progress

0.0.1

Generated by Doxygen 1.8.6

Fri Oct 21 2016 16:53:19

Contents

Chapter 1

A library for quantum chemistry solvers.

PROGRESS: Parallel, Rapid O(N) and Graph-based Recursive Electronic Structure Solver.

This library is focused on the development of general solvers that are commonly used in *quantum chemistry packages*.

This library has to be installed with the *Basic Matrix Library* to be able to use it. This library can be downloaded from: BML

Author

Anders M. N. Niklasson amn@lanl.gov
Christian F. A. Negre cnegre@lanl.gov
Marc J. Cawkwell cawkwell@lanl.gov
Nicolas Bock nbock@lanl.gov
Susan M. Mniszewski smm@lanl.gov
Michael E. Wall mewall@lanl.gov

Copyright

Los Alamos National Laboratory 2015

How to build:

```
PKG_CONFIG_PATH=<BML install path>/lib64/pkgconfig ./build.sh
```

You can use:

```
locate bml.pc
```

or sudo find / | grep bml.pc

to find the pkgconfig folder path.

How to install:

```
cd build
$ sudo make install
```

To specify intel fortran compiler:

```
FC=ifort PKG_CONFIG_PATH=<BML install path>/lib64/pkgconfig ./build.sh
```

To build with the gfortran compiler and OpenMP:

```
CC=gcc FC=gfortran CMAKE_BUILD_TYPE=Release PROGRESS_OPENMP=yes PKG_CONFIG_PATH=<BML install path>/lib64/pkgcc
```

To build with OpenMP and MPI and testing enabled:

CC=mpicc FC=mpif90 CMAKE_BUILD_TYPE=Release PROGRESS_OPENMP=yes PROGRESS_MPI=yes PROGRESS_TESTING=yes PKG_CONFIG_PATH=<BML install="" path>=""></BML>/lib64/pkgconfig CMAKE_INSTALL_PREFIX=<PROGRESS install="" path>=""> ./build.sh configure

To build with OpenMP and MPI and testing enabled and example programs built:

CC=mpicc FC=mpif90 CMAKE_BUILD_TYPE=Release PROGRESS_OPENMP=yes PROGRESS_MPI=yes PROGRESS_TESTING=yes PROGRESS_EXAMPLES=yes PKG_CONFIG_PATH=<BML install="" path>=""></BML>/lib64/pkgconfig CMAKE_INSTALL_PREFIX=<PROGRESS install="" path>=""> ./build.sh configure

To build with OpenMP and MPI and testing enabled and example programs built and the METIS graph partitioning library:

CC=mpicc FC=mpif90 CMAKE_BUILD_TYPE=Release PROGRESS_OPENMP=yes PROGRESS_MPI=yes PROGRESS_GRAPHLIB=yes PROGRESS_TESTING=yes PROGRESS_EXAMPLES=yes PKG_CONFIG_PATH=<BML install="" path>=""></BML>/lib64/pkgconfig CMAKE_INSTALL_PREFIX=<PROGRESS install="" path>=""> EXTRA_LINK_FLAGS="-L<metis directory> -lmetis" ./build.sh configure

Chapter 2

Glossary for the code

Please see variable_names for a full list of how to name variables.

Some useful tips for naming variables:

- Avoid UPPER CASE unless it is really necessary.
- Use _ symbol for naming variables that are used internally (eg: this_variable)
- Use combination of upper and lower case for input and output variables (eg: ThisVariable)
- AVOID using a single letter such as "r", "m" etc unless is really particular and private.

Chapter 3

Testing the Progress library

Testing program for the progress library

To run the tests:

Go into the build folder and type:

```
make test
```

To run the tests in verbose mode:

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

To run a single test:

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

```
ctest -R <test_name> --verbose
```

, 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

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 4

Todo List

Type dos_mod

Add LDOS.

Subprogram pulaycomponent_mod::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 pulaycomponent_mod::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.

Type pulaymixer_mod

add the density matrix mixer.

Type response_mod

Add the response scf

Change name response_SP2 to dm_prt_response

Change name response_rs to rs_prt_response More information about the teory can be found at **?** and Niklasson2015

Subprogram response_mod::pert_from_file (pert_bml, norb)

Add read perturbation from file

Subprogram system_mod::parse_system (system, filename, extin)

Integrate this loop in the loop for building the splist.

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Chapter 6

Modules Index

6.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 7

Data Type Index

7.1 Data Types List

Here are the data types with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Module Documentation

9.1 (LATTE related routines)

Data Types

- module graph_sp2parser_mod

  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:

9.1.1 Detailed Description

9.2 (PROGRESS related routines)

Data Types

- module charges_mod

  A module to handle the charges of the system.
- module density_mod

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

  Extra routines:
- module genz_mod

  To produce a matrix Z which is needed to orthogonalize H.
  $H_{orth} = Z^{\dagger} H Z$ Please see Negre 2016 **?**.
- module graph_mod

  The graph module.
- module homolumo_mod

  The homolumo module.
- module initmatrices_mod

  Initialization module.
- module kernelparser_mod

  Some general parsing functions.
- module nonortho_mod

  Module to orthogonalize and deorthogonalize any operator.
  Typically the Hamiltonin needs to be orthogonalized: $H_{\mathrm{ortho}} = Z^{\dagger} H Z$.
- module openfiles_mod

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

  The parallel module.
- module progress_mod

  The progress module.
- module ptable_mod

  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.
- module pulaycomponent_mod

  Produces a matrix to get the Pulay Component of the forces.
  Please see Niklasson 2008 **?**.
- module pulaymixer_mod

  Pulay mixer mode.
  Gets the best coefficient for mixing the charges during scf.
- module response_mod

  Module to compute the response and related quantities.
- module sp2_mod

  The SP2 module.
- module sp2parser_mod

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

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

The timer module.
Example use of dynamic timing:

- module xlbo_mod

  A module to perform XLBO integration.

9.2.1 Detailed Description

9.3 (EXTERNAL related routines)

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

Functions/Subroutines

- program gpmd

  High-level program to perform GRAPH-BASED QMD.
- program changecoords

  High-level program to change coordinates formats.

9.4.1 Detailed Description

9.4.2 Function/Subroutine Documentation

9.4.2.1 program changecoords ( )

High-level program to change coordinates formats.

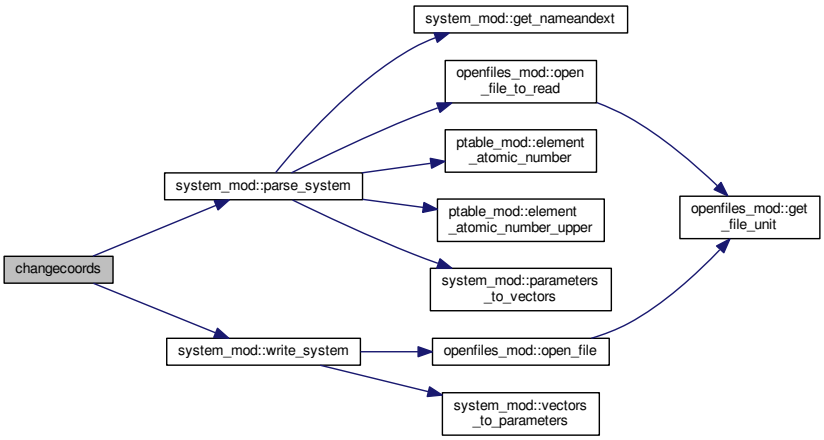
This program can be used to change coordinate formats sucha as .pdb, .xyz, .dat and .gen

Example using this code:

```
chancoords coords.xyz coords.pdb
```

Definition at line 12 of file changecoords.F90.

Here is the call graph for this function:



9.4.2.2 program gpmd ( )

High-level program to perform GRAPH-BASED QMD.

High-level program to perform GRAPH-BASED QMD with a DFTB Hamiltonian using a full parallel Graph-based approach.

Note

To test this program with the

```
run_test
```

script.

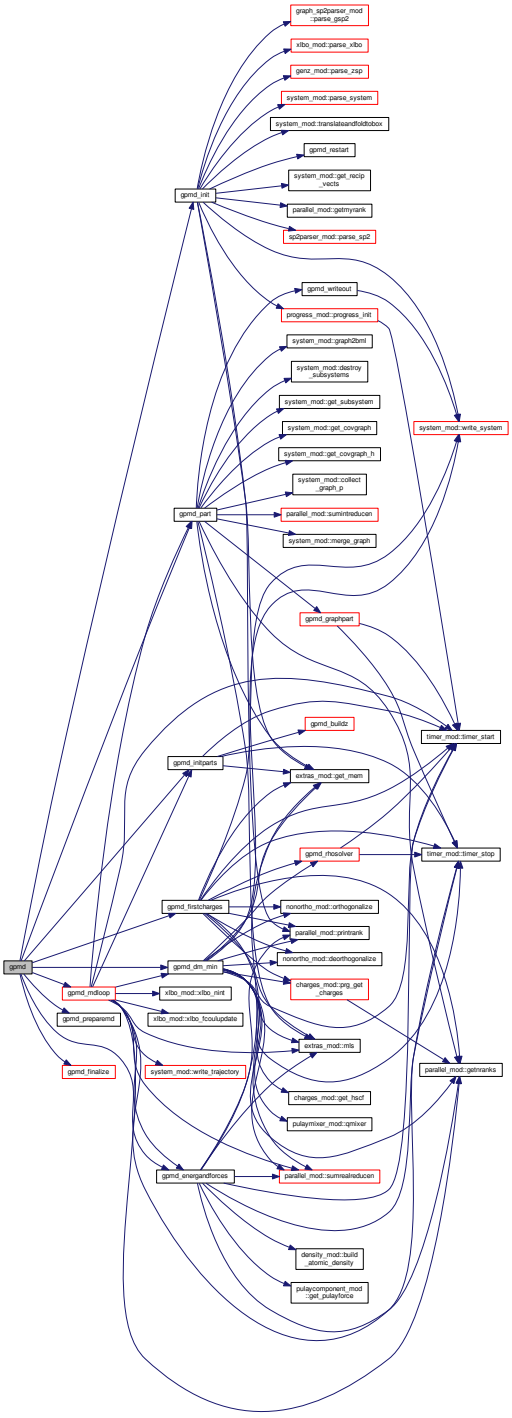
Author

C. F. A. Negre (cnegre@lanl.gov)
S. Mniszewski (smn@lanl.gov)
A. M. N. Niklasson (amn@lanl.gov)
M. E. Wall (mewall@lanl.gov)

Verbose levels: >= 0- Only print tags. >= 1- Print useful scalar physical data (e.g., Total Energy) >= 2- Print single file output data (e.g., 0-SCF Charges) >= 3- Write out trajectory data. >= 4- Write out physically meaningful 1D arrays (e.g., Charges for the species) >= 5- Write out physically meaningful 2D arrays (e.g., H matrix)

Definition at line 28 of file gpmdcov.F90.

Here is the call graph for this function:



Chapter 10

Module Documentation

10.1 get_energy Namespace Reference

Variables

- tuple MyFileName = str(sys.argv[1])
- int count = -1
- tuple MyFile = open(MyFileName,"r")
- Dim = count
- tuple datos = numpy.zeros(Dim+1)
- tuple lines_split = lines.split()

10.1.1 Variable Documentation

10.1.1.1 int get_energy.count = -1

Definition at line 8 of file get_energy.py.

10.1.1.2 tuple get_energy.datos = numpy.zeros(Dim+1)

Definition at line 15 of file get_energy.py.

10.1.1.3 get_energy.Dim = count

Definition at line 13 of file get_energy.py.

10.1.1.4 tuple get_energy.lines_split = lines.split()

Definition at line 20 of file get_energy.py.

10.1.1.5 tuple get_energy.MyFile = open(MyFileName,"r")

Definition at line 9 of file get_energy.py.

10.1.1.6 tuple get_energy.MyFileName = str(sys.argv[1])

Definition at line 5 of file get_energy.py.

10.2 test-energy Namespace Reference

Functions

- def compare_MD
- def main

10.2.1 Function Documentation

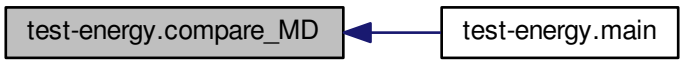
10.2.1.1 def test-energy.compare_MD (*reference, current, reltol*)

```
Compare MD energies.

Given a reference output and the current output, compare the MD
energies to within the relative tolerance given by reltol.
```

Definition at line 3 of file test-energy.py.

Here is the caller graph for this function:

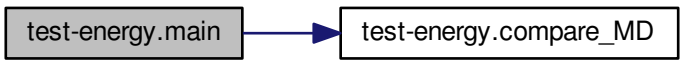


10.2.1.2 def test-energy.main ( )

```
The main function.
```

Definition at line 49 of file test-energy.py.

Here is the call graph for this function:



Chapter 11

Data Type Documentation

11.1 accuracy_mod Module Reference

Public Attributes

- integer, parameter dp = kind(1.0d0)

11.1.1 Detailed Description

Definition at line 1 of file accuracy.F90.

11.1.2 Member Data Documentation

11.1.2.1 integer, parameter accuracy_mod::dp = kind(1.0d0)

Definition at line 5 of file accuracy.F90.

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

- /home/christian/qmd-progress/tests/src/accuracy.F90

11.2 charges_mod Module Reference

A module to handle the charges of the system.

Public Member Functions

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

  Constructs the charges from the density matrix.
- subroutine, public 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, HubbadU 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.*

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.2.1 Detailed Description

A module to handle the charges of the system.

This module contains routines that computes properties related to charges.

Definition at line 5 of file charges_mod.F90.

11.2.2 Member Function/Subroutine Documentation

11.2.2.1 subroutine, public charges_mod::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*)

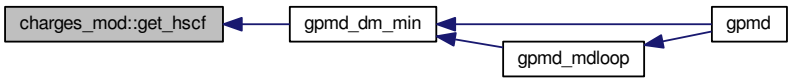
Constructs the SCF hamiltonian given H0, HubbadU and charges. This routine does: $H = \sum_i U_i q_i + V_i$;, where U is the Hubbard parameter for every atom i. V is the coulombic potential for every atom i.

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.
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 charges_mod.F90.

Here is the caller graph for this function:



11.2.2.2 subroutine, public 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.

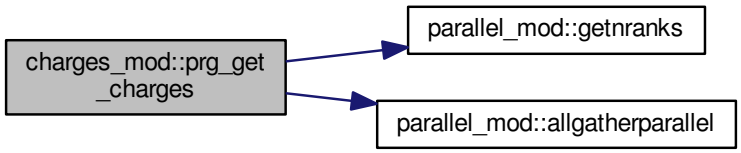
Parameters

rho_bml	Density matrix in bml format.
over_bml	Overlap matrix in bml format.

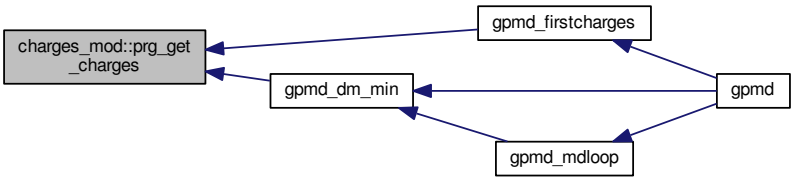
hindex	Start and end index for every atom in the system.
charges	Output parameter that gives the vectorized charges.
threshold	Threshold value for matrix elements.

Definition at line 29 of file charges_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.2.3 Member Data Documentation

11.2.3.1 integer, parameter charges_mod::dp = kind(1.0d0) [private]

Definition at line 17 of file charges_mod.F90.

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

- /home/christian/qmd-progress/src/charges_mod.F90

11.3 density_mod Module Reference

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

Public Member Functions

- subroutine, public build_density_t0 (ham_bml, rho_bml, threshold, bndfil)

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

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

- subroutine, public build_density_t_fermi (ham_bml, rho_bml, threshold, kbt, ef, verbose)

  Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \varepsilon)C^{\dagger}$ 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 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 get_flevel (eigenvalues, kbt, bndfil, tol, Ef)

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

- subroutine, public get_eigenvalues (ham_bml, eigenvalues, verbose)

  Gets the eigenvalues of the Orthogonalized Hamiltonian.

- subroutine, public check_idempotency (mat_bml, threshold, idempotency)

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

Private Member Functions

- real(dp) function fermi (e, ef, kbt)

  Gives the Fermi distribution value for energy e.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.3.1 Detailed Description

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

Definition at line 5 of file densitymatrix_mod.F90.

11.3.2 Member Function/Subroutine Documentation

11.3.2.1 subroutine, public density_mod::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*)

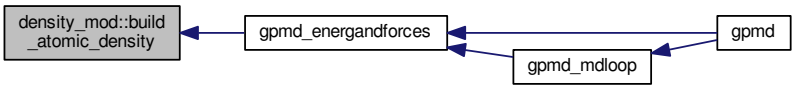
Builds the atomic density matrix. $\rho_{ii} = mathcalZ_{ii}$ Where, $mathcalZ_{ii}$ is the number of electrons for orbital i.

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 214 of file densitymatrix_mod.F90.

Here is the caller graph for this function:



11.3.2.2 subroutine, public density_mod::build_density_t (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(8), intent(in) *kbt,* real(8), intent(inout) *ef*)

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

Parameters

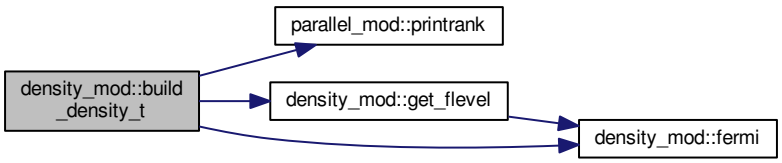
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.

Warning

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

Definition at line 93 of file densitymatrix_mod.F90.

Here is the call graph for this function:



11.3.2.3 subroutine, public density_mod::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*)

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

Parameters

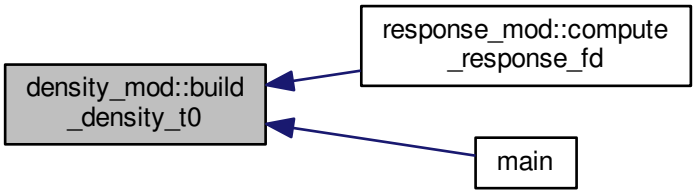
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.
bndfil	Filing factor.

Warning

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

Definition at line 32 of file densitymatrix_mod.F90.

Here is the caller graph for this function:



11.3.2.4 subroutine, public density_mod::build_density_t_fermi (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) *kbt,* real(8), intent(in) *ef,* integer, intent(in), optional *verbose*)

Builds the density matrix from H_0 for electronic temperature T. $\rho = Cf(\mu I - \varepsilon)C^{\dagger}$ 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

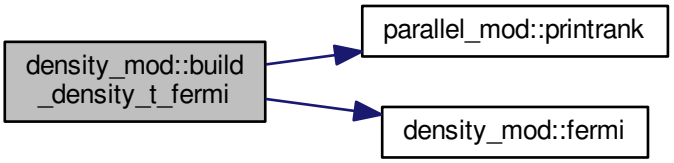
ham_bml	Input Orthogonalized Hamiltonian matrix.
rho_bml	Output density matrix,
threshold	Threshold for sparse matrix algebra.

Warning

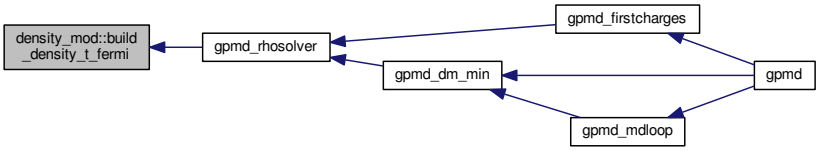
This does not solve the generalized eigenvalue problem. The Hamiltonian that comes in has to be preorthogonalized.

Definition at line 155 of file densitymatrix_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.2.5 subroutine, public density_mod::check_idempotency (type(bml_matrix_t), intent(in) *mat_bml,* real(dp), intent(in) *threshold,* real(dp), intent(out) *idempotency*)

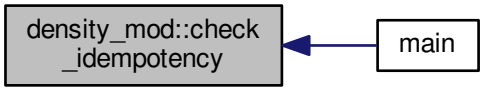
To check the idempotency error of a matrix. This is calculated as the Frobenius norm of $(A - A^2)$.

Parameters

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

Definition at line 388 of file densitymatrix_mod.F90.

Here is the caller graph for this function:



11.3.2.6 real(dp) function density_mod::fermi (real(dp), intent(in) *e,* real(dp), intent(in) *ef,* real(dp), intent(in) *kbt*) [private]

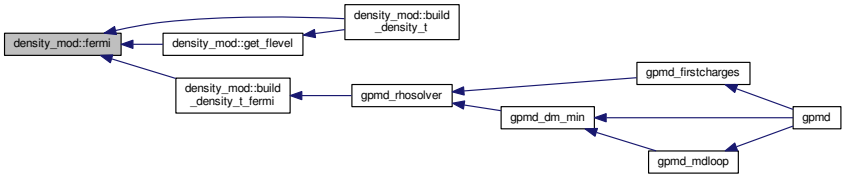
Gives the Fermi distribution value for energy e.

Parameters

e	Energy.
ef	Fermi energy.

Definition at line 413 of file densitymatrix_mod.F90.

Here is the caller graph for this function:



11.3.2.7 subroutine, public density_mod::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 343 of file densitymatrix_mod.F90.

11.3.2.8 subroutine, public density_mod::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*)

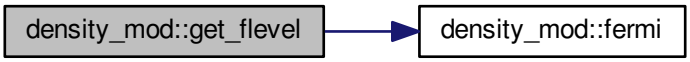
Routine to compute the Fermi level given a set of eigenvalues and a temperature. It applies the Bisection method over the function: $g(\mu) = \sum_k 2f(\varepsilon_k - \mu) - N = 0$ Where $f(\varepsilon_k - \mu) = \frac{1}{1+\exp(\varepsilon_k - \mu)/(k_b T)}$.

Parameters

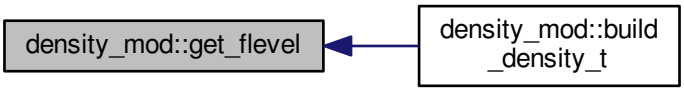
eigenvalues	Eigenvalues of the system ($\{\varepsilon_k\}$).
kbt	Temperature times the Boltzmans's constant ($k_b T$).
bndfil	Filing factor ($N_{el}/(2 * N_{orbs})$).
tol	Tolerance for the bisection method.
Ef	Fermi level (μ).

Definition at line 279 of file densitymatrix_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.3 Member Data Documentation

11.3.3.1 integer, parameter density_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file densitymatrix_mod.F90.

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

- /home/christian/qmd-progress/src/densitymatrix_mod.F90

11.4 dos_mod Module Reference

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

Public Member Functions

- subroutine, public write_tdos (eigenvals, gamma, npts, emin, emax, filename)
  Writes the total DOS into a file. $DOS(\varepsilon) = \sum_k L(\varepsilon - \varepsilon_k)$ *Where* $\int_{-\infty}^{\infty} DOS(\varepsilon) = Nstates.$

Private Member Functions

- real(dp) function lorentz (energy, eigenvals, loads, Gamma)
  Lorentzian Function.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.4.1 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.

Definition at line 8 of file dos_mod.F90.

11.4.2 Member Function/Subroutine Documentation

11.4.2.1 real(dp) function 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: $L(\varepsilon) = \sum_k \frac{\omega(k)\Gamma}{2\pi} \frac{1}{(\varepsilon-\varepsilon_k)^2+(\Gamma/2)^2}$

Parameters

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

Definition at line 77 of file dos_mod.F90.

Here is the caller graph for this function:

11.4.2.2 subroutine, public dos_mod::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. $DOS(\varepsilon) = \sum_k L(\varepsilon - \varepsilon_k)$ Where $\int_{-\infty}^{\infty} DOS(\varepsilon) = Nstates$.

Note

DOS is NOT shifted respect to Ef.

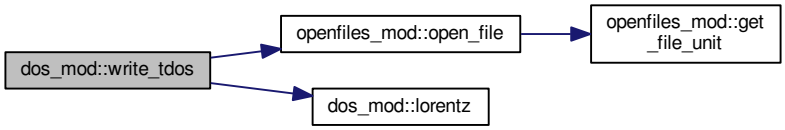
Parameters

eigenvals	Eigenvalues of the system. 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 dos_mod.F90.

Here is the call graph for this function:



11.4.3 Member Data Documentation

11.4.3.1 integer, parameter dos_mod::dp = kind(1.0d0) [private]

Definition at line 17 of file dos_mod.F90.

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

- /home/christian/qmd-progress/src/dos_mod.F90

11.5 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.

11.5.1 Detailed Description

Electronic structure type.

The electronic structure type.

Definition at line 22 of file system_mod.F90.

11.5.2 Member Data Documentation

11.5.2.1 real(dp), dimension(:), allocatable system_mod::estruct_type::coul_pot_k

Reciprocal Coulombic contribution.

Definition at line 58 of file system_mod.F90.

11.5.2.2 real(dp), dimension(:), allocatable system_mod::estruct_type::coul_pot_r

Real Coulombic contribution.

Definition at line 55 of file system_mod.F90.

11.5.2.3 real(dp) system_mod::estruct_type::eband

Band energy.

Definition at line 70 of file system_mod.F90.

11.5.2.4 real(dp), dimension(:,:), allocatable system_mod::estruct_type::fpul

Pulay force.

Definition at line 64 of file system_mod.F90.

11.5.2.5 real(dp), dimension(:,:), allocatable system_mod::estruct_type::fscoul

Nonorthogonal Coulombic force.

Definition at line 67 of file system_mod.F90.

11.5.2.6 type(bml_matrix_t) system_mod::estruct_type::ham

SCC-Hamiltonian of the system.

Definition at line 34 of file system_mod.F90.

11.5.2.7 type(bml_matrix_t) system_mod::estruct_type::ham0

Hamiltonian of the system.

Definition at line 37 of file system_mod.F90.

11.5.2.8 integer, dimension(:,:), allocatable system_mod::estruct_type::hindex

Hindex.

Definition at line 31 of file system_mod.F90.

11.5.2.9 integer system_mod::estruct_type::nel

Number of electrons.

Definition at line 28 of file system_mod.F90.

11.5.2.10 integer system_mod::estruct_type::norbs

Number of orbitals of the system.

Definition at line 25 of file system_mod.F90.

11.5.2.11 type(bml_matrix_t) system_mod::estruct_type::oham

Orthogonalized Hamiltonian.

Definition at line 40 of file system_mod.F90.

11.5.2.12 type(bml_matrix_t) system_mod::estruct_type::orho

Orthogonalized density matrix.

Definition at line 49 of file system_mod.F90.

11.5.2.13 type(bml_matrix_t) system_mod::estruct_type::over

Overlap matrix of the system.

Definition at line 43 of file system_mod.F90.

11.5.2.14 type(bml_matrix_t) system_mod::estruct_type::rho

Density matrix of the system.

Definition at line 46 of file system_mod.F90.

11.5.2.15 real(dp), dimension(:,:), allocatable system_mod::estruct_type::skforce

Slater Koster force.

Definition at line 61 of file system_mod.F90.

11.5.2.16 type(bml_matrix_t) system_mod::estruct_type::zmat

Congruence transformation.

Definition at line 52 of file system_mod.F90.

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

- /home/christian/qmd-progress/src/system_mod.F90

11.6 extras_mod Module Reference

Extra routines:

Public Member Functions

- subroutine, public 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 delta (x, s, nn, dta)

  Delta function ||X^tSX - I||. CFAN, March 2015.
- subroutine, public get_mem (procname, tag)

  Get proc memory.

Private Member Functions

- subroutine twonorm (a, nn, norm2)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.6.1 Detailed Description

Extra routines:

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

Definition at line 6 of file extras_mod.F90.

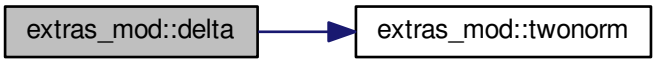
11.6.2 Member Function/Subroutine Documentation

11.6.2.1 subroutine, public extras_mod::delta (real(dp), dimension(nn,nn) *x,* real(dp), dimension(nn,nn) *s,* integer *nn,* real(dp) *dta*)

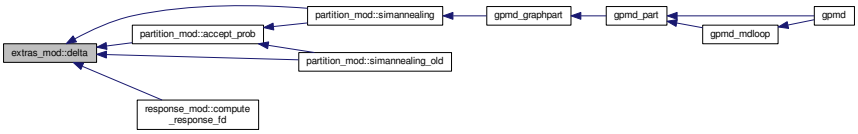
Delta function ||X^tSX - I||. CFAN, March 2015.

Definition at line 80 of file extras_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.6.2.2 subroutine, public extras_mod::get_mem (character(∗), intent(in) *procname,* character(∗), intent(in) *tag*)

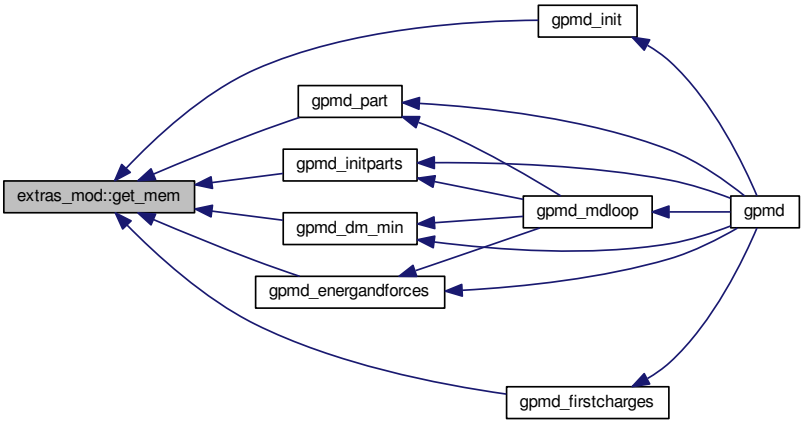
Get proc memory.

Parameters

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

Definition at line 116 of file extras_mod.F90.

Here is the caller graph for this function:



11.6.2.3 real(dp) function, public extras_mod::mls ( )

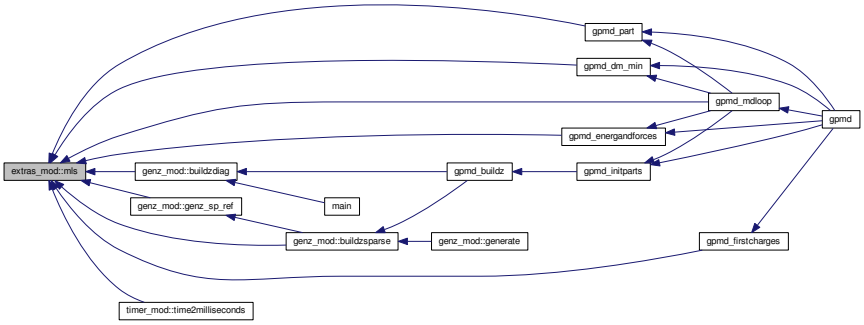
To get the actual time in milliseconds.

Parameters

mls	Output value with the machine time in milliseconds.

Definition at line 67 of file extras_mod.F90.

Here is the caller graph for this function:



11.6.2.4 subroutine, public extras_mod::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

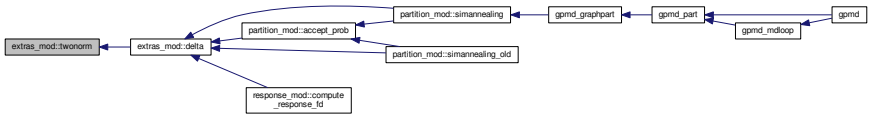
matname	Matrix name.
amat	Matrix to be printed.
i1	Print from row i1.
i2	Print up to from row i2.
j1	Print from column j1.
j2	Print up to column j2.

Definition at line 28 of file extras_mod.F90.

11.6.2.5 subroutine extras_mod::twonorm (real(dp), dimension(nn,nn) *a,* integer *nn,* real(dp) *norm2*) [private]

Definition at line 129 of file extras_mod.F90.

Here is the caller graph for this function:



11.6.3 Member Data Documentation

11.6.3.1 integer, parameter extras_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file extras_mod.F90.

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

- /home/christian/qmd-progress/src/extras_mod.F90

11.7 genz_mod Module Reference

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

$H_{orth} = Z^{\dagger} H Z$ Please see Negre 2016 **?**.

Data Types

- type genzspdata

  contains the data for the genZ driver.
- type genzspinp

  Input for the genz driver.

Public Member Functions

- subroutine, public parse_zsp (input, filename)

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

  Initiates the matrices for the Xl integration of Z.

- subroutine, public buildzdiag (smat_bml, zmat_bml, threshold, mdimin, bml_type, verbose)

  Usual subroutine involving diagonalization.
- subroutine, public 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 genz_sp_initial_zmat (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)

  Estimate Z matrix.
- subroutine, public genz_sp_ref (smat_bml, zmat_bml, nref, norb, bml_type, threshold)

Private Member Functions

- subroutine init (self, input)

  Initializes the genz input variables.
- subroutine allocatezspmat (self, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, norb, bml_type)

  Allocates the matrices for the Xl integration of Z.
- subroutine generate (self, over_bml, zmat_bml, igenz, mdim, bml_type, zk1_bml, zk2_bml, zk3_bml, zk4_-bml, zk5_bml, zk6_bml)

  Generates the Z matrix.
- subroutine genz_sp_initialz0 (smat_bml, zmat_bml, norb, mdim, bml_type_f, threshold)
- subroutine genz_sp_int (zmat_bml, zk1_bml, zk2_bml, zk3_bml, zk4_bml, zk5_bml, zk6_bml, igenz, norb, bml_type, threshold)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.7.1 Detailed Description

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

$H_{orth} = Z^{\dagger} H Z$ Please see Negre 2016 **?**.

Author

C. F. A. Negre (cnegre@lanl.gov)

Definition at line 12 of file genz_mod.F90.

11.7.2 Member Function/Subroutine Documentation

11.7.2.1 subroutine genz_mod::allocatezspmat (class(genzspdata), intent(in) *self,* 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*) [private]

Allocates the matrices for the XI 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 163 of file genz_mod.F90.

11.7.2.2 subroutine, public genz_mod::buildzdiag (type(bml_matrix_t), intent(inout) *smat_bml,* type(bml_matrix_t) *zmat_bml,* real(dp) *threshold,* integer, intent(in) *mdimin,* character(len=∗) *bml_type,* integer, intent(in), optional *verbose*)

Usual subroutine involving diagonalization.

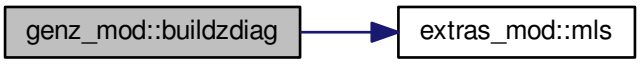
$Z = U\sqrt{s}U^{\dagger}$, where U = eigenvectors and s = eigenvalues. The purpose of this subroutine is to have an exact way of computing z for comparing with the sparse approach.

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 backtranformation to ellpack format.
mdim	Maximun nonzero to use, in this case, only in the backtranformation to ellpack format.
bml_type	the bml type we are passing.

Definition at line 271 of file genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

11.7.2.3 subroutine, public genz_mod::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*)

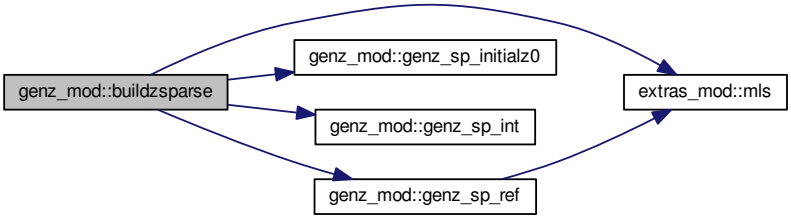
Inverse factorization using niklasson's algorithm.

Parameters

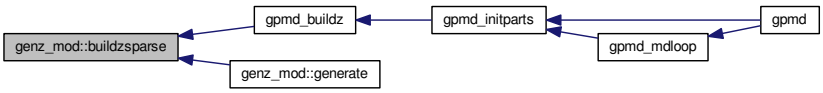
smat_bml	overlap matrix
zmat_bml	congruence transform to be updated or computed. (bml format)
igenz	counter to keep track of the calls to this subroutine.
mdim	dimension of the maxnonzero per row.
zk1_bml-zk6_-bml	history of the past congruence transforms.
nfirst	first pre septs with nrefi and thresholdi.
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 408 of file genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.7.2.4 subroutine genz_mod::generate (class(genzspdata), intent(in) *self,* type(bml_matrix_t), intent(in) *over_bml,* type(bml_matrix_t), intent(inout) *zmat_bml,* integer *igenz,* integer *mdim,* character(20) *bml_type,* type(bml_matrix_t), intent(inout) *zk1_bml,* type(bml_matrix_t), intent(inout) *zk2_bml,* type(bml_matrix_t), intent(inout) *zk3_bml,* type(bml_matrix_t), intent(inout) *zk4_bml,* type(bml_matrix_t), intent(inout) *zk5_bml,* type(bml_matrix_t), intent(inout) *zk6_bml*) [private]

Generates the Z matrix.

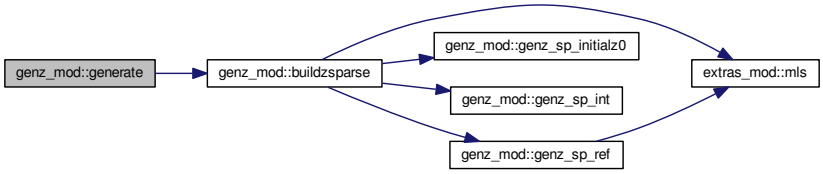
Parameters

over_bml	Overlap matrix.
zmat_bml	Congruence transform to be 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.

Definition at line 230 of file genz_mod.F90.

Here is the call graph for this function:



11.7.2.5 subroutine, public genz_mod::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*)

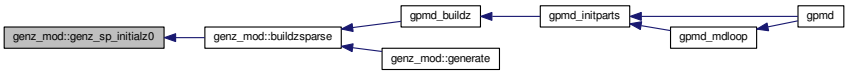
Estimate Z matrix.

Definition at line 587 of file genz_mod.F90.

11.7.2.6 subroutine genz_mod::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*) [private]

Definition at line 466 of file genz_mod.F90.

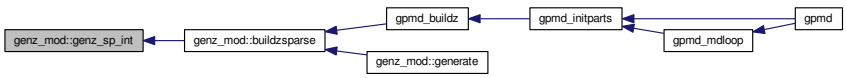
Here is the caller graph for this function:



11.7.2.7 subroutine genz_mod::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]

Definition at line 725 of file genz_mod.F90.

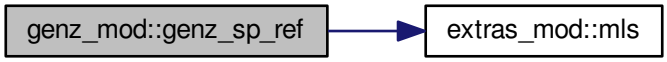
Here is the caller graph for this function:



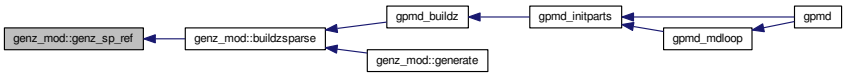
11.7.2.8 subroutine, public genz_mod::genz_sp_ref (type(bml_matrix_t), intent(in) *smat_bml,* type(bml_matrix_t), intent(inout) *zmat_bml,* integer, intent(in) *nref,* integer, intent(inout) *norb,* character(20), intent(in) *bml_type,* real(dp), intent(in) *threshold*)

Definition at line 793 of file genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.7.2.9 subroutine genz_mod::init (class(genzspdata), intent(out) *self,* type(genzspinp), intent(in) *input*) [private]

Initializes the genz input variables.

Parameters

self	basic input parameters.
input	basic input parameters from the parser.

Definition at line 142 of file genz_mod.F90.

11.7.2.10 subroutine, public genz_mod::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*)

Initiates the matrices for the XI 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.

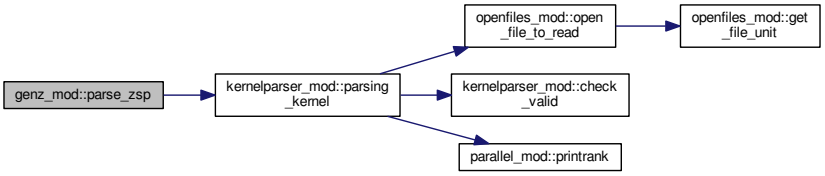
Definition at line 196 of file genz_mod.F90.

11.7.2.11 subroutine, public genz_mod::parse_zsp (type(genzspinp), intent(inout) *input,* character(len=*) *filename*)

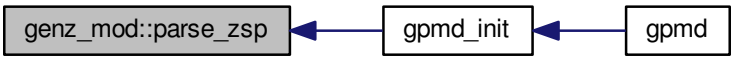
The parser for md solver.

Definition at line 84 of file genz_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.7.3 Member Data Documentation

11.7.3.1 integer, parameter genz_mod::dp = kind(1.0d0) [private]

Definition at line 24 of file genz_mod.F90.

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

- /home/christian/qmd-progress/src/genz_mod.F90

11.8 genz_mod::genzspdata Type Reference

contains the data for the genZ driver.

Public Member Functions

- procedure init
- procedure generate
- procedure allocatezspmat

Public Attributes

- integer verbose
- integer nfirst

- integer nrefi
- integer nreff
- real(dp) numthresi
- real(dp) numthresf
- logical integration

11.8.1 Detailed Description

contains the data for the genZ driver.

Definition at line 64 of file genz_mod.F90.

11.8.2 Member Function/Subroutine Documentation

11.8.2.1 procedure genz_mod::genzspdata::allocatezspmat ( )

Definition at line 76 of file genz_mod.F90.

11.8.2.2 procedure genz_mod::genzspdata::generate ( )

Definition at line 75 of file genz_mod.F90.

11.8.2.3 procedure genz_mod::genzspdata::init ( )

Definition at line 74 of file genz_mod.F90.

11.8.3 Member Data Documentation

11.8.3.1 logical genz_mod::genzspdata::integration

Definition at line 72 of file genz_mod.F90.

11.8.3.2 integer genz_mod::genzspdata::nfirst

Definition at line 67 of file genz_mod.F90.

11.8.3.3 integer genz_mod::genzspdata::nreff

Definition at line 69 of file genz_mod.F90.

11.8.3.4 integer genz_mod::genzspdata::nrefi

Definition at line 68 of file genz_mod.F90.

11.8.3.5 real(dp) genz_mod::genzspdata::numthresf

Definition at line 71 of file genz_mod.F90.

11.8.3.6 real(dp) genz_mod::genzspdata::numthresi

Definition at line 70 of file genz_mod.F90.

11.8.3.7 integer genz_mod::genzspdata::verbose

Definition at line 66 of file genz_mod.F90.

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

- /home/christian/qmd-progress/src/genz_mod.F90

11.9 genz_mod::genzspinp Type Reference

Input for the genz driver.

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

11.9.1 Detailed Description

Input for the genz driver.

This type controlls all the variables that are needed by genz

Definition at line 32 of file genz_mod.F90.

11.9.2 Member Data Documentation

11.9.2.1 integer genz_mod::genzspinp::igenz

To keep track of the genz iterations.

Definition at line 56 of file genz_mod.F90.

11.9.2.2 logical genz_mod::genzspinp::integration

If we want to do XL integration scheme for Z.

Definition at line 53 of file genz_mod.F90.

11.9.2.3 integer genz_mod::genzspinp::nfirst

!Lentgth of the "firsts iteration period".

Definition at line 38 of file genz_mod.F90.

11.9.2.4 integer genz_mod::genzspinp::nreff

!Initial number of recursive refinements.

Definition at line 44 of file genz_mod.F90.

11.9.2.5 integer genz_mod::genzspinp::nrefi

!Initial number of recursive refinements.

Definition at line 41 of file genz_mod.F90.

11.9.2.6 real(dp) genz_mod::genzspinp::numthresf

Final threshold value.

Definition at line 50 of file genz_mod.F90.

11.9.2.7 real(dp) genz_mod::genzspinp::numthresi

Initial threshold value.

Definition at line 47 of file genz_mod.F90.

11.9.2.8 integer genz_mod::genzspinp::verbose

To have different levels of verbose.

Definition at line 35 of file genz_mod.F90.

11.9.2.9 logical genz_mod::genzspinp::zsp

Definition at line 58 of file genz_mod.F90.

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

- /home/christian/qmd-progress/src/genz_mod.F90

11.10 graph_mod Module Reference

The graph module.

Data Types

- type graph_partitioning_t

  Trace per iteration.
- type subgraph_t

  Subgraph type.

Public Member Functions

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

  Initialize subgraph.
- subroutine, public destroysubgraph (sg)

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

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

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

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

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

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

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

  Accumulate trace norm across all subgraphs.

Private Member Functions

- subroutine readpart (gp, partFile)

  Read parts (core) from part file.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.10.1 Detailed Description

The graph module.

Definition at line 27 of file graph_mod.F90.

11.10.2 Member Function/Subroutine Documentation

11.10.2.1 subroutine, public graph_mod::destroygraphpartitioning (type (graph_partitioning_t), intent(inout) *gp*)

Destroy graph partitioning.

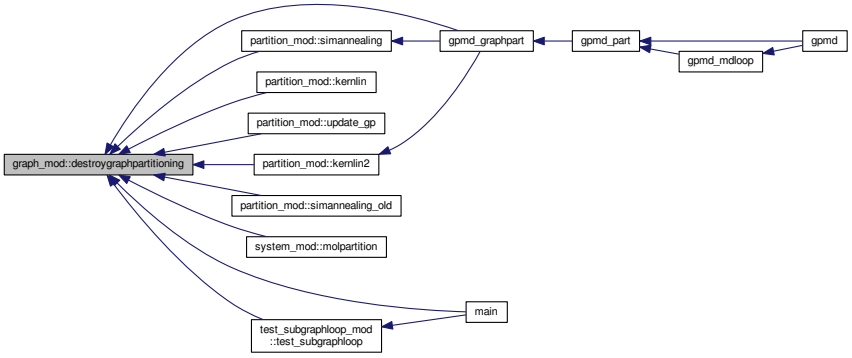
Parameters

sg	Subgraph

Definition at line 283 of file graph_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:



11.10.2.2 subroutine, public graph_mod::destroysubgraph (type (subgraph_t), intent(inout) *sg*)

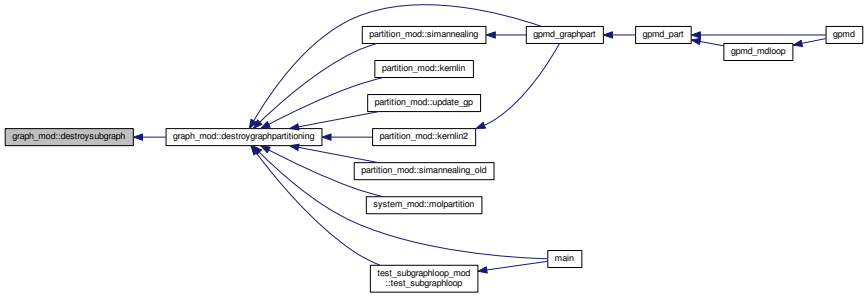
Destroy subgraph.

Parameters

sg	Subgraph

Definition at line 179 of file graph_mod.F90.

Here is the caller graph for this function:



11.10.2.3 subroutine, public graph_mod::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*)

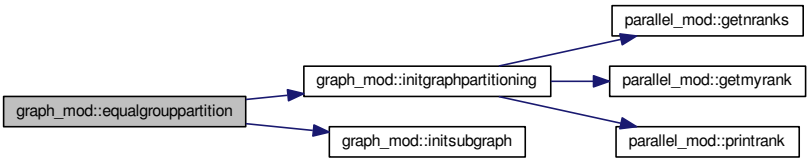
Create equal group graph partitions, based on number of atoms/groups.

Parameters

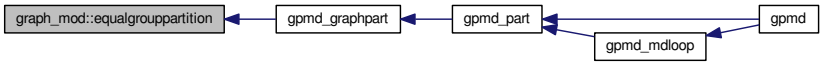
gp	Graph partitioning
hindex	Node indeces that represent ranges of atoms/groups
ngroup	Number of group nodes
nodesPerPart	Number of core nodes per partition
nnodes	Total nodes in Hamiltonian matrix

Definition at line 422 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.10.2.4 subroutine, public graph_mod::equalpartition (type (graph_partitioning_t), intent(inout) *gp,* integer, intent(in) *nodesPerPart,* integer, intent(in) *nnodes*)

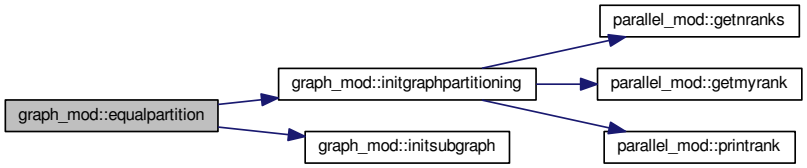
Create equal graph partitions, based on number of rows/orbitals.

Parameters

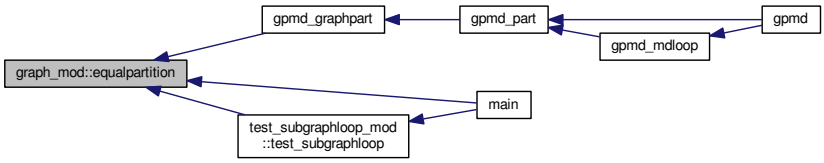
gp	Graph partitioning‘
nodesPerPart	Number of core nodes per partition
nnodes	Total nodes in Hamiltonian matrix

Definition at line 375 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.10.2.5 subroutine, public graph_mod::filepartition (type (graph_partitioning_t), intent(inout) *gp,* character(len=∗), intent(in) *partFile*)

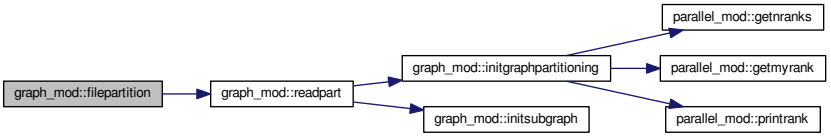
Read graph partitions from a file, based on number of rows/orbitals.

Parameters

partFile	File containing core nodes for each partition
gp	Graph partitioning

Definition at line 483 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

11.10.2.6 subroutine, public graph_mod::fnormgraph (type(graph_partitioning_t), intent(inout) *gp*)

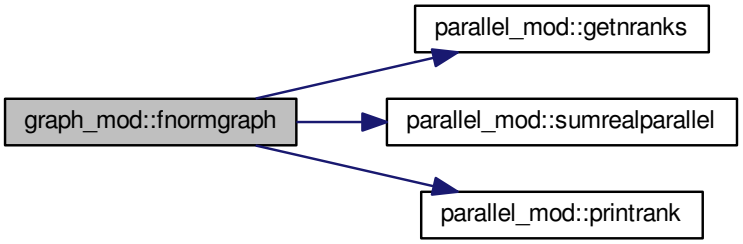
Accumulate trace norm across all subgraphs.

Parameters

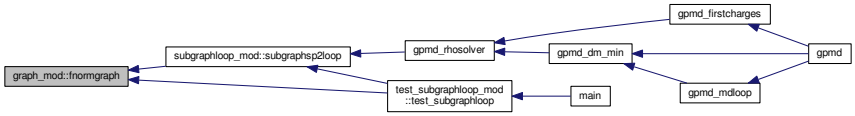
gp	Graph partitioning

Definition at line 536 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.10.2.7 subroutine, public graph_mod::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*)

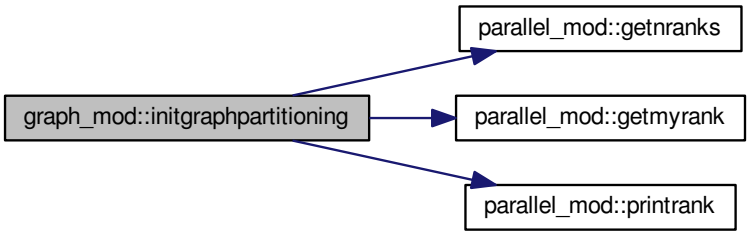
Initialize graph partitioning.

Parameters

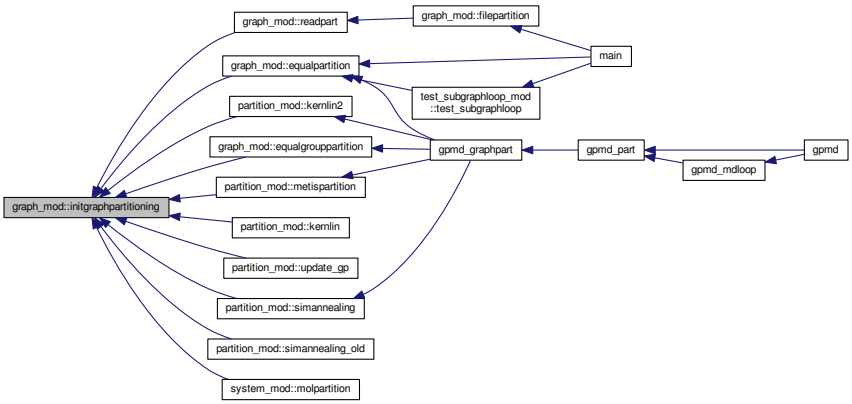
gp	Graph partitioning
pname	Partitioning name
np	Number of partitions
nnodes	Number of groups/nodes
nnodes2	Number of nodes

Definition at line 195 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.10.2.8 subroutine, public graph_mod::initsubgraph (type (subgraph_t), intent(inout) *sg,* integer, intent(in) *pnum,* integer, intent(in) *hsize*)

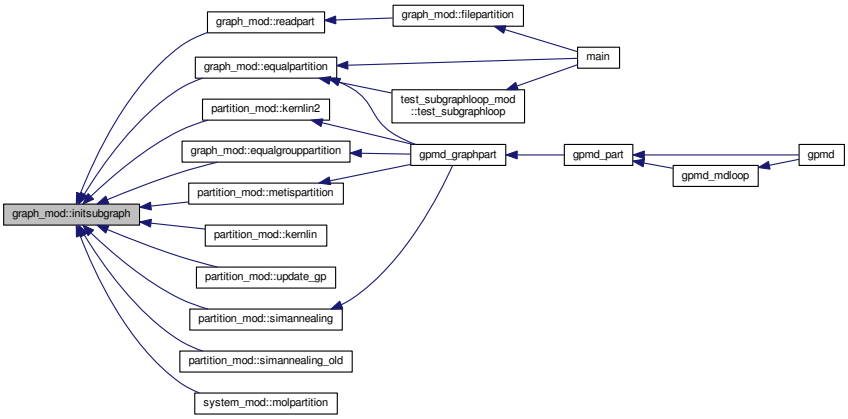
Initialize subgraph.

Parameters

sg	Subgraph
pnum	Part number
hsize	Size of full matrix

Definition at line 163 of file graph_mod.F90.

Here is the caller graph for this function:



11.10.2.9 subroutine, public graph_mod::printgraphpartitioning (type (graph_partitioning_t), intent(in) *gp*)

Print graph partitioning structure data.

Parameters

gp	Graph partitioning

Definition at line 312 of file graph_mod.F90.

Here is the caller graph for this function:

11.10.2.10 subroutine graph_mod::readpart (type (graph_partitioning_t), intent(inout) *gp*, character(len=∗), intent(in) *partFile*) [private]

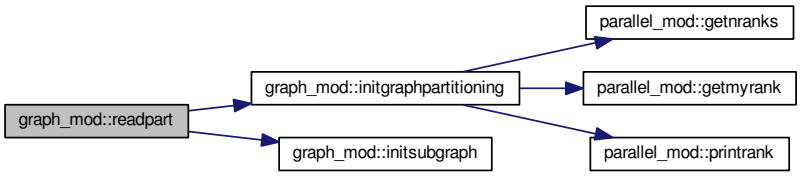
Read parts (core) from part file.

Parameters

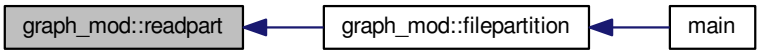
gp	Graph partitioning
partFile	Partition file

Definition at line 495 of file graph_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.10.3 Member Data Documentation

11.10.3.1 integer, parameter graph_mod::dp = kind(1.0d0) [private]

Definition at line 37 of file graph_mod.F90.

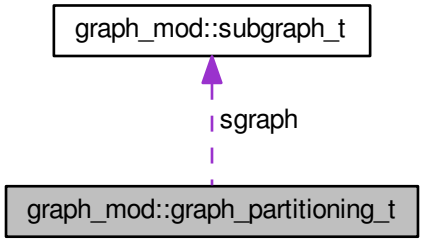
The documentation for this module was generated from the following file:

- /home/christian/qmd-progress/src/graph_mod.F90

11.11 graph_mod::graph_partitioning_t Type Reference

Trace per iteration.

Collaboration diagram for 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 normalize.
- real(dp) maxeval

  Max eval for normalize.
- real(dp), dimension(100) vv

  Trace per iteration.
- type(subgraph_t), dimension(:), allocatable sgraph

  Subgraph details.

11.11.1 Detailed Description

Trace per iteration.

Graph partitioning type

Definition at line 78 of file graph_mod.F90.

11.11.2 Member Data Documentation

11.11.2.1 real(dp) graph_mod::graph_partitioning_t::ehomo [private]

Homo value.

Definition at line 138 of file graph_mod.F90.

11.11.2.2 real(dp) graph_mod::graph_partitioning_t::elumo [private]

Lumo value.

Definition at line 141 of file graph_mod.F90.

11.11.2.3 integer graph_mod::graph_partitioning_t::globalpartextent [private]

Total global parts.

Definition at line 105 of file graph_mod.F90.

11.11.2.4 integer graph_mod::graph_partitioning_t::globalpartmax [private]

Maximum global part number.

Definition at line 102 of file graph_mod.F90.

11.11.2.5 integer graph_mod::graph_partitioning_t::globalpartmin [private]

Minimum global part number.

Definition at line 99 of file graph_mod.F90.

11.11.2.6 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::localpartextent [private]

Number of parts per processor.

Definition at line 114 of file graph_mod.F90.

11.11.2.7 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::localpartmax [private]

Maximum part per processor.

Definition at line 111 of file graph_mod.F90.

11.11.2.8 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::localpartmin [private]

Minimum part per processor.

Definition at line 108 of file graph_mod.F90.

11.11.2.9 real(dp) graph_mod::graph_partitioning_t::maxeval [private]

Max eval for normalize.

Definition at line 147 of file graph_mod.F90.

11.11.2.10 integer graph_mod::graph_partitioning_t::maxiter [private]

Number of SP2 iterations.

Definition at line 135 of file graph_mod.F90.

11.11.2.11 real(dp) graph_mod::graph_partitioning_t::mineval [private]

Min eval for normalize.

Definition at line 144 of file graph_mod.F90.

11.11.2.12 integer graph_mod::graph_partitioning_t::myrank [private]

Local processor.

Definition at line 84 of file graph_mod.F90.

11.11.2.13 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::nnodesinpart [private]

Number of nodes in each local partition.

Definition at line 126 of file graph_mod.F90.

11.11.2.14 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::nnodesinpartall [private]

Number of nodes in each partition.

Definition at line 129 of file graph_mod.F90.

11.11.2.15 integer graph_mod::graph_partitioning_t::nparts [private]

Total number of local partitions.

Definition at line 123 of file graph_mod.F90.

11.11.2.16 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::order [private]

Original ordering if required.

Definition at line 117 of file graph_mod.F90.

11.11.2.17 character(len=100) graph_mod::graph_partitioning_t::pname [private]

Partition name.

Definition at line 81 of file graph_mod.F90.

11.11.2.18 integer, dimension(100) graph_mod::graph_partitioning_t::pp [private]

Sequence for SP2.

Definition at line 132 of file graph_mod.F90.

11.11.2.19 integer, dimension(:), allocatable graph_mod::graph_partitioning_t::reorder [private]

Reordering if required.

Definition at line 120 of file graph_mod.F90.

11.11.2.20 type (subgraph_t), dimension(:), allocatable graph_mod::graph_partitioning_t::sgraph [private]

Subgraph details.

Definition at line 153 of file graph_mod.F90.

11.11.2.21 integer graph_mod::graph_partitioning_t::totalnodes [private]

Total number of global groups, nodes (or matrix rows)

Definition at line 93 of file graph_mod.F90.

11.11.2.22 integer graph_mod::graph_partitioning_t::totalnodes2 [private]

Total number of global nodes (or matrix rows)

Definition at line 96 of file graph_mod.F90.

11.11.2.23 integer graph_mod::graph_partitioning_t::totalparts [private]

Total number of global partitions.

Definition at line 90 of file graph_mod.F90.

11.11.2.24 integer graph_mod::graph_partitioning_t::totalprocs [private]

Number of processors.

Definition at line 87 of file graph_mod.F90.

11.11.2.25 real(dp), dimension(100) graph_mod::graph_partitioning_t::vv [private]

Trace per iteration.

Definition at line 150 of file graph_mod.F90.

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

- /home/christian/qmd-progress/src/graph_mod.F90

11.12 graph_sp2parser_mod Module Reference

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:

Data Types

- type gsp2data_type

  General SP2 solver type.

Public Member Functions

- subroutine, public parse_gsp2 (gsp2data, filename)

  The parser for SP2 solver.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.12.1 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.

Definition at line 12 of file graph_sp2parser_mod.F90.

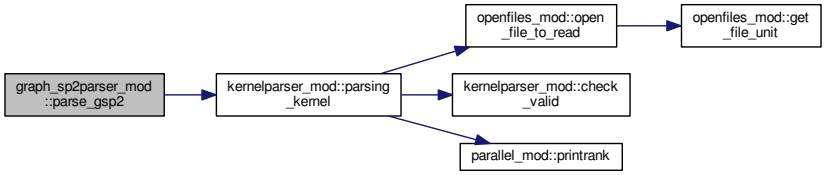
11.12.2 Member Function/Subroutine Documentation

11.12.2.1 subroutine, public graph_sp2parser_mod::parse_gsp2 (type(gsp2data_type), intent(inout) *gsp2data,* character(len=∗) *filename*)

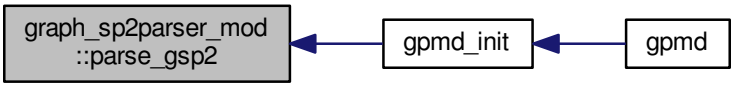
The parser for SP2 solver.

Definition at line 61 of file graph_sp2parser_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.12.3 Member Data Documentation

11.12.3.1 integer, parameter graph_sp2parser_mod::dp = kind(1.0d0) [private]

Definition at line 22 of file graph_sp2parser_mod.F90.

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

- /home/christian/qmd-progress/src/graph_sp2parser_mod.F90

11.13 graph_sp2parser_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

11.13.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file graph_sp2parser_mod.F90.

11.13.2 Member Data Documentation

11.13.2.1 character(20) graph_sp2parser_mod::gsp2data_type::bml_type

Definition at line 44 of file graph_sp2parser_mod.F90.

11.13.2.2 real(dp) graph_sp2parser_mod::gsp2data_type::bndfil

Definition at line 37 of file graph_sp2parser_mod.F90.

11.13.2.3 real(dp) graph_sp2parser_mod::gsp2data_type::covgfact

Definition at line 50 of file graph_sp2parser_mod.F90.

11.13.2.4 logical graph_sp2parser_mod::gsp2data_type::double_jump

Definition at line 49 of file graph_sp2parser_mod.F90.

11.13.2.5 real(dp) graph_sp2parser_mod::gsp2data_type::errlimit

Definition at line 39 of file graph_sp2parser_mod.F90.

11.13.2.6 character(10) graph_sp2parser_mod::gsp2data_type::graph_element

Definition at line 46 of file graph_sp2parser_mod.F90.

11.13.2.7 real(dp) graph_sp2parser_mod::gsp2data_type::gthreshold

Definition at line 38 of file graph_sp2parser_mod.F90.

11.13.2.8 character(50) graph_sp2parser_mod::gsp2data_type::hamfile

Definition at line 28 of file graph_sp2parser_mod.F90.

11.13.2.9 character(20) graph_sp2parser_mod::gsp2data_type::jobname

Definition at line 27 of file graph_sp2parser_mod.F90.

11.13.2.10 integer graph_sp2parser_mod::gsp2data_type::maxsp2iter

Definition at line 31 of file graph_sp2parser_mod.F90.

11.13.2.11 integer graph_sp2parser_mod::gsp2data_type::mdim

Definition at line 40 of file graph_sp2parser_mod.F90.

11.13.2.12 integer graph_sp2parser_mod::gsp2data_type::minsp2iter

Definition at line 30 of file graph_sp2parser_mod.F90.

11.13.2.13 integer graph_sp2parser_mod::gsp2data_type::natoms

Definition at line 33 of file graph_sp2parser_mod.F90.

11.13.2.14 integer graph_sp2parser_mod::gsp2data_type::ndim

Definition at line 41 of file graph_sp2parser_mod.F90.

11.13.2.15 real(dp) graph_sp2parser_mod::gsp2data_type::nlgcut

Definition at line 51 of file graph_sp2parser_mod.F90.

11.13.2.16 integer graph_sp2parser_mod::gsp2data_type::nodesperpart

Definition at line 32 of file graph_sp2parser_mod.F90.

11.13.2.17 integer graph_sp2parser_mod::gsp2data_type::parteach

Definition at line 52 of file graph_sp2parser_mod.F90.

11.13.2.18 integer graph_sp2parser_mod::gsp2data_type::partition_count

Definition at line 34 of file graph_sp2parser_mod.F90.

11.13.2.19 character(10) graph_sp2parser_mod::gsp2data_type::partition_refinement

Definition at line 48 of file graph_sp2parser_mod.F90.

11.13.2.20 character(10) graph_sp2parser_mod::gsp2data_type::partition_type

Definition at line 47 of file graph_sp2parser_mod.F90.

11.13.2.21 real(dp), dimension(3) graph_sp2parser_mod::gsp2data_type::pdim

Definition at line 43 of file graph_sp2parser_mod.F90.

11.13.2.22 character, dimension(3) graph_sp2parser_mod::gsp2data_type::sdim

Definition at line 42 of file graph_sp2parser_mod.F90.

11.13.2.23 character(10) graph_sp2parser_mod::gsp2data_type::sp2conv

Definition at line 45 of file graph_sp2parser_mod.F90.

11.13.2.24 real(dp) graph_sp2parser_mod::gsp2data_type::sp2tol

Definition at line 35 of file graph_sp2parser_mod.F90.

11.13.2.25 real(dp) graph_sp2parser_mod::gsp2data_type::threshold

Definition at line 36 of file graph_sp2parser_mod.F90.

11.13.2.26 integer graph_sp2parser_mod::gsp2data_type::verbose

Definition at line 29 of file graph_sp2parser_mod.F90.

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

- /home/christian/qmd-progress/src/graph_sp2parser_mod.F90

11.14 hamiltonian_mod Module Reference

Public Member Functions

- subroutine, public h_read (ham, hdim)
- subroutine, public read_matrix (mat, hdim, filename)

11.14.1 Detailed Description

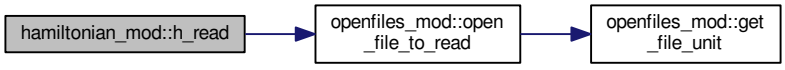
Definition at line 2 of file hamiltonian.F90.

11.14.2 Member Function/Subroutine Documentation

11.14.2.1 subroutine, public hamiltonian_mod::h_read (real(dp), dimension(:,:), allocatable *ham,* integer, intent(out) *hdim*)

Definition at line 18 of file hamiltonian.F90.

Here is the call graph for this function:

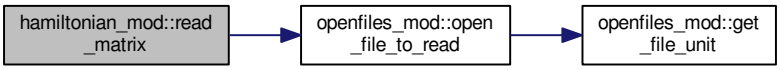


Here is the caller graph for this function:

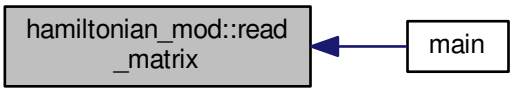
11.14.2.2 subroutine, public hamiltonian_mod::read_matrix (real(dp), dimension(:,:), allocatable *mat,* integer, intent(out) *hdim,* character(len=∗) *filename*)

Definition at line 58 of file hamiltonian.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /home/christian/qmd-progress/tests/src/hamiltonian.F90

11.15 homolumo_mod Module Reference

The homolumo module.

Public Member Functions

- subroutine, public homolumogap (vv, imax, pp, mineval, maxeval, ehomo, elumo, egap, verbose)
- subroutine, public sp2sequence (pp, imax, mineval, maxeval, ehomo, elumo, errlimit, verbose)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.15.1 Detailed Description

The homolumo module.

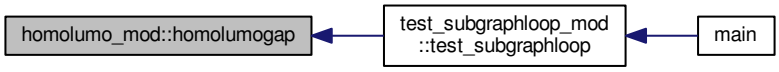
Definition at line 27 of file homolumo_mod.F90.

11.15.2 Member Function/Subroutine Documentation

11.15.2.1 subroutine, public homolumo_mod::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 43 of file homolumo_mod.F90.

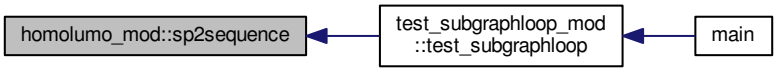
Here is the caller graph for this function:



11.15.2.2 subroutine, public homolumo_mod::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 114 of file homolumo_mod.F90.

Here is the caller graph for this function:



11.15.3 Member Data Documentation

11.15.3.1 integer, parameter homolumo_mod::dp = kind(1.0d0) [private]

Definition at line 35 of file homolumo_mod.F90.

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

- /home/christian/qmd-progress/src/homolumo_mod.F90

11.16 initmatrices_mod Module Reference

Initialization module.

Public Member Functions

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

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

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

  Initialize The orthogonal versions of Hamiltonian and Density Matrix.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.16.1 Detailed Description

Initialization module.

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

Definition at line 6 of file initmatrices_mod.F90.

11.16.2 Member Function/Subroutine Documentation

11.16.2.1 subroutine, public initmatrices_mod::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 **?** .
norb	Total number of orbitals.

Definition at line 27 of file initmatrices_mod.F90.

11.16.2.2 subroutine, public initmatrices_mod::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 **?** .
norb	Total number of orbitals.

Definition at line 69 of file initmatrices_mod.F90.

11.16.2.3 subroutine, public initmatrices_mod::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 **?** .
norb	Total number of orbitals.

Definition at line 48 of file initmatrices_mod.F90.

11.16.3 Member Data Documentation

11.16.3.1 integer, parameter initmatrices_mod::dp = kind(1.0d0) [private]

Definition at line 14 of file initmatrices_mod.F90.

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

- /home/christian/qmd-progress/src/initmatrices_mod.F90

11.17 kernelparser_mod Module Reference

Some general parsing functions.

Public Member Functions

- subroutine, public 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 parsing routine.

Private Member Functions

- subroutine check_valid (invalidc)

  Check for valid keywords (checks for an = sign)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.17.1 Detailed Description

Some general parsing functions.

Author

C. F. A. Negre (cnegre@lanl.gov)

Definition at line 7 of file kernelparser_mod.F90.

11.17.2 Member Function/Subroutine Documentation

11.17.2.1 subroutine kernelparser_mod::check_valid (character(len=*), intent(in) *invalidc*) [private]

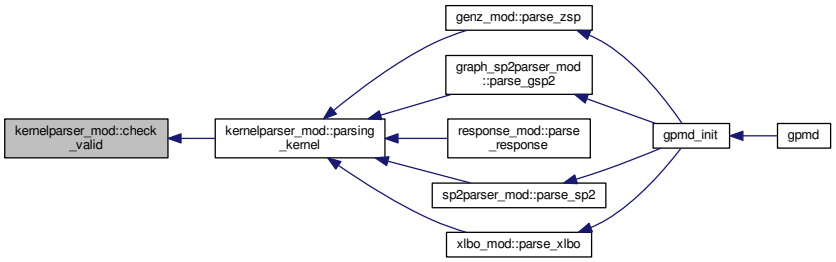
Check for valid keywords (checks for an = sign)

Parameters

invalidc	Keyword to check.

Definition at line 357 of file kernelparser_mod.F90.

Here is the caller graph for this function:



11.17.2.2 subroutine, public kernelparser_mod::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 parsing routine.

Note

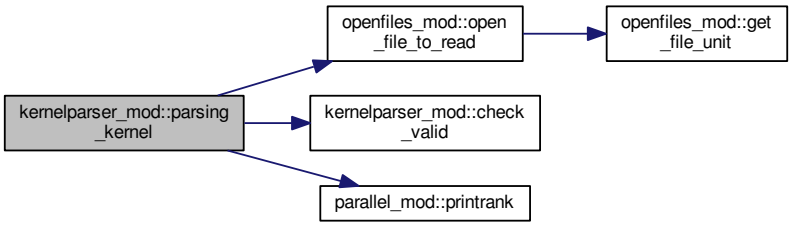
This parsing strategy can only parse a file of 500 lines by 500 words.

Warning

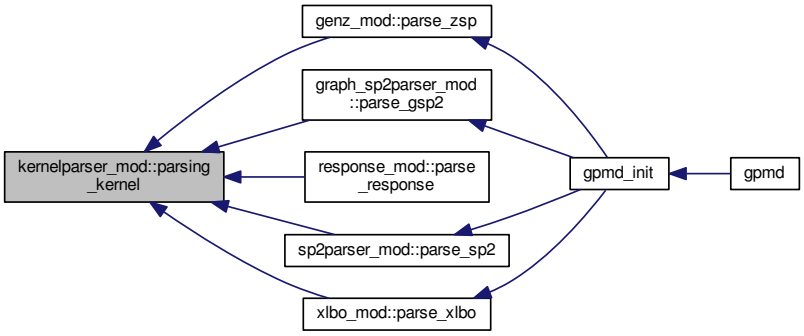
If the length of variable vect is changed, this could produce a segmentation fault.

Definition at line 30 of file kernelparser_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.17.3 Member Data Documentation

11.17.3.1 integer, parameter kernelparser_mod::dp = kind(1.0d0) [private]

Definition at line 16 of file kernelparser_mod.F90.

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

- /home/christian/qmd-progress/src/kernelparser_mod.F90

11.18 nonortho_mod Module Reference

Module to orthogonalize and deorthogonalize any operator.

Typically the Hamiltonin needs to be orthogonalized: $H_{\text{ortho}} = Z^{\dagger} H Z$.

Public Member Functions

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

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

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

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.18.1 Detailed Description

Module to orthogonalize and deorthogonalize any operator.

Typically the Hamiltonin needs to be orthogonalized: $H_{\text{ortho}} = Z^{\dagger} H Z$.

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

Definition at line 10 of file nonortho_mod.F90.

11.18.2 Member Function/Subroutine Documentation

11.18.2.1 subroutine, public nonortho_mod::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*)

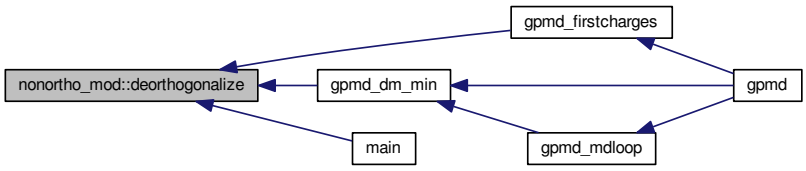
This routine performs: $A = Z A_{ortho} Z^{\dagger}$.

Parameters

orthoA_bml	Matrix to be deorthogonalized.
zmat_bml	Congruence transform to be used.
A_bml	Matrix resulting from the 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 80 of file nonortho_mod.F90.

Here is the caller graph for this function:



11.18.2.2 subroutine, public nonortho_mod::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*)

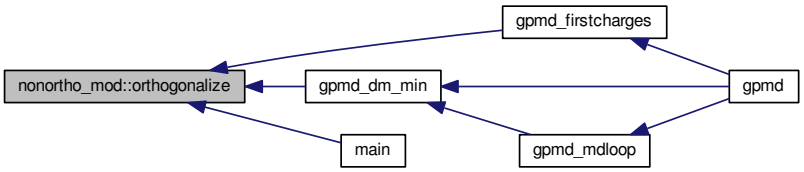
This routine performs: $A_{ortho} = Z^{\dagger} A Z$.

Parameters

A_bml	Matrix to be 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 35 of file nonortho_mod.F90.

Here is the caller graph for this function:



11.18.3 Member Data Documentation

11.18.3.1 integer, parameter nonortho_mod::dp = kind(1.0d0) [private]

Definition at line 19 of file nonortho_mod.F90.

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

- /home/christian/qmd-progress/src/nonortho_mod.F90

11.19 normalize_mod Module Reference

The normalize module.

Public Member Functions

- subroutine, public normalize (H_bml)

  Normalize a Hamiltonian matrix prior to running the SP2 algorithm.
- subroutine, public gershgorinreduction (gp)

  Determine gershgorin bounds across all parts, local and distributed.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.19.1 Detailed Description

The normalize module.

Definition at line 26 of file normalize_mod.F90.

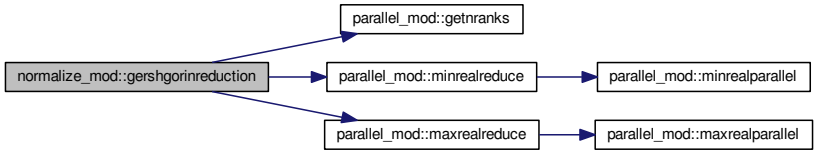
11.19.2 Member Function/Subroutine Documentation

11.19.2.1 subroutine, public normalize_mod::gershgorinreduction (type(graph_partitioning_t), intent(inout) *gp*)

Determine gershgorin bounds across all parts, local and distributed.

Definition at line 70 of file normalize_mod.F90.

Here is the call graph for this function:



11.19.2.2 subroutine, public normalize_mod::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)

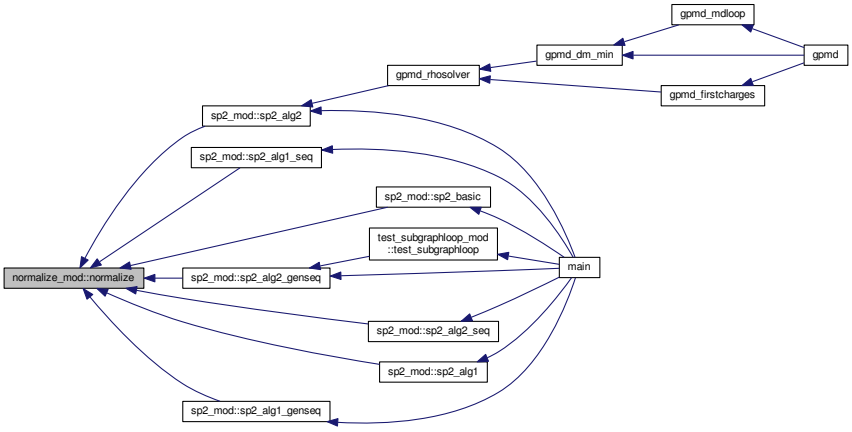
where e_max and e_min are obtained sing the Gershgorin circle theorem.

Parameters

H_bml	Input/Output Hamiltonian matrix

Definition at line 50 of file normalize_mod.F90.

Here is the caller graph for this function:



11.19.3 Member Data Documentation

11.19.3.1 integer, parameter normalize_mod::dp = kind(1.0d0) [private]

Definition at line 36 of file normalize_mod.F90.

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

- /home/christian/qmd-progress/src/normalize_mod.F90

11.20 openfiles_mod Module Reference

Module to handle input output files for the PROGRESS lib.

Public Member Functions

- integer function, public get_file_unit (io_max)
  Returns a unit number that is not in use.
- subroutine, public open_file (io, name)
  Opens a file to write.
- subroutine, public open_file_to_read (io, name)
  Opens a file to read.

11.20.1 Detailed Description

Module to handle input output files for the PROGRESS lib.

Definition at line 4 of file openfiles_mod.F90.

11.20.2 Member Function/Subroutine Documentation

11.20.2.1 integer function, public openfiles_mod::get_file_unit (integer *io_max*)

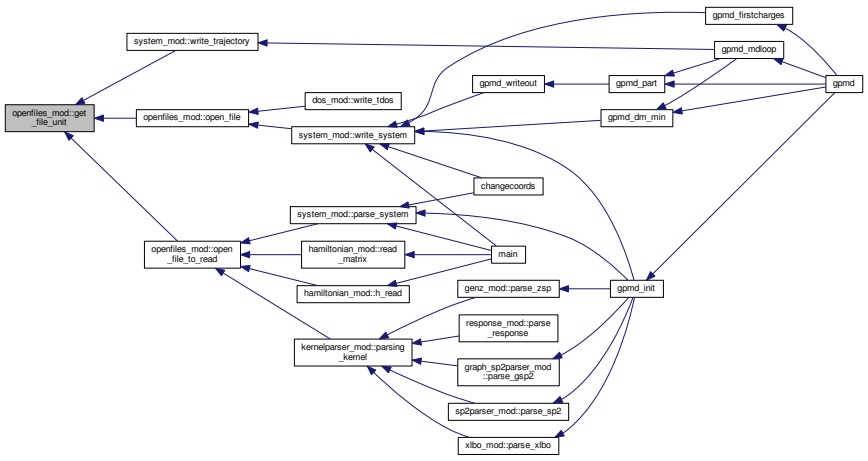
Returns a unit number that is not in use.

Parameters

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

Definition at line 18 of file openfiles_mod.F90.

Here is the caller graph for this function:



11.20.2.2 subroutine, public openfiles_mod::open_file (integer *io,* character(len=∗) *name*)

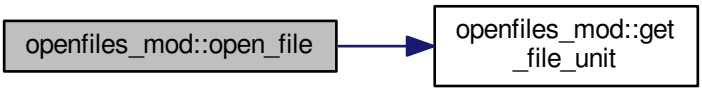
Opens a file to write.

Parameters

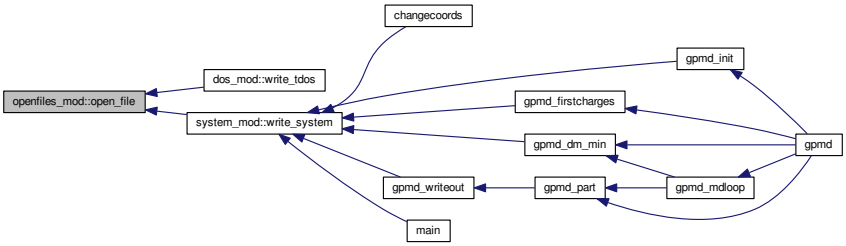
io	Unit for the file.
name	Name of the file.

Definition at line 37 of file openfiles_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.20.2.3 subroutine, public openfiles_mod::open_file_to_read (integer *io,* character(len=∗) *name*)

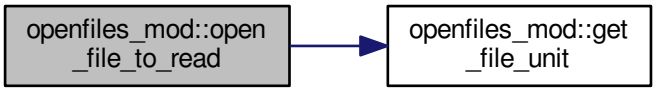
Opens a file to read.

Parameters

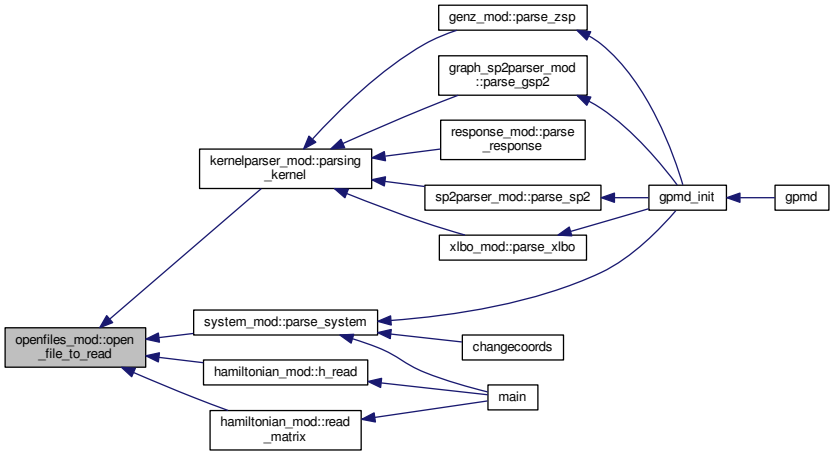
io	Unit for the file.
name	Name of the file.

Definition at line 53 of file openfiles_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /home/christian/qmd-progress/src/openfiles_mod.F90

11.21 parallel_mod Module Reference

The parallel module.

Data Types

- type rankreducedata_t

  Data structure for rection over MPI ranks.

Public Member Functions

- integer function, public getnranks ()
- integer function, public getmyrank ()
- integer function, public printrank ()
- subroutine, public initparallel ()

- subroutine, public shutdownparallel ()
- subroutine, public 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 irecvparallel (recvBuf, recvLen, rind)
- subroutine, public 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 minrealreduce (rvalue)
- subroutine, public maxrealreduce (rvalue)
- subroutine, public maxintreduce2 (value1, value2)
- subroutine, public sumintreduce2 (value1, value2)
- subroutine, public sumrealreduce (value1)
- subroutine, public sumrealreduce2 (value1, value2)
- subroutine, public sumrealreduce3 (value1, value2, value3)
- subroutine, public sumrealreducen (valueVec, N)
- subroutine, public sumintreducen (valueVec, N)
- subroutine, public minrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public maxrankrealparallel (sendBuf, recvBuf, icount)
- subroutine, public 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 allsumrealreduceparallel (buf, buflen)
- subroutine, public allsumintreduceparallel (buf, buflen)
- subroutine, public allgatherparallel (a)
- subroutine, public wait ()

Private Member Functions

- integer function saverequest (irequest)

Private Attributes

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

11.21.1 Detailed Description

The parallel module.

Definition at line 27 of file parallel_mod.F90.

11.21.2 Member Function/Subroutine Documentation

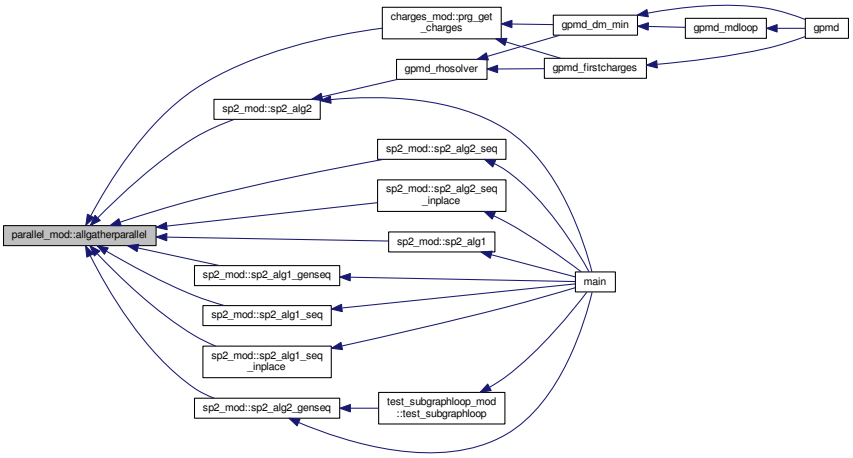
11.21.2.1 subroutine, public parallel_mod::allgatherintparallel (integer, dimension(∗), intent(in) *sendBuf,* integer, intent(in) *sendLen,* integer, dimension(∗), intent(out) *recvBuf,* integer, intent(in) *recvLen*)

Definition at line 680 of file parallel_mod.F90.

11.21.2.2 subroutine, public parallel_mod::allgatherparallel (type (bml_matrix_t), intent(inout) *a*)

Definition at line 764 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.3 subroutine, public 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 664 of file parallel_mod.F90.

11.21.2.4 subroutine, public 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 716 of file parallel_mod.F90.

11.21.2.5 subroutine, public 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 696 of file parallel_mod.F90.

11.21.2.6 subroutine, public parallel_mod::allsumintreduceparallel (integer, dimension(∗), intent(inout) *buf,* integer, intent(in) *buflen*)

Definition at line 749 of file parallel_mod.F90.

11.21.2.7 subroutine, public parallel_mod::allsumrealreduceparallel (real(dp), dimension(∗), intent(inout) *buf,* integer, intent(in) *buflen*)

Definition at line 734 of file parallel_mod.F90.

11.21.2.8 subroutine, public parallel_mod::barrierparallel ( )

Definition at line 216 of file parallel_mod.F90.

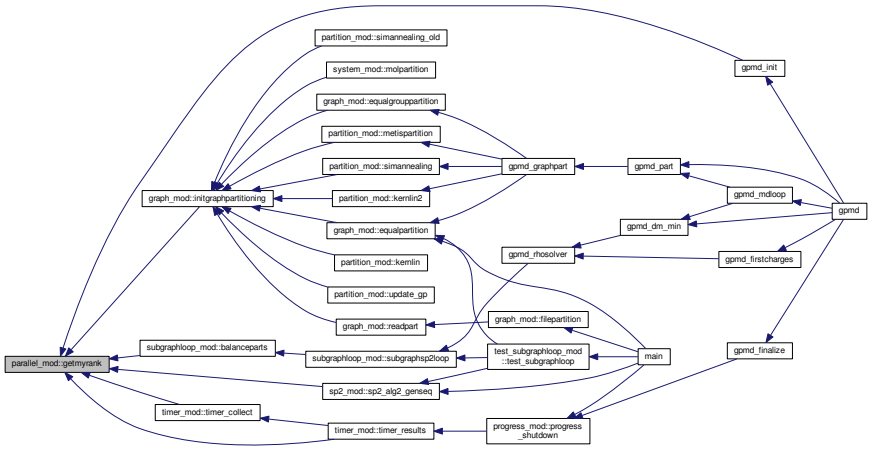
11.21.2.9 subroutine, public parallel_mod::bcastparallel (character, dimension(∗), intent(in) *buf,* integer, intent(in) *blen,* integer, intent(in) *root*)

Definition at line 650 of file parallel_mod.F90.

11.21.2.10 integer function, public parallel_mod::getmyrank ( )

Definition at line 119 of file parallel_mod.F90.

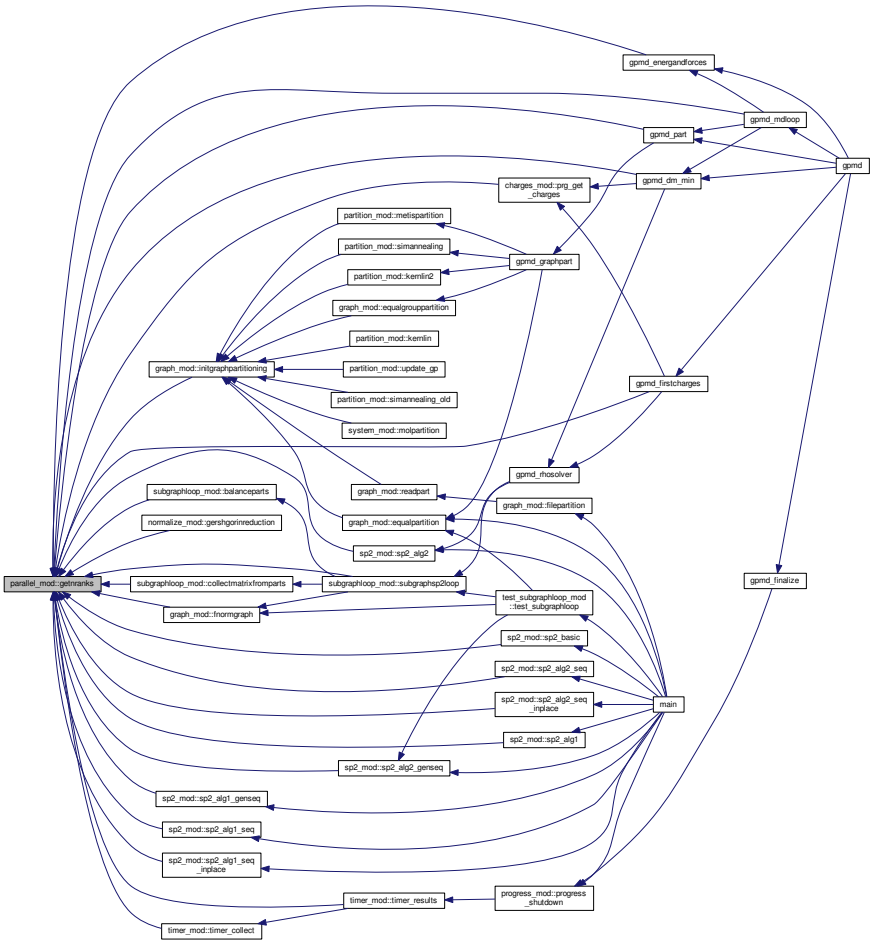
Here is the caller graph for this function:



11.21.2.11 integer function, public parallel_mod::getnranks ( )

Definition at line 108 of file parallel_mod.F90.

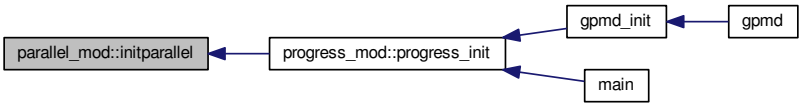
Here is the caller graph for this function:



11.21.2.12 subroutine, public parallel_mod::initparallel ( )

Definition at line 147 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.13 subroutine, public parallel_mod::irecvparallel (real(dp), dimension(∗) *recvBuf,* integer, intent(in) *recvLen,* integer *rind*)

Definition at line 281 of file parallel_mod.F90.

Here is the call graph for this function:

11.21.2.14 subroutine, public parallel_mod::isendparallel (real(dp), dimension(∗), intent(in) *sendBuf,* integer, intent(in) *sendLen,* integer, intent(in) *dest*)

Definition at line 250 of file parallel_mod.F90.

11.21.2.15 subroutine, public parallel_mod::maxintparallel (integer, dimension(∗), intent(in) *sendBuf,* integer, dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 357 of file parallel_mod.F90.

Here is the caller graph for this function:

11.21.2.16 subroutine, public parallel_mod::maxintreduce2 (integer, intent(inout) *value1,* integer, intent(inout) *value2*)

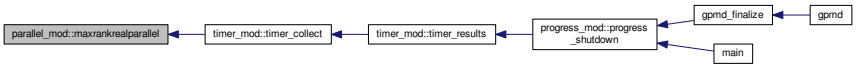
Definition at line 473 of file parallel_mod.F90.

Here is the call graph for this function:

11.21.2.17 subroutine, public parallel_mod::maxrankrealparallel (type(rankreducedata_t), dimension(∗), intent(in) *sendBuf,* type(rankreducedata_t), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 627 of file parallel_mod.F90.

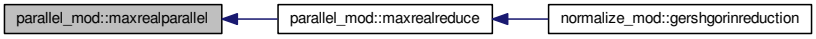
Here is the caller graph for this function:



11.21.2.18 subroutine, public parallel_mod::maxrealparallel (real(dp), dimension(∗), intent(in) *sendBuf,* real(dp), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 378 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.19 subroutine, public parallel_mod::maxrealreduce (real(dp), intent(inout) *rvalue*)

Definition at line 457 of file parallel_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:

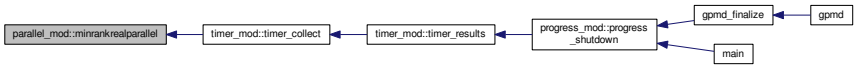
11.21.2.20 subroutine, public parallel_mod::minintparallel (integer, dimension(∗), intent(in) *sendBuf,* integer, dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 399 of file parallel_mod.F90.

11.21.2.21 subroutine, public parallel_mod::minrankrealparallel (type(rankreducedata_t), dimension(∗), intent(in) *sendBuf,* type(rankreducedata_t), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 604 of file parallel_mod.F90.

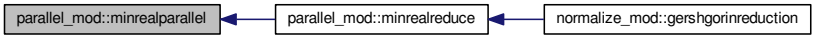
Here is the caller graph for this function:



11.21.2.22 subroutine, public parallel_mod::minrealparallel (real(dp), dimension(∗), intent(in) *sendBuf,* real(dp), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 420 of file parallel_mod.F90.

Here is the caller graph for this function:

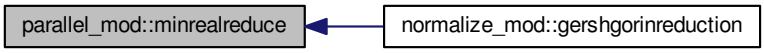


11.21.2.23 subroutine, public parallel_mod::minrealreduce (real(dp), intent(inout) *rvalue*)

Definition at line 441 of file parallel_mod.F90.

Here is the call graph for this function:

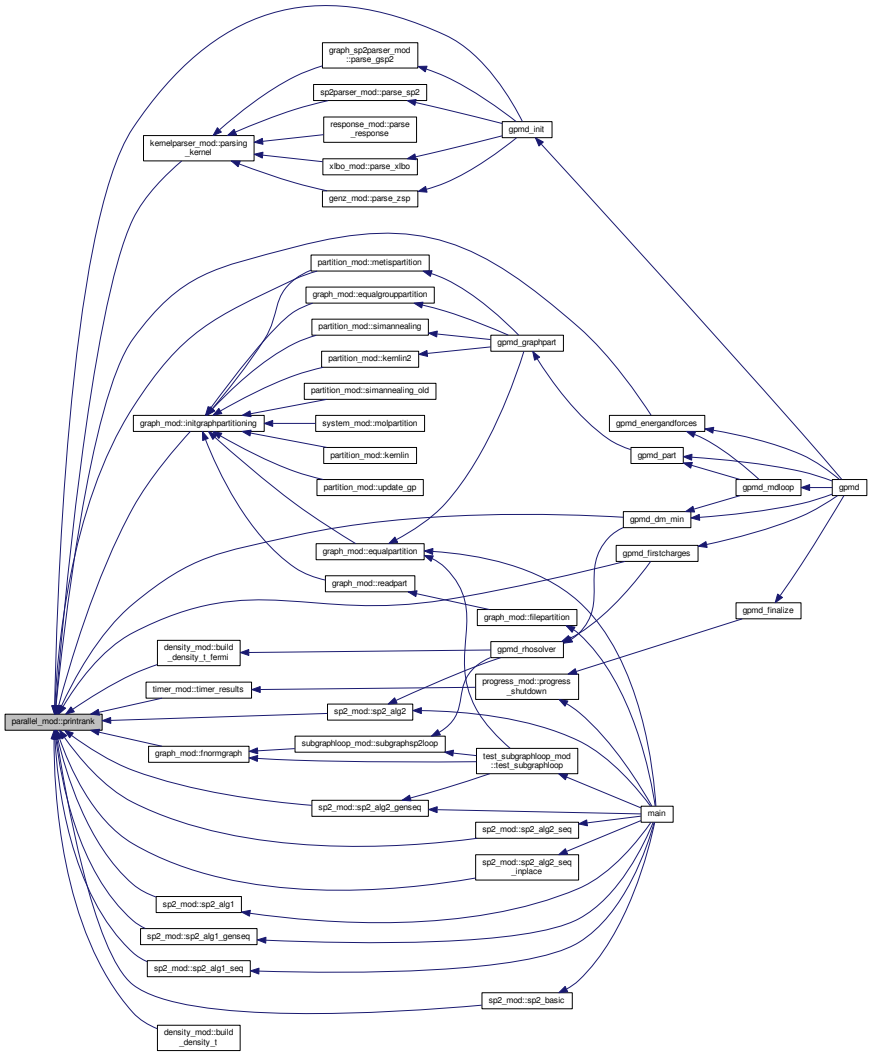
Here is the caller graph for this function:



11.21.2.24 integer function, public parallel_mod::printrank ( )

Definition at line 131 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.25 subroutine, public parallel_mod::recvparallel (real(dp), dimension(∗) *recvBuf,* integer, intent(in) *recvLen*)

Definition at line 299 of file parallel_mod.F90.

11.21.2.26 integer function parallel_mod::saverequest (integer, intent(in) *irequest*) [private]

Definition at line 190 of file parallel_mod.F90.

Here is the caller graph for this function:

11.21.2.27 subroutine, public parallel_mod::sendparallel (real(dp), dimension(∗), intent(in) *sendBuf,* integer, intent(in) *sendLen,* integer, intent(in) *dest*)

Definition at line 266 of file parallel_mod.F90.

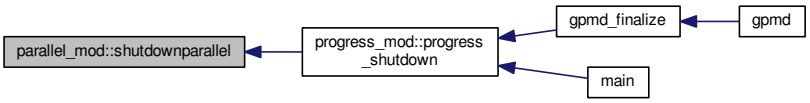
11.21.2.28 subroutine, public parallel_mod::sendreceiveparallel (real(dp), dimension(∗), intent(in) *sendBuf,* integer, intent(in) *sendLen,* integer, intent(in) *dest,* real(dp), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *recvLen,* integer, intent(in) *source,* integer, intent(out) *nreceived*)

Definition at line 227 of file parallel_mod.F90.

11.21.2.29 subroutine, public parallel_mod::shutdownparallel ( )

Definition at line 174 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.30 subroutine, public parallel_mod::sumintparallel (integer, dimension(∗), intent(in) *sendBuf,* integer, dimension(∗) *recvBuf,* integer, intent(in) *icount*)

Definition at line 315 of file parallel_mod.F90.

Here is the caller graph for this function:

11.21.2.31 subroutine, public parallel_mod::sumintreduce2 (integer, intent(inout) *value1,* integer, intent(inout) *value2*)

Definition at line 491 of file parallel_mod.F90.

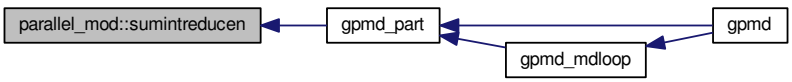
Here is the call graph for this function:

11.21.2.32 subroutine, public parallel_mod::sumintreducen (integer, dimension(n), intent(inout) *valueVec,* integer, intent(in) *N*)

Definition at line 584 of file parallel_mod.F90.

Here is the call graph for this function:

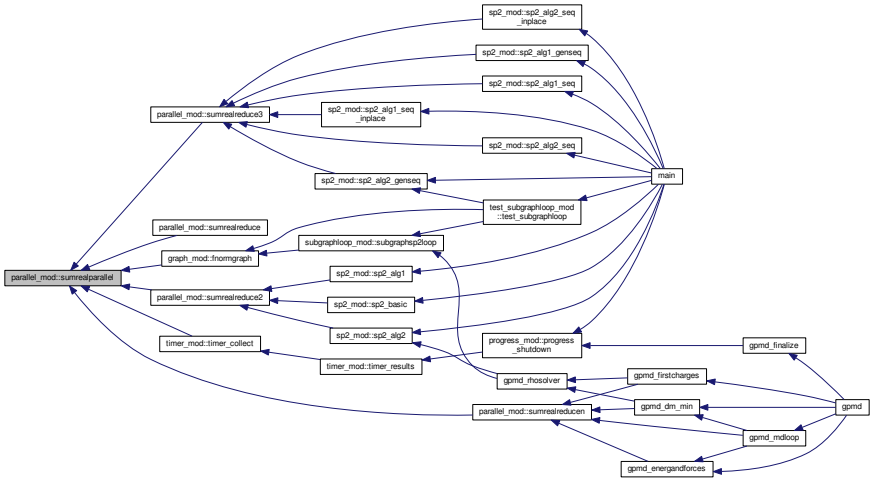
Here is the caller graph for this function:



11.21.2.33 subroutine, public parallel_mod::sumrealparallel (real(dp), dimension(∗), intent(in) *sendBuf,* real(dp), dimension(∗), intent(out) *recvBuf,* integer, intent(in) *icount*)

Definition at line 336 of file parallel_mod.F90.

Here is the caller graph for this function:



11.21.2.34 subroutine, public parallel_mod::sumrealreduce (real(dp), intent(inout) *value1*)

Definition at line 509 of file parallel_mod.F90.

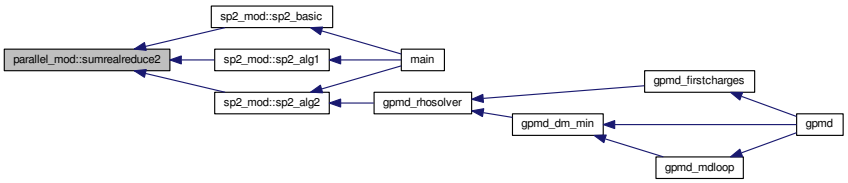
Here is the call graph for this function:

11.21.2.35 subroutine, public parallel_mod::sumrealreduce2 (real(dp), intent(inout) *value1,* real(dp), intent(inout) *value2*)

Definition at line 525 of file parallel_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:

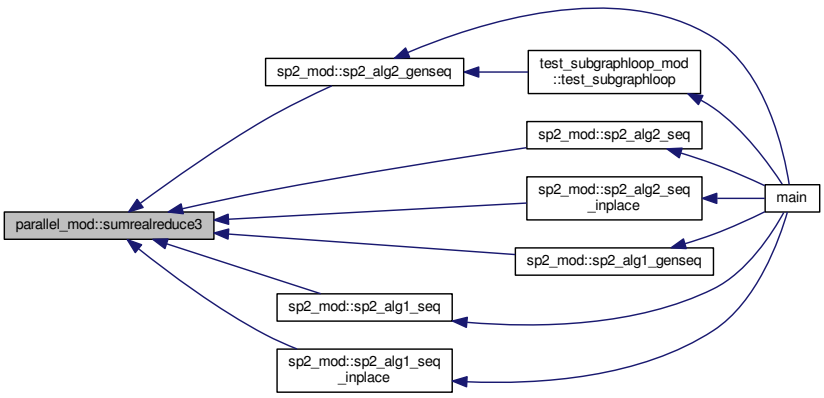


11.21.2.36 subroutine, public parallel_mod::sumrealreduce3 (real(dp), intent(inout) *value1,* real(dp), intent(inout) *value2,* real(dp), intent(inout) *value3*)

Definition at line 543 of file parallel_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:

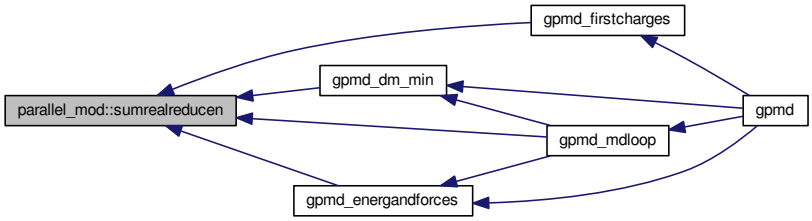


11.21.2.37 subroutine, public parallel_mod::sumrealreducen (real(dp), dimension(n), intent(inout) *valueVec,* integer, intent(in) *N*)

Definition at line 563 of file parallel_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:



11.21.2.38 subroutine, public parallel_mod::wait ( )

Definition at line 778 of file parallel_mod.F90.

11.21.3 Member Data Documentation

11.21.3.1 integer, parameter parallel_mod::dp = kind(1.0d0) [private]

Definition at line 47 of file parallel_mod.F90.

11.21.3.2 integer parallel_mod::ierr [private]

Definition at line 50 of file parallel_mod.F90.

11.21.3.3 integer parallel_mod::myrank [private]

Definition at line 49 of file parallel_mod.F90.

11.21.3.4 integer parallel_mod::nranks [private]

Definition at line 49 of file parallel_mod.F90.

11.21.3.5 integer parallel_mod::reqcount [private]

Definition at line 50 of file parallel_mod.F90.

11.21.3.6 integer, dimension(:), allocatable parallel_mod::requestlist [private]

Definition at line 51 of file parallel_mod.F90.

11.21.3.7 integer, dimension(:), allocatable parallel_mod::rused [private]

Definition at line 51 of file parallel_mod.F90.

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

- /home/christian/qmd-progress/src/parallel_mod.F90

11.22 partition_mod Module Reference

The partition module.

Public Member Functions

- subroutine, public 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 costpartition (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)

  Compute cost of a partition.
- subroutine, public update_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, public 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 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) =(delta, best_part),with delta = change in obj_value Dequeue and allow hill climbing.
- subroutine, public update_gp (gp, partNumber, core_count)
- subroutine, public check_arrays (gp, core_count, CH_count, Halo_count)

  Error checking Checking that core_count, CH_count, Halo_count match.
- subroutine, public 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, public kernlin2 (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm)
- subroutine, public simannealing_old (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, niter, seed)

Private Member Functions

- subroutine accept_prob (it, delta, r)

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

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

  Pick a random node.
- subroutine rand_shuffle (array, seed)

  Randomly shuffle array.
- subroutine 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 get_largest_hedge_in_part (gp, xadj, adjncy, partNumber, core_count, CH_count, Halo_count, sumCubes, maxCH, smooth_maxCH, pnorm, search_part, largest_Hedge)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.22.1 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.

Definition at line 27 of file partition_mod.F90.

11.22.2 Member Function/Subroutine Documentation

11.22.2.1 subroutine partition_mod::accept_prob (integer, intent(in) *it*, real(dp), intent(in) *delta*, real, intent(inout) *r*) [private]

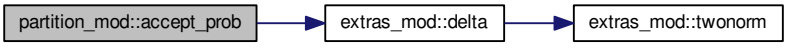
Compute acceptance probability for simulated annealing.

Parameters

it	iteration
delta	(new_obj_value - old_obj_value)
r	acceptance probability

Definition at line 328 of file partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

11.22.2.2 subroutine, public partition_mod::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 984 of file partition_mod.F90.

11.22.2.3 subroutine partition_mod::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]

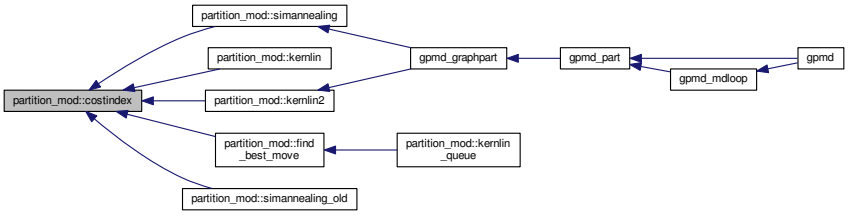
Choose objective function to work with.

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 346 of file partition_mod.F90.

Here is the caller graph for this function:



11.22.2.4 subroutine, public partition_mod::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*)

Compute cost of a partition.

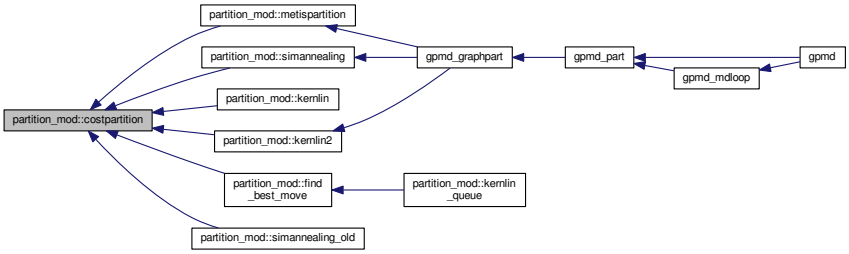
Parameters

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

initialize

Definition at line 172 of file partition_mod.F90.

Here is the caller graph for this function:

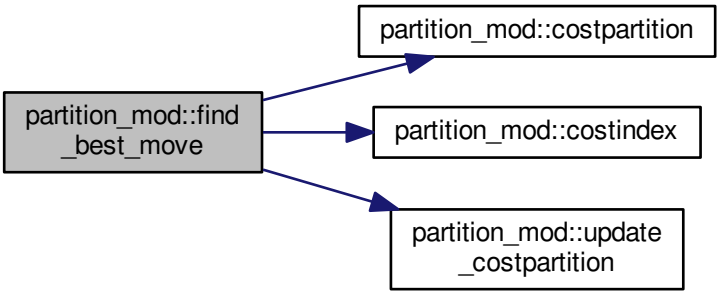


11.22.2.5 **subroutine partition_mod::find_best_move (type (graph_partitioning_t), intent(inout) *gp,* integer, dimension(:), intent(inout), allocatable *xadj,* integer, dimension(:), intent(inout), allocatable *adjncy,* integer, dimension(:), intent(inout), allocatable *partNumber,* integer, dimension(:), 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]

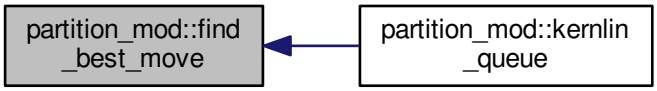
For kerlin_queue to find (vertex, new_part) pair with highest gain.

Definition at line 1047 of file partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

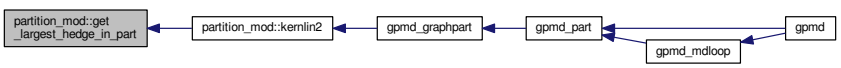


11.22.2.6 **subroutine partition_mod::get_largest_hedge_in_part (type (graph_partitioning_t), intent(inout) *gp,* integer, dimension(:), intent(inout), allocatable *xadj,* integer, dimension(:), intent(inout), allocatable *adjncy,* integer, dimension(:), intent(inout), allocatable *partNumber,* integer, 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 1265 of file partition_mod.F90.

Here is the caller graph for this function:



11.22.2.7 subroutine, public partition_mod::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) =(delta, best_part),with 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

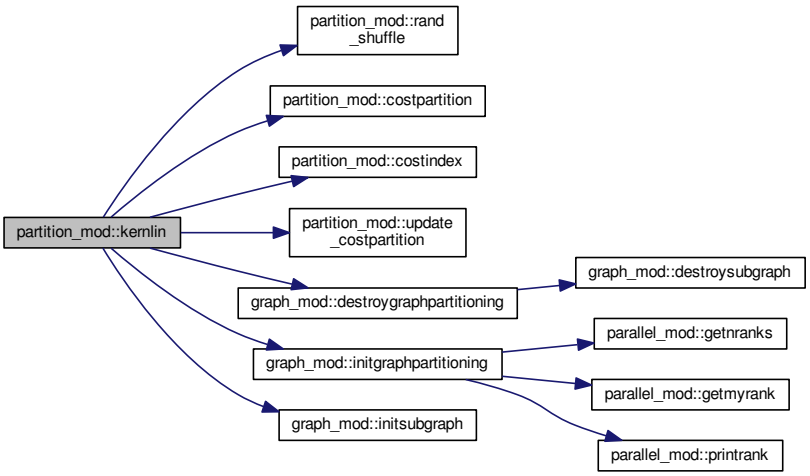
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 596 of file partition_mod.F90.

Here is the call graph for this function:



11.22.2.8 **subroutine, public partition_mod::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_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 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

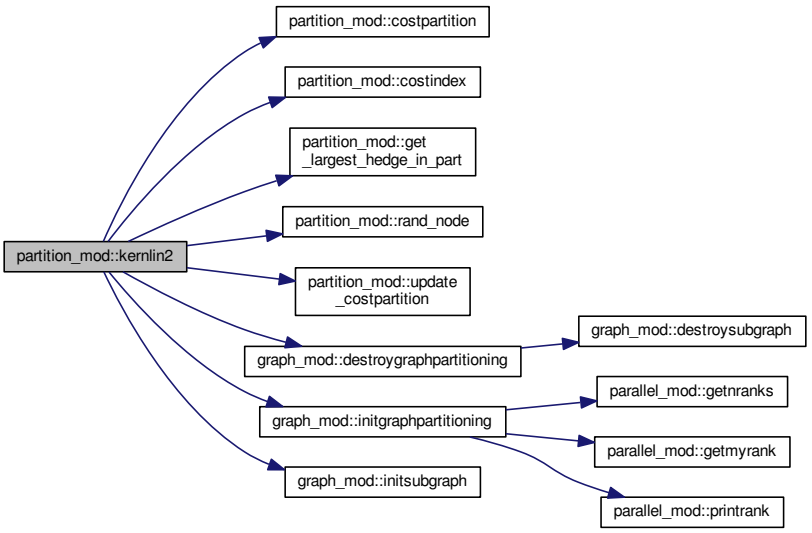
Update graph structure

Allocate subgraph structure

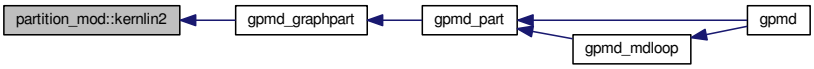
Assign node ids to sgraph

Definition at line 1100 of file partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

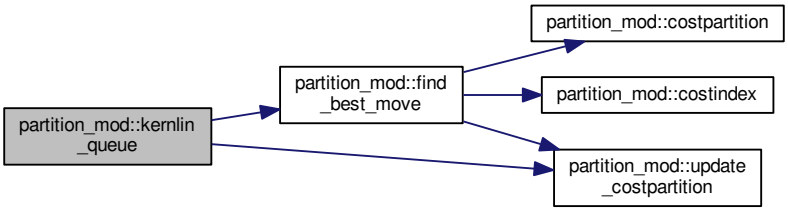


11.22.2.9 subroutine, public partition_mod::kernlin_queue (type (graph_partitioning_t), intent(inout) *gp,* integer, dimension(:), intent(inout), allocatable *xadj,* integer, dimension(:), intent(inout), allocatable *adjncy,* integer, dimension(:), intent(inout), allocatable *partNumber,* integer, dimension(:), 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 1011 of file partition_mod.F90.

Here is the call graph for this function:



11.22.2.10 **subroutine, public partition_mod::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

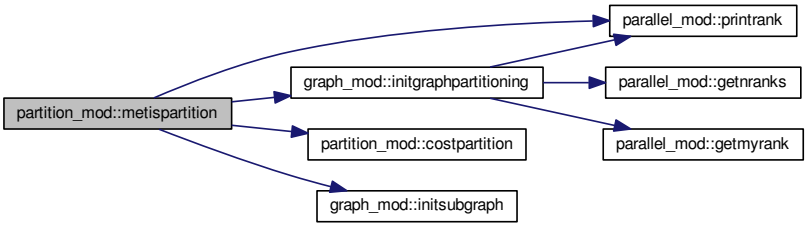
initialize

Partition graph into nparts’

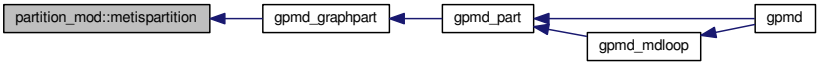
Compute cost of partition

Definition at line 68 of file partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.11 subroutine partition_mod::rand_node (type (graph_partitioning_t), intent(inout) *gp,* integer, intent(inout) *node,* integer, intent(inout) *seed*) [private]

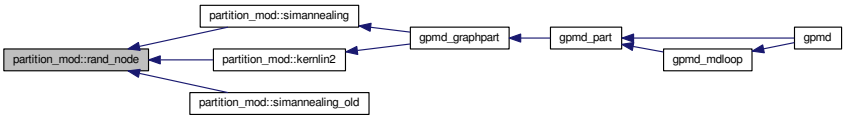
Pick a random node.

Parameters

gp	graph partitioning structure
node	output node
seed	random seed

Definition at line 366 of file partition_mod.F90.

Here is the caller graph for this function:



11.22.2.12 subroutine partition_mod::rand_shuffle (integer, dimension(:), intent(inout) *array,* integer, intent(inout) *seed*) [private]

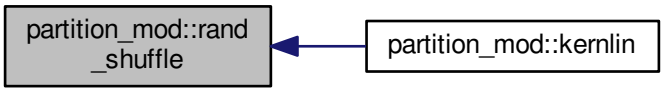
Randomly shuffle array.

Random seed

Shuffle array

Definition at line 961 of file partition_mod.F90.

Here is the caller graph for this function:



11.22.2.13 **subroutine, public partition_mod::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_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

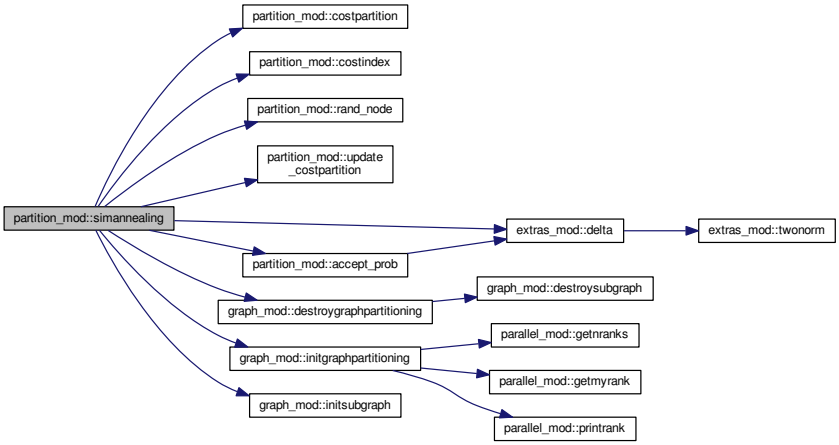
move it neighbor in the max parts to the newpart

Update graph structure

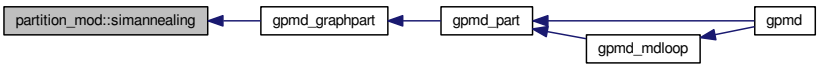
For debuging

Definition at line 391 of file partition_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.14 subroutine, public partition_mod::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

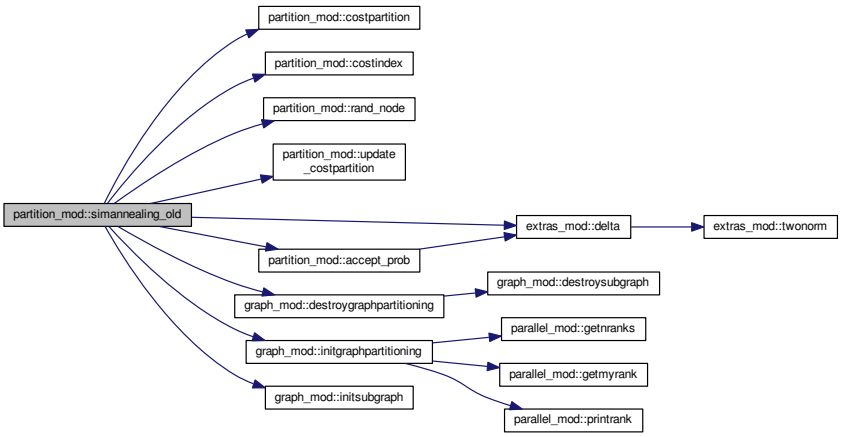
Perform SA

Update graph structure

For debuging

Definition at line 1298 of file partition_mod.F90.

Here is the call graph for this function:



11.22.2.15 subroutine, public partition_mod::update_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*)

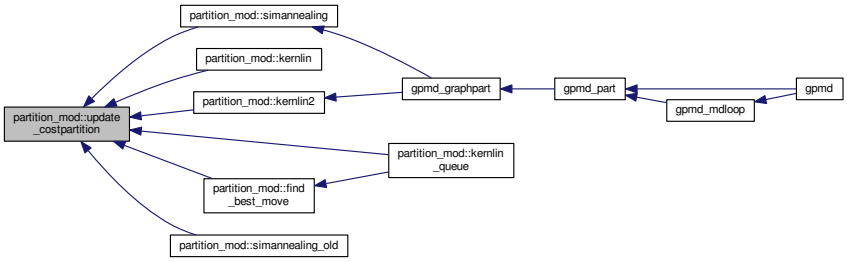
Update cost of partition and the different parameters node is moves into new_part For each neighbor of node, the following cases hold: Case 1: neighbor is in old_part Case 2: neighbor is in new_part Case 3: neighbor is neither in old_ or new_part.

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 240 of file partition_mod.F90.

Here is the caller graph for this function:



11.22.2.16 subroutine, public partition_mod::update_gp (type (graph_partitioning_t), intent(inout) *gp,* integer, dimension(:), intent(inout), allocatable *partNumber,* integer, dimension(:), intent(inout), allocatable *core_count*)

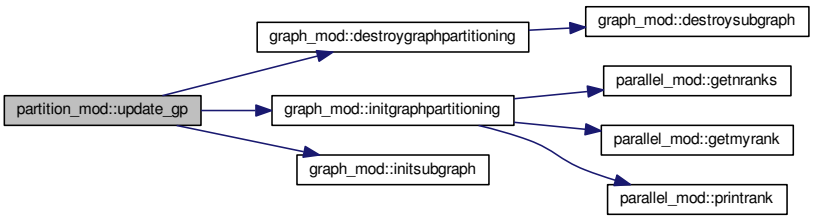
Update graph structure

Allocate subgraph structure

Assign node ids to sgraph

Definition at line 920 of file partition_mod.F90.

Here is the call graph for this function:



11.22.3 Member Data Documentation

11.22.3.1 integer, parameter partition_mod::dp = kind(1.0d0) [private]

Definition at line 38 of file partition_mod.F90.

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

- /home/christian/qmd-progress/src/partition_mod.F90

11.23 progress_mod Module Reference

The progress module.

Public Member Functions

- subroutine, public progress_init ()

  Initialize progress.
- subroutine, public progress_shutdown ()

  Shutdown progress.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.23.1 Detailed Description

The progress module.

Definition at line 27 of file progress_mod.F90.

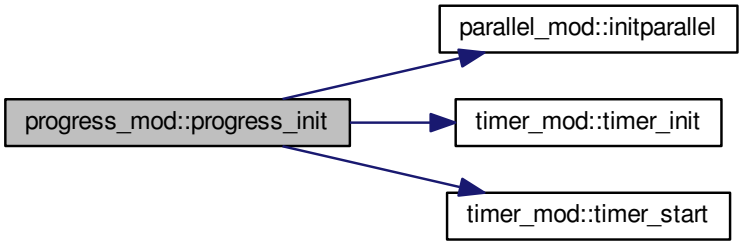
11.23.2 Member Function/Subroutine Documentation

11.23.2.1 subroutine, public progress_mod::progress_init ( )

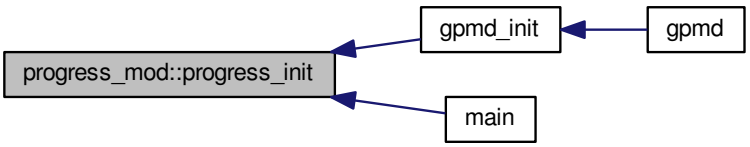
Initialize progress.

Definition at line 45 of file progress_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

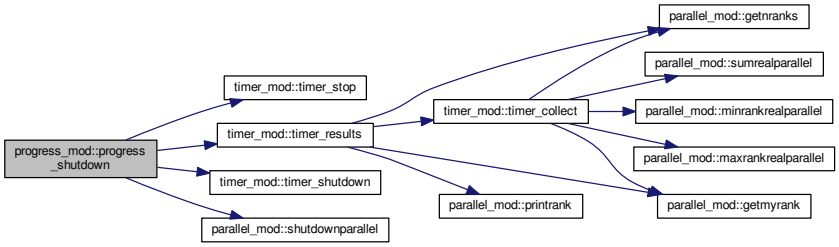


11.23.2.2 subroutine, public progress_mod::progress_shutdown ( )

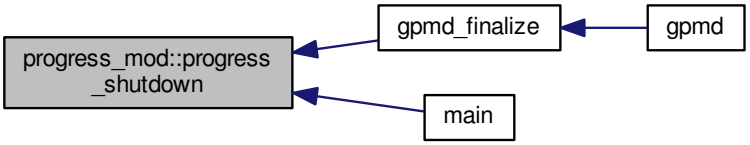
Shutdown progress.

Definition at line 57 of file progress_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.23.3 Member Data Documentation

11.23.3.1 integer, parameter progress_mod::dp = kind(1.0d0) [private]

Definition at line 37 of file progress_mod.F90.

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

- /home/christian/qmd-progress/src/progress_mod.F90

11.24 ptable_mod Module Reference

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.

Public Member Functions

- integer function, public element_atomic_number (symbol)

- integer function element_atomic_number_upper (symbol)

Public Attributes

- integer, parameter nz = 103
- 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 affinity (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.

Private Attributes

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

11.24.1 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.

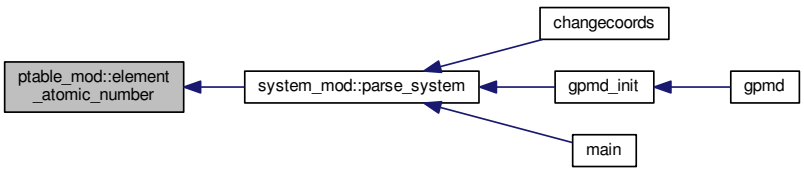
Definition at line 8 of file ptable_mod.F90.

11.24.2 Member Function/Subroutine Documentation

11.24.2.1 integer function, public ptable_mod::element_atomic_number (character(len=*) *symbol*)

Definition at line 393 of file ptable_mod.F90.

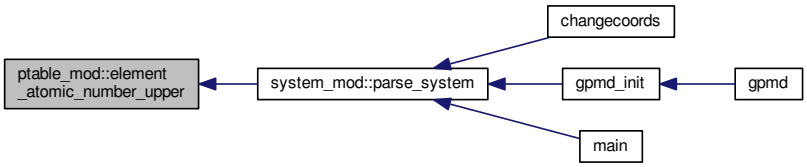
Here is the caller graph for this function:



11.24.2.2 integer function ptable_mod::element_atomic_number_upper (character(len=*) *symbol*)

Definition at line 407 of file ptable_mod.F90.

Here is the caller graph for this function:



11.24.3 Member Data Documentation

11.24.3.1 real(dp), dimension(nz), parameter 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 , 1.3 , 0.0 /)

The Pauling electronegativity for this element.

Definition at line 266 of file ptable_mod.F90.

11.24.3.2 integer, parameter, private ptable_mod::dp = kind(1.0d0) [private]

Definition at line 13 of file ptable_mod.F90.

11.24.3.3 real(dp), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.4 real(dp), dimension(nz), parameter 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 affinity (in eV)

Definition at line 235 of file ptable_mod.F90.

11.24.3.5 character(50), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.6 real(dp), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.7 real(dp), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.8 integer, dimension(nz), parameter 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 , 3 , 4 , 3 , 2 , 1 , 0 , 1 , 2 , 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 ptable_mod.F90.

11.24.3.9 character(20), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.10 integer, dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.11 character(2), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.12 character(2), dimension(nz), parameter 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 ptable_mod.F90.

11.24.3.13 real(dp), dimension(nz), parameter 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 , 2.0 /)

van der Waals radius (in Angstroms)

Definition at line 141 of file ptable_mod.F90.

11.24.3.14 integer, parameter ptable_mod::nz = 103

Definition at line 12 of file ptable_mod.F90.

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

- /home/christian/qmd-progress/src/ptable_mod.F90

11.25 pulaycomponent_mod Module Reference

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

Please see Niklasson 2008 **?**.

Public Member Functions

- subroutine, public pulaycomponent0 (rho_bml, ham_bml, pcm_bml, threshold, M, bml_type, verbose)

  At $T = 0K$, $P = \rho H \rho$.
- subroutine, public 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 get_pulayforce (nats, zmat_bml, ham_bml, rho_bml, dSx_bml, dSy_bml, dSz_bml, hindex, FPUL, threshold)

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

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.25.1 Detailed Description

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

Please see Niklasson 2008 **?**.

Definition at line 5 of file pulaycomponent_mod.F90.

11.25.2 Member Function/Subroutine Documentation

11.25.2.1 **subroutine, public pulaycomponent_mod::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*)**

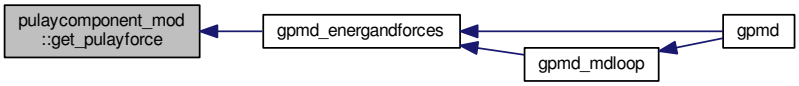
Pulay Force FPUL from $2Tr[ZZ'HD\frac{dS}{dR}]$.

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 150 of file pulaycomponent_mod.F90.

Here is the caller graph for this function:



11.25.2.2 subroutine, public pulaycomponent_mod::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 30 of file pulaycomponent_mod.F90.

11.25.2.3 subroutine, public pulaycomponent_mod::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 81 of file pulaycomponent_mod.F90.

11.25.3 Member Data Documentation

11.25.3.1 integer, parameter pulaycomponent_mod::dp = kind(1.0d0) [private]

Definition at line 13 of file pulaycomponent_mod.F90.

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

- /home/christian/qmd-progress/src/pulaycomponent_mod.F90

11.26 pulaymixer_mod Module Reference

Pulay mixer mode.

Gets the best coefficient for mixing the charges during scf.

Public Member Functions

- subroutine, public qmixer (charges, oldcharges, dqin, dqout, scferror, piter, pulaycoef, mpulay, verbose)
  Mixing the charges to acelerate scf convergence.
- subroutine, public linearmixer (charges, oldcharges, scferror, linmixcoef, verbose)
  Routine to perform linear mixing.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.26.1 Detailed Description

Pulay mixer mode.

Gets the best coefficient for mixing the charges during scf.

Todo add the density matrix mixer.

Definition at line 5 of file pulaymixer_mod.F90.

11.26.2 Member Function/Subroutine Documentation

11.26.2.1 subroutine, public pulaymixer_mod::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 163 of file pulaymixer_mod.F90.

11.26.2.2 subroutine, public pulaymixer_mod::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*)

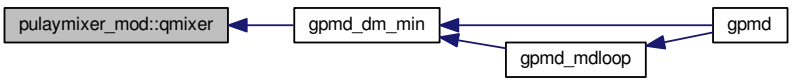
Mixing the charges to acelerate scf convergence.

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 29 of file pulaymixer_mod.F90.

Here is the caller graph for this function:



11.26.3 Member Data Documentation

11.26.3.1 integer, parameter pulaymixer_mod::dp = kind(1.0d0) [private]

Definition at line 13 of file pulaymixer_mod.F90.

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

- /home/christian/qmd-progress/src/pulaymixer_mod.F90

11.27 parallel_mod::rankreducedata_t Type Reference

Data structure for rection over MPI ranks.

Private Attributes

- real(dp) val
  Data value.
- integer rank
  MPI rank.

11.27.1 Detailed Description

Data structure for rection over MPI ranks.

Definition at line 93 of file parallel_mod.F90.

11.27.2 Member Data Documentation

11.27.2.1 integer parallel_mod::rankreducedata_t::rank [private]

MPI rank.

Definition at line 99 of file parallel_mod.F90.

11.27.2.2 real(dp) parallel_mod::rankreducedata_t::val [private]

Data value.

Definition at line 96 of file parallel_mod.F90.

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

- /home/christian/qmd-progress/src/parallel_mod.F90

11.28 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

11.28.1 Detailed Description

Definition at line 20 of file response_mod.F90.

11.28.2 Member Data Documentation

11.28.2.1 character(20) response_mod::respdata_type::bmltype

Definition at line 23 of file response_mod.F90.

11.28.2.2 logical response_mod::respdata_type::computedipole

Definition at line 26 of file response_mod.F90.

11.28.2.3 real(dp), dimension(3) response_mod::respdata_type::field

Definition at line 29 of file response_mod.F90.

11.28.2.4 real(dp) response_mod::respdata_type::fieldintensity

Definition at line 28 of file response_mod.F90.

11.28.2.5 logical response_mod::respdata_type::getresponse

Definition at line 27 of file response_mod.F90.

11.28.2.6 integer response_mod::respdata_type::mdim

Definition at line 24 of file response_mod.F90.

11.28.2.7 real(dp) response_mod::respdata_type::numthresh

Definition at line 25 of file response_mod.F90.

11.28.2.8 character(20) response_mod::respdata_type::respmode

Definition at line 21 of file response_mod.F90.

11.28.2.9 character(20) response_mod::respdata_type::typeofpert

Definition at line 22 of file response_mod.F90.

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

- /home/christian/qmd-progress/src/response_mod.F90

11.29 response_mod Module Reference

Module to compute the response and related quantities.

Data Types

- type respdata_type

Public Member Functions

- subroutine, public parse_response (RespData, filename)

  The parser for the response calculation.
- subroutine, public 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 compute_polarizability (resp_bml, pert_bml, polarizability, factor, verbose)

  To compute the polarizability of the system. The units of the directional polarizability are determined by the unis of the perturbation and Hamiltonian. This equation can be found in **?** *equation 4a. Note that in equation 4a of the reference there is a 2 that acount 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 pert_from_file (pert_bml, norb)

  Read perturbation from file.

- subroutine, public compute_response_rs (ham_bml, pert_bml, resp_bml, lambda, bndfil, threshold, verbose)

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

- subroutine, public compute_response_fd (ham_bml, pert_bml, resp_bml, delta, bndfil, threshold, verbose)

  Computes the first order response density matrix using Finite Differences. The transformation hereby performed are:

- subroutine, public pert_constant_field (field, intensity, coordinate, lambda, pert_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: $H^{(1)} = \lambda \frac{1}{2}(Se\mathbf{r}\cdot\mathbf{E} + e\mathbf{r}\cdot\mathbf{E}S)$. 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 implemetation $e = 1$ and units can be transformed by using the parameter λ.

- subroutine, public compute_response_sp2 (ham_bml, pert_bml, resp_bml, rho_bml, lambda, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, threshold, verbose)

  Finds the first order response matrix from a Hamiltonian matrix.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.29.1 Detailed Description

Module to compute the 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 teory can be found at **?** and Niklasson2015

Definition at line 7 of file response_mod.F90.

11.29.2 Member Function/Subroutine Documentation

11.29.2.1 subroutine, public response_mod::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 requiered).
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 112 of file response_mod.F90.

11.29.2.2 subroutine, public response_mod::compute_polarizability (type(bml_matrix_t), intent(in) *resp_bml,* type(bml_matrix_t), intent(in) *pert_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 unis of the perturbation and Hamiltonian. This equation can be found in **?** equation 4a. Note that in equation 4a of the reference there is a 2 that acount 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 requiered).
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 156 of file response_mod.F90.

11.29.2.3 subroutine, public response_mod::compute_response_fd (type(bml_matrix_t), intent(in) *ham_bml,* type(bml_matrix_t), intent(in) *pert_bml,* type(bml_matrix_t), intent(inout) *resp_bml,* real(dp) *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 ($H^{(0)}$).
pert_bml	Perturbation in bml format ($H^{(1)}$).
resp_bml	First order response to the perturbation ($\rho^{(1)}$).
bndfil	Filing factor.
threshold	Threshold value for matrix elements.

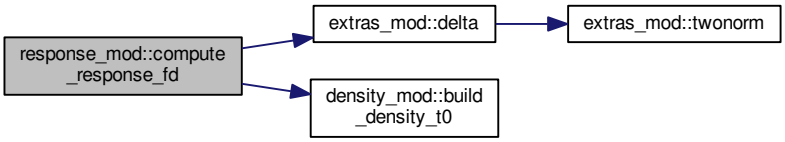
verbose	Different levels of verbosity.

Warning

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

Definition at line 353 of file response_mod.F90.

Here is the call graph for this function:



11.29.2.4 subroutine, public response_mod::compute_response_rs (type(bml_matrix_t), intent(in) *ham_bml,* type(bml_matrix_t), intent(in) *pert_bml,* type(bml_matrix_t), intent(inout) *resp_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 Pertrubations theory The transformation hereby performed are:

- $V = C^{\dagger} H^{(1)} C$
- $\tilde{V}_{ij} = \frac{V_{ij}}{\varepsilon_j - \varepsilon_i}$, 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 ($H^{(0)}$).
pert_bml	Perturbation in bml format ($H^{(1)}$).
resp_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 orthogonalized form of ham_bml.
The response must be in the orthogonalized form.

Definition at line 223 of file response_mod.F90.

11.29.2.5 subroutine, public response_mod::compute_response_sp2 (type(bml_matrix_t), intent(in) *ham_bml,* type(bml_matrix_t), intent(in) *pert_bml,* type(bml_matrix_t), intent(inout) *resp_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 488 of file response_mod.F90.

11.29.2.6 subroutine, public response_mod::parse_response (type(respdata_type) *RespData,* character(len=∗) *filename*)

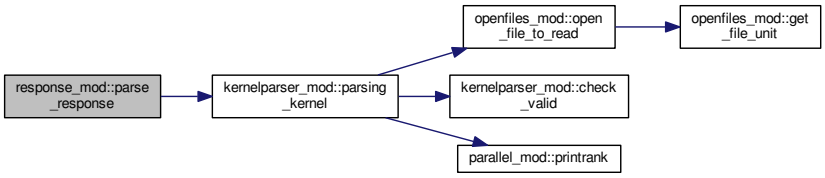
The parser for the response calculation.

Parameters

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

Definition at line 41 of file response_mod.F90.

Here is the call graph for this function:



11.29.2.7 subroutine, public response_mod::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) *pert_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: $H^{(1)} = \lambda \frac{1}{2}(Se\mathbf{r}\cdot\mathbf{E} + e\mathbf{r}\cdot\mathbf{E}S)$. 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 implemetation $e = 1$ and units can be transformed by using the parameter λ.

Note

If the Hamiltonian is already in the 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 (**r**).
lambda	Constant to premultiply the perturbation (λ).
pert_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 419 of file response_mod.F90.

11.29.2.8 subroutine, public response_mod::pert_from_file (type(bml_matrix_t), intent(inout) *pert_bml,* integer *norb*)

Read perturbation from file.

Todo Add read perturbation from file

Definition at line 182 of file response_mod.F90.

11.29.3 Member Data Documentation

11.29.3.1 integer, parameter response_mod::dp = kind(1.0d0) [private]

Definition at line 17 of file response_mod.F90.

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

- /home/christian/qmd-progress/src/response_mod.F90

11.30 sp2_mod Module Reference

The SP2 module.

Public Member Functions

- subroutine, public 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 sp2_alg2 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public sp2_alg2_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv, verbose)
- subroutine, public sp2_alg2_seq (h_bml, rho_bml, threshold, pp, icount, vv, verbose)
- subroutine, public sp2_alg2_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, verbose)
- subroutine, public sp2_alg1 (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, verbose)
- subroutine, public sp2_alg1_genseq (h_bml, rho_bml, threshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, pp, icount, vv)
- subroutine, public sp2_alg1_seq (h_bml, rho_bml, threshold, pp, icount, vv)
- subroutine, public sp2_alg1_seq_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval)
- subroutine, public 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 sp2_submatrix_inplace (rho_bml, threshold, pp, icount, vv, mineval, maxeval, core_size)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.30.1 Detailed Description

The SP2 module.

Author

S. Mniszewski (smn@lanl.gov)

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

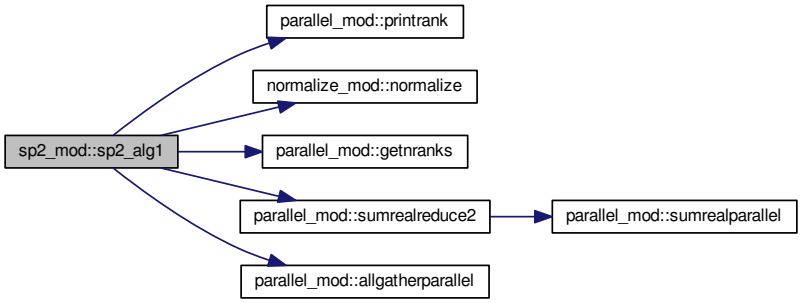
Definition at line 31 of file sp2_mod.F90.

11.30.2 Member Function/Subroutine Documentation

11.30.2.1 subroutine, public sp2_mod::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 643 of file sp2_mod.F90.

Here is the call graph for this function:

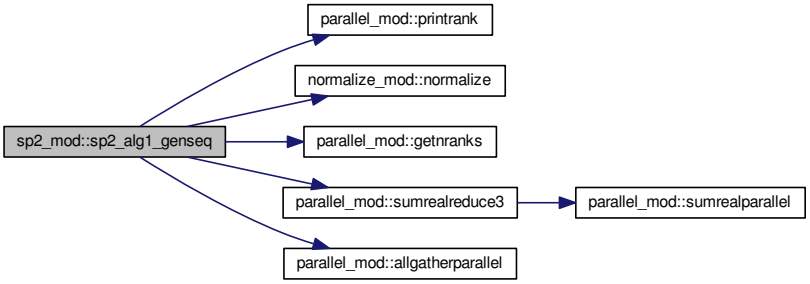


Here is the caller graph for this function:

11.30.2.2 subroutine, public sp2_mod::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 769 of file sp2_mod.F90.

Here is the call graph for this function:

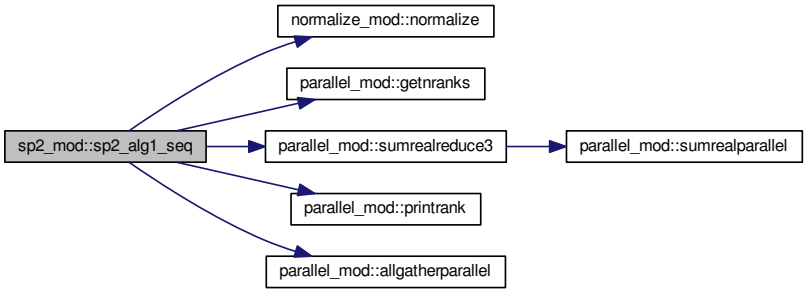


Here is the caller graph for this function:

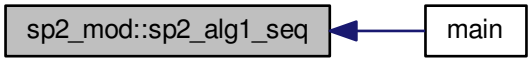
11.30.2.3 subroutine, public sp2_mod::sp2_alg1_seq (type(bml_matrix_t), intent(in) *h_bml,* type(bml_matrix_t), intent(inout) *rho_bml,* real(dp), intent(in) *threshold,* integer, dimension(:), intent(inout) *pp,* integer, intent(inout) *icount,* real(dp), dimension(:), intent(inout) *vv*)

Definition at line 900 of file sp2_mod.F90.

Here is the call graph for this function:



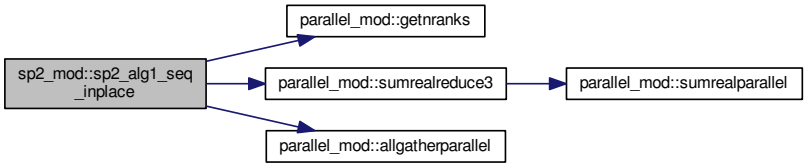
Here is the caller graph for this function:



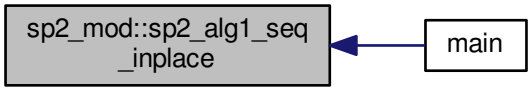
11.30.2.4 subroutine, public sp2_mod::sp2_alg1_seq_inplace (type(bml_matrix_t), intent(inout) *rho_bml,* real(dp), intent(in) *threshold,* integer, dimension(:), intent(inout) *pp,* integer, intent(inout) *icount,* real(dp), dimension(:), intent(inout) *vv,* real(dp), intent(in) *mineval,* real(dp), intent(in) *maxeval*)

Definition at line 990 of file sp2_mod.F90.

Here is the call graph for this function:



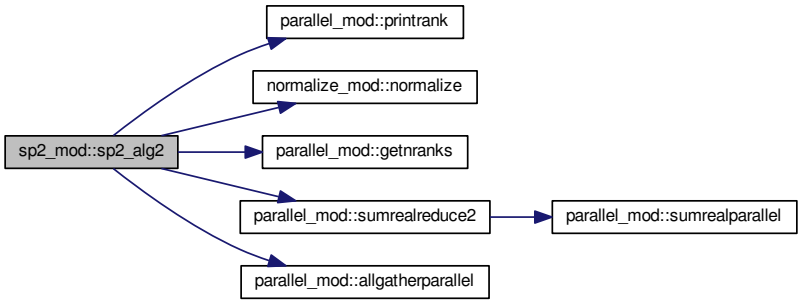
Here is the caller graph for this function:



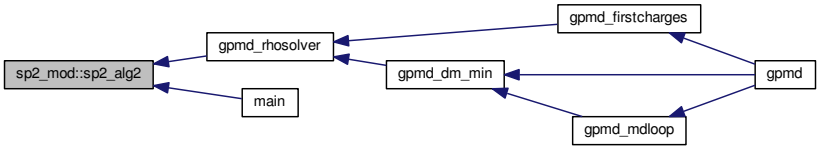
11.30.2.5 subroutine, public sp2_mod::sp2_alg2 (type(bml_matrix_t), intent(in) *h_bml,* type(bml_matrix_t), intent(inout) *rho_bml,* real(dp), intent(in) *threshold,* real(dp), intent(in) *bndfil,* integer, intent(in) *minsp2iter,* integer, intent(in) *maxsp2iter,* character(len=∗), intent(in) *sp2conv,* real(dp), intent(in) *idemtol,* integer, intent(in), optional *verbose*)

Definition at line 172 of file sp2_mod.F90.

Here is the call graph for this function:



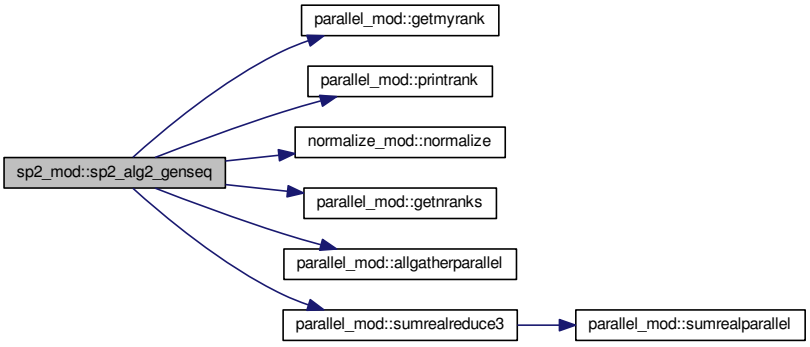
Here is the caller graph for this function:



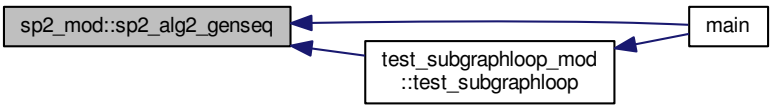
11.30.2.6 subroutine, public sp2_mod::sp2_alg2_genseq (type(bml_matrix_t), intent(in) *h_bml,* type(bml_matrix_t), intent(inout) *rho_bml,* real(dp), intent(in) *threshold,* real(dp), intent(in) *bndfil,* integer, intent(in) *minsp2iter,* integer, intent(in) *maxsp2iter,* character(len=∗), intent(in) *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 304 of file sp2_mod.F90.

Here is the call graph for this function:



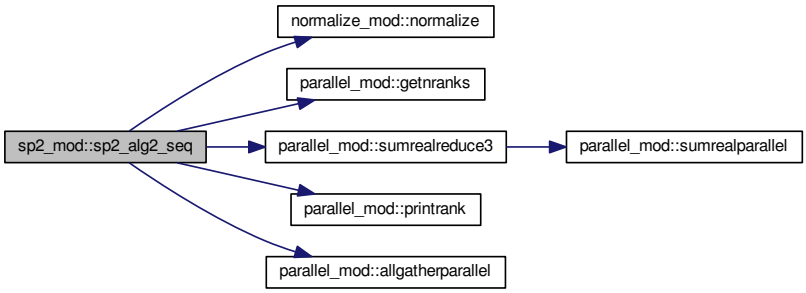
Here is the caller graph for this function:



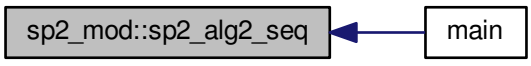
11.30.2.7 subroutine, public sp2_mod::sp2_alg2_seq (type(bml_matrix_t), intent(in) *h_bml,* type(bml_matrix_t), intent(inout) *rho_bml,* real(**dp**), intent(in) *threshold,* integer, dimension(:), intent(inout) *pp,* integer, intent(inout) *icount,* real(**dp**), dimension(:), intent(inout) *vv,* integer, intent(in), optional *verbose*)

Definition at line 452 of file sp2_mod.F90.

Here is the call graph for this function:



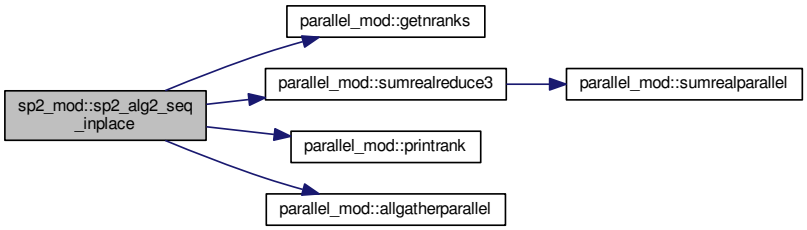
Here is the caller graph for this function:



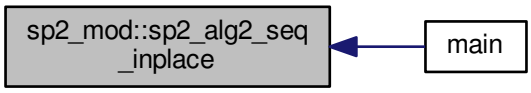
11.30.2.8 subroutine, public sp2_mod::sp2_alg2_seq_inplace (type(bml_matrix_t), intent(inout) *rho_bml,* real(**dp**), intent(in) *threshold,* integer, dimension(:), intent(inout) *pp,* integer, intent(inout) *icount,* real(**dp**), dimension(:), intent(inout) *vv,* real(**dp**), intent(in), optional *mineval,* real(**dp**), intent(in), optional *maxeval,* integer, intent(in), optional *verbose*)

Definition at line 547 of file sp2_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.30.2.9 subroutine, public sp2_mod::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*)

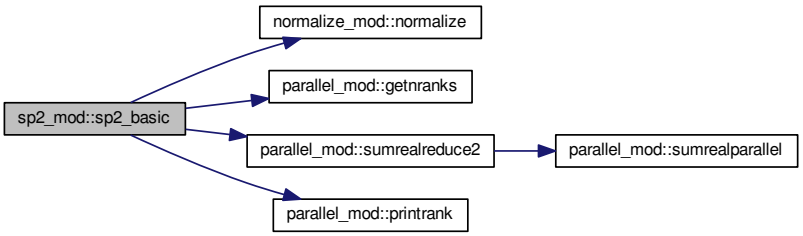
Calculates the density matrix from a Hamiltonian matrix by purification. The method implemented here is the very first verion of the SP2 method.

Parameters

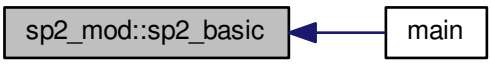
h_bml	Input Hamiltonian matrix
rho_bml	Output density matrix
threshold	Threshold for sparse matrix algebra
bndfil	Bond
minsp2iter	Minimum sp2 iterations
maxsp2iter	Maximum SP2 iterations
sp2conv	Convergence type
idemtol	Idempotency tolerance
verbose	A verbosity level

Definition at line 71 of file sp2_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.30.2.10 subroutine, public sp2_mod::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

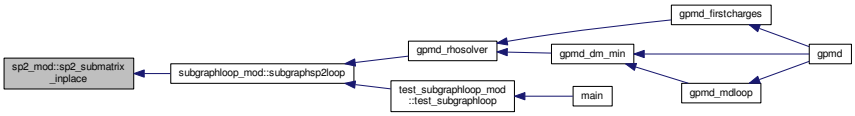
rho_bml	Input Hamiltonian/Output density matrix
threshold	Threshold for sparse matrix algebra
pp	Vector containing sequence of 0s and 1s
icount	Sequence count
vv	Vector of sum of squares per iteration
mineval	Min value used for normalization (optional)
maxeval	Max value used for normalization (optional)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

Definition at line 1080 of file sp2_mod.F90.

11.30.2.11 subroutine, public sp2_mod::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 1149 of file sp2_mod.F90.

Here is the caller graph for this function:



11.30.3 Member Data Documentation

11.30.3.1 integer, parameter sp2_mod::dp = kind(1.0d0) [private]

Definition at line 42 of file sp2_mod.F90.

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

- /home/christian/qmd-progress/src/sp2_mod.F90

11.31 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

11.31.1 Detailed Description

General SP2 solver type.

Definition at line 26 of file sp2parser_mod.F90.

11.31.2 Member Data Documentation

11.31.2.1 character(20) sp2parser_mod::sp2data_type::bml_type

Definition at line 38 of file sp2parser_mod.F90.

11.31.2.2 real(dp) sp2parser_mod::sp2data_type::bndfil

Definition at line 33 of file sp2parser_mod.F90.

11.31.2.3 character(10) sp2parser_mod::sp2data_type::flavor

Definition at line 40 of file sp2parser_mod.F90.

11.31.2.4 character(20) sp2parser_mod::sp2data_type::jobname

Definition at line 27 of file sp2parser_mod.F90.

11.31.2.5 integer sp2parser_mod::sp2data_type::maxsp2iter

Definition at line 30 of file sp2parser_mod.F90.

11.31.2.6 integer sp2parser_mod::sp2data_type::mdim

Definition at line 34 of file sp2parser_mod.F90.

11.31.2.7 integer sp2parser_mod::sp2data_type::minsp2iter

Definition at line 29 of file sp2parser_mod.F90.

11.31.2.8 integer sp2parser_mod::sp2data_type::ndim

Definition at line 35 of file sp2parser_mod.F90.

11.31.2.9 real(dp), dimension(3) sp2parser_mod::sp2data_type::pdim

Definition at line 37 of file sp2parser_mod.F90.

11.31.2.10 character, dimension(3) sp2parser_mod::sp2data_type::sdim

Definition at line 36 of file sp2parser_mod.F90.

11.31.2.11 character(10) sp2parser_mod::sp2data_type::sp2conv

Definition at line 39 of file sp2parser_mod.F90.

11.31.2.12 real(dp) sp2parser_mod::sp2data_type::sp2tol

Definition at line 31 of file sp2parser_mod.F90.

11.31.2.13 real(dp) sp2parser_mod::sp2data_type::threshold

Definition at line 32 of file sp2parser_mod.F90.

11.31.2.14 integer sp2parser_mod::sp2data_type::verbose

Definition at line 28 of file sp2parser_mod.F90.

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

- /home/christian/qmd-progress/src/sp2parser_mod.F90

11.32 sp2parser_mod Module Reference

SP2 parser.

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

Data Types

- type sp2data_type

  General SP2 solver type.

Public Member Functions

- subroutine, public parse_sp2 (sp2data, filename)

  The parser for SP2 solver.

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.32.1 Detailed Description

SP2 parser.

This module is used to parse all the neccesary input variables for and 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.

Definition at line 12 of file sp2parser_mod.F90.

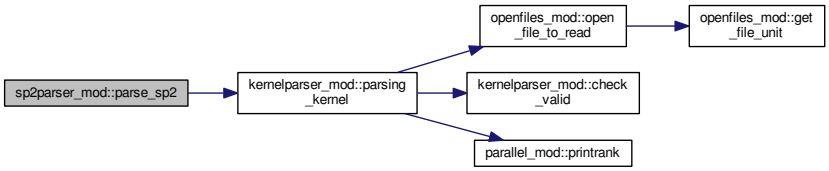
11.32.2 Member Function/Subroutine Documentation

11.32.2.1 subroutine, public sp2parser_mod::parse_sp2 (type(sp2data_type), intent(inout) *sp2data,* character(len=∗) *filename*)

The parser for SP2 solver.

Definition at line 49 of file sp2parser_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

11.32.3 Member Data Documentation

11.32.3.1 integer, parameter sp2parser_mod::dp = kind(1.0d0) [private]

Definition at line 22 of file sp2parser_mod.F90.

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

- /home/christian/qmd-progress/src/sp2parser_mod.F90

11.33 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.

11.33.1 Detailed Description

Subgraph type.

Definition at line 52 of file graph_mod.F90.

11.33.2 Member Data Documentation

11.33.2.1 integer, dimension(:), allocatable graph_mod::subgraph_t::core_halo_index [private]

Indeces from original matrix for subgraph core+halo extraction.

Definition at line 67 of file graph_mod.F90.

11.33.2.2 integer graph_mod::subgraph_t::hsize [private]

Size of original matrix (h x h)

Definition at line 58 of file graph_mod.F90.

11.33.2.3 integer graph_mod::subgraph_t::llsize [private]

Size of core subgraph.

Definition at line 64 of file graph_mod.F90.

11.33.2.4 integer graph_mod::subgraph_t::lsize [private]

Size of full subgraph (l x l)

Definition at line 61 of file graph_mod.F90.

11.33.2.5 integer, dimension(:), allocatable graph_mod::subgraph_t::nodeinpart [private]

Nodes in this partition.

Definition at line 70 of file graph_mod.F90.

11.33.2.6 integer graph_mod::subgraph_t::part [private]

Partition number.

Definition at line 55 of file graph_mod.F90.

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

- /home/christian/qmd-progress/src/graph_mod.F90

11.34 subgraphloop_mod Module Reference

The subgraphloop module.

Public Member Functions

- subroutine, public subgraphsp2loop (h_bml, g_bml, rho_bml, gp, threshold)

- subroutine, public collectmatrixfromparts (gp, rho_bml)

  Collect distributed parts into same matrix.
- subroutine, public balanceparts (gp)
- subroutine, public partordering (gp)

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

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

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

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.34.1 Detailed Description

The subgraphloop module.

Definition at line 26 of file subgraphloop_mod.F90.

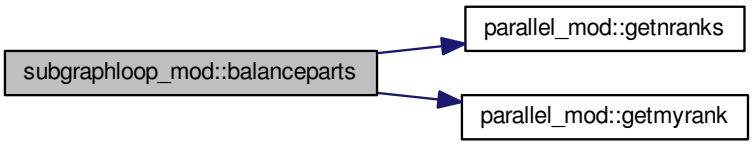
11.34.2 Member Function/Subroutine Documentation

11.34.2.1 subroutine, public subgraphloop_mod::balanceparts (type (graph_partitioning_t), intent(inout) *gp*)

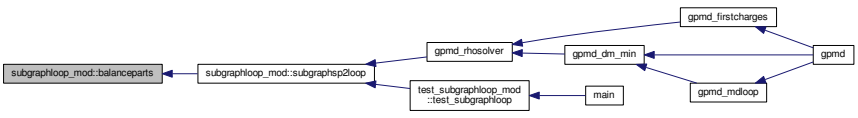
Renumber parts Handle unbalanced numbers of parts.

Definition at line 185 of file subgraphloop_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.34.2.2 subroutine, public subgraphloop_mod::collectmatrixfromparts (type (graph_partitioning_t), intent(inout) *gp,* type (bml_matrix_t), intent(inout) *rho_bml*)

Collect distributed parts into same matrix.

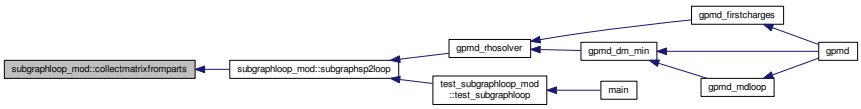
Parameters

gp	Graph partitioning
rho_bml	Matrix to be collected into

Definition at line 153 of file subgraphloop_mod.F90.

Here is the call graph for this function:

Here is the caller graph for this function:



11.34.2.3 subroutine, public subgraphloop_mod::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 312 of file subgraphloop_mod.F90.

11.34.2.4 subroutine, public subgraphloop_mod::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 357 of file subgraphloop_mod.F90.

11.34.2.5 subroutine, public subgraphloop_mod::partordering (type (graph_partitioning_t), intent(inout) *gp*)

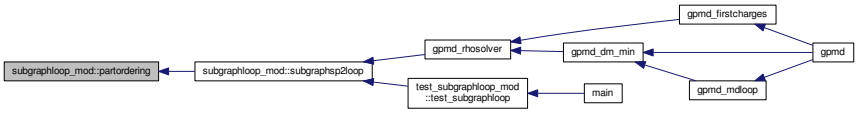
Set row ordering bases on parts.

Parameters

gp	Graph partitioning

Definition at line 283 of file subgraphloop_mod.F90.

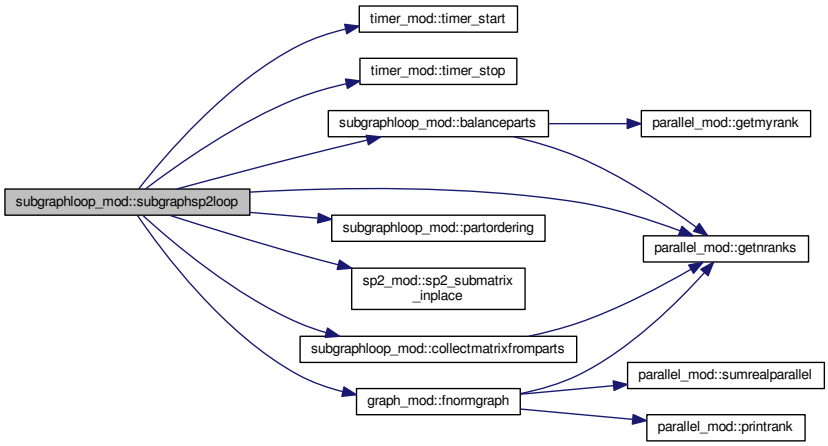
Here is the caller graph for this function:



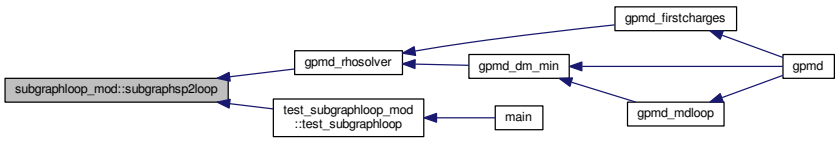
11.34.2.6 subroutine, public subgraphloop_mod::subgraphsp2loop (type (bml_matrix_t), intent(in) *h_bml,* type (bml_matrix_t), intent(in) *g_bml,* type (bml_matrix_t), intent(inout) *rho_bml,* type (graph_partitioning_t), intent(inout) *gp,* real(dp), intent(in) *threshold*)

Definition at line 57 of file subgraphloop_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.34.3 Member Data Documentation

11.34.3.1 integer, parameter subgraphloop_mod::dp = kind(1.0d0) [private]

Definition at line 39 of file subgraphloop_mod.F90.

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

- /home/christian/qmd-progress/src/subgraphloop_mod.F90

11.35 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.

Public Member Functions

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

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

  The parser for the chemical system.
- subroutine, public write_system (system, filename, extension)

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

  Write trajectory in .xyz, .dat or pdb file.
- subroutine, public make_random_system (system, nats, seed, lx, ly, lz)

  Make random Xx system.
- subroutine, public get_origin (coords, origin)

  Get the origin of the coordinates.
- subroutine, public translateandfoldtobox (coords, lattice_vectors, origin)

  Translate and fold to box.
- subroutine, public get_recip_vects (lattice_vectors, recip_vectors, volr, volk)

  Get the volume of the cell and the reciprocal vectors: This soubroutine computes:
- subroutine, public get_covgraph (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)

  Get the covalency graph in bml format.
- subroutine, public get_covgraph_h (sy, nnStructMindist, nnStruct, nrnnstruct, rcut, graph_h, mdimin, verbose)

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

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

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

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

  Get partial subgraph based on the Density matrix.

- subroutine, public 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 merge_graph (graph_p, graph_h)

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

  Get partial subgraph based on the Density matrix.
- subroutine, public adj2bml (xadj, adjncy, bml_type, g_bml)

  adj2bml
- subroutine, public graph2bml (graph, bml_type, g_bml)

  Graph2bml.
- subroutine, public graph2vector (graph, vector, maxnz)

  Vectorize graph.
- subroutine, public vector2graph (vector, graph, maxnz)

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

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

Private Member Functions

- subroutine parameters_to_vectors (abc_angles, lattice_vector)

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

  Transforms the lattice vectors into lattice parameters.
- subroutine get_covgraph_int (sy, nnStructMindist, nnStruct, nrnnstruct, bml_type, factor, gcov_bml, mdimin, verbose)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.35.1 Detailed Description

A module to read and handle chemical systems.

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

Author

C. F. A. Negre (cnegre@lanl.gov)

Definition at line 8 of file system_mod.F90.

11.35.2 Member Function/Subroutine Documentation

11.35.2.1 subroutine, public system_mod::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*)

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 1766 of file system_mod.F90.

11.35.2.2 subroutine, public system_mod::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*)

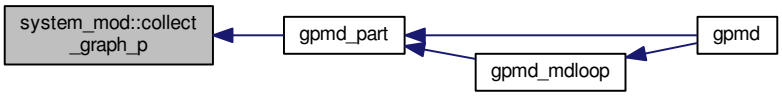
Collect the small graph to build the full graph.

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 1549 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.3 subroutine, public system_mod::destroy_subsystems (type(system_type), intent(inout) *sbsy,* integer, intent(in), optional *verbose*)

Destroy allocated subsystem.

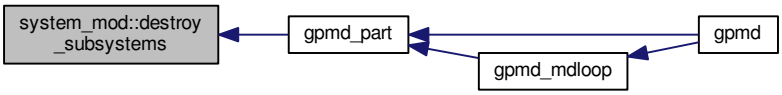
This routine will deallocate all the arrays of the structures.

Parameters

sy	System to de deallocated (see system_type).

Definition at line 1334 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.4 subroutine, public system_mod::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.

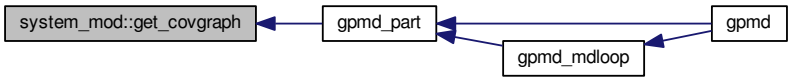
This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

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 1049 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.5 subroutine, public system_mod::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.

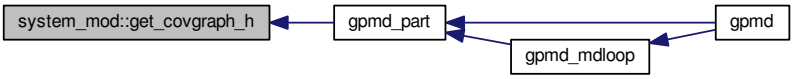
This is the graph composed by the covalent bonds (edges) that are determined with the VDW radius.

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 1181 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.6 subroutine system_mod::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 1123 of file system_mod.F90.

11.35.2.7 subroutine, public system_mod::get_nameandext (character(30), intent(in) *fullfilename,* character(30), intent(inout) *filename,* character(3), intent(inout) *ext*)

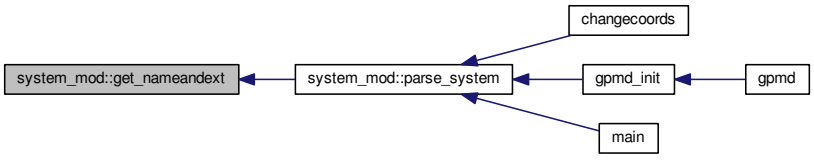
Get the name and extension of a file.

Parameters

fullfilename	Full filename.
filename	Filename of the system.
extension	Extension of the file.

Definition at line 204 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.8 subroutine, public system_mod::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 903 of file system_mod.F90.

11.35.2.9 subroutine, public system_mod::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 1483 of file system_mod.F90.

11.35.2.10 subroutine, public system_mod::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:

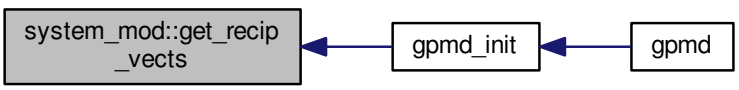
- $b_1 = \frac{1}{V_c} a_1 \times a_2$
- $b_2 = \frac{1}{V_c} a_2 \times a_3$
- $b_3 = \frac{1}{V_c} a_3 \times a_1$
- $V_c = ||a_1 \cdot (a_2 \times a_3)||$
- $V_{BZ} = ||b_1 \cdot (b_2 \times b_3)||$

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 997 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.11 subroutine, public system_mod::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.

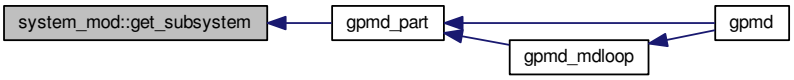
This will get a subsystem from the total system guided by a partition.

Parameters

sy	System structure (see system_type).
lsize	Core+Hallo subsystem size.
indices	Partition indices.
sbsy	Subsystem to be extracted.

Definition at line 1247 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.12 subroutine, public system_mod::graph2bml (integer, dimension(:,:), intent(inout), allocatable *graph,* character(20), intent(in) *bml_type,* type(bml_matrix_t), intent(inout) *g_bml*)

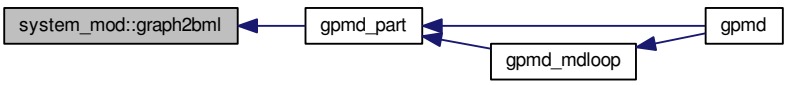
Graph2bml.

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 1800 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.13 subroutine, public system_mod::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 1843 of file system_mod.F90.

11.35.2.14 subroutine, public system_mod::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 778 of file system_mod.F90.

11.35.2.15 subroutine, public system_mod::merge_graph (integer, dimension(:,:), intent(inout) *graph_p,* integer, dimension(:,:), intent(inout) *graph_h*)

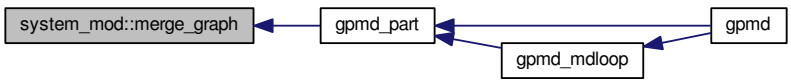
Get partial subgraph based on the Density matrix.

Parameters

graph_p	Density matix based graph in bml format.
graph_h	Hamiltonian matix based graph in bml format.

Definition at line 1641 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.16 subroutine, public system_mod::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 1692 of file system_mod.F90.

11.35.2.17 subroutine, public system_mod::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*)

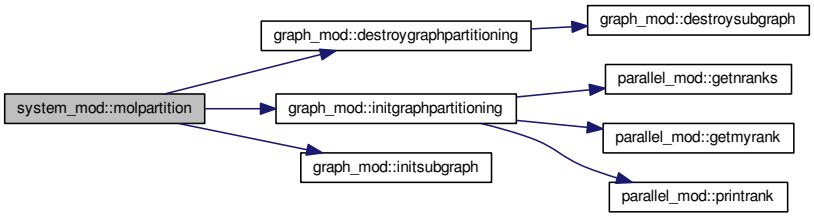
Partition by molecule.

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 1396 of file system_mod.F90.

Here is the call graph for this function:



11.35.2.18 subroutine system_mod::parameters_to_vectors (real(dp), dimension(2,3), intent(in) *abc_angles,* real(dp), dimension(3,3), intent(out) *lattice_vector*) [private]

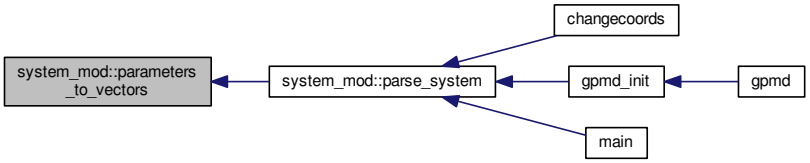
Transforms the lattice parameters into lattice vectors.

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,:) = $\vec{a}$

Definition at line 825 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.19 subroutine, public system_mod::parse_system (type(system_type), intent(out) *system,* character(len=∗) *filename,* character(3), intent(in), optional *extin*)

The parser for the chemical system.

Parameters

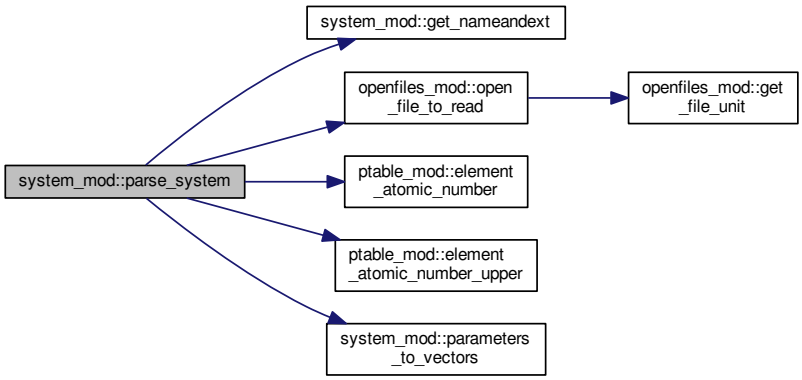
system	System to be constructed.
filename	Filename of the system.
extin	Extension of the file.

Assignment of species index for every atom.

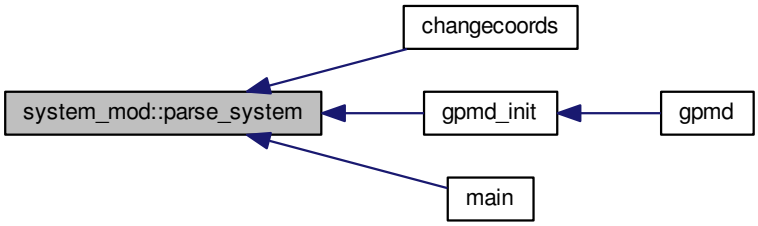
Todo Integrate this loop in the loop for building the splist.

Definition at line 227 of file system_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.35.2.20 subroutine, public system_mod::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 1900 of file system_mod.F90.

11.35.2.21 subroutine, public system_mod::translateandfoldtobox (real(dp), dimension(:,:), intent(inout), allocatable *coords,* real(dp), dimension(:,:), intent(in) *lattice_vectors,* real(dp), dimension(:), intent(inout), allocatable *origin*)

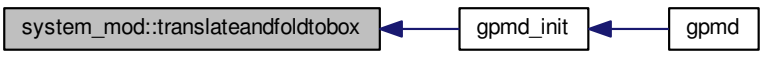
Translate and fold to box.

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 940 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.22 subroutine, public system_mod::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 1872 of file system_mod.F90.

11.35.2.23 subroutine system_mod::vectors_to_parameters (real(dp), dimension(3,3), intent(in) *lattice_vector,* real(dp), dimension(2,3), intent(out) *abc_angles*) [private]

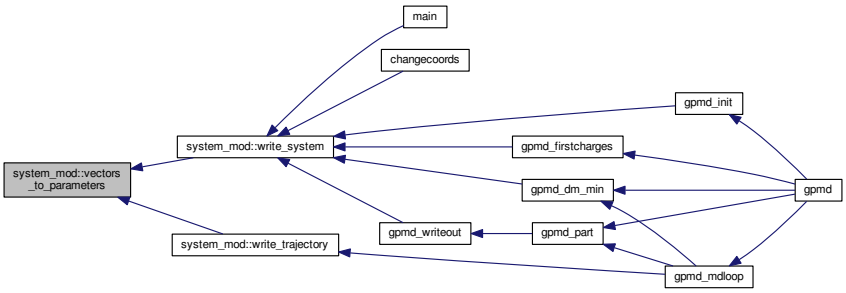
Transforms the lattice vectors into lattice parameters.

Parameters

lattice_vector	3x3 array containing the lattice vectors. lattice_vector(1,:) = $\vec{a}$
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 867 of file system_mod.F90.

Here is the caller graph for this function:



11.35.2.24 subroutine, public system_mod::write_system (type(system_type), intent(in) *system,* character(∗) *filename,* character(3) *extension*)

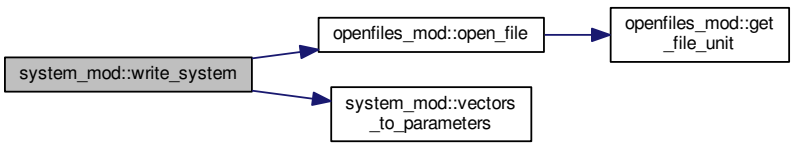
Write system in .xyz, .dat or pdb file.

Parameters

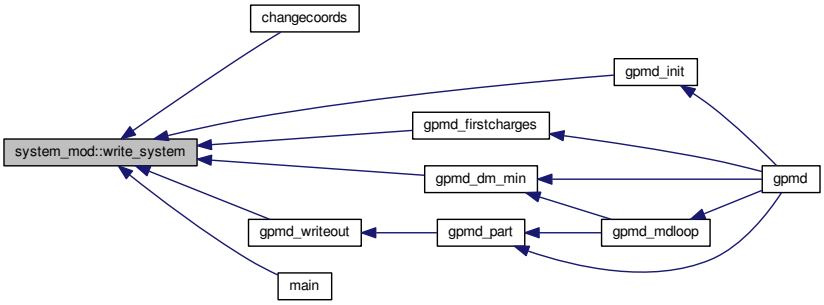
system	System to be constructed.
filename	File name.
extension	Extension of the file.

Definition at line 520 of file system_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.35.2.25 subroutine, public system_mod::write_trajectory (type(system_type), intent(in) *system,* integer, intent(in) *iter,* integer, intent(in) *each,* real(dp), intent(in) *deltat,* character(∗) *filename,* character(3) *extension*)

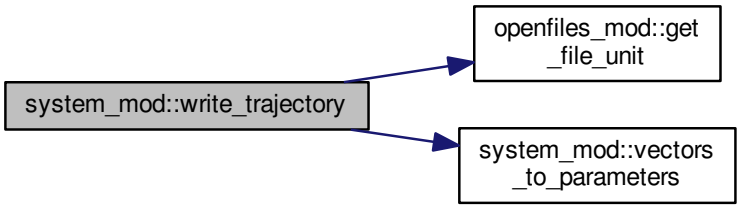
Write trajectory in .xyz, .dat or pdb file.

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 656 of file system_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

11.35.3 Member Data Documentation

11.35.3.1 integer, parameter system_mod::dp = kind(1.0d0) [private]

Definition at line 19 of file system_mod.F90.

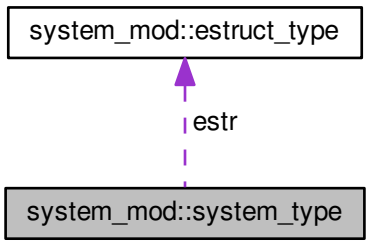
The documentation for this module was generated from the following file:

- /home/christian/qmd-progress/src/system_mod.F90

11.36 system_mod::system_type Type Reference

System type.

Collaboration diagram for system_mod::system_type:



Public Attributes

- integer nats

  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 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 constrain

  User define field.
- integer, dimension(:), allocatable resindex

  Residue index.
- type(estruct_type) estr

  Electronic structure.

11.36.1 Detailed Description

System type.

The molecular system type.

Definition at line 76 of file system_mod.F90.

11.36.2 Member Data Documentation

11.36.2.1 integer, dimension(:), allocatable system_mod::system_type::atomic_number

Atomic number for every atom in the system.

Definition at line 90 of file system_mod.F90.

11.36.2.2 real(dp), dimension(:), allocatable system_mod::system_type::constrain

User define field.

Definition at line 182 of file system_mod.F90.

11.36.2.3 real(dp), dimension(:,:), allocatable system_mod::system_type::coordinate

Coordinates of every atom in the system. Allocation:

```
coordinate(3,nats)
```

Definition at line 95 of file system_mod.F90.

11.36.2.4 type(estruct_type) system_mod::system_type::estr

Electronic structure.

Definition at line 188 of file system_mod.F90.

11.36.2.5 real(dp), dimension(:,:), allocatable system_mod::system_type::force

Forces acting on every atom in the system. Allocation:

```
force(3,nats)
```

Definition at line 105 of file system_mod.F90.

11.36.2.6 real(dp), dimension(:,:), allocatable system_mod::system_type::lattice_vector

Lattice vectors of the system. Use the 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 127 of file system_mod.F90.

11.36.2.7 real(dp), dimension(:), allocatable 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 117 of file system_mod.F90.

11.36.2.8 integer system_mod::system_type::nats

Number of atoms of the system.

Definition at line 79 of file system_mod.F90.

11.36.2.9 real(dp), dimension(:), allocatable system_mod::system_type::net_charge

Charges of every atom in the system. Allocation:

```
net_charge(nats)
```

Definition at line 110 of file system_mod.F90.

11.36.2.10 integer 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 150 of file system_mod.F90.

11.36.2.11 real(dp), dimension(:,:), allocatable 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 135 of file system_mod.F90.

11.36.2.12 integer, dimension(:), allocatable system_mod::system_type::resindex

Residue index.

Definition at line 185 of file system_mod.F90.

11.36.2.13 integer, dimension(:), allocatable system_mod::system_type::spatnum

Species atomic number list. A list with the atomic numbers for every species Allocation:

```
spatnum(nsp)
```

Definition at line 173 of file system_mod.F90.

11.36.2.14 integer, dimension(:), allocatable 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 158 of file system_mod.F90.

11.36.2.15 character(2), dimension(:), allocatable 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 166 of file system_mod.F90.

11.36.2.16 real(dp), dimension(:), allocatable system_mod::system_type::spmass

Species mass list. A list with the atomic mass for every species Allocation:

```
spmass(nsp)
```

Definition at line 179 of file system_mod.F90.

11.36.2.17 character(2), dimension(:), allocatable 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 87 of file system_mod.F90.

11.36.2.18 real(dp), dimension(:,:), allocatable system_mod::system_type::velocity

Velocities for every atom in the system. Allocation:

```
velocity(3,nats)
```

Definition at line 100 of file system_mod.F90.

11.36.2.19 real(dp) system_mod::system_type::volk

Volume of the system (direct space).

Note

use get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 143 of file system_mod.F90.

11.36.2.20 real(dp) system_mod::system_type::volr

Volume of the system (direct space).

Note

use get_recip_vects in coulomb_latte_mod to compute this.

Definition at line 139 of file system_mod.F90.

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

- /home/christian/qmd-progress/src/system_mod.F90

11.37 test_subgraphloop_mod Module Reference

The test_subgraphloop module.

Public Member Functions

- subroutine, public test_subgraphloop (h_bml, rho_bml, sthreshold, bndfil, minsp2iter, maxsp2iter, sp2conv, idemtol, gthreshold, errlimit, nodesPerPart)

Private Attributes

- integer, parameter dp = kind(1.0d0)

11.37.1 Detailed Description

The test_subgraphloop module.

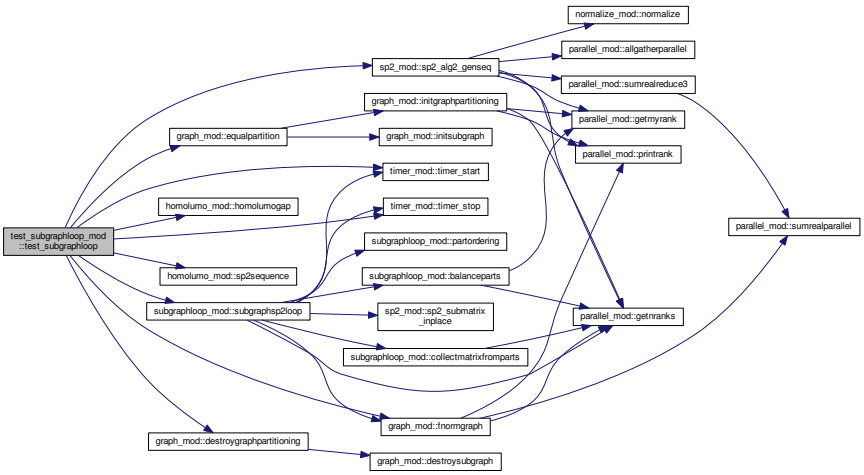
Definition at line 25 of file test_subgraphloop.F90.

11.37.2 Member Function/Subroutine Documentation

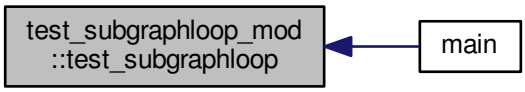
11.37.2.1 subroutine, public test_subgraphloop_mod::test_subgraphloop (type(bml_matrix_t), intent(in) *h_bml,* type(bml_matrix_t), intent(inout) *rho_bml,* real(dp), intent(in) *sthreshold,* real(dp), intent(in) *bndfil,* integer, intent(in) *minsp2iter,* integer, intent(in) *maxsp2iter,* character(len=*), intent(in) *sp2conv,* real(dp), intent(in) *idemtol,* real(dp), intent(in) *gthreshold,* real(dp), intent(in) *errlimit,* integer, intent(in) *nodesPerPart*)

Definition at line 59 of file test_subgraphloop.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.37.3 Member Data Documentation

11.37.3.1 integer, parameter test_subgraphloop_mod::dp = kind(1.0d0) [private]

Definition at line 39 of file test_subgraphloop.F90.

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

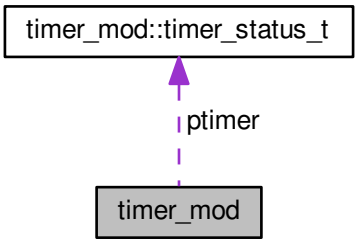
- /home/christian/qmd-progress/tests/src/test_subgraphloop.F90

11.38 timer_mod Module Reference

The timer module.

Example use of dynamic timing:

Collaboration diagram for timer_mod:



Data Types

- type timer_status_t

  Timer status type.

Public Member Functions

- subroutine, public timer_init ()

  Initialize timers.
- subroutine, public timer_shutdown ()

  Done with timers.
- subroutine, public timer_start (itimer, tag)

  Start Timing.
- subroutine, public timer_stop (itimer, verbose)

  Stop timing.
- subroutine, public timer_collect ()
- subroutine, public timer_results ()
- real(8) function, public time2milliseconds ()
- subroutine, public print_date_and_time (tag)

Public Attributes

- 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

Private Member Functions

- subroutine timer_getid ()

  Get timer id.
- character(2) function, private int2char (ival)

Private Attributes

- integer, parameter dp = kind(1.0d0)
- integer tstart_clock
- integer tstop_clock
- integer tclock_rate
- integer tclock_max
- integer num_timers
- type(timer_status_t), dimension(:), allocatable ptimer

11.38.1 Detailed Description

The timer module.

Example use of dynamic timing:

call timer_init()

call timer_start(dyn_timer,"timer_tag")

.... code lines ...

call timer_stop(dyn_timer,1)

This will write the time it takes to execute "code lines" and it will name it "timer_tag"

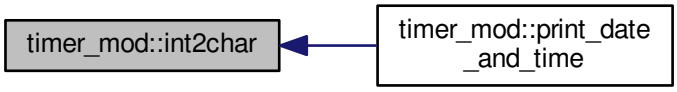
Definition at line 42 of file timer_mod.F90.

11.38.2 Member Function/Subroutine Documentation

11.38.2.1 character(2) function, private timer_mod::int2char (integer, intent(in) *ival*) [private]

Definition at line 411 of file timer_mod.F90.

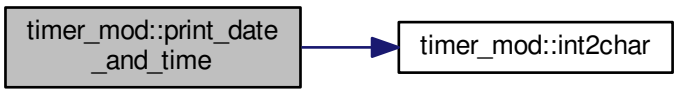
Here is the caller graph for this function:



11.38.2.2 subroutine, public timer_mod::print_date_and_time (character(len=*), intent(in) *tag*)

Definition at line 388 of file timer_mod.F90.

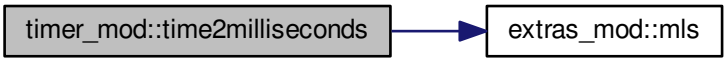
Here is the call graph for this function:



11.38.2.3 real(8) function, public timer_mod::time2milliseconds ( )

Definition at line 377 of file timer_mod.F90.

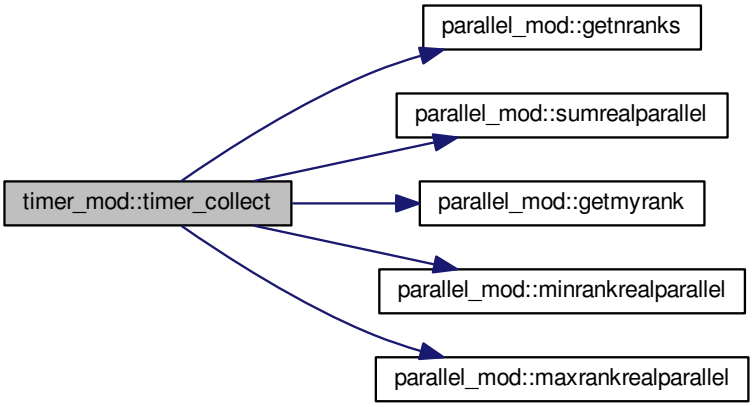
Here is the call graph for this function:



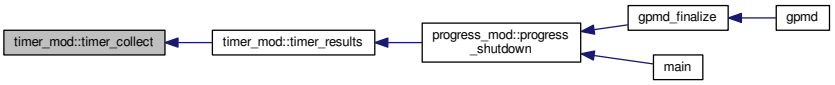
11.38.2.4 subroutine, public timer_mod::timer_collect ( )

Definition at line 270 of file timer_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.38.2.5 subroutine timer_mod::timer_getid ( ) [private]

Get timer id.

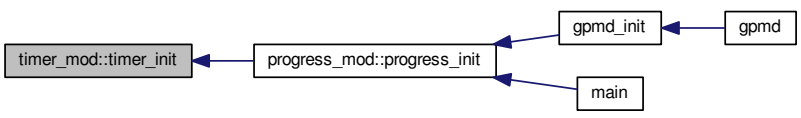
Definition at line 217 of file timer_mod.F90.

11.38.2.6 subroutine, public timer_mod::timer_init ( )

Initialize timers.

Definition at line 149 of file timer_mod.F90.

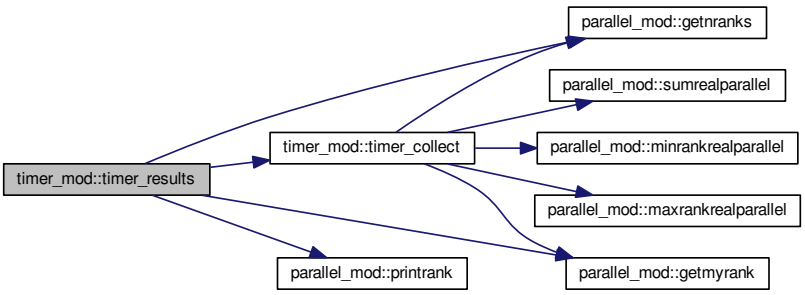
Here is the caller graph for this function:



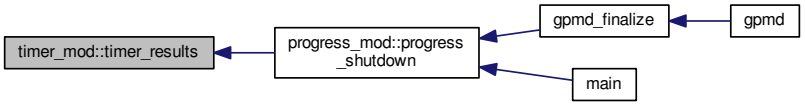
11.38.2.7 subroutine, public timer_mod::timer_results ( )

Definition at line 334 of file timer_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

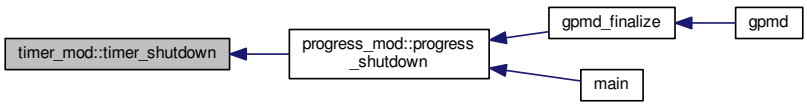


11.38.2.8 subroutine, public timer_mod::timer_shutdown ( )

Done with timers.

Definition at line 222 of file timer_mod.F90.

Here is the caller graph for this function:



11.38.2.9 subroutine, public timer_mod::timer_start (integer, intent(in) *itimer,* character(len=∗), intent(in), optional *tag*)

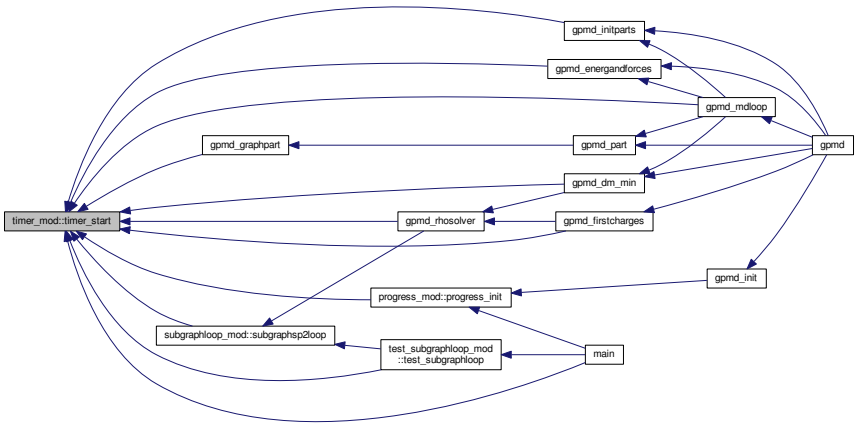
Start Timing.

Parameters

itimer	The index of the timer to start.
tag	Optional parameter to retag the timer on the fly.

Definition at line 232 of file timer_mod.F90.

Here is the caller graph for this function:



11.38.2.10 subroutine, public timer_mod::timer_stop (integer, intent(in) *itimer,* integer, intent(in), optional *verbose*)

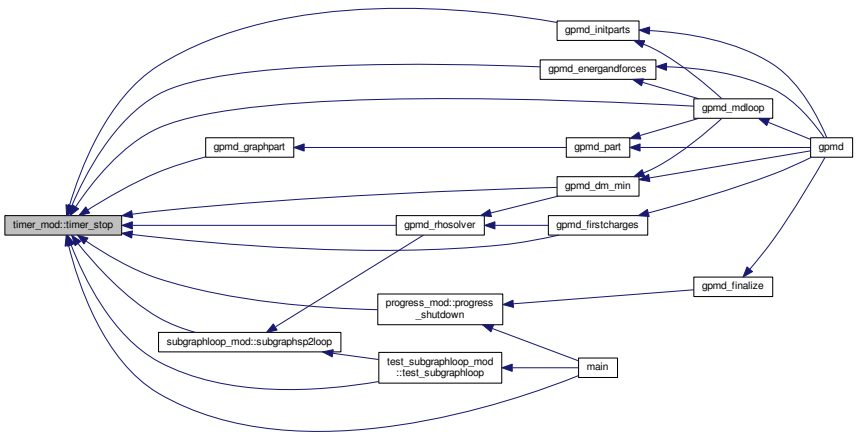
Stop timing.

Parameters

itimer	The index of the timer to stop.
verbose	Optional parameters to print partial times.

Definition at line 250 of file timer_mod.F90.

Here is the caller graph for this function:



11.38.3 Member Data Documentation

11.38.3.1 integer, public timer_mod::badd_timer

Definition at line 66 of file timer_mod.F90.

11.38.3.2 integer, public timer_mod::bmult_timer

Definition at line 66 of file timer_mod.F90.

11.38.3.3 integer, public timer_mod::buildz_timer

Definition at line 67 of file timer_mod.F90.

11.38.3.4 integer, public timer_mod::deortho_timer

Definition at line 63 of file timer_mod.F90.

11.38.3.5 integer, parameter timer_mod::dp = kind(1.0d0) [private]

Definition at line 50 of file timer_mod.F90.

11.38.3.6 integer, public timer_mod::dyn_timer

Definition at line 67 of file timer_mod.F90.

11.38.3.7 integer, public timer_mod::genx_timer

Definition at line 62 of file timer_mod.F90.

11.38.3.8 integer, public timer_mod::graphsp2_timer

Definition at line 64 of file timer_mod.F90.

11.38.3.9 integer, public timer_mod::halfverlet_timer

Definition at line 69 of file timer_mod.F90.

11.38.3.10 integer, public timer_mod::loop_timer

Definition at line 62 of file timer_mod.F90.

11.38.3.11 integer, public timer_mod::mdloop_timer

Definition at line 67 of file timer_mod.F90.

11.38.3.12 integer, public timer_mod::nlist_timer

Definition at line 69 of file timer_mod.F90.

11.38.3.13 integer timer_mod::num_timers [private]

Definition at line 140 of file timer_mod.F90.

11.38.3.14 integer, public timer_mod::ortho_timer

Definition at line 64 of file timer_mod.F90.

11.38.3.15 integer, public timer_mod::pairpot_timer

Definition at line 68 of file timer_mod.F90.

11.38.3.16 integer, public timer_mod::part_timer

Definition at line 63 of file timer_mod.F90.

11.38.3.17 integer, public timer_mod::pos_timer

Definition at line 69 of file timer_mod.F90.

11.38.3.18 type (timer_status_t), dimension(:), allocatable timer_mod::ptimer [private]

Definition at line 142 of file timer_mod.F90.

11.38.3.19 integer, public timer_mod::realcoul_timer

Definition at line 68 of file timer_mod.F90.

11.38.3.20 integer, public timer_mod::recipcoul_timer

Definition at line 68 of file timer_mod.F90.

11.38.3.21 integer, public timer_mod::sp2_timer

Definition at line 62 of file timer_mod.F90.

11.38.3.22 integer, public timer_mod::suball_timer

Definition at line 66 of file timer_mod.F90.

11.38.3.23 integer, public timer_mod::subext_timer

Definition at line 65 of file timer_mod.F90.

11.38.3.24 integer, public timer_mod::subgraph_timer

Definition at line 63 of file timer_mod.F90.

11.38.3.25 integer, public timer_mod::subind_timer

Definition at line 65 of file timer_mod.F90.

11.38.3.26 integer, public timer_mod::subsp2_timer

Definition at line 65 of file timer_mod.F90.

11.38.3.27 integer timer_mod::tclock_max [private]

Definition at line 139 of file timer_mod.F90.

11.38.3.28 integer timer_mod::tclock_rate [private]

Definition at line 139 of file timer_mod.F90.

11.38.3.29 integer timer_mod::tstart_clock [private]

Definition at line 139 of file timer_mod.F90.

11.38.3.30 integer timer_mod::tstop_clock [private]

Definition at line 139 of file timer_mod.F90.

11.38.3.31 integer, public timer_mod::zdiag_timer

Definition at line 64 of file timer_mod.F90.

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

- /home/christian/qmd-progress/src/timer_mod.F90

11.39 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.

11.39.1 Detailed Description

Timer status type.

Definition at line 72 of file timer_mod.F90.

11.39.2 Member Data Documentation

11.39.2.1 integer timer_mod::timer_status_t::maxrank [private]

Rank with max value.

Definition at line 90 of file timer_mod.F90.

11.39.2.2 real(dp) timer_mod::timer_status_t::maxvalue [private]

Maximum value over all ranks.

Definition at line 99 of file timer_mod.F90.

11.39.2.3 integer timer_mod::timer_status_t::minrank [private]

Rank with min value.

Definition at line 87 of file timer_mod.F90.

11.39.2.4 real(dp) timer_mod::timer_status_t::minvalue [private]

Minimum value over all ranks.

Definition at line 96 of file timer_mod.F90.

11.39.2.5 real(dp) timer_mod::timer_status_t::tavg [private]

Average value over all ranks.

Definition at line 102 of file timer_mod.F90.

11.39.2.6 integer timer_mod::timer_status_t::tcount [private]

Current call count.

Definition at line 84 of file timer_mod.F90.

11.39.2.7 character(len=20) timer_mod::timer_status_t::tname [private]

Timer name.

Definition at line 75 of file timer_mod.F90.

11.39.2.8 real(dp) timer_mod::timer_status_t::tpercent [private]

Percent of time across all timers.

Definition at line 108 of file timer_mod.F90.

11.39.2.9 integer timer_mod::timer_status_t::tstart [private]

Start time.

Definition at line 78 of file timer_mod.F90.

11.39.2.10 real(dp) timer_mod::timer_status_t::tstdev [private]

Stdev across all ranks.

Definition at line 105 of file timer_mod.F90.

11.39.2.11 real(dp) timer_mod::timer_status_t::tsum [private]

Sum time - total time in secs.

Definition at line 93 of file timer_mod.F90.

11.39.2.12 integer timer_mod::timer_status_t::ttotal [private]

Current total time.

Definition at line 81 of file timer_mod.F90.

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

- /home/christian/qmd-progress/src/timer_mod.F90

11.40 variable_names Module Reference

Glossary for the code.

Public Attributes

- integer norb

  Vector containing a set of eigenvalues.
- integer nocc

Number of occupied orbitals.

- integer nats

  Number of atoms.
- real(dp) threshold

  Threshold value for sparse matrices.
- type(bml_matrix_t) eigenvectors_bml

  Coefficient matrix in bml format.
- character(20) bml_type

  Character containing the bml format type.
- type(system_type) system

  System type containing coordinates, atom types, velocities, etc.
- type(bml_matrix_t) ham_bml

  Hamiltonian in bml format.
- type(bml_matrix_t) rho_bml

  Density matrix in bml format.
- type(bml_matrix_t) zmat_bml

  Inverse overlap factor in bml format.
- type(bml_matrix_t) occupation_bml

  Diagonal matrix containing the occupations.
- type(bml_matrix_t) aux_bml

  Auxiliary bml matrix.
- type(bml_matrix_t) mat_bml

  A general bml matrix.
- integer verbose

  Integer caring different levels of verbosity.
- character(3) extension

  File extension ("xyz" or "pdb").
- character(3) ext

  File extension ("xyz" or "pdb").
- character(20) filename

  Variable to contain a file name.
- integer io_unit

  Input unit Number to write or read.
- character(20) dummy

11.40.1 Detailed Description

Glossary for the code.

Definition at line 10 of file doxy_mod.F90.

11.40.2 Member Data Documentation

11.40.2.1 type(bml_matrix_t) variable_names::aux_bml

Auxiliary bml matrix.

Definition at line 49 of file doxy_mod.F90.

11.40.2.2 character(20) variable_names::bml_type

Character containing the bml format type.

Definition at line 31 of file doxy_mod.F90.

11.40.2.3 character(20) variable_names::dummy

Definition at line 70 of file doxy_mod.F90.

11.40.2.4 type(bml_matrix_t) variable_names::eigenvectors_bml

Coefficient matrix in bml format.

Definition at line 28 of file doxy_mod.F90.

11.40.2.5 character(3) variable_names::ext

File extension ("xyz" or "pdb").

Definition at line 61 of file doxy_mod.F90.

11.40.2.6 character(3) variable_names::extension

File extension ("xyz" or "pdb").

Definition at line 58 of file doxy_mod.F90.

11.40.2.7 character(20) variable_names::filename

Variable to contain a file name.

Definition at line 64 of file doxy_mod.F90.

11.40.2.8 type(bml_matrix_t) variable_names::ham_bml

Hamiltonian in bml format.

Definition at line 37 of file doxy_mod.F90.

11.40.2.9 integer variable_names::io_unit

Input unit Number to write or read.

Definition at line 67 of file doxy_mod.F90.

11.40.2.10 type(bml_matrix_t) variable_names::mat_bml

A general bml matrix.

Definition at line 52 of file doxy_mod.F90.

11.40.2.11 integer variable_names::nats

Number of atoms.

Definition at line 22 of file doxy_mod.F90.

11.40.2.12 integer variable_names::nocc

Number of occupied orbitals.

Definition at line 19 of file doxy_mod.F90.

11.40.2.13 integer variable_names::norb

Vector containing a set of eigenvalues.

Number of orbitals. Its usually the dimension of the Hamiltonian.

Definition at line 16 of file doxy_mod.F90.

11.40.2.14 type(bml_matrix_t) variable_names::occupation_bml

Diagonal matrix containing the occupations.

Definition at line 46 of file doxy_mod.F90.

11.40.2.15 type(bml_matrix_t) variable_names::rho_bml

Density matrix in bml format.

Definition at line 40 of file doxy_mod.F90.

11.40.2.16 type(system_type) variable_names::system

System type containing coordinates, atom types, velocities, etc.

Definition at line 34 of file doxy_mod.F90.

11.40.2.17 real(dp) variable_names::threshold

Threshold value for sparse matrices.

Definition at line 25 of file doxy_mod.F90.

11.40.2.18 integer variable_names::verbose

Integer caring different levels of verbosity.

Definition at line 55 of file doxy_mod.F90.

11.40.2.19 type(bml_matrix_t) variable_names::zmat_bml

Inverse overlap factor in bml format.

Definition at line 43 of file doxy_mod.F90.

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

- /home/christian/qmd-progress/src/doxy_mod.F90

11.41 xlbo_mod Module Reference

A module to perform XLBO integration.

Data Types

- type xlbo_type

  General xlbo solver type.

Public Member Functions

- subroutine, public parse_xlbo (xlbo, filename)

  The parser for XLBO parser.
- subroutine, public 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 xlbo_fcoulupdate (fcoul, charges, n)

  Adjust forces for the linearized XLBOMD functional.

Private Attributes

- 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

11.41.1 Detailed Description

A module to perform XLBO integration.

This module will be used to compute integrate the dynamical variable "n" in xlbo.

Definition at line 6 of file xlbo_mod.F90.

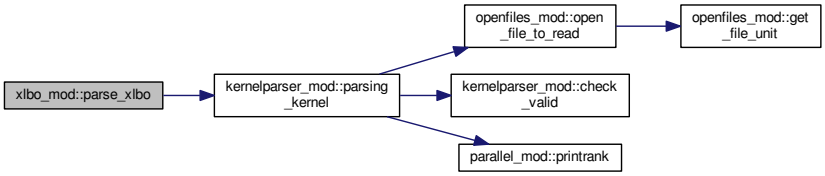
11.41.2 Member Function/Subroutine Documentation

11.41.2.1 subroutine, public xlbo_mod::parse_xlbo (type(xlbo_type), intent(inout) *xlbo,* character(len=∗) *filename*)

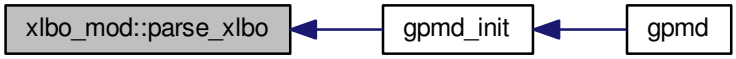
The parser for XLBO parser.

Definition at line 61 of file xlbo_mod.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



11.41.2.2 subroutine, public xlbo_mod::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 157 of file xlbo_mod.F90.

Here is the caller graph for this function:

11.41.2.3 subroutine, public xlbo_mod::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 117 of file xlbo_mod.F90.

Here is the caller graph for this function:

11.41.3 Member Data Documentation

11.41.3.1 real(dp), parameter xlbo_mod::alpha = 0.018_dp [private]

Definition at line 28 of file xlbo_mod.F90.

11.41.3.2 real(dp), parameter xlbo_mod::c0 = -6.0_dp [private]

Coefficients for modified Verlet integration.

Definition at line 19 of file xlbo_mod.F90.

11.41.3.3 real(dp), parameter xlbo_mod::c1 = 14.0_dp [private]

Definition at line 20 of file xlbo_mod.F90.

11.41.3.4 real(dp), parameter xlbo_mod::c2 = -8.0_dp [private]

Definition at line 21 of file xlbo_mod.F90.

11.41.3.5 real(dp), parameter xlbo_mod::c3 = -3.0_dp [private]

Definition at line 22 of file xlbo_mod.F90.

11.41.3.6 real(dp), parameter xlbo_mod::c4 = 4.0_dp [private]

Definition at line 23 of file xlbo_mod.F90.

11.41.3.7 real(dp), parameter xlbo_mod::c5 = -1.0_dp [private]

Definition at line 24 of file xlbo_mod.F90.

11.41.3.8 real(dp), parameter xlbo_mod::cc = 0.9_dp [private]

Definition at line 29 of file xlbo_mod.F90.

11.41.3.9 integer, parameter xlbo_mod::dp = kind(1.0d0) [private]

Definition at line 16 of file xlbo_mod.F90.

11.41.3.10 real(dp), parameter xlbo_mod::kappa = 1.82_dp [private]

Coefficients for modified Verlet integration.

Definition at line 27 of file xlbo_mod.F90.

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

- /home/christian/qmd-progress/src/xlbo_mod.F90

11.42 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_init MD steps.
- real(dp) cc

  Scaled delta Kernel.

11.42.1 Detailed Description

General xlbo solver type.

Definition at line 33 of file xlbo_mod.F90.

11.42.2 Member Data Documentation

11.42.2.1 real(dp) xlbo_mod::xlbo_type::cc

Scaled delta Kernel.

Definition at line 51 of file xlbo_mod.F90.

11.42.2.2 character(20) xlbo_mod::xlbo_type::jobname

Definition at line 35 of file xlbo_mod.F90.

11.42.2.3 integer xlbo_mod::xlbo_type::maxscfinititer

Max SCF iterations for the first minit steps.

Definition at line 43 of file xlbo_mod.F90.

11.42.2.4 integer xlbo_mod::xlbo_type::maxscfiter

Max SCF iterations at every XLBO MD step.

Definition at line 40 of file xlbo_mod.F90.

11.42.2.5 integer xlbo_mod::xlbo_type::minit

Use SCF the first M_init MD steps.

Definition at line 48 of file xlbo_mod.F90.

11.42.2.6 real(dp) xlbo_mod::xlbo_type::threshold

Definition at line 45 of file xlbo_mod.F90.

11.42.2.7 integer xlbo_mod::xlbo_type::verbose

Definition at line 37 of file xlbo_mod.F90.

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

- /home/christian/qmd-progress/src/xlbo_mod.F90

Chapter 12

File Documentation

12.1 /home/christian/qmd-progress/examplePrograms/changecoords/changecoords.F90 File Reference

Functions/Subroutines

- program changecoords

  High-level program to change coordinates formats.

12.2 /home/christian/qmd-progress/examplePrograms/gpmdcov/get_energy.py File Reference

Namespaces

- get_energy

Variables

- tuple get_energy.MyFileName = str(sys.argv[1])
- int get_energy.count = -1
- tuple get_energy.MyFile = open(MyFileName,"r")
- get_energy.Dim = count
- tuple get_energy.datos = numpy.zeros(Dim+1)
- tuple get_energy.lines_split = lines.split()

12.3 /home/christian/qmd-progress/examplePrograms/gpmdcov/gpmdcov.F90 File Reference

Functions/Subroutines

- program gpmd

  High-level program to perform GRAPH-BASED QMD.
- subroutine gpmd_init ()

  Main program driver.
- subroutine gpmd_part

Partition by systems.

- subroutine gpmd_initparts

  Initialize the partition.

- subroutine gpmd_firstcharges ()

  First Charge computation.

- subroutine gpmd_dm_min (Nr_SCF, nguess, mix)

  SCF loop.

- subroutine gpmd_energandforces (charges)
- subroutine gpmd_preparemd ()

  Preparing for MD.

- subroutine gpmd_mdloop ()

  Main MD loop This routine performs the MD loops up to "ls%mdsteps".

- subroutine gpmd_rhosolver (orthoh_bml, orthop_bml)

  Solver for computing the density matrix.

- subroutine gpmd_graphpart

  Graph partitioning subroutine.

- subroutine gpmd_buildz (overin_bml, zmatout_bml)
- subroutine gpmd_writeout ()

  To write output file or perform some analysis.

- subroutine gpmd_finalize ()

  Finalize gpmd.

- subroutine gpmd_dump ()

  Dump GPMD.

- subroutine gpmd_restart ()

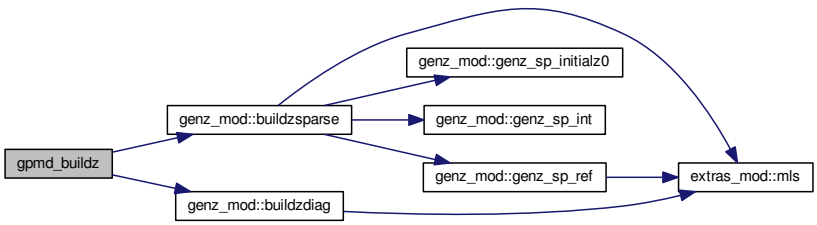
  Dump GPMD.

12.3.1 Function/Subroutine Documentation

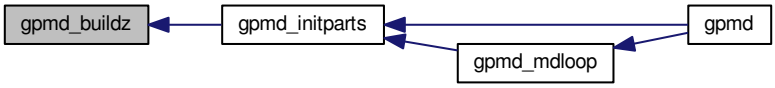
12.3.1.1 subroutine gpmd::gpmd_buildz (type(bml_matrix_t), intent(inout) *overin_bml,* type(bml_matrix_t), intent(inout) *zmatout_bml*)

Definition at line 1371 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.2 subroutine gpmd::gpmd_dm_min (integer *Nr_SCF,* real(dp), dimension(:), allocatable *nguess,* logical *mix*)

SCF loop.

Real contribution to the Coul energy. The outputs are coul_forces_r,coul_pot_r.

Reciprocal contribution to the Coul energy. The outputs are coul_forces_k,coul_pot_k.

Get Coulombic potential and charges for the part.

Get the scf hamiltonian. The outputs is ham_bml.

Orthogonalize ham.

Now sove for the desity matrix.

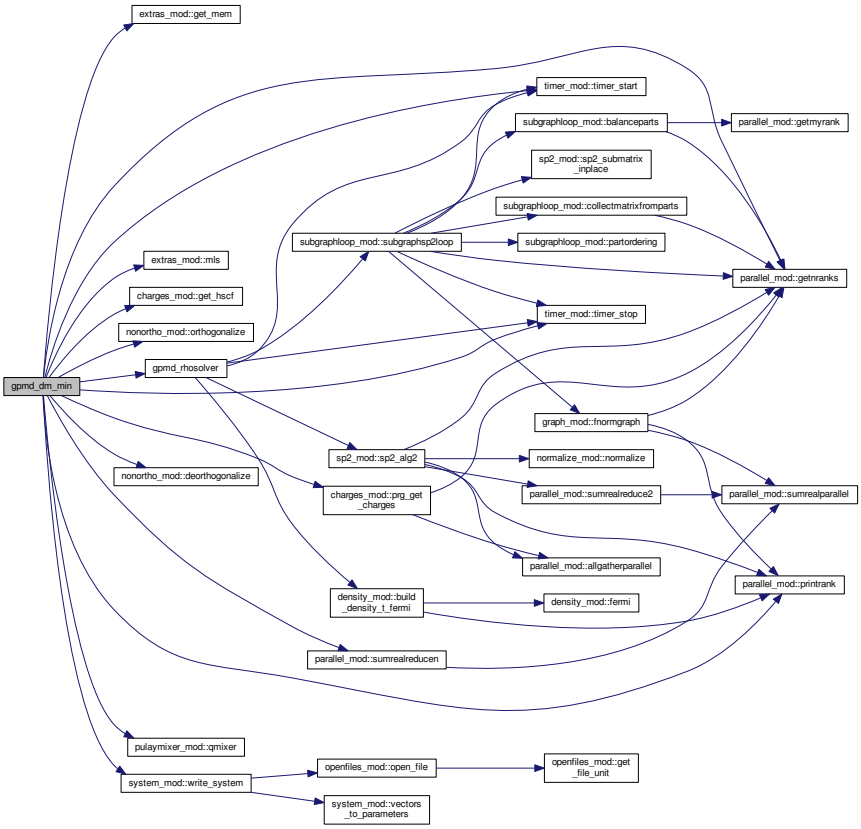
Deorthogonalize orthop_bml to get the density matrix rho_bml.

Get the system charges from rho

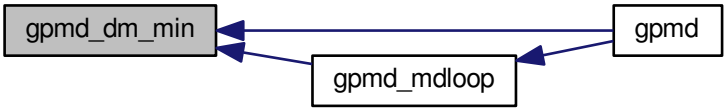
End of SCF loop.

Definition at line 601 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

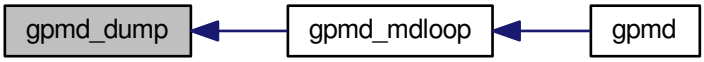


12.3.1.3 subroutine gpmd::gpmd_dump ( )

Dump GPMD.

Definition at line 1472 of file gpmdcov.F90.

Here is the caller graph for this function:



12.3.1.4 subroutine gpmd::gpmd_energandforces (real(dp), dimension(:), intent(in) *charges*)

Loop over all the parts

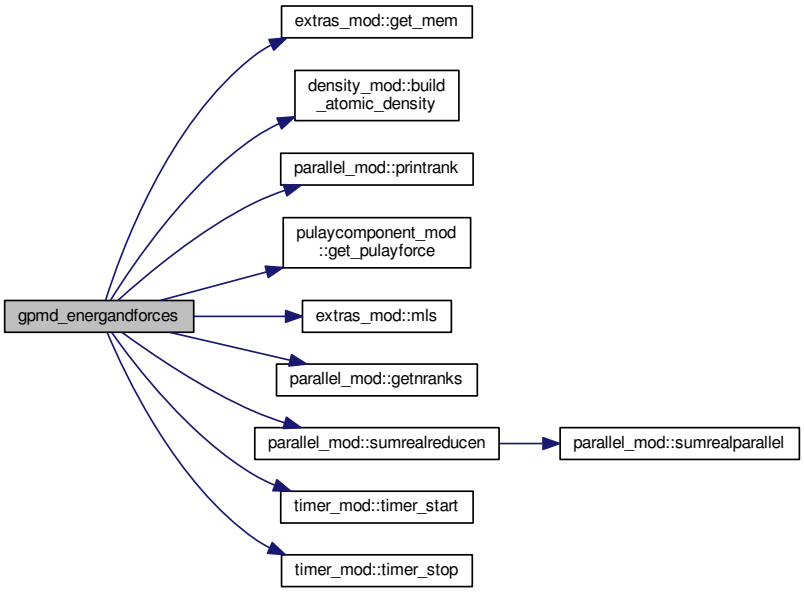
Get Electronic energy

Get Repulsive energy and forces

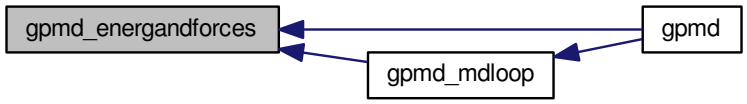
Get Coulombic energy

Definition at line 817 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

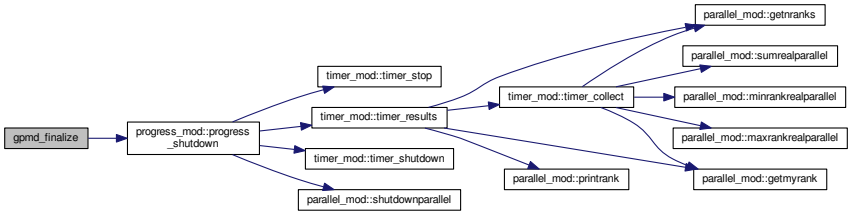


12.3.1.5 subroutine gpmd::gpmd_finalize ( )

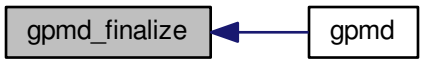
Finalize gpmd.

Definition at line 1453 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.6 subroutine gpmd::gpmd_firstcharges ( )

First Charge computation.

Here we compute the first "non-scf charges". Initialize the orthogonal versions of ham and rho.

Orthogonalize ham.

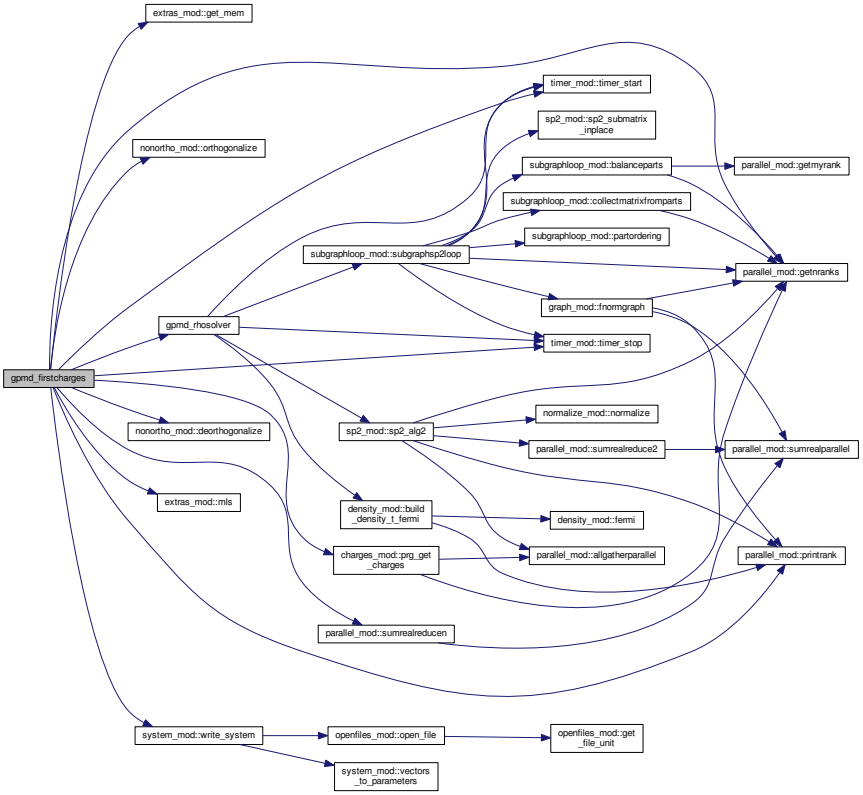
Deorthogonalize rho.

Get charges based on rho. rho_bml is the input and synet_charge is the outputs vector containing the charges.

Gather charges from all the parts.

Definition at line 500 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:

12.3.1.7 subroutine gpmd::gpmd_graphpart ( )

Graph partitioning subroutine.

Create graph partitioning - Use Block or METIS or METIS+SA or METIS+KL

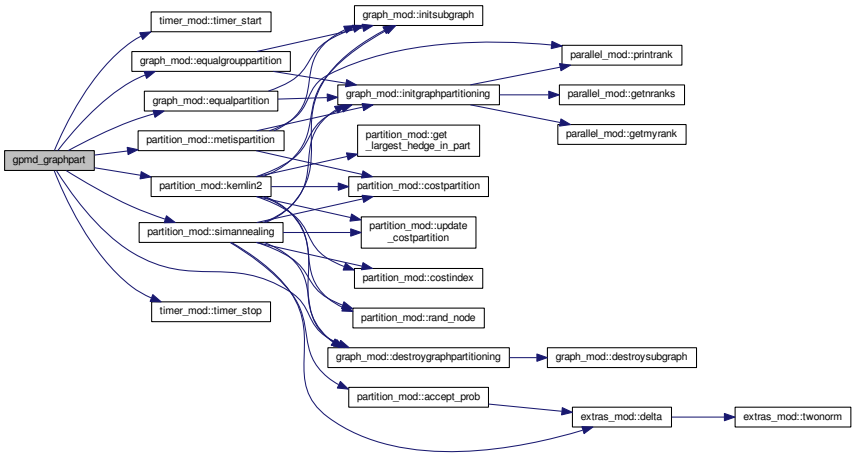
Block partitioning

Partition by orbital or atom

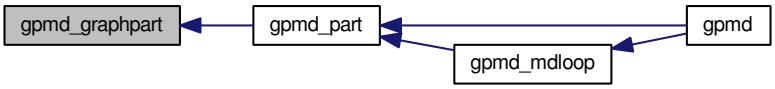
METIS, METIS+SA, or METIS+KL partitioning

Definition at line 1235 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.8 subroutine gpmd::gpmd_init ( )

Main program driver.

Initialize the program variables and parse input files. Initial partition of the system based on the covalency graph. This will need to be replace by a first SP2 algorithm to compute a first density matrix. Initialize partitions. Comput first charges. First SCF loop up to maxscf. First calculation of energies and forces. Setup the Molecular Dynamics (MD) calculation. Perform the MD simulation. Finalize the program. Initialize the program variables and parse input files. Start progress

Get MPI rank

Get the input file from argumets.

Parsing input file. This file contains all the variables needed to

Parsing SP2 input paramenters. This will read the variables in the input file.

Parsing GSP2 input paramenters. This will read the variables in the input file.

Parsing Extended Lagrangian input paramenters. This will read the variables in the input file.

Parsing Z sparse propagation.

Parsing system coordinates. This reads the coords.pdb file to get the position of every

Center sytem inside the box and fold it by the lattice_vectors.

Get the Coulombic cut off.

Building the neighbor list.

Get Huckel hamiltonian. Computes the Extended Huckel Hamiltonian from the

LATTE Hamiltonian parameter

Get the reciprocal vectors

Bond integrals parameters for LATTE Hamiltonian.

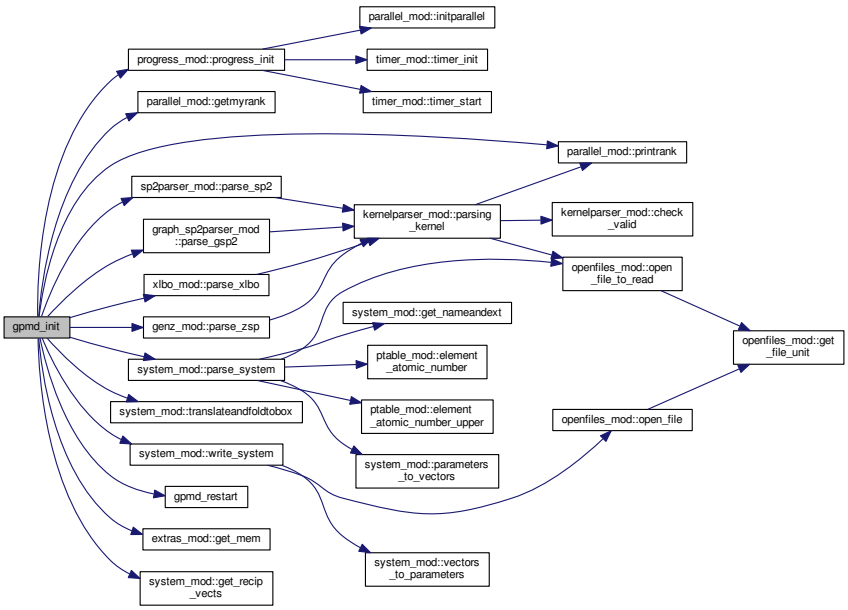
Load Pair potentials for LATTE TB.

Allocate bounds vector.

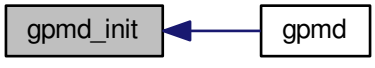
mdstep needs to be initialized.

Definition at line 180 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.9 subroutine gpmd::gpmd_initparts ( )

Initialize the partition.

Get the mapping of the Hamiltonian index with the atom index

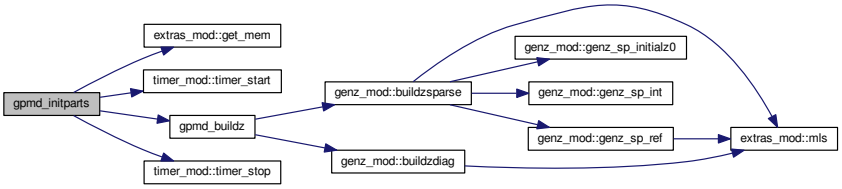
Get occupation based on last shell population.

Initialize the density matrix (rho_bml) and inverse overlap factor (zmat_bml).

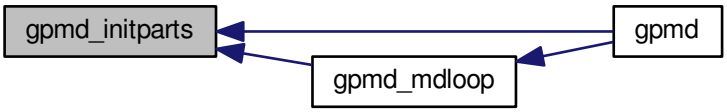
Get the Inverse square root overlap matrix.

Definition at line 428 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.10 subroutine gpmd::gpmd_mdloop ( )

Main MD loop This routine performs the MD loops up to "ls%mdsteps".

Get Kinetic energy

First 1/2 of Leapfrog step

Update positions

Update neighbor list (Actialized every nlisteach times steps)

Repartition.

Reinitialize parts.

Use SCF the first M_init MD steps

SCF loop

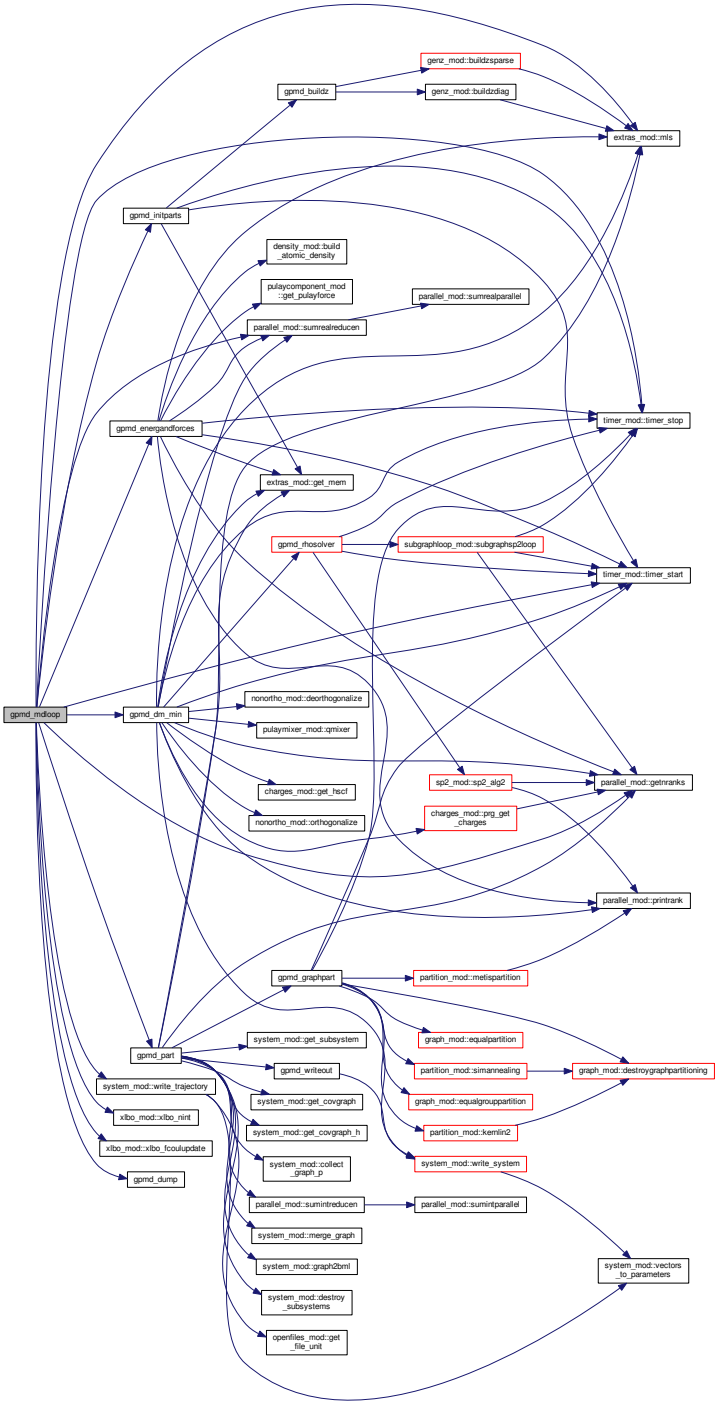
Adjust forces for the linearized XLBOMD functional

Total XLBOMD force

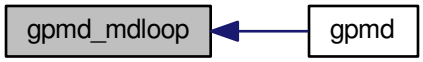
Integrate second 1/2 of leapfrog step

Definition at line 1042 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



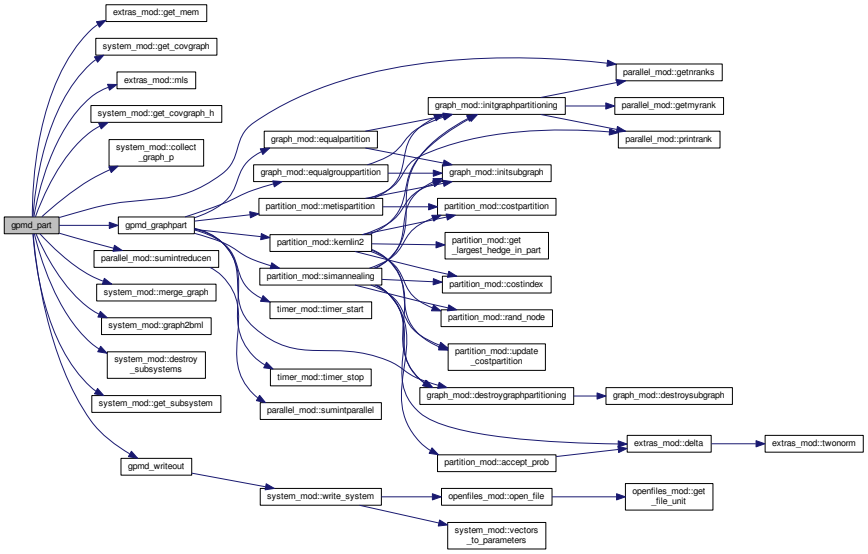
12.3.1.11 subroutine gpmd::gpmd_part ( )

Partition by systems.

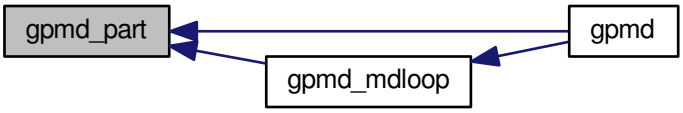
For every partition get the partial CH systems.

Definition at line 292 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.3.1.12 subroutine gpmd::gpmd_preparemd ( )

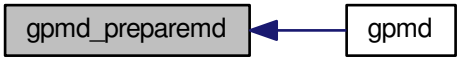
Preparing for MD.

Initialize velocities

Kinetic energy in eV (MVV2KE: unit conversion)

Definition at line 1023 of file gpmdcov.F90.

Here is the caller graph for this function:

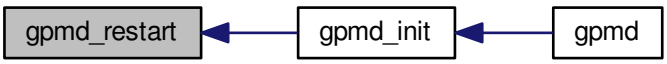


12.3.1.13 subroutine gpmd::gpmd_restart ( )

Dump GPMD.

Definition at line 1508 of file gpmdcov.F90.

Here is the caller graph for this function:

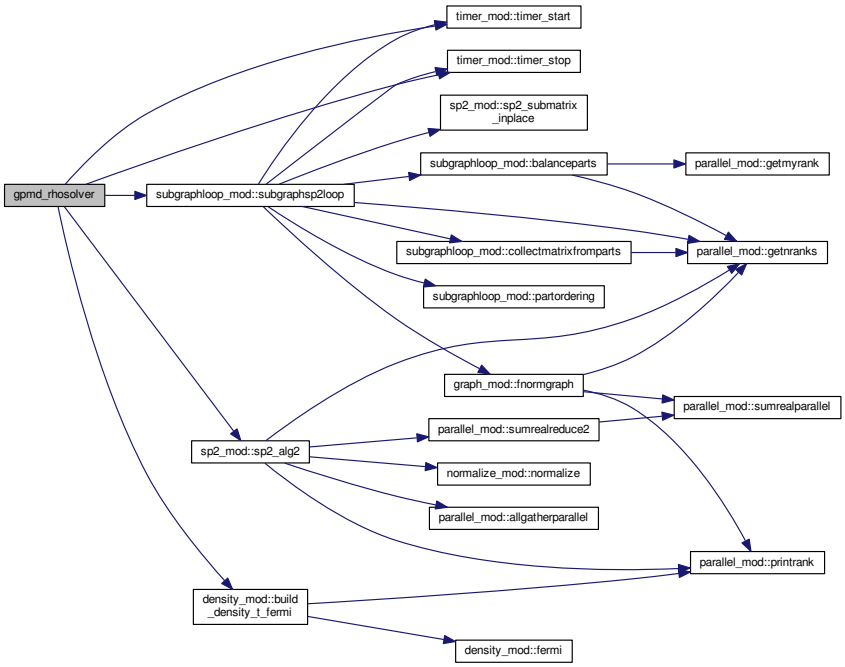


12.3.1.14 subroutine gpmd::gpmd_rhosolver (type(bml_matrix_t), intent(in) *orthoh_bml,* type(bml_matrix_t), intent(inout) *orthop_bml*)

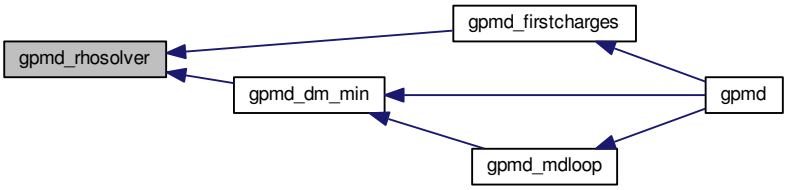
Solver for computing the density matrix.

Definition at line 1196 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



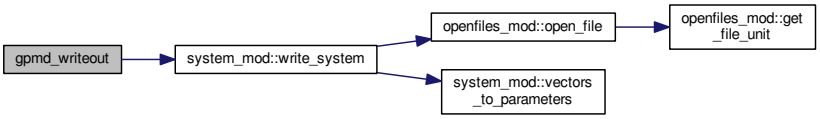
12.3.1.15 subroutine gpmd::gpmd_writeout ( )

To write output file or perform some analysis.

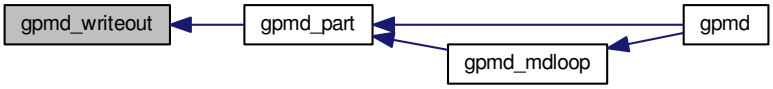
Writting the extension of the graph as a resindex

Definition at line 1396 of file gpmdcov.F90.

Here is the call graph for this function:



Here is the caller graph for this function:



12.4 /home/christian/qmd-progress/examplePrograms/gpmdcov/test-energy.py File Reference

Namespaces

- test-energy

Functions

- def test-energy.compare_MD
- def test-energy.main

12.5 /home/christian/qmd-progress/README.md File Reference

12.6 /home/christian/qmd-progress/tests/README.md File Reference

12.7 /home/christian/qmd-progress/src/charges_mod.F90 File Reference

Data Types

- module charges_mod

  A module to handle the charges of the system.

12.8 /home/christian/qmd-progress/src/densitymatrix_mod.F90 File Reference

Data Types

- module density_mod

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

12.9 /home/christian/qmd-progress/src/dos_mod.F90 File Reference

Data Types

- module dos_mod

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

12.10 /home/christian/qmd-progress/src/doxy_mod.F90 File Reference

Data Types

- module variable_names

  Glossary for the code.

12.11 /home/christian/qmd-progress/src/extras_mod.F90 File Reference

Data Types

- module extras_mod

  Extra routines:

12.12 /home/christian/qmd-progress/src/genz_mod.F90 File Reference

Data Types

- module genz_mod

  To produce a matrix Z which is needed to orthogonalize H.
  $H_{orth} = Z^{\dagger} H Z$ Please see Negre 2016 **?**.
- type genz_mod::genzspinp

  Input for the genz driver.
- type genz_mod::genzspdata

  contains the data for the genZ driver.

12.13 /home/christian/qmd-progress/src/glossary.md File Reference

12.14 /home/christian/qmd-progress/src/graph_mod.F90 File Reference

Data Types

- module graph_mod

  The graph module.
- type graph_mod::subgraph_t

Subgraph type.

- type graph_mod::graph_partitioning_t

  Trace per iteration.

12.15 /home/christian/qmd-progress/src/graph_sp2parser_mod.F90 File Reference

Data Types

- module graph_sp2parser_mod

  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:

- type graph_sp2parser_mod::gsp2data_type

  General SP2 solver type.

12.16 /home/christian/qmd-progress/src/homolumo_mod.F90 File Reference

Data Types

- module homolumo_mod

  The homolumo module.

12.17 /home/christian/qmd-progress/src/initmatrices_mod.F90 File Reference

Data Types

- module initmatrices_mod

  Initialization module.

12.18 /home/christian/qmd-progress/src/kernelparser_mod.F90 File Reference

Data Types

- module kernelparser_mod

  Some general parsing functions.

12.19 /home/christian/qmd-progress/src/nonortho_mod.F90 File Reference

Data Types

- module nonortho_mod

  Module to orthogonalize and deorthogonalize any operator.
  Typically the Hamiltonin needs to be orthogonalized: $H_{\mathrm{ortho}} = Z^{\dagger} H Z$.

12.20 /home/christian/qmd-progress/src/normalize_mod.F90 File Reference

Data Types

- module normalize_mod

  The normalize module.

12.21 /home/christian/qmd-progress/src/openfiles_mod.F90 File Reference

Data Types

- module openfiles_mod

  Module to handle input output files for the PROGRESS lib.

12.22 /home/christian/qmd-progress/src/parallel_mod.F90 File Reference

Data Types

- module parallel_mod

  The parallel module.
- type parallel_mod::rankreducedata_t

  Data structure for rection over MPI ranks.

12.23 /home/christian/qmd-progress/src/partition_mod.F90 File Reference

Data Types

- module partition_mod

  The partition module.

12.24 /home/christian/qmd-progress/src/progress_mod.F90 File Reference

Data Types

- module progress_mod

  The progress module.

12.25 /home/christian/qmd-progress/src/ptable_mod.F90 File Reference

Data Types

- module ptable_mod

  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.

12.26 /home/christian/qmd-progress/src/pulaycomponent_mod.F90 File Reference

Data Types

- module pulaycomponent_mod

  Produces a matrix to get the Pulay Component of the forces.
  Please see Niklasson 2008 **?**.

12.27 /home/christian/qmd-progress/src/pulaymixer_mod.F90 File Reference

Data Types

- module pulaymixer_mod

  Pulay mixer mode.
  Gets the best coefficient for mixing the charges during scf.

12.28 /home/christian/qmd-progress/src/response_mod.F90 File Reference

Data Types

- module response_mod

  Module to compute the response and related quantities.
- type response_mod::respdata_type

12.29 /home/christian/qmd-progress/src/sp2_mod.F90 File Reference

Data Types

- module sp2_mod

  The SP2 module.

12.30 /home/christian/qmd-progress/src/sp2parser_mod.F90 File Reference

Data Types

- module sp2parser_mod

  SP2 parser.
  This module is used to parse all the neccesary input variables for and SP2 electronic structure solver. Adding a new input keyword to the parser:
- type sp2parser_mod::sp2data_type

  General SP2 solver type.

12.31 /home/christian/qmd-progress/src/subgraphloop_mod.F90 File Reference

Data Types

- module subgraphloop_mod

  The subgraphloop module.

12.32 /home/christian/qmd-progress/src/system_mod.F90 File Reference

Data Types

- module system_mod

  A module to read and handle chemical systems.
- type system_mod::estruct_type

  Electronic structure type.
- type system_mod::system_type

  System type.

12.33 /home/christian/qmd-progress/src/timer_mod.F90 File Reference

Data Types

- module timer_mod

  The timer module.
  Example use of dynamic timing:
- type timer_mod::timer_status_t

  Timer status type.

12.34 /home/christian/qmd-progress/src/xlbo_mod.F90 File Reference

Data Types

- module xlbo_mod

  A module to perform XLBO integration.
- type xlbo_mod::xlbo_type

  General xlbo solver type.

12.35 /home/christian/qmd-progress/tests/src/accuracy.F90 File Reference

Data Types

- module accuracy_mod

12.36 /home/christian/qmd-progress/tests/src/hamiltonian.F90 File Reference

Data Types

- module hamiltonian_mod

12.37 /home/christian/qmd-progress/tests/src/main.F90 File Reference

Functions/Subroutines

- program main

  Applies a series of tests. The name of the test is pased by an argument. To use this program run: ./main test_name.

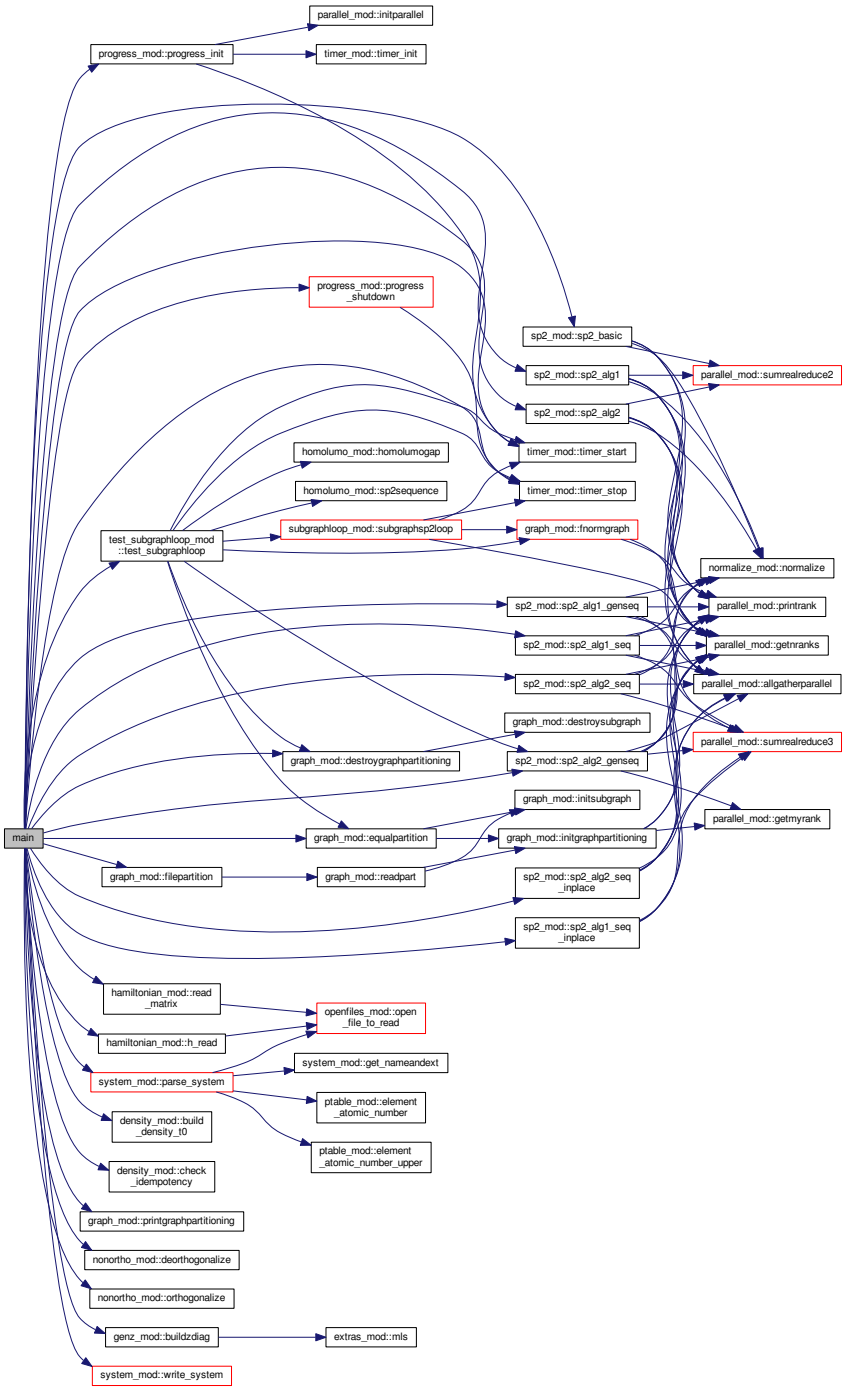
12.37.1 Function/Subroutine Documentation

12.37.1.1 program main ( )

Applies a series of tests. The name of the test is pased by an argument. To use this program run: ./main test_name.

Definition at line 6 of file main.F90.

Here is the call graph for this function:



12.38 /home/christian/qmd-progress/tests/src/test_subgraphloop.F90 File Reference

Data Types

- module test_subgraphloop_mod

  The test_subgraphloop module.