_mod::prg_progress_init

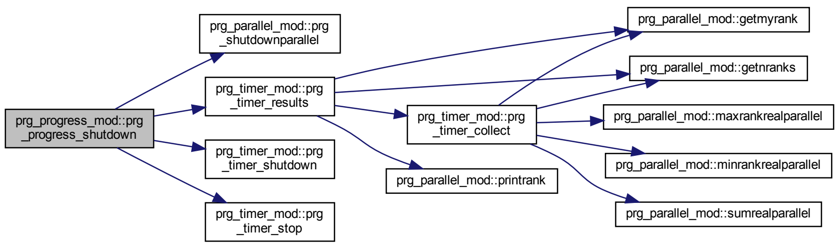
8.21.2.2 prg_progress_shutdown()

`subroutine, public prg_progress_mod::prg_progress_shutdown`

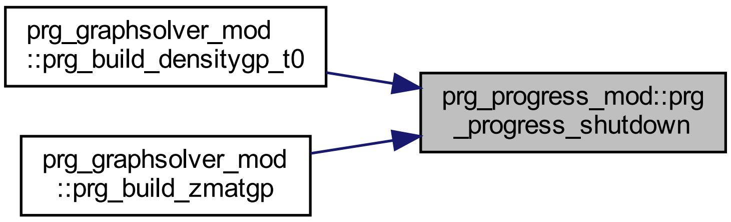
Shutdown progress.

Definition at line 37 of file prg_progress_mod.F90.

Here is the call graph for this function:


prg_progress_mod::prg_progress_shutdown
prg_parallel_mod::prg_shutdownparallel
prg_timer_mod::prg_timer_results
prg_timer_mod::prg_timer_shutdown
prg_timer_mod::prg_timer_stop
prg_timer_mod::prg_timer_collect
prg_parallel_mod::printrank
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::maxrankrealparallel
prg_parallel_mod::minrankrealparallel
prg_parallel_mod::sumrealparallel

Here is the caller graph for this function:


prg_graphsolver_mod::prg_build_densitygp_t0
prg_graphsolver_mod::prg_build_zmatgp
prg_progress_mod::prg_progress_shutdown

8.21.3 Variable Documentation

8.21.3.1 dp

```
integer, parameter prg_progress_mod::dp = kind(1.0d0)  [private]
```

Definition at line 16 of file prg_progress_mod.F90.

8.22 prg_ptable_mod Module Reference

Periodic table of elements.

Functions/Subroutines

- integer function, public element_atomic_number (symbol)
- integer function element_atomic_number_upper (symbol)

Variables

- integer, parameter nz = 103
- integer, parameter, private dp = kind(1.0d0)
- character(2), dimension(nz), parameter element_symbol = [character(2) :: "H" , "He" , "Li" , "Be" , "B" , "C" , "N" , "O" , "F" , "Ne" , "Na" , "Mg" , "Al" , "Si" , "P" , "S" , "Cl" , "Ar" , "K" , "Ca" , "Sc" , "Ti" , "V" , "Cr" , "Mn" , "Fe" , "Co" , "Ni" , "Cu" , "Zn" , "Ga" , "Ge" , "As" , "Se" , "Br" , "Kr" , "Rb" , "Sr" , "Y" , "Zr" , "Nb" , "Mo" , "Tc" , "Ru" , "Rh" , "Pd" , "Ag" , "Cd" , "In" , "Sn" , "Sb" , "Te" , "I" , "Xe" , "Cs" , "Ba" , "La" , "Ce" , "Pr" , "Nd" , "Pm" , "Sm" , "Eu" , "Gd" , "Tb" , "Dy" , "Ho" , "Er" , "Tm" , "Yb" , "Lu" , "Hf" , "Ta" , "W" , "Re" , "Os" , "Ir" , "Pt" , "Au" , "Hg" , "Tl" , "Pb" , "Bi" , "Po" , "At" , "Rn" , "Fr" , "Ra" , "Ac" , "Th" , "Pa" , "U" , "Np" , "Pu" , "Am" , "Cm" , "Bk" , "Cf" , "Es" , "Fm" , "Md" , "No" , "Lr"]

  Element symbol.
- character(2), dimension(nz), parameter element_symbol_upper = [character(2) :: "H" , "HE" , "LI" , "BE" , "B" , "C" , "N" , "O" , "F" , "NE" , "NA" , "MG" , "AL" , "SI" , "P" , "S" , "CL" , "AR" , "K" , "CA" , "SC" , "TI" , "V" , "CR" , "MN" , "FE" , "CO" , "NI" , "CU" , "ZN" , "GA" , "GE" , "AS" , "SE" , "BR" , "KR" , "RB" , "SR" , "Y" , "ZR" , "NB" , "MO" , "TC" , "RU" , "RH" , "PD" , "AG" , "CD" , "IN" , "SN" , "SB" , "TE" , "I" , "XE" , "CS" , "BA" , "LA" , "CE" , "PR" , "ND" , "PM" , "SM" , "EU" , "GD" , "TB" , "DY" , "HO" , "ER" , "TM" , "YB" , "LU" , "HF" , "TA" , "W" , "RE" , "OS" , "IR" , "PT" , "AU" , "HG" , "TL" , "PB" , "BI" , "PO" , "AT" , "RN" , "FR" , "RA" , "AC" , "TH" , "PA" , "U" , "NP" , "PU" , "AM" , "CM" , "BK" , "CF" , "ES" , "FM" , "MD" , "NO" , "LR"]

  Element symbol upper.
- character(20), dimension(nz), parameter element_name = [character(20) :: "Hydrogen" , "Helium" , "Lithium" , "Beryllium" , "Boron" , "Carbon" , "Nitrogen" , "Oxygen" , "Fluorine" , "Neon" , "Sodium" , "Magnesium" , "Aluminium" , "Silicon" , "Phosphorus" , "Sulfur" , "Chlorine" , "Argon" , "Potassium" , "Calcium" , "Scandium" , "Titanium" , "Vanadium" , "Chromium" , "Manganese" , "Iron" , "Cobalt" , "Nickel" , "Copper" , "Zinc" , "Gallium" , "Germanium" , "Arsenic" , "Selenium" , "Bromine" , "Krypton" , "Rubidium" , "Strontium" , "Yttrium" , "Zirconium" , "Niobium" , "Molybdenum" , "Technetium" , "Ruthenium" , "Rhodium" , "Palladium" , "Silver" , "Cadmium" , "Indium" , "Tin" , "Antimony" , "Tellurium" , "Iodine" , "Xenon" , "Caesium" , "Barium" , "Lanthanum" , "Cerium" , "Praseodymium" , "Neodymium" , "Promethium" , "Samarium" , "Europium" , "Gadolinium" , "Terbium" , "Dysprosium" , "Holmium" , "Erbium" , "Thulium" , "Ytterbium" , "Lutetium" , "Hafnium" , "Tantalum" , "Tungsten" , "Rhenium" , "Osmium" , "Iridium" , "Platinum" , "Gold" , "Mercury" , "Thallium" , "Lead" , "Bismuth" , "Polonium" , "Astatine" , "Radon" , "Francium" , "Radium" , "Actinium" , "Thorium" , "Protactinium" , "Uranium" , "Neptunium" , "Plutonium" , "Americium" , "Curium" , "Berkelium" , "Californium" , "Einsteinium" , "Fermium" , "Mendelevium" , "Nobelium" , "Lawrencium"]

  Element name.
- real(dp), dimension(nz), parameter element_mass = (/ 1.007825032 , 4.002603254 , 7.01600455 , 9.0121822 , 11.0093054 , 12.0 , 14.003074005 , 15.99491462 , 18.99840322 , 19.992440175 , 22.989769281 , 23.↩
  9850417 , 26.98153863 , 27.976926532 , 30.97376163 , 31.972071 , 34.96885268 , 39.962383123 , 38.↩
  96370668 , 39.96259098 , 44.9559119 , 47.9479463 , 50.9439595 , 51.9405075 , 54.9380451 , 55.9349375 , 58.933195 , 57.9353429 , 62.9295975 , 63.929142 , 68.925573 , 73.921177 , 74.921596 , 79.916521 , 78.↩
  918337 , 83.911507 , 84.911789 , 87.905612 , 88.905848 , 89.904704 , 92.906378 , 97.905408 , 97.907216 , 101.904349 , 102.905504 , 105.903486 , 106.905097 , 113.903358 , 114.903878 , 119.902194 , 120.↩
  903815 , 129.906224 , 126.904473 , 131.904153 , 132.905451 , 137.905247 , 138.906353 , 139.905438 , 140.907652 , 141.907723 , 144.912749 , 151.919732 , 152.92123 , 157.924103 , 158.925346 , 163.929174 , 164.930322 , 165.930293 , 168.934213 , 173.938862 , 174.940771 , 179.94655 , 180.947995 , 183.950931 , 186.955753 , 191.96148 , 192.962926 , 194.964791 , 196.966568 , 201.970643 , 204.974427 , 207.976652 , 208.980398 , 208.98243 , 209.987148 , 222.017577 , 223.019735 , 226.025409 , 227.027752 , 232.038055 , 231.035884 , 238.050788 , 237.048173 , 244.064204 , 243.061381 , 247.070354 , 247.070307 , 251.079587 , 252.08298 , 257.095105 , 258.098431 , 259.10103 , 262.10963 /)

  Element mass in atomic mass units (1.66 x 10-27 kg)
- real(dp), dimension(nz), parameter element_vdwr = (/ 1.1 , 1.4 , 1.81 , 1.53 , 1.92 , 1.7 , 1.55 , 1.52 , 1.47 , 1.54 , 2.27 , 1.73 , 1.84 , 2.1 , 1.8 , 1.8 , 1.75 , 1.88 , 2.75 , 2.31 , 2.3 , 2.15 , 2.05 , 2.05 , 2.05 , 2.05 , 2.0 , 2.0 , 2.0 , 2.1 , 1.87 , 2.11 , 1.85 , 1.9 , 1.83 , 2.02 , 3.03 , 2.49 , 2.4 , 2.3 , 2.15 , 2.1 , 2.05 , 2.05 , 2.0 , 2.05 , 2.1 , 2.2 , 2.2 , 1.93 , 2.17 , 2.06 , 1.98 , 2.16 , 3.43 , 2.68 , 2.5 , 2.48 , 2.47 , 2.45 , 2.43 , 2.42 , 2.4 , 2.38 , 2.37 , 2.35 , 2.33 , 2.32 , 2.3 , 2.28 , 2.27 , 2.25 , 2.2 , 2.1 , 2.05 , 2.0 , 2.0 , 2.05 , 2.1 , 2.05 , 1.96 , 2.02 , 2.07 , 1.97 , 2.02 , 2.2 , 3.48 , 2.83 , 2.0 , 2.4 , 2.0 , 2.3 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 /)

  van der Waals radius (in Angstroms)

- real(dp), dimension(nz), parameter element_covr = (/ 0.31 , 0.28 , 1.28 , 0.96 , 0.84 , 0.76 , 0.71 , 0.66 , 0.57 , 0.58 , 1.66 , 1.41 , 1.21 , 1.11 , 1.07 , 1.05 , 1.02 , 1.06 , 2.03 , 1.76 , 1.7 , 1.6 , 1.53 , 1.39 , 1.39 , 1.32 , 1.26 , 1.24 , 1.32 , 1.22 , 1.22 , 1.2 , 1.19 , 1.2 , 1.2 , 1.16 , 2.2 , 1.95 , 1.9 , 1.75 , 1.64 , 1.54 , 1.47 , 1.46 , 1.42 , 1.39 , 1.45 , 1.44 , 1.42 , 1.39 , 1.39 , 1.38 , 1.39 , 1.4 , 2.44 , 2.15 , 2.07 , 2.04 , 2.03 , 2.01 , 1.99 , 1.98 , 1.98 , 1.96 , 1.94 , 1.92 , 1.92 , 1.89 , 1.9 , 1.87 , 1.87 , 1.75 , 1.7 , 1.62 , 1.51 , 1.44 , 1.41 , 1.36 , 1.36 , 1.32 , 1.45 , 1.46 , 1.48 , 1.4 , 1.5 , 1.5 , 2.6 , 2.21 , 2.15 , 2.06 , 2.0 , 1.96 , 1.9 , 1.87 , 1.8 , 1.69 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 /)

  Covalent radius (in Angstroms)

- real(dp), dimension(nz), parameter element_ip = (/ 13.5984 , 24.5874 , 5.3917 , 9.3227 , 8.298 , 11.2603 , 14.5341 , 13.6181 , 17.4228 , 21.5645 , 5.1391 , 7.6462 , 5.9858 , 8.1517 , 10.4867 , 10.36 , 12.9676 , 15.7596 , 4.3407 , 6.1132 , 6.5615 , 6.8281 , 6.7462 , 6.7665 , 7.434 , 7.9024 , 7.881 , 7.6398 , 7.7264 , 9.3942 , 5.9993 , 7.8994 , 9.7886 , 9.7524 , 11.8138 , 13.9996 , 4.1771 , 5.6949 , 6.2173 , 6.6339 , 6.7589 , 7.0924 , 7.28 , 7.3605 , 7.4589 , 8.3369 , 7.5762 , 8.9938 , 5.7864 , 7.3439 , 8.6084 , 9.0096 , 10.4513 , 12.1298 , 3.8939 , 5.2117 , 5.5769 , 5.5387 , 5.473 , 5.525 , 5.582 , 5.6437 , 5.6704 , 6.1498 , 5.8638 , 5.9389 , 6.0215 , 6.1077 , 6.1843 , 6.2542 , 5.4259 , 6.8251 , 7.5496 , 7.864 , 7.8335 , 8.4382 , 8.967 , 8.9588 , 9.2255 , 10.4375 , 6.1082 , 7.4167 , 7.2855 , 8.414 , 0.0 , 10.7485 , 4.0727 , 5.2784 , 5.17 , 6.3067 , 5.89 , 6.1941 , 6.2657 , 6.026 , 5.9738 , 5.9914 , 6.1979 , 6.2817 , 6.42 , 6.5 , 6.58 , 6.65 , 4.9 /)

  Ionization energy (in eV)

- real(dp), dimension(nz), parameter element_ea = (/ 0.75420375 , 0.0 , 0.618049 , 0.0 , 0.279723 , 1.262118 , -0.07 , 1.461112 , 3.4011887 , 0.0 , 0.547926 , 0.0 , 0.43283 , 1.389521 , 0.7465 , 2.0771029 , 3.612724 , 0.0 , 0.501459 , 0.02455 , 0.188 , 0.084 , 0.525 , 0.67584 , 0.0 , 0.151 , 0.6633 , 1.15716 , 1.23578 , 0.0 , 0.41 , 1.232712 , 0.814 , 2.02067 , 3.363588 , 0.0 , 0.485916 , 0.05206 , 0.307 , 0.426 , 0.893 , 0.7472 , 0.55 , 1.04638 , 1.14289 , 0.56214 , 1.30447 , 0.0 , 0.404 , 1.112066 , 1.047401 , 1.970875 , 3.059038 , 0.0 , 0.471626 , 0.14462 , 0.47 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.0 , 0.322 , 0.815 , 0.15 , 1.0778 , 1.56436 , 2.1251 , 2.30861 , 0.0 , 0.377 , 0.364 , 0.942363 , 1.9 , 2.8 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 /)

  Electron affprg_inity (in eV)

- real(dp), dimension(nz), parameter atom_en = (/ 2.2 , 0.0 , 0.98 , 1.57 , 2.04 , 2.55 , 3.04 , 3.44 , 3.98 , 0.0 , 0.93 , 1.31 , 1.61 , 1.9 , 2.19 , 2.58 , 3.16 , 0.0 , 0.82 , 1.0 , 1.36 , 1.54 , 1.63 , 1.66 , 1.55 , 1.83 , 1.88 , 1.91 , 1.9 , 1.65 , 1.81 , 2.01 , 2.18 , 2.55 , 2.96 , 3.0 , 0.82 , 0.95 , 1.22 , 1.33 , 1.6 , 2.16 , 1.9 , 2.2 , 2.28 , 2.2 , 1.93 , 1.69 , 1.78 , 1.96 , 2.05 , 2.1 , 2.66 , 2.6 , 0.79 , 0.89 , 1.1 , 1.12 , 1.13 , 1.14 , 0.0 , 1.17 , 0.0 , 1.2 , 0.0 , 1.22 , 1.23 , 1.24 , 1.25 , 0.0 , 1.27 , 1.3 , 1.5 , 2.36 , 1.9 , 2.2 , 2.2 , 2.28 , 2.54 , 2.0 , 1.62 , 2.33 , 2.02 , 2.0 , 2.2 , 0.0 , 0.7 , 0.9 , 1.1 , 1.3 , 1.5 , 1.38 , 1.36 , 1.28 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 0.0 /)

  The Pauling electronegativity for this element.

- integer, dimension(nz), parameter element_maxbonds = (/ 1 , 0 , 1 , 2 , 4 , 4 , 4 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 8 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 12 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 /)

  The maximum expected number of bonds to this element.

- integer, dimension(nz), parameter element_numel = (/ 1 , 2 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 /)

  Last shell number of electrons.

- character(50), dimension(nz), parameter element_econf = [character(50) :: "1s" , "1s2" , "1s22s" , "1s22s2" , "1s22s22p" , "1s22s22p2" , "1s22s22p3" , "1s22s22p4" , "1s22s22p5" , "1s22s22p6" , "[Ne]3s" , "[Ne]3s2" , "[Ne]3s23p" , "[Ne]3s23p2" , "[Ne]3s23p3" , "[Ne]3s23p4" , "[Ne]3s23p5" , "[Ne]3s23p6" , "[Ar]4s" , "[Ar]4s2" , "[Ar]3d4s2" , "[Ar]3d24s2" , "[Ar]3d34s2" , "[Ar]3d54s" , "[Ar]3d54s2" , "[Ar]3d64s2" , "[Ar]3d74s2" , "[Ar]3d84s2" , "[Ar]3d104s" , "[Ar]3d104s2" , "[Ar]3d104s24p" , "[Ar]3d104s24p2" , "[Ar]3d104s24p3" , "[Ar]3d104s24p4" , "[Ar]3d104s24p5" , "[Ar]3d104s24p6" , "[Kr]5s" , "[Kr]5s2" , "[Kr]4d5s2" , "[Kr]4d25s2" , "[Kr]4d45s" , "[Kr]4d55s" , "[Kr]4d55s2" , "[Kr]4d75s" , "[Kr]4d85s" , "[Kr]4d10" , "[Kr]4d105s" , "[Kr]4d105s2" , "[Cd]5p" , "[Cd]5p2" , "[Cd]5p3" , "[Cd]5p4" , "[Cd]5p5" , "[Cd]5p6" , "[Xe]6s" , "[Xe]6s2" , "[Xe]5d6s2" , "[Xe]4f5d6s2" , "[Xe]4f36s2" , "[Xe]4f46s2" , "[Xe]4f56s2" , "[Xe]4f66s2" , "[Xe]4f76s2" , "[Xe]4f75d6s2" , "[Xe]4f96s2" , "[Xe]4f106s2" , "[Xe]4f116s2" , "[Xe]4f126s2" , "[Xe]4f136s2" , "[Xe]4f146s2" , "[Xe]4f145d6s2" , "[Xe]4f145d26s2" , "[Xe]4f145d36s2" , "[Xe]4f145d46s2" , "[Xe]4f145d56s2" , "[Xe]4f145d66s2" , "[Xe]4f145d76s2" , "[Xe]4f145d96s" , "[Xe]4f145d106s" , "[Xe]4f145d106s2" , "[Hg]6p" , "[Hg]6p2" , "[Hg]6p3" , "[Hg]6p4" , "[Hg]6p5" , "[Hg]6p6" , "[Rn]7s" , "[Rn]7s2" , "[Rn]6d7s2" , "[Rn]6d27s2" , "[Rn]5f26d7s2" , "[Rn]5f36d7s2" , "[Rn]5f46d7s2" , "[Rn]5f67s2" , "[Rn]5f77s2" , "[Rn]5f76d7s2" , "[Rn]5f97s2" , "[Rn]5f107s2" , "[Rn]5f117s2" , "[Rn]5f127s2" , "[Rn]5f137s2" , "[Rn]5f147s2" , "[Rn]5f147s27p"]

The electronic configuration.

8.22.1 Detailed Description

Periodic table of elements.

This data was generated with pybabel and openbable packages Openbabel: http://openbabel.org/dev-api/index.shtml Pybel: https://openbabel.org/docs/dev/UseTheLibrary/Python_Pybel.html# Other sources includes NIST: http://www.nist.gov/pml/data/ion_energy.cfm

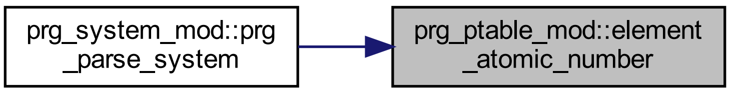
8.22.2 Function/Subroutine Documentation

8.22.2.1 element_atomic_number()

```
integer function, public prg_ptable_mod::element_atomic_number (
            character(len=*) symbol )
```

Definition at line 394 of file prg_ptable_mod.F90.

Here is the caller graph for this function:

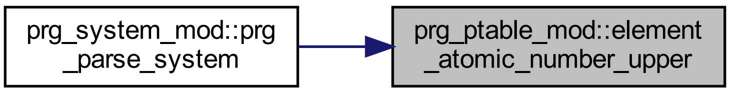

prg_system_mod::prg_parse_system
prg_ptable_mod::element_atomic_number

8.22.2.2 element_atomic_number_upper()

```
integer function prg_ptable_mod::element_atomic_number_upper (
            character(len=*) symbol )
```

Definition at line 408 of file prg_ptable_mod.F90.

Here is the caller graph for this function:


prg_system_mod::prg_parse_system
prg_ptable_mod::element_atomic_number_upper

8.22.3 Variable Documentation

8.22.3.1 atom_en

```
real(dp), dimension(nz), parameter prg_ptable_mod::atom_en = (/ 2.2 , 0.0 , 0.98 , 1.57 , 2.↩
04 , 2.55 , 3.04 , 3.44 , 3.98 , 0.0 , 0.93 , 1.31 , 1.61 , 1.9 , 2.19 , 2.58 , 3.16 , 0.0 ,
0.82 , 1.0 , 1.36 , 1.54 , 1.63 , 1.66 , 1.55 , 1.83 , 1.88 , 1.91 , 1.9 , 1.65 , 1.81 , 2.01
, 2.18 , 2.55 , 2.96 , 3.0 , 0.82 , 0.95 , 1.22 , 1.33 , 1.6 , 2.16 , 1.9 , 2.2 , 2.28 , 2.2
, 1.93 , 1.69 , 1.78 , 1.96 , 2.05 , 2.1 , 2.66 , 2.6 , 0.79 , 0.89 , 1.1 , 1.12 , 1.13 , 1.14
, 0.0 , 1.17 , 0.0 , 1.2 , 0.0 , 1.22 , 1.23 , 1.24 , 1.25 , 0.0 , 1.27 , 1.3 , 1.5 , 2.36 ,
1.9 , 2.2 , 2.2 , 2.28 , 2.54 , 2.0 , 1.62 , 2.33 , 2.02 , 2.0 , 2.2 , 0.0 , 0.7 , 0.9 , 1.1 ,
1.3 , 1.5 , 1.38 , 1.36 , 1.28 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 0.0 /)
```

The Pauling electronegativity for this element.

Definition at line 266 of file prg_ptable_mod.F90.

8.22.3.2 dp

```
integer, parameter, private prg_ptable_mod::dp = kind(1.0d0)  [private]
```

Definition at line 13 of file prg_ptable_mod.F90.

8.22.3.3 element_covr

```
real(dp), dimension(nz), parameter prg_ptable_mod::element_covr = (/ 0.31 , 0.28 , 1.28 , 0.96
, 0.84 , 0.76 , 0.71 , 0.66 , 0.57 , 0.58 , 1.66 , 1.41 , 1.21 , 1.11 , 1.07 , 1.05 , 1.02 ,
1.06 , 2.03 , 1.76 , 1.7 , 1.6 , 1.53 , 1.39 , 1.39 , 1.32 , 1.26 , 1.24 , 1.32 , 1.22 , 1.22
, 1.2 , 1.19 , 1.2 , 1.2 , 1.16 , 2.2 , 1.95 , 1.9 , 1.75 , 1.64 , 1.54 , 1.47 , 1.46 , 1.42 ,
1.39 , 1.45 , 1.44 , 1.42 , 1.39 , 1.39 , 1.38 , 1.39 , 1.4 , 2.44 , 2.15 , 2.07 , 2.04 , 2.↩
03 , 2.01 , 1.99 , 1.98 , 1.98 , 1.96 , 1.94 , 1.92 , 1.92 , 1.89 , 1.9 , 1.87 , 1.87 , 1.75 ,
1.7 , 1.62 , 1.51 , 1.44 , 1.41 , 1.36 , 1.36 , 1.32 , 1.45 , 1.46 , 1.48 , 1.4 , 1.5 , 1.5 ,
2.6 , 2.21 , 2.15 , 2.06 , 2.0 , 1.96 , 1.9 , 1.87 , 1.8 , 1.69 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6
, 1.6 , 1.6 /)
```

Covalent radius (in Angstroms)

Definition at line 173 of file prg_ptable_mod.F90.

8.22.3.4 element_ea

```
real(dp), dimension(nz), parameter prg_ptable_mod::element_ea = (/ 0.75420375 , 0.0 , 0.618049
, 0.0 , 0.279723 , 1.262118 , -0.07 , 1.461112 , 3.4011887 , 0.0 , 0.547926 , 0.0 , 0.43283 ,
1.389521 , 0.7465 , 2.0771029 , 3.612724 , 0.0 , 0.501459 , 0.02455 , 0.188 , 0.084 , 0.525 ,
0.67584 , 0.0 , 0.151 , 0.6633 , 1.15716 , 1.23578 , 0.0 , 0.41 , 1.232712 , 0.814 , 2.02067 ,
3.363588 , 0.0 , 0.485916 , 0.05206 , 0.307 , 0.426 , 0.893 , 0.7472 , 0.55 , 1.04638 , 1.↩
14289 , 0.56214 , 1.30447 , 0.0 , 0.404 , 1.112066 , 1.047401 , 1.970875 , 3.059038 , 0.0 ,
0.471626 , 0.14462 , 0.47 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.↩
5 , 0.5 , 0.5 , 0.5 , 0.0 , 0.322 , 0.815 , 0.15 , 1.0778 , 1.56436 , 2.1251 , 2.30861 , 0.0
, 0.377 , 0.364 , 0.942363 , 1.9 , 2.8 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 ,
0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 /)
```

Electron affprg_inity (in eV)

Definition at line 235 of file prg_ptable_mod.F90.

8.22.3.5 element_econf

```
character(50), dimension(nz), parameter prg_ptable_mod::element_econf = [character(50) ↩
::  "1s" , "1s2" , "1s22s" , "1s22s2" , "1s22s22p" , "1s22s22p2" , "1s22s22p3" , "1s22s22p4" ,
"1s22s22p5" , "1s22s22p6" , "[Ne]3s" , "[Ne]3s2" , "[Ne]3s23p" , "[Ne]3s23p2" , "[Ne]3s23p3" ,
"[Ne]3s23p4" , "[Ne]3s23p5" , "[Ne]3s23p6" , "[Ar]4s" , "[Ar]4s2" , "[Ar]3d4s2" , "[Ar]3d24s2"
, "[Ar]3d34s2" , "[Ar]3d54s" , "[Ar]3d54s2" , "[Ar]3d64s2" , "[Ar]3d74s2" , "[Ar]3d84s2" ,
"[Ar]3d104s" , "[Ar]3d104s2" , "[Ar]3d104s24p" , "[Ar]3d104s24p2" , "[Ar]3d104s24p3" , "[Ar]3d104s24p4"
, "[Ar]3d104s24p5" , "[Ar]3d104s24p6" , "[Kr]5s" , "[Kr]5s2" , "[Kr]4d5s2" , "[Kr]4d25s2" ,
"[Kr]4d45s" , "[Kr]4d55s" , "[Kr]4d55s2" , "[Kr]4d75s" , "[Kr]4d85s" , "[Kr]4d10" , "[Kr]4d105s"
, "[Kr]4d105s2" , "[Cd]5p" , "[Cd]5p2" , "[Cd]5p3" , "[Cd]5p4" , "[Cd]5p5" , "[Cd]5p6" , "[Xe]6s"
, "[Xe]6s2" , "[Xe]5d6s2" , "[Xe]4f5d6s2" , "[Xe]4f36s2" , "[Xe]4f46s2" , "[Xe]4f56s2" , "[Xe]4f66s2"
, "[Xe]4f76s2" , "[Xe]4f75d6s2" , "[Xe]4f96s2" , "[Xe]4f106s2" , "[Xe]4f116s2" , "[Xe]4f126s2"
, "[Xe]4f136s2" , "[Xe]4f146s2" , "[Xe]4f145d6s2" , "[Xe]4f145d26s2" , "[Xe]4f145d36s2" ,
"[Xe]4f145d46s2" , "[Xe]4f145d56s2" , "[Xe]4f145d66s2" , "[Xe]4f145d76s2" , "[Xe]4f145d96s" ,
"[Xe]4f145d106s" , "[Xe]4f145d106s2" , "[Hg]6p" , "[Hg]6p2" , "[Hg]6p3" , "[Hg]6p4" , "[Hg]6p5"
, "[Hg]6p6" , "[Rn]7s" , "[Rn]7s2" , "[Rn]6d7s2" , "[Rn]6d27s2" , "[Rn]5f26d7s2" , "[Rn]5f36d7s2"
, "[Rn]5f46d7s2" , "[Rn]5f67s2" , "[Rn]5f77s2" , "[Rn]5f76d7s2" , "[Rn]5f97s2" , "[Rn]5f107s2"
, "[Rn]5f117s2" , "[Rn]5f127s2" , "[Rn]5f137s2" , "[Rn]5f147s2" , "[Rn]5f147s27p" ]
```

The electronic configuration.

Definition at line 360 of file prg_ptable_mod.F90.

8.22.3.6 element_ip

```
real(dp), dimension(nz), parameter prg_ptable_mod::element_ip = (/ 13.5984 , 24.5874 , 5.3917
, 9.3227 , 8.298 , 11.2603 , 14.5341 , 13.6181 , 17.4228 , 21.5645 , 5.1391 , 7.6462 , 5.9858
, 8.1517 , 10.4867 , 10.36 , 12.9676 , 15.7596 , 4.3407 , 6.1132 , 6.5615 , 6.8281 , 6.7462 ,
6.7665 , 7.434 , 7.9024 , 7.881 , 7.6398 , 7.7264 , 9.3942 , 5.9993 , 7.8994 , 9.7886 , 9.↩
7524 , 11.8138 , 13.9996 , 4.1771 , 5.6949 , 6.2173 , 6.6339 , 6.7589 , 7.0924 , 7.28 , 7.↩
3605 , 7.4589 , 8.3369 , 7.5762 , 8.9938 , 5.7864 , 7.3439 , 8.6084 , 9.0096 , 10.4513 , 12.↩
1298 , 3.8939 , 5.2117 , 5.5769 , 5.5387 , 5.473 , 5.525 , 5.582 , 5.6437 , 5.6704 , 6.1498 ,
5.8638 , 5.9389 , 6.0215 , 6.1077 , 6.1843 , 6.2542 , 5.4259 , 6.8251 , 7.5496 , 7.864 , 7.↩
8335 , 8.4382 , 8.967 , 8.9588 , 9.2255 , 10.4375 , 6.1082 , 7.4167 , 7.2855 , 8.414 , 0.0 ,
10.7485 , 4.0727 , 5.2784 , 5.17 , 6.3067 , 5.89 , 6.1941 , 6.2657 , 6.026 , 5.9738 , 5.9914 ,
6.1979 , 6.2817 , 6.42 , 6.5 , 6.58 , 6.65 , 4.9 /)
```

Ionization energy (in eV)

Definition at line 204 of file prg_ptable_mod.F90.

8.22.3.7 element_mass

```
real(dp), dimension(nz), parameter prg_ptable_mod::element_mass = (/ 1.007825032 , 4.002603254
, 7.01600455 , 9.0121822 , 11.0093054 , 12.0 , 14.003074005 , 15.99491462 , 18.99840322 ,
19.992440175 , 22.989769281 , 23.9850417 , 26.98153863 , 27.976926532 , 30.97376163 , 31.↩
972071 , 34.96885268 , 39.962383123 , 38.96370668 , 39.96259098 , 44.9559119 , 47.9479463 ,
50.9439595 , 51.9405075 , 54.9380451 , 55.9349375 , 58.933195 , 57.9353429 , 62.9295975 ,
63.929142 , 68.925573 , 73.921177 , 74.921596 , 79.916521 , 78.918337 , 83.911507 , 84.911789
, 87.905612 , 88.905848 , 89.904704 , 92.906378 , 97.905408 , 97.907216 , 101.904349 , 102.↩
905504 , 105.903486 , 106.905097 , 113.903358 , 114.903878 , 119.902194 , 120.903815 , 129.↩
906224 , 126.904473 , 131.904153 , 132.905451 , 137.905247 , 138.906353 , 139.905438 , 140.↩
907652 , 141.907723 , 144.912749 , 151.919732 , 152.92123 , 157.924103 , 158.925346 , 163.↩
929174 , 164.930322 , 165.930293 , 168.934213 , 173.938862 , 174.940771 , 179.94655 , 180.↩
947995 , 183.950931 , 186.955753 , 191.96148 , 192.962926 , 194.964791 , 196.966568 , 201.↩
970643 , 204.974427 , 207.976652 , 208.980398 , 208.98243 , 209.987148 , 222.017577 , 223.↩
019735 , 226.025409 , 227.027752 , 232.038055 , 231.035884 , 238.050788 , 237.048173 , 244.↩
064204 , 243.061381 , 247.070354 , 247.070307 , 251.079587 , 252.08298 , 257.095105 , 258.↩
098431 , 259.10103 , 262.10963 /)
```

Element mass in atomic mass units (1.66 x 10-27 kg)

Definition at line 110 of file prg_ptable_mod.F90.

8.22.3.8 element_maxbonds

```
integer, dimension(nz), parameter prg_ptable_mod::element_maxbonds = (/ 1 , 0 , 1 , 2 , 4 , 4
, 4 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 8 , 6 , 6 , 6 , 6 ,
6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1
, 0 , 1 , 2 , 12 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 ,
6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6
, 6 , 6 , 6 /)
```

The maximum expected number of bonds to this element.

Definition at line 297 of file prg_ptable_mod.F90.

8.22.3.9 element_name

```
character(20), dimension(nz), parameter prg_ptable_mod::element_name = [character(20) ::  "Hydrogen"
, "Helium" , "Lithium" , "Beryllium" , "Boron" , "Carbon" , "Nitrogen" , "Oxygen" , "Fluorine"
, "Neon" , "Sodium" , "Magnesium" , "Aluminium" , "Silicon" , "Phosphorus" , "Sulfur" , "Chlorine"
, "Argon" , "Potassium" , "Calcium" , "Scandium" , "Titanium" , "Vanadium" , "Chromium" ,
"Manganese" , "Iron" , "Cobalt" , "Nickel" , "Copper" , "Zinc" , "Gallium" , "Germanium" ,
"Arsenic" , "Selenium" , "Bromine" , "Krypton" , "Rubidium" , "Strontium" , "Yttrium" , "Zirconium"
, "Niobium" , "Molybdenum" , "Technetium" , "Ruthenium" , "Rhodium" , "Palladium" , "Silver"
, "Cadmium" , "Indium" , "Tin" , "Antimony" , "Tellurium" , "Iodine" , "Xenon" , "Caesium" ,
"Barium" , "Lanthanum" , "Cerium" , "Praseodymium" , "Neodymium" , "Promethium" , "Samarium"
, "Europium" , "Gadolinium" , "Terbium" , "Dysprosium" , "Holmium" , "Erbium" , "Thulium"
, "Ytterbium" , "Lutetium" , "Hafnium" , "Tantalum" , "Tungsten" , "Rhenium" , "Osmium" ,
"Iridium" , "Platinum" , "Gold" , "Mercury" , "Thallium" , "Lead" , "Bismuth" , "Polonium"
, "Astatine" , "Radon" , "Francium" , "Radium" , "Actinium" , "Thorium" , "Protactinium" ,
"Uranium" , "Neptunium" , "Plutonium" , "Americium" , "Curium" , "Berkelium" , "Californium"
, "Einsteinium" , "Fermium" , "Mendelevium" , "Nobelium" , "Lawrencium" ]
```

Element name.

Definition at line 79 of file prg_ptable_mod.F90.

8.22.3.10 element_numel

```
integer, dimension(nz), parameter prg_ptable_mod::element_numel = (/ 1 , 2 , 1 , 2 , 3 , 4 , 5
, 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12
, 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 ,
15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17
, 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 1 , 2 , 3 , 4 , 5
, 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 /)
```

Last shell number of electrons.

Definition at line 329 of file prg_ptable_mod.F90.

8.22.3.11 element_symbol

```
character(2), dimension(nz), parameter prg_ptable_mod::element_symbol = [character(2) ::  "H"
, "He" , "Li" , "Be" , "B" , "C" , "N" , "O" , "F" , "Ne" , "Na" , "Mg" , "Al" , "Si" , "P" ,
"S" , "Cl" , "Ar" , "K" , "Ca" , "Sc" , "Ti" , "V" , "Cr" , "Mn" , "Fe" , "Co" , "Ni" , "Cu"
, "Zn" , "Ga" , "Ge" , "As" , "Se" , "Br" , "Kr" , "Rb" , "Sr" , "Y" , "Zr" , "Nb" , "Mo" ,
"Tc" , "Ru" , "Rh" , "Pd" , "Ag" , "Cd" , "In" , "Sn" , "Sb" , "Te" , "I" , "Xe" , "Cs" , "Ba"
, "La" , "Ce" , "Pr" , "Nd" , "Pm" , "Sm" , "Eu" , "Gd" , "Tb" , "Dy" , "Ho" , "Er" , "Tm" ,
"Yb" , "Lu" , "Hf" , "Ta" , "W" , "Re" , "Os" , "Ir" , "Pt" , "Au" , "Hg" , "Tl" , "Pb" , "Bi"
, "Po" , "At" , "Rn" , "Fr" , "Ra" , "Ac" , "Th" , "Pa" , "U" , "Np" , "Pu" , "Am" , "Cm" ,
"Bk" , "Cf" , "Es" , "Fm" , "Md" , "No" , "Lr" ]
```

Element symbol.

Definition at line 17 of file prg_ptable_mod.F90.

8.22.3.12 element_symbol_upper

```
character(2), dimension(nz), parameter prg_ptable_mod::element_symbol_upper = [character(2) ↩
:: "H" , "HE" , "LI" , "BE" , "B" , "C" , "N" , "O" , "F" , "NE" , "NA" , "MG" , "AL" , "SI"
, "P" , "S" , "CL" , "AR" , "K" , "CA" , "SC" , "TI" , "V" , "CR" , "MN" , "FE" , "CO" , "NI"
, "CU" , "ZN" , "GA" , "GE" , "AS" , "SE" , "BR" , "KR" , "RB" , "SR" , "Y" , "ZR" , "NB" ,
"MO" , "TC" , "RU" , "RH" , "PD" , "AG" , "CD" , "IN" , "SN" , "SB" , "TE" , "I" , "XE" , "CS"
, "BA" , "LA" , "CE" , "PR" , "ND" , "PM" , "SM" , "EU" , "GD" , "TB" , "DY" , "HO" , "ER" ,
"TM" , "YB" , "LU" , "HF" , "TA" , "W" , "RE" , "OS" , "IR" , "PT" , "AU" , "HG" , "TL" , "PB"
, "BI" , "PO" , "AT" , "RN" , "FR" , "RA" , "AC" , "TH" , "PA" , "U" , "NP" , "PU" , "AM" ,
"CM" , "BK" , "CF" , "ES" , "FM" , "MD" , "NO" , "LR" ]
```

Element symbol upper.

Definition at line 48 of file prg_ptable_mod.F90.

8.22.3.13 element_vdwr

```
real(dp), dimension(nz), parameter prg_ptable_mod::element_vdwr = (/ 1.1 , 1.4 , 1.81 , 1.53
, 1.92 , 1.7 , 1.55 , 1.52 , 1.47 , 1.54 , 2.27 , 1.73 , 1.84 , 2.1 , 1.8 , 1.8 , 1.75 , 1.88
, 2.75 , 2.31 , 2.3 , 2.15 , 2.05 , 2.05 , 2.05 , 2.05 , 2.0 , 2.0 , 2.0 , 2.1 , 1.87 , 2.11
, 1.85 , 1.9 , 1.83 , 2.02 , 3.03 , 2.49 , 2.4 , 2.3 , 2.15 , 2.1 , 2.05 , 2.05 , 2.0 , 2.05 ,
2.1 , 2.2 , 2.2 , 1.93 , 2.17 , 2.06 , 1.98 , 2.16 , 3.43 , 2.68 , 2.5 , 2.48 , 2.47 , 2.45 ,
2.43 , 2.42 , 2.4 , 2.38 , 2.37 , 2.35 , 2.33 , 2.32 , 2.3 , 2.28 , 2.27 , 2.25 , 2.2 , 2.1 ,
2.05 , 2.0 , 2.0 , 2.05 , 2.1 , 2.05 , 1.96 , 2.02 , 2.07 , 1.97 , 2.02 , 2.2 , 3.48 , 2.83 ,
2.0 , 2.4 , 2.0 , 2.3 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 /)
```

van der Waals radius (in Angstroms)

Definition at line 141 of file prg_ptable_mod.F90.

8.22.3.14 nz

```
integer, parameter prg_ptable_mod::nz = 103
```

Definition at line 12 of file prg_ptable_mod.F90.

8.23 prg_pulaycomponent_mod Module Reference

Produces a matrix to get the Pulay Component of the forces.

Functions/Subroutines

- subroutine, public prg_pulaycomponent0 (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)

  At $T = 0K, P = \rho H \rho$.
- subroutine, public prg_pulaycomponentt (rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)

  At $T > 0K, P = \rho H S^-1 + S^{-1} H \rho$.
- subroutine, public prg_get_pulayforce (nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, dSz_bml, hindex, FPUL, threshold)

  Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}]$.

Variables

- integer, parameter dp = kind(1.0d0)

8.23.1 Detailed Description

Produces a matrix to get the Pulay Component of the forces.

For a further explanation please see Niklasson 2008 [3]

8.23.2 Function/Subroutine Documentation

8.23.2.1 prg_get_pulayforce()

```
subroutine, public prg_pulaycomponent_mod::prg_get_pulayforce (
            integer, intent(in) nats,
            type(bml_matrix_t), intent(in) zmat_bml,
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(in) rho_bml,
            type(bml_matrix_t), intent(in) dSx_bml,
            type(bml_matrix_t), intent(in) dSy_bml,
            type(bml_matrix_t), intent(in) dSz_bml,
            integer, dimension(:,:), intent(in) hindex,
            real(dp), dimension(:,:), intent(inout), allocatable FPUL,
            real(dp), intent(in) threshold )
```

Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}]$.

Parameters

nats	Number of atoms.
zmat_bml	Congruence transform in bml format.
rho_bml	Density matrix.
dSx_bml	x derivative of S.
dSy_bml	y derivative of S.
dSz_bml	z derivative of S.
hindex	Contains the Hamiltonian indices for every atom (see get_hindex).

Definition at line 152 of file prg_pulaycomponent_mod.F90.

8.23.2.2 prg_pulaycomponent0()

```
subroutine, public prg_pulaycomponent_mod::prg_pulaycomponent0 (
            type(bml_matrix_t), intent(in) rho_bml,
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(inout) pcm_bml,
            real(dp), intent(in) threshold,
            integer, intent(in) M,
            character(20), intent(in) bml_type,
            integer verbose )
```

At $T = 0K$, $P = \rho H \rho$.

Parameters

rho_bml	Density matrix in bml format.
ham_bml	Hamiltonian matrix in bml format.
pcm_bml	Pulay matix output in bml format.
threshold	Threshold for the matrix elements.
M	Maximum nonzero values per row.
bml_type	Bml format type.
verbose	Verbosity level.

Todo M and bml_type will have to be removed from the input parameter.

Definition at line 32 of file prg_pulaycomponent_mod.F90.

8.23.2.3 prg_pulaycomponentt()

```
subroutine, public prg_pulaycomponent_mod::prg_pulaycomponentt (
             type(bml_matrix_t), intent(in) rho_bml,
             type(bml_matrix_t), intent(in) ham_bml,
             type(bml_matrix_t), intent(in) zmat_bml,
             type(bml_matrix_t), intent(inout) pcm_bml,
             real(dp), intent(in) threshold,
             integer, intent(in) M,
             character(20), intent(in) bml_type,
             integer verbose )
```

At $T > 0K$, $P = \rho H S^-1 + S^{-1} H \rho$.

Parameters

rho_bml	Density matrix in bml format.
ham_bml	Hamiltonian matrix in bml format.
Z_bml	Congruence transform in bml format.
pcm_bml	Pulay matrix output in bml format.
threshold	Threshold for the matrix elements.
M	Maximum nonzero values per row.
bml_type	Bml format type.
verbose	Verbosity level.

Todo M and bml_type will have to be removed from the input parameter.

Definition at line 83 of file prg_pulaycomponent_mod.F90.

8.23.3 Variable Documentation

8.23.3.1 dp

```
integer, parameter prg_pulaycomponent_mod::dp = kind(1.0d0)  [private]
```

Definition at line 13 of file prg_pulaycomponent_mod.F90.

8.24 prg_pulaymixer_mod Module Reference

Pulay mixer mode.

Data Types

- type mx_type

Functions/Subroutines

- subroutine, public prg_parse_mixer (input, filename)

  The parser for the mixer routines.
- subroutine, public prg_qmixer (charges, oldcharges, dqin, dqout, scferror, piter, pulaycoef, mpulay, verbose)

  Mixing the charges to acelerate scf convergence.
- subroutine, public prg_linearmixer (charges, oldcharges, scferror, linmixcoef, verbose)

  Routine to perform linear mixing.

Variables

- integer, parameter dp = kind(1.0d0)

8.24.1 Detailed Description

Pulay mixer mode.

Gets the best coefficient for mixing the charges during scf.

Todo add the density matrix mixer.

8.24.2 Function/Subroutine Documentation

8.24.2.1 prg_linearmixer()

```
subroutine, public prg_pulaymixer_mod::prg_linearmixer (
             real(dp), dimension(:), intent(inout), allocatable charges,
             real(dp), dimension(:), intent(inout), allocatable oldcharges,
             real(dp), intent(inout) scferror,
             real(dp), intent(in) linmixcoef,
             integer, intent(in) verbose )
```

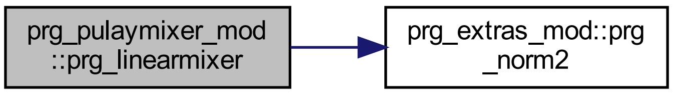
Routine to perform linear mixing.

Parameters

charges	Actual charges of the system.
oldcharges	Previous scf charges.
scferror	SCF error.
linmixcoef	Mixing coefficient.
verbose	Verbosity level.

Definition at line 238 of file prg_pulaymixer_mod.F90.

Here is the call graph for this function:


prg_pulaymixer_mod ::prg_linearmixer
prg_extras_mod::prg _norm2

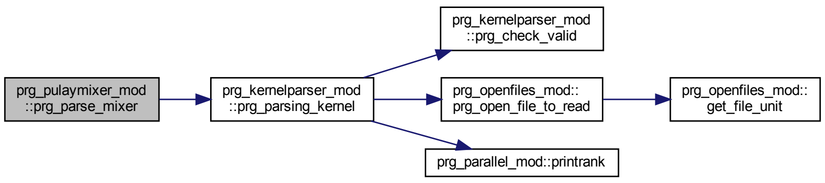
8.24.2.2 prg_parse_mixer()

```
subroutine, public prg_pulaymixer_mod::prg_parse_mixer (
             type(mx_type), intent(inout) input,
             character(len=*) filename )
```

The parser for the mixer routines.

Definition at line 43 of file prg_pulaymixer_mod.F90.

Here is the call graph for this function:


prg_pulaymixer_mod ::prg_parse_mixer
prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

8.24.2.3 prg_qmixer()

```
subroutine, public prg_pulaymixer_mod::prg_qmixer (
             real(dp), dimension(:), intent(inout) charges,
             real(dp), dimension(:), intent(inout), allocatable oldcharges,
             real(dp), dimension(:,:), intent(inout), allocatable dqin,
             real(dp), dimension(:,:), intent(inout), allocatable dqout,
             real(dp), intent(inout) scferror,
             integer piter,
             real(dp), intent(in) pulaycoef,
             integer, intent(in) mpulay,
             integer, intent(in) verbose )
```

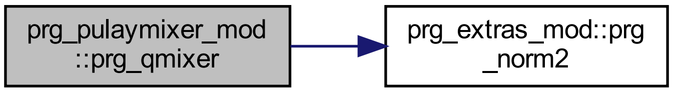
Mixing the charges to acelerate scf convergence.

Parameters

charges	System charges.
oldcharges	Old charges of the system.
dqin	Matrix for charges history in.
dqout	Matrix for charges history out.
scferror	SCF error.
piter	scf iteration number.
pulaycoef	Coefficient for pulay mixing (generally between 0.01 and 0.1).
mpulay	Number of matrices stored (generally 3-5).
verbose	Different levels of verbosity.

Definition at line 104 of file prg_pulaymixer_mod.F90.

Here is the call graph for this function:



8.24.3 Variable Documentation

8.24.3.1 dp

```
integer, parameter prg_pulaymixer_mod::dp = kind(1.0d0)  [private]
```

Definition at line 15 of file prg_pulaymixer_mod.F90.

8.25 prg_quantumdynamics_mod Module Reference

A module to add in common quantum dynamical operations.

Functions/Subroutines

- subroutine, public prg_kick_density (kick_direc, kick_mag, dens, norbs, mdim, S, SINV, which_atom, r, bml-type, thresh)

  Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\rho_{\hat{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ *where* $\hat{V}$ *is the field disturbance.*
- subroutine, public prg_get_sparsity_cplxmat (matrix_type, element_type, thresh, a_dense)

  This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ *where* f *is the sparsity,* N_0 *is the number of values less than the threshold, and* N_{tot} *is the total number of values. The sparsity and threshold are printed to the screen.*
- subroutine, public prg_get_sparsity_realmat (matrix_type, element_type, thresh, a_dense)

  This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ *where* f *is the sparsity,* N_0 *is the number of values less than the threshold, and* N_{tot} *is the total number of values. The sparsity and threshold are printed to the screen.*
- subroutine, public prg_kick_density_bml (kick_direc, kick_mag, rho_bml, s_bml, sinv_bml, mdim, which_↩ atom, r, matrix_type, thresh)

  Provides perturbation to initial density matrix in the form of an electric field kick given input matricies in BML format. This routine does: $\rho_{\hat{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ *where* $\hat{V}$ *is the field disturbance.*
- subroutine, public prg_lvni_bml (h1_bml, sinv_bml, dt, hbar, rhoold_bml, rho_bml, aux_bml, matrix_type, mdim, thresh)

  Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t+\Delta t) = \hat{\rho}(t-\Delta t) + 2\Delta t\frac{\partial\hat{\rho}(t)}{\partial t}$ *where the time derivative of the density matrix is defined as follows:* $\frac{\partial\hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar}\left(S^{-1}\hat{H}(t)\hat{\rho}(t) - \hat{\rho}(t)\hat{H}(t)S^{-1}\right)$.

- subroutine, public prg_getcharge (rho_bml, s_bml, charges, aux_bml, z, spindex, N, nats, thresh)

  Constructs the charges from the density matrix.
- subroutine, public prg_getdipole (charges, r, mu)

  This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.
- subroutine, public prg_excitation (fill_mat, orbit_orig, orbit_exci)

  Produce an excitation in the initially calculated density matrix to.

Variables

- integer, parameter dp = kind(1.0d0)

8.25.1 Detailed Description

A module to add in common quantum dynamical operations.

This module contains routines that perform the following tasks: apply a apply an excitation or perturbation to the initial density matrix, compute the comutator of two two matricies, calculate the sparsity of a real or complex matrix, and time evolve a density matrix using Liouville-von Neumann equation with the leap-frog method of integration.

8.25.2 Function/Subroutine Documentation

8.25.2.1 prg_excitation()

```
subroutine, public prg_quantumdynamics_mod::prg_excitation (
             integer, dimension(:), intent(inout) fill_mat,
             integer, intent(in) orbit_orig,
             integer, intent(in) orbit_exci )
```

Produce an excitation in the initially calculated density matrix to.

Definition at line 307 of file prg_quantumdynamics_mod.F90.

8.25.2.2 prg_get_sparsity_cplxmat()

```
subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_cplxmat (
             character(len=*), intent(in) matrix_type,
             character(len=*), intent(in) element_type,
             real(dp), intent(in) thresh,
             complex(dp), dimension(:,:), intent(in) a_dense )
```

This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.

Parameters

matrix_type	the BML matrix type
element_type	the BML element type
thresh	the threshold for sparsity evaluation
a_dense	the dense complex matrix to be evaluated for sparsity

Definition at line 98 of file prg_quantumdynamics_mod.F90.

8.25.2.3 prg_get_sparsity_realmat()

```
subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_realmat (
             character(len=*), intent(in) matrix_type,
             character(len=*), intent(in) element_type,
             real(dp), intent(in) thresh,
             real(dp), dimension(:,:), intent(in) a_dense )
```

This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ where f is the sparsity, N_0 is the number of values less than the threshold, and N_{tot} is the total number of values. The sparsity and threshold are printed to the screen.

Parameters

matrix_type	the BML matrix type
element_type	the BML element type
thresh	the threshold for sparsity evaluation
a_dense	the dense real matrix to be evaluated for sparsity

Definition at line 122 of file prg_quantumdynamics_mod.F90.

8.25.2.4 prg_getcharge()

```
subroutine, public prg_quantumdynamics_mod::prg_getcharge (
             type(bml_matrix_t), intent(in) rho_bml,
             type(bml_matrix_t), intent(in) s_bml,
             real(dp), dimension(:), allocatable charges,
             type(bml_matrix_t) aux_bml,
             real(dp), dimension(:), intent(in) z,
             integer, dimension(:), intent(in), allocatable spindex,
             integer, dimension(:), intent(in), allocatable N,
             integer nats,
             real(dp), intent(in) thresh )
```

Constructs the charges from the density matrix.

Parameters

rho_bml	Density matrix in BML format.
over_bml	Overlap matrix in BML format.
charges	the array of charges.
aux_bml	the auxiliary matrix in BML format.
spindex	Start and end index for every atom in the system.
z	
nats	the number of atoms
N	
thresh	threshold for the BML matrix

Definition at line 247 of file prg_quantumdynamics_mod.F90.

8.25.2.5 prg_getdipole()

```
subroutine, public prg_quantumdynamics_mod::prg_getdipole (
             real(dp), dimension(:), intent(in) charges,
             real(dp), dimension(:,:), intent(in) r,
             real(dp), dimension(3), intent(inout) mu )
```

This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.

Parameters

charges	Charge on each atom.
r	Coordinate matrix of the atoms.
p	Dipole moment vector.

Definition at line 282 of file prg_quantumdynamics_mod.F90.

8.25.2.6 prg_kick_density()

```
subroutine, public prg_quantumdynamics_mod::prg_kick_density (
             integer, intent(in) kick_direc,
             real(dp) kick_mag,
             complex(dp), dimension(:,:), intent(inout), allocatable dens,
             integer, intent(in) norbs,
             integer, intent(in) mdim,
             complex(dp), dimension(:,:), allocatable S,
             complex(dp), dimension(:,:), allocatable SINV,
             integer, dimension(:), intent(in), allocatable which_atom,
             real(dp), dimension(:,:), allocatable r,
             character(len=*), intent(in) bmltype,
             real(dp) thresh )
```

Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\hat{\rho_{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ where $\hat{V}$ is the field disturbance.

Parameters

kick_direc	the direction of the kick in the electric field
kick_mag	the magnitude of the kick in the electric field
dens	the initial density matrix to be kicked.
norbs	the number of orbitals in the density matrix
S	the overlap matrix
SINV	the inverse of the overlap matrix
which_atom	vector containing atom identification
r	direction vector for kick based on atom and kick_direc
bmltype	type of BML matrix desired for faster computation
thresh	threshold for BML matrix conversion

Definition at line 43 of file prg_quantumdynamics_mod.F90.

8.25.2.7 prg_kick_density_bml()

```
subroutine, public prg_quantumdynamics_mod::prg_kick_density_bml (
            integer, intent(in) kick_direc,
            real(dp) kick_mag,
            type(bml_matrix_t) rho_bml,
            type(bml_matrix_t) s_bml,
            type(bml_matrix_t) sinv_bml,
            integer mdim,
            integer, dimension(:), intent(in), allocatable which_atom,
            real(dp), dimension(:,:), allocatable r,
            character(len=*), intent(in) matrix_type,
            real(dp) thresh )
```

Provides perturbation to initial density matrix in the form of an electric field kick given input matricies in BML format. This routine does: $\rho_{\hat{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ where $\hat{V}$ is the field disturbance.

Parameters

kick_direc	the direction of the kick in the electric field
kick_mag	the magnitude of the kick in the electric field
rho_bml	the initial density matrix to be kicked in BML format.
s_bml	the overlap matrix
sinv_bml	the inverse of the overlap matrix
mdim	maximum number of nonzero values per row in BML matrix
which_atom	vector containing atom identification
r	position vector for kicked atom
matrix_type	the type of BML format
thresh	the threshold for the BML matrix

Definition at line 154 of file prg_quantumdynamics_mod.F90.

8.25.2.8 prg_lvni_bml()

```
subroutine, public prg_quantumdynamics_mod::prg_lvni_bml (
            type(bml_matrix_t) h1_bml,
            type(bml_matrix_t) sinv_bml,
            real(dp) dt,
            real(dp) hbar,
            type(bml_matrix_t) rhoold_bml,
            type(bml_matrix_t) rho_bml,
            type(bml_matrix_t) aux_bml,
            character(len=*), intent(in) matrix_type,
            integer mdim,
            real(dp), intent(in) thresh )
```

Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t+\Delta t) = \hat{\rho}(t-\Delta t) + 2\Delta t \frac{\partial\hat{\rho}(t)}{\partial t}$ where the time derivative of the density matrix is defined as follows: $\frac{\partial\hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar}\left(S^{-1}\hat{H}(t)\hat{\rho}(t) - \hat{\rho}(t)\hat{H}(t)S^{-1}\right)$.

Parameters

H	the Hamiltonian matrix at time t
sinv_bml	the inverse overlap matrix
dt	the timestep for integration
hbar	the Dirac constant (generally taken to be 1 in simulation units)
rho_old	the density matrix at previous time-step
rho_bml	the density matrix at current time-step
aux_bml	the temp matrix used for value storage during computation
matrix_type	the type of BML matrix
thresh	the threshold for the BML matrix

Definition at line 211 of file prg_quantumdynamics_mod.F90.

8.25.3 Variable Documentation

8.25.3.1 dp

```
integer, parameter prg_quantumdynamics_mod::dp = kind(1.0d0)  [private]
```

Definition at line 14 of file prg_quantumdynamics_mod.F90.

8.26 prg_response_mod Module Reference

Module to compute the density matrix response and related quantities.

Data Types

- type respdata_type

Functions/Subroutines

- subroutine, public prg_parse_response (RespData, filename)

  The parser for the calculation of the DM response.
- subroutine, public prg_compute_dipole (charges, coordinate, dipoleMoment, factor, verbose)

  To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.
- subroutine, public prg_write_dipole_tcl (dipoleMoment, file, factor, verbose)

  To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: $ vmd -e dipole.tcl.
- subroutine, public prg_compute_polarizability (rsp_bml, prt_bml, polarizability, factor, verbose)

  To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.
- subroutine, public prg_pert_from_file (prt_bml, norb)

  Read perturbation from file.
- subroutine, public prg_compute_response_rs (ham_bml, prt_bml, rsp_bml, lambda, bndfil, threshold, verbose)

  Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:
- subroutine, public prg_compute_response_fd (ham_bml, prt_bml, rsp_bml, prg_delta, bndfil, threshold, verbose)

  Computes the first order response density matrix using finite differences. The transformation hereby performed are:
- subroutine, public prg_pert_constant_field (field, intensity, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

  Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{\mathbf{r}}$). In the matrix representation, this is: $H^{(1)} = \lambda\frac{1}{2}(\,S\,e\mathbf{r}\cdot\mathbf{E} + e\mathbf{r}\cdot\mathbf{E}S)$. The symmetrization is done in order to preserve the Hermiticity of H. In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ.
- subroutine, public prg_pert_sin_pot (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

  Apply a sinusoidal length dependent potential ($\sin(\tilde{\mathbf{r}}_x)$) where $\mathbf{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.
- subroutine, public prg_pert_cos_pot (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

  Apply a cosine length dependent potential ($\cos(\tilde{\mathbf{r}}_x)$) where $\mathbf{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.
- subroutine, public prg_compute_response_sp2 (ham_bml, prt_bml, rsp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, threshold, verbose)

  Finds the first order response matrix from a Hamiltonian matrix.
- subroutine, public prg_project_response (rsp_bml, over_bml, spindex, norbi, coordinates, rspfunc, verbose)

  Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha\in i}\rho^{(1)}_{\alpha\alpha}$, where orbital α belong to atom i.
- subroutine, public prg_canon_response_vector (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, thresh, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.
- subroutine, public prg_canon_response (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.
- subroutine, public prg_canon_response_orig (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, thresh, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.

Variables

- integer, parameter dp = kind(1.0d0)
- real(dp), parameter pi = 3.14159265358979323846264338327950_dp

8.26.1 Detailed Description

Module to compute the density matrix response and related quantities.

Todo Add the response scf

Change name response_SP2 to dm_prt_response

Change name response_rs to rs_prt_response

More information about the theory can be found at [4] and Niklasson2015

8.26.2 Function/Subroutine Documentation

8.26.2.1 prg_canon_response()

```
subroutine, public prg_response_mod::prg_canon_response (
             type(bml_matrix_t), intent(inout) P1_bml,
             type(bml_matrix_t), intent(inout) H1_bml,
             real(dp), intent(in) Nocc,
             real(dp), intent(in) beta,
             real(dp), dimension(hdim), intent(in) evals,
             real(dp), intent(in) mu0,
             integer, intent(in) m,
             integer, intent(in) HDIM )
```

First-order Canonical Density Matrix Perturbation Theory.

Assuming known mu0 and representation in H0's eigenbasis Q, where H0 and P0 become diagonal. (mu0, eigenvalues e and eigenvectors Q of H0 are assumed known) Based on PRL 92, 193001 (2004) and PRE 92, 063301 (2015).

Parameters

P1_bml	First-order canonical response output.
H1_bml	Perturbative hamiltonian input.
Nocc	Number of ocupied orbitals.
beta	Inverse electronic temperature.
evals	Eigenvalues of the system.
mu0	Chemical potential.
m	Number of recursive steps.
HDIM	Number of orbitals - Hamiltonina zise.

Definition at line 948 of file prg_response_mod.F90.

8.26.2.2 prg_canon_response_orig()

```
subroutine, public prg_response_mod::prg_canon_response_orig (
            type(bml_matrix_t), intent(inout) P1_bml,
            type(bml_matrix_t), intent(inout) H1_bml,
            real(dp), intent(in) Nocc,
            real(dp), intent(in) beta,
            real(dp), dimension(hdim), intent(in) evals,
            real(dp), intent(in) mu0,
            integer, intent(in) m,
            real(dp), intent(in) thresh,
            integer, intent(in) HDIM )
```

First-order Canonical Density Matrix Perturbation Theory.

Assuming known mu0 and representation in H0's eigenbasis Q, where H0 and P0 become diagonal. Original routine. (mu0, eigenvalues e and eigenvectors Q of H0 are assumed known) Based on PRL 92, 193001 (2004) and PRE 92, 063301 (2015).

Parameters

P1_bml	First-order canonical response output.
H1_bml	Perturbative hamiltonian input.
Nocc	Number of ocupied orbitals.
beta	Inverse electronic temperature.
evals	Eigenvalues of the system.
mu0	Chemical potential.
m	Number of recursive steps.
HDIM	Number of orbitals - Hamiltonina zise.

Definition at line 1039 of file prg_response_mod.F90.

8.26.2.3 prg_canon_response_vector()

```
subroutine, public prg_response_mod::prg_canon_response_vector (
            type(bml_matrix_t), intent(inout) P1_bml,
            type(bml_matrix_t), intent(inout) H1_bml,
            real(dp), intent(in) Nocc,
            real(dp), intent(in) beta,
            real(dp), dimension(hdim), intent(in) evals,
            real(dp), intent(in) mu0,
            integer, intent(in) m,
            real(dp), intent(in) thresh,
            integer, intent(in) HDIM )
```

First-order Canonical Density Matrix Perturbation Theory.

Assuming known mu0 and representation in H0's eigenbasis Q, where H0 and P0 become diagonal. (mu0, eigenvalues e and eigenvectors Q of H0 are assumed known) Based on PRL 92, 193001 (2004) and PRE 92, 063301 (2015).

Parameters

P1_bml	First-order canonical response output.
H1_bml	Perturbative hamiltonian input.
Nocc	Number of ocupied orbitals.
beta	Inverse electronic temperature.
evals	Eigenvalues of the system.
mu0	Chemical potential.
m	Number of recursive steps.
HDIM	Number of orbitals - Hamiltonina zise.

Definition at line 850 of file prg_response_mod.F90.

8.26.2.4 prg_compute_dipole()

```
subroutine, public prg_response_mod::prg_compute_dipole (
            real(dp), dimension(:), intent(in) charges,
            real(dp), dimension(:,:), intent(in) coordinate,
            real(dp), dimension(3), intent(inout) dipoleMoment,
            real(dp), intent(in) factor,
            integer verbose )
```

To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.

Parameters

charges	Charges on each atomic position.
coordinate	Coordinates of the atoms.
nats	Number of atoms.
dipoleMoment	Dipole moment vector.
factor	Unit conversion factor (use 1.0 is no conversion is required).
verbose	To give different verbosity levels. If coordinates are in and charges are in fractions of electron, then transformation ea2debye form LATTE lib can be used to change units to Debye.

Definition at line 123 of file prg_response_mod.F90.

8.26.2.5 prg_compute_polarizability()

```
subroutine, public prg_response_mod::prg_compute_polarizability (
            type(bml_matrix_t), intent(in) rsp_bml,
            type(bml_matrix_t), intent(in) prt_bml,
            real(dp), intent(inout) polarizability,
            real(dp), intent(in) factor,
            integer verbose )
```

To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.

Parameters

charges	Charges on each atomic position.
coordinate	Coordinates of the atoms.
nats	Number of atoms.
dipoleMoment	Dipole moment vector.
factor	Unit conversion factor (use 1.0 is no conversion is required).
verbose	To give different verbosity levels. If coordinates are in and charges are in fractions of electron, then transformation ea2debye form LATTE lib can be used to change units to Debye.

Definition at line 201 of file prg_response_mod.F90.

8.26.2.6 prg_compute_response_fd()

```
subroutine, public prg_response_mod::prg_compute_response_fd (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(in) prt_bml,
            type(bml_matrix_t), intent(inout) rsp_bml,
            real(dp) prg_delta,
            real(dp), intent(in) bndfil,
            real(dp), intent(in) threshold,
            integer verbose )
```

Computes the first order response density matrix using finite differences. The transformation hereby performed are:

- $H^{+} = H^{(0)} + \delta H^{(1)}$
- $H^{-} = H^{(0)} - \delta H^{(1)}$
- $\rho^{+} = f(H^{+})$
- $\rho^{-} = f(H^{-})$
- $\rho^{(1)} = (\rho^{+} - \rho^{-})/(2\delta)$. Where f denotes the Fermi function (construction of the density matrix)

Parameters

ham_bml	Hamiltonian in bml format ($H^{(0)}$).
prt_bml	Perturbation in bml format ($H^{(1)}$).
rsp_bml	First order response to the perturbation ($\rho^{(1)}$).
bndfil	Filing factor.
threshold	Threshold value for matrix elements.
verbose	Different levels of verbosity.

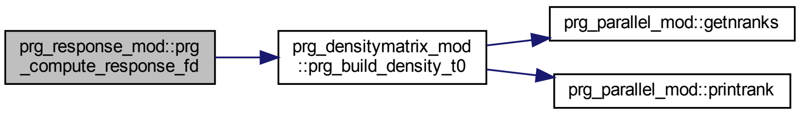
Warning

This works only for the prg_orthogonalized form of ham_bml.

The response must be in the prg_orthogonalized form.

Definition at line 382 of file prg_response_mod.F90.

Here is the call graph for this function:


prg_response_mod::prg_compute_response_fd
prg_densitymatrix_mod::prg_build_density_t0
prg_parallel_mod::getnranks
prg_parallel_mod::printrank

8.26.2.7 prg_compute_response_rs()

```
subroutine, public prg_response_mod::prg_compute_response_rs (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(in) prt_bml,
            type(bml_matrix_t), intent(inout) rsp_bml,
            real(dp) lambda,
            real(dp), intent(in) bndfil,
            real(dp), intent(in) threshold,
            integer verbose )
```

Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:

- $V = C^{\dagger} H^{(1)} C$
- $\tilde{V}_{ij} = \frac{V_{ij}}{\epsilon_j - \epsilon_i}$, with $\tilde{V}_{ii} = 0 \, \forall i$.
- $C^{(1)} = C\tilde{V}$
- And finally: $\rho^{(1)} = Cf(C^{(1)})^{\dagger} + C^{(1)} f C^{\dagger}$

Parameters

ham_bml	Hamiltonian in bml format ($H^{(0)}$).
prt_bml	Perturbation in bml format ($H^{(1)}$).
rsp_bml	First order response to the perturbation ($\rho^{(1)}$).
bndfil	Filing factor.
threshold	Threshold value for matrix elements.
verbose	Different levels of verbosity.

Warning

This works only for the prg_orthogonalized form of ham_bml.
The response must be in the prg_orthogonalized form.

Definition at line 252 of file prg_response_mod.F90.

8.26.2.8 prg_compute_response_sp2()

```
subroutine, public prg_response_mod::prg_compute_response_sp2 (
            type(bml_matrix_t), intent(in) ham_bml,
            type(bml_matrix_t), intent(in) prt_bml,
            type(bml_matrix_t), intent(inout) rsp_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp) lambda,
            real(dp), intent(in) bndfil,
            integer, intent(in) minsp2iter,
            integer, intent(in) maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp), intent(in) idemtol,
            real(dp), intent(in) threshold,
            integer verbose )
```

Finds the first order response matrix from a Hamiltonian matrix.

Definition at line 655 of file prg_response_mod.F90.

8.26.2.9 prg_parse_response()

```
subroutine, public prg_response_mod::prg_parse_response (
            type(respdata_type) RespData,
            character(len=*) filename )
```

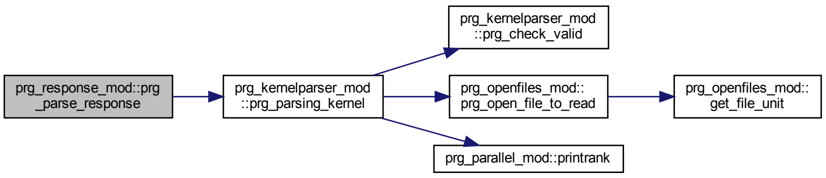
The parser for the calculation of the DM response.

Parameters

RespData	Response data type.
filename	Name of the file to parse.

Definition at line 46 of file prg_response_mod.F90.

Here is the call graph for this function:


prg_response_mod::prg_parse_response
prg_kernelparser_mod::prg_parsing_kernel
prg_kernelparser_mod::prg_check_valid
prg_openfiles_mod::prg_open_file_to_read
prg_openfiles_mod::get_file_unit
prg_parallel_mod::printrank

8.26.2.10 prg_pert_constant_field()

```
subroutine, public prg_response_mod::prg_pert_constant_field (
             real(dp), dimension(3), intent(in) field,
             real(dp) intensity,
             real(dp), dimension(:,:), intent(in) coordinate,
             real(dp) lambda,
             type(bml_matrix_t), intent(inout) prt_bml,
             real(dp) threshold,
             integer, dimension(:), intent(in) spindex,
             integer, dimension(:), intent(in) norbi,
             integer, intent(in) verbose,
             type(bml_matrix_t), intent(in), optional over_bml )
```

Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{\mathbf{r}}$). In the matrix representation, this is: $H^{(1)} = \lambda\frac{1}{2}(S\,e\mathbf{r}\cdot\mathbf{E} + e\mathbf{r}\cdot\mathbf{E}S)$. The symmetrization is done in order to preserve the Hermiticity of H. In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ.

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter over_bml can be omitted.

Parameters

field	Direction of the applied field ($\hat{\mathbf{E}}$).
intensity	Intensity of the field ($\|\|\mathbf{E}\|\|$)..
coordinate	Coordinates of the system ($\mathbf{r}$).
lambda	Constant to premultiply the perturbation (λ).
prt_bml	Perturbation in bml format ($H^{(1)}$).
threshold	Threshold value for bml format matrices.
spindex	Species index. It gives the species index of a particular atom.
norbi	Number of orbitals for each atomic site.
verbose	Different levels of verbosity.
over_bml	It has to be present for a nonorthogonal representation (S).

Definition at line 449 of file prg_response_mod.F90.

8.26.2.11 prg_pert_cos_pot()

```
subroutine, public prg_response_mod::prg_pert_cos_pot (
            character direction,
            real(dp) lx,
            real(dp), dimension(:,:), intent(in) coordinate,
            real(dp) lambda,
            type(bml_matrix_t), intent(inout) prt_bml,
            real(dp) threshold,
            integer, dimension(:), intent(in) spindex,
            integer, dimension(:), intent(in) norbi,
            integer, intent(in) verbose,
            type(bml_matrix_t), intent(in), optional over_bml )
```

Apply a cosine length dependent potential ($\cos(\tilde{\mathbf{r}}_x)$) where $\mathbf{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\mathbf{r}}_x) + \sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter over_bml can be omitted.

Parameters

direction	Direction of the potential gradient (x,y or z).
lx	Lenght of the box in x direction.
coordinate	Coordinates of the system ($\mathbf{r}$).
lambda	Constant to premultiply the perturbation (λ).
prt_bml	Perturbation in bml format ($H^{(1)}$).
threshold	Threshold value for bml format matrices.
norbi	Number of orbitals for each atomic site.
verbose	Different levels of verbosity.
over_bml	It has to be present for a nonorthogonal representation (S).

Definition at line 594 of file prg_response_mod.F90.

8.26.2.12 prg_pert_from_file()

```
subroutine, public prg_response_mod::prg_pert_from_file (
            type(bml_matrix_t), intent(inout) prt_bml,
            integer norb )
```

Read perturbation from file.

Todo Add read perturbation from file

Definition at line 226 of file prg_response_mod.F90.

8.26.2.13 prg_pert_sin_pot()

```
subroutine, public prg_response_mod::prg_pert_sin_pot (
             character direction,
             real(dp) lx,
             real(dp), dimension(:,:), intent(in) coordinate,
             real(dp) lambda,
             type(bml_matrix_t), intent(inout) prt_bml,
             real(dp) threshold,
             integer, dimension(:), intent(in) spindex,
             integer, dimension(:), intent(in) norbi,
             integer, intent(in) verbose,
             type(bml_matrix_t), intent(in), optional over_bml )
```

Apply a sinusoidal length dependent potential ($\sin(\tilde{\mathbf{r}}_x)$) where $\mathbf{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\mathbf{r}}_x)+\sin(\tilde{\mathbf{r}}_x)S)$. $\tilde{\mathbf{r}}_x = 2\pi(\mathbf{r}/l_x)-\pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.

Note

If the Hamiltonian is already in the prg_orthogonalized form, then parameter over_bml can be omitted.

Parameters

direction	Direction of the potential gradient (x,y or z).
lx	Length of the box in x direction.
coordinate	Coordinates of the system ($\mathbf{r}$).
lambda	Constant to premultiply the perturbation (λ).
prt_bml	Perturbation in bml format ($H^{(1)}$).
threshold	Threshold value for bml format matrices.
norbi	Number of orbitals for each atomic site.
verbose	Different levels of verbosity.
over_bml	It has to be present for a nonorthogonal representation (S).

Definition at line 526 of file prg_response_mod.F90.

8.26.2.14 prg_project_response()

```
subroutine, public prg_response_mod::prg_project_response (
             type(bml_matrix_t), intent(inout) rsp_bml,
             type(bml_matrix_t), intent(in) over_bml,
             integer, dimension(:), intent(in) spindex,
             integer, dimension(:), intent(in) norbi,
             real(dp), dimension(:,:), intent(in) coordinates,
             real(dp), dimension(:), intent(inout), allocatable rspfunc,
             integer, intent(in) verbose )
```

Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha\in i}\rho^{(1)}_{\alpha\alpha}$, where orbital α belong to atom i.

Parameters

rsp_bml	First order response density matrix.
spindex	It gives the species index of a particular atom.
norbi	Number of orbitals of species i.
coordinates	Atomic coordinates.
rspfunc	Response function at atomic positions.
verbose	Different levels of verbosity.

Definition at line 798 of file prg_response_mod.F90.

8.26.2.15 prg_write_dipole_tcl()

```
subroutine, public prg_response_mod::prg_write_dipole_tcl (
            real(dp), dimension(3), intent(in) dipoleMoment,
            character(*), intent(in) file,
            real(dp), intent(in) factor,
            integer verbose )
```

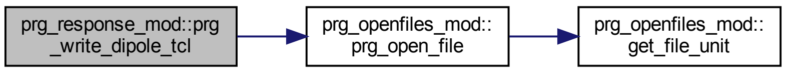
To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: $ vmd -e dipole.tcl.

Parameters

dipoleMoment	Dipole moment vector.
file	PDB/XYZ file to load for visualization.
factor	Arbitrary scale for visualization.
verbose	To give different verbosity levels.

Definition at line 161 of file prg_response_mod.F90.

Here is the call graph for this function:


prg_response_mod::prg_write_dipole_tcl
prg_openfiles_mod::prg_open_file
prg_openfiles_mod::get_file_unit

8.26.3 Variable Documentation

8.26.3.1 dp

```
integer, parameter prg_response_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_response_mod.F90.

8.26.3.2 pi

```
real(dp), parameter prg_response_mod::pi = 3.14159265358979323846264338327950_dp  [private]
```

Definition at line 19 of file prg_response_mod.F90.

8.27 prg_sp2_fermi_mod Module Reference

The SP2 Fermi module.

Functions/Subroutines

- subroutine, public prg_sp2_fermi_init (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist)

  Truncated SP2 prg_initialization.
- subroutine, public prg_sp2_fermi_init_norecs (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist, verbose)

  Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter beta = (1/KbT).
- subroutine, public prg_sp2_fermi (h_bml, osteps, nsteps, nocc, mu, beta, h1, hN, sgnlist, threshold, eps, traceLimit, x_bml)

  Calculate Truncated SP2.
- subroutine, public prg_sp2_entropy_function (mu, h1, hN, nsteps, sgnlist, GG, ee)

  Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.
- real(dp) function, public sp2_entropy_ts (D0_bml, GG, ee)

  Test SP2 entropy. Get the entropy contribution TS to the total free energy.
- real(dp) function, public sp2_inverse (f, mu, h1, hN, nsteps, sgnlist)

  Calculate the SP2 inverse.
- real(dp) function absmaxderivative (func, de)

  Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter dp = kind(1.0d0)

8.27.1 Detailed Description

The SP2 Fermi module.

This subroutine implements Niklasson's truncated SP2 density matrix purification algorithm.

8.27.2 Function/Subroutine Documentation

8.27.2.1 absmaxderivative()

```
real(dp) function prg_sp2_fermi_mod::absmaxderivative (
             real(dp), dimension(:), intent(in) func,
             real(dp), intent(in) de )  [private]
```

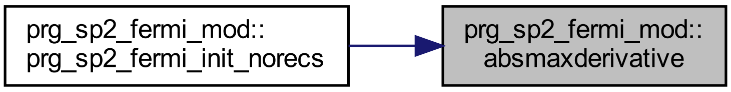
Gets the absolute maximum of the derivative of a function.

Parameters

func.	
de	Energy step.

Definition at line 618 of file prg_sp2_fermi_mod.F90.

Here is the caller graph for this function:



8.27.2.2 prg_sp2_entropy_function()

```
subroutine, public prg_sp2_fermi_mod::prg_sp2_entropy_function (
             real(dp), intent(in) mu,
             real(dp), intent(in) h1,
             real(dp), intent(in) hN,
             integer, intent(in) nsteps,
             integer, dimension(:), intent(in) sgnlist,
             real(dp), dimension(:), intent(inout), allocatable GG,
             real(dp), dimension(:), intent(inout), allocatable ee )
```

Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.

Parameters

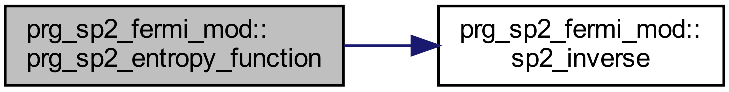
mu	Shifted chemical potential
h1	Minimum scaled Gershgorin bound

Parameters

hN	Maximum scaled Gershgorin bound
nsteps	Number of SP2 steps
sgnlist	SP2 sequence
GG	Entropy function
ee	1D mesh

Definition at line 483 of file prg_sp2_fermi_mod.F90.

Here is the call graph for this function:



8.27.2.3 prg_sp2_fermi()

```
subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi (
             type(bml_matrix_t), intent(in) h_bml,
             integer, intent(in) osteps,
             integer, intent(in) nsteps,
             real(dp), intent(in) nocc,
             real(dp), intent(inout) mu,
             real(dp), intent(inout) beta,
             real(dp), intent(inout) h1,
             real(dp), intent(inout) hN,
             integer, dimension(:), intent(in) sgnlist,
             real(dp), intent(in) threshold,
             real(dp), intent(in) eps,
             real(dp), intent(in) traceLimit,
             type(bml_matrix_t), intent(inout) x_bml )
```

Calculate Truncated SP2.

Parameters

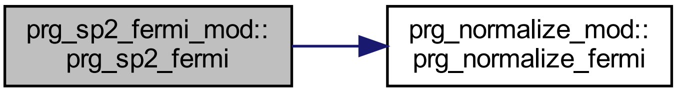
h_bml	Hamiltonian matrix
osteps	Outer loop steps
nsteps	Number of sequence branches
nocc	Number of occupation states
mu	Shifted chemical potential
beta	Inverse temperature

Parameters

h1	Minimum scaled Gershgorin bound.
hN	Maximum scaled Gershgorin bound.
sgnlist	SP2 sequence
threshold	Threshold for multiplies
eps	Occupation error limit
traceLimit	Trace limit
x_bml	Output density matrix

Definition at line 390 of file prg_sp2_fermi_mod.F90.

Here is the call graph for this function:



8.27.2.4 prg_sp2_fermi_init()

```
subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi_init (
              type(bml_matrix_t), intent(in) h_bml,
              integer, intent(in) nsteps,
              real(dp), intent(in) nocc,
              real(dp), intent(in) tscale,
              real(dp), intent(in) threshold,
              real(dp), intent(in) occErrLimit,
              real(dp), intent(in) traceLimit,
              type(bml_matrix_t), intent(inout) x_bml,
              real(dp), intent(inout) mu,
              real(dp), intent(inout) beta,
              real(dp), intent(inout) h1,
              real(dp), intent(inout) hN,
              integer, dimension(:), intent(inout) sgnlist )
```

Truncated SP2 prg_initialization.

Parameters

h_bml	Input Hamiltonian matrix.
nsteps	Number of sp2 iterations.
nocc	Number of occupied states.
tscale	Temperature rescaling factor.

Parameters

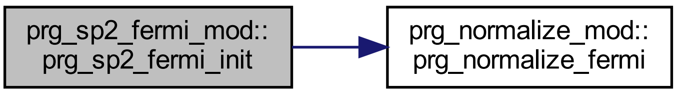
threshold	Threshold for multiplication.
occErrLimit	Occupation error limit.
traceLimit	Trace limit.
x_bml	Output prg_initial matrix.
mu	Shifted chemical potential
beta	Output inverse temperature.
h1	Output temperature-scaled minimum gershgorin bound.
hN	Output temperature-scaled maximum gershgorin bound.
sgnlist	SP2 sequence

Calculate Gershgorin bounds and rescale

Determine sequence branching first time through

Definition at line 45 of file prg_sp2_fermi_mod.F90.

Here is the call graph for this function:



8.27.2.5 prg_sp2_fermi_init_norecs()

```
subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi_init_norecs (
            type(bml_matrix_t), intent(in) h_bml,
            integer, intent(inout) nsteps,
            real(dp), intent(in) nocc,
            real(dp), intent(in) tscale,
            real(dp), intent(in) threshold,
            real(dp), intent(in) occErrLimit,
            real(dp), intent(in) traceLimit,
            type(bml_matrix_t), intent(inout) x_bml,
            real(dp), intent(inout) mu,
            real(dp), intent(inout) beta,
            real(dp), intent(inout) h1,
            real(dp), intent(inout) hN,
            integer, dimension(:), intent(inout) sgnlist,
            integer, optional verbose )
```

Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter beta = (1/KbT).

Parameters

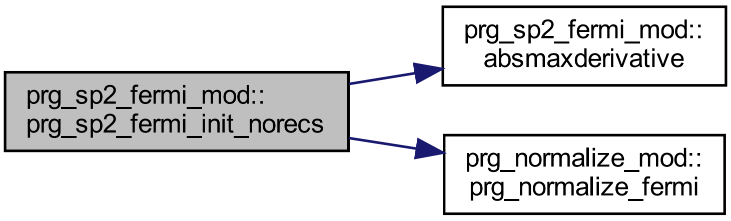
h_bml	Input Hamiltonian matrix.
nsteps	Output number of sp2 iterations.
nocc	Number of occupied states.
tscale	Temperature rescaling factor.
threshold	Threshold for multiplication.
occErrLimit	Occupation error limit.
traceLimit	Trace limit.
x_bml	Output prg_initial matrix.
mu	Shifted chemical potential
beta	Input guess and output inverse temperature.
h1	Output temperature-scaled minimum gershgorin bound.
hN	Output temperature-scaled maximum gershgorin bound.
sgnlist	SP2 sequence
verbose	Optional parameter for verbosity.

Calculate Gershgorin bounds and rescale

Determine sequence branching first time through

Definition at line 200 of file prg_sp2_fermi_mod.F90.

Here is the call graph for this function:



8.27.2.6 sp2_entropy_ts()

```
real(dp) function, public prg_sp2_fermi_mod::sp2_entropy_ts (
            type(bml_matrix_t), intent(in) D0_bml,
            real(dp), dimension(*), intent(in) GG,
            real(dp), dimension(*), intent(in) ee )
```

Test SP2 entropy. Get the entropy contribution TS to the total free energy.

Parameters

D0_bml	BML matrix
GG	Entropy function
ee	1D mesh
TS	Energy contribution

Definition at line 541 of file prg_sp2_fermi_mod.F90.

8.27.2.7 sp2_inverse()

```
real(dp) function, public prg_sp2_fermi_mod::sp2_inverse (
            real(dp), intent(in) f,
            real(dp), intent(in) mu,
            real(dp), intent(in) h1,
            real(dp), intent(in) hN,
            integer, intent(in) nsteps,
            integer, dimension(:), intent(in) sgnlist )
```

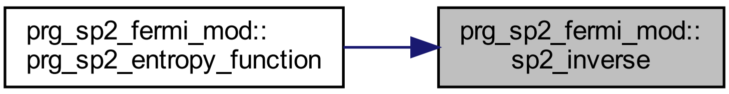
Calculate the SP2 inverse.

Parameters

f	Occupation factor
mu	Shifted chemical potential
h1	Minimum scaled Gershgorin bound
hN	Maximum scaled Gershgorin bound
nsteps	Numbers of SP2 iterations
sgnlist	SP2 sequence
ee	Energy value

Definition at line 593 of file prg_sp2_fermi_mod.F90.

Here is the caller graph for this function:



8.27.3 Variable Documentation

8.27.3.1 dp

```
integer, parameter prg_sp2_fermi_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_sp2_fermi_mod.F90.

8.28 prg_sp2_mod Module Reference

The SP2 module.

Functions/Subroutines

- subroutine, public prg_sp2_basic (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)

  Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.
- subroutine, public prg_sp2_basic_tcore (h_bml, rho_bml, rhofull_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_alg2 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_alg2_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv, verbose)
- subroutine, public prg_sp2_alg2_seq (h_bml, rho_bml, threshold, pp, icount, vv, verbose)
- subroutine, public prg_prg_sp2_alg2_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, verbose)
- subroutine, public prg_sp2_alg1 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_alg1_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv)
- subroutine, public prg_sp2_alg1_seq (h_bml, rho_bml, threshold, pp, icount, vv)
- subroutine, public prg_prg_sp2_alg1_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval)
- subroutine, public prg_sp2_submatrix (ham_bml, rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

  Perform SP2 algorithm using sequence and calculate norm for a submatrix.
- subroutine, public prg_sp2_submatrix_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

Variables

- integer, parameter dp = kind(1.0d0)
- integer, parameter dp1 = kind(1.0)

8.28.1 Detailed Description

The SP2 module.

This subroutine implements Niklasson's SP2 density matrix purification algorithm.

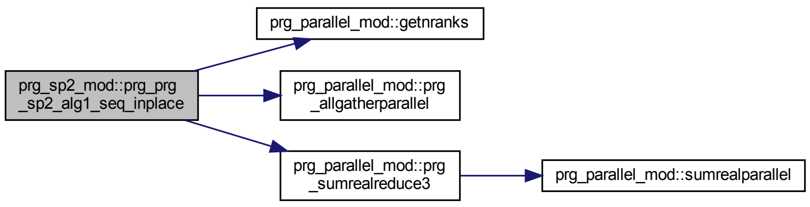
8.28.2 Function/Subroutine Documentation

8.28.2.1 prg_prg_sp2_alg1_seq_inplace()

```
subroutine, public prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace (
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            integer, dimension(:), intent(inout) pp,
            integer, intent(inout) icount,
            real(dp), dimension(:), intent(inout) vv,
            real(dp), intent(in) mineval,
            real(dp), intent(in) maxeval )
```

Definition at line 1099 of file prg_sp2_mod.F90.

Here is the call graph for this function:

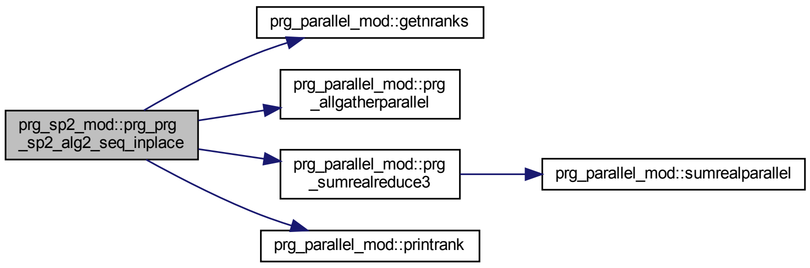

prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel

8.28.2.2 prg_prg_sp2_alg2_seq_inplace()

```
subroutine, public prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace (
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            integer, dimension(:), intent(inout) pp,
            integer, intent(inout) icount,
            real(dp), dimension(:), intent(inout) vv,
            real(dp), intent(in), optional mineval,
            real(dp), intent(in), optional maxeval,
            integer, intent(in), optional verbose )
```

Definition at line 640 of file prg_sp2_mod.F90.

Here is the call graph for this function:

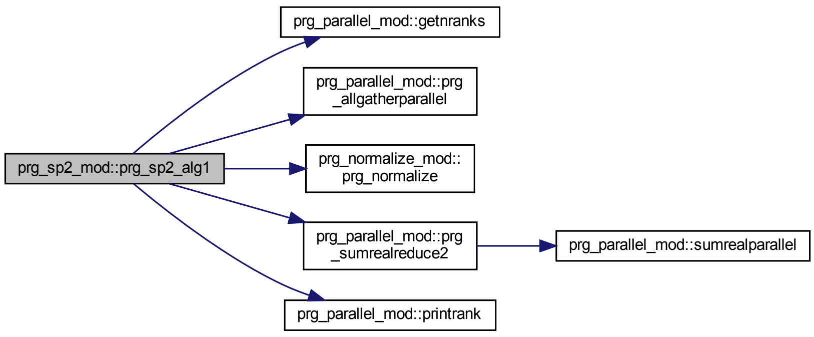

prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.3 prg_sp2_alg1()

```
subroutine, public prg_sp2_mod::prg_sp2_alg1 (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(in) bndfil,
            integer, intent(in) minsp2iter,
            integer, intent(in) maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp), intent(in) idemtol,
            integer, intent(in), optional verbose )
```

Definition at line 736 of file prg_sp2_mod.F90.

Here is the call graph for this function:

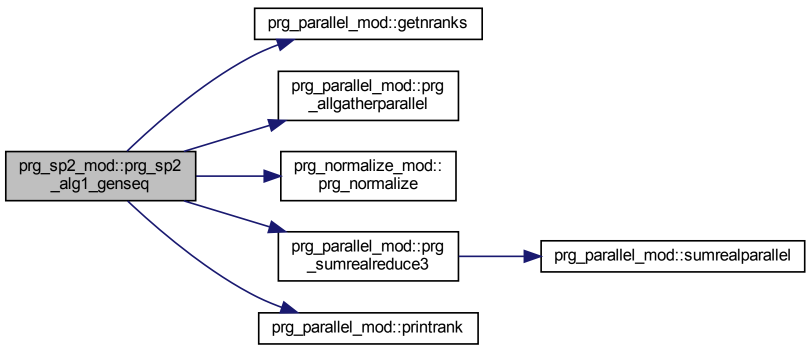

prg_sp2_mod::prg_sp2_alg1
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce2
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.4 prg_sp2_alg1_genseq()

```
subroutine, public prg_sp2_mod::prg_sp2_alg1_genseq (
             type(bml_matrix_t), intent(in) h_bml,
             type(bml_matrix_t), intent(inout) rho_bml,
             real(dp), intent(in) threshold,
             real(dp), intent(in) bndfil,
             integer, intent(in) minsp2iter,
             integer, intent(in) maxsp2iter,
             character(len=*), intent(in) sp2conv,
             real(dp), intent(in) idemtol,
             integer, dimension(:), intent(inout) pp,
             integer, intent(inout) icount,
             real(dp), dimension(:), intent(inout) vv )
```

Definition at line 879 of file prg_sp2_mod.F90.

Here is the call graph for this function:

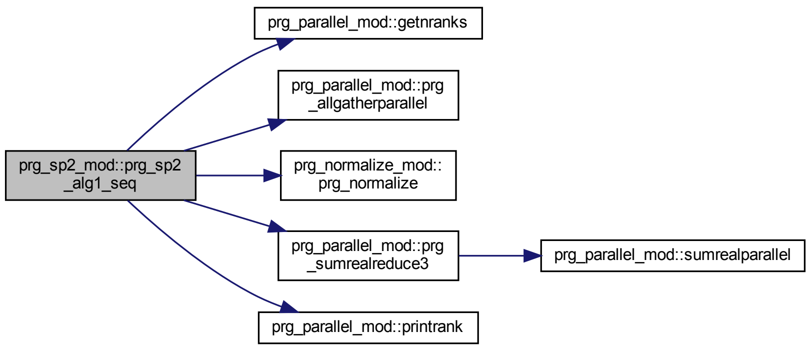

prg_sp2_mod::prg_sp2_alg1_genseq
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.5 prg_sp2_alg1_seq()

```
subroutine, public prg_sp2_mod::prg_sp2_alg1_seq (
             type(bml_matrix_t), intent(in) h_bml,
             type(bml_matrix_t), intent(inout) rho_bml,
             real(dp), intent(in) threshold,
             integer, dimension(:), intent(inout) pp,
             integer, intent(inout) icount,
             real(dp), dimension(:), intent(inout) vv )
```

Definition at line 1008 of file prg_sp2_mod.F90.

Here is the call graph for this function:

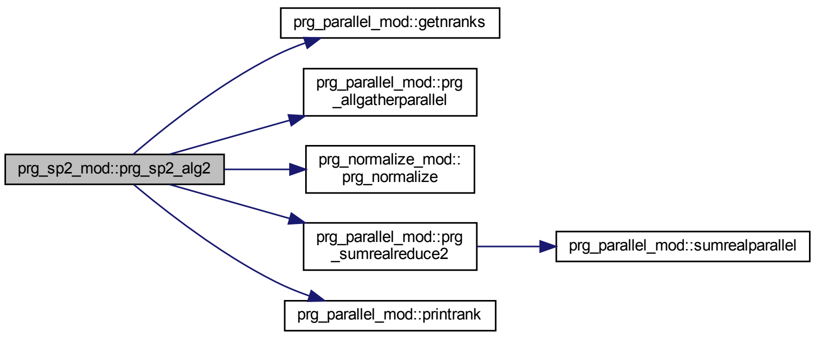

prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_sp2_mod::prg_sp2_alg1_seq
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.6 prg_sp2_alg2()

```
subroutine, public prg_sp2_mod::prg_sp2_alg2 (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(in) bndfil,
            integer, intent(in) minsp2iter,
            integer, intent(in) maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp), intent(in) idemtol,
            integer, intent(in), optional verbose )
```

Definition at line 265 of file prg_sp2_mod.F90.

Here is the call graph for this function:

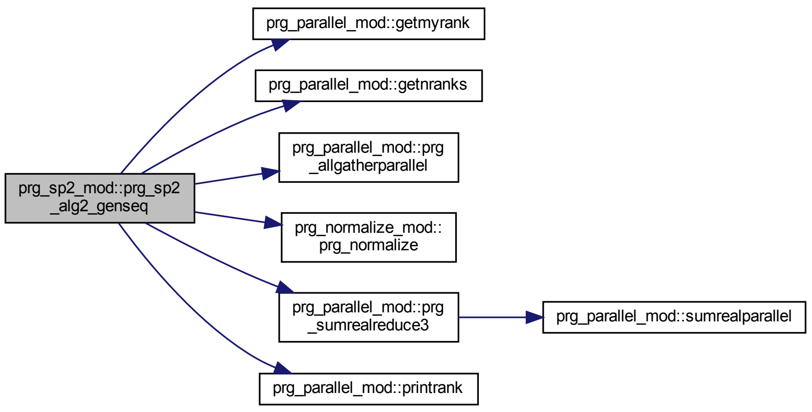

prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_sp2_mod::prg_sp2_alg2
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce2
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.7 prg_sp2_alg2_genseq()

```
subroutine, public prg_sp2_mod::prg_sp2_alg2_genseq (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            real(dp), intent(in) bndfil,
            integer, intent(in) minsp2iter,
            integer, intent(in) maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp), intent(in) idemtol,
            integer, dimension(:), intent(inout) pp,
            integer, intent(inout) icount,
            real(dp), dimension(:), intent(inout) vv,
            integer, intent(in), optional verbose )
```

Definition at line 397 of file prg_sp2_mod.F90.

Here is the call graph for this function:

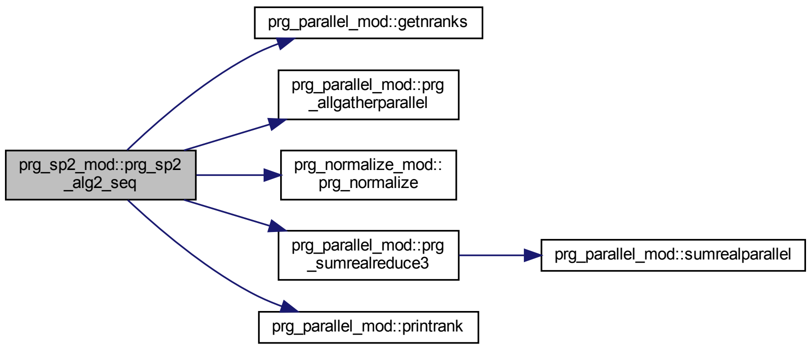

prg_sp2_mod::prg_sp2_alg2_genseq
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.8 prg_sp2_alg2_seq()

```
subroutine, public prg_sp2_mod::prg_sp2_alg2_seq (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp), intent(in) threshold,
            integer, dimension(:), intent(inout) pp,
            integer, intent(inout) icount,
            real(dp), dimension(:), intent(inout) vv,
            integer, intent(in), optional verbose )
```

Definition at line 544 of file prg_sp2_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::getnranks
prg_parallel_mod::prg_allgatherparallel
prg_sp2_mod::prg_sp2_alg2_seq
prg_normalize_mod::prg_normalize
prg_parallel_mod::prg_sumrealreduce3
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.9 prg_sp2_basic()

```
subroutine, public prg_sp2_mod::prg_sp2_basic (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            real(dp) threshold,
            real(dp) bndfil,
            integer minsp2iter,
            integer maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp) idemtol,
            integer verbose )
```

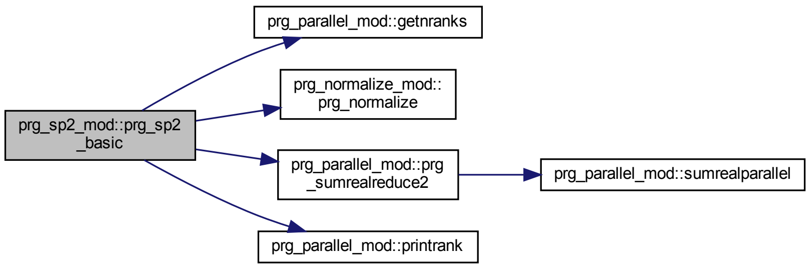
Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.

Parameters

h_bml	Input Hamiltonian matrix
rho_bml	Output density matrix
threshold	Threshold for sparse matrix algebra
bndfil	Bond
minsp2iter	Minimum sp2 iterations
maxsp2iter	Maximum SP2 iterations
sp2conv	Convergence type
idemtol	Idempotency tolerance
verbose	A verbosity level

Definition at line 51 of file prg_sp2_mod.F90.

Here is the call graph for this function:

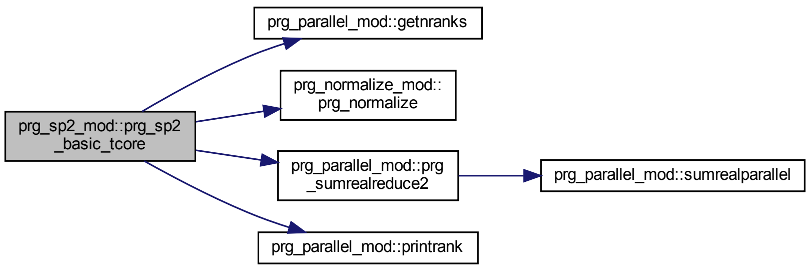

prg_parallel_mod::getnranks
prg_normalize_mod::prg_normalize
prg_sp2_mod::prg_sp2_basic
prg_parallel_mod::prg_sumrealreduce2
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.10 prg_sp2_basic_tcore()

```
subroutine, public prg_sp2_mod::prg_sp2_basic_tcore (
            type(bml_matrix_t), intent(in) h_bml,
            type(bml_matrix_t), intent(inout) rho_bml,
            type(bml_matrix_t), intent(inout) rhofull_bml,
            real(dp) threshold,
            real(dp) bndfil,
            integer minsp2iter,
            integer maxsp2iter,
            character(len=*), intent(in) sp2conv,
            real(dp) idemtol,
            integer verbose )
```

Definition at line 140 of file prg_sp2_mod.F90.

Here is the call graph for this function:


prg_parallel_mod::getnranks
prg_normalize_mod::prg_normalize
prg_sp2_mod::prg_sp2_basic_tcore
prg_parallel_mod::prg_sumrealreduce2
prg_parallel_mod::sumrealparallel
prg_parallel_mod::printrank

8.28.2.11 prg_sp2_submatrix()

```
subroutine, public prg_sp2_mod::prg_sp2_submatrix (
             type(bml_matrix_t), intent(in) ham_bml,
             type(bml_matrix_t), intent(inout) rho_bml,
             real(dp), intent(in) threshold,
             integer, dimension(:), intent(in) pp,
             integer, intent(in) icount,
             real(dp), dimension(:), intent(inout) vv,
             real(dp), intent(in) mineval,
             real(dp), intent(in) maxeval,
             integer, intent(in) core_size )
```

Perform SP2 algorithm using sequence and calculate norm for a submatrix.

Parameters

rho_bml	Input Hamiltonian/Output density matrix
threshold	Threshold for sparse matrix algebra
pp	Vector containing sequence of 0s and 1s
icount	Sequence count
vv	Vector of sum of squares per iteration
mineval	Min value used for normalization (optional)
maxeval	Max value used for normalization (optional)
core_size	Number of core rows

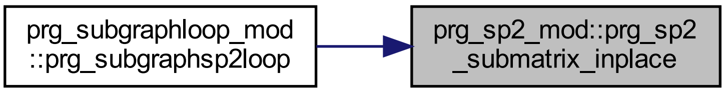
Definition at line 1189 of file prg_sp2_mod.F90.

8.28.2.12 prg_sp2_submatrix_inplace()

```
subroutine, public prg_sp2_mod::prg_sp2_submatrix_inplace (
             type(bml_matrix_t), intent(inout) rho_bml,
             real(dp), intent(in) threshold,
             integer, dimension(:), intent(inout) pp,
             integer, intent(inout) icount,
             real(dp), dimension(:), intent(inout) vv,
             real(dp), intent(in) mineval,
             real(dp), intent(in) maxeval,
             integer, intent(in) core_size )
```

Definition at line 1258 of file prg_sp2_mod.F90.

Here is the caller graph for this function:


prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_sp2_mod::prg_sp2 _submatrix_inplace

8.28.3 Variable Documentation

8.28.3.1 dp

```
integer, parameter prg_sp2_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_sp2_mod.F90.

8.28.3.2 dp1

```
integer, parameter prg_sp2_mod::dp1 = kind(1.0)  [private]
```

Definition at line 19 of file prg_sp2_mod.F90.

8.29 prg_sp2parser_mod Module Reference

SP2 parser.

Data Types

- type sp2data_type
  General SP2 solver type.

Functions/Subroutines

- subroutine, public prg_parse_sp2 (sp2data, filename)
  The parser for SP2 solver.

Variables

- integer, parameter dp = kind(1.0d0)

8.29.1 Detailed Description

SP2 parser.

This module is used to parse all the input variables for the SP2 method electronic structure solver. Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.
- Add the keyword (character type) in the keyvector_re vector.
- Add a default value (real type) in the valvector_re.
- Define a new variable and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.

8.29.2 Function/Subroutine Documentation

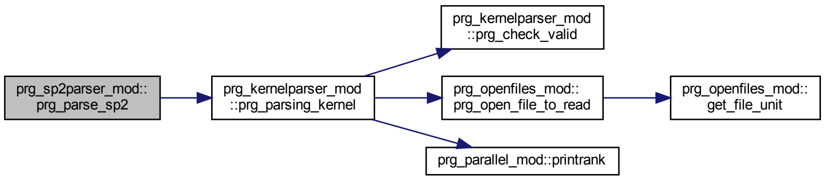
8.29.2.1 prg_parse_sp2()

```
subroutine, public prg_sp2parser_mod::prg_parse_sp2 (
             type(sp2data_type), intent(inout) sp2data,
             character(len=*) filename )
```

The parser for SP2 solver.

Definition at line 50 of file prg_sp2parser_mod.F90.

Here is the call graph for this function:



8.29.3 Variable Documentation

8.29.3.1 dp

```
integer, parameter prg_sp2parser_mod::dp = kind(1.0d0)  [private]
```

Definition at line 22 of file prg_sp2parser_mod.F90.

8.30 prg_subgraphloop_mod Module Reference

The subgraphloop module.

Functions/Subroutines

- subroutine, public prg_subgraphsp2loop (h_bml, g_bml, rho_bml, gp, threshold)
- subroutine, public prg_collectmatrixfromparts (gp, rho_bml)

    Collect distributed parts into same matrix.
- subroutine, public prg_balanceparts (gp)
- subroutine, public prg_partordering (gp)

    Set row ordering bases on parts.
- subroutine, public prg_getgrouppartitionhalosfromgraph (gp, g_bml, hnode, djflag)

    Get core+halo indeces for all partitions only using the graph.
- subroutine, public prg_getpartitionhalosfromgraph (gp, g_bml, djflag)

    Get core+halo indeces for all partitions only using the graph.

Variables

- integer, parameter dp = kind(1.0d0)

8.30.1 Detailed Description

The subgraphloop module.

8.30.2 Function/Subroutine Documentation

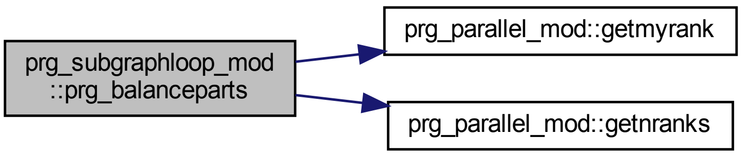
8.30.2.1 prg_balanceparts()

```
subroutine, public prg_subgraphloop_mod::prg_balanceparts (
            type (graph_partitioning_t), intent(inout) gp )
```

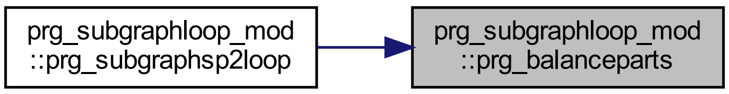
Renumber parts Handle unbalanced numbers of parts.

Definition at line 165 of file prg_subgraphloop_mod.F90.

Here is the call graph for this function:


prg_subgraphloop_mod ::prg_balanceparts
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks

Here is the caller graph for this function:


prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_subgraphloop_mod ::prg_balanceparts

8.30.2.2 prg_collectmatrixfromparts()

```
subroutine, public prg_subgraphloop_mod::prg_collectmatrixfromparts (
            type (graph_partitioning_t), intent(inout) gp,
            type (bml_matrix_t), intent(inout) rho_bml )
```

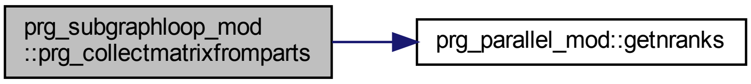
Collect distributed parts into same matrix.

Parameters

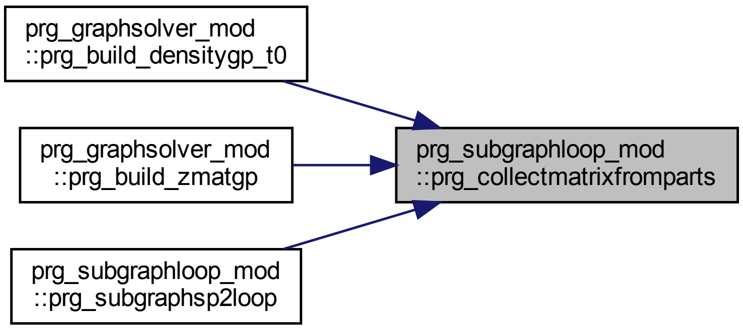
gp	Graph partitioning
rho_bml	Matrix to be collected into

Definition at line 133 of file prg_subgraphloop_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.30.2.3 prg_getgrouppartitionhalosfromgraph()

```
subroutine, public prg_subgraphloop_mod::prg_getgrouppartitionhalosfromgraph (
            type (graph_partitioning_t), intent(inout) gp,
            type (bml_matrix_t), intent(in) g_bml,
            integer, dimension(*), intent(in) hnode,
            logical, intent(in) djflag )
```

Get core+halo indeces for all partitions only using the graph.

Parameters

gp	Graph partitioning
g_bml	Graph
hnode	Group start indeces
djflg	Double jump flag (true/false)

Determine halo elements for each subgraph

Definition at line 292 of file prg_subgraphloop_mod.F90.

8.30.2.4 prg_getpartitionhalosfromgraph()

```
subroutine, public prg_subgraphloop_mod::prg_getpartitionhalosfromgraph (
            type (graph_partitioning_t), intent(inout) gp,
            type (bml_matrix_t), intent(in) g_bml,
            logical, intent(in) djflag )
```

Get core+halo indeces for all partitions only using the graph.

Parameters

gp	Graph partitioning
g_bml	Graph
djflg	Double jump flag (true/false)

Determine halo elements for each subgraph

Definition at line 337 of file prg_subgraphloop_mod.F90.

8.30.2.5 prg_partordering()

```
subroutine, public prg_subgraphloop_mod::prg_partordering (
            type (graph_partitioning_t), intent(inout) gp )
```

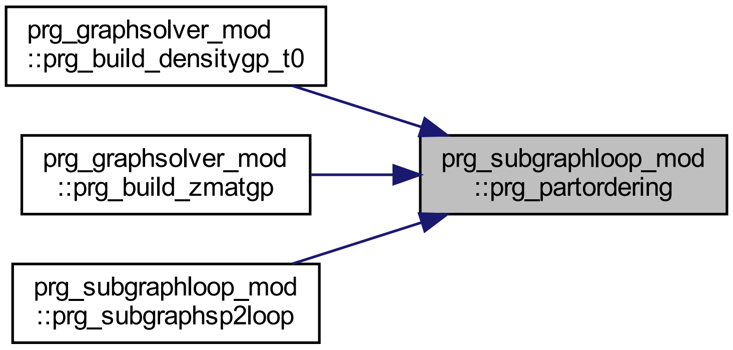
Set row ordering bases on parts.

Parameters

gp	Graph partitioning

Definition at line 263 of file prg_subgraphloop_mod.F90.

Here is the caller graph for this function:

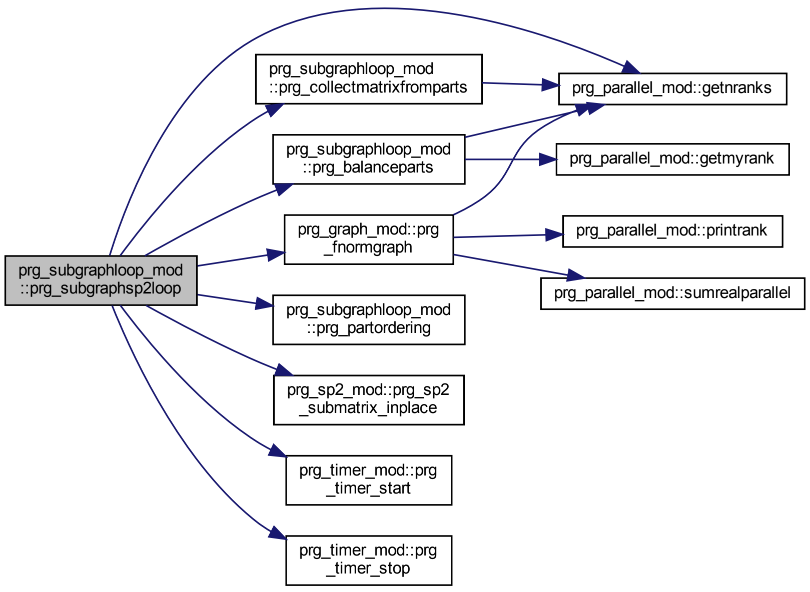

prg_graphsolver_mod ::prg_build_densitygp_t0
prg_graphsolver_mod ::prg_build_zmatgp
prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_subgraphloop_mod ::prg_partordering

8.30.2.6 prg_subgraphsp2loop()

```
subroutine, public prg_subgraphloop_mod::prg_subgraphsp2loop (
            type (bml_matrix_t), intent(in) h_bml,
            type (bml_matrix_t), intent(in) g_bml,
            type (bml_matrix_t), intent(inout) rho_bml,
            type (graph_partitioning_t), intent(inout) gp,
            real(dp), intent(in) threshold )
```

Definition at line 37 of file prg_subgraphloop_mod.F90.

Here is the call graph for this function:


prg_subgraphloop_mod ::prg_subgraphsp2loop
prg_subgraphloop_mod ::prg_collectmatrixfromparts
prg_subgraphloop_mod ::prg_balanceparts
prg_graph_mod::prg _fnormgraph
prg_subgraphloop_mod ::prg_partordering
prg_sp2_mod::prg_sp2 _submatrix_inplace
prg_timer_mod::prg _timer_start
prg_timer_mod::prg _timer_stop
prg_parallel_mod::getnranks
prg_parallel_mod::getmyrank
prg_parallel_mod::printrank
prg_parallel_mod::sumrealparallel

8.30.3 Variable Documentation

8.30.3.1 dp

```
integer, parameter prg_subgraphloop_mod::dp = kind(1.0d0)  [private]
```

Definition at line 18 of file prg_subgraphloop_mod.F90.

8.31 prg_syrotation_mod Module Reference

A module to rotate the coordinates of a sybsystem in chemical systems.

Data Types

- type rotation_type

  Rotation type.

Functions/Subroutines

- subroutine, public prg_parse_rotation (rot, filename)

  The parser for rotation.
- subroutine, public prg_rotate (rot, r, verbose)

  Rotation routine.

Variables

- integer, parameter dp = kind(1.0d0)

8.31.1 Detailed Description

A module to rotate the coordinates of a sybsystem in chemical systems.

It works by specifying two orientations and a rotation point.

8.31.2 Function/Subroutine Documentation

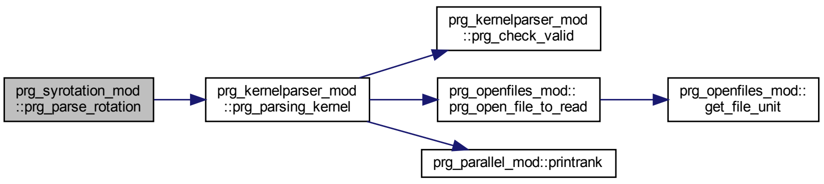
8.31.2.1 prg_parse_rotation()

```
subroutine, public prg_syrotation_mod::prg_parse_rotation (
            type(rotation_type), intent(inout) rot,
            character(len=*) filename )
```

The parser for rotation.

Definition at line 46 of file prg_syrotation_mod.F90.

Here is the call graph for this function:


prg_syrotation_mod ::prg_parse_rotation
prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

8.31.2.2 prg_rotate()

```
subroutine, public prg_syrotation_mod::prg_rotate (
            type(rotation_type), intent(in) rot,
            real(dp), dimension(:,:), intent(inout) r,
            integer, intent(in) verbose )
```

Rotation routine.

It works by indicating the orientations (v1 and v1) and a rotation center. The orientation can be passed either directly by setting v1 and v2 or by indicating two points pQ1 and pQ2. Orientation can also be specified with an atom position if patom1 and patom2 indices are not zero this atoms are used to determine the initial and final orientation.

Parameters

rot	Rotation type
r	Coordinates to be rotated
verbose	Verbosity level

Example:

```
        rot%patom1 = 4
        rot%patom2 = 0
        rot%catom2 = 6
        rot%v2 = 0.0 ; rot%v2(1) = 1
        call prg_rotate(rot,r)
```

The latter will orient the system such that atom 4 points to the (1,0,0) direction.

Definition at line 139 of file prg_syrotation_mod.F90.

8.31.3 Variable Documentation

8.31.3.1 dp

```
integer, parameter prg_syrotation_mod::dp = kind(1.0d0)  [private]
```

Definition at line 11 of file prg_syrotation_mod.F90.

8.32 prg_system_mod Module Reference

A module to read and handle chemical systems.

Data Types

- type estruct_type

  Electronic structure type.
- type system_type

  System type.

Functions/Subroutines

- subroutine, public prg_get_nameandext (fullfilename, filename, ext)

  Get the name and extension of a file.
- subroutine, public prg_parse_system (system, filename, extin)

  The parser for the chemical system.
- subroutine, public prg_destroy_system (sy)

  Deallocates all the arrays within a system.
- subroutine, public prg_write_system (system, filename, extin)

  Write system in .xyz, .dat or pdb file.
- subroutine, public prg_write_trajectory (system, iter, each, prg_deltat, filename, extension)

  Write trajectory in .xyz, .dat or pdb file.
- subroutine, public prg_write_trajectoryandproperty (system, iter, each, prg_deltat, scalarprop, filename, extension)

  Write trajectory and atomic properties. Only pdb file.
- subroutine, public prg_make_random_system (system, nats, seed, lx, ly, lz)

  Make random Xx system.
- subroutine, public prg_parameters_to_vectors (abc_angles, lattice_vector)

  Transforms the lattice parameters into lattice vectors.
- subroutine, public prg_vectors_to_parameters (lattice_vector, abc_angles)

  Transforms the lattice vectors into lattice parameters.
- subroutine, public prg_get_origin (coords, origin)

  Get the origin of the coordinates.
- subroutine, public prg_get_distancematrix (coords, dmat)

  Get the distance matrix.
- subroutine, public prg_translateandfoldtobox (coords, lattice_vectors, origin, verbose)

Translate and fold to box.

- subroutine, public prg_centeratbox (coords, lattice_vectors, verbose)

  Translate geometric center to the center of the box.
- subroutine, public prg_wraparound (coords, lattice_vectors, index, verbose)

  Wrap around atom i using pbc.
- subroutine, public prg_translatetogeomcandfoldtobox (coords, lattice_vectors, origin)

  Translate to geometric center.
- subroutine, public prg_replicate (coords, symbols, lattice_vectors, nx, ny, nz)

  Extend/replicate system along lattice vectors.
- subroutine, public prg_replicate_system (sy, syf, nx, ny, nz)

  Extend/replicate a system type along the lattice vectors.
- subroutine, public prg_cleanuprepeatedatoms (nats, coords, symbols, verbose)

  Cleanup repeated atoms we might have in the system.
- subroutine, public prg_get_recip_vects (lattice_vectors, recip_vectors, volr, volk)

  Get the volume of the cell and the reciprocal vectors: This soubroutine computes:
- subroutine, public prg_get_dihedral (coords, id1, id2, id3, id4, dihedral)

  Get the dihedral angle given four atomic positions.
- subroutine, public prg_get_covgraph (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)

  Get the covalency graph in bml format.
- subroutine prg_get_covgraph_int (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
- subroutine, public prg_get_covgraph_h (sy, nnStructMindist, nnStruct, nrnnstruct, rcut, graph_h, mdimin, verbose)

  Get the covanlency graph.
- subroutine, public prg_get_subsystem (sy, lsize, indices, sbsy, verbose)

  Get a subsystem out of the total system.
- subroutine, public prg_destroy_subsystems (sbsy, verbose)

  Destroy allocated subsystem.
- subroutine, public prg_molpartition (sy, npart, nnStructMindist, nnStruct, nrnnstruct, hetatm, gp, verbose)

  Partition by molecule.
- subroutine, public prg_get_partial_atomgraph (rho_bml, hindex, gch_bml, threshold, verbose)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_collect_graph_p (rho_bml, nc, nats, hindex, chindex, graph_p, threshold, mdimin, verbose)

  Collect the small graph to build the full graph.
- subroutine, public prg_merge_graph (graph_p, graph_h)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_merge_graph_adj (graph_p, graph_h, xadj, adjncy)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_adj2bml (xadj, adjncy, bml_type, g_bml)

  prg_adj2bml
- subroutine, public prg_graph2bml (graph, bml_type, g_bml)

  Graph2bml.
- subroutine, public prg_graph2vector (graph, vector, maxnz)

  Vectorize graph.
- subroutine, public prg_vector2graph (vector, graph, maxnz)

  Back to graph.
- subroutine, public prg_sortadj (xadj, adjncy)

  Sort adj NOTE: this might not be needed anymre since the bml_get_adj routine is sorting the values.

Variables

- integer, parameter dp = kind(1.0d0)

8.32.1 Detailed Description

A module to read and handle chemical systems.

This module will be used to build and handle a molecular system.

8.32.2 Function/Subroutine Documentation

8.32.2.1 prg_adj2bml()

```
subroutine, public prg_system_mod::prg_adj2bml (
             integer, dimension(:), intent(in) xadj,
             integer, dimension(:), intent(in) adjncy,
             character(20), intent(in) bml_type,
             type(bml_matrix_t), intent(inout) g_bml )
```

prg_adj2bml

Parameters

xadj	CSR start values for the adjacency matrix.
adjncy	CSR positions of adjacency matrix.
bml_type	bml format.
g_bml	graph in bml format.

Definition at line 2551 of file prg_system_mod.F90.

8.32.2.2 prg_centeratbox()

```
subroutine, public prg_system_mod::prg_centeratbox (
             real(dp), dimension(:,:), intent(inout), allocatable coords,
             real(dp), dimension(:,:), intent(in) lattice_vectors,
             integer, intent(in), optional verbose )
```

Translate geometric center to the center of the box.

Parameters

coords	Coordinates of the system (see system_type).
lattice_vectors	System lattice vectors.
verbose	Verbosity level.

Definition at line 1354 of file prg_system_mod.F90.

8.32.2.3 prg_cleanuprepeatedatoms()

```
subroutine, public prg_system_mod::prg_cleanuprepeatedatoms (
             integer, intent(inout) nats,
             real(dp), dimension(:,:), intent(inout), allocatable coords,
             character(len=*), dimension(:), intent(inout), allocatable symbols,
             integer, intent(in), optional verbose )
```

Cleanup repeated atoms we might have in the system.

Parameters

nats	Number of atoms in the system.
coords	Coordinates of the system (see system_type).
symbols	Atomic symbols (see system_type). \verbose Verbosity level.

Definition at line 1634 of file prg_system_mod.F90.

8.32.2.4 prg_collect_graph_p()

```
subroutine, public prg_system_mod::prg_collect_graph_p (
             type(bml_matrix_t), intent(in) rho_bml,
             integer, intent(in) nc,
             integer, intent(in) nats,
             integer, dimension(:,:), intent(in) hindex,
             integer, dimension(:), intent(in) chindex,
             integer, dimension(:,:), intent(inout), allocatable graph_p,
             real(dp), intent(in) threshold,
             integer, intent(in) mdimin,
             integer, intent(in), optional verbose )
```

Collect the small graph to build the full graph.

Parameters

rho_bml	Density matix in bml format.
nc	Number of core atoms.
nats	Number of atoms.
hindex	Hindex for the small part (see haindex)
chindex	Core-hallo index for the small part.
graph_p	Graph in an "ellpack" format.
threshold	Threshold to buil the density based atom projected graph.
verbose	Verbosity level.

Definition at line 2334 of file prg_system_mod.F90.

8.32.2.5 prg_destroy_subsystems()

```
subroutine, public prg_system_mod::prg_destroy_subsystems (
            type(system_type), intent(inout) sbsy,
            integer, intent(in), optional verbose )
```

Destroy allocated subsystem.

This routine will deallocate all the arrays of the structures.

Parameters

sy	System to de deallocated (see system_type).

Definition at line 2115 of file prg_system_mod.F90.

8.32.2.6 prg_destroy_system()

```
subroutine, public prg_system_mod::prg_destroy_system (
            type(system_type), intent(inout) sy )
```

Deallocates all the arrays within a system.

Parameters

sy	System type

Definition at line 644 of file prg_system_mod.F90.

8.32.2.7 prg_get_covgraph()

```
subroutine, public prg_system_mod::prg_get_covgraph (
            type(system_type), intent(in) sy,
            real(dp), dimension(:,:), intent(in) nnStructMindist,
            integer, dimension(:,:), intent(in) nnStruct,
            integer, dimension(:), intent(in) nrnnstruct,
            character(20), intent(in) bml_type,
            real(dp) factor,
            type(bml_matrix_t), intent(inout) gcov_bml,
            integer, intent(in) mdimin,
            integer, intent(in), optional verbose )
```

Get the covalency graph in bml format.

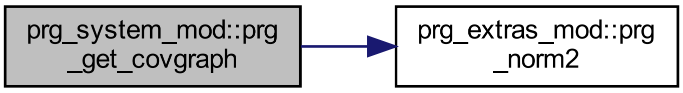
This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

Parameters

sy	System structure (see system_type).
nnStructMindist	Minimun distance between atoms.
nnStruct	The neigbors J to I within Rcut that are all within the box.
nrnnstruct	Number of neigbors to I within Rcut that are all within the box.
bml_type	The bml type for constructing the graph.
gconv_bml	Covanlency graph in bml format.
verbose	Verbosity level.

Definition at line 1820 of file prg_system_mod.F90.

Here is the call graph for this function:



8.32.2.8 prg_get_covgraph_h()

```
subroutine, public prg_system_mod::prg_get_covgraph_h (
             type(system_type), intent(in) sy,
             real(dp), dimension(:,:), intent(in) nnStructMindist,
             integer, dimension(:,:), intent(in) nnStruct,
             integer, dimension(:), intent(in) nrnnstruct,
             real(dp), intent(in) rcut,
             integer, dimension(:,:), intent(inout), allocatable graph_h,
             integer, intent(in) mdimin,
             integer, intent(in), optional verbose )
```

Get the covanlency graph.

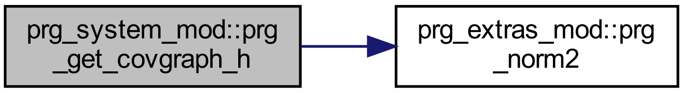
This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

Parameters

sy	System structure (see system_type).
nnStructMindist	Minimun distance between atoms.
nnStruct	The neigbors J to I within Rcut that are all within the box.
nrnnstruct	Number of neigbors to I within Rcut that are all within the box.
bml_type	The bml type for constructing the graph.
gconv_bml	Covanlency graph in bml format.
verbose	Verbosity level.

Definition at line 1957 of file prg_system_mod.F90.

Here is the call graph for this function:



8.32.2.9 prg_get_covgraph_int()

```
subroutine prg_system_mod::prg_get_covgraph_int (
             type(system_type), intent(in) sy,
             real(dp), dimension(:,:), intent(in) nnStructMindist,
             integer, dimension(:,:), intent(in) nnStruct,
             integer, dimension(:), intent(in) nrnnstruct,
             character(20), intent(in) bml_type,
             real(dp) factor,
             type(bml_matrix_t), intent(inout) gcov_bml,
             integer, intent(in) mdimin,
             integer, intent(in), optional verbose )  [private]
```

Definition at line 1896 of file prg_system_mod.F90.

8.32.2.10 prg_get_dihedral()

```
subroutine, public prg_system_mod::prg_get_dihedral (
             real(dp), dimension(:,:), intent(in) coords,
             integer, intent(in) id1,
             integer, intent(in) id2,
             integer, intent(in) id3,
             integer, intent(in) id4,
             real(dp), intent(out) dihedral )
```

Get the dihedral angle given four atomic positions.

Parameters

sy	System structure
id1	Atom index 1
id2	Atom index 1
id3	Atom index 1
id4	Atom index 1
dihedral	Output dihedral angle

Definition at line 1764 of file prg_system_mod.F90.

8.32.2.11 prg_get_distancematrix()

```
subroutine, public prg_system_mod::prg_get_distancematrix (
            real(dp), dimension(:,:), intent(in) coords,
            real(dp), dimension(:,:), intent(out), allocatable dmat )
```

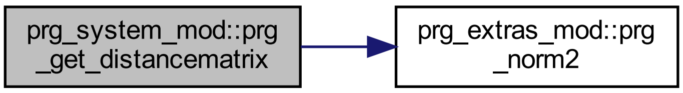
Get the distance matrix.

Parameters

coords	Coordinates of the system (see system_type).
dmat	Distance matrix (nats x nats).

Definition at line 1276 of file prg_system_mod.F90.

Here is the call graph for this function:


prg_system_mod::prg_get_distancematrix
prg_extras_mod::prg_norm2

8.32.2.12 prg_get_nameandext()

```
subroutine, public prg_system_mod::prg_get_nameandext (
            character(len=*), intent(in) fullfilename,
            character(50), intent(inout) filename,
            character(3), intent(inout) ext )
```

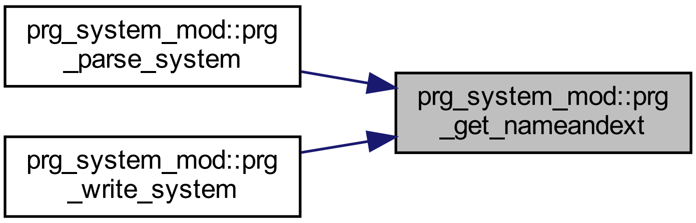
Get the name and extension of a file.

Parameters

fullfilename	Full filename.
filename	Filename of the system.
extension	Extension of the file.

Definition at line 211 of file prg_system_mod.F90.

Here is the caller graph for this function:


prg_system_mod::prg_parse_system
prg_system_mod::prg_write_system
prg_system_mod::prg_get_nameandext

8.32.2.13 prg_get_origin()

```
subroutine, public prg_system_mod::prg_get_origin (
            real(dp), dimension(:,:), intent(in) coords,
            real(dp), dimension(:), intent(inout), allocatable origin )
```

Get the origin of the coordinates.

Parameters

coords	Coordinates of teh system (see system_type).
origin	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1243 of file prg_system_mod.F90.

8.32.2.14 prg_get_partial_atomgraph()

```
subroutine, public prg_system_mod::prg_get_partial_atomgraph (
            type(bml_matrix_t), intent(in) rho_bml,
            integer, dimension(:,:), intent(in) hindex,
            type(bml_matrix_t), intent(inout) gch_bml,
            real(dp), intent(in) threshold,
            integer, intent(in), optional verbose )
```

Get partial subgraph based on the Density matrix.

Parameters

rho_bml	Density matix in bml format.
hindex	Start and end index for every atom in the system.
gch_bml	Atom based graph in bml format.
threshold	Threshold value for constructing the graph.
verbose	Verbosity levels.

Definition at line 2268 of file prg_system_mod.F90.

8.32.2.15 prg_get_recip_vects()

```
subroutine, public prg_system_mod::prg_get_recip_vects (
            real(dp), dimension(:,:), intent(in) lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable recip_vectors,
            real(dp), intent(inout) volr,
            real(dp), intent(inout) volk )
```

Get the volume of the cell and the reciprocal vectors: This soubroutine computes:

- $b_1 = \frac{1}{V_c} a_1 \times a_2$
- $b_2 = \frac{1}{V_c} a_2 \times a_3$
- $b_3 = \frac{1}{V_c} a_3 \times a_1$
- $V_c = ||a_1 \cdot (a_2 \times a_3)||$
- $V_{BZ} = ||b_1 \cdot (b_2 \times b_3)||$

Parameters

lattice_vectors	Lattice vectors for the system.
recip_vectors	Reciprocal vectors of the system.
volr	Volume of the cell.
volk	Volume of the reciprocal cell.

Definition at line 1715 of file prg_system_mod.F90.

8.32.2.16 prg_get_subsystem()

```
subroutine, public prg_system_mod::prg_get_subsystem (
            type(system_type), intent(in) sy,
            integer, intent(in) lsize,
            integer, dimension(:), intent(in) indices,
            type(system_type), intent(inout) sbsy,
            integer, intent(in), optional verbose )
```

Get a subsystem out of the total system.

This will get a subsystem from the total system guided by a partition.

Parameters

sy	System structure (see system_type).
lsize	Core+Hallo subsystem size.
indices	Partition indices.
sbsy	Subsystem to be extracted.

Definition at line 2026 of file prg_system_mod.F90.

8.32.2.17 prg_graph2bml()

```
subroutine, public prg_system_mod::prg_graph2bml (
            integer, dimension(:,:), intent(inout), allocatable graph,
            character(20), intent(in) bml_type,
            type(bml_matrix_t), intent(inout) g_bml )
```

Graph2bml.

Parameters

graph	Atom based graph in "ellpack" like format.
bml_type	Bml type (usually ellpack for graph starage)
g_bml	Graph in bml format.

Definition at line 2585 of file prg_system_mod.F90.

8.32.2.18 prg_graph2vector()

```
subroutine, public prg_system_mod::prg_graph2vector (
            integer, dimension(:,:), intent(inout) graph,
            integer, dimension(:), allocatable vector,
            integer maxnz )
```

Vectorize graph.

Parameters

graph	Ellpack graph.
vector	Vector to store the graph.

Definition at line 2628 of file prg_system_mod.F90.

8.32.2.19 prg_make_random_system()

```
subroutine, public prg_system_mod::prg_make_random_system (
            type(system_type), intent(out) system,
            integer nats,
            integer seed,
            real(dp) lx,
            real(dp) ly,
            real(dp) lz )
```

Make random Xx system.

Parameters

system	System to be construucted.
nats	Number of atoms.
lx	length of the box for the x coordinate.
ly	length of the box for the y coordinate.
lz	length of the box for the z coordinate.

Definition at line 1119 of file prg_system_mod.F90.

8.32.2.20 prg_merge_graph()

```
subroutine, public prg_system_mod::prg_merge_graph (
              integer, dimension(:,:), intent(inout) graph_p,
              integer, dimension(:,:), intent(inout) graph_h )
```

Get partial subgraph based on the Density matrix.

Parameters

graph↩ _p	Density matix based graph in bml format.
graph↩ _h	Hamiltonian matix based graph in bml format.

Definition at line 2426 of file prg_system_mod.F90.

8.32.2.21 prg_merge_graph_adj()

```
subroutine, public prg_system_mod::prg_merge_graph_adj (
              integer, dimension(:,:), intent(inout), allocatable graph_p,
              integer, dimension(:,:), intent(inout), allocatable graph_h,
              integer, dimension(:), intent(inout), allocatable xadj,
              integer, dimension(:), intent(inout), allocatable adjncy )
```

Get partial subgraph based on the Density matrix.

Parameters

graph↩ _p	Density matix based graph in "ellpack type format".
graph↩ _h	Hamiltonian matix based graph in "ellpack type format".
xadj	CSR start values for the adjacency matrix.
adjncy	CSR positions of adjacency matrix.

Definition at line 2477 of file prg_system_mod.F90.

8.32.2.22 prg_molpartition()

```
subroutine, public prg_system_mod::prg_molpartition (
            type(system_type), intent(in) sy,
            integer, intent(inout) npart,
            real(dp), dimension(:,:), intent(in) nnStructMindist,
            integer, dimension(:,:), intent(in) nnStruct,
            integer, dimension(:), intent(in) nrnnstruct,
            character(2), intent(in) hetatm,
            type(graph_partitioning_t), intent(inout) gp,
            integer, intent(inout), optional verbose )
```

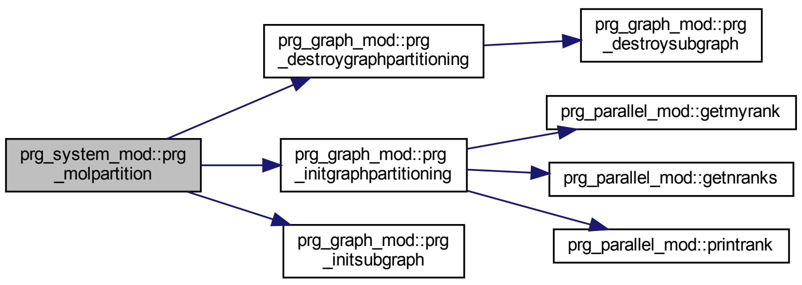
Partition by molecule.

Parameters

sy	System structure.
npart	Number of parts.
nnStructMindist	Minimum distance between neighbors.
nnStruct	The neighbors J to I within Rcut that are all within the box.
nrnnstruct	Number of neighbors to I within Rcut that are all within the box.
hetatm	Atom to be taken as the "center" of the by molecule partition.
gp	Graph partition structure.
verbose	Verbosity level.

Definition at line 2179 of file prg_system_mod.F90.

Here is the call graph for this function:


prg_system_mod::prg_molpartition
prg_graph_mod::prg_destroygraphpartitioning
prg_graph_mod::prg_destroysubgraph
prg_graph_mod::prg_initgraphpartitioning
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::printrank
prg_graph_mod::prg_initsubgraph

8.32.2.23 prg_parameters_to_vectors()

```
subroutine, public prg_system_mod::prg_parameters_to_vectors (
            real(dp), dimension(2,3), intent(in) abc_angles,
            real(dp), dimension(3,3), intent(out) lattice_vector )
```

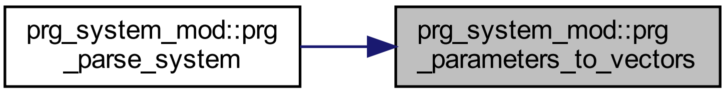
Transforms the lattice parameters into lattice vectors.

Parameters

abc_angles	2x3 array containing the lattice parameters. abc_angles(1,1) = a, abc_angles(1,2) = b, and abc_angles(1,3) = c abc_angles(2,1) = α , abc_angles(2,2) = β and abc_angles(2,3) = γ
lattice_vector	3x3 array containing the lattice vectors. lattice_vector(1,:) = $\overrightarrow{a}$

Definition at line 1165 of file prg_system_mod.F90.

Here is the caller graph for this function:


prg_system_mod::prg_parse_system
prg_system_mod::prg_parameters_to_vectors

8.32.2.24 prg_parse_system()

```
subroutine, public prg_system_mod::prg_parse_system (
            type(system_type), intent(out) system,
            character(len=*) filename,
            character(3), intent(in), optional extin )
```

The parser for the chemical system.

Parameters

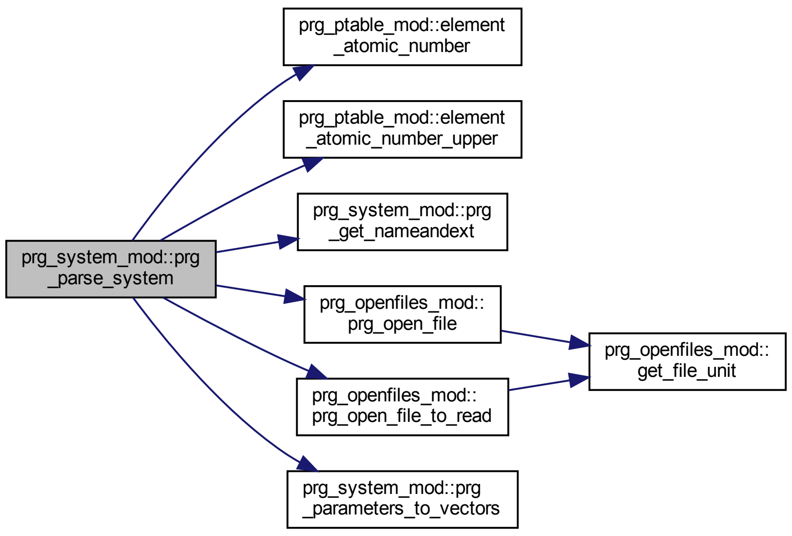
system	System to be constructed.
filename	Filename of the system.
extin	Extension of the file.

Assignment of species index for every atom.

Todo Integrate this loop in the loop for building the splist.

Definition at line 234 of file prg_system_mod.F90.

Here is the call graph for this function:


prg_ptable_mod::element_atomic_number
prg_ptable_mod::element_atomic_number_upper
prg_system_mod::prg_get_nameandext
prg_system_mod::prg_parse_system
prg_openfiles_mod::prg_open_file
prg_openfiles_mod::get_file_unit
prg_openfiles_mod::prg_open_file_to_read
prg_system_mod::prg_parameters_to_vectors

8.32.2.25 prg_replicate()

```
subroutine, public prg_system_mod::prg_replicate (
             real(dp), dimension(:,:), intent(inout), allocatable coords,
             character(2), dimension(:), intent(inout), allocatable symbols,
             real(dp), dimension(:,:), intent(inout) lattice_vectors,
             integer, intent(in) nx,
             integer, intent(in) ny,
             integer, intent(in) nz )
```

Extend/replicate system along lattice vectors.

Parameters

coords	Coordinates of the system (see system_type).
symbols	Symbols for elements.
lattice_vectors	System lattice vectors.
nx	Number of lattice points in the v1 direction.
ny	Number of lattice points in the v2 direction.
nz	Number of lattice points in the v2 direction.

Definition at line 1480 of file prg_system_mod.F90.

8.32.2.26 prg_replicate_system()

```
subroutine, public prg_system_mod::prg_replicate_system (
             type(system_type), intent(inout) sy,
             type(system_type) syf,
             integer, intent(in) nx,
             integer, intent(in) ny,
             integer, intent(in) nz )
```

Extend/replicate a system type along the lattice vectors.

Parameters

sy	System type.
syf	System type output.
nx	Number of lattice points in the v1 direction.
ny	Number of lattice points in the v2 direction.
nz	Number of lattice points in the v2 direction.

Definition at line 1537 of file prg_system_mod.F90.

8.32.2.27 prg_sortadj()

```
subroutine, public prg_system_mod::prg_sortadj (
             integer, dimension(:), intent(inout) xadj,
             integer, dimension(:), intent(inout), allocatable adjncy )
```

Sort adj NOTE: this might not be needed anymre since the bml_get_adj routine is sorting the values.

Definition at line 2685 of file prg_system_mod.F90.

8.32.2.28 prg_translateandfoldtobox()

```
subroutine, public prg_system_mod::prg_translateandfoldtobox (
             real(dp), dimension(:,:), intent(inout), allocatable coords,
             real(dp), dimension(:,:), intent(in) lattice_vectors,
             real(dp), dimension(:), intent(inout), allocatable origin,
             integer, intent(in), optional verbose )
```

Translate and fold to box.

Parameters

coords	Coordinates of the system (see system_type).
lattice_vectors	System lattice vectors.
origin	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1301 of file prg_system_mod.F90.

8.32.2.29 prg_translatetogeomcandfoldtobox()

```
subroutine, public prg_system_mod::prg_translatetogeomcandfoldtobox (
            real(dp), dimension(:,:), intent(inout), allocatable coords,
            real(dp), dimension(:,:), intent(in) lattice_vectors,
            real(dp), dimension(:), intent(inout), allocatable origin )
```

Translate to geometric center.

Parameters

coords	Coordinates of the system (see system_type).
lattice_vectors	System lattice vectors.
origin	(min(x),min(y),min(z)) set as the origin of the system.

Definition at line 1441 of file prg_system_mod.F90.

8.32.2.30 prg_vector2graph()

```
subroutine, public prg_system_mod::prg_vector2graph (
            integer, dimension(:), intent(inout), allocatable vector,
            integer, dimension(:,:), intent(inout) graph,
            integer maxnz )
```

Back to graph.

Parameters

vector	Vector to store the graph.
graph	Ellpack graph.

Definition at line 2657 of file prg_system_mod.F90.

8.32.2.31 prg_vectors_to_parameters()

```
subroutine, public prg_system_mod::prg_vectors_to_parameters (
            real(dp), dimension(3,3), intent(in) lattice_vector,
            real(dp), dimension(2,3), intent(out) abc_angles )
```

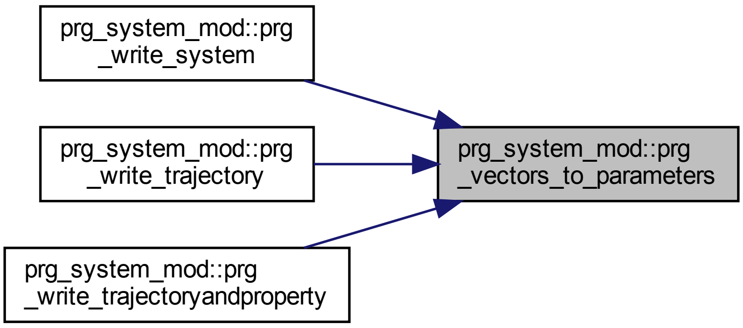
Transforms the lattice vectors into lattice parameters.

Parameters

lattice_vector	3x3 array containing the lattice vectors. lattice_vector(1,:) = $\overrightarrow{a}$
abc_angles	2x3 array containing the lattice parameters. abc_angles(1,1) = a, abc_angles(1,2) = b and abc_angles(1,3) = c abc_angles(2,1) = α, abc_angles(2,2) = β, and abc_angles(2,3) = γ.

Definition at line 1207 of file prg_system_mod.F90.

Here is the caller graph for this function:


prg_system_mod::prg_write_system
prg_system_mod::prg_write_trajectory
prg_system_mod::prg_write_trajectoryandproperty
prg_system_mod::prg_vectors_to_parameters

8.32.2.32 prg_wraparound()

```
subroutine, public prg_system_mod::prg_wraparound (
            real(dp), dimension(:,:), intent(inout), allocatable coords,
            real(dp), dimension(:,:), intent(in) lattice_vectors,
            integer, intent(in) index,
            integer, intent(in), optional verbose )
```

Wrap around atom i using pbc.

Parameters

coords	Coordinates of the system (see system_type).
lattice_vectors	System lattice vectors.
index	Index atom to wrap around

Definition at line 1396 of file prg_system_mod.F90.

8.32.2.33 prg_write_system()

```
subroutine, public prg_system_mod::prg_write_system (
              type(system_type), intent(inout) system,
              character(len=*) filename,
              character(3), intent(in), optional extin )
```

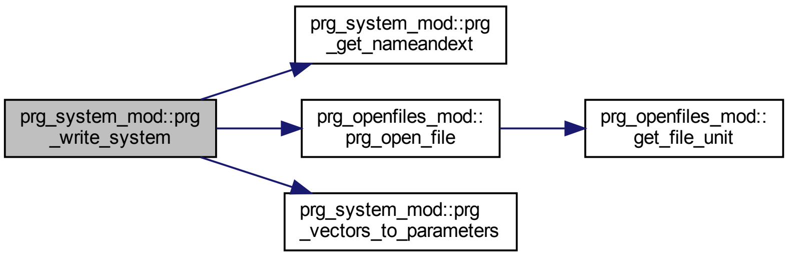
Write system in .xyz, .dat or pdb file.

Parameters

system	System to be constructed.
filename	File name.
extension	Extension of the file.

Definition at line 674 of file prg_system_mod.F90.

Here is the call graph for this function:


prg_system_mod::prg _get_nameandext
prg_system_mod::prg _write_system
prg_openfiles_mod:: prg_open_file
prg_openfiles_mod:: get_file_unit
prg_system_mod::prg _vectors_to_parameters

8.32.2.34 prg_write_trajectory()

```
subroutine, public prg_system_mod::prg_write_trajectory (
              type(system_type), intent(in) system,
              integer, intent(in) iter,
              integer, intent(in) each,
              real(dp), intent(in) prg_deltat,
              character(len=*) filename,
              character(3) extension )
```

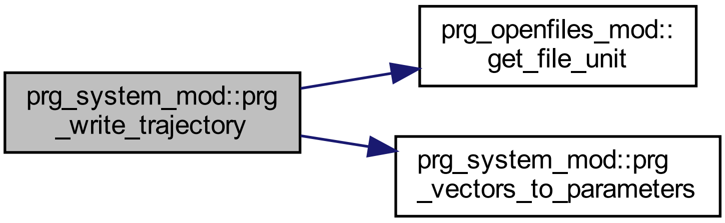
Write trajectory in .xyz, .dat or pdb file.

Parameters

system	System to be appended to the trajectory file.
iter	Simulation step.
each	Writing frequency.
filename	File name for the trajectory.
extension	Extension of the file.

Definition at line 886 of file prg_system_mod.F90.

Here is the call graph for this function:


prg_system_mod::prg_write_trajectory
prg_openfiles_mod::get_file_unit
prg_system_mod::prg_vectors_to_parameters

8.32.2.35 prg_write_trajectoryandproperty()

```
subroutine, public prg_system_mod::prg_write_trajectoryandproperty (
              type(system_type), intent(in) system,
              integer, intent(in) iter,
              integer, intent(in) each,
              real(dp), intent(in) prg_deltat,
              real(dp), dimension(:), intent(in) scalarprop,
              character(len=*) filename,
              character(3) extension )
```

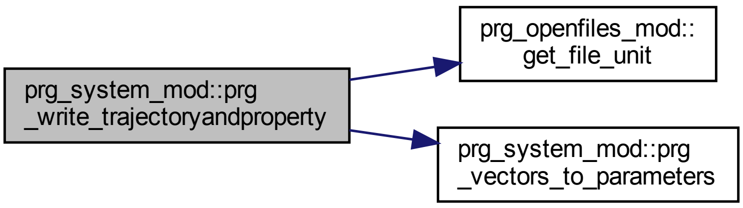
Write trajectory and atomic properties. Only pdb file.

Parameters

system	System to be appended to the trajectory file.
iter	Simulation step.
each	Writing frequency.
prg_deltat	Integration step.
scalarprop	Scalar property to plot on atoms.
filename	File name for the trajectory.
extension	Extension of the file.

Definition at line 1009 of file prg_system_mod.F90.

Here is the call graph for this function:



8.32.3 Variable Documentation

8.32.3.1 dp

```
integer, parameter prg_system_mod::dp = kind(1.0d0)  [private]
```

Definition at line 17 of file prg_system_mod.F90.

8.33 prg_timer_mod Module Reference

The timer module.

Data Types

- type timer_status_t

  Timer status type.

Functions/Subroutines

- subroutine, public timer_prg_init ()

  Initialize timers.
- subroutine prg_timer_getid ()

  Get timer id.
- subroutine, public prg_timer_shutdown ()

  Done with timers.
- subroutine, public prg_timer_start (itimer, tag)

  Start Timing.
- subroutine, public prg_timer_stop (itimer, verbose)

  Stop timing.
- subroutine, public prg_timer_collect ()
- subroutine, public prg_timer_results ()
- real(8) function, public time2milliseconds ()
- subroutine, public prg_print_date_and_time (tag)
- character(2) function, private int2char (ival)

Variables

- integer, parameter dp = kind(1.0d0)
- integer, public loop_timer
- integer, public sp2_timer
- integer, public genx_timer
- integer, public part_timer
- integer, public subgraph_timer
- integer, public deortho_timer
- integer, public ortho_timer
- integer, public zdiag_timer
- integer, public graphsp2_timer
- integer, public subind_timer
- integer, public subext_timer
- integer, public subsp2_timer
- integer, public suball_timer
- integer, public bmult_timer
- integer, public badd_timer
- integer, public dyn_timer
- integer, public mdloop_timer
- integer, public buildz_timer
- integer, public realcoul_timer
- integer, public recipcoul_timer
- integer, public pairpot_timer
- integer, public halfverlet_timer
- integer, public pos_timer
- integer, public nlist_timer
- integer tstart_clock
- integer tstop_clock
- integer tclock_rate
- integer tclock_max
- integer num_timers
- type(timer_status_t), dimension(:), allocatable ptimer

8.33.1 Detailed Description

The timer module.

Sets up timers that can be used to time other routines.

Example use of dynamic timing:

call timer_prg_init()

call prg_timer_start(dyn_timer,"timer_tag")

.... code lines ...

call prg_timer_stop(dyn_timer,1)

This will write the time it takes to execute "code lines" and it will name it "timer_tag"

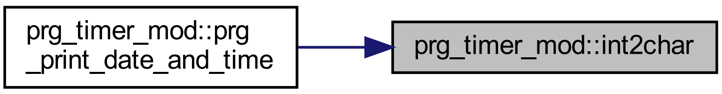
8.33.2 Function/Subroutine Documentation

8.33.2.1 int2char()

```
character(2) function, private prg_timer_mod::int2char (
             integer, intent(in) ival )  [private]
```

Definition at line 394 of file prg_timer_mod.F90.

Here is the caller graph for this function:

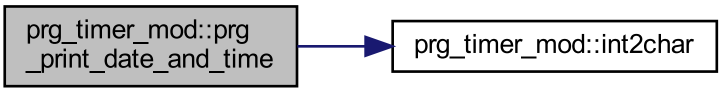


8.33.2.2 prg_print_date_and_time()

```
subroutine, public prg_timer_mod::prg_print_date_and_time (
             character(len=*), intent(in) tag )
```

Definition at line 371 of file prg_timer_mod.F90.

Here is the call graph for this function:

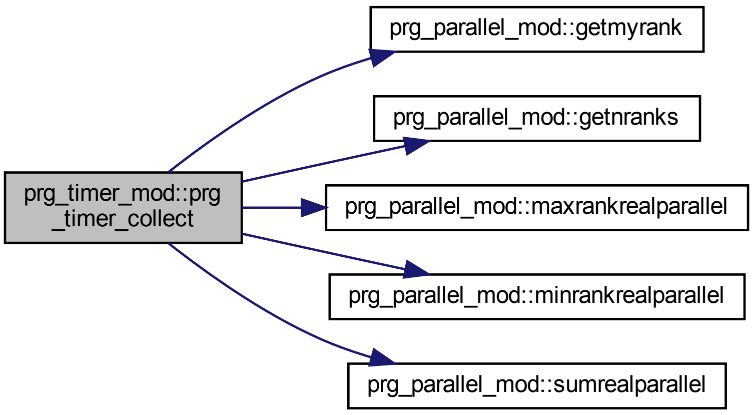


8.33.2.3 prg_timer_collect()

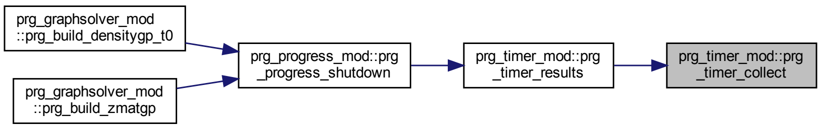
```
subroutine, public prg_timer_mod::prg_timer_collect
```

Definition at line 253 of file prg_timer_mod.F90.

Here is the call graph for this function:


prg_timer_mod::prg_timer_collect
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_parallel_mod::maxrankrealparallel
prg_parallel_mod::minrankrealparallel
prg_parallel_mod::sumrealparallel

Here is the caller graph for this function:


prg_graphsolver_mod::prg_build_densitygp_t0
prg_graphsolver_mod::prg_build_zmatgp
prg_progress_mod::prg_progress_shutdown
prg_timer_mod::prg_timer_results
prg_timer_mod::prg_timer_collect

8.33.2.4 prg_timer_getid()

```
subroutine prg_timer_mod::prg_timer_getid  [private]
```

Get timer id.

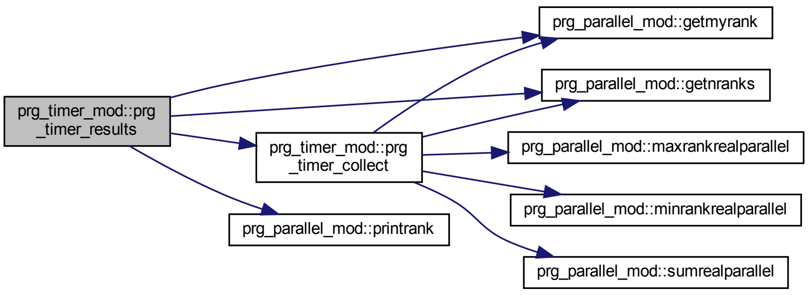
Definition at line 200 of file prg_timer_mod.F90.

8.33.2.5 prg_timer_results()

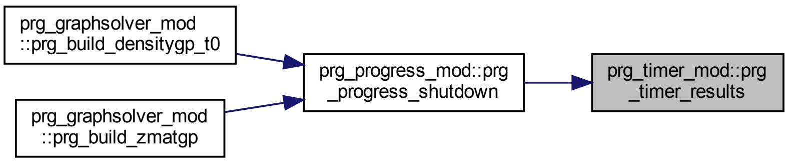
```
subroutine, public prg_timer_mod::prg_timer_results
```

Definition at line 317 of file prg_timer_mod.F90.

Here is the call graph for this function:


prg_timer_mod::prg_timer_results
prg_parallel_mod::getmyrank
prg_parallel_mod::getnranks
prg_timer_mod::prg_timer_collect
prg_parallel_mod::maxrankrealparallel
prg_parallel_mod::minrankrealparallel
prg_parallel_mod::printrank
prg_parallel_mod::sumrealparallel

Here is the caller graph for this function:


prg_graphsolver_mod::prg_build_densitygp_t0
prg_graphsolver_mod::prg_build_zmatgp
prg_progress_mod::prg_progress_shutdown
prg_timer_mod::prg_timer_results

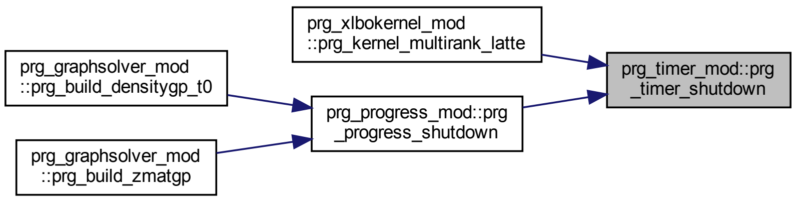
8.33.2.6 prg_timer_shutdown()

```
subroutine, public prg_timer_mod::prg_timer_shutdown
```

Done with timers.

Definition at line 205 of file prg_timer_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod::prg_kernel_multirank_latte
prg_graphsolver_mod::prg_build_densitygp_t0
prg_graphsolver_mod::prg_build_zmatgp
prg_progress_mod::prg_progress_shutdown
prg_timer_mod::prg_timer_shutdown

8.33.2.7 prg_timer_start()

```
subroutine, public prg_timer_mod::prg_timer_start (
              integer, intent(in) itimer,
              character(len=*), intent(in), optional tag )
```

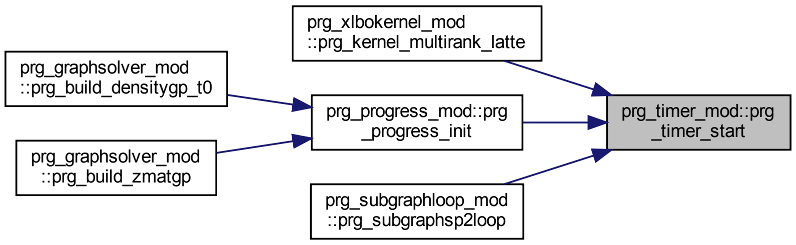
Start Timing.

Parameters

itimer	The index of the timer to start.
tag	Optional parameter to retag the timer on the fly.

Definition at line 215 of file prg_timer_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_graphsolver_mod ::prg_build_densitygp_t0
prg_progress_mod::prg _progress_init
prg_timer_mod::prg _timer_start
prg_graphsolver_mod ::prg_build_zmatgp
prg_subgraphloop_mod ::prg_subgraphsp2loop

8.33.2.8 prg_timer_stop()

```
subroutine, public prg_timer_mod::prg_timer_stop (
              integer, intent(in) itimer,
              integer, intent(in), optional verbose )
```

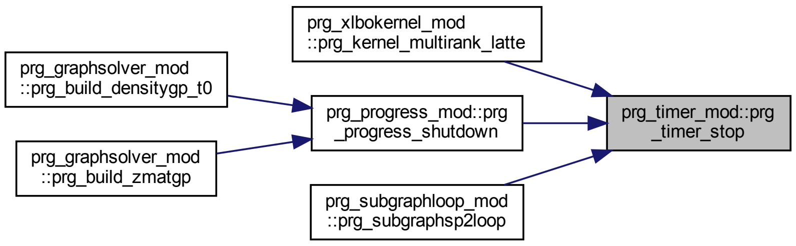
Stop timing.

Parameters

itimer	The index of the timer to stop.
verbose	Optional parameters to print partial times.

Definition at line 233 of file prg_timer_mod.F90.

Here is the caller graph for this function:

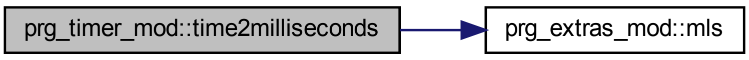


8.33.2.9 time2milliseconds()

```
real(8) function, public prg_timer_mod::time2milliseconds
```

Definition at line 360 of file prg_timer_mod.F90.

Here is the call graph for this function:



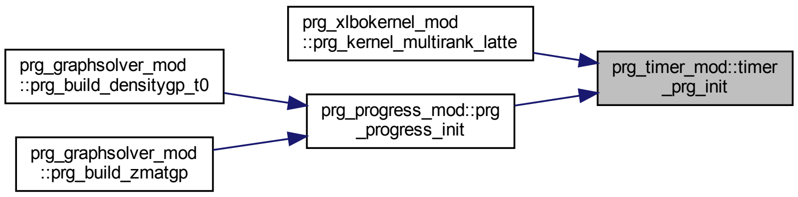
8.33.2.10 timer_prg_init()

```
subroutine, public prg_timer_mod::timer_prg_init
```

Initialize timers.

Definition at line 132 of file prg_timer_mod.F90.

Here is the caller graph for this function:



8.33.3 Variable Documentation

8.33.3.1 badd_timer

```
integer, public prg_timer_mod::badd_timer
```

Definition at line 48 of file prg_timer_mod.F90.

8.33.3.2 bmult_timer

```
integer, public prg_timer_mod::bmult_timer
```

Definition at line 48 of file prg_timer_mod.F90.

8.33.3.3 buildz_timer

```
integer, public prg_timer_mod::buildz_timer
```

Definition at line 49 of file prg_timer_mod.F90.

8.33.3.4 deortho_timer

```
integer, public prg_timer_mod::deortho_timer
```

Definition at line 45 of file prg_timer_mod.F90.

8.33.3.5 dp

```
integer, parameter prg_timer_mod::dp = kind(1.0d0)  [private]
```

Definition at line 32 of file prg_timer_mod.F90.

8.33.3.6 dyn_timer

```
integer, public prg_timer_mod::dyn_timer
```

Definition at line 49 of file prg_timer_mod.F90.

8.33.3.7 genx_timer

```
integer, public prg_timer_mod::genx_timer
```

Definition at line 44 of file prg_timer_mod.F90.

8.33.3.8 graphsp2_timer

```
integer, public prg_timer_mod::graphsp2_timer
```

Definition at line 46 of file prg_timer_mod.F90.

8.33.3.9 halfverlet_timer

```
integer, public prg_timer_mod::halfverlet_timer
```

Definition at line 51 of file prg_timer_mod.F90.

8.33.3.10 loop_timer

```
integer, public prg_timer_mod::loop_timer
```

Definition at line 44 of file prg_timer_mod.F90.

8.33.3.11 mdloop_timer

```
integer, public prg_timer_mod::mdloop_timer
```

Definition at line 49 of file prg_timer_mod.F90.

8.33.3.12 nlist_timer

```
integer, public prg_timer_mod::nlist_timer
```

Definition at line 51 of file prg_timer_mod.F90.

8.33.3.13 num_timers

```
integer prg_timer_mod::num_timers  [private]
```

Definition at line 122 of file prg_timer_mod.F90.

8.33.3.14 ortho_timer

```
integer, public prg_timer_mod::ortho_timer
```

Definition at line 46 of file prg_timer_mod.F90.

8.33.3.15 pairpot_timer

```
integer, public prg_timer_mod::pairpot_timer
```

Definition at line 50 of file prg_timer_mod.F90.

8.33.3.16 part_timer

```
integer, public prg_timer_mod::part_timer
```

Definition at line 45 of file prg_timer_mod.F90.

8.33.3.17 pos_timer

```
integer, public prg_timer_mod::pos_timer
```

Definition at line 51 of file prg_timer_mod.F90.

8.33.3.18 ptimer

```
type (timer_status_t), dimension(:), allocatable prg_timer_mod::ptimer  [private]
```

Definition at line 124 of file prg_timer_mod.F90.

8.33.3.19 realcoul_timer

```
integer, public prg_timer_mod::realcoul_timer
```

Definition at line 50 of file prg_timer_mod.F90.

8.33.3.20 recipcoul_timer

```
integer, public prg_timer_mod::recipcoul_timer
```

Definition at line 50 of file prg_timer_mod.F90.

8.33.3.21 sp2_timer

```
integer, public prg_timer_mod::sp2_timer
```

Definition at line 44 of file prg_timer_mod.F90.

8.33.3.22 suball_timer

```
integer, public prg_timer_mod::suball_timer
```

Definition at line 48 of file prg_timer_mod.F90.

8.33.3.23 subext_timer

```
integer, public prg_timer_mod::subext_timer
```

Definition at line 47 of file prg_timer_mod.F90.

8.33.3.24 subgraph_timer

```
integer, public prg_timer_mod::subgraph_timer
```

Definition at line 45 of file prg_timer_mod.F90.

8.33.3.25 subind_timer

```
integer, public prg_timer_mod::subind_timer
```

Definition at line 47 of file prg_timer_mod.F90.

8.33.3.26 subsp2_timer

```
integer, public prg_timer_mod::subsp2_timer
```

Definition at line 47 of file prg_timer_mod.F90.

8.33.3.27 tclock_max

```
integer prg_timer_mod::tclock_max  [private]
```

Definition at line 121 of file prg_timer_mod.F90.

8.33.3.28 tclock_rate

```
integer prg_timer_mod::tclock_rate  [private]
```

Definition at line 121 of file prg_timer_mod.F90.

8.33.3.29 tstart_clock

```
integer prg_timer_mod::tstart_clock  [private]
```

Definition at line 121 of file prg_timer_mod.F90.

8.33.3.30 tstop_clock

```
integer prg_timer_mod::tstop_clock  [private]
```

Definition at line 121 of file prg_timer_mod.F90.

8.33.3.31 zdiag_timer

```
integer, public prg_timer_mod::zdiag_timer
```

Definition at line 46 of file prg_timer_mod.F90.

8.34 prg_xlbo_mod Module Reference

A module to perform XLBO integration.

Data Types

- type xlbo_type
  General xlbo solver type.

Functions/Subroutines

- subroutine, public prg_parse_xlbo (xlbo, filename)
  The parser for XLBO parser.
- subroutine, public prg_xlbo_nint (charges, n, n_0, n_1, n_2, n_3, n_4, n_5, mdstep, xl)
  This routine integrates the dynamical variable "n".
- subroutine, public prg_xlbo_fcoulupdate (fcoul, charges, n)
  Adjust forces for the linearized XLBOMD functional.

Variables

- integer, parameter dp = kind(1.0d0)
- real(dp), parameter c0 = -6.0_dp
  Coefficients for modified Verlet integration.
- real(dp), parameter c1 = 14.0_dp
- real(dp), parameter c2 = -8.0_dp
- real(dp), parameter c3 = -3.0_dp
- real(dp), parameter c4 = 4.0_dp
- real(dp), parameter c5 = -1.0_dp
- real(dp), parameter kappa = 1.82_dp
  Coefficients for modified Verlet integration.
- real(dp), parameter alpha = 0.018_dp
- real(dp), parameter cc = 0.9_dp

8.34.1 Detailed Description

A module to perform XLBO integration.

This module will be used to compute integrate the dynamical variable "n" in xlbo.

8.34.2 Function/Subroutine Documentation

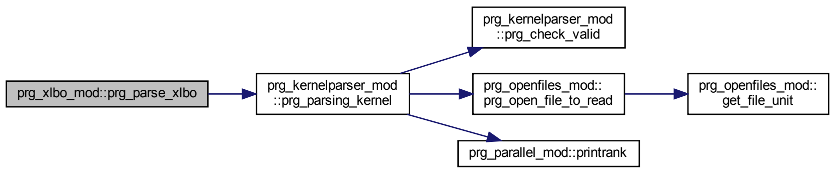
8.34.2.1 prg_parse_xlbo()

```
subroutine, public prg_xlbo_mod::prg_parse_xlbo (
            type(xlbo_type), intent(inout) xlbo,
            character(len=*) filename )
```

The parser for XLBO parser.

Definition at line 62 of file prg_xlbo_mod.F90.

Here is the call graph for this function:


prg_xlbo_mod::prg_parse_xlbo
prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

8.34.2.2 prg_xlbo_fcoulupdate()

```
subroutine, public prg_xlbo_mod::prg_xlbo_fcoulupdate (
            real(dp), dimension(:,:), intent(inout) fcoul,
            real(dp), dimension(:), intent(inout) charges,
            real(dp), dimension(:), intent(inout) n )
```

Adjust forces for the linearized XLBOMD functional.

Parameters

charges	

Definition at line 158 of file prg_xlbo_mod.F90.

8.34.2.3 prg_xlbo_nint()

```
subroutine, public prg_xlbo_mod::prg_xlbo_nint (
            real(dp), dimension(:), intent(in), allocatable charges,
            real(dp), dimension(:), intent(inout), allocatable n,
            real(dp), dimension(:), intent(inout), allocatable n_0,
            real(dp), dimension(:), intent(inout), allocatable n_1,
            real(dp), dimension(:), intent(inout), allocatable n_2,
            real(dp), dimension(:), intent(inout), allocatable n_3,
            real(dp), dimension(:), intent(inout), allocatable n_4,
            real(dp), dimension(:), intent(inout), allocatable n_5,
            integer, intent(in) mdstep,
            type(xlbo_type), intent(in) xl )
```

This routine integrates the dynamical variable "n".

Parameters

charges	

Definition at line 118 of file prg_xlbo_mod.F90.

8.34.3 Variable Documentation

8.34.3.1 alpha

```
real(dp), parameter prg_xlbo_mod::alpha = 0.018_dp  [private]
```

Definition at line 28 of file prg_xlbo_mod.F90.

8.34.3.2 c0

```
real(dp), parameter prg_xlbo_mod::c0 = -6.0_dp  [private]
```

Coefficients for modified Verlet integration.

Definition at line 19 of file prg_xlbo_mod.F90.

8.34.3.3 c1

```
real(dp), parameter prg_xlbo_mod::c1 = 14.0_dp  [private]
```

Definition at line 20 of file prg_xlbo_mod.F90.

8.34.3.4 c2

```
real(dp), parameter prg_xlbo_mod::c2 = -8.0_dp  [private]
```

Definition at line 21 of file prg_xlbo_mod.F90.

8.34.3.5 c3

```
real(dp), parameter prg_xlbo_mod::c3 = -3.0_dp  [private]
```

Definition at line 22 of file prg_xlbo_mod.F90.

8.34.3.6 c4

```
real(dp), parameter prg_xlbo_mod::c4 = 4.0_dp  [private]
```

Definition at line 23 of file prg_xlbo_mod.F90.

8.34.3.7 c5

```
real(dp), parameter prg_xlbo_mod::c5 = -1.0_dp  [private]
```

Definition at line 24 of file prg_xlbo_mod.F90.

8.34.3.8 cc

```
real(dp), parameter prg_xlbo_mod::cc = 0.9_dp  [private]
```

Definition at line 29 of file prg_xlbo_mod.F90.

8.34.3.9 dp

```
integer, parameter prg_xlbo_mod::dp = kind(1.0d0)  [private]
```

Definition at line 16 of file prg_xlbo_mod.F90.

8.34.3.10 kappa

```
real(dp), parameter prg_xlbo_mod::kappa = 1.82_dp  [private]
```

Coefficients for modified Verlet integration.

Definition at line 27 of file prg_xlbo_mod.F90.

8.35 prg_xlbokernel_mod Module Reference

Pre-conditioned O(N) calculation of the kernel for XL-BOMD.

Functions/Subroutines

- subroutine invert (A, AI, N)
- subroutine, public prg_kernel_multirank_latte (KRes, KK0_bml, Res, FelTol, L, LMAX, NUMRANK, HO_↩ bml, mu, beta, RXYZ, Box, Hubbard_U, Element_Pointer, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, nebcoul, totnebcoul, Hinxlist, S_bml, Z_bml, Nocc, Inv_bml, H1_bml, DO_bml, D1_bml, m_rec, threshold, Nr_elem, mdstep, update)

  Compute low rank approximation of (K0∗J)^(-1)∗K0∗(q[n]-n)(for LATTE)
- subroutine prg_update_preconditioner (K0, v, fv, Nr_atoms, threshold)
- subroutine, public prg_kernel_multirank (KRes, KK0_bml, Res, FelTol, L, LMAX, HO_bml, mu, beta, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nn↩ Rx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, S_bml, Z_bml, Nocc, Znuc, Inv_bml, H1_bml, X_bml, Y_bml, DO_bml, D1_bml, m_rec, threshold)
- subroutine, public prg_full_kernel_latte (KK, DO_bml, mu0, RXYZ, Box, Hubbard_U, Element_Pointer, Nr↩ _atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, Hinxlist, S_bml, Z_bml, Inv_bml, D1_bml, H1_bml, HO_bml, Nocc, m_rec, threshold, beta, Nr_elem, nebcoul, totnebcoul)

  Compute full inverse Jacobian of q[n]-n (for LATTE)
- subroutine, public prg_full_kernel (KK, DO_bml, mu0, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_↩ atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDE↩ X_START, H_INDEX_END, S_bml, Z_bml, Inv_bml, D1_bml, H1_bml, HO_bml, Y_bml, X_bml, Nocc, Znuc, m_rec, threshold, beta, diagonal)

  Compute full inverse Jacobian of q[n]-n (for development code)
- subroutine, public prg_kernel_matrix_multirank (KRes, KK0_bml, Res, FelTol, L, LMAX, HO_bml, mu, beta, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMER↩ ATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, S_bml, Z_bml, Nocc, Znuc, Inv_bml, H1_bml, X_bml, Y_bml, DO_bml, D1_bml, m_rec, threshold)

Variables

- integer, parameter dp = kind(1.0d0)

8.35.1 Detailed Description

Pre-conditioned O(N) calculation of the kernel for XL-BOMD.

Here are subroutines implementing Niklasson's scheme for low-rank, Krylov subspace approximation of the kernel.

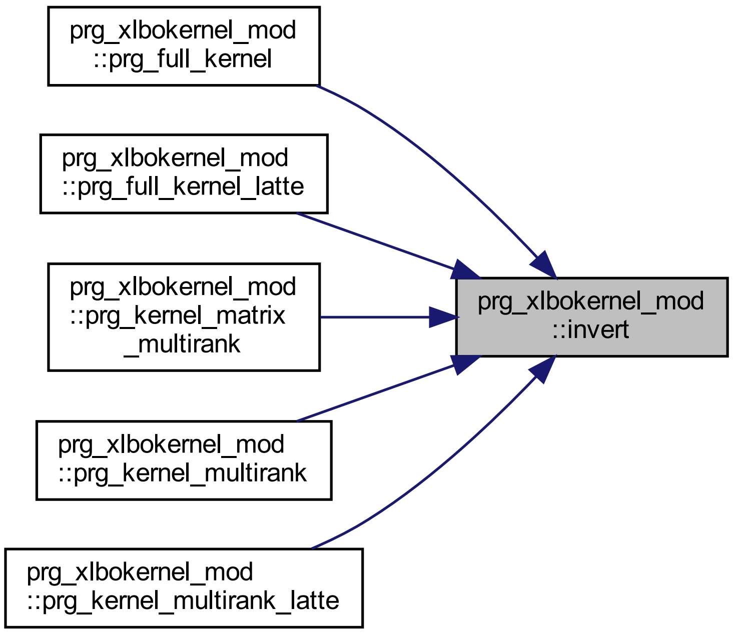
8.35.2 Function/Subroutine Documentation

8.35.2.1 invert()

```
subroutine prg_xlbokernel_mod::invert (
            real(prec), dimension(n,n), intent(in) A,
            real(prec), dimension(n,n), intent(out) AI,
            integer, intent(in) N ) [private]
```

Definition at line 31 of file prg_xlbokernel_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod ::prg_full_kernel
prg_xlbokernel_mod ::prg_full_kernel_latte
prg_xlbokernel_mod ::prg_kernel_matrix _multirank
prg_xlbokernel_mod ::prg_kernel_multirank
prg_xlbokernel_mod ::prg_kernel_multirank_latte
prg_xlbokernel_mod ::invert

8.35.2.2 prg_full_kernel()

```
subroutine, public prg_xlbokernel_mod::prg_full_kernel (
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) KK,
            type(bml_matrix_t), intent(inout) DO_bml,
            real(prec), intent(inout) mu0,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) Hubbard_U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer, intent(in) Nr_atoms,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh,
            real(prec), intent(in) Coulomb_acc,
            real(prec), intent(in) TIMERATIO,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
            integer, dimension(nr_atoms), intent(in) nrnnlist,
            integer, dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
            integer, dimension(nr_atoms), intent(in) H_INDEX_START,
            integer, dimension(nr_atoms), intent(in) H_INDEX_END,
            type(bml_matrix_t), intent(in) S_bml,
            type(bml_matrix_t), intent(in) Z_bml,
            type(bml_matrix_t), dimension(m_rec), intent(in) Inv_bml,
            type(bml_matrix_t), intent(inout) D1_bml,
            type(bml_matrix_t), intent(inout) H1_bml,
            type(bml_matrix_t), intent(in) HO_bml,
            type(bml_matrix_t), intent(inout) Y_bml,
            type(bml_matrix_t), intent(inout) X_bml,
            integer, intent(in) Nocc,
            real(prec), dimension(nr_atoms), intent(in) Znuc,
            integer, intent(in) m_rec,
            real(prec), intent(in) threshold,
            real(prec), intent(in) beta,
            real(prec), dimension(hdim), intent(inout) diagonal )
```

Compute full inverse Jacobian of q[n]-n (for development code)

Parameters

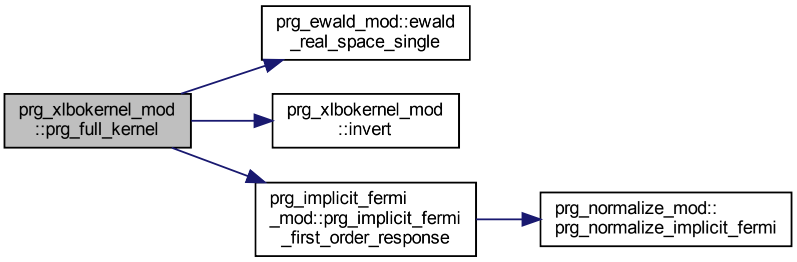
KK	The inverse Jacobian.
DO_bml	Orthogonalized density matrix.
mu0	The chemical potiential.
RX,RY,RZ	Nuclear coordinates.
Lbox	Box dimensions.
Hubbard_U	Hubbard U list.
Element_Type	List to keep track of elements.
Nr_atoms	The number of atoms.
HDIM	Hamiltonian matrix dimension.
Max_Nr_Neigh	Max neighbours for Ewald.
Coulomb_acc	Coulomb accuracy
TIMERATIO	Parameter for Ewald
nnRx,nnRy,nnRz	Neighbour lists.
nrnnlist	Number of neighbours list.
nnType	Refers to original order of atoms.
H_INDEX_START,H_INDEX_END	Lists to keep track of atomic positions in the Hamiltonian.
S_bml	The S matrix.
Z_bml,The	Z matrix.
Inv_bml,Inverses	generated by prg_implicit_fermi_save_inverse.
HO_bml,Orthogonalized	Hamiltonian matrix.

Parameters

D1_bml,H1_bml,Y_bml,X_bml	Auxillary matrices.
Nocc	Occupation.
Znuc	List of nuclear charges.
m_rec	Number of recursion steps.
threshold	Threshold value for matrix truncation.
beta	Scaled inverse temperature.
diagonal	Auxillary vector.

Definition at line 684 of file prg_xlbokernel_mod.F90.

Here is the call graph for this function:


prg_ewald_mod::ewald_real_space_single
prg_xlbokernel_mod::prg_full_kernel
prg_xlbokernel_mod::invert
prg_implicit_fermi_mod::prg_implicit_fermi_first_order_response
prg_normalize_mod::prg_normalize_implicit_fermi

8.35.2.3 prg_full_kernel_latte()

```
subroutine, public prg_xlbokernel_mod::prg_full_kernel_latte (
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) KK,
            type(bml_matrix_t), intent(inout) DO_bml,
            real(prec), intent(inout) mu0,
            real(prec), dimension(3,nr_atoms), intent(in) RXYZ,
            real(prec), dimension(3,3), intent(in) Box,
            real(prec), dimension(nr_elem), intent(in) Hubbard_U,
            integer, dimension(nr_atoms), intent(in) Element_Pointer,
            integer, intent(in) Nr_atoms,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh,
            real(prec), intent(in) Coulomb_acc,
            integer, dimension(hdim), intent(in) Hinxlist,
            type(bml_matrix_t), intent(in) S_bml,
            type(bml_matrix_t), intent(in) Z_bml,
            type(bml_matrix_t), dimension(m_rec), intent(in) Inv_bml,
            type(bml_matrix_t), intent(inout) D1_bml,
            type(bml_matrix_t), intent(inout) H1_bml,
            type(bml_matrix_t), intent(in) HO_bml,
            real(prec), intent(in) Nocc,
            integer, intent(in) m_rec,
            real(prec), intent(in) threshold,
            real(prec), intent(in) beta,
            integer, intent(in) Nr_elem,
            integer, dimension(4,max_nr_neigh,nr_atoms), intent(in) nebcoul,
            integer, dimension(nr_atoms), intent(in) totnebcoul )
```

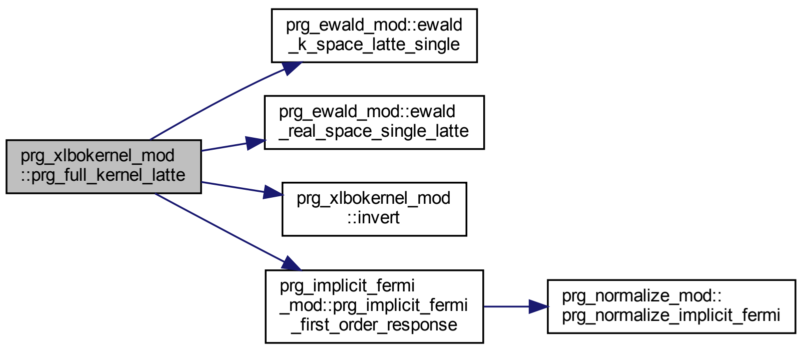
Compute full inverse Jacobian of q[n]-n (for LATTE)

Parameters

KK	The inverse Jacobian.
DO_bml	Orthogonalized density matrix.
mu0	The chemical potiential.
RXYZ	Nuclear coordinates.
Box	Box dimensions.
Hubbard_U	Hubbard U list.
Element_Pointer	List to keep track of elements.
Nr_atoms	The number of atoms.
HDIM	Hamiltonian matrix dimension.
Max_Nr_Neigh	Max neighbours for Ewald.
Coulomb_acc	Coulomb accuracy
Hinxlist	List to keep track of atomic positions in the Hamiltonian.
S_bml	The S matrix.
Z_bml,The	Z matrix.
Inv_bml,Inverses	generated by prg_implicit_fermi_save_inverse.
HO_bml,Orthogonalized	Hamiltonian matrix.
D1_bml,H1_bml,Y_bml,X_bml	Auxillary matrices.
Nocc	Occupation.
m_rec	Number of recursion steps.
threshold	Threshold value for matrix truncation.
beta	Scaled inverse temperature.
Nr_elem	Number of elements in Hubbard list.

Definition at line 515 of file prg_xlbokernel_mod.F90.

Here is the call graph for this function:

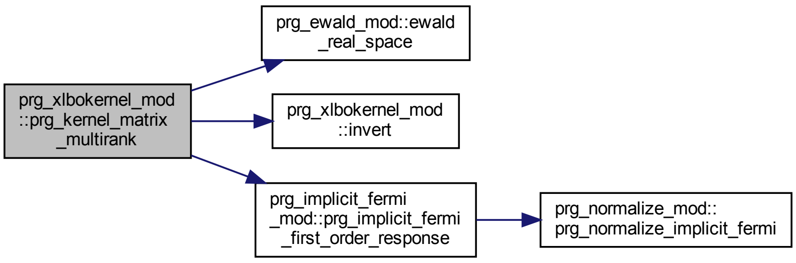

prg_xlbokernel_mod ::prg_full_kernel_latte
prg_ewald_mod::ewald _k_space_latte_single
prg_ewald_mod::ewald _real_space_single_latte
prg_xlbokernel_mod ::invert
prg_implicit_fermi _mod::prg_implicit_fermi _first_order_response
prg_normalize_mod:: prg_normalize_implicit_fermi

8.35.2.4 prg_kernel_matrix_multirank()

```
subroutine, public prg_xlbokernel_mod::prg_kernel_matrix_multirank (
              real(dp), dimension(nr_atoms), intent(out) KRes,
              type(bml_matrix_t), intent(inout) KK0_bml,
              real(dp), dimension(nr_atoms), intent(in) Res,
              real(dp), intent(in) FelTol,
              integer, intent(out) L,
              integer, intent(in) LMAX,
              type(bml_matrix_t), intent(in) HO_bml,
              real(dp), intent(in) mu,
              real(dp), intent(in) beta,
              real(dp), dimension(nr_atoms), intent(in) RX,
              real(dp), dimension(nr_atoms), intent(in) RY,
              real(dp), dimension(nr_atoms), intent(in) RZ,
              real(dp), dimension(3), intent(in) LBox,
              real(dp), dimension(nr_atoms), intent(in) Hubbard_U,
              character(10), dimension(nr_atoms), intent(in) Element_Type,
              integer, intent(in) Nr_atoms,
              integer, intent(in) HDIM,
              integer, intent(in) Max_Nr_Neigh,
              real(dp), intent(in) Coulomb_acc,
              real(dp), intent(in) TIMERATIO,
              real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
              real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
              real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
              integer, dimension(nr_atoms), intent(in) nrnnlist,
              integer, dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
              integer, dimension(nr_atoms), intent(in) H_INDEX_START,
              integer, dimension(nr_atoms), intent(in) H_INDEX_END,
              type(bml_matrix_t), intent(in) S_bml,
              type(bml_matrix_t), intent(in) Z_bml,
              real(dp), intent(in) Nocc,
              real(dp), dimension(nr_atoms), intent(in) Znuc,
              type(bml_matrix_t), dimension(m_rec), intent(in) Inv_bml,
              type(bml_matrix_t), intent(inout) H1_bml,
              type(bml_matrix_t), intent(inout) X_bml,
              type(bml_matrix_t), intent(inout) Y_bml,
              type(bml_matrix_t), intent(inout) DO_bml,
              type(bml_matrix_t), intent(inout) D1_bml,
              integer, intent(in) m_rec,
              real(dp), intent(in) threshold )
```

Definition at line 789 of file prg_xlbokernel_mod.F90.

Here is the call graph for this function:

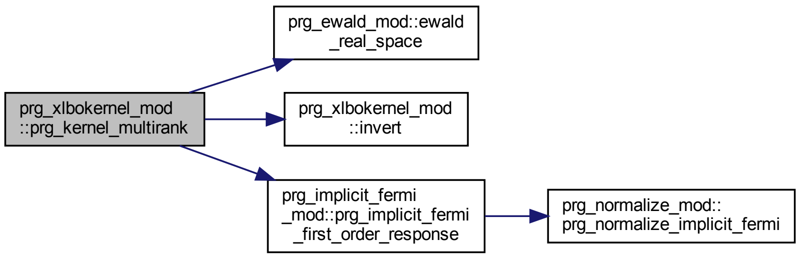


8.35.2.5 prg_kernel_multirank()

```
subroutine, public prg_xlbokernel_mod::prg_kernel_multirank (
            real(dp), dimension(nr_atoms), intent(inout) KRes,
            type(bml_matrix_t), intent(in) KK0_bml,
            real(dp), dimension(nr_atoms), intent(in) Res,
            real(dp), intent(in) FelTol,
            integer, intent(out) L,
            integer, intent(in) LMAX,
            type(bml_matrix_t), intent(in) HO_bml,
            real(dp), intent(in) mu,
            real(dp), intent(in) beta,
            real(dp), dimension(nr_atoms), intent(in) RX,
            real(dp), dimension(nr_atoms), intent(in) RY,
            real(dp), dimension(nr_atoms), intent(in) RZ,
            real(dp), dimension(3), intent(in) LBox,
            real(dp), dimension(nr_atoms), intent(in) Hubbard_U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer, intent(in) Nr_atoms,
            integer, intent(in) HDIM,
            integer, intent(in) Max_Nr_Neigh,
            real(dp), intent(in) Coulomb_acc,
            real(dp), intent(in) TIMERATIO,
            real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
            real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
            real(dp), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
            integer, dimension(nr_atoms), intent(in) nrnnlist,
            integer, dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
            integer, dimension(nr_atoms), intent(in) H_INDEX_START,
            integer, dimension(nr_atoms), intent(in) H_INDEX_END,
            type(bml_matrix_t), intent(in) S_bml,
            type(bml_matrix_t), intent(in) Z_bml,
            integer, intent(in) Nocc,
            real(dp), dimension(nr_atoms), intent(in) Znuc,
            type(bml_matrix_t), dimension(m_rec), intent(in) Inv_bml,
            type(bml_matrix_t), intent(inout) H1_bml,
            type(bml_matrix_t), intent(inout) X_bml,
            type(bml_matrix_t), intent(inout) Y_bml,
            type(bml_matrix_t), intent(inout) D0_bml,
            type(bml_matrix_t), intent(inout) D1_bml,
            integer, intent(in) m_rec,
            real(dp), intent(in) threshold )
```

Definition at line 337 of file prg_xlbokernel_mod.F90.

Here is the call graph for this function:


prg_xlbokernel_mod ::prg_kernel_multirank
prg_ewald_mod::ewald _real_space
prg_xlbokernel_mod ::invert
prg_implicit_fermi _mod::prg_implicit_fermi _first_order_response
prg_normalize_mod:: prg_normalize_implicit_fermi

8.35.2.6 prg_kernel_multirank_latte()

```
subroutine, public prg_xlbokernel_mod::prg_kernel_multirank_latte (
              real(dp), dimension(nr_atoms), intent(inout) KRes,
              type(bml_matrix_t), intent(inout) KK0_bml,
              real(dp), dimension(nr_atoms), intent(in) Res,
              real(dp), intent(in) FelTol,
              integer, intent(out) L,
              integer, intent(in) LMAX,
              integer, intent(in) NUMRANK,
              type(bml_matrix_t), intent(inout) H0_bml,
              real(dp), intent(in) mu,
              real(dp), intent(in) beta,
              real(dp), dimension(3,nr_atoms), intent(in) RXYZ,
              real(dp), dimension(3,3), intent(in) Box,
              real(dp), dimension(nr_elem), intent(in) Hubbard_U,
              integer, dimension(nr_atoms), intent(in) Element_Pointer,
              integer, intent(in) Nr_atoms,
              integer, intent(in) HDIM,
              integer, intent(in) Max_Nr_Neigh,
              real(dp), intent(in) Coulomb_acc,
              integer, dimension(4,max_nr_neigh,nr_atoms), intent(in) nebcoul,
              integer, dimension(nr_atoms), intent(in) totnebcoul,
              integer, dimension(hdim), intent(in) Hinxlist,
              type(bml_matrix_t), intent(inout) S_bml,
              type(bml_matrix_t), intent(inout) Z_bml,
              real(dp), intent(in) Nocc,
              type(bml_matrix_t), dimension(m_rec), intent(inout) Inv_bml,
              type(bml_matrix_t), intent(inout) H1_bml,
              type(bml_matrix_t), intent(inout) DO_bml,
              type(bml_matrix_t), intent(inout) D1_bml,
              integer, intent(in) m_rec,
              real(dp), intent(in) threshold,
              integer, intent(in) Nr_elem,
              integer, intent(in) mdstep,
              integer, intent(in) update )
```

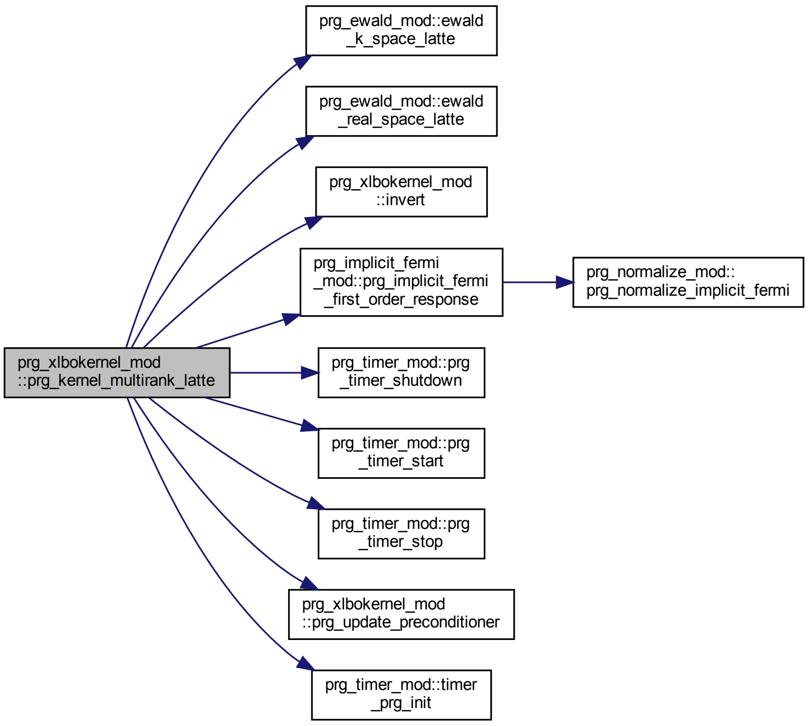
Compute low rank approximation of (K0∗J)^(-1)∗K0∗(q[n]-n)(for LATTE)

Parameters

KRes	The low rank approximation
KK0_bml	The pre-conditioner K0.
Res	The residual q[n]-n
FelTol	Relative error tolerance for approximation
L	Number of vectors used.
LMAX	Maximum nr of vectors to use.
NUMRANK	Nr of vectors to use.
HO_bml,Orthogonalized	Hamiltonian matrix.
mu	The chemical potiential.
beta	Scaled inverse temperature.
RXYZ	Nuclear coordinates.
Box	Box dimensions.
Hubbard_U	Hubbard U list.
Element_Pointer	List to keep track of elements.
Nr_atoms	The number of atoms.
HDIM	Hamiltonian matrix dimension.
Max_Nr_Neigh	Max neighbours for Ewald.
Coulomb_acc	Coulomb accuracy.
nebcoul	Neighbour lists.
totnebcoul	Number of neighbours list.
Hinxlist	List to keep track of atomic positions in the Hamiltonian.
S_bml	The S matrix.
Z_bml,The	Z matrix.
Nocc	Occupation.
Inv_bml,Inverses	generated by prg_implicit_fermi_save_inverse.
DO_bml,D1_bml,H1_bml,Y_bml,X_bml	Auxillary matrices.
m_rec	Number of recursion steps.
threshold	Threshold value for matrix truncation.
Nr_elem	Number of elements in Hubbard list.

Definition at line 87 of file prg_xlbokernel_mod.F90.

Here is the call graph for this function:

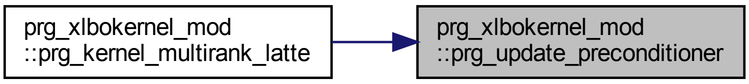

prg_ewald_mod::ewald_k_space_latte
prg_ewald_mod::ewald_real_space_latte
prg_xlbokernel_mod::invert
prg_implicit_fermi_mod::prg_implicit_fermi_first_order_response
prg_normalize_mod::prg_normalize_implicit_fermi
prg_xlbokernel_mod::prg_kernel_multirank_latte
prg_timer_mod::prg_timer_shutdown
prg_timer_mod::prg_timer_start
prg_timer_mod::prg_timer_stop
prg_xlbokernel_mod::prg_update_preconditioner
prg_timer_mod::timer_prg_init

8.35.2.7 prg_update_preconditioner()

```
subroutine prg_xlbokernel_mod::prg_update_preconditioner (
            type(bml_matrix_t), intent(inout) K0,
            real(dp), dimension(nr_atoms), intent(in) v,
            real(dp), dimension(nr_atoms), intent(in) fv,
            integer, intent(in) Nr_atoms,
            real(dp), intent(in) threshold )  [private]
```

Definition at line 266 of file prg_xlbokernel_mod.F90.

Here is the caller graph for this function:


prg_xlbokernel_mod::prg_kernel_multirank_latte
prg_xlbokernel_mod::prg_update_preconditioner

8.35.3 Variable Documentation

8.35.3.1 dp

```
integer, parameter prg_xlbokernel_mod::dp = kind(1.0d0)  [private]
```

Definition at line 20 of file prg_xlbokernel_mod.F90.

8.36 prg_xlkernel_mod Module Reference

Add name.

Data Types

- type xlk_type

Functions/Subroutines

- subroutine, public prg_parse_xlkernel (input, filename)

    The parser for the mixer routines.
- subroutine, public prg_fermi (D0, QQ, ee, gap, Fe_vec, mu0, H, Z, Nocc, T, OccErrLim, MaxIt, HDIM)
- subroutine, public prg_kernel_fermi_full (KK, JJ, D0, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element↩ _Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, public prg_v_kernel_fermi (D0, dq_dv, v, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element↩ _Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, private prg_get_deriv_finite_temp (P1, H0, H1, Nocc, T, Q, ev, fe, mu0, eps, HDIM)
- subroutine, private prg_mmult (alpha, A, B, beta, C, TA, TB, HDIM)
- subroutine, private prg_eig (A, Q, ee, type, HDIM)
- subroutine, private prg_inv (X, XI, HDIM)
- subroutine, public prg_rank1 (verbose)

    Rank1 kernel ....

Variables

- integer, parameter dp = kind(1.0d0)

8.36.1 Detailed Description

Add name.

XL kernel (To be integrated)

Note

This module is still not functional

8.36.2 Function/Subroutine Documentation

8.36.2.1 prg_eig()

```
subroutine, private prg_xlkernel_mod::prg_eig (
             real(prec), dimension(hdim,hdim), intent(in) A,
             real(prec), dimension(hdim,hdim), intent(out) Q,
             real(prec), dimension(hdim), intent(out) ee,
             character(1), intent(in) type,
             integer(prec), intent(in) HDIM )  [private]
```

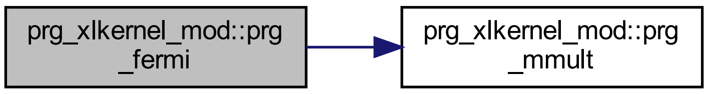
Definition at line 385 of file prg_xlkernel_mod.F90.

8.36.2.2 prg_fermi()

```
subroutine, public prg_xlkernel_mod::prg_fermi (
             real(prec), dimension(hdim,hdim), intent(out) D0,
             real(prec), dimension(hdim,hdim), intent(out) QQ,
             real(prec), dimension(hdim), intent(out) ee,
             real(prec), intent(out) gap,
             real(prec), dimension(hdim), intent(out) Fe_vec,
             real(prec), intent(inout) mu0,
             real(prec), dimension(hdim,hdim), intent(in) H,
             real(prec), dimension(hdim,hdim), intent(in) Z,
             integer(prec), intent(in) Nocc,
             real(prec), intent(in) T,
             real(prec), intent(in) OccErrLim,
             integer(prec), intent(in) MaxIt,
             integer(prec), intent(in) HDIM )
```

Definition at line 89 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:

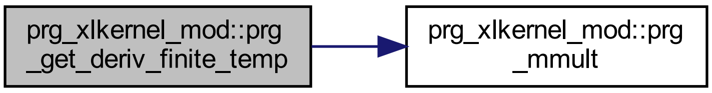


8.36.2.3 prg_get_deriv_finite_temp()

```
subroutine, private prg_xlkernel_mod::prg_get_deriv_finite_temp (
            real(prec), dimension(hdim,hdim), intent(out) P1,
            real(prec), dimension(hdim,hdim), intent(in) H0,
            real(prec), dimension(hdim,hdim), intent(in) H1,
            integer(prec), intent(in) Nocc,
            real(prec), intent(in) T,
            real(prec), dimension(hdim,hdim), intent(in) Q,
            real(prec), dimension(hdim), intent(in) ev,
            real(prec), dimension(hdim), intent(in) fe,
            real(prec), intent(inout) mu0,
            real(prec), intent(in) eps,
            integer(prec), intent(in) HDIM )  [private]
```

Definition at line 307 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:


prg_xlkernel_mod::prg_get_deriv_finite_temp
prg_xlkernel_mod::prg_mmult

8.36.2.4 prg_inv()

```
subroutine, private prg_xlkernel_mod::prg_inv (
            real(prec), dimension(hdim,hdim), intent(in) X,
            real(prec), dimension(hdim,hdim), intent(out) XI,
            integer(prec), intent(in) HDIM )  [private]
```

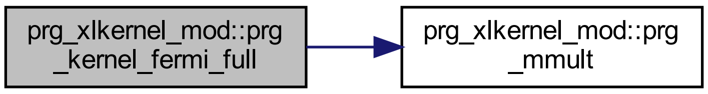
Definition at line 412 of file prg_xlkernel_mod.F90.

8.36.2.5 prg_kernel_fermi_full()

```
subroutine, public prg_xlkernel_mod::prg_kernel_fermi_full (
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) KK,
            real(prec), dimension(nr_atoms,nr_atoms), intent(out) JJ,
            real(prec), dimension(hdim,hdim), intent(inout) D0,
            real(prec), intent(inout) mu0,
            real(prec), intent(inout) mu1,
            real(prec), intent(in) T,
            real(prec), dimension(nr_atoms), intent(in) RX,
            real(prec), dimension(nr_atoms), intent(in) RY,
            real(prec), dimension(nr_atoms), intent(in) RZ,
            real(prec), dimension(3), intent(in) LBox,
            real(prec), dimension(nr_atoms), intent(in) Hubbard_U,
            character(10), dimension(nr_atoms), intent(in) Element_Type,
            integer(prec), intent(in) Nr_atoms,
            integer(prec), intent(in) MaxIt,
            real(prec), intent(in) eps,
            integer(prec), intent(in) m,
            integer(prec), intent(in) HDIM,
            integer(prec), intent(in) Max_Nr_Neigh,
            real(prec), intent(in) Coulomb_acc,
            real(prec), intent(in) TIMERATIO,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
            real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
            integer(prec), dimension(nr_atoms), intent(in) nrnnlist,
            integer(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
            integer(prec), dimension(nr_atoms), intent(in) H_INDEX_START,
            integer(prec), dimension(nr_atoms), intent(in) H_INDEX_END,
            real(prec), dimension(hdim,hdim), intent(in) H,
            real(prec), dimension(hdim,hdim), intent(in) S,
            real(prec), dimension(hdim,hdim), intent(in) Z,
            integer(prec), intent(in) Nocc,
            real(prec), dimension(nr_atoms), intent(in) Znuc,
            real(prec), dimension(hdim,hdim), intent(in) QQ,
            real(prec), dimension(hdim), intent(in) ee,
            real(prec), dimension(hdim), intent(in) Fe_vec )
```

Definition at line 145 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:

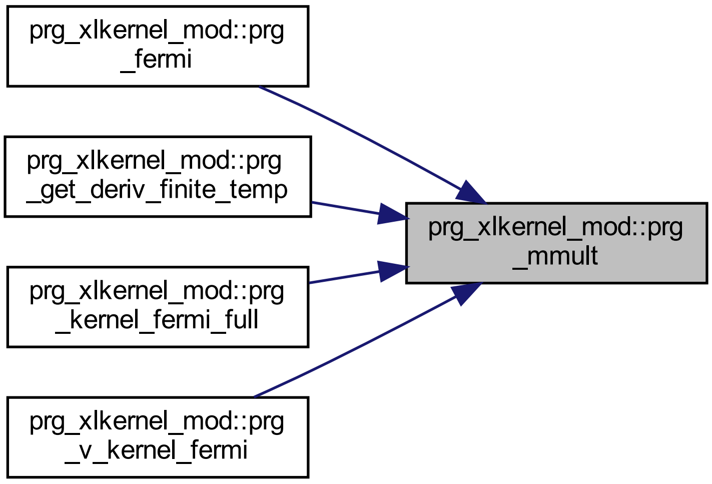

prg_xlkernel_mod::prg_kernel_fermi_full
prg_xlkernel_mod::prg_mmult

8.36.2.6 prg_mmult()

```
subroutine, private prg_xlkernel_mod::prg_mmult (
            real(prec), intent(in) alpha,
            real(prec), dimension(hdim,hdim), intent(in) A,
            real(prec), dimension(hdim,hdim), intent(in) B,
            real(prec), intent(in) beta,
            real(prec), dimension(hdim,hdim), intent(inout) C,
            character(1), intent(in) TA,
            character(1), intent(in) TB,
            integer(prec), intent(in) HDIM )  [private]
```

Definition at line 367 of file prg_xlkernel_mod.F90.

Here is the caller graph for this function:


prg_xlkernel_mod::prg _fermi
prg_xlkernel_mod::prg _get_deriv_finite_temp
prg_xlkernel_mod::prg _kernel_fermi_full
prg_xlkernel_mod::prg _v_kernel_fermi
prg_xlkernel_mod::prg _mmult

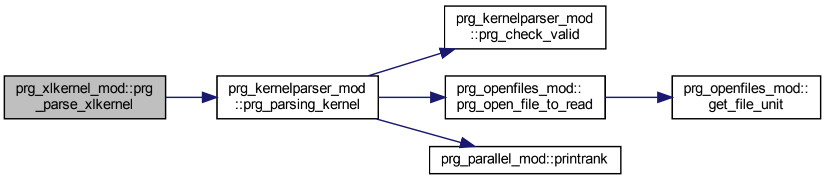
8.36.2.7 prg_parse_xlkernel()

```
subroutine, public prg_xlkernel_mod::prg_parse_xlkernel (
            type(xlk_type), intent(inout) input,
            character(len=*) filename )
```

The parser for the mixer routines.

Definition at line 40 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:


prg_xlkernel_mod::prg _parse_xlkernel
prg_kernelparser_mod ::prg_parsing_kernel
prg_kernelparser_mod ::prg_check_valid
prg_openfiles_mod:: prg_open_file_to_read
prg_openfiles_mod:: get_file_unit
prg_parallel_mod::printrank

8.36.2.8 prg_rank1()

```
subroutine, public prg_xlkernel_mod::prg_rank1 (
             integer, intent(in) verbose )
```

Rank1 kernel ....

Parameters

param1	..
verbose	Different levels of verbosity.

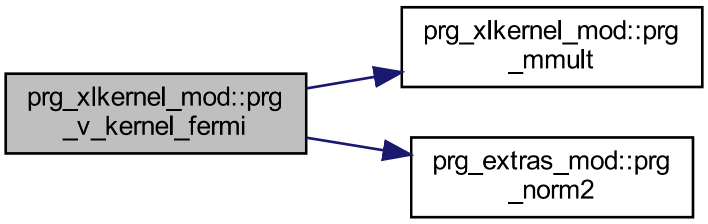
Definition at line 440 of file prg_xlkernel_mod.F90.

8.36.2.9 prg_v_kernel_fermi()

```
subroutine, public prg_xlkernel_mod::prg_v_kernel_fermi (
             real(prec), dimension(hdim,hdim), intent(inout) D0,
             real(prec), dimension(nr_atoms), intent(out) dq_dv,
             real(prec), dimension(nr_atoms), intent(in) v,
             real(prec), intent(inout) mu0,
             real(prec), intent(inout) mu1,
             real(prec), intent(in) T,
             real(prec), dimension(nr_atoms), intent(in) RX,
             real(prec), dimension(nr_atoms), intent(in) RY,
             real(prec), dimension(nr_atoms), intent(in) RZ,
             real(prec), dimension(3), intent(in) LBox,
             real(prec), dimension(nr_atoms), intent(in) Hubbard_U,
             character(10), dimension(nr_atoms), intent(in) Element_Type,
             integer(prec), intent(in) Nr_atoms,
             integer(prec), intent(in) MaxIt,
             real(prec), intent(in) eps,
             integer(prec), intent(in) m,
             integer(prec), intent(in) HDIM,
             integer(prec), intent(in) Max_Nr_Neigh,
             real(prec), intent(in) Coulomb_acc,
             real(prec), intent(in) TIMERATIO,
             real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRx,
             real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRy,
             real(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnRz,
             integer(prec), dimension(nr_atoms), intent(in) nrnnlist,
             integer(prec), dimension(nr_atoms,max_nr_neigh), intent(in) nnType,
             integer(prec), dimension(nr_atoms), intent(in) H_INDEX_START,
             integer(prec), dimension(nr_atoms), intent(in) H_INDEX_END,
             real(prec), dimension(hdim,hdim), intent(in) H,
             real(prec), dimension(hdim,hdim), intent(in) S,
             real(prec), dimension(hdim,hdim), intent(in) Z,
             integer(prec), intent(in) Nocc,
             real(prec), dimension(nr_atoms), intent(in) Znuc,
             real(prec), dimension(hdim,hdim), intent(in) QQ,
             real(prec), dimension(hdim), intent(in) ee,
             real(prec), dimension(hdim), intent(in) Fe_vec )
```

Definition at line 236 of file prg_xlkernel_mod.F90.

Here is the call graph for this function:



8.36.3 Variable Documentation

8.36.3.1 dp

```
integer, parameter prg_xlkernel_mod::dp = kind(1.0d0)  [private]
```

Definition at line 16 of file prg_xlkernel_mod.F90.

Chapter 9

Data Type Documentation

9.1 prg_chebyshev_mod::chebdata_type Type Reference

General Cheb solver type.

Public Attributes

- character(100) flavor
- character(100) bml_type
- character(100) jobname
- integer mdim
- integer ncoeffs
- integer ndim
- integer verbose
- integer npts
- real(dp) atr
- real(dp) bndfil
- real(dp) ef
- real(dp) estep
- real(dp) fermitol
- real(dp) kbt
- real(dp) threshold
- logical getef
- logical jon
- logical trkfunc

9.1.1 Detailed Description

General Cheb solver type.

Definition at line 28 of file prg_chebyshev_mod.F90.

9.1.2 Member Data Documentation

9.1.2.1 atr

```
real(dp) prg_chebyshev_mod::chebdata_type::atr
```

Definition at line 33 of file prg_chebyshev_mod.F90.

9.1.2.2 bml_type

```
character(100) prg_chebyshev_mod::chebdata_type::bml_type
```

Definition at line 30 of file prg_chebyshev_mod.F90.

9.1.2.3 bndfil

```
real(dp) prg_chebyshev_mod::chebdata_type::bndfil
```

Definition at line 33 of file prg_chebyshev_mod.F90.

9.1.2.4 ef

```
real(dp) prg_chebyshev_mod::chebdata_type::ef
```

Definition at line 33 of file prg_chebyshev_mod.F90.

9.1.2.5 estep

```
real(dp) prg_chebyshev_mod::chebdata_type::estep
```

Definition at line 33 of file prg_chebyshev_mod.F90.

9.1.2.6 fermitol

```
real(dp) prg_chebyshev_mod::chebdata_type::fermitol
```

Definition at line 34 of file prg_chebyshev_mod.F90.

9.1.2.7 flavor

```
character(100) prg_chebyshev_mod::chebdata_type::flavor
```

Definition at line 29 of file prg_chebyshev_mod.F90.

9.1.2.8 getef

```
logical prg_chebyshev_mod::chebdata_type::getef
```

Definition at line 35 of file prg_chebyshev_mod.F90.

9.1.2.9 jobname

```
character(100) prg_chebyshev_mod::chebdata_type::jobname
```

Definition at line 30 of file prg_chebyshev_mod.F90.

9.1.2.10 jon

```
logical prg_chebyshev_mod::chebdata_type::jon
```

Definition at line 35 of file prg_chebyshev_mod.F90.

9.1.2.11 kbt

```
real(dp) prg_chebyshev_mod::chebdata_type::kbt
```

Definition at line 34 of file prg_chebyshev_mod.F90.

9.1.2.12 mdim

```
integer prg_chebyshev_mod::chebdata_type::mdim
```

Definition at line 31 of file prg_chebyshev_mod.F90.

9.1.2.13 ncoeffs

```
integer prg_chebyshev_mod::chebdata_type::ncoeffs
```

Definition at line 31 of file prg_chebyshev_mod.F90.

9.1.2.14 ndim

```
integer prg_chebyshev_mod::chebdata_type::ndim
```

Definition at line 31 of file prg_chebyshev_mod.F90.

9.1.2.15 npts

```
integer prg_chebyshev_mod::chebdata_type::npts
```

Definition at line 32 of file prg_chebyshev_mod.F90.

9.1.2.16 threshold

```
real(dp) prg_chebyshev_mod::chebdata_type::threshold
```

Definition at line 34 of file prg_chebyshev_mod.F90.

9.1.2.17 trkfunc

```
logical prg_chebyshev_mod::chebdata_type::trkfunc
```

Definition at line 35 of file prg_chebyshev_mod.F90.

9.1.2.18 verbose

```
integer prg_chebyshev_mod::chebdata_type::verbose
```

Definition at line 31 of file prg_chebyshev_mod.F90.

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

- /tmp/qmd-progress/src/prg_chebyshev_mod.F90

9.2 prg_system_mod::estruct_type Type Reference

Electronic structure type.

Public Attributes

- integer norbs

    Number of orbitals of the system.
- integer nel

    Number of electrons.
- integer, dimension(:,:), allocatable hindex

    Hindex.
- type(bml_matrix_t) ham

    SCC-Hamiltonian of the system.
- type(bml_matrix_t) ham0

    Hamiltonian of the system.
- type(bml_matrix_t) oham

    Orthogonalized Hamiltonian.
- type(bml_matrix_t) over

    Overlap matrix of the system.
- type(bml_matrix_t) rho

    Density matrix of the system.
- type(bml_matrix_t) orho

    Orthogonalized density matrix.
- type(bml_matrix_t) zmat

    Congruence transformation.
- real(dp), dimension(:), allocatable coul_pot_r

    Real Coulombic contribution.
- real(dp), dimension(:), allocatable coul_pot_k

    Reciprocal Coulombic contribution.
- real(dp), dimension(:,:), allocatable skforce

    Slater Koster force.
- real(dp), dimension(:,:), allocatable fpul

    Pulay force.
- real(dp), dimension(:,:), allocatable fscoul

    Nonorthogonal Coulombic force.
- real(dp) eband

    Band energy.

9.2.1 Detailed Description

Electronic structure type.

The electronic structure type.

Definition at line 20 of file prg_system_mod.F90.

9.2.2 Member Data Documentation

9.2.2.1 coul_pot_k

```
real(dp), dimension(:), allocatable prg_system_mod::estruct_type::coul_pot_k
```

Reciprocal Coulombic contribution.

Definition at line 56 of file prg_system_mod.F90.

9.2.2.2 coul_pot_r

```
real(dp), dimension(:), allocatable prg_system_mod::estruct_type::coul_pot_r
```

Real Coulombic contribution.

Definition at line 53 of file prg_system_mod.F90.

9.2.2.3 eband

```
real(dp) prg_system_mod::estruct_type::eband
```

Band energy.

Definition at line 68 of file prg_system_mod.F90.

9.2.2.4 fpul

```
real(dp), dimension(:,:), allocatable prg_system_mod::estruct_type::fpul
```

Pulay force.

Definition at line 62 of file prg_system_mod.F90.

9.2.2.5 fscoul

```
real(dp), dimension(:,:), allocatable prg_system_mod::estruct_type::fscoul
```

Nonorthogonal Coulombic force.

Definition at line 65 of file prg_system_mod.F90.

9.2.2.6 ham

```
type(bml_matrix_t) prg_system_mod::estruct_type::ham
```

SCC-Hamiltonian of the system.

Definition at line 32 of file prg_system_mod.F90.

9.2.2.7 ham0

```
type(bml_matrix_t) prg_system_mod::estruct_type::ham0
```

Hamiltonian of the system.

Definition at line 35 of file prg_system_mod.F90.

9.2.2.8 hindex

```
integer, dimension(:,:), allocatable prg_system_mod::estruct_type::hindex
```

Hindex.

Definition at line 29 of file prg_system_mod.F90.

9.2.2.9 nel

```
integer prg_system_mod::estruct_type::nel
```

Number of electrons.

Definition at line 26 of file prg_system_mod.F90.

9.2.2.10 norbs

```
integer prg_system_mod::estruct_type::norbs
```

Number of orbitals of the system.

Definition at line 23 of file prg_system_mod.F90.

9.2.2.11 oham

```
type(bml_matrix_t) prg_system_mod::estruct_type::oham
```

Orthogonalized Hamiltonian.

Definition at line 38 of file prg_system_mod.F90.

9.2.2.12 orho

```
type(bml_matrix_t) prg_system_mod::estruct_type::orho
```

Orthogonalized density matrix.

Definition at line 47 of file prg_system_mod.F90.

9.2.2.13 over

```
type(bml_matrix_t) prg_system_mod::estruct_type::over
```

Overlap matrix of the system.

Definition at line 41 of file prg_system_mod.F90.

9.2.2.14 rho

```
type(bml_matrix_t) prg_system_mod::estruct_type::rho
```

Density matrix of the system.

Definition at line 44 of file prg_system_mod.F90.

9.2.2.15 skforce

```
real(dp), dimension(:,:), allocatable prg_system_mod::estruct_type::skforce
```

Slater Koster force.

Definition at line 59 of file prg_system_mod.F90.

9.2.2.16 zmat

`type(bml_matrix_t) prg_system_mod::estruct_type::zmat`

Congruence transformation.

Definition at line 50 of file prg_system_mod.F90.

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

- /tmp/qmd-progress/src/prg_system_mod.F90

9.3 prg_genz_mod::genzspinp Type Reference

Input for the genz driver. This type controlls all the variables that are needed by genz.

Public Attributes

- integer verbose

  To have different levels of verbose.
- integer nfirst

  !Lentgth of the "firsts iteration period".
- integer nrefi

  !Initial number of recursive refinements.
- integer nreff

  !Initial number of recursive refinements.
- real(dp) numthresi

  Initial threshold value.
- real(dp) numthresf

  Final threshold value.
- logical integration

  If we want to do XL integration scheme for Z.
- integer igenz

  To keep track of the genz iterations.
- logical zsp

  Logical variable to compute in sparse or dense mode.
- integer mdim

  Max nonzero elements per row for every row see [1] .
- character(20) bml_type

  Matrix format (Dense or Ellpack).

9.3.1 Detailed Description

Input for the genz driver. This type controlls all the variables that are needed by genz.

Definition at line 26 of file prg_genz_mod.F90.

9.3.2 Member Data Documentation

9.3.2.1 bml_type

`character(20) prg_genz_mod::genzspinp::bml_type`

Matrix format (Dense or Ellpack).

Definition at line 59 of file prg_genz_mod.F90.

9.3.2.2 igenz

`integer prg_genz_mod::genzspinp::igenz`

To keep track of the genz iterations.

Definition at line 50 of file prg_genz_mod.F90.

9.3.2.3 integration

`logical prg_genz_mod::genzspinp::integration`

If we want to do XL integration scheme for Z.

Definition at line 47 of file prg_genz_mod.F90.

9.3.2.4 mdim

`integer prg_genz_mod::genzspinp::mdim`

Max nonzero elements per row for every row see [1] .

Definition at line 56 of file prg_genz_mod.F90.

9.3.2.5 nfirst

`integer prg_genz_mod::genzspinp::nfirst`

!Lentgth of the "firsts iteration period".

Definition at line 32 of file prg_genz_mod.F90.

9.3.2.6 nreff

`integer prg_genz_mod::genzspinp::nreff`

!Initial number of recursive refinements.

Definition at line 38 of file prg_genz_mod.F90.

9.3.2.7 nrefi

`integer prg_genz_mod::genzspinp::nrefi`

!Initial number of recursive refinements.

Definition at line 35 of file prg_genz_mod.F90.

9.3.2.8 numthresf

`real(dp) prg_genz_mod::genzspinp::numthresf`

Final threshold value.

Definition at line 44 of file prg_genz_mod.F90.

9.3.2.9 numthresi

`real(dp) prg_genz_mod::genzspinp::numthresi`

Initial threshold value.

Definition at line 41 of file prg_genz_mod.F90.

9.3.2.10 verbose

`integer prg_genz_mod::genzspinp::verbose`

To have different levels of verbose.

Definition at line 29 of file prg_genz_mod.F90.

9.3.2.11 zsp

```
logical prg_genz_mod::genzspinp::zsp
```

Logical variable to compute in sparse or dense mode.

Definition at line 53 of file prg_genz_mod.F90.

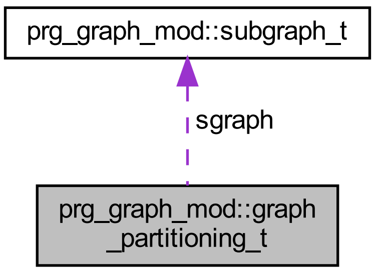
The documentation for this type was generated from the following file:

- /tmp/qmd-progress/src/prg_genz_mod.F90

9.4 prg_graph_mod::graph_partitioning_t Type Reference

Trace per iteration.

Collaboration diagram for prg_graph_mod::graph_partitioning_t:


prg_graph_mod::subgraph_t
sgraph
prg_graph_mod::graph_partitioning_t

Private Attributes

- character(len=100) pname

    Partition name.
- integer myrank

    Local processor.
- integer totalprocs

    Number of processors.
- integer totalparts

    Total number of global partitions.
- integer totalnodes

    Total number of global groups, nodes (or matrix rows)
- integer totalnodes2

    Total number of global nodes (or matrix rows)
- integer globalpartmin

    Minimum global part number.

- integer globalpartmax

  Maximum global part number.
- integer globalpartextent

  Total global parts.
- integer, dimension(:), allocatable localpartmin

  Minimum part per processor.
- integer, dimension(:), allocatable localpartmax

  Maximum part per processor.
- integer, dimension(:), allocatable localpartextent

  Number of parts per processor.
- integer, dimension(:), allocatable order

  Original ordering if required.
- integer, dimension(:), allocatable reorder

  Reordering if required.
- integer nparts

  Total number of local partitions.
- integer, dimension(:), allocatable nnodesinpart

  Number of nodes in each local partition.
- integer, dimension(:), allocatable nnodesinpartall

  Number of nodes in each partition.
- integer, dimension(100) pp

  Sequence for SP2.
- integer maxiter

  Number of SP2 iterations.
- real(dp) ehomo

  Homo value.
- real(dp) elumo

  Lumo value.
- real(dp) mineval

  Min eval for prg_normalize.
- real(dp) maxeval

  Max eval for prg_normalize.
- real(dp), dimension(100) vv

  Trace per iteration.
- type(subgraph_t), dimension(:), allocatable sgraph

  Subgraph details.

9.4.1 Detailed Description

Trace per iteration.

Graph partitioning type

Definition at line 57 of file prg_graph_mod.F90.

9.4.2 Member Data Documentation

9.4.2.1 ehomo

`real(dp) prg_graph_mod::graph_partitioning_t::ehomo [private]`

Homo value.

Definition at line 117 of file prg_graph_mod.F90.

9.4.2.2 elumo

`real(dp) prg_graph_mod::graph_partitioning_t::elumo [private]`

Lumo value.

Definition at line 120 of file prg_graph_mod.F90.

9.4.2.3 globalpartextent

`integer prg_graph_mod::graph_partitioning_t::globalpartextent [private]`

Total global parts.

Definition at line 84 of file prg_graph_mod.F90.

9.4.2.4 globalpartmax

`integer prg_graph_mod::graph_partitioning_t::globalpartmax [private]`

Maximum global part number.

Definition at line 81 of file prg_graph_mod.F90.

9.4.2.5 globalpartmin

`integer prg_graph_mod::graph_partitioning_t::globalpartmin [private]`

Minimum global part number.

Definition at line 78 of file prg_graph_mod.F90.

9.4.2.6 localpartextent

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartextent  [private]
```

Number of parts per processor.

Definition at line 93 of file prg_graph_mod.F90.

9.4.2.7 localpartmax

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartmax  [private]
```

Maximum part per processor.

Definition at line 90 of file prg_graph_mod.F90.

9.4.2.8 localpartmin

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::localpartmin  [private]
```

Minimum part per processor.

Definition at line 87 of file prg_graph_mod.F90.

9.4.2.9 maxeval

```
real(dp) prg_graph_mod::graph_partitioning_t::maxeval  [private]
```

Max eval for prg_normalize.

Definition at line 126 of file prg_graph_mod.F90.

9.4.2.10 maxiter

```
integer prg_graph_mod::graph_partitioning_t::maxiter  [private]
```

Number of SP2 iterations.

Definition at line 114 of file prg_graph_mod.F90.

9.4.2.11 mineval

```
real(dp) prg_graph_mod::graph_partitioning_t::mineval  [private]
```

Min eval for prg_normalize.

Definition at line 123 of file prg_graph_mod.F90.

9.4.2.12 myrank

```
integer prg_graph_mod::graph_partitioning_t::myrank  [private]
```

Local processor.

Definition at line 63 of file prg_graph_mod.F90.

9.4.2.13 nnodesinpart

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::nnodesinpart  [private]
```

Number of nodes in each local partition.

Definition at line 105 of file prg_graph_mod.F90.

9.4.2.14 nnodesinpartall

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::nnodesinpartall  [private]
```

Number of nodes in each partition.

Definition at line 108 of file prg_graph_mod.F90.

9.4.2.15 nparts

```
integer prg_graph_mod::graph_partitioning_t::nparts  [private]
```

Total number of local partitions.

Definition at line 102 of file prg_graph_mod.F90.

9.4.2.16 order

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::order  [private]
```

Original ordering if required.

Definition at line 96 of file prg_graph_mod.F90.

9.4.2.17 pname

```
character(len=100) prg_graph_mod::graph_partitioning_t::pname  [private]
```

Partition name.

Definition at line 60 of file prg_graph_mod.F90.

9.4.2.18 pp

```
integer, dimension(100) prg_graph_mod::graph_partitioning_t::pp  [private]
```

Sequence for SP2.

Definition at line 111 of file prg_graph_mod.F90.

9.4.2.19 reorder

```
integer, dimension(:), allocatable prg_graph_mod::graph_partitioning_t::reorder  [private]
```

Reordering if required.

Definition at line 99 of file prg_graph_mod.F90.

9.4.2.20 sgraph

```
type (subgraph_t), dimension(:), allocatable prg_graph_mod::graph_partitioning_t::sgraph  [private]
```

Subgraph details.

Definition at line 132 of file prg_graph_mod.F90.

9.4.2.21 totalnodes

```
integer prg_graph_mod::graph_partitioning_t::totalnodes  [private]
```

Total number of global groups, nodes (or matrix rows)

Definition at line 72 of file prg_graph_mod.F90.

9.4.2.22 totalnodes2

```
integer prg_graph_mod::graph_partitioning_t::totalnodes2  [private]
```

Total number of global nodes (or matrix rows)

Definition at line 75 of file prg_graph_mod.F90.

9.4.2.23 totalparts

```
integer prg_graph_mod::graph_partitioning_t::totalparts  [private]
```

Total number of global partitions.

Definition at line 69 of file prg_graph_mod.F90.

9.4.2.24 totalprocs

```
integer prg_graph_mod::graph_partitioning_t::totalprocs  [private]
```

Number of processors.

Definition at line 66 of file prg_graph_mod.F90.

9.4.2.25 vv

```
real(dp), dimension(100) prg_graph_mod::graph_partitioning_t::vv  [private]
```

Trace per iteration.

Definition at line 129 of file prg_graph_mod.F90.

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

- /tmp/qmd-progress/src/prg_graph_mod.F90

9.5 prg_graphsp2parser_mod::gsp2data_type Type Reference

General SP2 solver type.

Public Attributes

- character(20) jobname
- character(50) hamfile
- integer verbose
- integer minsp2iter
- integer maxsp2iter
- integer nodesperpart
- integer natoms
- integer partition_count
- real(dp) sp2tol
- real(dp) threshold
- real(dp) bndfil
- real(dp) gthreshold
- real(dp) errlimit
- integer mdim
- integer ndim
- character, dimension(3) sdim
- real(dp), dimension(3) pdim
- character(20) bml_type
- character(10) sp2conv
- character(10) graph_element
- character(10) partition_type
- character(10) partition_refinement
- logical double_jump
- real(dp) covgfact
- real(dp) nlgcut
- integer parteach

9.5.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file prg_graphsp2parser_mod.F90.

9.5.2 Member Data Documentation

9.5.2.1 bml_type

```
character(20) prg_graphsp2parser_mod::gsp2data_type::bml_type
```

Definition at line 44 of file prg_graphsp2parser_mod.F90.

9.5.2.2 bndfil

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::bndfil
```

Definition at line 37 of file prg_graphsp2parser_mod.F90.

9.5.2.3 covgfact

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::covgfact
```

Definition at line 50 of file prg_graphsp2parser_mod.F90.

9.5.2.4 double_jump

```
logical prg_graphsp2parser_mod::gsp2data_type::double_jump
```

Definition at line 49 of file prg_graphsp2parser_mod.F90.

9.5.2.5 errlimit

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::errlimit
```

Definition at line 39 of file prg_graphsp2parser_mod.F90.

9.5.2.6 graph_element

```
character(10) prg_graphsp2parser_mod::gsp2data_type::graph_element
```

Definition at line 46 of file prg_graphsp2parser_mod.F90.

9.5.2.7 gthreshold

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::gthreshold
```

Definition at line 38 of file prg_graphsp2parser_mod.F90.

9.5.2.8 hamfile

```
character(50) prg_graphsp2parser_mod::gsp2data_type::hamfile
```

Definition at line 28 of file prg_graphsp2parser_mod.F90.

9.5.2.9 jobname

```
character(20) prg_graphsp2parser_mod::gsp2data_type::jobname
```

Definition at line 27 of file prg_graphsp2parser_mod.F90.

9.5.2.10 maxsp2iter

```
integer prg_graphsp2parser_mod::gsp2data_type::maxsp2iter
```

Definition at line 31 of file prg_graphsp2parser_mod.F90.

9.5.2.11 mdim

```
integer prg_graphsp2parser_mod::gsp2data_type::mdim
```

Definition at line 40 of file prg_graphsp2parser_mod.F90.

9.5.2.12 minsp2iter

```
integer prg_graphsp2parser_mod::gsp2data_type::minsp2iter
```

Definition at line 30 of file prg_graphsp2parser_mod.F90.

9.5.2.13 natoms

```
integer prg_graphsp2parser_mod::gsp2data_type::natoms
```

Definition at line 33 of file prg_graphsp2parser_mod.F90.

9.5.2.14 ndim

```
integer prg_graphsp2parser_mod::gsp2data_type::ndim
```

Definition at line 41 of file prg_graphsp2parser_mod.F90.

9.5.2.15 nlgcut

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::nlgcut
```

Definition at line 51 of file prg_graphsp2parser_mod.F90.

9.5.2.16 nodesperpart

```
integer prg_graphsp2parser_mod::gsp2data_type::nodesperpart
```

Definition at line 32 of file prg_graphsp2parser_mod.F90.

9.5.2.17 parteach

```
integer prg_graphsp2parser_mod::gsp2data_type::parteach
```

Definition at line 52 of file prg_graphsp2parser_mod.F90.

9.5.2.18 partition_count

```
integer prg_graphsp2parser_mod::gsp2data_type::partition_count
```

Definition at line 34 of file prg_graphsp2parser_mod.F90.

9.5.2.19 partition_refinement

```
character(10) prg_graphsp2parser_mod::gsp2data_type::partition_refinement
```

Definition at line 48 of file prg_graphsp2parser_mod.F90.

9.5.2.20 partition_type

```
character(10) prg_graphsp2parser_mod::gsp2data_type::partition_type
```

Definition at line 47 of file prg_graphsp2parser_mod.F90.

9.5.2.21 pdim

```
real(dp), dimension(3) prg_graphsp2parser_mod::gsp2data_type::pdim
```

Definition at line 43 of file prg_graphsp2parser_mod.F90.

9.5.2.22 sdim

```
character, dimension(3) prg_graphsp2parser_mod::gsp2data_type::sdim
```

Definition at line 42 of file prg_graphsp2parser_mod.F90.

9.5.2.23 sp2conv

```
character(10) prg_graphsp2parser_mod::gsp2data_type::sp2conv
```

Definition at line 45 of file prg_graphsp2parser_mod.F90.

9.5.2.24 sp2tol

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::sp2tol
```

Definition at line 35 of file prg_graphsp2parser_mod.F90.

9.5.2.25 threshold

```
real(dp) prg_graphsp2parser_mod::gsp2data_type::threshold
```

Definition at line 36 of file prg_graphsp2parser_mod.F90.

9.5.2.26 verbose

```
integer prg_graphsp2parser_mod::gsp2data_type::verbose
```

Definition at line 29 of file prg_graphsp2parser_mod.F90.

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

- /tmp/qmd-progress/src/prg_graphsp2parser_mod.F90

9.6 prg_modelham_mod::mham_type Type Reference

General ModelHam type.

Public Attributes

- integer norbs
- integer seed
- character(100) jobname
- character(100) bml_type
- real(dp) ea
- real(dp) eb
- real(dp) dab
- real(dp) daiaj
- real(dp) dbibj
- real(dp) dec
- real(dp) rcoeff
- logical reshuffle

9.6.1 Detailed Description

General ModelHam type.

Definition at line 18 of file prg_modelham_mod.F90.

9.6.2 Member Data Documentation

9.6.2.1 bml_type

```
character(100) prg_modelham_mod::mham_type::bml_type
```

Definition at line 21 of file prg_modelham_mod.F90.

9.6.2.2 dab

```
real(dp) prg_modelham_mod::mham_type::dab
```

Definition at line 24 of file prg_modelham_mod.F90.

9.6.2.3 daiaj

```
real(dp) prg_modelham_mod::mham_type::daiaj
```

Definition at line 25 of file prg_modelham_mod.F90.

9.6.2.4 dbibj

```
real(dp) prg_modelham_mod::mham_type::dbibj
```

Definition at line 26 of file prg_modelham_mod.F90.

9.6.2.5 dec

```
real(dp) prg_modelham_mod::mham_type::dec
```

Definition at line 27 of file prg_modelham_mod.F90.

9.6.2.6 ea

```
real(dp) prg_modelham_mod::mham_type::ea
```

Definition at line 22 of file prg_modelham_mod.F90.

9.6.2.7 eb

```
real(dp) prg_modelham_mod::mham_type::eb
```

Definition at line 23 of file prg_modelham_mod.F90.

9.6.2.8 jobname

```
character(100) prg_modelham_mod::mham_type::jobname
```

Definition at line 20 of file prg_modelham_mod.F90.

9.6.2.9 norbs

```
integer prg_modelham_mod::mham_type::norbs
```

Definition at line 19 of file prg_modelham_mod.F90.

9.6.2.10 rcoeff

```
real(dp) prg_modelham_mod::mham_type::rcoeff
```

Definition at line 27 of file prg_modelham_mod.F90.

9.6.2.11 reshuffle

```
logical prg_modelham_mod::mham_type::reshuffle
```

Definition at line 28 of file prg_modelham_mod.F90.

9.6.2.12 seed

```
integer prg_modelham_mod::mham_type::seed
```

Definition at line 19 of file prg_modelham_mod.F90.

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

- /tmp/qmd-progress/src/prg_modelham_mod.F90

9.7 prg_pulaymixer_mod::mx_type Type Reference

Public Attributes

- character(20) mixertype

    Type or mixing scheme to be used (Linear or Pulay)
- integer verbose

    Verbosity level.
- integer mpulay

    Pulay dimension for matrix.
- real(dp) mixcoeff

    Coefficient for mixing.
- logical mixeron

    Mixer on or off (Not implemented)

9.7.1 Detailed Description

Definition at line 17 of file prg_pulaymixer_mod.F90.

9.7.2 Member Data Documentation

9.7.2.1 mixcoeff

```
real(dp) prg_pulaymixer_mod::mx_type::mixcoeff
```

Coefficient for mixing.

Definition at line 29 of file prg_pulaymixer_mod.F90.

9.7.2.2 mixeron

```
logical prg_pulaymixer_mod::mx_type::mixeron
```

Mixer on or off (Not implemented)

Definition at line 32 of file prg_pulaymixer_mod.F90.

9.7.2.3 mixertype

```
character(20) prg_pulaymixer_mod::mx_type::mixertype
```

Type or mixing scheme to be used (Linear or Pulay)

Definition at line 20 of file prg_pulaymixer_mod.F90.

9.7.2.4 mpulay

```
integer prg_pulaymixer_mod::mx_type::mpulay
```

Pulay dimension for matrix.

Definition at line 26 of file prg_pulaymixer_mod.F90.

9.7.2.5 verbose

```
integer prg_pulaymixer_mod::mx_type::verbose
```

Verbosity level.

Definition at line 23 of file prg_pulaymixer_mod.F90.

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

- /tmp/qmd-progress/src/prg_pulaymixer_mod.F90

9.8 prg_extras_mod::prg_memory_consumption Interface Reference

Private Member Functions

- subroutine prg_memory_consumption (vm_peak, vm_size, pid, ppid)

9.8.1 Detailed Description

Definition at line 15 of file prg_extras_mod.F90.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 prg_memory_consumption()

```
subroutine prg_extras_mod::prg_memory_consumption::prg_memory_consumption (
             integer(c_long_long), intent(inout) vm_peak,
             integer(c_long_long), intent(inout) vm_size,
             integer(c_long_long), intent(inout) pid,
             integer(c_long_long), intent(inout) ppid )  [private]
```

Definition at line 17 of file prg_extras_mod.F90.

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

- /tmp/qmd-progress/src/prg_extras_mod.F90

9.9 prg_parallel_mod::rankreducedata_t Type Reference

Data structure for rection over MPI ranks.

Private Attributes

- real(dp) val

  Data value.
- integer rank

  MPI rank.

9.9.1 Detailed Description

Data structure for rection over MPI ranks.

Definition at line 72 of file prg_parallel_mod.F90.

9.9.2 Member Data Documentation

9.9.2.1 rank

```
integer prg_parallel_mod::rankreducedata_t::rank  [private]
```

MPI rank.

Definition at line 78 of file prg_parallel_mod.F90.

9.9.2.2 val

```
real(dp) prg_parallel_mod::rankreducedata_t::val  [private]
```

Data value.

Definition at line 75 of file prg_parallel_mod.F90.

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

- /tmp/qmd-progress/src/prg_parallel_mod.F90

9.10 prg_response_mod::respdata_type Type Reference

Public Attributes

- character(20) respmode
- character(20) typeofpert
- character(20) bmltype
- integer mdim
- real(dp) numthresh
- logical computedipole
- logical getresponse
- real(dp) fieldintensity
- real(dp), dimension(3) field

9.10.1 Detailed Description

Definition at line 21 of file prg_response_mod.F90.

9.10.2 Member Data Documentation

9.10.2.1 bmltype

```
character(20) prg_response_mod::respdata_type::bmltype
```

Definition at line 24 of file prg_response_mod.F90.

9.10.2.2 computedipole

```
logical prg_response_mod::respdata_type::computedipole
```

Definition at line 27 of file prg_response_mod.F90.

9.10.2.3 field

```
real(dp), dimension(3) prg_response_mod::respdata_type::field
```

Definition at line 30 of file prg_response_mod.F90.

9.10.2.4 fieldintensity

```
real(dp) prg_response_mod::respdata_type::fieldintensity
```

Definition at line 29 of file prg_response_mod.F90.

9.10.2.5 getresponse

```
logical prg_response_mod::respdata_type::getresponse
```

Definition at line 28 of file prg_response_mod.F90.

9.10.2.6 mdim

```
integer prg_response_mod::respdata_type::mdim
```

Definition at line 25 of file prg_response_mod.F90.

9.10.2.7 numthresh

```
real(dp) prg_response_mod::respdata_type::numthresh
```

Definition at line 26 of file prg_response_mod.F90.

9.10.2.8 respmode

```
character(20) prg_response_mod::respdata_type::respmode
```

Definition at line 22 of file prg_response_mod.F90.

9.10.2.9 typeofpert

```
character(20) prg_response_mod::respdata_type::typeofpert
```

Definition at line 23 of file prg_response_mod.F90.

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

- /tmp/qmd-progress/src/prg_response_mod.F90

9.11 prg_syrotation_mod::rotation_type Type Reference

Rotation type.

Public Attributes

- character(20) jobname
- character(50) typeofrot
- integer patom1

  Atomic point to determine the initial orientation.
- integer patom2

  Atomic point to determine initial orientation.
- integer catom

  Atomic point to determine the rotation center.
- integer catom2

  Atomic point to determine a second rotation center.
- real(dp), dimension(3) pq1

  Point to determine initial orientation.
- real(dp), dimension(3) pq2

  Point to determine final orientation.
- real(dp), dimension(3) v1

  Initial orientation.
- real(dp), dimension(3) v2

  Final orientation.
- real(dp), dimension(3) vq

  Center of rotation.
- integer, dimension(2) rotate_atoms

  First and last rotated atom in the list.

9.11.1 Detailed Description

Rotation type.

Definition at line 14 of file prg_syrotation_mod.F90.

9.11.2 Member Data Documentation

9.11.2.1 catom

`integer prg_syrotation_mod::rotation_type::catom`

Atomic point to determine the rotation center.

Definition at line 22 of file prg_syrotation_mod.F90.

9.11.2.2 catom2

`integer prg_syrotation_mod::rotation_type::catom2`

Atomic point to determine a second rotation center.

Definition at line 24 of file prg_syrotation_mod.F90.

9.11.2.3 jobname

`character(20) prg_syrotation_mod::rotation_type::jobname`

Definition at line 15 of file prg_syrotation_mod.F90.

9.11.2.4 patom1

`integer prg_syrotation_mod::rotation_type::patom1`

Atomic point to determine the initial orientation.

Definition at line 18 of file prg_syrotation_mod.F90.

9.11.2.5 patom2

`integer prg_syrotation_mod::rotation_type::patom2`

Atomic point to determine initial orientation.

Definition at line 20 of file prg_syrotation_mod.F90.

9.11.2.6 pq1

```
real(dp), dimension(3) prg_syrotation_mod::rotation_type::pq1
```

Point to determine initial orientation.

Definition at line 26 of file prg_syrotation_mod.F90.

9.11.2.7 pq2

```
real(dp), dimension(3) prg_syrotation_mod::rotation_type::pq2
```

Point to determine final orientation.

Definition at line 28 of file prg_syrotation_mod.F90.

9.11.2.8 rotate_atoms

```
integer, dimension(2) prg_syrotation_mod::rotation_type::rotate_atoms
```

First and last rotated atom in the list.

Definition at line 36 of file prg_syrotation_mod.F90.

9.11.2.9 typeofrot

```
character(50) prg_syrotation_mod::rotation_type::typeofrot
```

Definition at line 16 of file prg_syrotation_mod.F90.

9.11.2.10 v1

```
real(dp), dimension(3) prg_syrotation_mod::rotation_type::v1
```

Initial orientation.

Definition at line 30 of file prg_syrotation_mod.F90.

9.11.2.11 v2

```
real(dp), dimension(3) prg_syrotation_mod::rotation_type::v2
```

Final orientation.

Definition at line 32 of file prg_syrotation_mod.F90.

9.11.2.12 vq

```
real(dp), dimension(3) prg_syrotation_mod::rotation_type::vq
```

Center of rotation.

Definition at line 34 of file prg_syrotation_mod.F90.

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

- /tmp/qmd-progress/src/prg_syrotation_mod.F90

9.12 prg_sp2parser_mod::sp2data_type Type Reference

General SP2 solver type.

Public Attributes

- character(20) jobname
- integer verbose
- integer minsp2iter
- integer maxsp2iter
- real(dp) sp2tol
- real(dp) threshold
- real(dp) bndfil
- integer mdim
- integer ndim
- character, dimension(3) sdim
- real(dp), dimension(3) pdim
- character(20) bml_type
- character(10) sp2conv
- character(10) flavor

9.12.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file prg_sp2parser_mod.F90.

9.12.2 Member Data Documentation

9.12.2.1 bml_type

`character(20) prg_sp2parser_mod::sp2data_type::bml_type`

Definition at line 38 of file prg_sp2parser_mod.F90.

9.12.2.2 bndfil

`real(dp) prg_sp2parser_mod::sp2data_type::bndfil`

Definition at line 33 of file prg_sp2parser_mod.F90.

9.12.2.3 flavor

`character(10) prg_sp2parser_mod::sp2data_type::flavor`

Definition at line 40 of file prg_sp2parser_mod.F90.

9.12.2.4 jobname

`character(20) prg_sp2parser_mod::sp2data_type::jobname`

Definition at line 27 of file prg_sp2parser_mod.F90.

9.12.2.5 maxsp2iter

`integer prg_sp2parser_mod::sp2data_type::maxsp2iter`

Definition at line 30 of file prg_sp2parser_mod.F90.

9.12.2.6 mdim

`integer prg_sp2parser_mod::sp2data_type::mdim`

Definition at line 34 of file prg_sp2parser_mod.F90.

9.12.2.7 minsp2iter

`integer prg_sp2parser_mod::sp2data_type::minsp2iter`

Definition at line 29 of file prg_sp2parser_mod.F90.

9.12.2.8 ndim

`integer prg_sp2parser_mod::sp2data_type::ndim`

Definition at line 35 of file prg_sp2parser_mod.F90.

9.12.2.9 pdim

`real(dp), dimension(3) prg_sp2parser_mod::sp2data_type::pdim`

Definition at line 37 of file prg_sp2parser_mod.F90.

9.12.2.10 sdim

`character, dimension(3) prg_sp2parser_mod::sp2data_type::sdim`

Definition at line 36 of file prg_sp2parser_mod.F90.

9.12.2.11 sp2conv

`character(10) prg_sp2parser_mod::sp2data_type::sp2conv`

Definition at line 39 of file prg_sp2parser_mod.F90.

9.12.2.12 sp2tol

```
real(dp) prg_sp2parser_mod::sp2data_type::sp2tol
```

Definition at line 31 of file prg_sp2parser_mod.F90.

9.12.2.13 threshold

```
real(dp) prg_sp2parser_mod::sp2data_type::threshold
```

Definition at line 32 of file prg_sp2parser_mod.F90.

9.12.2.14 verbose

```
integer prg_sp2parser_mod::sp2data_type::verbose
```

Definition at line 28 of file prg_sp2parser_mod.F90.

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

- /tmp/qmd-progress/src/prg_sp2parser_mod.F90

9.13 prg_graph_mod::subgraph_t Type Reference

Subgraph type.

Private Attributes

- integer part

  Partition number.
- integer hsize

  Size of original matrix (h x h)
- integer lsize

  Size of full subgraph (l x l)
- integer llsize

  Size of core subgraph.
- integer, dimension(:), allocatable core_halo_index

  Indeces from original matrix for subgraph core+halo extraction.
- integer, dimension(:), allocatable nodeinpart

  Nodes in this partition.

9.13.1 Detailed Description

Subgraph type.

Definition at line 31 of file prg_graph_mod.F90.

9.13.2 Member Data Documentation

9.13.2.1 core_halo_index

```
integer, dimension(:), allocatable prg_graph_mod::subgraph_t::core_halo_index  [private]
```

Indeces from original matrix for subgraph core+halo extraction.

Definition at line 46 of file prg_graph_mod.F90.

9.13.2.2 hsize

```
integer prg_graph_mod::subgraph_t::hsize  [private]
```

Size of original matrix (h x h)

Definition at line 37 of file prg_graph_mod.F90.

9.13.2.3 llsize

```
integer prg_graph_mod::subgraph_t::llsize  [private]
```

Size of core subgraph.

Definition at line 43 of file prg_graph_mod.F90.

9.13.2.4 lsize

```
integer prg_graph_mod::subgraph_t::lsize  [private]
```

Size of full subgraph (l x l)

Definition at line 40 of file prg_graph_mod.F90.

9.13.2.5 nodeinpart

```
integer, dimension(:), allocatable prg_graph_mod::subgraph_t::nodeinpart  [private]
```

Nodes in this partition.

Definition at line 49 of file prg_graph_mod.F90.

9.13.2.6 part

```
integer prg_graph_mod::subgraph_t::part  [private]
```

Partition number.

Definition at line 34 of file prg_graph_mod.F90.

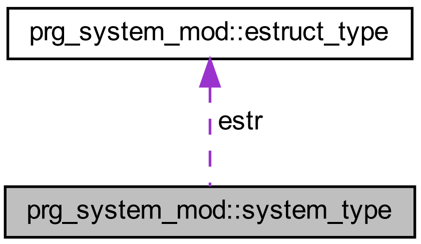
The documentation for this type was generated from the following file:

- /tmp/qmd-progress/src/prg_graph_mod.F90

9.14 prg_system_mod::system_type Type Reference

System type.

Collaboration diagram for prg_system_mod::system_type:



Public Attributes

- integer nats = 0

  Number of atoms of the system.
- character(2), dimension(:), allocatable symbol

  Chemical Symbols for every atom of the system. Symbol can be recovered using ptable module and calling the following routine:
- integer, dimension(:), allocatable atomic_number

  Atomic number for every atom in the system.
- real(dp), dimension(:,:), allocatable coordinate

  Coordinates of every atom in the system. Allocation:
- real(dp), dimension(:,:), allocatable velocity

  Velocities for every atom in the system. Allocation:
- real(dp), dimension(:,:), allocatable force

  Forces acting on every atom in the system. Allocation:
- real(dp), dimension(:), allocatable net_charge

  Charges of every atom in the system. Allocation:
- real(dp), dimension(:), allocatable mass

  Mass of every atom in the system. These can be automatically loaded by using the structures of the ptable mod:
- real(dp), dimension(:,:), allocatable lattice_vector

  Lattice vectors of the system. Use the prg_vectors_to_parameters and parameters_to_vector to transform from lattice vector to lattice parameters. Allocation:
- real(dp), dimension(:,:), allocatable recip_vector

  Reciprocal vectors of the system. Allocation:
- real(dp) volr

  Volume of the system (direct space).
- real(dp) volk

  Volume of the system (direct space).
- integer nsp

  Number of different species. Number of species or number of differet antom types (symbols) in the system. This integer is alwas less or equal than the total number of atoms (nsp <= nats). This information can also be found in tbparams structure and the following equality holds:
- integer, dimension(:), allocatable spindex

  Species index. It gives the species index of a particulat atom. Allocation:
- character(2), dimension(:), allocatable splist

  Species symbol list. A list with the different species e.g. H, C, N, etc with the order corresponding to the appearence in systemsymbol. Allocation:
- integer, dimension(:), allocatable spatnum

  Species atomic number list. A list with the atomic numbers for every species Allocation:
- real(dp), dimension(:), allocatable spmass

  Species mass list. A list with the atomic mass for every species Allocation:
- real(dp), dimension(:), allocatable userdef

  User define field.
- integer, dimension(:), allocatable resindex

  Residue index.
- character(3), dimension(:), allocatable resname

  Residue name.
- character(3), dimension(:), allocatable atomname

  Atom name (to distinguish atoms with same symbol)
- type(estruct_type) estr

  Electronic structure.

9.14.1 Detailed Description

System type.

The molecular system type.

Definition at line 73 of file prg_system_mod.F90.

9.14.2 Member Data Documentation

9.14.2.1 atomic_number

```
integer, dimension(:), allocatable prg_system_mod::system_type::atomic_number
```

Atomic number for every atom in the system.

Definition at line 87 of file prg_system_mod.F90.

9.14.2.2 atomname

```
character(3), dimension(:), allocatable prg_system_mod::system_type::atomname
```

Atom name (to distinguish atoms with same symbol)

Definition at line 188 of file prg_system_mod.F90.

9.14.2.3 coordinate

```
real(dp), dimension(:,:), allocatable prg_system_mod::system_type::coordinate
```

Coordinates of every atom in the system. Allocation:

```
coordinate(3,nats)
```

Definition at line 92 of file prg_system_mod.F90.

9.14.2.4 estr

```
type(estruct_type) prg_system_mod::system_type::estr
```

Electronic structure.

Definition at line 191 of file prg_system_mod.F90.

9.14.2.5 force

```
real(dp), dimension(:,:), allocatable prg_system_mod::system_type::force
```

Forces acting on every atom in the system. Allocation:

```
force(3,nats)
```

Definition at line 102 of file prg_system_mod.F90.

9.14.2.6 lattice_vector

```
real(dp), dimension(:,:), allocatable prg_system_mod::system_type::lattice_vector
```

Lattice vectors of the system. Use the prg_vectors_to_parameters and parameters_to_vector to transform from lattice vector to lattice parameters. Allocation:

```
lattice_vector(3,3)
```

```
v1 = lattice_vector(1,:)
```

```
v2 = lattice_vector(2,:)
```

```
v3 = lattice_vector(3,:)
```

Definition at line 124 of file prg_system_mod.F90.

9.14.2.7 mass

```
real(dp), dimension(:), allocatable prg_system_mod::system_type::mass
```

Mass of every atom in the system. These can be automatically loaded by using the structures of the ptable mod:

```
system%mass(i) = mass(mystem%atomic_number(i))
```

Allocation:

```
mass(nats)
```

Definition at line 114 of file prg_system_mod.F90.

9.14.2.8 nats

```
integer prg_system_mod::system_type::nats = 0
```

Number of atoms of the system.

Definition at line 76 of file prg_system_mod.F90.

9.14.2.9 net_charge

```
real(dp), dimension(:), allocatable prg_system_mod::system_type::net_charge
```

Charges of every atom in the system. Allocation:

```
net_charge(nats)
```

Definition at line 107 of file prg_system_mod.F90.

9.14.2.10 nsp

```
integer prg_system_mod::system_type::nsp
```

Number of different species. Number of species or number of differet antom types (symbols) in the system. This integer is alwas less or equal than the total number of atoms (nsp <= nats). This information can also be found in tbparams structure and the following equality holds:

```
system%nsp = tbparams%nsp
```

Definition at line 147 of file prg_system_mod.F90.

9.14.2.11 recip_vector

```
real(dp), dimension(:,:), allocatable prg_system_mod::system_type::recip_vector
```

Reciprocal vectors of the system. Allocation:

```
recip_vector(3,3)
```

```
v1 = recip_vector(1,:)
```

```
v2 = recip_vector(2,:)
```

```
v3 = recip_vector(3,:)
```

Definition at line 132 of file prg_system_mod.F90.

9.14.2.12 resindex

```
integer, dimension(:), allocatable prg_system_mod::system_type::resindex
```

Residue index.

Definition at line 182 of file prg_system_mod.F90.

9.14.2.13 resname

```
character(3), dimension(:), allocatable prg_system_mod::system_type::resname
```

Residue name.

Definition at line 185 of file prg_system_mod.F90.

9.14.2.14 spatnum

```
integer, dimension(:), allocatable prg_system_mod::system_type::spatnum
```

Species atomic number list. A list with the atomic numbers for every species Allocation:

```
spatnum(nsp)
```

Definition at line 170 of file prg_system_mod.F90.

9.14.2.15 spindex

```
integer, dimension(:), allocatable prg_system_mod::system_type::spindex
```

Species index. It gives the species index of a particulat atom. Allocation:

```
spindex(nats)
```

If we need the index of atom 30 then:

```
system%spindex(30)
```

Definition at line 155 of file prg_system_mod.F90.

9.14.2.16 splist

```
character(2), dimension(:), allocatable prg_system_mod::system_type::splist
```

Species symbol list. A list with the different species e.g. H, C, N, etc with the order corresponding to the appearence in systemsymbol. Allocation:

```
splist(nsp)
```

Definition at line 163 of file prg_system_mod.F90.

9.14.2.17 spmass

```
real(dp), dimension(:), allocatable prg_system_mod::system_type::spmass
```

Species mass list. A list with the atomic mass for every species Allocation:

```
spmass(nsp)
```

Definition at line 176 of file prg_system_mod.F90.

9.14.2.18 symbol

```
character(2), dimension(:), allocatable prg_system_mod::system_type::symbol
```

Chemical Symbols for every atom of the system. Symbol can be recovered using ptable module and calling the following routine:

```
system%symbol(i) = element_symbol(system%atomic_number(i))
```

Allocation:

```
symbol(nats)
```

Definition at line 84 of file prg_system_mod.F90.

9.14.2.19 userdef

```
real(dp), dimension(:), allocatable prg_system_mod::system_type::userdef
```

User define field.

Definition at line 179 of file prg_system_mod.F90.

9.14.2.20 velocity

```
real(dp), dimension(:,:), allocatable prg_system_mod::system_type::velocity
```

Velocities for every atom in the system. Allocation:

```
velocity(3,nats)
```

Definition at line 97 of file prg_system_mod.F90.

9.14.2.21 volk

```
real(dp) prg_system_mod::system_type::volk
```

Volume of the system (direct space).

Note

use prg_get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 140 of file prg_system_mod.F90.

9.14.2.22 volr

```
real(dp) prg_system_mod::system_type::volr
```

Volume of the system (direct space).

Note

use prg_get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 136 of file prg_system_mod.F90.

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

- /tmp/qmd-progress/src/prg_system_mod.F90

9.15 prg_timer_mod::timer_status_t Type Reference

Timer status type.

Private Attributes

- character(len=20) tname

    Timer name.
- integer tstart

    Start time.
- integer ttotal

    Current total time.
- integer tcount

    Current call count.
- integer minrank

    Rank with min value.
- integer maxrank

    Rank with max value.
- real(dp) tsum

    Sum time - total time in secs.
- real(dp) minvalue

    Minimum value over all ranks.
- real(dp) maxvalue

    Maximum value over all ranks.
- real(dp) tavg

    Average value over all ranks.
- real(dp) tstdev

    Stdev across all ranks.
- real(dp) tpercent

    Percent of time across all timers.

9.15.1 Detailed Description

Timer status type.

Definition at line 54 of file prg_timer_mod.F90.

9.15.2 Member Data Documentation

9.15.2.1 maxrank

```
integer prg_timer_mod::timer_status_t::maxrank  [private]
```

Rank with max value.

Definition at line 72 of file prg_timer_mod.F90.

9.15.2.2 maxvalue

```
real(dp) prg_timer_mod::timer_status_t::maxvalue  [private]
```

Maximum value over all ranks.

Definition at line 81 of file prg_timer_mod.F90.

9.15.2.3 minrank

```
integer prg_timer_mod::timer_status_t::minrank  [private]
```

Rank with min value.

Definition at line 69 of file prg_timer_mod.F90.

9.15.2.4 minvalue

```
real(dp) prg_timer_mod::timer_status_t::minvalue  [private]
```

Minimum value over all ranks.

Definition at line 78 of file prg_timer_mod.F90.

9.15.2.5 tavg

```
real(dp) prg_timer_mod::timer_status_t::tavg [private]
```

Average value over all ranks.

Definition at line 84 of file prg_timer_mod.F90.

9.15.2.6 tcount

```
integer prg_timer_mod::timer_status_t::tcount [private]
```

Current call count.

Definition at line 66 of file prg_timer_mod.F90.

9.15.2.7 tname

```
character(len=20) prg_timer_mod::timer_status_t::tname [private]
```

Timer name.

Definition at line 57 of file prg_timer_mod.F90.

9.15.2.8 tpercent

```
real(dp) prg_timer_mod::timer_status_t::tpercent [private]
```

Percent of time across all timers.

Definition at line 90 of file prg_timer_mod.F90.

9.15.2.9 tstart

```
integer prg_timer_mod::timer_status_t::tstart [private]
```

Start time.

Definition at line 60 of file prg_timer_mod.F90.

9.15.2.10 tstdev

```
real(dp) prg_timer_mod::timer_status_t::tstdev  [private]
```

Stdev across all ranks.

Definition at line 87 of file prg_timer_mod.F90.

9.15.2.11 tsum

```
real(dp) prg_timer_mod::timer_status_t::tsum  [private]
```

Sum time - total time in secs.

Definition at line 75 of file prg_timer_mod.F90.

9.15.2.12 ttotal

```
integer prg_timer_mod::timer_status_t::ttotal  [private]
```

Current total time.

Definition at line 63 of file prg_timer_mod.F90.

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

- /tmp/qmd-progress/src/prg_timer_mod.F90

9.16 prg_extras_mod::to_string Interface Reference

Private Member Functions

- character(len=:) function, allocatable to_string_integer (i)
  Convert integer to string.
- character(len=:) function, allocatable to_string_long_long (i)
  Convert integer to string.
- character(len=:) function, allocatable to_string_double (x)
  Convert double to string.

9.16.1 Detailed Description

Definition at line 25 of file prg_extras_mod.F90.

9.16.2 Member Function/Subroutine Documentation

9.16.2.1 to_string_double()

```
character(len=:)  function, allocatable prg_extras_mod::to_string::to_string_double (
             double precision, intent(in) x )  [private]
```

Convert double to string.

Parameters

x	The double

Returns

The string

Definition at line 81 of file prg_extras_mod.F90.

9.16.2.2 to_string_integer()

```
character(len=:)  function, allocatable prg_extras_mod::to_string::to_string_integer (
             integer, intent(in) i )  [private]
```

Convert integer to string.

Parameters

i	The integer

Returns

The string

Definition at line 47 of file prg_extras_mod.F90.

9.16.2.3 to_string_long_long()

```
character(len=:)  function, allocatable prg_extras_mod::to_string::to_string_long_long (
             integer(kind=c_long_long), intent(in) i )  [private]
```

Convert integer to string.

Parameters

i	The integer

Returns

The string

Definition at line 63 of file prg_extras_mod.F90.

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

- /tmp/qmd-progress/src/prg_extras_mod.F90

9.17 prg_xlbo_mod::xlbo_type Type Reference

General xlbo solver type.

Public Attributes

- character(20) jobname
- integer verbose
- integer maxscfiter

  Max SCF iterations at every XLBO MD step.
- integer maxscfinititer

  Max SCF iterations for the first minit steps.
- real(dp) threshold
- integer minit

  Use SCF the first M_prg_init MD steps.
- real(dp) cc

  Scaled prg_delta Kernel.

9.17.1 Detailed Description

General xlbo solver type.

Definition at line 33 of file prg_xlbo_mod.F90.

9.17.2 Member Data Documentation

9.17.2.1 cc

```
real(dp) prg_xlbo_mod::xlbo_type::cc
```

Scaled prg_delta Kernel.

Definition at line 51 of file prg_xlbo_mod.F90.

9.17.2.2 jobname

```
character(20) prg_xlbo_mod::xlbo_type::jobname
```

Definition at line 35 of file prg_xlbo_mod.F90.

9.17.2.3 maxscfinititer

```
integer prg_xlbo_mod::xlbo_type::maxscfinititer
```

Max SCF iterations for the first minit steps.

Definition at line 43 of file prg_xlbo_mod.F90.

9.17.2.4 maxscfiter

```
integer prg_xlbo_mod::xlbo_type::maxscfiter
```

Max SCF iterations at every XLBO MD step.

Definition at line 40 of file prg_xlbo_mod.F90.

9.17.2.5 minit

```
integer prg_xlbo_mod::xlbo_type::minit
```

Use SCF the first M_prg_init MD steps.

Definition at line 48 of file prg_xlbo_mod.F90.

9.17.2.6 threshold

```
real(dp) prg_xlbo_mod::xlbo_type::threshold
```

Definition at line 45 of file prg_xlbo_mod.F90.

9.17.2.7 verbose

```
integer prg_xlbo_mod::xlbo_type::verbose
```

Definition at line 37 of file prg_xlbo_mod.F90.

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

- /tmp/qmd-progress/src/prg_xlbo_mod.F90

9.18 prg_xlkernel_mod::xlk_type Type Reference

Public Attributes

- character(20) kerneltype

  Kernel type.
- integer verbose

  Verbosity level.
- integer nrank
- real(dp) scalecoeff

  Coefficient for mixing.

9.18.1 Detailed Description

Definition at line 18 of file prg_xlkernel_mod.F90.

9.18.2 Member Data Documentation

9.18.2.1 kerneltype

```
character(20) prg_xlkernel_mod::xlk_type::kerneltype
```

Kernel type.

Definition at line 21 of file prg_xlkernel_mod.F90.

9.18.2.2 nrank

```
integer prg_xlkernel_mod::xlk_type::nrank
```

Definition at line 24 of file prg_xlkernel_mod.F90.

9.18.2.3 scalecoeff

```
real(dp) prg_xlkernel_mod::xlk_type::scalecoeff
```

Coefficient for mixing.

Definition at line 27 of file prg_xlkernel_mod.F90.

9.18.2.4 verbose

```
integer prg_xlkernel_mod::xlk_type::verbose
```

Verbosity level.

Definition at line 24 of file prg_xlkernel_mod.F90.

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

- /tmp/qmd-progress/src/prg_xlkernel_mod.F90

Chapter 10

File Documentation

10.1 /tmp/qmd-progress/src/prg_charges_mod.F90 File Reference

Modules

- module prg_charges_mod

    A module to compute the Mulliken charges of a chemical system.

Functions/Subroutines

- subroutine, public prg_charges_mod::prg_get_charges (rho_bml, over_bml, hindex, charges, numel, spindex, mdimin, threshold)

    Constructs the charges from the density matrix.
- subroutine, public prg_charges_mod::prg_get_hscf (ham0_bml, over_bml, ham_bml, spindex, hindex, hubbardu, charges, coulomb_pot_r, coulomb_pot_k, mdimin, threshold)

    Constructs the SCF Hamiltonian given H0, HubbardU and charges. This routine does: $H = \sum_i U_i q_i + V_i;$, where U is the Hubbard parameter for every atom i. V is the coulombic potential for every atom i.

Variables

- integer, parameter prg_charges_mod::dp = kind(1.0d0)

10.2 /tmp/qmd-progress/src/prg_chebyshev_mod.F90 File Reference

Data Types

- type prg_chebyshev_mod::chebdata_type

    General Cheb solver type.

Modules

- module prg_chebyshev_mod

    Module to obtain the density matrix by applying a Chebyshev polynomial expansion.

Functions/Subroutines

- subroutine, public prg_chebyshev_mod::prg_parse_cheb (chebdata, filename)

  Chebyshev parser. This module is used to parse all the input variables for the cheb electronic structure solver. Adding a new input keyword to the parser:
- subroutine, public prg_chebyshev_mod::prg_build_density_cheb (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, jon, verbose)

  Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion.
- subroutine, public prg_chebyshev_mod::prg_build_density_cheb_fermi (ham_bml, rho_bml, athr, threshold, ncoeffs, kbt, ef, bndfil, getef, fermitol, jon, npts, trkfunc, verbose)

  Builds the density matrix from H_0 for a Fermi function approximated with a Chebyshev polynomial expansion. In this case the self-consistent recursion is applied to converge to the correct number of electrons and obtain the Fermi level.
- real(dp) function prg_chebyshev_mod::jackson (ncoeffs, i, jon)

  Evaluates the Jackson Kernel Coefficients.
- subroutine prg_chebyshev_mod::prg_get_chebcoeffs (npts, kbt, ef, ncoeffs, coeffs, emin, emax)

  Gets the coefficients of the Chebyshev expansion.
- subroutine prg_chebyshev_mod::prg_get_chebcoeffs_fermi_bs (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)

  Gets the coefficients of the Chebyshev expansion with Ef computation.
- subroutine prg_chebyshev_mod::prg_get_chebcoeffs_fermi_nt (npts, kbt, ef, tracesT, ncoeffs, coeffs, emin, emax, bndfil, norb, tol, jon, verbose)

  Gets the coefficients of the Chebyshev expansion with Ef computation.
- real(dp) function prg_chebyshev_mod::tr (r, x)

  Chebyshev polynomial obtained by recursion.
- real(dp) function prg_chebyshev_mod::fermi (e, ef, kbt)

  Gives the Fermi distribution value for energy e.
- real(dp) function prg_chebyshev_mod::absmaxderivative (func, de)

  Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter prg_chebyshev_mod::dp = kind(1.0d0)
- real(dp), parameter prg_chebyshev_mod::pi = 3.14159265358979323846264338327950_dp

10.3 /tmp/qmd-progress/src/prg_densitymatrix_mod.F90 File Reference

Modules

- module prg_densitymatrix_mod

  Module to obtain the density matrix by diagonalizing an orthogonalized Hamiltonian.

Functions/Subroutines

- subroutine, public prg_densitymatrix_mod::prg_build_density_t0 (ham_bml, rho_bml, threshold, bndfil, eigenvalues_out)

  Builds the density matrix from H_0 for zero electronic temperature. $\rho = C\Theta(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. $\Theta()$ is the Heaviside function.
- subroutine, public prg_densitymatrix_mod::prg_build_density_t (ham_bml, rho_bml, threshold, bndfil, kbt, ef, eigenvalues_out)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public prg_densitymatrix_mod::prg_build_density_t_fulldata (ham_bml, rho_bml, threshold, bndfil, kbt, ef, eigenvalues_out, evects_bml, fvals)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function.
- subroutine, public prg_densitymatrix_mod::prg_build_density_t_fermi (ham_bml, rho_bml, threshold, kbt, ef, verbose)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \epsilon)C^\dagger$ Where, C is the matrix eigenvector and ϵ is the matrix eigenvalue. f is the Fermi function. In this routine the Fermi level is passed as an argument.
- subroutine, public prg_densitymatrix_mod::prg_build_atomic_density (rhoat_bml, numel, hindex, spindex, norb, bml_type)

  Builds the atomic density matrix. $\rho_{ii} = mathcal{Z}_{ii}$ Where, $mathcal{Z}_{ii}$ is the number of electrons for orbital i.
- subroutine, public prg_densitymatrix_mod::prg_get_flevel (eigenvalues, kbt, bndfil, tol, Ef, err)

  Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.
- subroutine, public prg_densitymatrix_mod::prg_get_flevel_nt (eigenvalues, kbt, bndfil, tol, ef, err, verbose)

  Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Newton-Raphson method over the function: $g(\mu) = \sum_k 2f(\epsilon_k - \mu) - N = 0$ Where $f(\epsilon_k - \mu) = \frac{1}{1+\exp{(\epsilon_k - \mu)/(k_bT)}}$.
- subroutine, public prg_densitymatrix_mod::prg_get_eigenvalues (ham_bml, eigenvalues, verbose)

  Gets the eigenvalues of the Orthogonalized Hamiltonian.
- subroutine, public prg_densitymatrix_mod::prg_check_idempotency (mat_bml, threshold, idempotency)

  To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.
- real(dp) function prg_densitymatrix_mod::fermi (e, ef, kbt)

  Gives the Fermi distribution value for energy e.

Variables

- integer, parameter prg_densitymatrix_mod::dp = kind(1.0d0)

10.4 /tmp/qmd-progress/src/prg_dos_mod.F90 File Reference

Modules

- module prg_dos_mod

  A module to compute the Density of state (DOS) and IDOS.

Functions/Subroutines

- subroutine, public prg_dos_mod::prg_write_tdos (eigenvals, gamma, npts, emin, emax, filename)

  Writes the total DOS into a file. $DOS(\epsilon) = \sum_k L(\epsilon - \epsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\epsilon) = Nstates.$
- real(dp) function prg_dos_mod::lorentz (energy, eigenvals, loads, Gamma)

  Lorentzian Function.

Variables

- integer, parameter prg_dos_mod::dp = kind(1.0d0)

10.5 /tmp/qmd-progress/src/prg_doxy_mod.F90 File Reference

10.6 /tmp/qmd-progress/src/prg_ewald_mod.F90 File Reference

Modules

- module prg_ewald_mod

Functions/Subroutines

- subroutine, public prg_ewald_mod::ewald_real_space_single_latte (COULOMBV, I, RXYZ, Box, Nr_elem, DELTAQ, J, U, Element_Pointer, Nr_atoms, COULACC, HDIM, Max_Nr_Neigh)

  Find Coulomb potential on site I from single charge at site J.
- subroutine, public prg_ewald_mod::ewald_real_space_single (COULOMBV, FCOUL, I, RX, RY, RZ, LBox, DELTAQ, J, U, Element_Type, Nr_atoms, COULACC, TIMERATIO, HDIM, Max_Nr_Neigh)
- subroutine, public prg_ewald_mod::ewald_real_space_matrix_latte (E, RXYZ, Box, U, Element_Pointer, Nr↩_atoms, COULACC, nebcoul, totnebcoul, HDIM, Max_Nr_Neigh, Nr_Elem)
- subroutine, public prg_ewald_mod::ewald_real_space_latte (COULOMBV, I, RXYZ, Box, DELTAQ, U, Element_Pointer, Nr_atoms, COULACC, nebcoul, totnebcoul, HDIM, Max_Nr_Neigh, Nr_Elem)
- subroutine, public prg_ewald_mod::ewald_real_space_test (COULOMBV, I, RX, RY, RZ, LBox, DELTAQ, U, Element_Type, Nr_atoms, COULACC, nnRx, nnRy, nnRz, nrnnlist, nnType, Max_Nr_Neigh)
- subroutine, public prg_ewald_mod::ewald_real_space (COULOMBV, FCOUL, I, RX, RY, RZ, LBox, DELTAQ, U, Element_Type, Nr_atoms, COULACC, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, HDIM, Max_↩Nr_Neigh)
- subroutine, public prg_ewald_mod::ewald_k_space_latte (COULOMBV, RXYZ, Box, DELTAQ, Nr_atoms, COULACC, Max_Nr_Neigh)
- subroutine, public prg_ewald_mod::ewald_k_space_matrix_latte (E, RXYZ, Box, Nr_atoms, COULACC, Max_Nr_Neigh, nebcoul, totnebcoul)
- subroutine, public prg_ewald_mod::ewald_k_space_latte_single (COULOMBV, J, RXYZ, Box, DELTAQ, Nr↩_atoms, COULACC)
- subroutine, public prg_ewald_mod::ewald_k_space_test (COULOMBV, RX, RY, RZ, LBox, DELTAQ, Nr_↩atoms, COULACC, Max_Nr_Neigh)

Variables

- integer, parameter prg_ewald_mod::dp = kind(1.0d0)

10.7 /tmp/qmd-progress/src/prg_extras_mod.F90 File Reference

Data Types

- interface prg_extras_mod::prg_memory_consumption
- interface prg_extras_mod::to_string

Modules

- module prg_extras_mod

  Extra routines.

Functions/Subroutines

- character(len=:) function, allocatable prg_extras_mod::to_string_integer (i)

  Convert integer to string.
- character(len=:) function, allocatable prg_extras_mod::to_string_long_long (i)

  Convert integer to string.
- character(len=:) function, allocatable prg_extras_mod::to_string_double (x)

  Convert double to string.
- subroutine, public prg_extras_mod::prg_print_matrix (matname, amat, i1, i2, j1, j2)

  To write a dense matrix to screen.
- real(dp) function, public prg_extras_mod::mls ()

  To get the actual time in milliseconds.
- subroutine, public prg_extras_mod::prg_delta (x, s, nn, dta)

  Delta function $||X^\wedge tSX - I||$.
- subroutine, public prg_extras_mod::prg_get_mem (procname, tag)

  Get proc memory.
- subroutine prg_extras_mod::prg_twonorm (a, nn, norm2)

  Gets the norm2 of a square matrix.
- real(dp) function, public prg_extras_mod::prg_norm2 (a)

  Gets the norm2 of a vector.

Variables

- integer, parameter prg_extras_mod::dp = kind(1.0d0)

10.8 /tmp/qmd-progress/src/prg_genz_mod.F90 File Reference

Data Types

- type prg_genz_mod::genzspinp

  Input for the genz driver. This type controlls all the variables that are needed by genz.

Modules

- module prg_genz_mod

  To produce a matrix Z *which is needed to orthogonalize* H.

Functions/Subroutines

- subroutine, public prg_genz_mod::prg_parse_zsp (input, filename)

  The parser for genz solver.
- subroutine, public prg_genz_mod::prg_init_zspmat (igenz, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type, bml_element_type)

  Initiates the matrices for the Xl integration of Z.
- subroutine, public prg_genz_mod::prg_buildzdiag (smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)

  Usual subroutine involving diagonalization. $Z = U\sqrt{s}U^\dagger$, where U = eigenvectors and s = eigenvalues. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.
- subroutine, public prg_genz_mod::prg_buildzsparse (smat_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, nfirst, nrefi, nreff, thresholdi, thresholdf, integration, verbose)

  Inverse factorization using Niklasson's algorithm.
- subroutine, public prg_genz_mod::prg_genz_sp_initialz0 (smat_bml, zmat_bml, norb, mdim, bml_type_↩ f, threshold)

  Initial estimation of Z.
- subroutine, public prg_genz_mod::prg_genz_sp_initial_zmat (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)

  Initial estimation of Z.
- subroutine prg_genz_mod::prg_genz_sp_int (zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)

  Inverse factorization using Niklasson's algorithm.
- subroutine, public prg_genz_mod::prg_genz_sp_ref (smat_bml, zmat_bml, nref, norb, bml_type, threshold)

  Iterative refinement.

Variables

- integer, parameter prg_genz_mod::dp = kind(1.0d0)

10.9 /tmp/qmd-progress/src/prg_graph_mod.F90 File Reference

Data Types

- type prg_graph_mod::subgraph_t

  Subgraph type.
- type prg_graph_mod::graph_partitioning_t

  Trace per iteration.

Modules

- module prg_graph_mod

  The graph module.

Functions/Subroutines

- subroutine, public prg_graph_mod::prg_initsubgraph (sg, pnum, hsize)

  Initialize subgraph.
- subroutine, public prg_graph_mod::prg_destroysubgraph (sg)

  Destroy subgraph.
- subroutine, public prg_graph_mod::prg_initgraphpartitioning (gp, pname, np, nnodes, nnodes2)

  Initialize graph partitioning.
- subroutine, public prg_graph_mod::prg_destroygraphpartitioning (gp)

  Destroy graph partitioning.
- subroutine, public prg_graph_mod::prg_printgraphpartitioning (gp)

  Print graph partitioning structure data.
- subroutine, public prg_graph_mod::prg_equalpartition (gp, nodesPerPart, nnodes)

  Create equal graph partitions, based on number of rows/orbitals.
- subroutine, public prg_graph_mod::prg_equalgrouppartition (gp, hindex, ngroup, nodesPerPart, nnodes)

  Create equal group graph partitions, based on number of atoms/groups.
- subroutine, public prg_graph_mod::prg_filepartition (gp, partFile)

  Read graph partitions from a file, based on number of rows/orbitals.
- subroutine prg_graph_mod::prg_readpart (gp, partFile)

  Read parts (core) from part file.
- subroutine, public prg_graph_mod::prg_fnormgraph (gp)

  Accumulate trace norm across all subgraphs.

Variables

- integer, parameter prg_graph_mod::dp = kind(1.0d0)

10.10 /tmp/qmd-progress/src/prg_graphsolver_mod.F90 File Reference

Modules

- module prg_graphsolver_mod

  Module for graph-based solvers.

Functions/Subroutines

- subroutine, public prg_graphsolver_mod::prg_build_densitygp_t0 (ham_bml, g_bml, rho_bml, threshold, bndfil, Ef, nparts, verbose)

  Builds the density matrix from H_0 using a graph-based approach.
- subroutine, public prg_graphsolver_mod::prg_build_zmatgp (over_bml, g_bml, zmat_bml, threshold, nparts, verbose)

  Builds the inverse overlap factor matrix from S using a graph-based approach.

Variables

- integer, parameter prg_graphsolver_mod::dp = kind(1.0d0)

10.11 /tmp/qmd-progress/src/prg_graphsp2parser_mod.F90 File Reference

Data Types

- type prg_graphsp2parser_mod::gsp2data_type
  General SP2 solver type.

Modules

- module prg_graphsp2parser_mod
  Graph partitioning SP2 parser.

Functions/Subroutines

- subroutine, public prg_graphsp2parser_mod::prg_parse_gsp2 (gsp2data, filename)
  The parser for SP2 solver.

Variables

- integer, parameter prg_graphsp2parser_mod::dp = kind(1.0d0)

10.12 /tmp/qmd-progress/src/prg_homolumo_mod.F90 File Reference

Modules

- module prg_homolumo_mod
  The homolumo module.

Functions/Subroutines

- subroutine, public prg_homolumo_mod::prg_homolumogap (vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)
- subroutine, public prg_homolumo_mod::prg_sp2sequence (pp, imax, mineval, maxeval, ehomo, elumo, errlimit, verbose)

Variables

- integer, parameter prg_homolumo_mod::dp = kind(1.0d0)

10.13 /tmp/qmd-progress/src/prg_implicit_fermi_mod.F90 File Reference

Modules

- module prg_implicit_fermi_mod

Functions/Subroutines

- subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_save_inverse (Inv_bml, h_bml, p_bml, nsteps, nocc, mu, beta, occErrLimit, threshold, tol, SCF_IT, occiter, totns)

  Recursive Implicit Fermi Dirac for finite temperature.
- subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi (h_bml, p_bml, nsteps, k, nocc, mu, beta, method, osteps, occErrLimit, threshold, tol)

  Recursive Implicit Fermi Dirac for finite temperature.
- subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_zero (h_bml, p_bml, nsteps, mu, method, threshold, tol)

  Recursive Implicit Fermi Dirac for zero temperature.
- subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_first_order_response (H0_bml, H1_bml, P0_↩ bml, P1_bml, Inv_bml, nsteps, mu0, beta, nocc, threshold)

  Calculate first order density matrix response to perturbations using Implicit Fermi Dirac.
- subroutine, public prg_implicit_fermi_mod::prg_implicit_fermi_response (H0_bml, H1_bml, H2_bml, H3_bml, P0_bml, P1_bml, P2_bml, P3_bml, nsteps, mu0, mu, beta, nocc, occ_tol, lin_tol, order, threshold)

  Calculate density matrix response to perturbations using Implicit Fermi Dirac.
- subroutine, public prg_implicit_fermi_mod::prg_finite_diff (H0_bml, H_list, mu0, mu_list, beta, order, lambda, h, threshold)

  Calculate density matrix response from perturbations in the Hamiltonian.
- subroutine prg_implicit_fermi_mod::prg_setup_linsys (p_bml, A_bml, b_bml, p2_bml, y_bml, aux_bml, aux1_bml, k, threshold)

  Set up linear system for Implicit Fermi Dirac.
- subroutine prg_implicit_fermi_mod::prg_newtonschulz (a_bml, ai_bml, r_bml, tmp_bml, d_bml, I_bml, tol, threshold, num_iter)

  Find the inverse of the matrix A with Newton-Schulz iteration.
- subroutine prg_implicit_fermi_mod::prg_pcg (A_bml, p_bml, p2_bml, d_bml, wtmp_bml, cg_tol, threshold)

  Solve the system $AX = B$ with conjugate gradient.
- subroutine prg_implicit_fermi_mod::prg_conjgrad (A_bml, p_bml, p2_bml, tmp_bml, d_bml, w_bml, cg_tol, threshold)

  Solve the system $AX = B$ with conjugate gradient.
- subroutine prg_implicit_fermi_mod::prg_get_density_matrix (ham_bml, p_bml, beta, mu, threshold)

  Calculate the density matrix with diagonalization.
- subroutine, public prg_implicit_fermi_mod::prg_test_density_matrix (ham_bml, p_bml, beta, mu, nocc, osteps, occErrLimit, threshold)

  Calculate the density matrix with diagonalization and converge chemical.
- real(dp) function prg_implicit_fermi_mod::fermi (e, mu, beta)

  Gives the Fermi distribution value for energy e.

Variables

- integer, parameter prg_implicit_fermi_mod::dp = kind(1.0d0)

10.14 /tmp/qmd-progress/src/prg_initmatrices_mod.F90 File Reference

Modules

- module prg_initmatrices_mod

  Initialization module.

Functions/Subroutines

- subroutine, public prg_initmatrices_mod::prg_init_hsmat (ham_bml, over_bml, bml_type, mdim, norb)

  Initialize Hamiltonian and Overlap Matrix.
- subroutine, public prg_initmatrices_mod::prg_init_pzmat (rho_bml, zmat_bml, bml_type, mdim, norb)

  Initialize Density matrix and Inverse square root Overlap.
- subroutine, public prg_initmatrices_mod::prg_init_ortho (orthoh_bml, orthop_bml, bml_type, mdim, norb)

  Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Variables

- integer, parameter prg_initmatrices_mod::dp = kind(1.0d0)

10.15 /tmp/qmd-progress/src/prg_kernelparser_mod.F90 File Reference

Modules

- module prg_kernelparser_mod

  Some general parsing functions.

Functions/Subroutines

- subroutine, public prg_kernelparser_mod::prg_parsing_kernel (keyvector_char, valvector_char, keyvector_↩ int, valvector_int, keyvector_re, valvector_re, keyvector_log, valvector_log, filename, startstop)

  The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.
- subroutine prg_kernelparser_mod::prg_check_valid (invalidc)

  Check for valid keywords (checks for an = sign)

Variables

- integer, parameter prg_kernelparser_mod::dp = kind(1.0d0)

10.16 /tmp/qmd-progress/src/prg_modelham_mod.F90 File Reference

Data Types

- type prg_modelham_mod::mham_type

  General ModelHam type.

Modules

- module prg_modelham_mod

  The prg_hamiltonian module.

Functions/Subroutines

- subroutine, public prg_modelham_mod::prg_parse_mham (mham, filename)

  Model Ham parse.
- subroutine, public prg_modelham_mod::prg_twolevel_model (ea, eb, dab, daiaj, dbibj, dec, rcoeff, reshuffle, seed, h_bml, verbose)

  Construct a two-level model Hamiltonian.

Variables

- integer, parameter prg_modelham_mod::dp = kind(1.0d0)

10.17 /tmp/qmd-progress/src/prg_nonortho_mod.F90 File Reference

Modules

- module prg_nonortho_mod

  Module to prg_orthogonalize and prg_deorthogonalize any operator.

Functions/Subroutines

- subroutine, public prg_nonortho_mod::prg_orthogonalize (A_bml, zmat_bml, orthoA_bml, threshold, bml_↩ type, verbose)

  This routine performs: $A_{ortho} = Z^{\dagger}AZ$.
- subroutine, public prg_nonortho_mod::prg_deorthogonalize (orthoA_bml, zmat_bml, a_bml, threshold, bml↩ _type, verbose)

  This routine performs: $A = ZA_{ortho}Z^{\dagger}$.

Variables

- integer, parameter prg_nonortho_mod::dp = kind(1.0d0)

10.18 /tmp/qmd-progress/src/prg_normalize_mod.F90 File Reference

Modules

- module prg_normalize_mod

  The prg_normalize module.

Functions/Subroutines

- subroutine, public prg_normalize_mod::prg_normalize (h_bml)
  Normalize a Hamiltonian matrix prior to running the SP2 algorithm.
- subroutine, public prg_normalize_mod::prg_normalize_fermi (h_bml, h1, hN, mu)
  Normalize a Hamiltonian matrix prior to running the truncated SP2 algorithm.
- subroutine, public prg_normalize_mod::prg_normalize_implicit_fermi (h_bml, cnst, mu)
  Normalize a Hamiltonian matrix prior to running the implicit fermi dirac algorithm.
- subroutine, public prg_normalize_mod::prg_gershgorinreduction (gp)
  Determine gershgorin bounds across all parts, local and distributed.
- subroutine, public prg_normalize_mod::prg_normalize_cheb (h_bml, mu, emin, emax, alpha, scaledmu)
  Normalize a Hamiltonian matrix prior to running the Chebyshev algorithm.

Variables

- integer, parameter prg_normalize_mod::dp = kind(1.0d0)

10.19 /tmp/qmd-progress/src/prg_openfiles_mod.F90 File Reference

Modules

- module prg_openfiles_mod
  Module to handle input output files for the PROGRESS lib.

Functions/Subroutines

- integer function, public prg_openfiles_mod::get_file_unit (io_max)
  Returns a unit number that is not in use.
- subroutine, public prg_openfiles_mod::prg_open_file (io, name)
  Opens a file to write.
- subroutine, public prg_openfiles_mod::prg_open_file_to_read (io, name)
  Opens a file to read.

10.20 /tmp/qmd-progress/src/prg_parallel_mod.F90 File Reference

Data Types

- type prg_parallel_mod::rankreducedata_t
  Data structure for rection over MPI ranks.

Modules

- module prg_parallel_mod
  The parallel module.

Functions/Subroutines

- integer function, public prg_parallel_mod::getnranks ()
- integer function, public prg_parallel_mod::getmyrank ()
- integer function, public prg_parallel_mod::printrank ()
- subroutine, public prg_parallel_mod::prg_initparallel ()
- subroutine, public prg_parallel_mod::prg_shutdownparallel ()
- integer function prg_parallel_mod::saverequest (irequest)
- subroutine, public prg_parallel_mod::prg_barrierparallel ()
- subroutine, public prg_parallel_mod::sendreceiveparallel (sendBuf, sendLen, dest, recvBuf, recvLen, source, nreceived)
- subroutine, public prg_parallel_mod::isendparallel (sendBuf, sendLen, dest)
- subroutine, public prg_parallel_mod::sendparallel (sendBuf, sendLen, dest)
- subroutine, public prg_parallel_mod::prg_iprg_recvparallel (recvBuf, recvLen, rind)
- subroutine, public prg_parallel_mod::prg_recvparallel (recvBuf, recvLen)
- subroutine, public prg_parallel_mod::sumintparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::sumrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::maxintparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::maxrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::minintparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::minrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::prg_minrealreduce (rvalue)
- subroutine, public prg_parallel_mod::prg_maxrealreduce (rvalue)
- subroutine, public prg_parallel_mod::prg_maxintreduce2 (value1, value2)
- subroutine, public prg_parallel_mod::prg_sumintreduce2 (value1, value2)
- subroutine, public prg_parallel_mod::prg_sumrealreduce (value1)
- subroutine, public prg_parallel_mod::prg_sumrealreduce2 (value1, value2)
- subroutine, public prg_parallel_mod::prg_sumrealreduce3 (value1, value2, value3)
- subroutine, public prg_parallel_mod::prg_sumrealreducen (valueVec, N)
- subroutine, public prg_parallel_mod::prg_sumintreducen (valueVec, N)
- subroutine, public prg_parallel_mod::minrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::maxrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public prg_parallel_mod::prg_bcastparallel (buf, blen, root)
- subroutine, public prg_parallel_mod::allgatherrealparallel (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public prg_parallel_mod::allgatherintparallel (sendBuf, sendLen, recvBuf, recvLen)
- subroutine, public prg_parallel_mod::allgathervrealparallel (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public prg_parallel_mod::allgathervintparallel (sendBuf, sendLen, recvBuf, recvLen, recvDispl)
- subroutine, public prg_parallel_mod::prg_allsumrealreduceparallel (buf, buflen)
- subroutine, public prg_parallel_mod::prg_allsumintreduceparallel (buf, buflen)
- subroutine, public prg_parallel_mod::prg_allgatherparallel (a)
- subroutine, public prg_parallel_mod::prg_wait ()

Variables

- integer, parameter prg_parallel_mod::dp = kind(1.0d0)
- integer prg_parallel_mod::myrank
- integer prg_parallel_mod::nranks
- integer prg_parallel_mod::ierr
- integer prg_parallel_mod::reqcount
- integer, dimension(:), allocatable prg_parallel_mod::requestlist
- integer, dimension(:), allocatable prg_parallel_mod::rused

10.21 /tmp/qmd-progress/src/prg_partition_mod.F90 File Reference

Modules

- module prg_partition_mod

  The partition module.

Functions/Subroutines

- subroutine, public prg_partition_mod::prg_metispartition (gp, ngroups, nnodes, xadj, adjncy, nparts, part, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Create graph partitions minizing number of cut edges.
- subroutine, public prg_partition_mod::prg_costpartition (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Compute cost of a partition.
- subroutine, public prg_partition_mod::update_prg_costpartition (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, node, new_part)

  Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.
- subroutine prg_partition_mod::prg_accept_prob (it, prg_delta, r)

  Compute acceptance probability for simulated annealing.
- subroutine prg_partition_mod::prg_costindex (cost, sumCubes, maxCH, smooth_maxCH, obj_fun)

  Choose objective function to work with.
- subroutine prg_partition_mod::prg_rand_node (gp, node, seed)

  Pick a random node.
- subroutine, public prg_partition_mod::prg_simannealing (gp, xadj, adjncy, partNumber, core_count, CH_↩ count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

  Graph partitioning based on Simulated Annealing.
- subroutine, public prg_partition_mod::prg_kernlin (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, nconverg, seed)

  Graph partitioning based on inspired by Kernighan-Lin Review METiS manual for description of k-way implementation of KL Pick a core together with its halos Place free vertices on a priority queue with (key, value) =(prg_delta, best_↩ part),with prg_delta = change in obj_value Dequeue and allow hill climbing.
- subroutine, public prg_partition_mod::prg_update_gp (gp, partNumber, core_count)
- subroutine prg_partition_mod::prg_rand_shuffle (array, seed)

  Randomly shuffle array.
- subroutine, public prg_partition_mod::prg_check_arrays (gp, core_count, CH_count, Halo_count)

  Error checking Checking that core_count, CH_count, Halo_count match.
- subroutine, public prg_partition_mod::prg_kernlin_queue (gp, xadj, adjncy, partNumber, core_count, CH_↩ count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Greedy algorithm. At each step it chooses the (vertex, new_part) pair with highest gain Currently implementation is very slow.
- subroutine prg_partition_mod::prg_find_best_move (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, best_node, best_part)

  For kerlin_queue to find (vertex, new_part) pair with highest gain.
- subroutine, public prg_partition_mod::prg_kernlin2 (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
- subroutine prg_partition_mod::prg_get_largest_hedge_in_part (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)
- subroutine, public prg_partition_mod::prg_simannealing_old (gp, xadj, adjncy, partNumber, core_count, C↩ H_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

Variables

- integer, parameter prg_partition_mod::dp = kind(1.0d0)
- integer, parameter prg_partition_mod::metis_index_kind = METIS_INDEX_KIND

  From /usr/include/metis.h.
- integer, parameter prg_partition_mod::metis_real_kind = kind(METIS_REAL_KIND)

  From /usr/include/metis.h.

10.22 /tmp/qmd-progress/src/prg_progress_mod.F90 File Reference

Modules

- module prg_progress_mod

  The progress module.

Functions/Subroutines

- subroutine, public prg_progress_mod::prg_progress_init ()

  Initialize progress.
- subroutine, public prg_progress_mod::prg_progress_shutdown ()

  Shutdown progress.

Variables

- integer, parameter prg_progress_mod::dp = kind(1.0d0)

10.23 /tmp/qmd-progress/src/prg_ptable_mod.F90 File Reference

Modules

- module prg_ptable_mod

  Periodic table of elements.

Functions/Subroutines

- integer function, public prg_ptable_mod::element_atomic_number (symbol)
- integer function prg_ptable_mod::element_atomic_number_upper (symbol)

Variables

- integer, parameter prg_ptable_mod::nz = 103
- integer, parameter, private prg_ptable_mod::dp = kind(1.0d0)
- character(2), dimension(nz), parameter prg_ptable_mod::element_symbol = [character(2) :: "H" , "He" , "Li" , "Be" , "B" , "C" , "N" , "O" , "F" , "Ne" , "Na" , "Mg" , "Al" , "Si" , "P" , "S" , "Cl" , "Ar" , "K" , "Ca" , "Sc" , "Ti" , "V" , "Cr" , "Mn" , "Fe" , "Co" , "Ni" , "Cu" , "Zn" , "Ga" , "Ge" , "As" , "Se" , "Br" , "Kr" , "Rb" , "Sr" , "Y" , "Zr" , "Nb" , "Mo" , "Tc" , "Ru" , "Rh" , "Pd" , "Ag" , "Cd" , "In" , "Sn" , "Sb" , "Te" , "I" , "Xe" , "Cs" , "Ba" , "La" , "Ce" , "Pr" , "Nd" , "Pm" , "Sm" , "Eu" , "Gd" , "Tb" , "Dy" , "Ho" , "Er" , "Tm" , "Yb" , "Lu" , "Hf" , "Ta" , "W" , "Re" , "Os" , "Ir" , "Pt" , "Au" , "Hg" , "Tl" , "Pb" , "Bi" , "Po" , "At" , "Rn" , "Fr" , "Ra" , "Ac" , "Th" , "Pa" , "U" , "Np" , "Pu" , "Am" , "Cm" , "Bk" , "Cf" , "Es" , "Fm" , "Md" , "No" , "Lr"]

  Element symbol.
- character(2), dimension(nz), parameter prg_ptable_mod::element_symbol_upper = [character(2) :: "H" , "HE" , "LI" , "BE" , "B" , "C" , "N" , "O" , "F" , "NE" , "NA" , "MG" , "AL" , "SI" , "P" , "S" , "CL" , "AR" , "K" , "CA" , "SC" , "TI" , "V" , "CR" , "MN" , "FE" , "CO" , "NI" , "CU" , "ZN" , "GA" , "GE" , "AS" , "SE" , "BR" , "KR" , "RB" , "SR" , "Y" , "ZR" , "NB" , "MO" , "TC" , "RU" , "RH" , "PD" , "AG" , "CD" , "IN" , "SN" , "SB" , "TE" , "I" , "XE" , "CS" , "BA" , "LA" , "CE" , "PR" , "ND" , "PM" , "SM" , "EU" , "GD" , "TB" , "DY" , "HO" , "ER" , "TM" , "YB" , "LU" , "HF" , "TA" , "W" , "RE" , "OS" , "IR" , "PT" , "AU" , "HG" , "TL" , "PB" , "BI" , "PO" , "AT" , "RN" , "FR" , "RA" , "AC" , "TH" , "PA" , "U" , "NP" , "PU" , "AM" , "CM" , "BK" , "CF" , "ES" , "FM" , "MD" , "NO" , "LR"]

  Element symbol upper.
- character(20), dimension(nz), parameter prg_ptable_mod::element_name = [character(20) :: "Hydrogen" , "Helium" , "Lithium" , "Beryllium" , "Boron" , "Carbon" , "Nitrogen" , "Oxygen" , "Fluorine" , "Neon" , "Sodium" , "Magnesium" , "Aluminium" , "Silicon" , "Phosphorus" , "Sulfur" , "Chlorine" , "Argon" , "Potassium" , "Calcium" , "Scandium" , "Titanium" , "Vanadium" , "Chromium" , "Manganese" , "Iron" , "Cobalt" , "Nickel" , "Copper" , "Zinc" , "Gallium" , "Germanium" , "Arsenic" , "Selenium" , "Bromine" , "Krypton" , "Rubidium" , "Strontium" , "Yttrium" , "Zirconium" , "Niobium" , "Molybdenum" , "Technetium" , "Ruthenium" , "Rhodium" , "Palladium" , "Silver" , "Cadmium" , "Indium" , "Tin" , "Antimony" , "Tellurium" , "Iodine" , "Xenon" , "Caesium" , "Barium" , "Lanthanum" , "Cerium" , "Praseodymium" , "Neodymium" , "Promethium" , "Samarium" , "Europium" , "Gadolinium" , "Terbium" , "Dysprosium" , "Holmium" , "Erbium" , "Thulium" , "Ytterbium" , "Lutetium" , "Hafnium" , "Tantalum" , "Tungsten" , "Rhenium" , "Osmium" , "Iridium" , "Platinum" , "Gold" , "Mercury" , "Thallium" , "Lead" , "Bismuth" , "Polonium" , "Astatine" , "Radon" , "Francium" , "Radium" , "Actinium" , "Thorium" , "Protactinium" , "Uranium" , "Neptunium" , "Plutonium" , "Americium" , "Curium" , "Berkelium" , "Californium" , "Einsteinium" , "Fermium" , "Mendelevium" , "Nobelium" , "Lawrencium"]

  Element name.
- real(dp), dimension(nz), parameter prg_ptable_mod::element_mass = (/ 1.007825032 , 4.002603254 , 7.↩
01600455 , 9.0121822 , 11.0093054 , 12.0 , 14.003074005 , 15.99491462 , 18.99840322 , 19.992440175 , 22.989769281 , 23.9850417 , 26.98153863 , 27.976926532 , 30.97376163 , 31.972071 , 34.96885268 , 39.962383123 , 38.96370668 , 39.96259098 , 44.9559119 , 47.9479463 , 50.9439595 , 51.9405075 , 54.↩
9380451 , 55.9349375 , 58.933195 , 57.9353429 , 62.9295975 , 63.929142 , 68.925573 , 73.921177 , 74.↩
921596 , 79.916521 , 78.918337 , 83.911507 , 84.911789 , 87.905612 , 88.905848 , 89.904704 , 92.906378 , 97.905408 , 97.907216 , 101.904349 , 102.905504 , 105.903486 , 106.905097 , 113.903358 , 114.903878 , 119.902194 , 120.903815 , 129.906224 , 126.904473 , 131.904153 , 132.905451 , 137.905247 , 138.↩
906353 , 139.905438 , 140.907652 , 141.907723 , 144.912749 , 151.919732 , 152.92123 , 157.924103 , 158.925346 , 163.929174 , 164.930322 , 165.930293 , 168.934213 , 173.938862 , 174.940771 , 179.94655 , 180.947995 , 183.950931 , 186.955753 , 191.96148 , 192.962926 , 194.964791 , 196.966568 , 201.970643 , 204.974427 , 207.976652 , 208.980398 , 208.98243 , 209.987148 , 222.017577 , 223.019735 , 226.025409 , 227.027752 , 232.038055 , 231.035884 , 238.050788 , 237.048173 , 244.064204 , 243.061381 , 247.070354 , 247.070307 , 251.079587 , 252.08298 , 257.095105 , 258.098431 , 259.10103 , 262.10963 /)

  Element mass in atomic mass units (1.66 x 10-27 kg)
- real(dp), dimension(nz), parameter prg_ptable_mod::element_vdwr = (/ 1.1 , 1.4 , 1.81 , 1.53 , 1.92 , 1.7 , 1.55 , 1.52 , 1.47 , 1.54 , 2.27 , 1.73 , 1.84 , 2.1 , 1.8 , 1.8 , 1.75 , 1.88 , 2.75 , 2.31 , 2.3 , 2.15 , 2.05 , 2.05 , 2.05 , 2.05 , 2.0 , 2.0 , 2.0 , 2.1 , 1.87 , 2.11 , 1.85 , 1.9 , 1.83 , 2.02 , 3.03 , 2.49 , 2.4 , 2.3 , 2.15 , 2.1 , 2.05 , 2.05 , 2.0 , 2.05 , 2.1 , 2.2 , 2.2 , 1.93 , 2.17 , 2.06 , 1.98 , 2.16 , 3.43 , 2.68 , 2.5 , 2.48 , 2.47 , 2.45 , 2.43 , 2.42 , 2.4 , 2.38 , 2.37 , 2.35 , 2.33 , 2.32 , 2.3 , 2.28 , 2.27 , 2.25 , 2.2 , 2.1 , 2.05 , 2.0 , 2.0 , 2.05 , 2.1 , 2.05 , 1.96 , 2.02 , 2.07 , 1.97 , 2.02 , 2.2 , 3.48 , 2.83 , 2.0 , 2.4 , 2.0 , 2.3 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 , 2.0 /)

  van der Waals radius (in Angstroms)

- real(dp), dimension(nz), parameter prg_ptable_mod::element_covr = (/ 0.31 , 0.28 , 1.28 , 0.96 , 0.84 , 0.76 , 0.71 , 0.66 , 0.57 , 0.58 , 1.66 , 1.41 , 1.21 , 1.11 , 1.07 , 1.05 , 1.02 , 1.06 , 2.03 , 1.76 , 1.7 , 1.6 , 1.53 , 1.39 , 1.39 , 1.32 , 1.26 , 1.24 , 1.32 , 1.22 , 1.22 , 1.2 , 1.19 , 1.2 , 1.2 , 1.16 , 2.2 , 1.95 , 1.9 , 1.75 , 1.64 , 1.54 , 1.47 , 1.46 , 1.42 , 1.39 , 1.45 , 1.44 , 1.42 , 1.39 , 1.39 , 1.38 , 1.39 , 1.4 , 2.44 , 2.15 , 2.07 , 2.04 , 2.03 , 2.01 , 1.99 , 1.98 , 1.98 , 1.96 , 1.94 , 1.92 , 1.92 , 1.89 , 1.9 , 1.87 , 1.87 , 1.75 , 1.7 , 1.62 , 1.51 , 1.44 , 1.41 , 1.36 , 1.36 , 1.32 , 1.45 , 1.46 , 1.48 , 1.4 , 1.5 , 1.5 , 2.6 , 2.21 , 2.15 , 2.06 , 2.0 , 1.96 , 1.9 , 1.87 , 1.8 , 1.69 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 , 1.6 /)

  Covalent radius (in Angstroms)

- real(dp), dimension(nz), parameter prg_ptable_mod::element_ip = (/ 13.5984 , 24.5874 , 5.3917 , 9.3227 , 8.298 , 11.2603 , 14.5341 , 13.6181 , 17.4228 , 21.5645 , 5.1391 , 7.6462 , 5.9858 , 8.1517 , 10.4867 , 10.36 , 12.9676 , 15.7596 , 4.3407 , 6.1132 , 6.5615 , 6.8281 , 6.7462 , 6.7665 , 7.434 , 7.9024 , 7.881 , 7.6398 , 7.7264 , 9.3942 , 5.9993 , 7.8994 , 9.7886 , 9.7524 , 11.8138 , 13.9996 , 4.1771 , 5.6949 , 6.2173 , 6.6339 , 6.7589 , 7.0924 , 7.28 , 7.3605 , 7.4589 , 8.3369 , 7.5762 , 8.9938 , 5.7864 , 7.3439 , 8.6084 , 9.0096 , 10.4513 , 12.1298 , 3.8939 , 5.2117 , 5.5769 , 5.5387 , 5.473 , 5.525 , 5.582 , 5.6437 , 5.6704 , 6.1498 , 5.8638 , 5.9389 , 6.0215 , 6.1077 , 6.1843 , 6.2542 , 5.4259 , 6.8251 , 7.5496 , 7.864 , 7.8335 , 8.4382 , 8.967 , 8.9588 , 9.2255 , 10.4375 , 6.1082 , 7.4167 , 7.2855 , 8.414 , 0.0 , 10.7485 , 4.0727 , 5.2784 , 5.17 , 6.3067 , 5.89 , 6.1941 , 6.2657 , 6.026 , 5.9738 , 5.9914 , 6.1979 , 6.2817 , 6.42 , 6.5 , 6.58 , 6.65 , 4.9 /)

  Ionization energy (in eV)

- real(dp), dimension(nz), parameter prg_ptable_mod::element_ea = (/ 0.75420375 , 0.0 , 0.618049 , 0.0 , 0.279723 , 1.262118 , -0.07 , 1.461112 , 3.4011887 , 0.0 , 0.547926 , 0.0 , 0.43283 , 1.389521 , 0.7465 , 2.0771029 , 3.612724 , 0.0 , 0.501459 , 0.02455 , 0.188 , 0.084 , 0.525 , 0.67584 , 0.0 , 0.151 , 0.6633 , 1.15716 , 1.23578 , 0.0 , 0.41 , 1.232712 , 0.814 , 2.02067 , 3.363588 , 0.0 , 0.485916 , 0.05206 , 0.307 , 0.426 , 0.893 , 0.7472 , 0.55 , 1.04638 , 1.14289 , 0.56214 , 1.30447 , 0.0 , 0.404 , 1.112066 , 1.047401 , 1.970875 , 3.059038 , 0.0 , 0.471626 , 0.14462 , 0.47 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.5 , 0.0 , 0.322 , 0.815 , 0.15 , 1.0778 , 1.56436 , 2.1251 , 2.30861 , 0.0 , 0.377 , 0.364 , 0.942363 , 1.9 , 2.8 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 /)

  Electron affprg_inity (in eV)

- real(dp), dimension(nz), parameter prg_ptable_mod::atom_en = (/ 2.2 , 0.0 , 0.98 , 1.57 , 2.04 , 2.55 , 3.04 , 3.44 , 3.98 , 0.0 , 0.93 , 1.31 , 1.61 , 1.9 , 2.19 , 2.58 , 3.16 , 0.0 , 0.82 , 1.0 , 1.36 , 1.54 , 1.63 , 1.66 , 1.55 , 1.83 , 1.88 , 1.91 , 1.9 , 1.65 , 1.81 , 2.01 , 2.18 , 2.55 , 2.96 , 3.0 , 0.82 , 0.95 , 1.22 , 1.33 , 1.6 , 2.16 , 1.9 , 2.2 , 2.28 , 2.2 , 1.93 , 1.69 , 1.78 , 1.96 , 2.05 , 2.1 , 2.66 , 2.6 , 0.79 , 0.89 , 1.1 , 1.12 , 1.13 , 1.14 , 0.0 , 1.17 , 0.0 , 1.2 , 0.0 , 1.22 , 1.23 , 1.24 , 1.25 , 0.0 , 1.27 , 1.3 , 1.5 , 2.36 , 1.9 , 2.2 , 2.2 , 2.28 , 2.54 , 2.0 , 1.62 , 2.33 , 2.02 , 2.0 , 2.2 , 0.0 , 0.7 , 0.9 , 1.1 , 1.3 , 1.5 , 1.38 , 1.36 , 1.28 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 1.3 , 0.0 /)

  The Pauling electronegativity for this element.

- integer, dimension(nz), parameter prg_ptable_mod::element_maxbonds = (/ 1 , 0 , 1 , 2 , 4 , 4 , 4 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 8 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 12 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 /)

  The maximum expected number of bonds to this element.

- integer, dimension(nz), parameter prg_ptable_mod::element_numel = (/ 1 , 2 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 /)

  Last shell number of electrons.

- character(50), dimension(nz), parameter prg_ptable_mod::element_econf = [character(50) :: "1s" , "1s2" , "1s22s" , "1s22s2" , "1s22s22p" , "1s22s22p2" , "1s22s22p3" , "1s22s22p4" , "1s22s22p5" , "1s22s22p6" , "[Ne]3s" , "[Ne]3s2" , "[Ne]3s23p" , "[Ne]3s23p2" , "[Ne]3s23p3" , "[Ne]3s23p4" , "[Ne]3s23p5" , "[Ne]3s23p6" , "[Ar]4s" , "[Ar]4s2" , "[Ar]3d4s2" , "[Ar]3d24s2" , "[Ar]3d34s2" , "[Ar]3d54s" , "[Ar]3d54s2" , "[Ar]3d64s2" , "[Ar]3d74s2" , "[Ar]3d84s2" , "[Ar]3d104s" , "[Ar]3d104s2" , "[Ar]3d104s24p" , "[Ar]3d104s24p2" , "[Ar]3d104s24p3" , "[Ar]3d104s24p4" , "[Ar]3d104s24p5" , "[Ar]3d104s24p6" , "[Kr]5s" , "[Kr]5s2" , "[Kr]4d5s2" , "[Kr]4d25s2" , "[Kr]4d45s" , "[Kr]4d55s" , "[Kr]4d55s2" , "[Kr]4d75s" , "[Kr]4d85s" , "[Kr]4d10" , "[Kr]4d105s" , "[Kr]4d105s2" , "[Cd]5p" , "[Cd]5p2" , "[Cd]5p3" , "[Cd]5p4" , "[Cd]5p5" , "[Cd]5p6" , "[Xe]6s" , "[Xe]6s2" , "[Xe]5d6s2" , "[Xe]4f5d6s2" , "[Xe]4f36s2" , "[Xe]4f46s2" , "[Xe]4f56s2" , "[Xe]4f66s2" , "[Xe]4f76s2" , "[Xe]4f75d6s2" , "[Xe]4f96s2" , "[Xe]4f106s2" , "[Xe]4f116s2" , "[Xe]4f126s2" , "[Xe]4f136s2" , "[Xe]4f146s2" , "[Xe]4f145d6s2" , "[Xe]4f145d26s2" , "[Xe]4f145d36s2" , "[Xe]4f145d46s2" , "[Xe]4f145d56s2" , "[Xe]4f145d66s2" , "[Xe]4f145d76s2" , "[Xe]4f145d96s" , "[Xe]4f145d106s" , "[Xe]4f145d106s2" , "[Hg]6p" , "[Hg]6p2" , "[Hg]6p3" , "[Hg]6p4" , "[Hg]6p5" , "[Hg]6p6" , "[Rn]7s" , "[Rn]7s2" , "[Rn]6d7s2" , "[Rn]6d27s2" , "[Rn]5f26d7s2" , "[Rn]5f36d7s2" , "[Rn]5f46d7s2" , "[Rn]5f67s2" , "[Rn]5f77s2" , "[Rn]5f76d7s2" , "[Rn]5f97s2" , "[Rn]5f107s2" , "[Rn]5f117s2" , "[Rn]5f127s2" , "[Rn]5f137s2" , "[Rn]5f147s2" , "[Rn]5f147s27p"]

The electronic configuration.

10.24 /tmp/qmd-progress/src/prg_pulaycomponent_mod.F90 File Reference

Modules

- module prg_pulaycomponent_mod

  Produces a matrix to get the Pulay Component of the forces.

Functions/Subroutines

- subroutine, public prg_pulaycomponent_mod::prg_pulaycomponent0 (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)

  At $T = 0K, P = \rho H \rho.$
- subroutine, public prg_pulaycomponent_mod::prg_pulaycomponentt (rho_bml, ham_bml, zmat_bml, pcm_bml, threshold, M, bml_type, verbose)

  At $T > 0K, P = \rho H S^-1 + S^{-1} H \rho.$
- subroutine, public prg_pulaycomponent_mod::prg_get_pulayforce (nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, dSz_bml, hindex, FPUL, threshold)

  Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}].$

Variables

- integer, parameter prg_pulaycomponent_mod::dp = kind(1.0d0)

10.25 /tmp/qmd-progress/src/prg_pulaymixer_mod.F90 File Reference

Data Types

- type prg_pulaymixer_mod::mx_type

Modules

- module prg_pulaymixer_mod

  Pulay mixer mode.

Functions/Subroutines

- subroutine, public prg_pulaymixer_mod::prg_parse_mixer (input, filename)

  The parser for the mixer routines.
- subroutine, public prg_pulaymixer_mod::prg_qmixer (charges, oldcharges, dqin, dqout, scferror, piter, pulaycoef, mpulay, verbose)

  Mixing the charges to acelerate scf convergence.
- subroutine, public prg_pulaymixer_mod::prg_linearmixer (charges, oldcharges, scferror, linmixcoef, verbose)

  Routine to perform linear mixing.

Variables

- integer, parameter prg_pulaymixer_mod::dp = kind(1.0d0)

10.26 /tmp/qmd-progress/src/prg_quantumdynamics_mod.F90 File Reference

Modules

- module prg_quantumdynamics_mod

  A module to add in common quantum dynamical operations.

Functions/Subroutines

- subroutine, public prg_quantumdynamics_mod::prg_kick_density (kick_direc, kick_mag, dens, norbs, mdim, S, SINV, which_atom, r, bmltype, thresh)

  Provides perturbation to initial density matrix in the form of an electric field kick. This routine does: $\rho_{\hat{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ *where* $\hat{V}$ *is the field disturbance.*
- subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_cplxmat (matrix_type, element_type, thresh, a_dense)

  This computes the sparsity of a complex matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ *where* f *is the sparsity,* N_0 *is the number of values less than the threshold, and* N_{tot} *is the total number of values. The sparsity and threshold are printed to the screen.*
- subroutine, public prg_quantumdynamics_mod::prg_get_sparsity_realmat (matrix_type, element_type, thresh, a_dense)

  This computes the sparsity of a real matrix given a threshold value This routine does: $f = \frac{N_0}{N_{tot}}$ *where* f *is the sparsity,* N_0 *is the number of values less than the threshold, and* N_{tot} *is the total number of values. The sparsity and threshold are printed to the screen.*
- subroutine, public prg_quantumdynamics_mod::prg_kick_density_bml (kick_direc, kick_mag, rho_bml, s_↩
bml, sinv_bml, mdim, which_atom, r, matrix_type, thresh)

  Provides perturbation to initial density matrix in the form of an electric field kick given input matricies in BML format. This routine does: $\rho_{\hat{kick}} = \exp\frac{-i}{\hbar}\hat{V}\hat{\rho}\hat{S}\exp\frac{i}{\hbar}\hat{V}\hat{S^{-1}}$ *where* $\hat{V}$ *is the field disturbance.*
- subroutine, public prg_quantumdynamics_mod::prg_lvni_bml (h1_bml, sinv_bml, dt, hbar, rhoold_bml, rho↩
_bml, aux_bml, matrix_type, mdim, thresh)

  Performs Liouville-von Neumann integration using leap-frog method. This routine does: $\hat{\rho}(t+\Delta t) = \hat{\rho}(t-\Delta t) + 2\Delta t\frac{\partial\hat{\rho}(t)}{\partial t}$ *where the time derivative of the density matrix is defined as follows:* $\frac{\partial\hat{\rho}(t)}{\partial t} = \frac{-i}{\hbar}\left(S^{-1}\hat{H}(t)\hat{\rho}(t) - \hat{\rho}(t)\hat{H}(t)S^{-1}\right)$.
- subroutine, public prg_quantumdynamics_mod::prg_getcharge (rho_bml, s_bml, charges, aux_bml, z, spindex, N, nats, thresh)

  Constructs the charges from the density matrix.
- subroutine, public prg_quantumdynamics_mod::prg_getdipole (charges, r, mu)

  This routine computes the dipole moment of the system with units determined by the units of the coordinate matrix and charges given.
- subroutine, public prg_quantumdynamics_mod::prg_excitation (fill_mat, orbit_orig, orbit_exci)

  Produce an excitation in the initially calculated density matrix to.

Variables

- integer, parameter prg_quantumdynamics_mod::dp = kind(1.0d0)

10.27 /tmp/qmd-progress/src/prg_response_mod.F90 File Reference

Data Types

- type prg_response_mod::respdata_type

Modules

- module prg_response_mod

  Module to compute the density matrix response and related quantities.

Functions/Subroutines

- subroutine, public prg_response_mod::prg_parse_response (RespData, filename)

  The parser for the calculation of the DM response.
- subroutine, public prg_response_mod::prg_compute_dipole (charges, coordinate, dipoleMoment, factor, verbose)

  To compute the dipole moment of the system. The units of the dipole moment are determined by the units of the coordinates and charges that are given.
- subroutine, public prg_response_mod::prg_write_dipole_tcl (dipoleMoment, file, factor, verbose)

  To visualize a dipole moment using VMD. This will prg_generate a .tcl script that could be run using VMD To visualize with VMD: $ vmd -e dipole.tcl.
- subroutine, public prg_response_mod::prg_compute_polarizability (rsp_bml, prt_bml, polarizability, factor, verbose)

  To compute the polarizability of the system. The units of the directional polarizability are determined by the units of the perturbation and Hamiltonian. This equation can be found in [5] equation 4a. Note that in equation 4a of the reference there is a 2 that account for the double occupancy which is not present in this case cause the density matrix construction is done by taking the occupancy into account.
- subroutine, public prg_response_mod::prg_pert_from_file (prt_bml, norb)

  Read perturbation from file.
- subroutine, public prg_response_mod::prg_compute_response_rs (ham_bml, prt_bml, rsp_bml, lambda, bndfil, threshold, verbose)

  Computes the first order response density matrix using Rayleigh Schrodinger Perturbation theory The transformation hereby performed are:
- subroutine, public prg_response_mod::prg_compute_response_fd (ham_bml, prt_bml, rsp_bml, prg_delta, bndfil, threshold, verbose)

  Computes the first order response density matrix using finite differences. The transformation hereby performed are:
- subroutine, public prg_response_mod::prg_pert_constant_field (field, intensity, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

  Apply a constant field perturbation through the dipole moment operator ($\hat{\mu} = e\hat{\mathbf{r}}$). In the matrix representation, this is: $H^{(1)} = \lambda\frac{1}{2}(S\,e\mathbf{r}\cdot\mathbf{E} + e\mathbf{r}\cdot\mathbf{E}S)$. The symmetrization is done in order to preserve the Hermiticity of H. In this case the whole system will be affected by the field. In a latter version we will add the possibility of applying this field to a region of the system. In this implementation $e = 1$ and units can be transformed by using the parameter λ.
- subroutine, public prg_response_mod::prg_pert_sin_pot (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

Apply a sinusoidal length dependent potential ($\sin(\tilde{\boldsymbol{r}}_x)$) where $\boldsymbol{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\boldsymbol{r}}_x) + \sin(\tilde{\boldsymbol{r}}_x)S)$. $\tilde{\boldsymbol{r}}_x = 2\pi(\boldsymbol{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.

- subroutine, public prg_response_mod::prg_pert_cos_pot (direction, lx, coordinate, lambda, prt_bml, threshold, spindex, norbi, verbose, over_bml)

  Apply a cosine length dependent potential ($\cos(\tilde{\boldsymbol{r}}_x)$) where $\boldsymbol{r}_x$ is the x coordinate. The Hamiltonian gets modified as follows: $H^{(1)} = \frac{1}{2}\lambda(S\sin(\tilde{\boldsymbol{r}}_x) + \sin(\tilde{\boldsymbol{r}}_x)S)$. $\tilde{\boldsymbol{r}}_x = 2\pi(\boldsymbol{r}/l_x) - \pi$. The symmetrization is done in order to preserve the Hermiticity of H. Units can be transformed by using the parameter λ.
- subroutine, public prg_response_mod::prg_compute_response_sp2 (ham_bml, prt_bml, rsp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, threshold, verbose)

  Finds the first order response matrix from a Hamiltonian matrix.
- subroutine, public prg_response_mod::prg_project_response (rsp_bml, over_bml, spindex, norbi, coordinates, rspfunc, verbose)

  Project the response onto atomic positions. First order response to the perturbation ($\rho^{(1)}$) projected onto the atomic position. Basically: $rsp(i) = \sum_{\alpha \in i} \rho^{(1)}_{\alpha\alpha}$, where orbital α belong to atom i.
- subroutine, public prg_response_mod::prg_canon_response_vector (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, thresh, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.
- subroutine, public prg_response_mod::prg_canon_response (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.
- subroutine, public prg_response_mod::prg_canon_response_orig (P1_bml, H1_bml, Nocc, beta, evals, mu0, m, thresh, HDIM)

  First-order Canonical Density Matrix Perturbation Theory.

Variables

- integer, parameter prg_response_mod::dp = kind(1.0d0)
- real(dp), parameter prg_response_mod::pi = 3.14159265358979323846264338327950_dp

10.28 /tmp/qmd-progress/src/prg_sp2_fermi_mod.F90 File Reference

Modules

- module prg_sp2_fermi_mod

  The SP2 Fermi module.

Functions/Subroutines

- subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi_init (h_bml, nsteps, nocc, tscale, threshold, occErr↩ Limit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist)

  Truncated SP2 prg_initialization.
- subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi_init_norecs (h_bml, nsteps, nocc, tscale, threshold, occErrLimit, traceLimit, x_bml, mu, beta, h1, hN, sgnlist, verbose)

  Truncated SP2 prg_initialization. This routine also gives back the Number of SP2 recursive steps that gets a Pseudo-Fermi distribution with a temperature close to the target temperature which is entered using parameter beta = (1/KbT).
- subroutine, public prg_sp2_fermi_mod::prg_sp2_fermi (h_bml, osteps, nsteps, nocc, mu, beta, h1, hN, sgnlist, threshold, eps, traceLimit, x_bml)

  Calculate Truncated SP2.
- subroutine, public prg_sp2_fermi_mod::prg_sp2_entropy_function (mu, h1, hN, nsteps, sgnlist, GG, ee)

Calculate SP2 entropy function using gaussian quadrature. Note that GG and ee are allocated and returned from this routine.

- real(dp) function, public prg_sp2_fermi_mod::sp2_entropy_ts (D0_bml, GG, ee)

  Test SP2 entropy. Get the entropy contribution TS to the total free energy.
- real(dp) function, public prg_sp2_fermi_mod::sp2_inverse (f, mu, h1, hN, nsteps, sgnlist)

  Calculate the SP2 inverse.
- real(dp) function prg_sp2_fermi_mod::absmaxderivative (func, de)

  Gets the absolute maximum of the derivative of a function.

Variables

- integer, parameter prg_sp2_fermi_mod::dp = kind(1.0d0)

10.29 /tmp/qmd-progress/src/prg_sp2_mod.F90 File Reference

Modules

- module prg_sp2_mod

  The SP2 module.

Functions/Subroutines

- subroutine, public prg_sp2_mod::prg_sp2_basic (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)

  Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.
- subroutine, public prg_sp2_mod::prg_sp2_basic_tcore (h_bml, rho_bml, rhofull_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_mod::prg_sp2_alg2 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_mod::prg_sp2_alg2_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv, verbose)
- subroutine, public prg_sp2_mod::prg_sp2_alg2_seq (h_bml, rho_bml, threshold, pp, icount, vv, verbose)
- subroutine, public prg_sp2_mod::prg_prg_sp2_alg2_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, verbose)
- subroutine, public prg_sp2_mod::prg_sp2_alg1 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public prg_sp2_mod::prg_sp2_alg1_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv)
- subroutine, public prg_sp2_mod::prg_sp2_alg1_seq (h_bml, rho_bml, threshold, pp, icount, vv)
- subroutine, public prg_sp2_mod::prg_prg_sp2_alg1_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval)
- subroutine, public prg_sp2_mod::prg_sp2_submatrix (ham_bml, rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

  Perform SP2 algorithm using sequence and calculate norm for a submatrix.
- subroutine, public prg_sp2_mod::prg_sp2_submatrix_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

Variables

- integer, parameter prg_sp2_mod::dp = kind(1.0d0)
- integer, parameter prg_sp2_mod::dp1 = kind(1.0)

10.30 /tmp/qmd-progress/src/prg_sp2parser_mod.F90 File Reference

Data Types

- type prg_sp2parser_mod::sp2data_type

  General SP2 solver type.

Modules

- module prg_sp2parser_mod

  SP2 parser.

Functions/Subroutines

- subroutine, public prg_sp2parser_mod::prg_parse_sp2 (sp2data, filename)

  The parser for SP2 solver.

Variables

- integer, parameter prg_sp2parser_mod::dp = kind(1.0d0)

10.31 /tmp/qmd-progress/src/prg_subgraphloop_mod.F90 File Reference

Modules

- module prg_subgraphloop_mod

  The subgraphloop module.

Functions/Subroutines

- subroutine, public prg_subgraphloop_mod::prg_subgraphsp2loop (h_bml, g_bml, rho_bml, gp, threshold)
- subroutine, public prg_subgraphloop_mod::prg_collectmatrixfromparts (gp, rho_bml)

  Collect distributed parts into same matrix.
- subroutine, public prg_subgraphloop_mod::prg_balanceparts (gp)
- subroutine, public prg_subgraphloop_mod::prg_partordering (gp)

  Set row ordering bases on parts.
- subroutine, public prg_subgraphloop_mod::prg_getgrouppartitionhalosfromgraph (gp, g_bml, hnode, djflag)

  Get core+halo indeces for all partitions only using the graph.
- subroutine, public prg_subgraphloop_mod::prg_getpartitionhalosfromgraph (gp, g_bml, djflag)

  Get core+halo indeces for all partitions only using the graph.

Variables

- integer, parameter prg_subgraphloop_mod::dp = kind(1.0d0)

10.32 /tmp/qmd-progress/src/prg_syrotation_mod.F90 File Reference

Data Types

- type prg_syrotation_mod::rotation_type

  Rotation type.

Modules

- module prg_syrotation_mod

  A module to rotate the coordinates of a sybsystem in chemical systems.

Functions/Subroutines

- subroutine, public prg_syrotation_mod::prg_parse_rotation (rot, filename)

  The parser for rotation.
- subroutine, public prg_syrotation_mod::prg_rotate (rot, r, verbose)

  Rotation routine.

Variables

- integer, parameter prg_syrotation_mod::dp = kind(1.0d0)

10.33 /tmp/qmd-progress/src/prg_system_mod.F90 File Reference

Data Types

- type prg_system_mod::estruct_type

  Electronic structure type.
- type prg_system_mod::system_type

  System type.

Modules

- module prg_system_mod

  A module to read and handle chemical systems.

Functions/Subroutines

- subroutine, public prg_system_mod::prg_get_nameandext (fullfilename, filename, ext)

  Get the name and extension of a file.
- subroutine, public prg_system_mod::prg_parse_system (system, filename, extin)

  The parser for the chemical system.
- subroutine, public prg_system_mod::prg_destroy_system (sy)

  Deallocates all the arrays within a system.
- subroutine, public prg_system_mod::prg_write_system (system, filename, extin)

  Write system in .xyz, .dat or pdb file.
- subroutine, public prg_system_mod::prg_write_trajectory (system, iter, each, prg_deltat, filename, extension)

  Write trajectory in .xyz, .dat or pdb file.
- subroutine, public prg_system_mod::prg_write_trajectoryandproperty (system, iter, each, prg_deltat, scalarprop, filename, extension)

  Write trajectory and atomic properties. Only pdb file.
- subroutine, public prg_system_mod::prg_make_random_system (system, nats, seed, lx, ly, lz)

  Make random Xx system.
- subroutine, public prg_system_mod::prg_parameters_to_vectors (abc_angles, lattice_vector)

  Transforms the lattice parameters into lattice vectors.
- subroutine, public prg_system_mod::prg_vectors_to_parameters (lattice_vector, abc_angles)

  Transforms the lattice vectors into lattice parameters.
- subroutine, public prg_system_mod::prg_get_origin (coords, origin)

  Get the origin of the coordinates.
- subroutine, public prg_system_mod::prg_get_distancematrix (coords, dmat)

  Get the distance matrix.
- subroutine, public prg_system_mod::prg_translateandfoldtobox (coords, lattice_vectors, origin, verbose)

  Translate and fold to box.
- subroutine, public prg_system_mod::prg_centeratbox (coords, lattice_vectors, verbose)

  Translate geometric center to the center of the box.
- subroutine, public prg_system_mod::prg_wraparound (coords, lattice_vectors, index, verbose)

  Wrap around atom i using pbc.
- subroutine, public prg_system_mod::prg_translatetogeomcandfoldtobox (coords, lattice_vectors, origin)

  Translate to geometric center.
- subroutine, public prg_system_mod::prg_replicate (coords, symbols, lattice_vectors, nx, ny, nz)

  Extend/replicate system along lattice vectors.
- subroutine, public prg_system_mod::prg_replicate_system (sy, syf, nx, ny, nz)

  Extend/replicate a system type along the lattice vectors.
- subroutine, public prg_system_mod::prg_cleanuprepeatedatoms (nats, coords, symbols, verbose)

  Cleanup repeated atoms we might have in the system.
- subroutine, public prg_system_mod::prg_get_recip_vects (lattice_vectors, recip_vectors, volr, volk)

  Get the volume of the cell and the reciprocal vectors: This soubroutine computes:
- subroutine, public prg_system_mod::prg_get_dihedral (coords, id1, id2, id3, id4, dihedral)

  Get the dihedral angle given four atomic positions.
- subroutine, public prg_system_mod::prg_get_covgraph (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)

  Get the covalency graph in bml format.
- subroutine prg_system_mod::prg_get_covgraph_int (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)
- subroutine, public prg_system_mod::prg_get_covgraph_h (sy, nnStructMindist, nnStruct, nrnnstruct, rcut, graph_h, mdimin, verbose)

  Get the covanlency graph.
- subroutine, public prg_system_mod::prg_get_subsystem (sy, lsize, indices, sbsy, verbose)

Get a subsystem out of the total system.

- subroutine, public prg_system_mod::prg_destroy_subsystems (sbsy, verbose)

  Destroy allocated subsystem.
- subroutine, public prg_system_mod::prg_molpartition (sy, npart, nnStructMindist, nnStruct, nrnnstruct, hetatm, gp, verbose)

  Partition by molecule.
- subroutine, public prg_system_mod::prg_get_partial_atomgraph (rho_bml, hindex, gch_bml, threshold, verbose)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_system_mod::prg_collect_graph_p (rho_bml, nc, nats, hindex, chindex, graph_↩ p, threshold, mdimin, verbose)

  Collect the small graph to build the full graph.
- subroutine, public prg_system_mod::prg_merge_graph (graph_p, graph_h)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_system_mod::prg_merge_graph_adj (graph_p, graph_h, xadj, adjncy)

  Get partial subgraph based on the Density matrix.
- subroutine, public prg_system_mod::prg_adj2bml (xadj, adjncy, bml_type, g_bml)

  prg_adj2bml
- subroutine, public prg_system_mod::prg_graph2bml (graph, bml_type, g_bml)

  Graph2bml.
- subroutine, public prg_system_mod::prg_graph2vector (graph, vector, maxnz)

  Vectorize graph.
- subroutine, public prg_system_mod::prg_vector2graph (vector, graph, maxnz)

  Back to graph.
- subroutine, public prg_system_mod::prg_sortadj (xadj, adjncy)

  Sort adj NOTE: this might not be needed anymre since the bml_get_adj routine is sorting the values.

Variables

- integer, parameter prg_system_mod::dp = kind(1.0d0)

10.34 /tmp/qmd-progress/src/prg_timer_mod.F90 File Reference

Data Types

- type prg_timer_mod::timer_status_t

  Timer status type.

Modules

- module prg_timer_mod

  The timer module.

Functions/Subroutines

- subroutine, public prg_timer_mod::timer_prg_init ()

  Initialize timers.
- subroutine prg_timer_mod::prg_timer_getid ()

  Get timer id.
- subroutine, public prg_timer_mod::prg_timer_shutdown ()

  Done with timers.
- subroutine, public prg_timer_mod::prg_timer_start (itimer, tag)

  Start Timing.
- subroutine, public prg_timer_mod::prg_timer_stop (itimer, verbose)

  Stop timing.
- subroutine, public prg_timer_mod::prg_timer_collect ()
- subroutine, public prg_timer_mod::prg_timer_results ()
- real(8) function, public prg_timer_mod::time2milliseconds ()
- subroutine, public prg_timer_mod::prg_print_date_and_time (tag)
- character(2) function, private prg_timer_mod::int2char (ival)

Variables

- integer, parameter prg_timer_mod::dp = kind(1.0d0)
- integer, public prg_timer_mod::loop_timer
- integer, public prg_timer_mod::sp2_timer
- integer, public prg_timer_mod::genx_timer
- integer, public prg_timer_mod::part_timer
- integer, public prg_timer_mod::subgraph_timer
- integer, public prg_timer_mod::deortho_timer
- integer, public prg_timer_mod::ortho_timer
- integer, public prg_timer_mod::zdiag_timer
- integer, public prg_timer_mod::graphsp2_timer
- integer, public prg_timer_mod::subind_timer
- integer, public prg_timer_mod::subext_timer
- integer, public prg_timer_mod::subsp2_timer
- integer, public prg_timer_mod::suball_timer
- integer, public prg_timer_mod::bmult_timer
- integer, public prg_timer_mod::badd_timer
- integer, public prg_timer_mod::dyn_timer
- integer, public prg_timer_mod::mdloop_timer
- integer, public prg_timer_mod::buildz_timer
- integer, public prg_timer_mod::realcoul_timer
- integer, public prg_timer_mod::recipcoul_timer
- integer, public prg_timer_mod::pairpot_timer
- integer, public prg_timer_mod::halfverlet_timer
- integer, public prg_timer_mod::pos_timer
- integer, public prg_timer_mod::nlist_timer
- integer prg_timer_mod::tstart_clock
- integer prg_timer_mod::tstop_clock
- integer prg_timer_mod::tclock_rate
- integer prg_timer_mod::tclock_max
- integer prg_timer_mod::num_timers
- type(timer_status_t), dimension(:), allocatable prg_timer_mod::ptimer

10.35 /tmp/qmd-progress/src/prg_xlbo_mod.F90 File Reference

Data Types

- type prg_xlbo_mod::xlbo_type

  General xlbo solver type.

Modules

- module prg_xlbo_mod

  A module to perform XLBO integration.

Functions/Subroutines

- subroutine, public prg_xlbo_mod::prg_parse_xlbo (xlbo, filename)

  The parser for XLBO parser.
- subroutine, public prg_xlbo_mod::prg_xlbo_nint (charges, n, n_0, n_1, n_2, n_3, n_4, n_5, mdstep, xl)

  This routine integrates the dynamical variable "n".
- subroutine, public prg_xlbo_mod::prg_xlbo_fcoulupdate (fcoul, charges, n)

  Adjust forces for the linearized XLBOMD functional.

Variables

- integer, parameter prg_xlbo_mod::dp = kind(1.0d0)
- real(dp), parameter prg_xlbo_mod::c0 = -6.0_dp

  Coefficients for modified Verlet integration.
- real(dp), parameter prg_xlbo_mod::c1 = 14.0_dp
- real(dp), parameter prg_xlbo_mod::c2 = -8.0_dp
- real(dp), parameter prg_xlbo_mod::c3 = -3.0_dp
- real(dp), parameter prg_xlbo_mod::c4 = 4.0_dp
- real(dp), parameter prg_xlbo_mod::c5 = -1.0_dp
- real(dp), parameter prg_xlbo_mod::kappa = 1.82_dp

  Coefficients for modified Verlet integration.
- real(dp), parameter prg_xlbo_mod::alpha = 0.018_dp
- real(dp), parameter prg_xlbo_mod::cc = 0.9_dp

10.36 /tmp/qmd-progress/src/prg_xlbokernel_mod.F90 File Reference

Modules

- module prg_xlbokernel_mod

  Pre-conditioned O(N) calculation of the kernel for XL-BOMD.

Functions/Subroutines

- subroutine prg_xlbokernel_mod::invert (A, AI, N)
- subroutine, public prg_xlbokernel_mod::prg_kernel_multirank_latte (KRes, KK0_bml, Res, FelTol, L, LMAX, NUMRANK, HO_bml, mu, beta, RXYZ, Box, Hubbard_U, Element_Pointer, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, nebcoul, totnebcoul, Hinxlist, S_bml, Z_bml, Nocc, Inv_bml, H1_bml, DO_bml, D1_bml, m_↩ rec, threshold, Nr_elem, mdstep, update)

    Compute low rank approximation of (K0∗J)^(-1)∗K0∗(q[n]-n)(for LATTE)
- subroutine prg_xlbokernel_mod::prg_update_preconditioner (K0, v, fv, Nr_atoms, threshold)
- subroutine, public prg_xlbokernel_mod::prg_kernel_multirank (KRes, KK0_bml, Res, FelTol, L, LMAX, HO↩ _bml, mu, beta, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb↩ _acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, S_bml, Z_bml, Nocc, Znuc, Inv_bml, H1_bml, X_bml, Y_bml, DO_bml, D1_bml, m_rec, threshold)
- subroutine, public prg_xlbokernel_mod::prg_full_kernel_latte (KK, DO_bml, mu0, RXYZ, Box, Hubbard_U, Element_Pointer, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, Hinxlist, S_bml, Z_bml, Inv_bml, D1_bml, H1_bml, HO_bml, Nocc, m_rec, threshold, beta, Nr_elem, nebcoul, totnebcoul)

    Compute full inverse Jacobian of q[n]-n (for LATTE)
- subroutine, public prg_xlbokernel_mod::prg_full_kernel (KK, DO_bml, mu0, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, S_bml, Z_bml, Inv_bml, D1_bml, H1_bml, HO_bml, Y_bml, X↩ _bml, Nocc, Znuc, m_rec, threshold, beta, diagonal)

    Compute full inverse Jacobian of q[n]-n (for development code)
- subroutine, public prg_xlbokernel_mod::prg_kernel_matrix_multirank (KRes, KK0_bml, Res, FelTol, L, LM↩ AX, HO_bml, mu, beta, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERATIO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, S_bml, Z_bml, Nocc, Znuc, Inv_bml, H1_bml, X_bml, Y_bml, DO_bml, D1_bml, m_rec, threshold)

Variables

- integer, parameter prg_xlbokernel_mod::dp = kind(1.0d0)

10.37 /tmp/qmd-progress/src/prg_xlkernel_mod.F90 File Reference

Data Types

- type prg_xlkernel_mod::xlk_type

Modules

- module prg_xlkernel_mod

    Add name.

Functions/Subroutines

- subroutine, public prg_xlkernel_mod::prg_parse_xlkernel (input, filename)

  The parser for the mixer routines.
- subroutine, public prg_xlkernel_mod::prg_fermi (D0, QQ, ee, gap, Fe_vec, mu0, H, Z, Nocc, T, OccErrLim, MaxIt, HDIM)
- subroutine, public prg_xlkernel_mod::prg_kernel_fermi_full (KK, JJ, D0, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERAT↩ IO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, public prg_xlkernel_mod::prg_v_kernel_fermi (D0, dq_dv, v, mu0, mu1, T, RX, RY, RZ, LBox, Hubbard_U, Element_Type, Nr_atoms, MaxIt, eps, m, HDIM, Max_Nr_Neigh, Coulomb_acc, TIMERAT↩ IO, nnRx, nnRy, nnRz, nrnnlist, nnType, H_INDEX_START, H_INDEX_END, H, S, Z, Nocc, Znuc, QQ, ee, Fe_vec)
- subroutine, private prg_xlkernel_mod::prg_get_deriv_finite_temp (P1, H0, H1, Nocc, T, Q, ev, fe, mu0, eps, HDIM)
- subroutine, private prg_xlkernel_mod::prg_mmult (alpha, A, B, beta, C, TA, TB, HDIM)
- subroutine, private prg_xlkernel_mod::prg_eig (A, Q, ee, type, HDIM)
- subroutine, private prg_xlkernel_mod::prg_inv (X, XI, HDIM)
- subroutine, public prg_xlkernel_mod::prg_rank1 (verbose)

  Rank1 kernel ....

Variables

- integer, parameter prg_xlkernel_mod::dp = kind(1.0d0)

10.38 /tmp/qmd-progress/tests/README.md File Reference

Bibliography

[1] S. M. Mniszewski, M. J. Cawkwell, M. E. Wall, J. Mohd-Yusof, N. Bock, T. C. Germann, and A. M. N. Niklasson. Efficient parallel linear scaling construction of the density matrix for born–oppenheimer molecular dynamics. *Journal of Chemical Theory and Computation*, 11(10):4644–4654, 2015. PMID: 26574255. 95, 96, 275, 276

[2] Christian F. A. Negre, Susan M. Mniszewski, Marc J. Cawkwell, Nicolas Bock, Michael E. Wall, and Anders M. N. Niklasson. Recursive factorization of the inverse overlap matrix in linear-scaling quantum molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 12(7):3063–3073, 2016. PMID: 27267207. 58

[3] Anders M. N. Niklasson. A note on the pulay force at finite electronic temperatures. *The Journal of Chemical Physics*, 129(24), 2008. 165

[4] Anders M. N. Niklasson, Valéry Weber, and Matt Challacombe. Nonorthogonal density-matrix perturbation theory. *The Journal of Chemical Physics*, 123(4), 2005. 178

[5] Valéry Weber, Anders M. N. Niklasson, and Matt Challacombe. Higher-order response in o(n) by perturbed projection. *The Journal of Chemical Physics*, 123(4), 2005. 177, 182, 342

Index