APBS

SVN version 1562

Generated by Doxygen 1.6.3

Sun Mar 7 10:01:25 2010

Contents

Chapter 1

APBS Programmers Guide

APBS was written by Nathan A. Baker.

Additional contributing authors listed in the code documentation.

1.1 Table of Contents

1.2 License

Primary author: `Nathan A. Baker` (`baker@biochem.wustl.edu`)

Department of Biochemistry and Molecular Biophysics

Center for Computational Biology

Washington University in St. Louis

Additional contributing authors are listed in the code documentation.

This documentation provides information about the programming interface provided by the APBS software and a general guide to linking to the APBS libraries. Information about installation, configuration, and general usage can be found in the `User's Guide`.

1.3 Programming Style

APBS was developed following the `Clean OO C` style of Mike Holst. In short, Clean OO C code is written in a object-oriented, ISO C-compliant fashion, and can be compiled with either a C or C++ compiler.

Following this formalism, all public data is enclosed in structures which resemble C++ classes. These structures and member functions are then declared in a public header file which provides a concise description of the interface for the class. Private functions and data are included in private header files (or simply the source code files themselves) which are not distributed. When using the library, the end-user only sees the public header file and the compiled library and is therefore (hopefully) oblivious to the private members and functions. Each class is also equipped with a constructor and destructor function which is responsible for allocating and freeing any memory required by the instatiated objects.

As mentioned above, public data members are enclosed in C structures which are visible to the end-user. Public member functions are generated by mangling the class and function names *and* passing a pointer to the object on which the member function is supposed to act. For example, a public member function with the C++ declaration

```
public double Foo::bar(int i, double d)
```

would be declared as

```
VEXTERNC double Foo_bar(Foo *thee, int i, double d)
```

where `VEXTERNC` is a compiler-dependent macro, the underscore _ replaces the C++ double-colon `::`, and `thee` replaces the `this` variable implicit in all C++ classes. Since they do not appear in public header files, private functions could be declared in any format pleasing to the user, however, the above declaration convention should generally be used for both public and private functions. Within the source code, the public and private function declarations/definitions are prefaced by the macros `VPUBLIC` and `VPRIVATE`, respectively. These are macros which reduce global name pollution, similar to encapsulating private data withing C++ classes.

The only C++ functions not explicitly covered by the above declaration scheme are the constructors (used to allocate and initialize class data members) and destructors (used to free allocated memory). These are declared in the following fashion: a constructor with the C++ declaration

```
public void Foo::Foo(int i, double d)
```

would be declared as

```
VEXTERNC Foo* Foo_ctor(int i, double d)
```

which returns a pointer to the newly constructed `Foo` object. Likewise, a destructor declared as

```
public void Foo::~Foo()
```

in C++ would be

```
VEXTERNC void Foo_dtor(Foo **thee)
```

in Clean OO C.

Finally, inline functions in C++ are simply treated as macros in Clean OO C and declared/defined using `define` statements in the public header file.

See any of the APBS header files for more information on Clean OO C programming styles.

1.4 Application programming interface documentation

The API documentation for this code was generated by `doxygen`. You can either view the API documentation by using the links at the top of this page, or the slight re-worded/re-interpreted list below:

- `Class overview`
- `Class declarations`
- `Class members`
- `Class methods`

Chapter 2

Todo List

Global Vfetk_PDE_initElement(PDE ∗thee, int elementType, int chart, double tvx[][VAPBS_DIM], void ∗d

Jump term is not implemented

Chapter 3

Deprecated List

Global nlev Just ignored now

Chapter 4

Bug List

Global Bmat_printHB(Bmat ∗thee, char ∗fname) Hardwired to only handle the single block symmetric case.

Class sVpmgp Value ipcon does not currently allow for preconditioning in PMG

Global Vacc_fastMolAcc(Vacc ∗thee, double center[VAPBS_DIM], double radius)
This routine has a slight bug which can generate very small internal regions of high dielectric (thanks to John Mongan and Jess Swanson for finding this)

Global Vacc_molAcc(Vacc ∗thee, double center[VAPBS_DIM], double radius)
This routine has a slight bug which can generate very small internal regions of high dielectric (thanks to John Mongan and Jess Swanson for finding this)

Global Vfetk_dumpLocalVar() This function is not thread-safe

Global Vfetk_externalUpdateFunction(SS ∗∗simps, int num) This function is not thread-safe.

Global Vfetk_fillArray(Vfetk ∗thee, Bvec ∗vec, Vdata_Type type) Several values of type are not implemented

Global Vfetk_PDE_ctor(Vfetk ∗fetk) Not thread-safe

Global Vfetk_PDE_ctor2(PDE ∗thee, Vfetk ∗fetk) Not thread-safe

Global Vfetk_PDE_delta(PDE ∗thee, int type, int chart, double txq[], void ∗user, double F[])
This function is not thread-safe

Global Vfetk_PDE_DFu_wv(PDE ∗thee, int key, double W[], double dW[][VAPBS_DIM], double V[], doub
This function is not thread-safe

Global Vfetk_PDE_Fu(PDE ∗thee, int key, double F[]) This function is not thread-safe

This function is not implemented (sets error to zero)

Global Vfetk_PDE_Fu_v(PDE ∗thee, int key, double V[], double dV[][VAPBS_DIM])
This function is not thread-safe

Global Vfetk_PDE_initElement(PDE ∗thee, int elementType, int chart, double tvx[][VAPBS_DIM], void ∗data)
This function is not thread-safe

Global Vfetk_PDE_initFace(PDE ∗thee, int faceType, int chart, double tnvec[])
This function is not thread-safe

Global Vfetk_PDE_initPoint(PDE ∗thee, int pointType, int chart, double txq[], double tU[], double tdU[][VAPBS_DIM
This function is not thread-safe

This function uses pre-defined boudnary definitions for the molecular surface.

Global Vfetk_PDE_Ju(PDE ∗thee, int key) This function is not thread-safe.

Global Vfetk_PDE_markSimplex(int dim, int dimII, int simplexType, int faceType[VAPBS_NVS], int vertexType[VAPI
This function is not thread-safe

Global Vfetk_PDE_u_D(PDE ∗thee, int type, int chart, double txq[], double F[])
This function is hard-coded to call only multiple-sphere Debye-Hü functions.

This function is not thread-safe.

Global Vfetk_PDE_u_T(PDE ∗thee, int type, int chart, double txq[], double F[])
This function is not thread-safe.

Global Vfetk_write(Vfetk ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, Bvec ∗vec,
Some values of format are not implemented

Global Vgreen_helmholtz(Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val, double kappa)
Not implemented yet

Global Vgreen_helmholtzD(Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗gradx, double ∗grady, dou
Not implemented yet

Global Vgrid_writeUHBD(Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char
This routine does not respect partition information

Global Vpbe_ctor2(Vpbe ∗thee, Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ion
The focusing flag is currently not used!!

Global Vpee_markRefine(Vpee ∗thee, AM ∗am, int level, int akey, int rcol, double etol, int bkey)
This function is no longer up-to-date with FEtk and may not function properly

Global Vpmg_printColComp(Vpmg ∗thee, char path[72], char title[72], char mxtype[3], int flag)
Can this path variable be replaced with a Vio socket?

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 8

Module Documentation

8.1 Vcsm class

A charge-simplex map for evaluating integrals of delta functions in a finite element setting.

Data Structures

- struct sVcsm

  Charge-simplex map class.

Files

- file vcsm.h

  Contains declarations for the Vcsm class.

- file vcsm.c

  Class Vcsm methods.

Typedefs

- typedef struct sVcsm Vcsm

  Declaration of the Vcsm class as the Vcsm structure.

Functions

- VEXTERNC void Gem_setExternalUpdateFunction (Gem ∗thee, void(∗externalUpdate)(SS ∗∗simps, int num))

  External function for FEtk Gem class to use during mesh refinement.

- VEXTERNC Valist ∗ Vcsm_getValist (Vcsm ∗thee)

  Get atom list.

- VEXTERNC int Vcsm_getNumberAtoms (Vcsm ∗thee, int isimp)

  Get number of atoms associated with a simplex.

- VEXTERNC Vatom ∗ Vcsm_getAtom (Vcsm ∗thee, int iatom, int isimp)

  Get particular atom associated with a simplex.

- VEXTERNC int Vcsm_getAtomIndex (Vcsm ∗thee, int iatom, int isimp)

  Get ID of particular atom in a simplex.

- VEXTERNC int Vcsm_getNumberSimplices (Vcsm ∗thee, int iatom)

  Get number of simplices associated with an atom.

- VEXTERNC SS ∗ Vcsm_getSimplex (Vcsm ∗thee, int isimp, int iatom)

  Get particular simplex associated with an atom.

- VEXTERNC int Vcsm_getSimplexIndex (Vcsm ∗thee, int isimp, int iatom)

  Get index particular simplex associated with an atom.

- VEXTERNC unsigned long int Vcsm_memChk (Vcsm ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vcsm ∗ Vcsm_ctor (Valist ∗alist, Gem ∗gm)

  Construct Vcsm object.

- VEXTERNC int Vcsm_ctor2 (Vcsm ∗thee, Valist ∗alist, Gem ∗gm)

  FORTRAN stub to construct Vcsm object.

- VEXTERNC void Vcsm_dtor (Vcsm ∗∗thee)

  Destroy Vcsm object.

- VEXTERNC void Vcsm_dtor2 (Vcsm ∗thee)

  FORTRAN stub to destroy Vcsm object.

- VEXTERNC void Vcsm_init (Vcsm ∗thee)

Initialize charge-simplex map with mesh and atom data.

- VEXTERNC int Vcsm_update (Vcsm ∗thee, SS ∗∗simps, int num)

  Update the charge-simplex and simplex-charge maps after refinement.

8.1.1 Detailed Description

A charge-simplex map for evaluating integrals of delta functions in a finite element setting.

8.1.2 Function Documentation

8.1.2.1 VEXTERNC void Gem_setExternalUpdateFunction (Gem ∗ *thee*, void(∗)(SS ∗∗simps, int num) *externalUpdate*)

External function for FEtk Gem class to use during mesh refinement.

Author

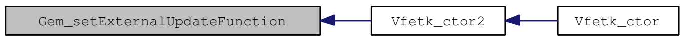
Nathan Baker

Parameters

thee The FEtk geometry manager

externalUpdate Function pointer for call during mesh refinement

Here is the caller graph for this function:



8.1.2.2 VEXTERNC Vcsm∗ Vcsm_ctor (Valist ∗ *alist*, Gem ∗ *gm*)

Construct Vcsm object.

Author

Nathan Baker

Note

- The initial mesh must be sufficiently coarse for the assignment procedures to be efficient

- The map is not built until Vcsm_init is called

Returns

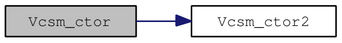
Pointer to newly allocated Vcsm object

Parameters

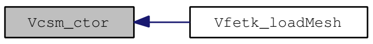
alist List of atoms

gm FEtk geometry manager defining the mesh

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.2.3 VEXTERNC int Vcsm_ctor2 (Vcsm ∗ *thee*, Valist ∗ *alist*, Gem ∗ *gm*)

FORTRAN stub to construct Vcsm object.

Author

Nathan Baker

Note

- The initial mesh must be sufficiently coarse for the assignment procedures to be efficient
- The map is not built until Vcsm_init is called

Returns

1 if successful, 0 otherwise

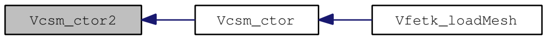
Parameters

thee The Vcsm object

alist The list of atoms

gm The FEtk geometry manager defining the mesh

Here is the caller graph for this function:



8.1.2.4 VEXTERNC void Vcsm_dtor (Vcsm ∗∗ *thee*)

Destroy Vcsm object.

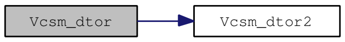
Author

Nathan Baker

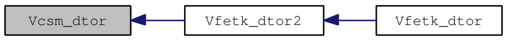
Parameters

thee Pointer to memory location for Vcsm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.2.5 VEXTERNC void Vcsm_dtor2 (Vcsm ∗ *thee*)

FORTRAN stub to destroy Vcsm object.

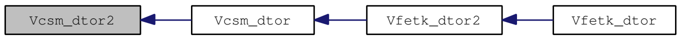
Author

Nathan Baker

Parameters

thee Pointer to Vcsm object

Here is the caller graph for this function:



8.1.2.6 VEXTERNC Vatom∗ Vcsm_getAtom (Vcsm ∗ *thee*, int *iatom*, int *isimp*)

Get particular atom associated with a simplex.

Author

Nathan Baker

Returns

Array of atoms associated with a simplex

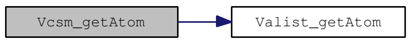
Parameters

thee The Vcsm object

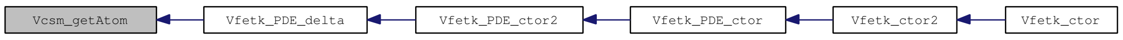
iatom Index of atom in Vcsm list ofr this simplex

isimp Simplex ID

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.2.7 VEXTERNC int Vcsm_getAtomIndex (Vcsm ∗ *thee*, int *iatom*, int *isimp*)

Get ID of particular atom in a simplex.

Author

Nathan Baker

Returns

Index of atom in Valist object

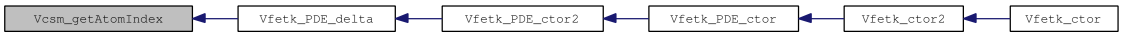
Parameters

thee The Vcsm object

iatom Index of atom in Vcsm list for this simplex

isimp Simplex ID

Here is the caller graph for this function:



8.1.2.8 VEXTERNC int Vcsm_getNumberAtoms (Vcsm ∗ *thee*, int *isimp*)

Get number of atoms associated with a simplex.

Author

Nathan Baker

Returns

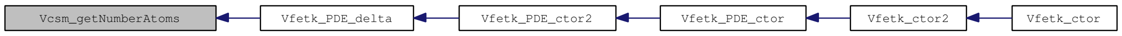
Number of atoms associated with a simplex

Parameters

thee The Vcsm object

isimp Simplex ID

Here is the caller graph for this function:



8.1.2.9 VEXTERNC int Vcsm_getNumberSimplices (Vcsm ∗ *thee*, int *iatom*)

Get number of simplices associated with an atom.

Author

Nathan Baker

Returns

Number of simplices associated with an atom

Parameters

thee The Vcsm object

iatom The Valist atom index

8.1.2.10 VEXTERNC SS∗ Vcsm_getSimplex (Vcsm ∗ *thee*, int *isimp*, int *iatom*)

Get particular simplex associated with an atom.

Author

Nathan Baker

Returns

Pointer to simplex object

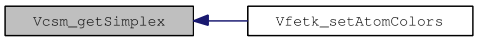
Parameters

thee The Vcsm object

isimp Index of simplex in Vcsm list

iatom Valist atom index

Here is the caller graph for this function:



8.1.2.11 VEXTERNC int Vcsm_getSimplexIndex (Vcsm ∗ *thee*, int *isimp*, int *iatom*)

Get index particular simplex associated with an atom.

Author

Nathan Baker

Returns

Gem index of specified simplex

Parameters

thee The Vcsm object

isimp Index of simplex in Vcsm list

iatom Index of atom in Valist

8.1.2.12 VEXTERNC Valist∗ Vcsm_getValist (Vcsm ∗ *thee*)

Get atom list.

Author

Nathan Baker

Returns

Pointer to Valist atom list

Parameters

thee The Vcsm object

8.1.2.13 VEXTERNC void Vcsm_init (Vcsm ∗ *thee*)

Initialize charge-simplex map with mesh and atom data.

Author

Nathan Baker

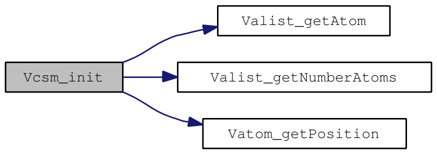
Note

The initial mesh must be sufficiently coarse for the assignment procedures to be efficient

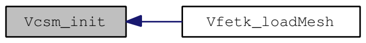
Parameters

thee The Vcsm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.2.14 VEXTERNC unsigned long int Vcsm_memChk (Vcsm ∗ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

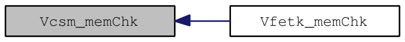
Returns

The memory used by this structure and its contents in bytes

Parameters

thee The Vcsm object

Here is the caller graph for this function:



8.1.2.15 VEXTERNC int Vcsm_update (Vcsm ∗ *thee*, SS ∗∗ *simps*, int *num*)

Update the charge-simplex and simplex-charge maps after refinement.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

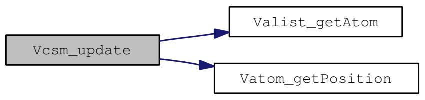
Parameters

thee The Vcsm object

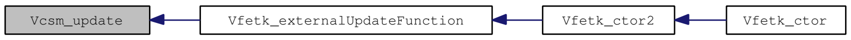
simps List of pointer to newly created (by refinement) simplex objects. The first simplex is expected to be derived from the parent simplex and therefore have the same ID. The remaining simplices are the children and should represent new entries in the charge-simplex map.

num Number of simplices in simps list

Here is the call graph for this function:



Here is the caller graph for this function:



8.2 Vfetk class

FEtk master class (interface between FEtk and APBS).

Data Structures

- struct sVfetk

  Contains public data members for Vfetk class/module.

- struct sVfetk_LocalVar

  Vfetk LocalVar subclass.

Files

- file vfetk.h

  Contains declarations for class Vfetk.

- file vfetk.c

  Class Vfetk methods.

Defines

- #define VRINGMAX 1000

  Maximum number of simplices in a simplex ring.

- #define VATOMMAX 1000000

  Maximum number of atoms associated with a vertex.

Typedefs

- typedef enum eVfetk_LsolvType Vfetk_LsolvType

  Declare FEMparm_LsolvType type.

- typedef enum eVfetk_MeshLoad Vfetk_MeshLoad

  Declare FEMparm_GuessType type.

- typedef enum eVfetk_NsolvType Vfetk_NsolvType

  Declare FEMparm_NsolvType type.

- typedef enum eVfetk_GuessType Vfetk_GuessType

  Declare FEMparm_GuessType type.

- typedef enum eVfetk_PrecType Vfetk_PrecType

  Declare FEMparm_GuessType type.

- typedef struct sVfetk_LocalVar Vfetk_LocalVar

  Declaration of the Vfetk_LocalVar subclass as the Vfetk_LocalVar structure.

- typedef struct sVfetk Vfetk

  Declaration of the Vfetk class as the Vfetk structure.

Enumerations

- enum eVfetk_LsolvType { VLT_SLU = 0, VLT_MG = 1, VLT_CG = 2, VLT_-BCG = 3 }

  Linear solver type.

- enum eVfetk_MeshLoad { VML_DIRICUBE, VML_NEUMCUBE, VML_-EXTERNAL }

  Mesh loading operation.

- enum eVfetk_NsolvType { VNT_NEW = 0, VNT_INC = 1, VNT_ARC = 2 }

  Non-linear solver type.

- enum eVfetk_GuessType { VGT_ZERO = 0, VGT_DIRI = 1, VGT_PREV = 2 }

  Initial guess type.

- enum eVfetk_PrecType { VPT_IDEN = 0, VPT_DIAG = 1, VPT_MG = 2 }

  Preconditioner type.

Functions

- VEXTERNC Gem ∗ Vfetk_getGem (Vfetk ∗thee)

  Get a pointer to the Gem (grid manager) object.

- VEXTERNC AM ∗ Vfetk_getAM (Vfetk ∗thee)

  Get a pointer to the AM (algebra manager) object.

- VEXTERNC Vpbe ∗ Vfetk_getVpbe (Vfetk ∗thee)

  Get a pointer to the Vpbe (PBE manager) object.

- VEXTERNC Vcsm ∗ Vfetk_getVcsm (Vfetk ∗thee)

  Get a pointer to the Vcsm (charge-simplex map) object.

- VEXTERNC int Vfetk_getAtomColor (Vfetk ∗thee, int iatom)

  Get the partition information for a particular atom.

- VEXTERNC Vfetk ∗ Vfetk_ctor (Vpbe ∗pbe, Vhal_PBEType type)

  Constructor for Vfetk object.

- VEXTERNC int Vfetk_ctor2 (Vfetk ∗thee, Vpbe ∗pbe, Vhal_PBEType type)

  FORTRAN stub constructor for Vfetk object.

- VEXTERNC void Vfetk_dtor (Vfetk ∗∗thee)

  Object destructor.

- VEXTERNC void Vfetk_dtor2 (Vfetk ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC double ∗ Vfetk_getSolution (Vfetk ∗thee, int ∗length)

  Create an array containing the solution (electrostatic potential in units of $k_B T/e$) at the finest mesh level.

- VEXTERNC void Vfetk_setParameters (Vfetk ∗thee, PBEparm ∗pbeparm, FEMparm ∗feparm)

  Set the parameter objects.

- VEXTERNC double Vfetk_energy (Vfetk ∗thee, int color, int nonlin)

  Return the total electrostatic energy.

- VEXTERNC double Vfetk_dqmEnergy (Vfetk ∗thee, int color)

  Get the "mobile charge" and "polarization" contributions to the electrostatic energy.

- VEXTERNC double Vfetk_qfEnergy (Vfetk ∗thee, int color)

  Get the "fixed charge" contribution to the electrostatic energy.

- VEXTERNC unsigned long int Vfetk_memChk (Vfetk ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC void Vfetk_setAtomColors (Vfetk ∗thee)

Transfer color (partition ID) information frmo a partitioned mesh to the atoms.

- VEXTERNC void Bmat_printHB (Bmat ∗thee, char ∗fname)

  Writes a Bmat to disk in Harwell-Boeing sparse matrix format.

- VEXTERNC Vrc_Codes Vfetk_genCube (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType)

  Construct a rectangular mesh (in the current Vfetk object).

- VEXTERNC Vrc_Codes Vfetk_loadMesh (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType, Vio ∗sock)

  Loads a mesh into the Vfetk (and associated) object(s).

- VEXTERNC PDE ∗ Vfetk_PDE_ctor (Vfetk ∗fetk)

  Constructs the FEtk PDE object.

- VEXTERNC int Vfetk_PDE_ctor2 (PDE ∗thee, Vfetk ∗fetk)

  Intializes the FEtk PDE object.

- VEXTERNC void Vfetk_PDE_dtor (PDE ∗∗thee)

  Destroys FEtk PDE object.

- VEXTERNC void Vfetk_PDE_dtor2 (PDE ∗thee)

  FORTRAN stub: destroys FEtk PDE object.

- VEXTERNC void Vfetk_PDE_initAssemble (PDE ∗thee, int ip[ ], double rp[ ])

  Do once-per-assembly initialization.

- VEXTERNC void Vfetk_PDE_initElement (PDE ∗thee, int elementType, int chart, double tvx[ ][VAPBS_DIM], void ∗data)

  Do once-per-element initialization.

- VEXTERNC void Vfetk_PDE_initFace (PDE ∗thee, int faceType, int chart, double tnvec[ ])

  Do once-per-face initialization.

- VEXTERNC void Vfetk_PDE_initPoint (PDE ∗thee, int pointType, int chart, double txq[ ], double tU[ ], double tdU[ ][VAPBS_DIM])

  Do once-per-point initialization.

- VEXTERNC void Vfetk_PDE_Fu (PDE ∗thee, int key, double F[ ])

Evaluate strong form of PBE. For interior points, this is:

$$-\nabla \cdot \epsilon \nabla u + b(u) - f$$

where $b(u)$ is the (possibly nonlinear) mobile ion term and f is the source charge distribution term (for PBE) or the induced surface charge distribution (for RPBE). For an interior-boundary (simplex face) point, this is:

$$[\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^+} - [\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^-}$$

where $n(x)$ is the normal to the simplex face and the term represents the jump in dielectric displacement across the face. There is no outer-boundary contribution for this problem.

- VEXTERNC double Vfetk_PDE_Fu_v (PDE ∗thee, int key, double V[ ], double dV[ ][VAPBS_DIM])

  This is the weak form of the PBE; i.e. the strong form integrated with a test function to give:

$$\int_\Omega [\epsilon \nabla u \cdot \nabla v + b(u)v - fv]\, dx$$

  where $b(u)$ denotes the mobile ion term.

- VEXTERNC double Vfetk_PDE_DFu_wv (PDE ∗thee, int key, double W[ ], double dW[ ][VAPBS_DIM], double V[ ], double dV[ ][VAPBS_DIM])

  This is the linearization of the weak form of the PBE; e.g., for use in a Newton iteration. This is the functional linearization of the strong form integrated with a test function to give:

$$\int_\Omega \left[\epsilon \nabla w \cdot \nabla v + b'(u)wv - fv\right] dx$$

  where $b'(u)$ denotes the functional derivation of the mobile ion term.

- VEXTERNC void Vfetk_PDE_delta (PDE ∗thee, int type, int chart, double txq[ ], void ∗user, double F[ ])

  Evaluate a (discretized) delta function source term at the given point.

- VEXTERNC void Vfetk_PDE_u_D (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the Dirichlet boundary condition at the given point.

- VEXTERNC void Vfetk_PDE_u_T (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the "true solution" at the given point for comparison with the numerical solution.

- VEXTERNC void Vfetk_PDE_bisectEdge (int dim, int dimII, int edgeType, int chart[ ], double vx[ ][VAPBS_DIM])

  Define the way manifold edges are bisected.

- VEXTERNC void Vfetk_PDE_mapBoundary (int dim, int dimII, int vertexType, int chart, double vx[VAPBS_DIM])

  Map a boundary point to some pre-defined shape.

- VEXTERNC int Vfetk_PDE_markSimplex (int dim, int dimII, int simplexType, int faceType[VAPBS_NVS], int vertexType[VAPBS_NVS], int chart[ ], double vx[ ][VAPBS_DIM], void ∗simplex)

  User-defined error estimator -- in our case, a geometry-based refinement method; forcing simplex refinement at the dielectric boundary and (for non-regularized PBE) the charges.

- VEXTERNC void Vfetk_PDE_oneChart (int dim, int dimII, int objType, int chart[ ], double vx[ ][VAPBS_DIM], int dimV)

  Unify the chart for different coordinate systems -- a no-op for us.

- VEXTERNC double Vfetk_PDE_Ju (PDE ∗thee, int key)

  Energy functional. This returns the energy (less delta function terms) in the form:

$$c^{-1}/2 \int (\epsilon(\nabla u)^2 + \kappa^2(cosh u - 1))dx$$

  for a 1:1 electrolyte where c is the output from Vpbe_getZmagic.

- VEXTERNC void Vfetk_externalUpdateFunction (SS ∗∗simps, int num)

  External hook to simplex subdivision routines in Gem. Called each time a simplex is subdivided (we use it to update the charge-simplex map).

- VEXTERNC int Vfetk_PDE_simplexBasisInit (int key, int dim, int comp, int ∗ndof, int dof[ ])

  Initialize the bases for the trial or the test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VEXTERNC void Vfetk_PDE_simplexBasisForm (int key, int dim, int comp, int pdkey, double xq[ ], double basis[ ])

  Evaluate the bases for the trial or test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VEXTERNC void Vfetk_readMesh (Vfetk ∗thee, int skey, Vio ∗sock)

  Read in mesh and initialize associated internal structures.

- VEXTERNC void Vfetk_dumpLocalVar ()

  Debugging routine to print out local variables used by PDE object.

- VEXTERNC int Vfetk_fillArray (Vfetk ∗thee, Bvec ∗vec, Vdata_Type type)

Fill an array with the specified data.

- VEXTERNC int Vfetk_write (Vfetk ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, Bvec ∗vec, Vdata_Format format)

  Write out data.

- VEXTERNC Vrc_Codes Vfetk_loadGem (Vfetk ∗thee, Gem ∗gm)

  Load a Gem geometry manager object into Vfetk.

8.2.1 Detailed Description

FEtk master class (interface between FEtk and APBS).

8.2.2 Enumeration Type Documentation

8.2.2.1 enum eVfetk_GuessType

Initial guess type.

Note

Do not change these values; they correspond to settings in FEtk

Enumerator:

VGT_ZERO Zero initial guess

VGT_DIRI Dirichlet boundary condition initial guess

VGT_PREV Previous level initial guess

8.2.2.2 enum eVfetk_LsolvType

Linear solver type.

Note

Do not change these values; they correspond to settings in FEtk

Enumerator:

VLT_SLU SuperLU direct solve

VLT_MG Multigrid

VLT_CG Conjugate gradient

VLT_BCG BiCGStab

8.2.2.3 enum eVfetk_MeshLoad

Mesh loading operation.

Enumerator:

VML_DIRICUBE Dirichlet cube
VML_NEUMCUBE Neumann cube
VML_EXTERNAL External mesh (from socket)

8.2.2.4 enum eVfetk_NsolvType

Non-linear solver type.

Note

Do not change these values; they correspond to settings in FEtk

Enumerator:

VNT_NEW Newton solver
VNT_INC Incremental
VNT_ARC Psuedo-arclength

8.2.2.5 enum eVfetk_PrecType

Preconditioner type.

Note

Do not change these values; they correspond to settings in FEtk

Enumerator:

VPT_IDEN Identity matrix
VPT_DIAG Diagonal scaling
VPT_MG Multigrid

8.2.3 Function Documentation

8.2.3.1 VEXTERNC void Bmat_printHB (Bmat ∗ *thee*, char ∗ *fname*)

Writes a Bmat to disk in Harwell-Boeing sparse matrix format.

Author

Stephen Bond

Note

This is a friend function of Bmat

Bug

Hardwired to only handle the single block symmetric case.

Parameters

thee The matrix to write

fname Filename for output

8.2.3.2 VEXTERNC Vfetk∗ Vfetk_ctor (Vpbe ∗ *pbe*, Vhal_PBEType *type*)

Constructor for Vfetk object.

Author

Nathan Baker

Returns

Pointer to newly allocated Vfetk object

Note

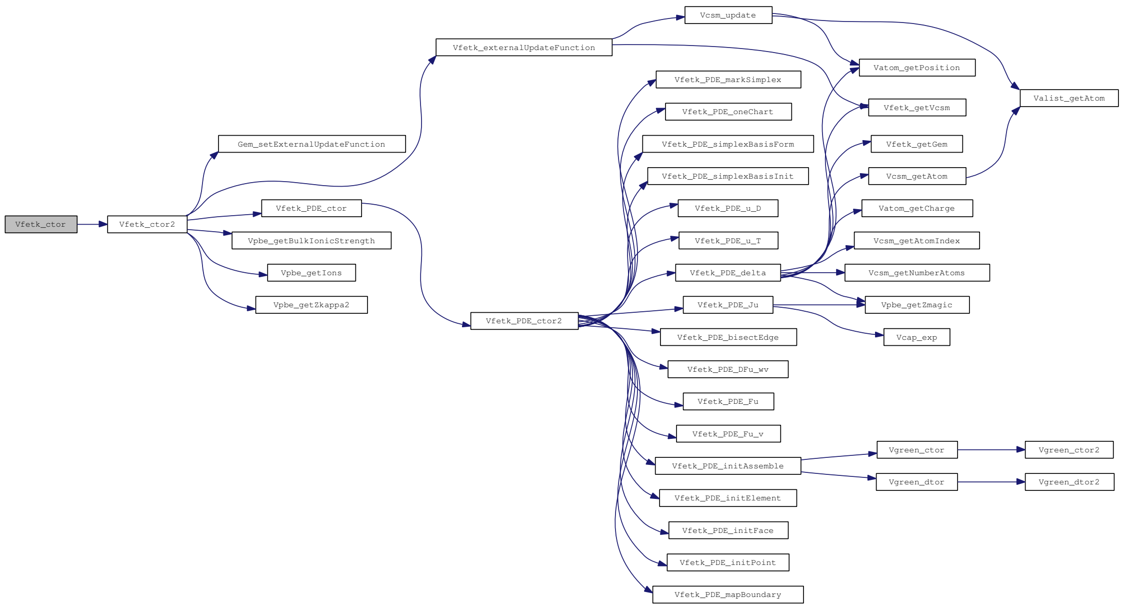
This sets up the Gem, AM, and Aprx FEtk objects but does not create a mesh. The easiest way to create a mesh is to then call Vfetk_genCube

Parameters

pbe Vpbe (PBE manager object)

type Version of PBE to solve

Here is the call graph for this function:



8.2.3.3 VEXTERNC int Vfetk_ctor2 (Vfetk ∗ *thee*, Vpbe ∗ *pbe*, Vhal_PBEType *type*)

FORTRAN stub constructor for Vfetk object.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Note

This sets up the Gem, AM, and Aprx FEtk objects but does not create a mesh. The easiest way to create a mesh is to then call Vfetk_genCube

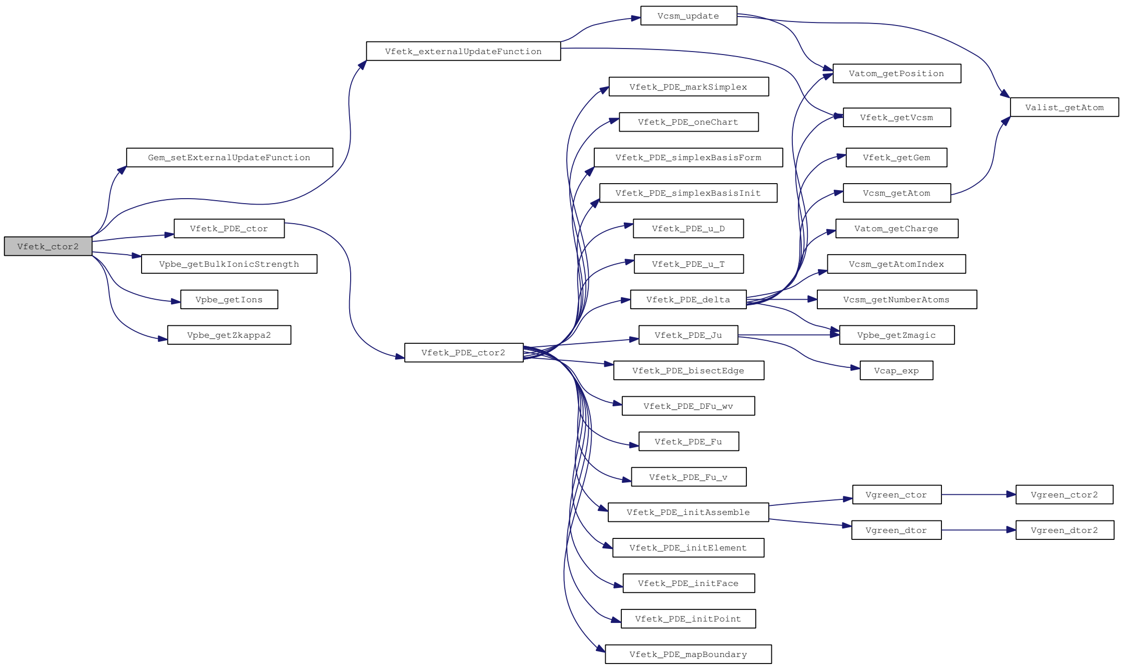
Parameters

thee Vfetk object memory

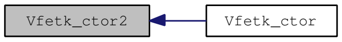
pbe PBE manager object

type Version of PBE to solve

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.4 VEXTERNC double Vfetk_dqmEnergy (Vfetk * *thee*, int *color*)

Get the "mobile charge" and "polarization" contributions to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the mobile charges with the potential and polarization of the dielectric medium:

$$G = \frac{1}{4I_s}\sum_i c_i q_i^2 \int \overline{\kappa}^2(x) e^{-q_i u(x)} dx + \frac{1}{2}\int \epsilon(\nabla u)^2 dx$$

for the NPBE and

$$G = \frac{1}{2}\int \overline{\kappa}^2(x) u^2(x) dx + \frac{1}{2}\int \epsilon(\nabla u)^2 dx$$

for the LPBE. Here i denotes the counterion species, I_s is the bulk ionic strength, $\overline{\kappa}^2(x)$ is the modified Debye-Huckel parameter, c_i is the concentration of species i, q_i

is the charge of species i, ϵ is the dielectric function, and $u(x)$ is the dimensionless electrostatic potential. The energy is scaled to units of $k_b T$.

Author

Nathan Baker

Parameters

thee Vfetk object

color Partition restriction for energy evaluation, only used if non-negative

Returns

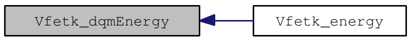
The "mobile charge" and "polarization" contributions to the electrostatic energy in units of $k_B T$.

Parameters

thee The Vfetk object

color Partition restriction for energy calculation; if non-negative, energy calculation is restricted to the specified partition (indexed by simplex and atom colors

Here is the caller graph for this function:



8.2.3.5 VEXTERNC void Vfetk_dtor (Vfetk $**$ *thee*)

Object destructor.

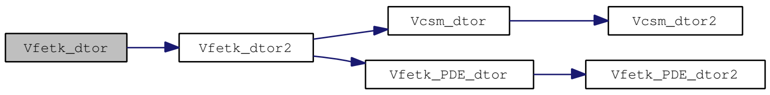
Author

Nathan Baker

Parameters

thee Pointer to memory location of Vfetk object

Here is the call graph for this function:



8.2.3.6 VEXTERNC void Vfetk_dtor2 (Vfetk ∗ *thee*)

FORTRAN stub object destructor.

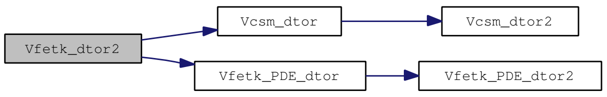
Author

Nathan Baker

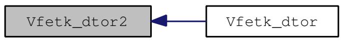
Parameters

thee Pointer to Vfetk object to be destroyed

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.7 VEXTERNC void Vfetk_dumpLocalVar ()

Debugging routine to print out local variables used by PDE object.

Author

Nathan Baker

Bug

This function is not thread-safe

8.2.3.8 VEXTERNC double Vfetk_energy (Vfetk ∗ *thee*, int *color*, int *nonlin*)

Return the total electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy using the free energy functional for the Poisson-Boltzmann equation without removing any self-interaction terms (i.e., removing the reference state of isolated charges present in an infinite dielectric continuum with the same relative permittivity as the interior of the protein) and return the result in units of $k_B T$. The argument color allows the user to control the partition on which this energy is calculated; if (color == -1) no restrictions are used. The solution is obtained from the finest level of the passed AM object, but atomic data from the Vfetk object is used to calculate the energy.

Author

Nathan Baker

Returns

Total electrostatic energy in units of $k_B T$.

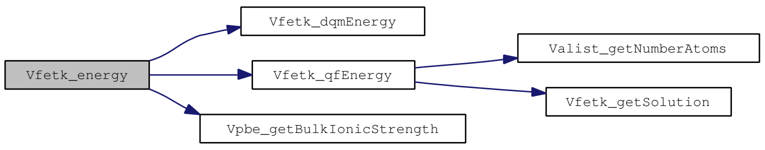
Parameters

thee THe Vfetk object

color Partition restriction for energy calculation; if non-negative, energy calculation is restricted to the specified partition (indexed by simplex and atom colors

nonlin If 1, the NPBE energy functional is used; otherwise, the LPBE energy functional is used. If -2, SMPBE is used.

Here is the call graph for this function:



8.2.3.9 VEXTERNC void Vfetk_externalUpdateFunction (SS ∗∗ *simps*, int *num*)

External hook to simplex subdivision routines in Gem. Called each time a simplex is subdivided (we use it to update the charge-simplex map).

Author

Nathan Baker

Bug

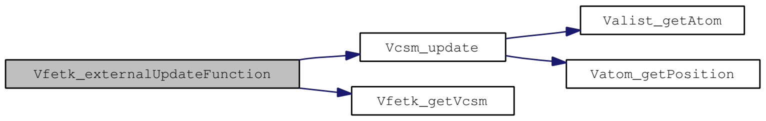
This function is not thread-safe.

Parameters

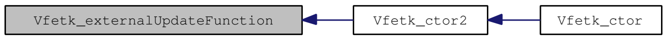
simps List of parent (simps[0]) and children (remainder) simplices

num Number of simplices in list

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.10 VEXTERNC int Vfetk_fillArray (Vfetk ∗ *thee*, Bvec ∗ *vec*, Vdata_Type *type*)

Fill an array with the specified data.

Author

Nathan Baker

Note

This function is thread-safe

Bug

Several values of type are not implemented

Returns

1 if successful, 0 otherwise

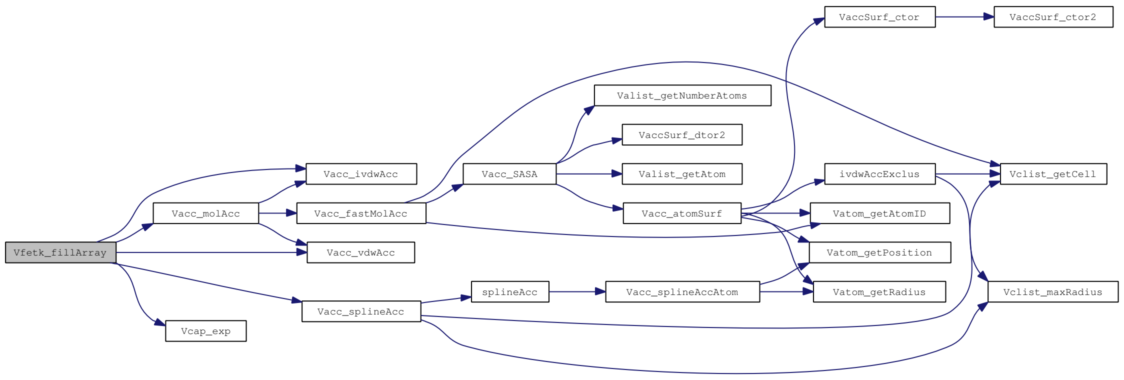
Parameters

thee The Vfetk object with the data

vec The vector to hold the data

type THe type of data to write

Here is the call graph for this function:



8.2.3.11 VEXTERNC Vrc_Codes Vfetk_genCube (Vfetk ∗ *thee*, double *center*[3], double *length*[3], Vfetk_MeshLoad *meshType*)

Construct a rectangular mesh (in the current Vfetk object).

Author

Nathan Baker

Parameters

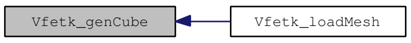
thee Vfetk object

center Center for mesh

length Mesh lengths

meshType Mesh boundary conditions

Here is the caller graph for this function:



8.2.3.12 VEXTERNC AM∗ Vfetk_getAM (Vfetk ∗ *thee*)

Get a pointer to the AM (algebra manager) object.

Author

Nathan Baker

Returns

Pointer to the AM (algebra manager) object

Parameters

thee The Vfetk object

8.2.3.13 VEXTERNC int Vfetk_getAtomColor (Vfetk ∗ *thee*, int *iatom*)

Get the partition information for a particular atom.

Author

Nathan Baker

Note

Friend function of Vatom

Returns

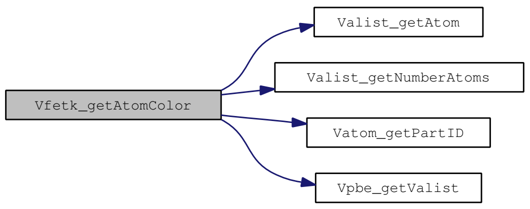
Partition ID

Parameters

thee The Vfetk object

iatom Valist atom index

Here is the call graph for this function:



8.2.3.14 VEXTERNC Gem$*$ Vfetk_getGem (Vfetk $*$ *thee*)

Get a pointer to the Gem (grid manager) object.

Author

Nathan Baker

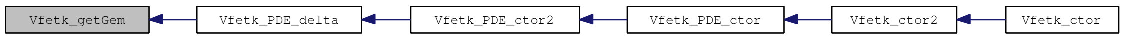
Returns

Pointer to the Gem (grid manager) object

Parameters

thee Vfetk object

Here is the caller graph for this function:



8.2.3.15 VEXTERNC double$*$ Vfetk_getSolution (Vfetk $*$ *thee*, int $*$ *length*)

Create an array containing the solution (electrostatic potential in units of $k_B T/e$) at the finest mesh level.

Author

Nathan Baker and Michael Holst

Note

The user is responsible for destroying the newly created array

Returns

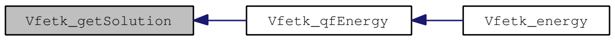
Newly created array of length "length" (see above); the user is responsible for destruction

Parameters

thee Vfetk object with solution

length Ste to length of the newly created solution array

Here is the caller graph for this function:



8.2.3.16 VEXTERNC Vcsm∗ Vfetk_getVcsm (Vfetk ∗ *thee*)

Get a pointer to the Vcsm (charge-simplex map) object.

Author

Nathan Baker

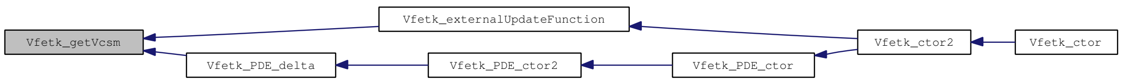
Returns

Pointer to the Vcsm (charge-simplex map) object

Parameters

thee The Vfetk object

Here is the caller graph for this function:



8.2.3.17 VEXTERNC Vpbe∗ Vfetk_getVpbe (Vfetk ∗ *thee*)

Get a pointer to the Vpbe (PBE manager) object.

Author

Nathan Baker

Returns

Pointer to the Vpbe (PBE manager) object

Parameters

thee The Vfetk object

8.2.3.18 VEXTERNC Vrc_Codes Vfetk_loadGem (Vfetk ∗ *thee*, Gem ∗ *gm*)

Load a Gem geometry manager object into Vfetk.

Author

Nathan Baker

Parameters

thee Destination

gm Geometry manager source

8.2.3.19 VEXTERNC Vrc_Codes Vfetk_loadMesh (Vfetk ∗ *thee*, double *center*[3], double *length*[3], Vfetk_MeshLoad *meshType*, Vio ∗ *sock*)

Loads a mesh into the Vfetk (and associated) object(s).

Author

Nathan Baker

Parameters

thee Vfetk object to load into

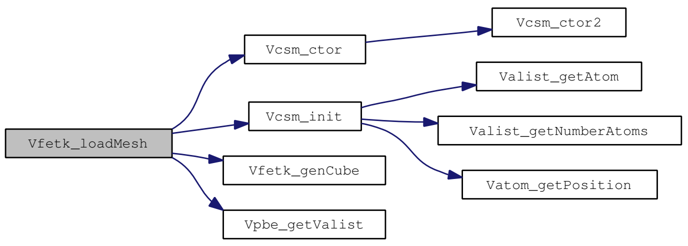
center Center for mesh (if constructed)

length Mesh lengths (if constructed)

meshType Type of mesh to load

sock Socket for external mesh data (NULL otherwise)

Here is the call graph for this function:



8.2.3.20 VEXTERNC unsigned long int Vfetk_memChk (Vfetk ∗ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

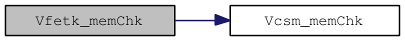
Returns

The memory used by this structure and its contents in bytes

Parameters

thee THe Vfetk object

Here is the call graph for this function:



8.2.3.21 VEXTERNC void Vfetk_PDE_bisectEdge (int *dim*, int *dimII*, int *edgeType*, int *chart*[ ], double *vx*[ ][VAPBS_DIM])

Define the way manifold edges are bisected.

Author

Nathan Baker and Mike Holst

Note

This function is thread-safe.

Parameters

dim Intrinsic dimension of manifold

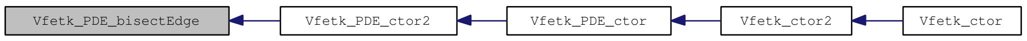
dimII Embedding dimension of manifold

edgeType Type of edge being refined

chart Chart for edge vertices, used here as accessibility bitfields

vx Edge vertex coordindates

Here is the caller graph for this function:



8.2.3.22 VEXTERNC PDE∗ Vfetk_PDE_ctor (Vfetk ∗ *fetk*)

Constructs the FEtk PDE object.

Author

Nathan Baker

Returns

Newly-allocated PDE object

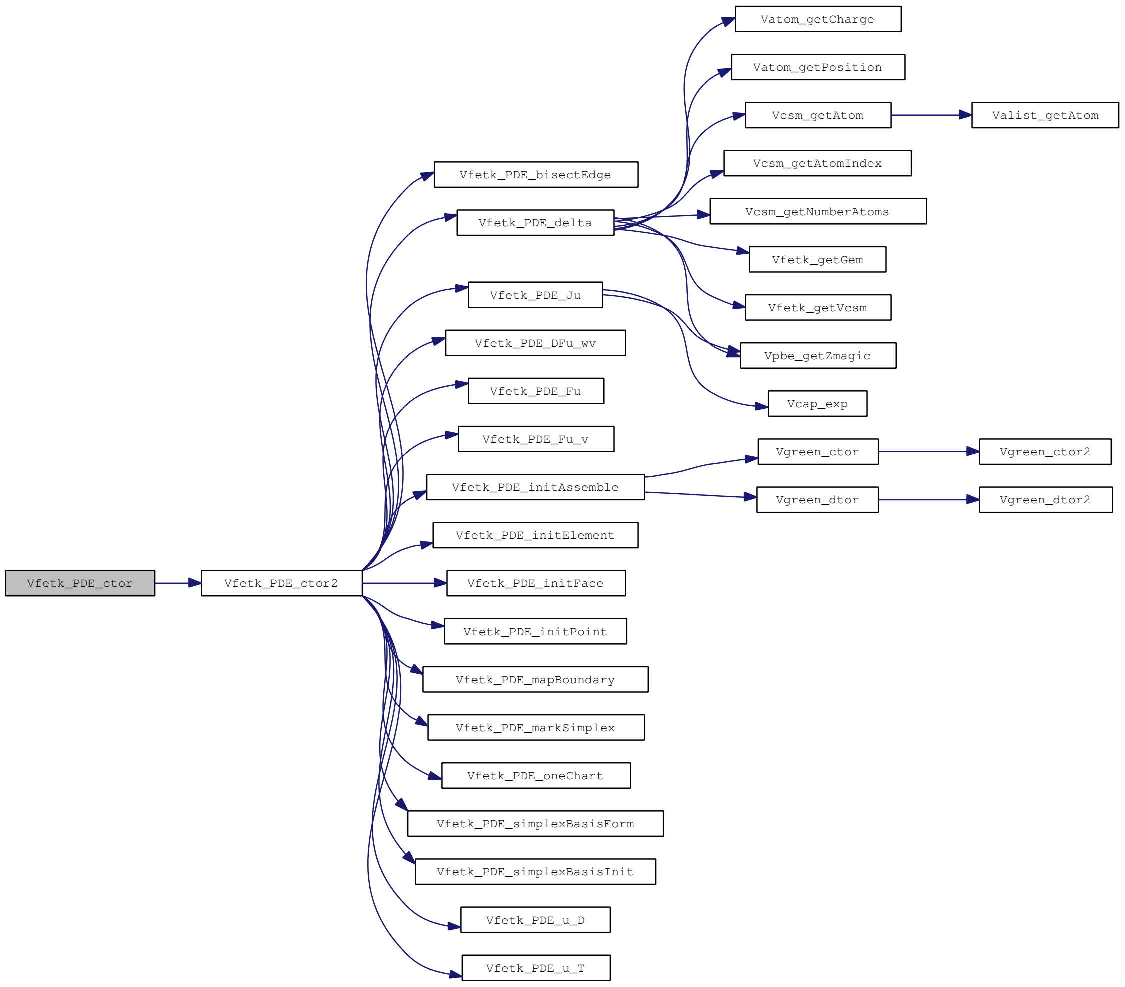
Bug

Not thread-safe

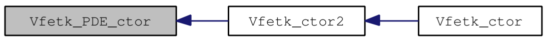
Parameters

fetk The Vfetk object

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.23 VEXTERNC int Vfetk_PDE_ctor2 (PDE ∗ *thee*, Vfetk ∗ *fetk*)

Intializes the FEtk PDE object.

Author

Nathan Baker (with code by Mike Holst)

Returns

1 if successful, 0 otherwise

Bug

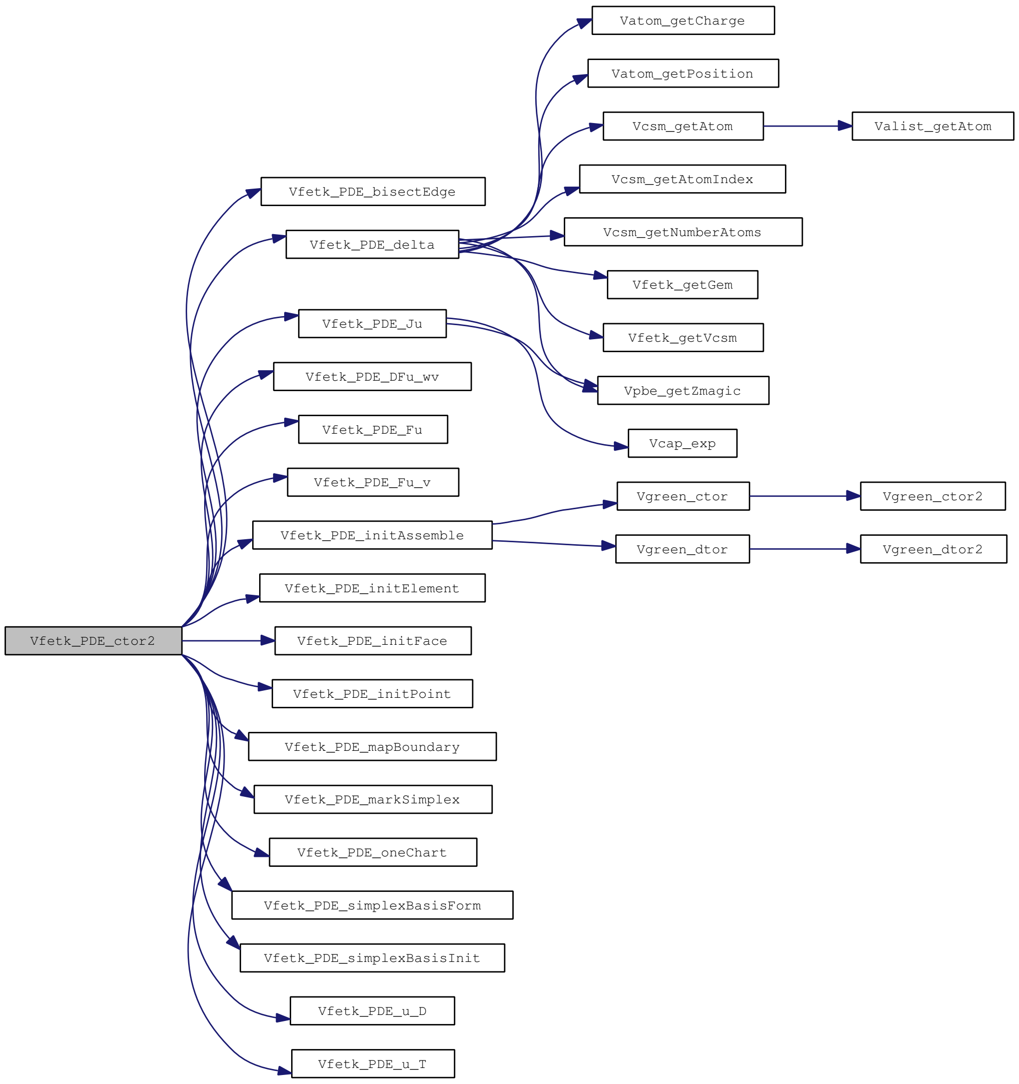
Not thread-safe

Parameters

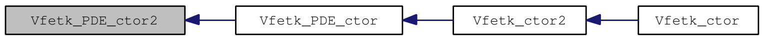
thee The newly-allocated PDE object

fetk The parent Vfetk object

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.24 VEXTERNC void Vfetk_PDE_delta (PDE * *thee*, int *type*, int *chart*, double *txq*[ ], void * *user*, double *F*[ ])

Evaluate a (discretized) delta function source term at the given point.

Author

Nathan Baker

Bug

This function is not thread-safe

Parameters

thee PDE object

type Vertex type

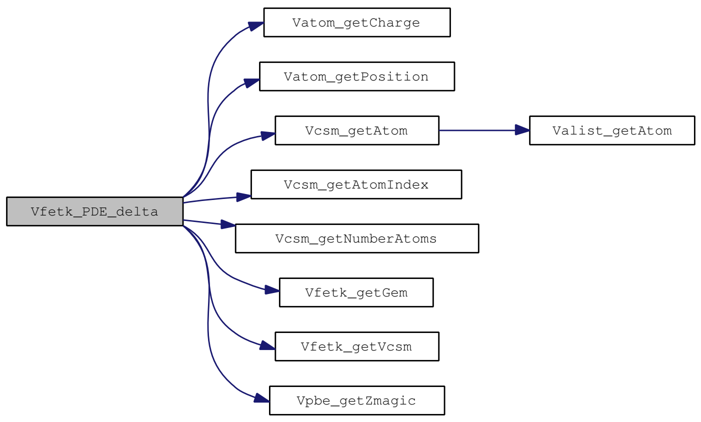
chart Chart for point coordinates

txq Point coordinates

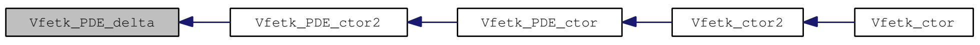
user Vertex object pointer

F Set to delta function value

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.25 VEXTERNC double Vfetk_PDE_DFu_wv (PDE $*$ *thee*, int *key*, double *W*[ ], double *dW*[ ][VAPBS_DIM], double *V*[ ], double *dV*[ ][VAPBS_DIM])

This is the linearization of the weak form of the PBE; e.g., for use in a Newton iteration. This is the functional linearization of the strong form integrated with a test function to give:

$$\int_{\Omega} \left[\epsilon \nabla w \cdot \nabla v + b'(u) w v - f v\right] dx$$

where $b'(u)$ denotes the functional derivation of the mobile ion term.

Author

Nathan Baker and Mike Holst

Returns

Integrand value

Bug

This function is not thread-safe

Parameters

thee The PDE object

key Integrand to evaluate (0 = interior weak form, 1 = boundary weak form)

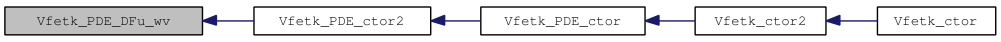
W Trial function value at current point

dW Trial function gradient at current point

V Test function value at current point

dV Test function gradient

Here is the caller graph for this function:



8.2.3.26 VEXTERNC void Vfetk_PDE_dtor (PDE $**$ *thee*)

Destroys FEtk PDE object.

Author

Nathan Baker

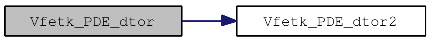
Note

Thread-safe

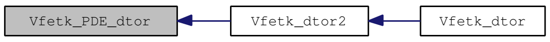
Parameters

thee Pointer to PDE object memory

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.27 VEXTERNC void Vfetk_PDE_dtor2 (PDE $*$ *thee*)

FORTRAN stub: destroys FEtk PDE object.

Author

Nathan Baker

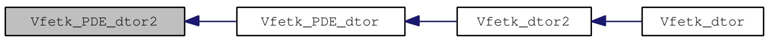
Note

Thread-safe

Parameters

thee PDE object memory

Here is the caller graph for this function:



8.2.3.28 VEXTERNC void Vfetk_PDE_Fu (PDE $*$ *thee*, int *key*, double *F*[ ])

Evaluate strong form of PBE. For interior points, this is:

$$-\nabla \cdot \epsilon \nabla u + b(u) - f$$

where $b(u)$ is the (possibly nonlinear) mobile ion term and f is the source charge distribution term (for PBE) or the induced surface charge distribution (for RPBE). For an interior-boundary (simplex face) point, this is:

$$[\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^+} - [\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^-}$$

where $n(x)$ is the normal to the simplex face and the term represents the jump in dielectric displacement across the face. There is no outer-boundary contribution for this problem.

Author

Nathan Baker

Bug

This function is not thread-safe
This function is not implemented (sets error to zero)

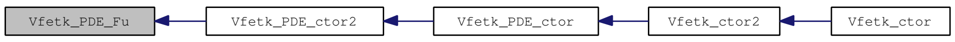
Parameters

thee The PDE object

key Type of point (0 = interior, 1 = boundary, 2 = interior boundary

F Set to value of residual

Here is the caller graph for this function:



8.2.3.29 VEXTERNC double Vfetk_PDE_Fu_v (PDE ∗ *thee*, int *key*, double *V*[ ], double *dV*[ ][VAPBS_DIM])

This is the weak form of the PBE; i.e. the strong form integrated with a test function to give:

$$\int_{\Omega} [\epsilon \nabla u \cdot \nabla v + b(u)v - fv] \, dx$$

where $b(u)$ denotes the mobile ion term.

Author

Nathan Baker and Mike Holst

Returns

Integrand value

Bug

This function is not thread-safe

Parameters

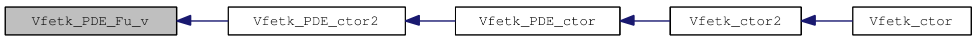
thee The PDE object

key Integrand to evaluate (0 = interior weak form, 1 = boundary weak form

V Test function at current point

dV Test function derivative at current point

Here is the caller graph for this function:



8.2.3.30 VEXTERNC void Vfetk_PDE_initAssemble (PDE ∗ *thee*, int *ip*[ ], double *rp*[ ])

Do once-per-assembly initialization.

Author

Nathan Baker and Mike Holst

Note

Thread-safe

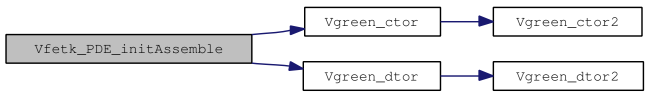
Parameters

thee PDE object

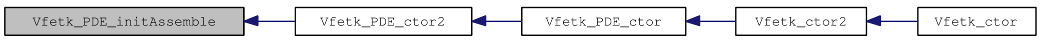
ip Integer parameter array (not used)

rp Double parameter array (not used)

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.31 VEXTERNC void Vfetk_PDE_initElement (PDE $*$ *thee*, int *elementType*, int *chart*, double *tvx*[ ][VAPBS_DIM], void $*$ *data*)

Do once-per-element initialization.

Author

Nathan Baker and Mike Holst

Todo

Jump term is not implemented

Bug

This function is not thread-safe

Parameters

thee PDE object

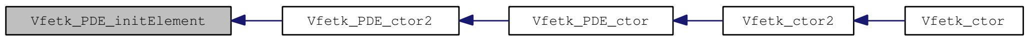
elementType Material type (not used)

chart Chart in which the vertex coordinates are provided, used here as a bitfield to store molecular accessibility

tvx Vertex coordinates

data Simplex pointer (hack)

Here is the caller graph for this function:



8.2.3.32 VEXTERNC void Vfetk_PDE_initFace (PDE $*$ *thee*, int *faceType*, int *chart*, double *tnvec*[ ])

Do once-per-face initialization.

Author

Nathan Baker and Mike Holst

Bug

This function is not thread-safe

Parameters

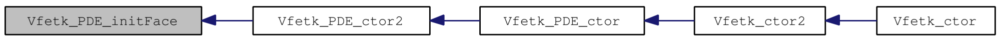
thee THe PDE object

faceType Simplex face type (interior or various boundary types)

chart Chart in which the vertex coordinates are provided, used here as a bitfield for molecular accessibility

tnvec Coordinates of outward normal vector for face

Here is the caller graph for this function:



8.2.3.33 VEXTERNC void Vfetk_PDE_initPoint (PDE ∗ *thee*, int *pointType*, int *chart*, double *txq*[ ], double *tU*[ ], double *tdU*[ ][VAPBS_DIM])

Do once-per-point initialization.

Author

Nathan Baker

Bug

This function is not thread-safe
This function uses pre-defined boudnary definitions for the molecular surface.

Parameters

thee The PDE object

pointType The type of point -- interior or various faces

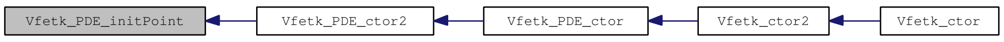
chart The chart in which the point coordinates are provided, used here as bitfield for molecular accessibility

txq Point coordinates

tU Solution value at point

tdU Solution derivative at point

Here is the caller graph for this function:



8.2.3.34 VEXTERNC double Vfetk_PDE_Ju (PDE $*$ *thee*, int *key*)

Energy functional. This returns the energy (less delta function terms) in the form:

$$c^{-1}/2 \int (\epsilon(\nabla u)^2 + \kappa^2(cosh u - 1))dx$$

for a 1:1 electrolyte where c is the output from Vpbe_getZmagic.

Author

Nathan Baker

Returns

Energy value (in kT)

Bug

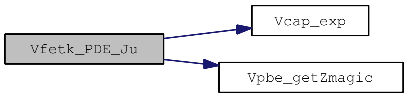
This function is not thread-safe.

Parameters

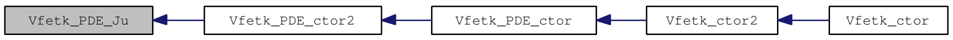
thee The PDE object

key What to evluate: interior (0) or boundary (1)?

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.35 VEXTERNC void Vfetk_PDE_mapBoundary (int *dim*, int *dimII*, int *vertexType*, int *chart*, double *vx*[VAPBS_DIM])

Map a boundary point to some pre-defined shape.

Author

Nathan Baker and Mike Holst

Note

This function is thread-safe and is a no-op

Parameters

dim Intrinsic dimension of manifold

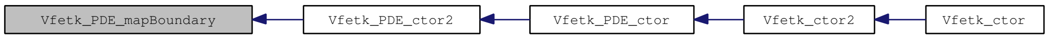
dimII Embedding dimension of manifold

vertexType Type of vertex

chart Chart for vertex coordinates

vx Vertex coordinates

Here is the caller graph for this function:



8.2.3.36 VEXTERNC int Vfetk_PDE_markSimplex (int *dim*, int *dimII*, int *simplexType*, int *faceType*[VAPBS_NVS], int *vertexType*[VAPBS_NVS], int *chart*[ ], double *vx*[ ][VAPBS_DIM], void ∗ *simplex*)

User-defined error estimator -- in our case, a geometry-based refinement method; forcing simplex refinement at the dielectric boundary and (for non-regularized PBE) the charges.

Author

Nathan Baker

Returns

1 if mark simplex for refinement, 0 otherwise

Bug

This function is not thread-safe

Parameters

dim Intrinsic manifold dimension

dimII Embedding manifold dimension

simplexType Type of simplex being refined

faceType Types of faces in simplex

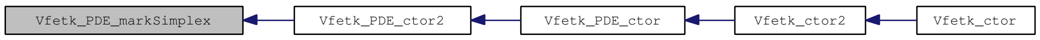
vertexType Types of vertices in simplex

chart Charts for vertex coordinates

vx Vertex coordinates

simplex Simplex pointer

Here is the caller graph for this function:



8.2.3.37 VEXTERNC void Vfetk_PDE_oneChart (int *dim*, int *dimII*, int *objType*, int *chart*[ ], double *vx*[ ][VAPBS_DIM], int *dimV*)

Unify the chart for different coordinate systems -- a no-op for us.

Author

Nathan Baker

Note

Thread-safe; a no-op

Parameters

dim Intrinsic manifold dimension

dimII Embedding manifold dimension

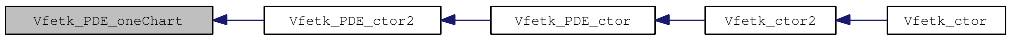
objType ???

chart Charts of vertices' coordinates

vx Vertices' coordinates

dimV Number of vertices

Here is the caller graph for this function:



8.2.3.38 VEXTERNC void Vfetk_PDE_simplexBasisForm (int *key*, int *dim*, int *comp*, int *pdkey*, double *xq*[ ], double *basis*[ ])

Evaluate the bases for the trial or test space, for a particular component of the system, at all quadrature points on the master simplex element.

Author

Mike Holst

Parameters

key Basis type to evaluate (0 = trial, 1 = test, 2 = trialB, 3 = testB)

dim Spatial dimension

comp Which component of elliptic system to produce basis for

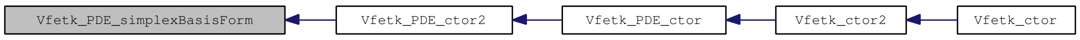
pdkey Basis partial differential equation evaluation key:

- 0 = evaluate basis(x,y,z)
- 1 = evaluate basis_x(x,y,z)
- 2 = evaluate basis_y(x,y,z)
- 3 = evaluate basis_z(x,y,z)
- 4 = evaluate basis_xx(x,y,z)
- 5 = evaluate basis_yy(x,y,z)
- 6 = evaluate basis_zz(x,y,z)
- 7 = etc...

xq Set to quad pt coordinate

basis Set to all basis functions evaluated at all quadrature pts

Here is the caller graph for this function:



8.2.3.39 VEXTERNC int Vfetk_PDE_simplexBasisInit (int *key*, int *dim*, int *comp*, int ∗ *ndof*, int *dof*[ ])

Initialize the bases for the trial or the test space, for a particular component of the system, at all quadrature points on the master simplex element.

Author

Mike Holst

Note

```
*   The basis ordering is important.  For a fixed quadrature
*   point iq, you must follow the following ordering in p[iq][],
*   based on how you specify the degrees of freedom in dof[]:
*
*   <v_0 vDF_0>,      <v_1 vDF_0>,      ..., <v_{nv} vDF_0>
*   <v_0 vDF_1>,      <v_1 vDF_1>,      ..., <v_{nv} vDF_1>
*                           ...
*   <v_0 vDF_{nvDF}>, <v_0 vDF_{nvDF}>, ..., <v_{nv} vDF_{nvDF}>
*
*   <e_0 eDF_0>,      <e_1 eDF_0>,      ..., <e_{ne} eDF_0>
*   <e_0 eDF_1>,      <e_1 eDF_1>,      ..., <e_{ne} eDF_1>
*                           ...
*   <e_0 eDF_{neDF}>, <e_1 eDF_{neDF}>, ..., <e_{ne} eDF_{neDF}>
*
*   <f_0 fDF_0>,      <f_1 fDF_0>,      ..., <f_{nf} fDF_0>
*   <f_0 fDF_1>,      <f_1 fDF_1>,      ..., <f_{nf} fDF_1>
*                           ...
*   <f_0 fDF_{nfDF}>, <f_1 fDF_{nfDF}>, ..., <f_{nf} fDF_{nfDF}>
*
*   <s_0 sDF_0>,      <s_1 sDF_0>,      ..., <s_{ns} sDF_0>
*   <s_0 sDF_1>,      <s_1 sDF_1>,      ..., <s_{ns} sDF_1>
*                           ...
*   <s_0 sDF_{nsDF}>, <s_1 sDF_{nsDF}>, ..., <s_{ns} sDF_{nsDF}>
*
*   For example, linear elements in R^3, with one degree of freedom at each *
*   vertex, would use the following ordering:
*
*     <v_0 vDF_0>, <v_1 vDF_0>, <v_2 vDF_0>, <v_3 vDF_0>
*
*   Quadratic elements in R^2, with one degree of freedom at each vertex and
*   edge, would use the following ordering:
*
*     <v_0 vDF_0>, <v_1 vDF_0>, <v_2 vDF_0>
*     <e_0 eDF_0>, <e_1 eDF_0>, <e_2 eDF_0>
*
*   You can use different trial and test spaces for each component of the
*   elliptic system, thereby allowing for the use of Petrov-Galerkin methods.
*   You MUST then tag the bilinear form symmetry entries as nonsymmetric in
*   your PDE constructor to reflect that DF(u)(w,v) will be different from
*   DF(u)(v,w), even if your form acts symmetrically when the same basis is
*   used for w and v.
*
*   You can also use different trial spaces for each component of the elliptic
*   system, and different test spaces for each component of the elliptic
*   system.  This allows you to e.g.  use a basis which is vertex-based for
*   one component, and a basis which is edge-based for another.  This is
*   useful in fluid mechanics, eletromagnetics, or simply to play around with
*   different elements.
*
*   This function is called by MC to build new master elements whenever it
*   reads in a new mesh.  Therefore, this function does not have to be all
*   that fast, and e.g.  could involve symbolic computation.
*
```

Parameters

key Basis type to evaluate (0 = trial, 1 = test, 2 = trialB, 3 = testB)

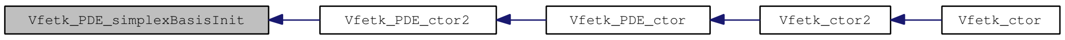
dim Spatial dimension

comp Which component of elliptic system to produce basis for?

ndof Set to the number of degrees of freedom

dof Set to degree of freedom per v/e/f/s

Here is the caller graph for this function:



8.2.3.40 VEXTERNC void Vfetk_PDE_u_D (PDE ∗ *thee*, int *type*, int *chart*, double *txq*[ ], double *F*[ ])

Evaluate the Dirichlet boundary condition at the given point.

Author

Nathan Baker

Bug

This function is hard-coded to call only multiple-sphere Debye-Hü functions.
This function is not thread-safe.

Parameters

thee PDE object

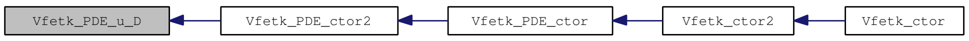
type Vertex boundary type

chart Chart for point coordinates

txq Point coordinates

F Set to boundary values

Here is the caller graph for this function:



8.2.3.41 VEXTERNC void Vfetk_PDE_u_T (PDE * *thee*, int *type*, int *chart*, double *txq*[ ], double *F*[ ])

Evaluate the "true solution" at the given point for comparison with the numerical solution.

Author

Nathan Baker

Note

This function only returns zero.

Bug

This function is not thread-safe.

Parameters

thee PDE object

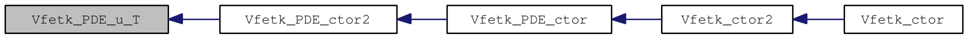
type Point type

chart Chart for point coordinates

txq Point coordinates

F Set to value at point

Here is the caller graph for this function:



8.2.3.42 VEXTERNC double Vfetk_qfEnergy (Vfetk * *thee*, int *color*)

Get the "fixed charge" contribution to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the fixed charges with the potential:

$$G = \sum_i q_i u(r_i)$$

and return the result in units of $k_B T$. Clearly, no self-interaction terms are removed. A factor a 1/2 has to be included to convert this to a real energy.

Author

Nathan Baker

Parameters

thee Vfetk object

color Partition restriction for energy evaluation, only used if non-negative

Returns

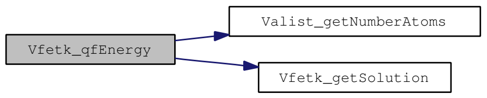
The fixed charge electrostatic energy in units of $k_B T$.

Parameters

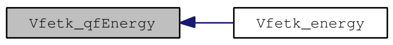
thee The Vfetk object

color Partition restriction for energy evaluation, only used if non-negative

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.43 VEXTERNC void Vfetk_readMesh (Vfetk ∗ *thee*, int *skey*, Vio ∗ *sock*)

Read in mesh and initialize associated internal structures.

Author

Nathan Baker

Note

See also

Vfetk_genCube

Parameters

thee THe Vfetk object

skey The sock format key (0 = MCSF simplex format)

sock Socket object ready for reading

8.2.3.44 VEXTERNC void Vfetk_setAtomColors (Vfetk ∗ *thee*)

Transfer color (partition ID) information frmo a partitioned mesh to the atoms.

Transfer color information from partitioned mesh to the atoms. In the case that a charge is shared between two partitions, the partition color of the first simplex is selected. Due to the arbitrary nature of this selection, THIS METHOD SHOULD ONLY BE USED IMMEDIATELY AFTER PARTITIONING!!!

Warning

This function should only be used immediately after mesh partitioning

Author

Nathan Baker

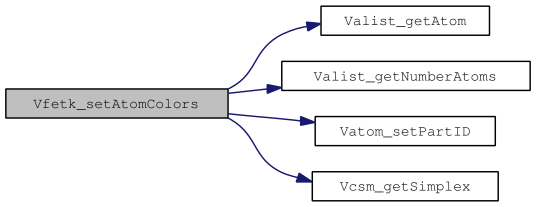
Note

This is a friend function of Vcsm

Parameters

thee THe Vfetk object

Here is the call graph for this function:



8.2.3.45 VEXTERNC void Vfetk_setParameters (Vfetk ∗ *thee*, PBEparm ∗ *pbeparm*, FEMparm ∗ *feparm*)

Set the parameter objects.

Author

Nathan Baker

Parameters

thee The Vfetk object
pbeparm Parameters for solution of the PBE
feparm FEM-speecific solution parameters

8.2.3.46 VEXTERNC int Vfetk_write (Vfetk ∗ *thee*, const char ∗ *iodev*, const char ∗ *iofmt*, const char ∗ *thost*, const char ∗ *fname*, Bvec ∗ *vec*, Vdata_Format *format*)

Write out data.

Author

Nathan Baker

Parameters

thee Vfetk object

vec FEtk Bvec vector to use

format Format for data

iodev Output device type (FILE/BUFF/UNIX/INET)

iofmt Output device format (ASCII/XDR)

thost Output hostname (for sockets)

fname Output FILE/BUFF/UNIX/INET name

Note

This function is thread-safe

Bug

Some values of format are not implemented

Returns

1 if successful, 0 otherwise

Parameters

thee The Vfetk object

iodev Output device type (FILE = file, BUFF = buffer, UNIX = unix pipe, INET = network socket)

iofmt Output device format (ASCII = ascii/plaintext, XDR = xdr)

thost Output hostname for sockets

fname Output filename for other

vec Data vector

format Data format

8.3 Vpee class

This class provides some functionality for error esimation in parallel.

Data Structures

- struct sVpee

  Contains public data members for Vpee class/module.

Files

- file vpee.h

  Contains declarations for class Vpee.

- file vpee.c

  Class Vpee methods.

Typedefs

- typedef struct sVpee Vpee

  Declaration of the Vpee class as the Vpee structure.

Functions

- VEXTERNC Vpee ∗ Vpee_ctor (Gem ∗gm, int localPartID, int killFlag, double killParam)

  Construct the Vpee object.

- VEXTERNC int Vpee_ctor2 (Vpee ∗thee, Gem ∗gm, int localPartID, int killFlag, double killParam)

  FORTRAN stub to construct the Vpee object.

- VEXTERNC void Vpee_dtor (Vpee ∗∗thee)

  Object destructor.

- VEXTERNC void Vpee_dtor2 (Vpee ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vpee_markRefine (Vpee ∗thee, AM ∗am, int level, int akey, int rcol, double etol, int bkey)

  Mark simplices for refinement based on attenuated error estimates.

- VEXTERNC int Vpee_numSS (Vpee ∗thee)

  Returns the number of simplices in the local partition.

8.3.1 Detailed Description

This class provides some functionality for error esimation in parallel. This class provides some functionality for error esimation in parallel. The purpose is to modulate the error returned by some external error estimator according to the partitioning of the mesh. For example, the Bank/Holst parallel refinement routine essentially reduces the error outside the “local" partition to zero. However, this leads to the need for a few final overlapping Schwarz solves to smooth out the errors near partition boundaries. Supposedly, if the region in which we allow error-based refinement includes the “local" partition and an external buffer zone approximately equal in size to the local region, then the solution will asymptotically approach the solution obtained via more typical methods. This is essentially a more flexible parallel implementation of MC’s AM_markRefine.

8.3.2 Function Documentation

8.3.2.1 VEXTERNC Vpee∗ Vpee_ctor (Gem ∗ *gm*, int *localPartID*, int *killFlag*, double *killParam*)

Construct the Vpee object.

Author

Nathan Baker

Returns

Newly constructed Vpee object

Parameters

gm FEtk geometry manager object

localPartID ID of the local partition (focus of refinement)

killFlag A flag to indicate how error estimates are to be attenuated outside the local partition:

- 0: no attenuation

- 1: all error outside the local partition set to zero
- 2: all error is set to zero outside a sphere of radius (killParam∗partRadius), where partRadius is the radius of the sphere circumscribing the local partition
- 3: all error is set to zero except for the local partition and its immediate neighbors

killParam

See also

killFlag for usage

8.3.2.2 VEXTERNC int Vpee_ctor2 (Vpee ∗ *thee*, Gem ∗ *gm*, int *localPartID*, int *killFlag*, double *killParam*)

FORTRAN stub to construct the Vpee object.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Parameters

thee The Vpee object

gm FEtk geometry manager object

localPartID ID of the local partition (focus of refinement)

killFlag A flag to indicate how error estimates are to be attenuated outside the local partition:

- 0: no attenuation
- 1: all error outside the local partition set to zero
- 2: all error is set to zero outside a sphere of radius (killParam∗partRadius), where partRadius is the radius of the sphere circumscribing the local partition
- 3: all error is set to zero except for the local partition and its immediate neighbors

killParam

See also

killFlag for usage

8.3.2.3 VEXTERNC void Vpee_dtor (Vpee ** *thee*)

Object destructor.

Author

Nathan Baker

Parameters

thee Pointer to memory location of the Vpee object

8.3.2.4 VEXTERNC void Vpee_dtor2 (Vpee * *thee*)

FORTRAN stub object destructor.

Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

8.3.2.5 VEXTERNC int Vpee_markRefine (Vpee * *thee*, AM * *am*, int *level*, int *akey*, int *rcol*, double *etol*, int *bkey*)

Mark simplices for refinement based on attenuated error estimates.

A wrapper/reimplementation of AM_markRefine that allows for more flexible attenuation of error-based markings outside the local partition. The error in each simplex is modified by the method (see killFlag) specified in the Vpee constructor. This allows the user to confine refinement to an arbitrary area around the local partition.

Author

Nathan Baker and Mike Holst

Note

This routine borrows very heavily from FEtk routines by Mike Holst.

Returns

The number of simplices marked for refinement.

Bug

This function is no longer up-to-date with FEtk and may not function properly

Parameters

thee The Vpee object

am The FEtk algebra manager currently used to solve the PB

level The current level of the multigrid hierarchy

akey The marking method:

- -1: Reset markings --> killFlag has no effect.
- 0: Uniform.
- 1: User defined (geometry-based).
- >1: A numerical estimate for the error has already been set in am and should be attenuated according to killFlag and used, in conjunction with etol, to mark simplices for refinement.

rcol The ID of the main parition on which to mark (or -1 if all partitions should be marked). NOte that we shouldhave (rcol == thee->localPartID) for (thee->killFlag == 2 or 3)

etol The error tolerance criterion for marking

bkey How the error tolerance is interpreted:

- 0: Simplex marked if error > etol.
- 1: Simplex marked if error > sqrt(etol^2/L) where L$ is the number of simplices

8.3.2.6 VEXTERNC int Vpee_numSS (Vpee * *thee*)

Returns the number of simplices in the local partition.

Author

Nathan Baker

Returns

Number of simplices in the local partition

Parameters

thee The Vpee object

8.4 APOLparm class

Parameter structure for APOL-specific variables from input files.

Data Structures

- struct sAPOLparm

  Parameter structure for APOL-specific variables from input files.

Files

- file femparm.h

  Contains declarations for class APOLparm.

- file apolparm.c

  Class APOLparm methods.

Typedefs

- typedef enum eAPOLparm_calcEnergy APOLparm_calcEnergy

  Define eAPOLparm_calcEnergy enumeration as APOLparm_calcEnergy.

- typedef enum eAPOLparm_calcForce APOLparm_calcForce

  Define eAPOLparm_calcForce enumeration as APOLparm_calcForce.

- typedef enum eAPOLparm_doCalc APOLparm_doCalc

  Define eAPOLparm_calcForce enumeration as APOLparm_calcForce.

- typedef struct sAPOLparm APOLparm

  Declaration of the APOLparm class as the APOLparm structure.

Enumerations

- enum eAPOLparm_calcEnergy { ACE_NO = 0, ACE_TOTAL = 1, ACE_-COMPS = 2 }

  Define energy calculation enumeration.

- enum eAPOLparm_calcForce { ACF_NO = 0, ACF_TOTAL = 1, ACF_COMPS = 2 }

  Define force calculation enumeration.

- enum eAPOLparm_doCalc { ACD_NO = 0, ACD_YES = 1, ACD_ERROR = 2 }

  Define force calculation enumeration.

Functions

- VEXTERNC APOLparm ∗ APOLparm_ctor ()

  Construct APOLparm.

- VEXTERNC Vrc_Codes APOLparm_ctor2 (APOLparm ∗thee)

  FORTRAN stub to construct APOLparm.

- VEXTERNC void APOLparm_dtor (APOLparm ∗∗thee)

  Object destructor.

- VEXTERNC void APOLparm_dtor2 (APOLparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC Vrc_Codes APOLparm_check (APOLparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void APOLparm_copy (APOLparm ∗thee, APOLparm ∗source)

  Copy target object into thee.

8.4.1 Detailed Description

Parameter structure for APOL-specific variables from input files.

8.4.2 Enumeration Type Documentation

8.4.2.1 enum eAPOLparm_calcEnergy

Define energy calculation enumeration.

Enumerator:

ACE_NO Do not perform energy calculation

ACE_TOTAL Calculate total energy only

ACE_COMPS Calculate per-atom energy components

8.4.2.2 enum eAPOLparm_calcForce

Define force calculation enumeration.

Enumerator:

ACF_NO Do not perform force calculation

ACF_TOTAL Calculate total force only

ACF_COMPS Calculate per-atom force components

8.4.2.3 enum eAPOLparm_doCalc

Define force calculation enumeration.

Enumerator:

ACD_NO Do not perform calculation

ACD_YES Perform calculations

ACD_ERROR Error setting up calculation

8.4.3 Function Documentation

8.4.3.1 VEXTERNC Vrc_Codes APOLparm_check (APOLparm ∗ *thee*)

Consistency check for parameter values stored in object.

Author

David Gohara, Yong Huang

Parameters

thee APOLparm object

Returns

Success enumeration

8.4.3.2 VEXTERNC void APOLparm_copy (APOLparm ∗ *thee*, APOLparm ∗ *source*)

Copy target object into thee.

Author

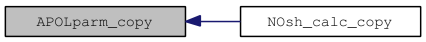
Nathan Baker

Parameters

thee Destination object

source Source object

Here is the caller graph for this function:



8.4.3.3 VEXTERNC APOLparm∗ APOLparm_ctor ()

Construct APOLparm.

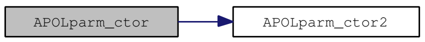
Author

David Gohara

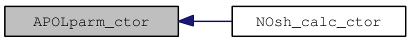
Returns

Newly allocated and initialized Vpmgp object

Here is the call graph for this function:



Here is the caller graph for this function:



8.4.3.4 VEXTERNC Vrc_Codes APOLparm_ctor2 (APOLparm ∗ *thee*)

FORTRAN stub to construct APOLparm.

Author

David Gohara, Yong Huang

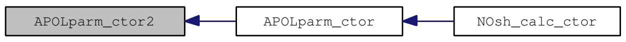
Parameters

thee Pointer to allocated APOLparm object

Returns

Success enumeration

Here is the caller graph for this function:



8.4.3.5 VEXTERNC void APOLparm_dtor (APOLparm ∗∗ *thee*)

Object destructor.

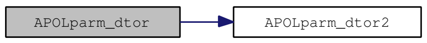
Author

David Gohara

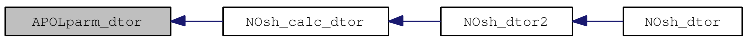
Parameters

thee Pointer to memory location of APOLparm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.4.3.6 VEXTERNC void APOLparm_dtor2 (APOLparm ∗ *thee*)

FORTRAN stub for object destructor.

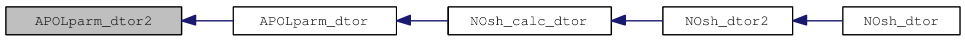
Author

David Gohara

Parameters

thee Pointer to APOLparm object

Here is the caller graph for this function:



8.5 FEMparm class

Parameter structure for FEM-specific variables from input files.

Data Structures

- struct sFEMparm

  Parameter structure for FEM-specific variables from input files.

Files

- file femparm.h

  Contains declarations for class APOLparm.

- file femparm.c

  Class FEMparm methods.

Typedefs

- typedef enum eFEMparm_EtolType FEMparm_EtolType

  Declare FEparm_EtolType type.

- typedef enum eFEMparm_EstType FEMparm_EstType

  Declare FEMparm_EstType type.

- typedef enum eFEMparm_CalcType FEMparm_CalcType

  Declare FEMparm_CalcType type.

- typedef struct sFEMparm FEMparm

  Declaration of the FEMparm class as the FEMparm structure.

Enumerations

- enum eFEMparm_EtolType { FET_SIMP = 0, FET_GLOB = 1, FET_FRAC = 2 }

  Adaptive refinment error estimate tolerance key.

- enum eFEMparm_EstType {
  FRT_UNIF = 0, FRT_GEOM = 1, FRT_RESI = 2, FRT_DUAL = 3,
  FRT_LOCA = 4 }

  Adaptive refinment error estimator method.
- enum eFEMparm_CalcType { FCT_MANUAL, FCT_NONE }

  Calculation type.

Functions

- VEXTERNC FEMparm ∗ FEMparm_ctor (FEMparm_CalcType type)

  Construct FEMparm.
- VEXTERNC int FEMparm_ctor2 (FEMparm ∗thee, FEMparm_CalcType type)

  FORTRAN stub to construct FEMparm.
- VEXTERNC void FEMparm_dtor (FEMparm ∗∗thee)

  Object destructor.
- VEXTERNC void FEMparm_dtor2 (FEMparm ∗thee)

  FORTRAN stub for object destructor.
- VEXTERNC int FEMparm_check (FEMparm ∗thee)

  Consistency check for parameter values stored in object.
- VEXTERNC void FEMparm_copy (FEMparm ∗thee, FEMparm ∗source)

  Copy target object into thee.

8.5.1 Detailed Description

Parameter structure for FEM-specific variables from input files.

8.5.2 Typedef Documentation

8.5.2.1 typedef enum eFEMparm_EtolType FEMparm_EtolType

Declare FEparm_EtolType type.

Author

Nathan Baker

8.5.3 Enumeration Type Documentation

8.5.3.1 enum eFEMparm_CalcType

Calculation type.

Enumerator:

FCT_MANUAL fe-manual

FCT_NONE unspecified

8.5.3.2 enum eFEMparm_EstType

Adaptive refinment error estimator method.

Note

Do not change these values; they correspond to settings in FEtk

Author

Nathan Baker

Enumerator:

FRT_UNIF Uniform refinement

FRT_GEOM Geometry-based (i.e. surfaces and charges) refinement

FRT_RESI Nonlinear residual estimate-based refinement

FRT_DUAL Dual-solution weight nonlinear residual estimate-based refinement

FRT_LOCA Local problem error estimate-based refinement

8.5.3.3 enum eFEMparm_EtolType

Adaptive refinment error estimate tolerance key.

Author

Nathan Baker

Enumerator:

FET_SIMP per-simplex error tolerance

FET_GLOB global error tolerance

FET_FRAC fraction of simplices we want to have refined

8.5.4 Function Documentation

8.5.4.1 VEXTERNC int FEMparm_check (FEMparm ∗ *thee*)

Consistency check for parameter values stored in object.

Author

Nathan Baker

Parameters

thee FEMparm object

Returns

1 if OK, 0 otherwise

8.5.4.2 VEXTERNC void FEMparm_copy (FEMparm ∗ *thee*, FEMparm ∗ *source*)

Copy target object into thee.

Author

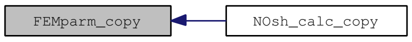
Nathan Baker

Parameters

thee Destination object

source Source object

Here is the caller graph for this function:



8.5.4.3 VEXTERNC FEMparm∗ FEMparm_ctor (FEMparm_CalcType *type*)

Construct FEMparm.

Author

Nathan Baker

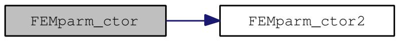
Parameters

type FEM calculation type

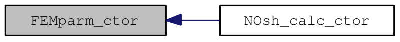
Returns

Newly allocated and initialized Vpmgp object

Here is the call graph for this function:



Here is the caller graph for this function:



8.5.4.4 VEXTERNC int FEMparm_ctor2 (FEMparm ∗ *thee*, FEMparm_CalcType *type*)

FORTRAN stub to construct FEMparm.

Author

Nathan Baker

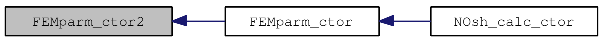
Parameters

thee Pointer to allocated FEMparm object
type FEM calculation type

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.5.4.5 VEXTERNC void FEMparm_dtor (FEMparm $**$ *thee*)

Object destructor.

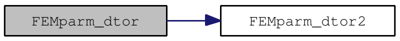
Author

Nathan Baker

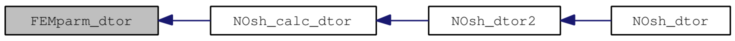
Parameters

thee Pointer to memory location of FEMparm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.5.4.6 VEXTERNC void FEMparm_dtor2 (FEMparm $*$ *thee*)

FORTRAN stub for object destructor.

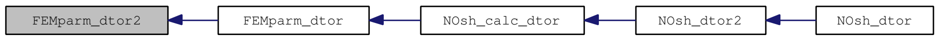
Author

Nathan Baker

Parameters

thee Pointer to FEMparm object

Here is the caller graph for this function:



8.6 MGparm class

Parameter which holds useful parameters for generic multigrid calculations.

Data Structures

- struct sMGparm

  Parameter structure for MG-specific variables from input files.

Files

- file mgparm.h

  Contains declarations for class MGparm.

- file mgparm.c

  Class MGparm methods.

Typedefs

- typedef enum eMGparm_CalcType MGparm_CalcType

  Declare MGparm_CalcType type.

- typedef enum eMGparm_CentMeth MGparm_CentMeth

  Declare MGparm_CentMeth type.

- typedef struct sMGparm MGparm

  Declaration of the MGparm class as the MGparm structure.

Enumerations

- enum eMGparm_CalcType {
  MCT_MANUAL = 0, MCT_AUTO = 1, MCT_PARALLEL = 2, MCT_-DUMMY = 3,
  MCT_NONE = 4 }

  Calculation type.

- enum eMGparm_CentMeth { MCM_POINT = 0, MCM_MOLECULE = 1, MCM_FOCUS = 2 }

Centering method.

Functions

- VEXTERNC Vrc_Codes APOLparm_parseToken (APOLparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

- VEXTERNC Vrc_Codes FEMparm_parseToken (FEMparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

- VEXTERNC int MGparm_getNx (MGparm ∗thee)

  Get number of grid points in x direction.

- VEXTERNC int MGparm_getNy (MGparm ∗thee)

  Get number of grid points in y direction.

- VEXTERNC int MGparm_getNz (MGparm ∗thee)

  Get number of grid points in z direction.

- VEXTERNC double MGparm_getHx (MGparm ∗thee)

  Get grid spacing in x direction (Å).

- VEXTERNC double MGparm_getHy (MGparm ∗thee)

  Get grid spacing in y direction (Å).

- VEXTERNC double MGparm_getHz (MGparm ∗thee)

  Get grid spacing in z direction (Å).

- VEXTERNC void MGparm_setCenterX (MGparm ∗thee, double x)

  Set center x-coordinate.

- VEXTERNC void MGparm_setCenterY (MGparm ∗thee, double y)

  Set center y-coordinate.

- VEXTERNC void MGparm_setCenterZ (MGparm ∗thee, double z)

  Set center z-coordinate.

- VEXTERNC double MGparm_getCenterX (MGparm ∗thee)

  Get center x-coordinate.

- VEXTERNC double MGparm_getCenterY (MGparm ∗thee)

  Get center y-coordinate.

- VEXTERNC double MGparm_getCenterZ (MGparm ∗thee)

  Get center z-coordinate.

- VEXTERNC MGparm ∗ MGparm_ctor (MGparm_CalcType type)

  Construct MGparm object.

- VEXTERNC Vrc_Codes MGparm_ctor2 (MGparm ∗thee, MGparm_CalcType type)

  FORTRAN stub to construct MGparm object.

- VEXTERNC void MGparm_dtor (MGparm ∗∗thee)

  Object destructor.

- VEXTERNC void MGparm_dtor2 (MGparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC Vrc_Codes MGparm_check (MGparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void MGparm_copy (MGparm ∗thee, MGparm ∗parm)

  Copy MGparm object into thee.

- VEXTERNC Vrc_Codes MGparm_parseToken (MGparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

8.6.1 Detailed Description

Parameter which holds useful parameters for generic multigrid calculations.

8.6.2 Enumeration Type Documentation

8.6.2.1 enum eMGparm_CalcType

Calculation type.

Enumerator:

- ***MCT_MANUAL*** mg-manual
- ***MCT_AUTO*** mg-auto
- ***MCT_PARALLEL*** mg-para
- ***MCT_DUMMY*** mg-dummy
- ***MCT_NONE*** unspecified

8.6.2.2 enum eMGparm_CentMeth

Centering method.

Enumerator:

- ***MCM_POINT*** Center on a point
- ***MCM_MOLECULE*** Center on a molecule
- ***MCM_FOCUS*** Determined by focusing

8.6.3 Function Documentation

8.6.3.1 VEXTERNC Vrc_Codes APOLparm_parseToken (APOLparm ∗ *thee*, char *tok*[VMAX_BUFSIZE], Vio ∗ *sock*)

Parse an MG keyword from an input file.

Author

David Gohara

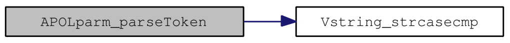
Parameters

- ***thee*** MGparm object
- ***tok*** Token to parse
- ***sock*** Stream for more tokens

Returns

Success enumeration (1 if matched and assigned; -1 if matched, but there's some sort of error (i.e., too few args); 0 if not matched)

Here is the call graph for this function:



8.6.3.2 VEXTERNC Vrc_Codes FEMparm_parseToken (FEMparm ∗ *thee*, char *tok*[VMAX_BUFSIZE], Vio ∗ *sock*)

Parse an MG keyword from an input file.

Author

Nathan Baker

Parameters

thee MGparm object

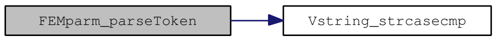
tok Token to parse

sock Stream for more tokens

Returns

VRC_SUCCESS if matched and assigned; VRC_FAILURE if matched, but there's some sort of error (i.e., too few args); VRC_WARNING if not matched

Here is the call graph for this function:



8.6.3.3 VEXTERNC Vrc_Codes MGparm_check (MGparm ∗ *thee*)

Consistency check for parameter values stored in object.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Success enumeration

8.6.3.4 VEXTERNC void MGparm_copy (MGparm ∗ *thee*, MGparm ∗ *parm*)

Copy MGparm object into thee.

Author

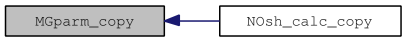
Nathan Baker and Todd Dolinsky

Parameters

thee MGparm object (target for copy)

parm MGparm object (source for copy)

Here is the caller graph for this function:



8.6.3.5 VEXTERNC MGparm∗ MGparm_ctor (MGparm_CalcType *type*)

Construct MGparm object.

Author

Nathan Baker

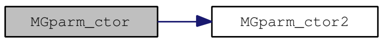
Parameters

type Type of MG calculation

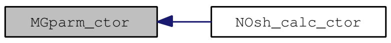
Returns

Newly allocated and initialized MGparm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.6 VEXTERNC Vrc_Codes MGparm_ctor2 (MGparm ∗ *thee*, MGparm_CalcType *type*)

FORTRAN stub to construct MGparm object.

Author

Nathan Baker and Todd Dolinsky

Parameters

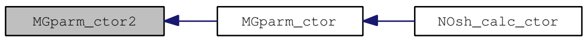
thee Space for MGparm object

type Type of MG calculation

Returns

Success enumeration

Here is the caller graph for this function:



8.6.3.7 VEXTERNC void MGparm_dtor (MGparm ∗∗ *thee*)

Object destructor.

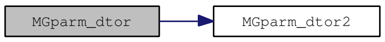
Author

Nathan Baker

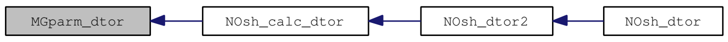
Parameters

thee Pointer to memory location of MGparm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.8 VEXTERNC void MGparm_dtor2 (MGparm ∗ *thee*)

FORTRAN stub for object destructor.

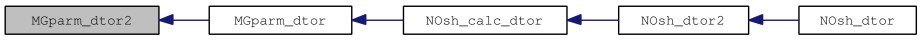
Author

Nathan Baker

Parameters

thee Pointer to MGparm object

Here is the caller graph for this function:



8.6.3.9 VEXTERNC double MGparm_getCenterX (MGparm ∗ *thee*)

Get center x-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

x-coordinate

8.6.3.10 VEXTERNC double MGparm_getCenterY (MGparm ∗ *thee*)

Get center y-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

y-coordinate

8.6.3.11 VEXTERNC double MGparm_getCenterZ (MGparm ∗ *thee*)

Get center z-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

z-coordinate

8.6.3.12 VEXTERNC double MGparm_getHx (MGparm ∗ *thee*)

Get grid spacing in x direction (Å).

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Grid spacing in the x direction

8.6.3.13 VEXTERNC double MGparm_getHy (MGparm ∗ *thee*)

Get grid spacing in y direction (Å).

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Grid spacing in the y direction

8.6.3.14 VEXTERNC double MGparm_getHz (MGparm * *thee*)

Get grid spacing in z direction (Å).

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Grid spacing in the z direction

8.6.3.15 VEXTERNC int MGparm_getNx (MGparm * *thee*)

Get number of grid points in x direction.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Number of grid points in the x direction

8.6.3.16 VEXTERNC int MGparm_getNy (MGparm * *thee*)

Get number of grid points in y direction.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Number of grid points in the y direction

8.6.3.17 VEXTERNC int MGparm_getNz (MGparm ∗ *thee*)

Get number of grid points in z direction.

Author

Nathan Baker

Parameters

thee MGparm object

Returns

Number of grid points in the z direction

8.6.3.18 VEXTERNC Vrc_Codes MGparm_parseToken (MGparm ∗ *thee*, char *tok*[VMAX_BUFSIZE], Vio ∗ *sock*)

Parse an MG keyword from an input file.

Author

Nathan Baker and Todd Dolinsky

Parameters

thee MGparm object

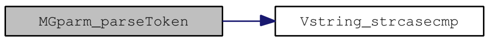
tok Token to parse

sock Stream for more tokens

Returns

Success enumeration (1 if matched and assigned; -1 if matched, but there's some sort of error (i.e., too few args); 0 if not matched)

Here is the call graph for this function:



8.6.3.19 VEXTERNC void MGparm_setCenterX (MGparm ∗ *thee*, double *x*)

Set center x-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

x x-coordinate

8.6.3.20 VEXTERNC void MGparm_setCenterY (MGparm ∗ *thee*, double *y*)

Set center y-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

y y-coordinate

8.6.3.21 VEXTERNC void MGparm_setCenterZ (MGparm ∗ *thee*, double *z*)

Set center z-coordinate.

Author

Nathan Baker

Parameters

thee MGparm object

z z-coordinate

8.7 NOsh class

Class for parsing for fixed format input files.

Data Structures

- struct sNOsh_calc

  Calculation class for use when parsing fixed format input files.

- struct sNOsh

  Class for parsing fixed format input files.

Files

- file nosh.h

  Contains declarations for class NOsh.

- file nosh.c

  Class NOsh methods.

Defines

- #define NOSH_MAXMOL 20

  Maximum number of molecules in a run.

- #define NOSH_MAXCALC 20

  Maximum number of calculations in a run.

- #define NOSH_MAXPRINT 20

  Maximum number of PRINT statements in a run.

- #define NOSH_MAXPOP 20

  Maximum number of operations in a PRINT statement.

Typedefs

- typedef enum eNOsh_MolFormat NOsh_MolFormat

  Declare NOsh_MolFormat type.

- typedef enum eNOsh_CalcType NOsh_CalcType

    Declare NOsh_CalcType type.

- typedef enum eNOsh_ParmFormat NOsh_ParmFormat

    Declare NOsh_ParmFormat type.

- typedef enum eNOsh_PrintType NOsh_PrintType

    Declare NOsh_PrintType type.

- typedef struct sNOsh NOsh

    Declaration of the NOsh class as the NOsh structure.

- typedef struct sNOsh_calc NOsh_calc

    Declaration of the NOsh_calc class as the NOsh_calc structure.

Enumerations

- enum eNOsh_MolFormat { NMF_PQR = 0, NMF_PDB = 1, NMF_XML = 2 }

    Molecule file format types.

- enum eNOsh_CalcType { NCT_MG = 0, NCT_FEM = 1, NCT_APOL = 2 }

    NOsh calculation types.

- enum eNOsh_ParmFormat { NPF_FLAT = 0, NPF_XML = 1 }

    Parameter file format types.

- enum eNOsh_PrintType {

    NPT_ENERGY = 0, NPT_FORCE = 1, NPT_ELECENERGY, NPT_-ELECFORCE,

    NPT_APOLENERGY, NPT_APOLFORCE }

    NOsh print types.

Functions

- VEXTERNC char ∗ NOsh_getMolpath (NOsh ∗thee, int imol)

    Returns path to specified molecule.

- VEXTERNC char ∗ NOsh_getDielXpath (NOsh ∗thee, int imap)

  Returns path to specified x-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getDielYpath (NOsh ∗thee, int imap)

  Returns path to specified y-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getDielZpath (NOsh ∗thee, int imap)

  Returns path to specified z-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getKappapath (NOsh ∗thee, int imap)

  Returns path to specified kappa map.

- VEXTERNC char ∗ NOsh_getChargepath (NOsh ∗thee, int imap)

  Returns path to specified charge distribution map.

- VEXTERNC NOsh_calc ∗ NOsh_getCalc (NOsh ∗thee, int icalc)

  Returns specified calculation object.

- VEXTERNC int NOsh_getDielfmt (NOsh ∗thee, int imap)

  Returns format of specified dielectric map.

- VEXTERNC int NOsh_getKappafmt (NOsh ∗thee, int imap)

  Returns format of specified kappa map.

- VEXTERNC int NOsh_getChargefmt (NOsh ∗thee, int imap)

  Returns format of specified charge map.

- VEXTERNC NOsh_PrintType NOsh_printWhat (NOsh ∗thee, int iprint)

  Return an integer ID of the observable to print (.

- VEXTERNC char ∗ NOsh_elecname (NOsh ∗thee, int ielec)

  Return an integer mapping of an ELEC statement to a calculation ID (.

- VEXTERNC int NOsh_elec2calc (NOsh ∗thee, int icalc)

  Return the name of an elec statement.

- VEXTERNC int NOsh_apol2calc (NOsh ∗thee, int icalc)

  Return the name of an apol statement.

- VEXTERNC int NOsh_printNarg (NOsh ∗thee, int iprint)

  Return number of arguments to PRINT statement (.

- VEXTERNC int NOsh_printOp (NOsh ∗thee, int iprint, int iarg)

  Return integer ID for specified operation (.

- VEXTERNC int NOsh_printCalc (NOsh ∗thee, int iprint, int iarg)

  Return calculation ID for specified PRINT statement (.

- VEXTERNC NOsh ∗ NOsh_ctor (int rank, int size)

  Construct NOsh.

- VEXTERNC NOsh_calc ∗ NOsh_calc_ctor (NOsh_CalcType calcType)

  Construct NOsh_calc.

- VEXTERNC int NOsh_calc_copy (NOsh_calc ∗thee, NOsh_calc ∗source)

  Copy NOsh_calc object into thee.

- VEXTERNC void NOsh_calc_dtor (NOsh_calc ∗∗thee)

  Object destructor.

- VEXTERNC int NOsh_ctor2 (NOsh ∗thee, int rank, int size)

  FORTRAN stub to construct NOsh.

- VEXTERNC void NOsh_dtor (NOsh ∗∗thee)

  Object destructor.

- VEXTERNC void NOsh_dtor2 (NOsh ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC int NOsh_parseInput (NOsh ∗thee, Vio ∗sock)

  Parse an input file from a socket.

- VEXTERNC int NOsh_parseInputFile (NOsh ∗thee, char ∗filename)

  Parse an input file only from a file.

- VEXTERNC int NOsh_setupElecCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of electrostatics calculations.

- VEXTERNC int NOsh_setupApolCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of non-polar calculations.

8.7.1 Detailed Description

Class for parsing for fixed format input files.

8.7.2 Enumeration Type Documentation

8.7.2.1 enum eNOsh_CalcType

NOsh calculation types.

Enumerator:

NCT_MG	Multigrid
NCT_FEM	Finite element
NCT_APOL	non-polar

8.7.2.2 enum eNOsh_MolFormat

Molecule file format types.

Enumerator:

NMF_PQR	PQR format
NMF_PDB	PDB format
NMF_XML	XML format

8.7.2.3 enum eNOsh_ParmFormat

Parameter file format types.

Enumerator:

NPF_FLAT	Flat-file format
NPF_XML	XML format

8.7.2.4 enum eNOsh_PrintType

NOsh print types.

Enumerator:

NPT_ENERGY Energy (deprecated)

NPT_FORCE Force (deprecated)

NPT_ELECENERGY Elec Energy

NPT_ELECFORCE Elec Force

NPT_APOLENERGY Apol Energy

NPT_APOLFORCE Apol Force

8.7.3 Function Documentation

8.7.3.1 VEXTERNC int NOsh_apol2calc (NOsh ∗ *thee*, int *icalc*)

Return the name of an apol statement.

Author

David Gohara

Parameters

thee NOsh object to use

icalc ID of CALC statement

Returns

The name (if present) of an APOL statement

8.7.3.2 VEXTERNC int NOsh_calc_copy (NOsh_calc ∗ *thee*, NOsh_calc ∗ *source*)

Copy NOsh_calc object into thee.

Author

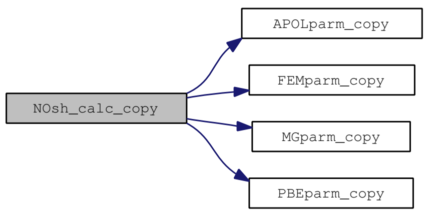
Nathan Baker

Parameters

thee Target object

source Source object

Here is the call graph for this function:



8.7.3.3 VEXTERNC NOsh_calc* NOsh_calc_ctor (NOsh_CalcType *calcType*)

Construct NOsh_calc.

Author

Nathan Baker

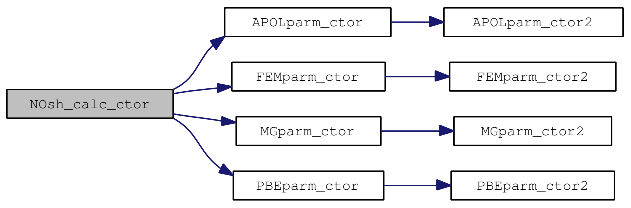
Parameters

calcType Calculation type

Returns

Newly allocated and initialized NOsh object

Here is the call graph for this function:



8.7.3.4 VEXTERNC void NOsh_calc_dtor (NOsh_calc ** *thee*)

Object destructor.

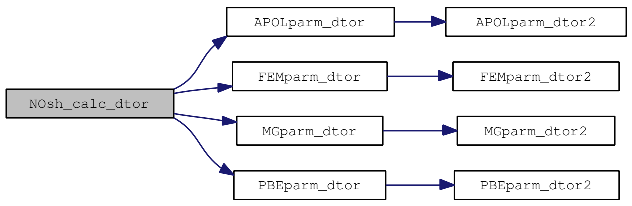
Author

Nathan Baker

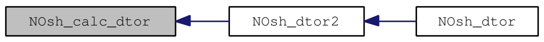
Parameters

thee Pointer to memory location of NOsh_calc object

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.5 VEXTERNC NOsh∗ NOsh_ctor (int *rank*, int *size*)

Construct NOsh.

Author

Nathan Baker

Parameters

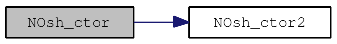
rank Rank of current processor in parallel calculation (0 if not parallel)

size Number of processors in parallel calculation (1 if not parallel)

Returns

Newly allocated and initialized NOsh object

Here is the call graph for this function:



8.7.3.6 VEXTERNC int NOsh_ctor2 (NOsh ∗ *thee*, int *rank*, int *size*)

FORTRAN stub to construct NOsh.

Author

Nathan Baker

Parameters

thee Space for NOsh objet

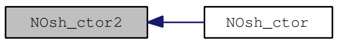
rank Rank of current processor in parallel calculation (0 if not parallel)

size Number of processors in parallel calculation (1 if not parallel)

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.7.3.7 VEXTERNC void NOsh_dtor (NOsh ∗∗ *thee*)

Object destructor.

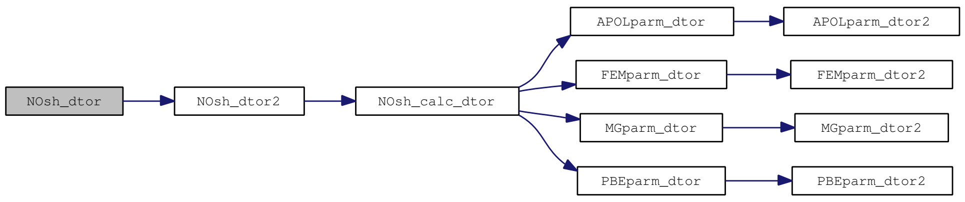
Author

Nathan Baker

Parameters

thee Pointer to memory location of NOsh object

Here is the call graph for this function:



8.7.3.8 VEXTERNC void NOsh_dtor2 (NOsh ∗ *thee*)

FORTRAN stub for object destructor.

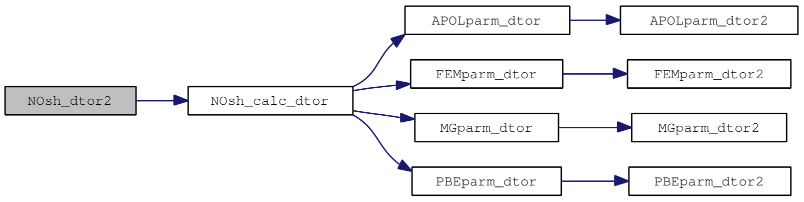
Author

Nathan Baker

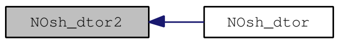
Parameters

thee Pointer to NOsh object

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.9 VEXTERNC int NOsh_elec2calc (NOsh ∗ *thee*, int *icalc*)

Return the name of an elec statement.

Author

Todd Dolinsky

Parameters

thee NOsh object to use

icalc ID of CALC statement

Returns

The name (if present) of an ELEC statement

8.7.3.10 VEXTERNC char∗ NOsh_elecname (NOsh ∗ *thee*, int *ielec*)

Return an integer mapping of an ELEC statement to a calculation ID (.

See also

elec2calc)

Author

Nathan Baker

Parameters

thee NOsh object to use

ielec ID of ELEC statement

Returns

An integer mapping of an ELEC statement to a calculation ID (

See also

elec2calc)

8.7.3.11 VEXTERNC NOsh_calc∗ NOsh_getCalc (NOsh ∗ *thee*, int *icalc*)

Returns specified calculation object.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object

icalc Calculation ID of interest

Returns

Pointer to specified calculation object

8.7.3.12 VEXTERNC int NOsh_getChargefmt (NOsh ∗ *thee*, int *imap*)

Returns format of specified charge map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object

imap Calculation ID of interest

Returns

Format of charge map

8.7.3.13 VEXTERNC char∗ NOsh_getChargepath (NOsh ∗ *thee*, int *imap*)

Returns path to specified charge distribution map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object

imap Map ID of interest

Returns

Path string

8.7.3.14 VEXTERNC int NOsh_getDielfmt (NOsh ∗ *thee*, int *imap*)

Returns format of specified dielectric map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object

imap Calculation ID of interest

Returns

Format of dielectric map

8.7.3.15 VEXTERNC char∗ NOsh_getDielXpath (NOsh ∗ *thee*, int *imap*)

Returns path to specified x-shifted dielectric map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imap Map ID of interest

Returns

Path string

8.7.3.16 VEXTERNC char∗ NOsh_getDielYpath (NOsh ∗ *thee*, int *imap*)

Returns path to specified y-shifted dielectric map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imap Map ID of interest

Returns

Path string

8.7.3.17 VEXTERNC char∗ NOsh_getDielZpath (NOsh ∗ *thee*, int *imap*)

Returns path to specified z-shifted dielectric map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imap Map ID of interest

Returns

Path string

8.7.3.18 VEXTERNC int NOsh_getKappafmt (NOsh * *thee*, int *imap*)

Returns format of specified kappa map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imap Calculation ID of interest

Returns

Format of kappa map

8.7.3.19 VEXTERNC char* NOsh_getKappapath (NOsh * *thee*, int *imap*)

Returns path to specified kappa map.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imap Map ID of interest

Returns

Path string

8.7.3.20 VEXTERNC char* NOsh_getMolpath (NOsh * *thee*, int *imol*)

Returns path to specified molecule.

Author

Nathan Baker

Parameters

thee Pointer to NOsh object
imol Molecule ID of interest

Returns

Path string

8.7.3.21 VEXTERNC int NOsh_parseInput (NOsh ∗ *thee*, Vio ∗ *sock*)

Parse an input file from a socket.

Note

Should be called before NOsh_setupCalc

Author

Nathan Baker and Todd Dolinsky

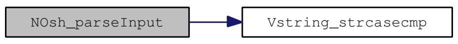
Parameters

thee Pointer to NOsh object
sock Stream of tokens to parse

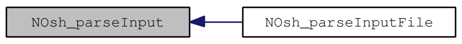
Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.22 VEXTERNC int NOsh_parseInputFile (NOsh ∗ *thee*, char ∗ *filename*)

Parse an input file only from a file.

Note

Included for SWIG wrapper compatibility
Should be called before NOsh_setupCalc

Author

Nathan Baker and Todd Dolinsky

Parameters

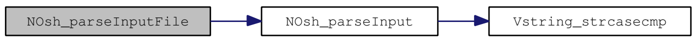
thee Pointer to NOsh object

filename Name/path of readable file

Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



8.7.3.23 VEXTERNC int NOsh_printCalc (NOsh ∗ *thee*, int *iprint*, int *iarg*)

Return calculation ID for specified PRINT statement (.

See also

printcalc)

Author

Nathan Baker

Parameters

thee NOsh object to use

iprint ID of PRINT statement

iarg ID of operation in PRINT statement

Returns

Calculation ID for specified PRINT statement (

See also

printcalc)

8.7.3.24 VEXTERNC int NOsh_printNarg (NOsh ∗ *thee*, int *iprint*)

Return number of arguments to PRINT statement (.

See also

printnarg)

Author

Nathan Baker

Parameters

thee NOsh object to use

iprint ID of PRINT statement

Returns

Number of arguments to PRINT statement (

See also

printnarg)

8.7.3.25 VEXTERNC int NOsh_printOp (NOsh ∗ *thee*, int *iprint*, int *iarg*)

Return integer ID for specified operation (.

See also

printop)

Author

Nathan Baker

Parameters

thee NOsh object to use

iprint ID of PRINT statement

iarg ID of operation in PRINT statement

Returns

Integer ID for specified operation (

See also

printop)

8.7.3.26 VEXTERNC NOsh_PrintType NOsh_printWhat (NOsh ∗ *thee*, int *iprint*)

Return an integer ID of the observable to print (.

See also

printwhat)

Author

Nathan Baker

Parameters

thee NOsh object to use

iprint ID of PRINT statement

Returns

An integer ID of the observable to print (

See also

printwhat)

8.7.3.27 VEXTERNC int NOsh_setupApolCalc (NOsh ∗ *thee*, Valist ∗ *alist*[NOSH_MAXMOL])

Setup the series of non-polar calculations.

Note

Should be called after NOsh_parseInput∗

Author

Nathan Baker and Todd Dolinsky

Parameters

thee Pointer to NOsh object

alist Array of pointers to Valist objects (molecules used to center mesh);

Returns

1 if successful, 0 otherwise

Parameters

thee NOsh object

alist Atom list for calculation

8.7.3.28 VEXTERNC int NOsh_setupElecCalc (NOsh ∗ *thee*, Valist ∗ *alist*[NOSH_MAXMOL])

Setup the series of electrostatics calculations.

Note

Should be called after NOsh_parseInput∗

Author

Nathan Baker and Todd Dolinsky

Parameters

thee Pointer to NOsh object

alist Array of pointers to Valist objects (molecules used to center mesh);

Returns

1 if successful, 0 otherwise

Parameters

thee NOsh object

alist Atom list for calculation

8.8 PBEparm class

Parameter structure for PBE variables independent of solver.

Data Structures

- struct sPBEparm

  Parameter structure for PBE variables from input files.

Files

- file pbeparm.h

  Contains declarations for class PBEparm.

- file pbeparm.c

  Class PBEparm methods.

Defines

- #define PBEPARM_MAXWRITE 20

  Number of things that can be written out in a single calculation.

Typedefs

- typedef enum ePBEparm_calcEnergy PBEparm_calcEnergy

  Define ePBEparm_calcEnergy enumeration as PBEparm_calcEnergy.

- typedef enum ePBEparm_calcForce PBEparm_calcForce

  Define ePBEparm_calcForce enumeration as PBEparm_calcForce.

- typedef struct sPBEparm PBEparm

  Declaration of the PBEparm class as the PBEparm structure.

Enumerations

- enum ePBEparm_calcEnergy { PCE_NO = 0, PCE_TOTAL = 1, PCE_COMPS = 2 }

  Define energy calculation enumeration.

- enum ePBEparm_calcForce { PCF_NO = 0, PCF_TOTAL = 1, PCF_COMPS = 2 }

  Define force calculation enumeration.

Functions

- VEXTERNC double PBEparm_getIonCharge (PBEparm ∗thee, int iion)

  Get charge (e) of specified ion species.

- VEXTERNC double PBEparm_getIonConc (PBEparm ∗thee, int iion)

  Get concentration (M) of specified ion species.

- VEXTERNC double PBEparm_getIonRadius (PBEparm ∗thee, int iion)

  Get radius (A) of specified ion species.

- VEXTERNC PBEparm ∗ PBEparm_ctor ()

  Construct PBEparm object.

- VEXTERNC int PBEparm_ctor2 (PBEparm ∗thee)

  FORTRAN stub to construct PBEparm object.

- VEXTERNC void PBEparm_dtor (PBEparm ∗∗thee)

  Object destructor.

- VEXTERNC void PBEparm_dtor2 (PBEparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC int PBEparm_check (PBEparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void PBEparm_copy (PBEparm ∗thee, PBEparm ∗parm)

  Copy PBEparm object into thee.

- VEXTERNC int PBEparm_parseToken (PBEparm ∗thee, char tok[VMAX_-BUFSIZE], Vio ∗sock)

  Parse a keyword from an input file.

8.8.1 Detailed Description

Parameter structure for PBE variables independent of solver.

8.8.2 Enumeration Type Documentation

8.8.2.1 enum ePBEparm_calcEnergy

Define energy calculation enumeration.

Enumerator:

PCE_NO	Do not perform energy calculation
PCE_TOTAL	Calculate total energy only
PCE_COMPS	Calculate per-atom energy components

8.8.2.2 enum ePBEparm_calcForce

Define force calculation enumeration.

Enumerator:

PCF_NO	Do not perform force calculation
PCF_TOTAL	Calculate total force only
PCF_COMPS	Calculate per-atom force components

8.8.3 Function Documentation

8.8.3.1 VEXTERNC int PBEparm_check (PBEparm ∗ *thee*)

Consistency check for parameter values stored in object.

Author

Nathan Baker

Returns

1 if OK, 0 otherwise

Parameters

thee Object to be checked

8.8.3.2 VEXTERNC void PBEparm_copy (PBEparm ∗ *thee*, PBEparm ∗ *parm*)

Copy PBEparm object into thee.

Author

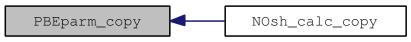
Nathan Baker

Parameters

thee Target for copy

parm Source for copy

Here is the caller graph for this function:



8.8.3.3 VEXTERNC PBEparm∗ PBEparm_ctor ()

Construct PBEparm object.

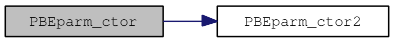
Author

Nathan Baker

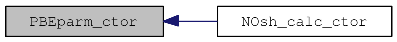
Returns

Newly allocated and initialized PBEparm object

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.4 VEXTERNC int PBEparm_ctor2 (PBEparm $*$ *thee*)

FORTRAN stub to construct PBEparm object.

Author

Nathan Baker

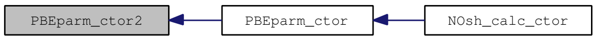
Returns

1 if succesful, 0 otherwise

Parameters

thee Memory location for object

Here is the caller graph for this function:



8.8.3.5 VEXTERNC void PBEparm_dtor (PBEparm $**$ *thee*)

Object destructor.

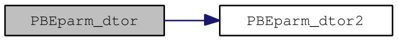
Author

Nathan Baker

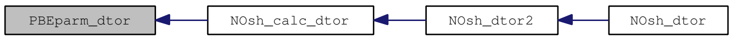
Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.6 VEXTERNC void PBEparm_dtor2 (PBEparm ∗ *thee*)

FORTRAN stub for object destructor.

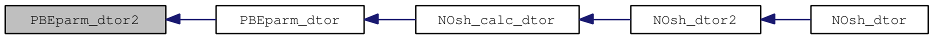
Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

Here is the caller graph for this function:



8.8.3.7 VEXTERNC double PBEparm_getIonCharge (PBEparm ∗ *thee*, int *iion*)

Get charge (e) of specified ion species.

Author

Nathan Baker

Returns

Charge of ion species (e)

Parameters

thee PBEparm object

iion Ion species ID/index

8.8.3.8 VEXTERNC double PBEparm_getIonConc (PBEparm ∗ *thee*, int *iion*)

Get concentration (M) of specified ion species.

Author

Nathan Baker

Returns

Concentration of ion species (M)

Parameters

thee PBEparm object

iion Ion species ID/index

8.8.3.9 VEXTERNC double PBEparm_getIonRadius (PBEparm ∗ *thee*, int *iion*)

Get radius (A) of specified ion species.

Author

Nathan Baker

Returns

Radius of ion species (A)

Parameters

thee PBEparm object

iion Ion species ID/index

8.8.3.10 VEXTERNC int PBEparm_parseToken (PBEparm ∗ *thee*, char *tok*[VMAX_BUFSIZE], Vio ∗ *sock*)

Parse a keyword from an input file.

Author

Nathan Baker

Returns

1 if matched and assigned; -1 if matched, but there's some sort of error (i.e., too few args); 0 if not matched

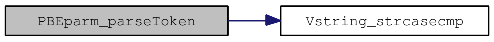
Parameters

thee Parsing object

tok Token to parse

sock Socket for additional tokens

Here is the call graph for this function:



8.9 Vacc class

Solvent- and ion-accessibility oracle.

Data Structures

- struct sVaccSurf

  Surface object list of per-atom surface points.

- struct sVacc

  Oracle for solvent- and ion-accessibility around a biomolecule.

Files

- file vacc.h

  Contains declarations for class Vacc.

- file vacc.c

  Class Vacc methods.

Typedefs

- typedef struct sVaccSurf VaccSurf

  Declaration of the VaccSurf class as the VaccSurf structure.

- typedef struct sVacc Vacc

  Declaration of the Vacc class as the Vacc structure.

Functions

- VEXTERNC unsigned long int Vacc_memChk (Vacc ∗thee)

  Get number of bytes in this object and its members.

- VEXTERNC VaccSurf ∗ VaccSurf_ctor (Vmem ∗mem, double probe_radius, int nsphere)

  Allocate and construct the surface object; do not assign surface points to positions.

- VEXTERNC int VaccSurf_ctor2 (VaccSurf ∗thee, Vmem ∗mem, double probe_-radius, int nsphere)

  Construct the surface object using previously allocated memory; do not assign surface points to positions.

- VEXTERNC void VaccSurf_dtor (VaccSurf ∗∗thee)

  Destroy the surface object and free its memory.

- VEXTERNC void VaccSurf_dtor2 (VaccSurf ∗thee)

  Destroy the surface object.

- VEXTERNC VaccSurf ∗ VaccSurf_refSphere (Vmem ∗mem, int npts)

  Set up an array of points for a reference sphere of unit radius.

- VEXTERNC VaccSurf ∗ Vacc_atomSurf (Vacc ∗thee, Vatom ∗atom, VaccSurf ∗ref, double probe_radius)

  Set up an array of points corresponding to the SAS due to a particular atom.

- VEXTERNC Vacc ∗ Vacc_ctor (Valist ∗alist, Vclist ∗clist, double surf_-density)

  Construct the accessibility object.

- VEXTERNC int Vacc_ctor2 (Vacc ∗thee, Valist ∗alist, Vclist ∗clist, double surf_density)

  FORTRAN stub to construct the accessibility object.

- VEXTERNC void Vacc_dtor (Vacc ∗∗thee)

  Destroy object.

- VEXTERNC void Vacc_dtor2 (Vacc ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC double Vacc_vdwAcc (Vacc ∗thee, double center[VAPBS_-DIM])

  Report van der Waals accessibility.

- VEXTERNC double Vacc_ivdwAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report inflated van der Waals accessibility.

- VEXTERNC double Vacc_molAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report molecular accessibility.

- VEXTERNC double Vacc_fastMolAcc (Vacc ∗thee, double center[VAPBS_-DIM], double radius)

  Report molecular accessibility quickly.

- VEXTERNC double Vacc_splineAcc (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad)

  Report spline-based accessibility.

- VEXTERNC void Vacc_splineAccGrad (Vacc ∗thee, double center[VAPBS_-DIM], double win, double infrad, double ∗grad)

  Report gradient of spline-based accessibility.

- VEXTERNC double Vacc_splineAccAtom (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom)

  Report spline-based accessibility for a given atom.

- VEXTERNC void Vacc_splineAccGradAtomUnnorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom (see Vpmg_splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by the accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm4 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 4th order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm3 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 3rd order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC double Vacc_SASA (Vacc ∗thee, double radius)

  Build the solvent accessible surface (SAS) and calculate the solvent accessible surface area.

- VEXTERNC double Vacc_totalSASA (Vacc ∗thee, double radius)

  Return the total solvent accessible surface area (SASA).

- VEXTERNC double Vacc_atomSASA (Vacc ∗thee, double radius, Vatom ∗atom)

  Return the atomic solvent accessible surface area (SASA).

- VEXTERNC VaccSurf ∗ Vacc_atomSASPoints (Vacc ∗thee, double radius, Vatom ∗atom)

  Get the set of points for this atom's solvent-accessible surface.

- VEXTERNC void Vacc_atomdSAV (Vacc ∗thee, double radius, Vatom ∗atom, double ∗dSA)

  Get the derivatve of solvent accessible volume.

- VEXTERNC void Vacc_atomdSASA (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA)

  Get the derivatve of solvent accessible area.

- VEXTERNC void Vacc_totalAtomdSASA (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA)

  Testing purposes only.

- VEXTERNC void Vacc_totalAtomdSAV (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA, Vclist ∗clist)

  Total solvent accessible volume.

- VEXTERNC double Vacc_totalSAV (Vacc ∗thee, Vclist ∗clist, APOLparm ∗apolparm, double radius)

  Return the total solvent accessible volume (SAV).

- VPUBLIC int Vacc_wcaEnergy (Vacc ∗thee, APOLparm ∗apolparm, Valist ∗alist, Vclist ∗clist)

  Return the WCA integral energy.

- VPUBLIC int Vacc_wcaForceAtom (Vacc ∗thee, APOLparm ∗apolparm, Vclist ∗clist, Vatom ∗atom, double ∗force)

  Return the WCA integral force.

8.9.1 Detailed Description

Solvent- and ion-accessibility oracle.

8.9.2 Function Documentation

8.9.2.1 VEXTERNC void Vacc_atomdSASA (Vacc ∗ *thee*, double *dpos*, double *radius*, Vatom ∗ *atom*, double ∗ *dSA*)

Get the derivatve of solvent accessible area.

Author

Jason Wagoner, David Gohara, Nathan Baker

Parameters

thee Acessibility object

dpos Atom position offset

radius Probe radius (Å)

atom Atom of interest

dSA Array holding answers of calc

8.9.2.2 VEXTERNC void Vacc_atomdSAV (Vacc ∗ *thee*, double *radius*, Vatom ∗ *atom*, double ∗ *dSA*)

Get the derivatve of solvent accessible volume.

Author

Jason Wagoner, Nathan Baker

Parameters

thee Acessibility object

radius Probe radius (Å)

atom Atom of interest

dSA Array holding answers of calc

8.9.2.3 VEXTERNC double Vacc_atomSASA (Vacc ∗ *thee*, double *radius*, Vatom ∗ *atom*)

Return the atomic solvent accessible surface area (SASA).

Note

Alias for Vacc_SASA

Author

Nathan Baker

Returns

Atomic solvent accessible area (A^2)

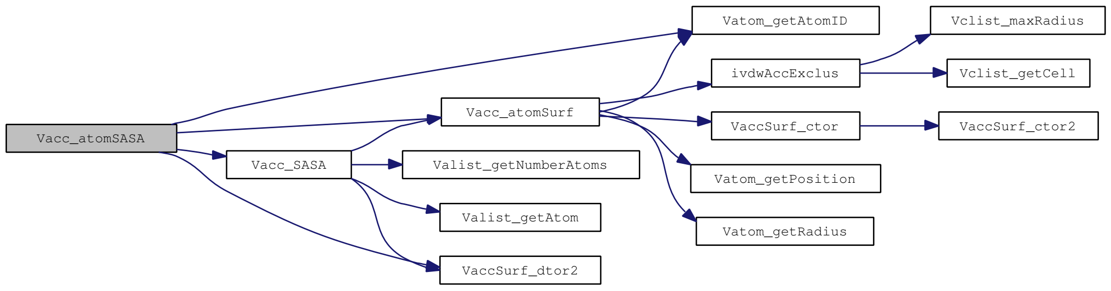
Parameters

thee Accessibility object

radius Probe molecule radius (Å)

atom Atom of interest

Here is the call graph for this function:



8.9.2.4 VEXTERNC VaccSurf∗ Vacc_atomSASPoints (Vacc ∗ *thee*, double *radius*, Vatom ∗ *atom*)

Get the set of points for this atom's solvent-accessible surface.

Author

Nathan Baker

Returns

Pointer to VaccSurf object for this atom

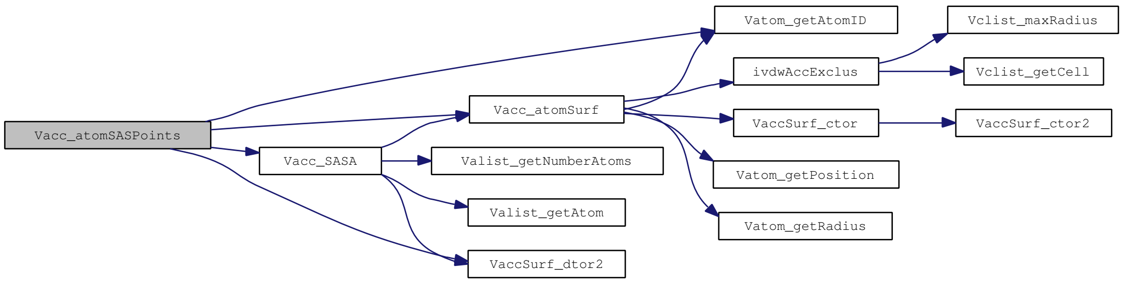
Parameters

thee Accessibility object

radius Probe molecule radius (Å)

atom Atom of interest

Here is the call graph for this function:



8.9.2.5 VEXTERNC VaccSurf∗ Vacc_atomSurf (Vacc ∗ *thee*, Vatom ∗ *atom*, VaccSurf ∗ *ref*, double *probe_radius*)

Set up an array of points corresponding to the SAS due to a particular atom.

Author

Nathan Baker

Returns

Atom sphere surface object

Parameters

thee Accessibility object for molecule

atom Atom for which the surface should be constructed

ref Reference sphere which sets the resolution for the surface.

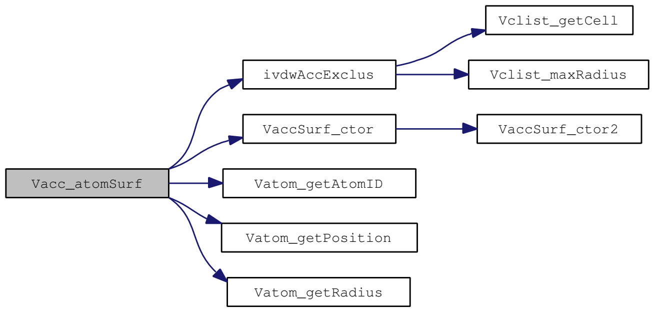
See also

VaccSurf_refSphere

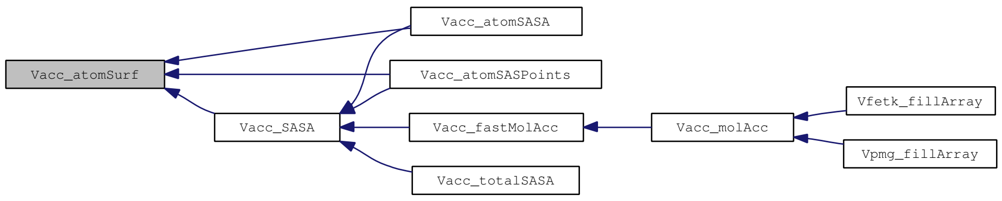
Parameters

probe_radius Probe radius (in A)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.6 VEXTERNC Vacc∗ Vacc_ctor (Valist ∗ *alist*, Vclist ∗ *clist*, double *surf_density*)

Construct the accessibility object.

Author

Nathan Baker

Returns

Newly allocated Vacc object

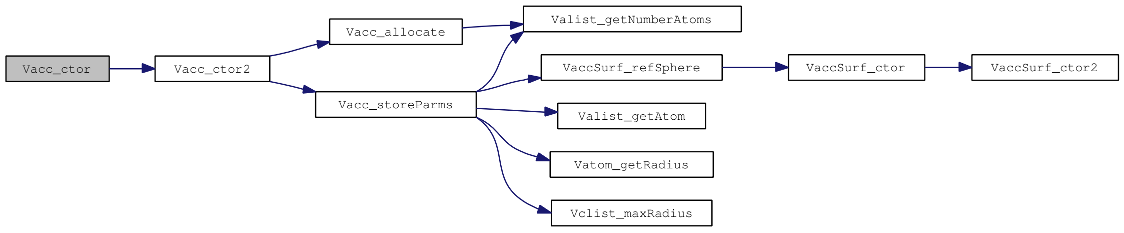
Parameters

alist Molecule for accessibility queries

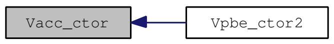
clist Pre-constructed cell list for looking up atoms near specific positions

surf_density Minimum per-atom solvent accessible surface point density (in pts/A^2)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.7 VEXTERNC int Vacc_ctor2 (Vacc ∗ *thee*, Valist ∗ *alist*, Vclist ∗ *clist*, double *surf_density*)

FORTRAN stub to construct the accessibility object.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Parameters

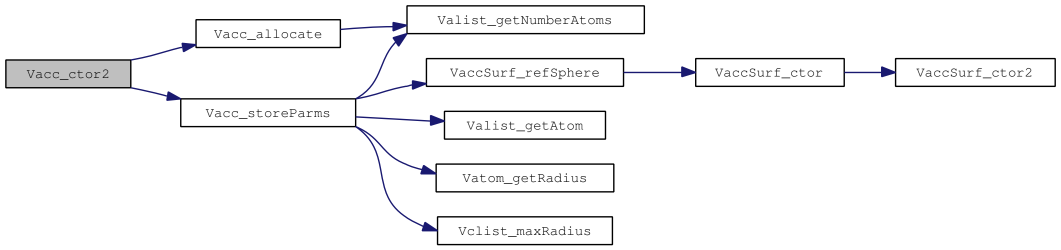
thee Memory for Vacc objet

alist Molecule for accessibility queries

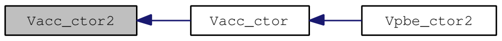
clist Pre-constructed cell list for looking up atoms near specific positions

surf_density Minimum per-atom solvent accessible surface point density (in pts/A^2)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.8 VEXTERNC void Vacc_dtor (Vacc ∗∗ *thee*)

Destroy object.

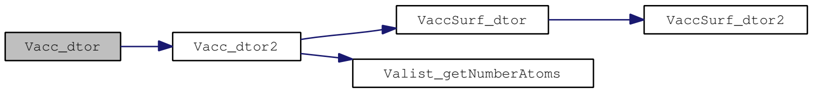
Author

Nathan Baker

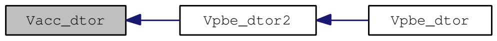
Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.9 VEXTERNC void Vacc_dtor2 (Vacc ∗ *thee*)

FORTRAN stub to destroy object.

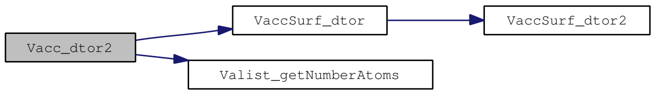
Author

Nathan Baker

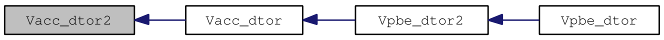
Parameters

thee Pointer to object

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.10 VEXTERNC double Vacc_fastMolAcc (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *radius*)

Report molecular accessibility quickly.

Given a point which is INSIDE the collection of inflated van der Waals spheres, but OUTSIDE the collection of non-inflated van der Waals spheres, determine accessibility of a probe (of radius radius) at a given point, given a collection of atomic spheres. Uses molecular (Connolly) surface definition.

Note

THIS ASSUMES YOU HAVE TESTED THAT THIS POINT IS DEFINITELY INSIDE THE INFLATED AND NON-INFLATED VAN DER WAALS SURFACES!

Author

Nathan Baker

Returns

Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Bug

This routine has a slight bug which can generate very small internal regions of high dielectric (thanks to John Mongan and Jess Swanson for finding this)

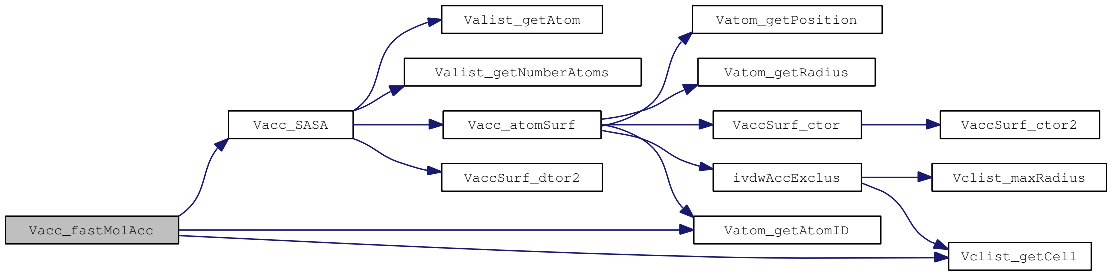
Parameters

thee Accessibility object

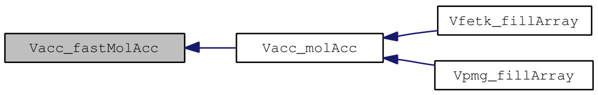
center Probe center coordinates

radius Probe radius (in Å)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.11 VEXTERNC double Vacc_ivdwAcc (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *radius*)

Report inflated van der Waals accessibility.

Determines if a point is within the union of the spheres centered at the atomic centers with radii equal to the sum of the atomic van der Waals radius and the probe radius.

Author

Nathan Baker

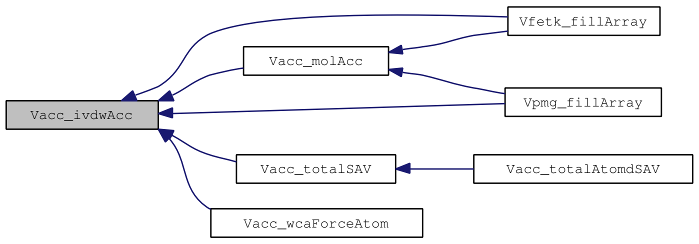
Returns

Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Parameters

thee Accessibility object
center Probe center coordinates
radius Probe radius (Å)

Here is the caller graph for this function:



8.9.2.12 VEXTERNC unsigned long int Vacc_memChk (Vacc $*$ *thee*)

Get number of bytes in this object and its members.

Author

Nathan Baker

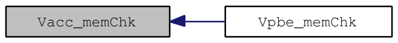
Returns

Number of bytes allocated for object

Parameters

thee Object for memory check

Here is the caller graph for this function:



8.9.2.13 VEXTERNC double Vacc_molAcc (Vacc $*$ *thee*, double *center*[VAPBS_DIM], double *radius*)

Report molecular accessibility.

Determine accessibility of a probe (of radius radius) at a given point, given a collection of atomic spheres. Uses molecular (Connolly) surface definition.

Author

Nathan Baker

Returns

Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Bug

This routine has a slight bug which can generate very small internal regions of high dielectric (thanks to John Mongan and Jess Swanson for finding this)

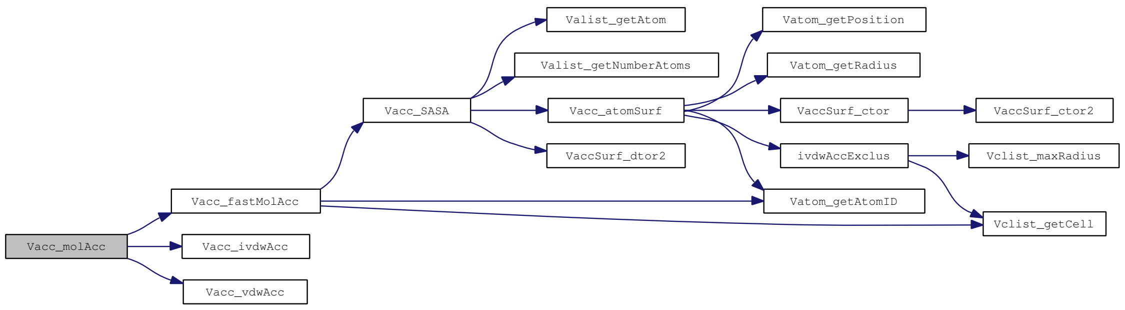
Parameters

thee Accessibility object

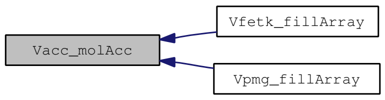
center Probe center coordinates

radius Probe radius (in Å)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.14 VEXTERNC double Vacc_SASA (Vacc $*$ *thee*, double *radius*)

Build the solvent accessible surface (SAS) and calculate the solvent accessible surface area.

Note

Similar to UHBD FORTRAN routine by Brock Luty (returns UHBD's asas2)

Author

Nathan Baker (original FORTRAN routine by Brock Luty)

Returns

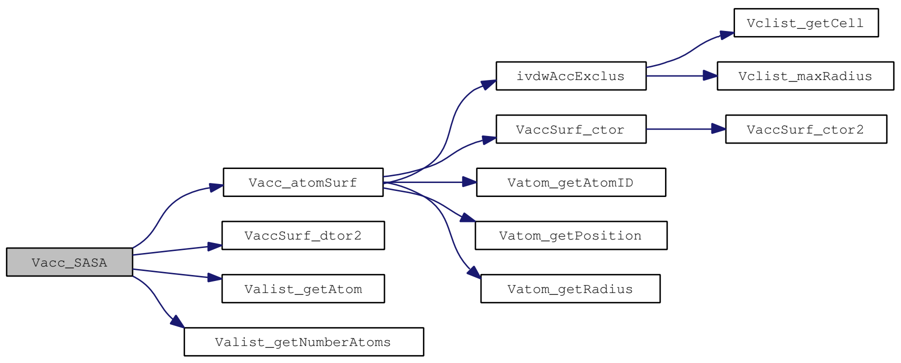
Total solvent accessible area (A^2)

Parameters

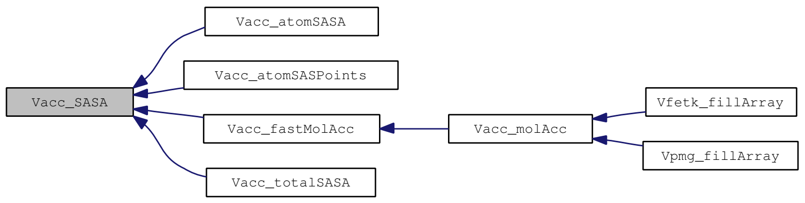
thee Accessibility object

radius Probe molecule radius (Å)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.15 VEXTERNC double Vacc_splineAcc (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*)

Report spline-based accessibility.

Determine accessibility at a given point, given a collection of atomic spheres. Uses Benoit Roux (Im et al, Comp Phys Comm, 111, 59--75, 1998) definition suitable for force evalation; basically a cubic spline.

Author

Nathan Baker

Returns

Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Parameters

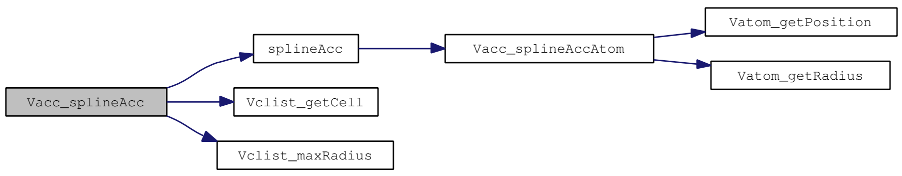
thee Accessibility object

center Probe center coordinates

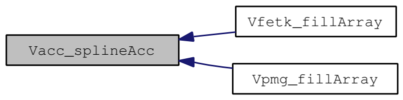
win Spline window (Å)

infrad Inflation radius (Å) for ion access.

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.16 VEXTERNC double Vacc_splineAccAtom (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, Vatom ∗ *atom*)

Report spline-based accessibility for a given atom.

Determine accessibility at a given point for a given atomic spheres. Uses Benoit Roux (Im et al, Comp Phys Comm, 111, 59--75, 1998) definition suitable for force evalation; basically a cubic spline.

Author

Nathan Baker

Returns

Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Parameters

thee Accessibility object

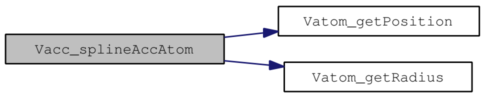
center Probe center coordinates

win Spline window (Å)

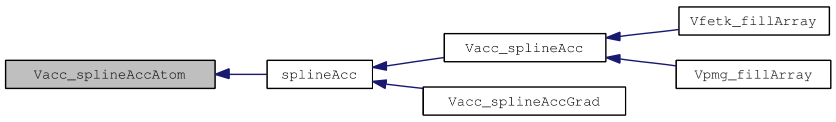
infrad Inflation radius (Å) for ion access.

atom Atom

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.17 VEXTERNC void Vacc_splineAccGrad (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, double ∗ *grad*)

Report gradient of spline-based accessibility.

Author

Nathan Baker

Parameters

thee Accessibility object

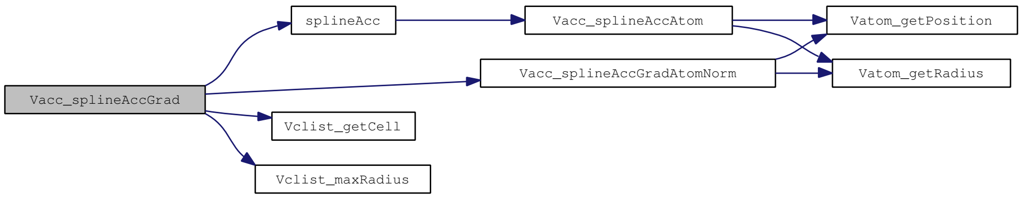
center Probe center coordinates

win Spline window (Å)

infrad Inflation radius (Å) for ion access.

grad 3-vector set to gradient of accessibility

Here is the call graph for this function:



8.9.2.18 VEXTERNC void Vacc_splineAccGradAtomNorm (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, Vatom ∗ *atom*, double ∗ *force*)

Report gradient of spline-based accessibility with respect to a particular atom normalized by the accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

Determine accessibility at a given point, given a collection of atomic spheres. Uses Benoit Roux (Im et al, Comp Phys Comm, 111, 59--75, 1998) definition suitable for force evalation; basically a cubic spline.

Author

Nathan Baker

Parameters

thee Accessibility object

center Probe center coordinates

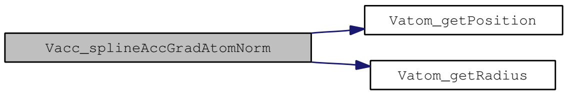
win Spline window (Å)

infrad Inflation radius (Å) for ion access.

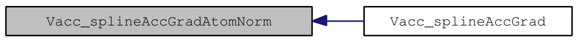
atom Atom

force VAPBS_DIM-vector set to gradient of accessibility

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.19 VEXTERNC void Vacc_splineAccGradAtomNorm3 (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, Vatom ∗ *atom*, double ∗ *force*)

Report gradient of spline-based accessibility with respect to a particular atom normalized by a 3rd order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

Author

Michael Schnieders

Parameters

thee Accessibility object

center Probe center coordinates

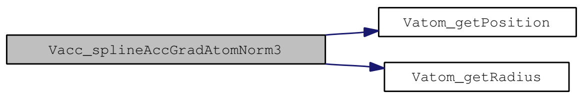
win Spline window (Å)

infrad Inflation radius (Å) for ion access.

atom Atom

force VAPBS_DIM-vector set to gradient of accessibility

Here is the call graph for this function:



8.9.2.20 VEXTERNC void Vacc_splineAccGradAtomNorm4 (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, Vatom ∗ *atom*, double ∗ *force*)

Report gradient of spline-based accessibility with respect to a particular atom normalized by a 4th order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

Author

Michael Schnieders

Parameters

thee Accessibility object

center Probe center coordinates

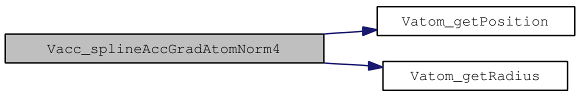
win Spline window (Å)

infrad Inflation radius (Å) for ion access.

atom Atom

force VAPBS_DIM-vector set to gradient of accessibility

Here is the call graph for this function:



8.9.2.21 VEXTERNC void Vacc_splineAccGradAtomUnnorm (Vacc ∗ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, Vatom ∗ *atom*, double ∗ *force*)

Report gradient of spline-based accessibility with respect to a particular atom (see Vpmg_splineAccAtom).

Determine accessibility at a given point, given a collection of atomic spheres. Uses Benoit Roux (Im et al, Comp Phys Comm, 111, 59--75, 1998) definition suitable for force evalation; basically a cubic spline.

Author

Nathan Baker

Parameters

thee Accessibility object

center Probe center coordinates

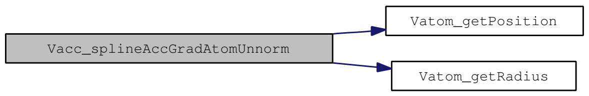
win Spline window (Å)

infrad Inflation radius (Å) for ion access.

atom Atom

force VAPBS_DIM-vector set to gradient of accessibility

Here is the call graph for this function:



8.9.2.22 VEXTERNC void Vacc_totalAtomdSASA (Vacc ∗ *thee*, double *dpos*, double *radius*, Vatom ∗ *atom*, double ∗ *dSA*)

Testing purposes only.

Author

David Gohara, Nathan Baker

Parameters

thee Acessibility object

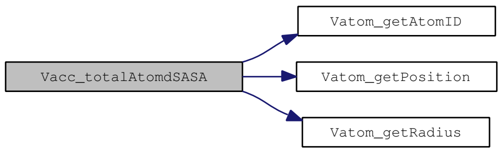
dpos Atom position offset

radius Probe radius (Å)

atom Atom of interest

dSA Array holding answers of calc

Here is the call graph for this function:



8.9.2.23 VEXTERNC void Vacc_totalAtomdSAV (Vacc ∗ *thee*, double *dpos*, double *radius*, Vatom ∗ *atom*, double ∗ *dSA*, Vclist ∗ *clist*)

Total solvent accessible volume.

Author

David Gohara, Nathan Baker

Parameters

thee Acessibility object

dpos Atom position offset

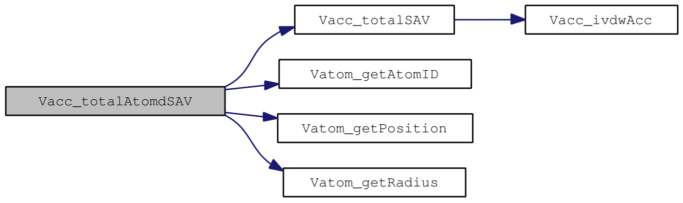
radius Probe radius (Å)

atom Atom of interest

dSA Array holding answers of calc

clist clist for this calculation

Here is the call graph for this function:



8.9.2.24 VEXTERNC double Vacc_totalSASA (Vacc ∗ *thee*, double *radius*)

Return the total solvent accessible surface area (SASA).

Note

Alias for Vacc_SASA

Author

Nathan Baker

Returns

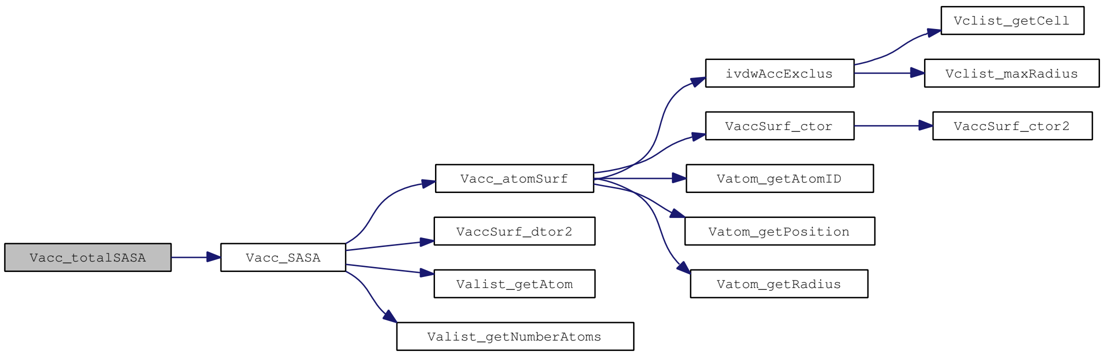
Total solvent accessible area (A^2)

Parameters

thee Accessibility object

radius Probe molecule radius (Å)

Here is the call graph for this function:



8.9.2.25 VEXTERNC double Vacc_totalSAV (Vacc ∗ *thee*, Vclist ∗ *clist*, APOLparm ∗ *apolparm*, double *radius*)

Return the total solvent accessible volume (SAV).

Note

Alias for Vacc_SAV

Author

David Gohara

Returns

Total solvent accessible volume (A^3)

Parameters

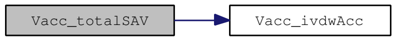
thee Accessibility object

clist Clist for acc object

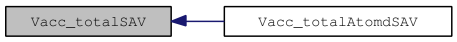
apolparm Apolar parameters -- could be VNULL if none required for this calculation. If VNULL, then default settings are used

radius Probe molecule radius (Å)

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.26 VEXTERNC double Vacc_vdwAcc (Vacc ∗ *thee*, double *center*[VAPBS_DIM])

Report van der Waals accessibility.

Determines if a point is within the union of the atomic spheres (with radii equal to their van der Waals radii).

Author

Nathan Baker

Returns

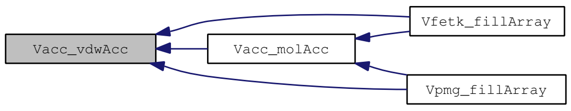
Characteristic function value between 1.0 (accessible) and 0.0 (inaccessible)

Parameters

thee Accessibility object

center Probe center coordinates

Here is the caller graph for this function:



8.9.2.27 VPUBLIC int Vacc_wcaEnergy (Vacc ∗ *thee*, APOLparm ∗ *apolparm*, Valist ∗ *alist*, Vclist ∗ *clist*)

Return the WCA integral energy.

Author

David Gohara

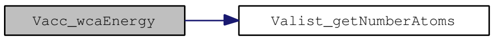
Returns

WCA energy (kJ/mol)

Parameters

thee Accessibility object
apolparm Apolar calculation parameters
alist Alist for acc object
clist Clist for acc object

Here is the call graph for this function:



8.9.2.28 VPUBLIC int Vacc_wcaForceAtom (Vacc ∗ *thee*, APOLparm ∗ *apolparm*, Vclist ∗ *clist*, Vatom ∗ *atom*, double ∗ *force*)

Return the WCA integral force.

Author

David Gohara

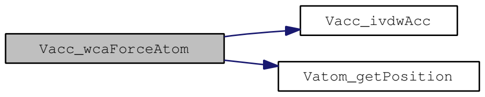
Returns

WCA energy (kJ/mol/A)

Parameters

thee Accessibility object
apolparm Apolar calculation parameters
clist Clist for acc object
atom Current atom
force Force for atom

Here is the call graph for this function:



8.9.2.29 VEXTERNC VaccSurf∗ VaccSurf_ctor (Vmem ∗ *mem*, double *probe_radius*, int *nsphere*)

Allocate and construct the surface object; do not assign surface points to positions.

Author

Nathan Baker

Returns

Newly allocated and constructed surface object

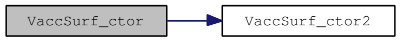
Parameters

mem Memory manager (can be VNULL)

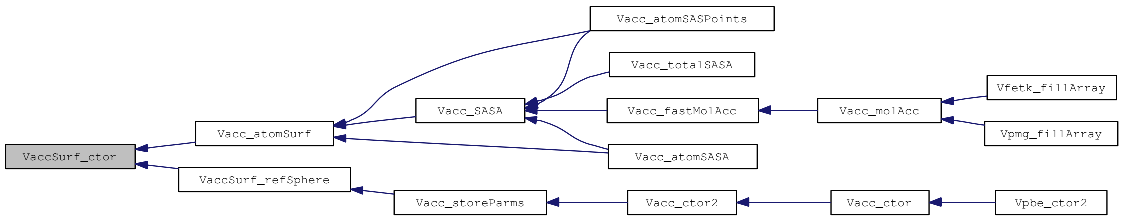
probe_radius Probe radius (in A) for this surface

nsphere Number of points in sphere

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.30 VEXTERNC int VaccSurf_ctor2 (VaccSurf ∗ *thee*, Vmem ∗ *mem*, double *probe_radius*, int *nsphere*)

Construct the surface object using previously allocated memory; do not assign surface points to positions.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Parameters

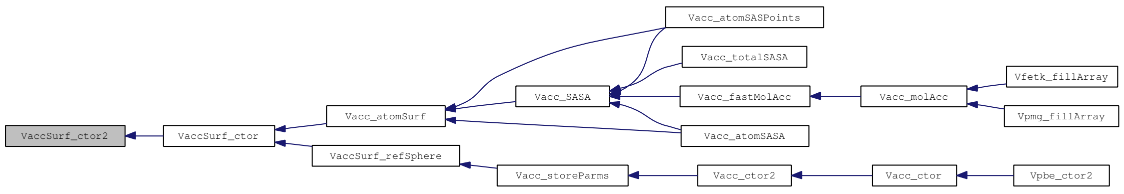
thee Allocated memory

mem Memory manager (can be VNULL)

probe_radius Probe radius (in A) for this surface

nsphere Number of points in sphere

Here is the caller graph for this function:



8.9.2.31 VEXTERNC void VaccSurf_dtor (VaccSurf ** *thee*)

Destroy the surface object and free its memory.

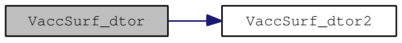
Author

Nathan Baker

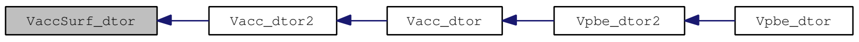
Parameters

thee Object to be destroyed

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.2.32 VEXTERNC void VaccSurf_dtor2 (VaccSurf ∗ *thee*)

Destroy the surface object.

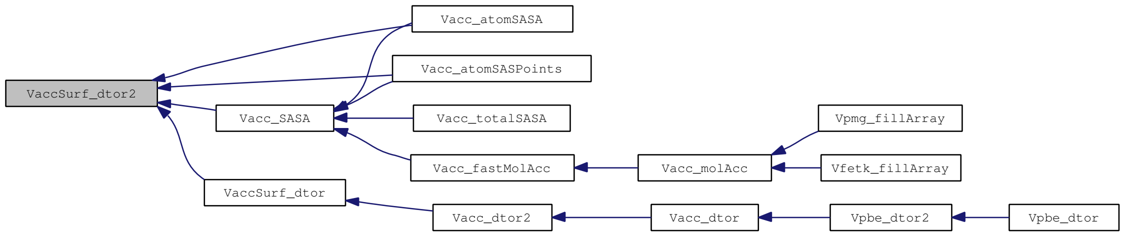
Author

Nathan Baker

Parameters

thee Object to be destroyed

Here is the caller graph for this function:



8.9.2.33 VEXTERNC VaccSurf∗ VaccSurf_refSphere (Vmem ∗ *mem*, int *npts*)

Set up an array of points for a reference sphere of unit radius.

Generates approximately npts # of points (actual number stored in thee->npts) somewhat uniformly distributed across a sphere of unit radius centered at the origin.

Note

This routine was shamelessly ripped off from sphere.f from UHBD as developed by Michael K. Gilson.

Author

Nathan Baker (original FORTRAN code by Mike Gilson)

Returns

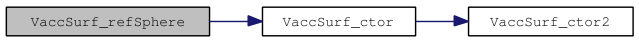
Reference sphere surface object

Parameters

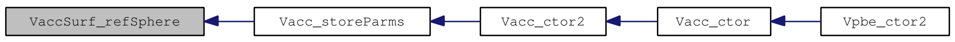
mem Memory object

npts Requested number of points on sphere

Here is the call graph for this function:



Here is the caller graph for this function:



8.10 Valist class

Container class for list of atom objects.

Data Structures

- struct sValist

    Container class for list of atom objects.

Files

- file valist.h

    Contains declarations for class Valist.

Typedefs

- typedef struct sValist Valist

    Declaration of the Valist class as the Valist structure.

Functions

- VEXTERNC Vatom ∗ Valist_getAtomList (Valist ∗thee)

    Get actual array of atom objects from the list.

- VEXTERNC double Valist_getCenterX (Valist ∗thee)

    Get x-coordinate of molecule center.

- VEXTERNC double Valist_getCenterY (Valist ∗thee)

    Get y-coordinate of molecule center.

- VEXTERNC double Valist_getCenterZ (Valist ∗thee)

    Get z-coordinate of molecule center.

- VEXTERNC int Valist_getNumberAtoms (Valist ∗thee)

    Get number of atoms in the list.

- VEXTERNC Vatom ∗ Valist_getAtom (Valist ∗thee, int i)

    Get pointer to particular atom in list.

- VEXTERNC unsigned long int Valist_memChk (Valist ∗thee)

  Get total memory allocated for this object and its members.

- VEXTERNC Valist ∗ Valist_ctor ()

  Construct the atom list object.

- VEXTERNC Vrc_Codes Valist_ctor2 (Valist ∗thee)

  FORTRAN stub to construct the atom list object.

- VEXTERNC void Valist_dtor (Valist ∗∗thee)

  Destroys atom list object.

- VEXTERNC void Valist_dtor2 (Valist ∗thee)

  FORTRAN stub to destroy atom list object.

- VEXTERNC Vrc_Codes Valist_readPQR (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from a PQR file.

- VEXTERNC Vrc_Codes Valist_readPDB (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from a PDB file.

- VEXTERNC Vrc_Codes Valist_readXML (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from an XML file.

- VEXTERNC Vrc_Codes Valist_getStatistics (Valist ∗thee)

  Load up Valist with various statistics.

8.10.1 Detailed Description

Container class for list of atom objects.

8.10.2 Function Documentation

8.10.2.1 VEXTERNC Valist∗ Valist_ctor ()

Construct the atom list object.

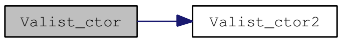
Author

Nathan Baker

Returns

Pointer to newly allocated (empty) atom list

Here is the call graph for this function:



8.10.2.2 VEXTERNC Vrc_Codes Valist_ctor2 (Valist $*$ *thee*)

FORTRAN stub to construct the atom list object.

Author

Nathan Baker, Yong Huang

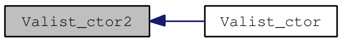
Returns

Success enumeration

Parameters

thee Storage for new atom list

Here is the caller graph for this function:



8.10.2.3 VEXTERNC void Valist_dtor (Valist $**$ *thee*)

Destroys atom list object.

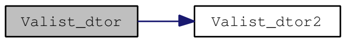
Author

Nathan Baker

Parameters

thee Pointer to storage for atom list

Here is the call graph for this function:



8.10.2.4 VEXTERNC void Valist_dtor2 (Valist ∗ *thee*)

FORTRAN stub to destroy atom list object.

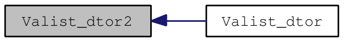
Author

Nathan Baker

Parameters

thee Pointer to atom list object

Here is the caller graph for this function:



8.10.2.5 VEXTERNC Vatom∗ Valist_getAtom (Valist ∗ *thee*, int *i*)

Get pointer to particular atom in list.

Author

Nathan Baker

Returns

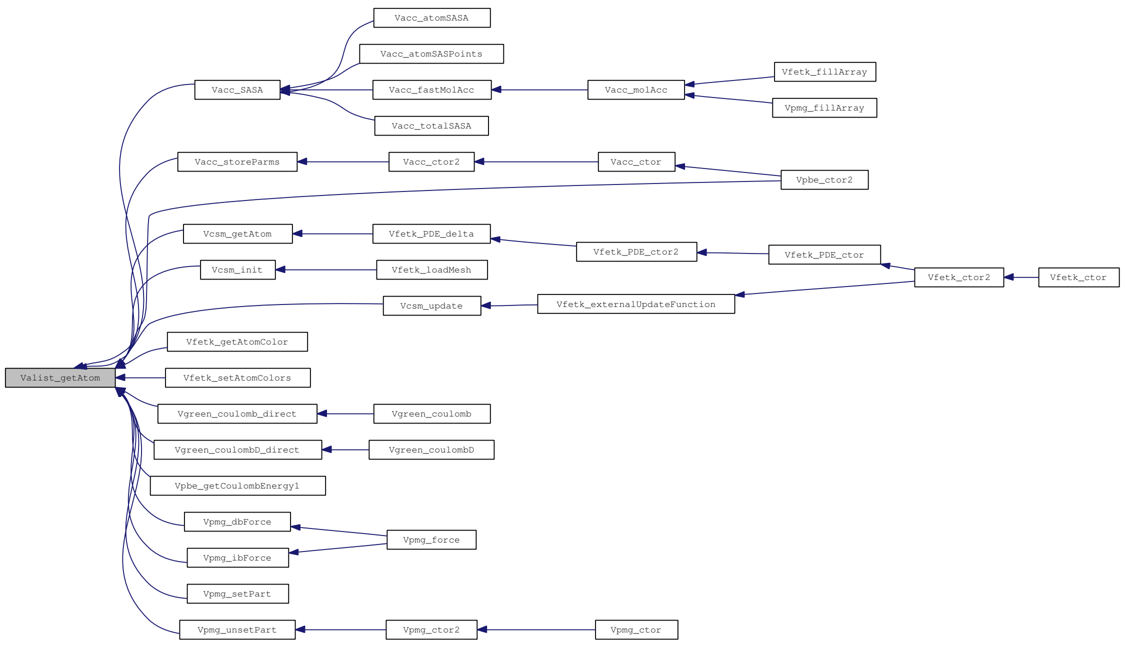
Pointer to atom object i

Parameters

thee Atom list object

i Index of atom in list

Here is the caller graph for this function:



8.10.2.6 VEXTERNC Vatom∗ Valist_getAtomList (Valist ∗ *thee*)

Get actual array of atom objects from the list.

Author

Nathan Baker

Returns

Array of atom objects

Parameters

thee Atom list object

8.10.2.7 VEXTERNC double Valist_getCenterX (Valist ∗ *thee*)

Get x-coordinate of molecule center.

Author

Nathan Baker

Returns

X-coordinate of molecule center

Parameters

thee Atom list object

8.10.2.8 VEXTERNC double Valist_getCenterY (Valist ∗ *thee*)

Get y-coordinate of molecule center.

Author

Nathan Baker

Returns

Y-coordinate of molecule center

Parameters

thee Atom list object

8.10.2.9 VEXTERNC double Valist_getCenterZ (Valist ∗ *thee*)

Get z-coordinate of molecule center.

Author

Nathan Baker

Returns

Z-coordinate of molecule center

Parameters

thee Atom list object

8.10.2.10 VEXTERNC int Valist_getNumberAtoms (Valist ∗ *thee*)

Get number of atoms in the list.

Author

Nathan Baker

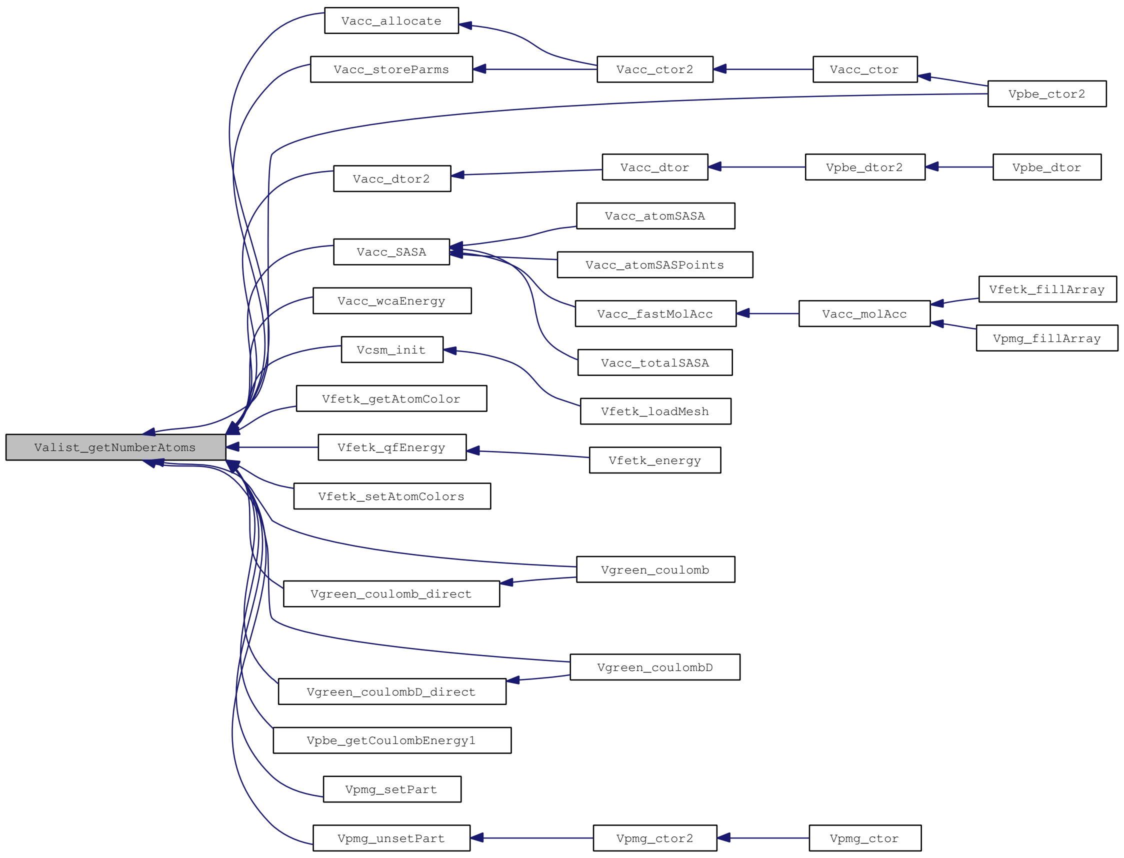
Returns

Number of atoms in list

Parameters

thee Atom list object

Here is the caller graph for this function:



8.10.2.11 VEXTERNC Vrc_Codes Valist_getStatistics (Valist ∗ *thee*)

Load up Valist with various statistics.

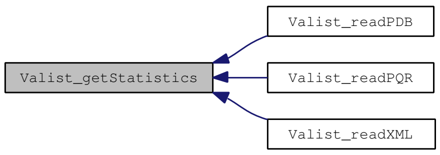
Author

Nathan Baker, Yong Huang

Returns

Success enumeration

Here is the caller graph for this function:



8.10.2.12 VEXTERNC unsigned long int Valist_memChk (Valist ∗ *thee*)

Get total memory allocated for this object and its members.

Author

Nathan Baker

Returns

Total memory in bytes

Parameters

thee Atom list object

8.10.2.13 VEXTERNC Vrc_Codes Valist_readPDB (Valist ∗ *thee*, Vparam ∗ *param*, Vio ∗ *sock*)

Fill atom list with information from a PDB file.

Author

Nathan Baker, Todd Dolinsky, Yong Huang

Returns

Success enumeration

Note

We don't actually respect PDB format; instead recognize whitespace- or tab-delimited fields which allows us to deal with structures with coordinates > 999 or < -999.

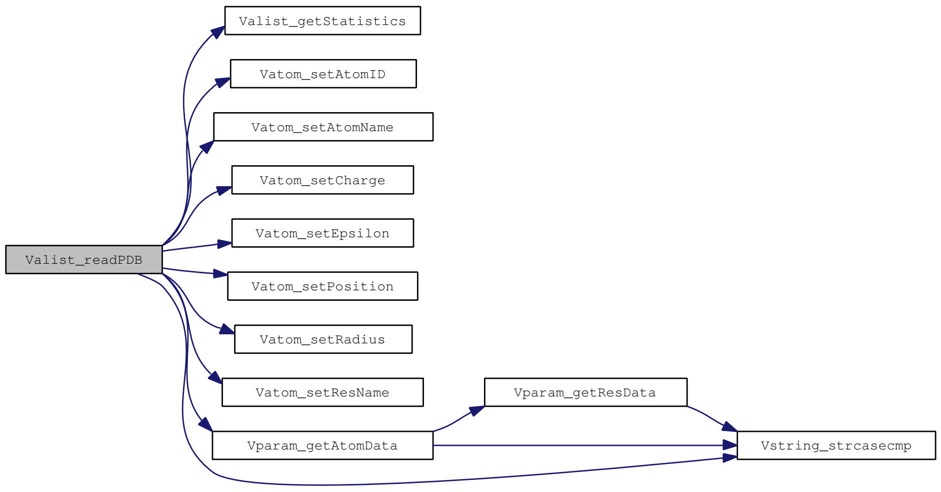
Parameters

thee Atom list object

param A pre-initialized parameter object

sock Socket read for reading PDB file

Here is the call graph for this function:



8.10.2.14 VEXTERNC Vrc_Codes Valist_readPQR (Valist $*$ *thee*, Vparam $*$ *param*, Vio $*$ *sock*)

Fill atom list with information from a PQR file.

Author

Nathan Baker, Yong Huang

Returns

Success enumeration

Note

- A PQR file has PDB structure with charge and radius in the last two columns instead of weight and occupancy
- We don't actually respect PDB format; instead recognize whitespace- or tab-delimited fields which allows us to deal with structures with coordinates > 999 or < -999.

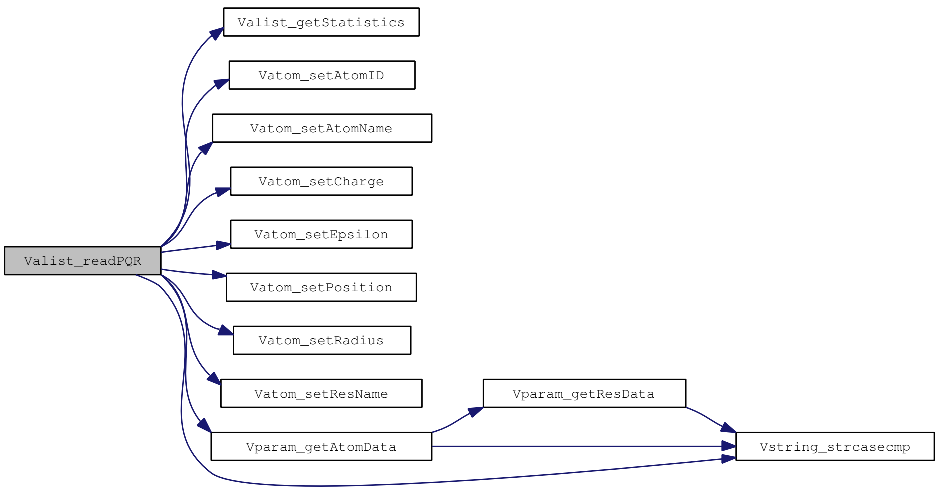
Parameters

thee Atom list object

param A pre-initialized parameter object

sock Socket reading for reading PQR file

Here is the call graph for this function:



8.10.2.15 VEXTERNC Vrc_Codes Valist_readXML (Valist $*$ *thee*, Vparam $*$ *param*, Vio $*$ *sock*)

Fill atom list with information from an XML file.

Author

Todd Dolinsky, Yong Huang

Returns

Success enumeration

Note

- The XML file must adhere to some guidelines, notably the presence of an <atom> tag with all other useful information (x, y, z, charge, and radius) as nested elements.

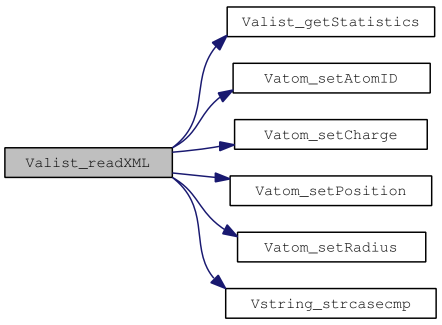
Parameters

thee Atom list object

param A pre-initialized parameter object

sock Socket reading for reading PQR file

Here is the call graph for this function:



8.11 Vatom class

Atom class for interfacing APBS with PDB files.

Data Structures

- struct sVatom

  Contains public data members for Vatom class/module.

Files

- file vatom.h

  Contains declarations for class Vatom.

- file vatom.c

  Class Vatom methods.

Defines

- #define VMAX_RECLEN 64

  Residue name length.

Typedefs

- typedef struct sVatom Vatom

  Declaration of the Vatom class as the Vatom structure.

Functions

- VEXTERNC double $*$ Vatom_getPosition (Vatom $*$thee)

  Get atomic position.

- VEXTERNC void Vatom_setRadius (Vatom $*$thee, double radius)

  Set atomic radius.

- VEXTERNC double Vatom_getRadius (Vatom $*$thee)

Get atomic position.

- VEXTERNC void Vatom_setPartID (Vatom ∗thee, int partID)

  Set partition ID.

- VEXTERNC double Vatom_getPartID (Vatom ∗thee)

  Get partition ID.

- VEXTERNC void Vatom_setAtomID (Vatom ∗thee, int id)

  Set atom ID.

- VEXTERNC double Vatom_getAtomID (Vatom ∗thee)

  Get atom ID.

- VEXTERNC void Vatom_setCharge (Vatom ∗thee, double charge)

  Set atomic charge.

- VEXTERNC double Vatom_getCharge (Vatom ∗thee)

  Get atomic charge.

- VEXTERNC void Vatom_setEpsilon (Vatom ∗thee, double epsilon)

  Set atomic epsilon.

- VEXTERNC double Vatom_getEpsilon (Vatom ∗thee)

  Get atomic epsilon.

- VEXTERNC unsigned long int Vatom_memChk (Vatom ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC void Vatom_setResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Set residue name.

- VEXTERNC void Vatom_setAtomName (Vatom ∗thee, char atomName[VMAX_RECLEN])

  Set atom name.

- VEXTERNC void Vatom_getResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Retrieve residue name.

- VEXTERNC void Vatom_getAtomName (Vatom ∗thee, char atomName[VMAX_RECLEN])

Retrieve atom name.

- VEXTERNC Vatom ∗ Vatom_ctor ()

  Constructor for the Vatom class.

- VEXTERNC int Vatom_ctor2 (Vatom ∗thee)

  FORTRAN stub constructor for the Vatom class.

- VEXTERNC void Vatom_dtor (Vatom ∗∗thee)

  Object destructor.

- VEXTERNC void Vatom_dtor2 (Vatom ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC void Vatom_setPosition (Vatom ∗thee, double position[3])

  Set the atomic position.

- VEXTERNC void Vatom_copyTo (Vatom ∗thee, Vatom ∗dest)

  Copy information to another atom.

- VEXTERNC void Vatom_copyFrom (Vatom ∗thee, Vatom ∗src)

  Copy information to another atom.

8.11.1 Detailed Description

Atom class for interfacing APBS with PDB files.

8.11.2 Define Documentation

8.11.2.1 #define VMAX_RECLEN 64

Residue name length.

Author

Nathan Baker, David Gohara, Mike Schneiders

8.11.3 Function Documentation

8.11.3.1 VEXTERNC void Vatom_copyFrom (Vatom ∗ *thee*, Vatom ∗ *src*)

Copy information to another atom.

Author

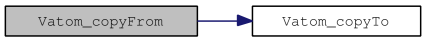
Nathan Baker

Parameters

thee Destination for atom information

src Source for atom information

Here is the call graph for this function:



8.11.3.2 VEXTERNC void Vatom_copyTo (Vatom $*$ *thee*, Vatom $*$ *dest*)

Copy information to another atom.

Author

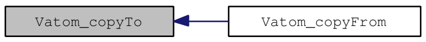
Nathan Baker

Parameters

thee Source for atom information

dest Destination for atom information

Here is the caller graph for this function:



8.11.3.3 VEXTERNC Vatom$*$ Vatom_ctor ()

Constructor for the Vatom class.

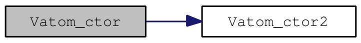
Author

Nathan Baker

Returns

Pointer to newly allocated Vatom object

Here is the call graph for this function:



8.11.3.4 VEXTERNC int Vatom_ctor2 (Vatom $*$ *thee*)

FORTRAN stub constructor for the Vatom class.

Author

Nathan Baker

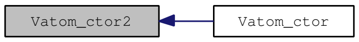
Parameters

thee Pointer to Vatom allocated memory location

Returns

1 if succesful, 0 otherwise

Here is the caller graph for this function:



8.11.3.5 VEXTERNC void Vatom_dtor (Vatom $**$ *thee*)

Object destructor.

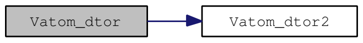
Author

Nathan Baker

Parameters

thee Pointer to memory location of object to be destroyed

Here is the call graph for this function:



8.11.3.6 VEXTERNC void Vatom_dtor2 (Vatom ∗ *thee*)

FORTRAN stub object destructor.

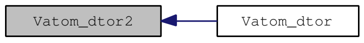
Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

Here is the caller graph for this function:



8.11.3.7 VEXTERNC double Vatom_getAtomID (Vatom ∗ *thee*)

Get atom ID.

Author

Nathan Baker

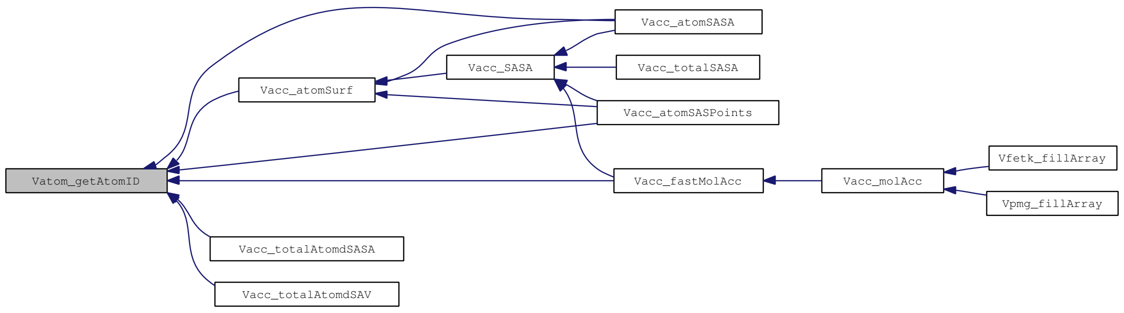
Parameters

thee Vatom object

Returns

Unique non-negative number

Here is the caller graph for this function:



8.11.3.8 VEXTERNC void Vatom_getAtomName (Vatom ∗ *thee*, char *atomName*[VMAX_RECLEN])

Retrieve atom name.

Author

Jason Wagoner

Parameters

thee Vatom object

atomName Atom name

8.11.3.9 VEXTERNC double Vatom_getCharge (Vatom ∗ *thee*)

Get atomic charge.

Author

Nathan Baker

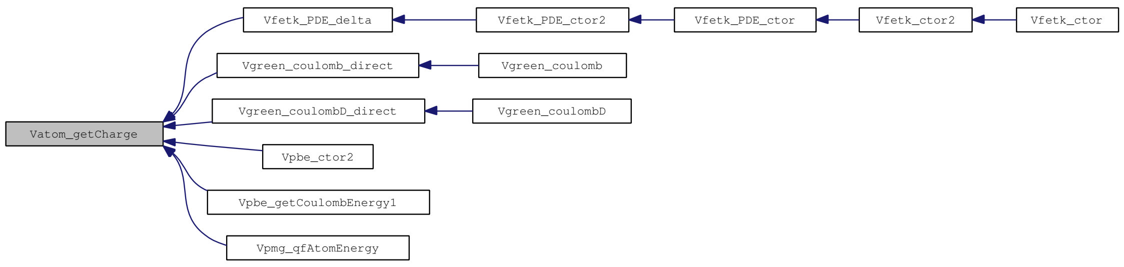
Parameters

thee Vatom object

Returns

Atom partial charge (in e)

Here is the caller graph for this function:



8.11.3.10 VEXTERNC double Vatom_getEpsilon (Vatom $*$ *thee*)

Get atomic epsilon.

Author

David Gohara

Parameters

thee Vatom object

Returns

Atomic epsilon (in Å)

8.11.3.11 VEXTERNC double Vatom_getPartID (Vatom $*$ *thee*)

Get partition ID.

Author

Nathan Baker

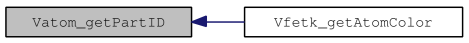
Parameters

thee Vatom object

Returns

Partition ID; a negative value means this atom is not assigned to any partition

Here is the caller graph for this function:



8.11.3.12 VEXTERNC double$*$ Vatom_getPosition (Vatom $*$ *thee*)

Get atomic position.

Author

Nathan Baker

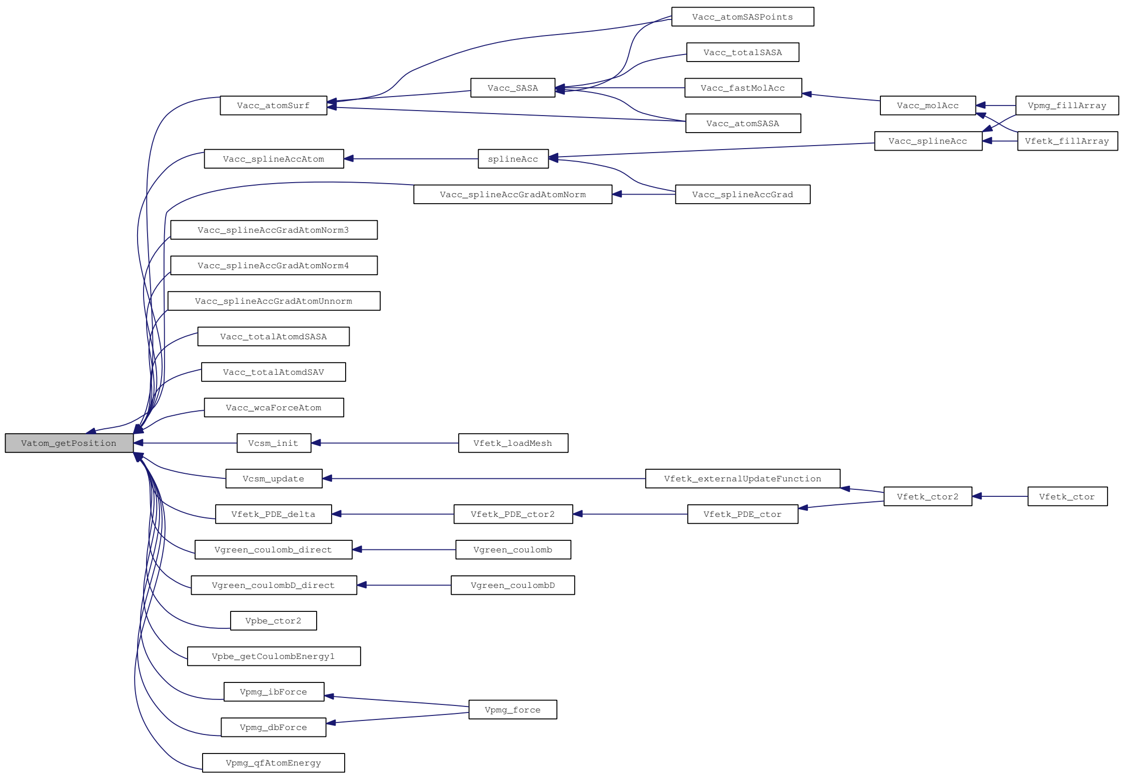
Parameters

thee Vatom object

Returns

Pointer to 3∗double array of atomic coordinates (in Å)

Here is the caller graph for this function:



8.11.3.13 VEXTERNC double Vatom_getRadius (Vatom ∗ *thee*)

Get atomic position.

Author

Nathan Baker

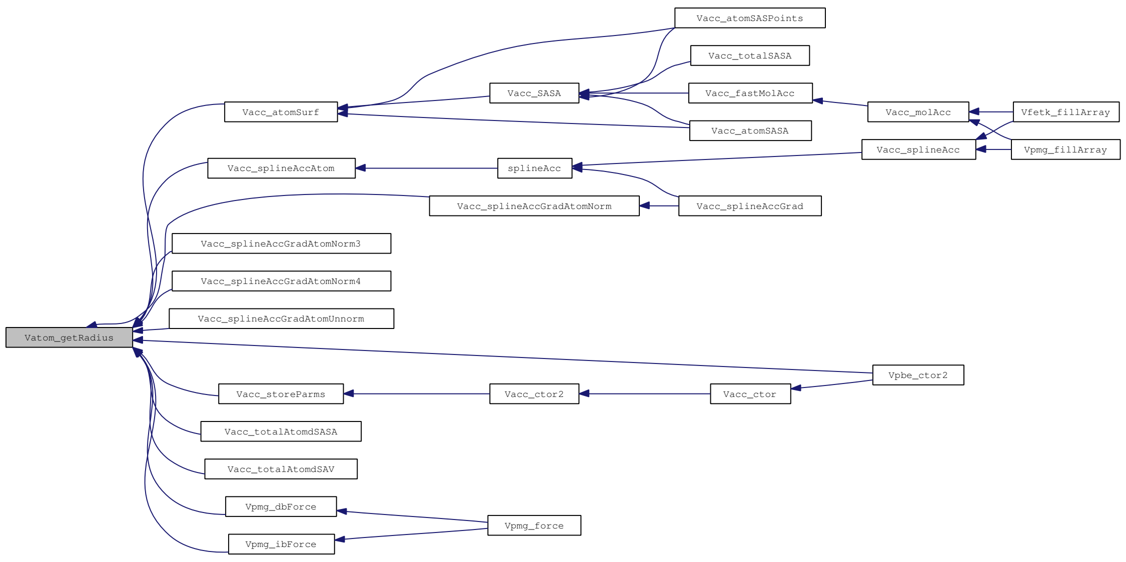
Parameters

thee Vatom object

Returns

Atomic radius (in Å)

Here is the caller graph for this function:



8.11.3.14 VEXTERNC void Vatom_getResName (Vatom ∗ *thee*, char *resName*[VMAX_RECLEN])

Retrieve residue name.

Author

Jason Wagoner

Parameters

thee Vatom object

resName Residue Name

8.11.3.15 VEXTERNC unsigned long int Vatom_memChk (Vatom ∗ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

Parameters

thee Vpmg object

Returns

The memory used by this structure and its contents in bytes

8.11.3.16 VEXTERNC void Vatom_setAtomID (Vatom ∗ *thee*, int *id*)

Set atom ID.

Author

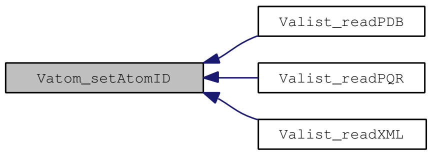
Nathan Baker

Parameters

thee Vatom object

id Unique non-negative number

Here is the caller graph for this function:



8.11.3.17 VEXTERNC void Vatom_setAtomName (Vatom ∗ *thee*, char *atomName*[VMAX_RECLEN])

Set atom name.

Author

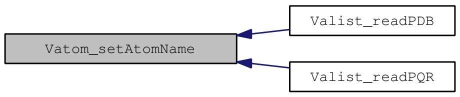
Jason Wagoner

Parameters

thee Vatom object

atomName Atom name

Here is the caller graph for this function:



8.11.3.18 VEXTERNC void Vatom_setCharge (Vatom ∗ *thee*, double *charge*)

Set atomic charge.

Author

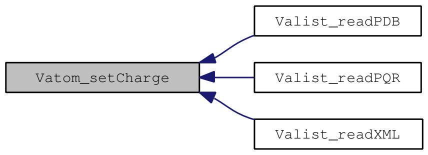
Nathan Baker

Parameters

thee Vatom object

charge Atom partial charge (in e)

Here is the caller graph for this function:



8.11.3.19 VEXTERNC void Vatom_setEpsilon (Vatom ∗ *thee*, double *epsilon*)

Set atomic epsilon.

Author

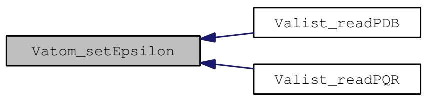
David Gohara

Parameters

thee Vatom object

epsilon Atomic epsilon (in Å)

Here is the caller graph for this function:



8.11.3.20 VEXTERNC void Vatom_setPartID (Vatom ∗ *thee*, int *partID*)

Set partition ID.

Author

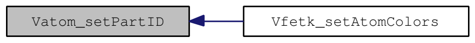
Nathan Baker

Parameters

thee Vatom object

partID Partition ID; a negative value means this atom is not assigned to any partition

Here is the caller graph for this function:



8.11.3.21 VEXTERNC void Vatom_setPosition (Vatom ∗ *thee*, double *position*[3])

Set the atomic position.

Author

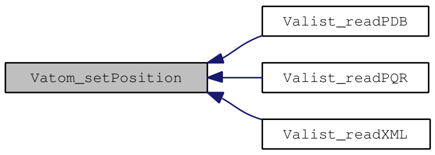
Nathan Baker

Parameters

thee Vatom object to be modified

position Coordinates (in Å)

Here is the caller graph for this function:



8.11.3.22 VEXTERNC void Vatom_setRadius (Vatom ∗ *thee*, double *radius*)

Set atomic radius.

Author

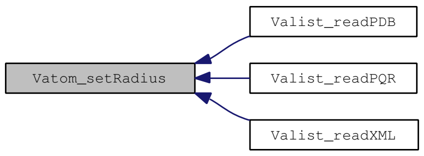
Nathan Baker

Parameters

thee Vatom object

radius Atomic radius (in Å)

Here is the caller graph for this function:



8.11.3.23 VEXTERNC void Vatom_setResName (Vatom ∗ *thee*, char *resName*[VMAX_RECLEN])

Set residue name.

Author

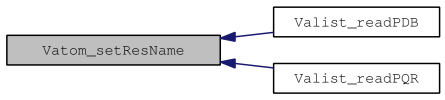
Jason Wagoner

Parameters

thee Vatom object

resName Residue Name

Here is the caller graph for this function:



8.12 Vcap class

Collection of routines which cap certain exponential and hyperbolic functions.

Files

- file vcap.h

  Contains declarations for class Vcap.

- file vcap.c

  Class Vcap methods.

Defines

- #define EXPMAX 85.00

  Maximum argument for exp(), sinh(), or cosh().

- #define EXPMIN -85.00

  Minimum argument for exp(), sinh(), or cosh().

Functions

- VEXTERNC double Vcap_exp (double x, int ∗ichop)

  Provide a capped exp() function.

- VEXTERNC double Vcap_sinh (double x, int ∗ichop)

  Provide a capped sinh() function.

- VEXTERNC double Vcap_cosh (double x, int ∗ichop)

  Provide a capped cosh() function.

8.12.1 Detailed Description

Collection of routines which cap certain exponential and hyperbolic functions.

Note

These routines are based on FORTRAN code by Mike Holst

8.12.2 Function Documentation

8.12.2.1 VEXTERNC double Vcap_cosh (double *x*, int ∗ *ichop*)

Provide a capped cosh() function.

If the argument x of Vcap_cosh() exceeds EXPMAX or EXPMIN, then we return cosh(EXPMAX) or cosh(EXPMIN) rather than cosh(x).

Note

Original FORTRAN routine from PMG library by Mike Holst Original notes: to control overflow in the hyperbolic and exp functions, note that the following are the argument limits of the various functions on various machines after which overflow occurs: Convex C240, Sun 3/60, Sun SPARC, IBM RS/6000: sinh, cosh, exp: maximal argument (abs value) = 88.0d0 dsinh, dcosh, dexp: maximal argument (abs value) = 709.0d0

Author

Nathan Baker (based on FORTRAN code by Mike Holst)

Returns

cosh(x) or capped equivalent

Parameters

x Argument to cosh()

ichop Set to 1 if function capped, 0 otherwise

8.12.2.2 VEXTERNC double Vcap_exp (double *x*, int ∗ *ichop*)

Provide a capped exp() function.

If the argument x of Vcap_exp() exceeds EXPMAX or EXPMIN, then we return exp(EXPMAX) or exp(EXPMIN) rather than exp(x).

Note

Original FORTRAN routine from PMG library by Mike Holst Original notes: to control overflow in the hyperbolic and exp functions, note that the following are the argument limits of the various functions on various machines after which overflow occurs: Convex C240, Sun 3/60, Sun SPARC, IBM RS/6000: sinh, cosh, exp: maximal argument (abs value) = 88.0d0 dsinh, dcosh, dexp: maximal argument (abs value) = 709.0d0

Author

Nathan Baker (based on FORTRAN code by Mike Holst)

Returns

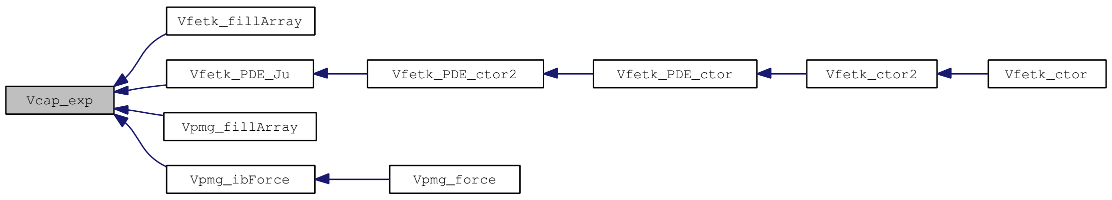
exp(x) or capped equivalent

Parameters

x Argument to exp()

ichop Set to 1 if function capped, 0 otherwise

Here is the caller graph for this function:



8.12.2.3 VEXTERNC double Vcap_sinh (double *x*, int ∗ *ichop*)

Provide a capped sinh() function.

If the argument x of Vcap_sinh() exceeds EXPMAX or EXPMIN, then we return sinh(EXPMAX) or sinh(EXPMIN) rather than sinh(x).

Note

Original FORTRAN routine from PMG library by Mike Holst Original notes: to control overflow in the hyperbolic and exp functions, note that the following are the argument limits of the various functions on various machines after which overflow occurs: Convex C240, Sun 3/60, Sun SPARC, IBM RS/6000: sinh, cosh, exp: maximal argument (abs value) = 88.0d0 dsinh, dcosh, dexp: maximal argument (abs value) = 709.0d0

Author

Nathan Baker (based on FORTRAN code by Mike Holst)

Returns

sinh(x) or capped equivalent

Parameters

x Argument to sinh()

ichop Set to 1 if function capped, 0 otherwise

8.13 Vclist class

Atom cell list.

Data Structures

- struct sVclistCell

  Atom cell list cell.

- struct sVclist

  Atom cell list.

Files

- file vclist.h

  Contains declarations for class Vclist.

- file vclist.c

  Class Vclist methods.

Typedefs

- typedef struct sVclistCell VclistCell

  Declaration of the VclistCell class as the VclistCell structure.

- typedef struct sVclist Vclist

  Declaration of the Vclist class as the Vclist structure.

- typedef enum eVclist_DomainMode Vclist_DomainMode

  Declaration of Vclist_DomainMode enumeration type.

Enumerations

- enum eVclist_DomainMode { CLIST_AUTO_DOMAIN, CLIST_MANUAL_-DOMAIN }

  Atom cell list domain setup mode.

Functions

- VEXTERNC unsigned long int Vclist_memChk (Vclist ∗thee)

  Get number of bytes in this object and its members.

- VEXTERNC double Vclist_maxRadius (Vclist ∗thee)

  Get the max probe radius value (in A) the cell list was constructed with.

- VEXTERNC Vclist ∗ Vclist_ctor (Valist ∗alist, double max_radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  Construct the cell list object.

- VEXTERNC Vrc_Codes Vclist_ctor2 (Vclist ∗thee, Valist ∗alist, double max_-radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  FORTRAN stub to construct the cell list object.

- VEXTERNC void Vclist_dtor (Vclist ∗∗thee)

  Destroy object.

- VEXTERNC void Vclist_dtor2 (Vclist ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC VclistCell ∗ Vclist_getCell (Vclist ∗thee, double position[VAPBS_DIM])

  Return cell corresponding to specified position or return VNULL.

- VEXTERNC VclistCell ∗ VclistCell_ctor (int natoms)

  Allocate and construct a cell list cell object.

- VEXTERNC Vrc_Codes VclistCell_ctor2 (VclistCell ∗thee, int natoms)

  Construct a cell list object.

- VEXTERNC void VclistCell_dtor (VclistCell ∗∗thee)

  Destroy object.

- VEXTERNC void VclistCell_dtor2 (VclistCell ∗thee)

  FORTRAN stub to destroy object.

8.13.1 Detailed Description

Atom cell list.

8.13.2 Enumeration Type Documentation

8.13.2.1 enum eVclist_DomainMode

Atom cell list domain setup mode.

Author

Nathan Baker

Enumerator:

CLIST_AUTO_DOMAIN Setup the cell list domain automatically to encompass the entire molecule

CLIST_MANUAL_DOMAIN Specify the cell list domain manually through the constructor

8.13.3 Function Documentation

8.13.3.1 VEXTERNC Vclist∗ Vclist_ctor (Valist ∗ *alist*, double *max_radius*, int *npts*[VAPBS_DIM], Vclist_DomainMode *mode*, double *lower_corner*[VAPBS_DIM], double *upper_corner*[VAPBS_DIM])

Construct the cell list object.

Author

Nathan Baker

Returns

Newly allocated Vclist object

Parameters

alist Molecule for cell list queries

max_radius Max probe radius (Å) to be queried

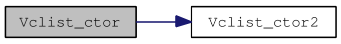
npts Number of in hash table points in each direction

mode Mode to construct table

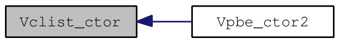
lower_corner Hash table lower corner for manual construction (see mode variable); ignored otherwise

upper_corner Hash table upper corner for manual construction (see mode variable); ignored otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



8.13.3.2 VEXTERNC Vrc_Codes Vclist_ctor2 (Vclist ∗ *thee*, Valist ∗ *alist*, double *max_radius*, int *npts*[VAPBS_DIM], Vclist_DomainMode *mode*, double *lower_corner*[VAPBS_DIM], double *upper_corner*[VAPBS_DIM])

FORTRAN stub to construct the cell list object.

Author

Nathan Baker, Yong Huang

Returns

Success enumeration

Parameters

thee Memory for Vclist objet

alist Molecule for cell list queries

max_radius Max probe radius (Å) to be queried

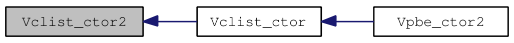
npts Number of in hash table points in each direction

mode Mode to construct table

lower_corner Hash table lower corner for manual construction (see mode variable); ignored otherwise

upper_corner Hash table upper corner for manual construction (see mode variable); ignored otherwise

Here is the caller graph for this function:



8.13.3.3 VEXTERNC void Vclist_dtor (Vclist ∗∗ *thee*)

Destroy object.

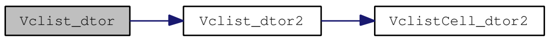
Author

Nathan Baker

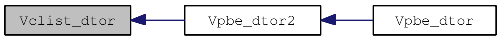
Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



Here is the caller graph for this function:



8.13.3.4 VEXTERNC void Vclist_dtor2 (Vclist ∗ *thee*)

FORTRAN stub to destroy object.

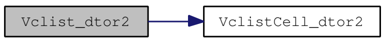
Author

Nathan Baker

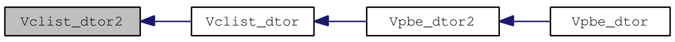
Parameters

thee Pointer to object

Here is the call graph for this function:



Here is the caller graph for this function:



8.13.3.5 VEXTERNC VclistCell∗ Vclist_getCell (Vclist ∗ *thee*, double *position*[VAPBS_DIM])

Return cell corresponding to specified position or return VNULL.

Author

Nathan Baker

Returns

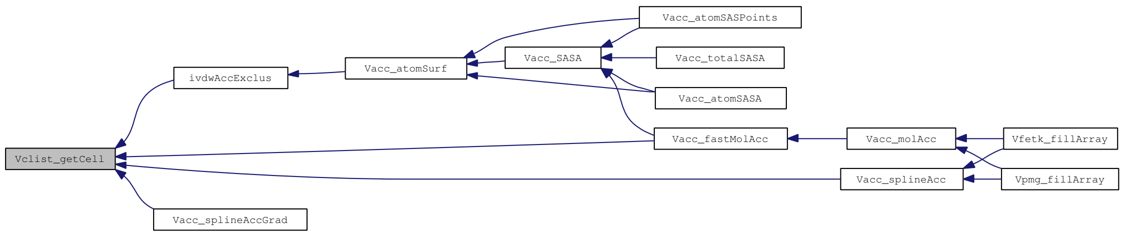
Pointer to VclistCell object or VNULL if no cell available (away from molecule).

Parameters

thee Pointer to Vclist cell list

position Position to evaluate

Here is the caller graph for this function:



8.13.3.6 VEXTERNC double Vclist_maxRadius (Vclist ∗ *thee*)

Get the max probe radius value (in A) the cell list was constructed with.

Author

Nathan Baker

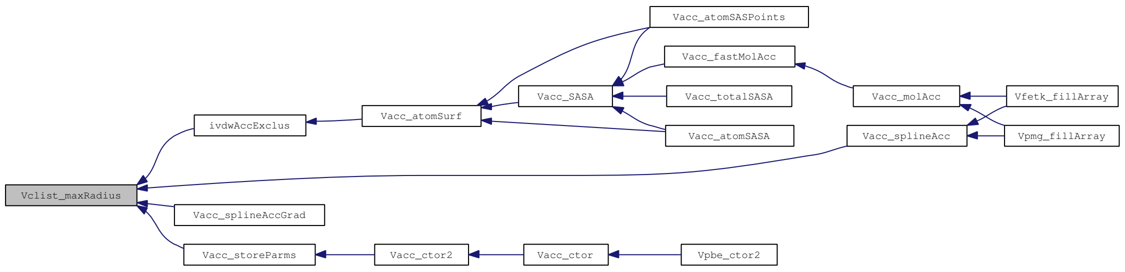
Returns

Max probe radius (in A)

Parameters

thee Cell list object

Here is the caller graph for this function:



8.13.3.7 VEXTERNC unsigned long int Vclist_memChk (Vclist $*$ *thee*)

Get number of bytes in this object and its members.

Author

Nathan Baker

Returns

Number of bytes allocated for object

Parameters

thee Object for memory check

8.13.3.8 VEXTERNC VclistCell$*$ VclistCell_ctor (int *natoms*)

Allocate and construct a cell list cell object.

Author

Nathan Baker

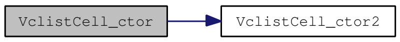
Returns

Pointer to newly-allocated and constructed object.

Parameters

natoms Number of atoms associated with this cell

Here is the call graph for this function:



8.13.3.9 VEXTERNC Vrc_Codes VclistCell_ctor2 (VclistCell ∗ *thee*, int *natoms*)

Construct a cell list object.

Author

Nathan Baker, Yong Huang

Returns

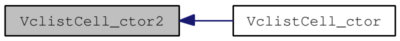
Success enumeration

Parameters

thee Memory location for object

natoms Number of atoms associated with this cell

Here is the caller graph for this function:



8.13.3.10 VEXTERNC void VclistCell_dtor (VclistCell ∗∗ *thee*)

Destroy object.

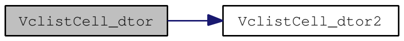
Author

Nathan Baker

Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



8.13.3.11 VEXTERNC void VclistCell_dtor2 (VclistCell ∗ *thee*)

FORTRAN stub to destroy object.

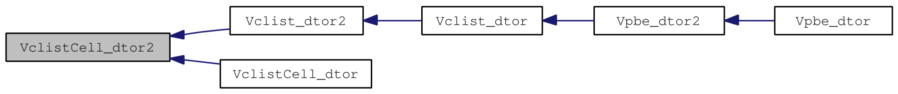
Author

Nathan Baker

Parameters

thee Pointer to object

Here is the caller graph for this function:



8.14 Vgreen class

Provides capabilities for pointwise evaluation of free space Green's function for point charges in a uniform dielectric.

Data Structures

- struct sVgreen

  Contains public data members for Vgreen class/module.

Files

- file vgreen.h

  Contains declarations for class Vgreen.

- file vgreen.c

  Class Vgreen methods.

Typedefs

- typedef struct sVgreen Vgreen

  Declaration of the Vgreen class as the Vgreen structure.

Functions

- VEXTERNC Valist ∗ Vgreen_getValist (Vgreen ∗thee)

  Get the atom list associated with this Green's function object.

- VEXTERNC unsigned long int Vgreen_memChk (Vgreen ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vgreen ∗ Vgreen_ctor (Valist ∗alist)

  Construct the Green's function oracle.

- VEXTERNC int Vgreen_ctor2 (Vgreen ∗thee, Valist ∗alist)

  FORTRAN stub to construct the Green's function oracle.

- VEXTERNC void Vgreen_dtor (Vgreen ∗∗thee)

Destruct the Green's function oracle.

- VEXTERNC void Vgreen_dtor2 (Vgreen ∗thee)

  FORTRAN stub to destruct the Green's function oracle.

- VEXTERNC int Vgreen_helmholtz (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val, double kappa)

  Get the Green's function for Helmholtz's equation integrated over the atomic point charges.

- VEXTERNC int Vgreen_helmholtzD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗gradx, double ∗grady, double ∗gradz, double kappa)

  Get the gradient of Green's function for Helmholtz's equation integrated over the atomic point charges.

- VEXTERNC int Vgreen_coulomb_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

  Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VEXTERNC int Vgreen_coulomb (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

  Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation or H. E. Johnston, R. Krasny FMM library (if available).

- VEXTERNC int Vgreen_coulombD_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VEXTERNC int Vgreen_coulombD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using either direct summation or H. E. Johnston/R. Krasny FMM library (if available).

8.14.1 Detailed Description

Provides capabilities for pointwise evaluation of free space Green's function for point charges in a uniform dielectric.

Note

Right now, these are very slow methods without any fast multipole acceleration.

Attention

8.14.2 Function Documentation

8.14.2.1 VEXTERNC int Vgreen_coulomb (Vgreen $*$ *thee*, int *npos*, double $*$ *x*, double $*$ *y*, double $*$ *z*, double $*$ *val*)

Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation or H. E. Johnston, R. Krasny FMM library (if available).

Returns the potential ϕ defined by

$$\phi(r) = \sum_i \frac{q_i}{r_i}$$

where q_i is the atomic charge (in e) and r_i is the distance to the observation point r. The potential is scaled to units of V.

Author

Nathan Baker

Parameters

thee Vgreen object

npos The number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

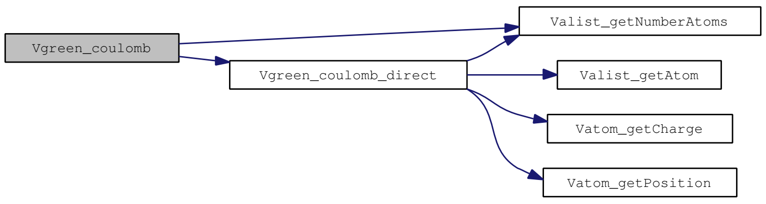
z The npos z-coordinates

val The npos values

Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



8.14.2.2 VEXTERNC int Vgreen_coulomb_direct (Vgreen $*$ *thee*, int *npos*, double $*$ *x*, double $*$ *y*, double $*$ *z*, double $*$ *val*)

Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

Returns the potential ϕ defined by

$$\phi(r) = \sum_i \frac{q_i}{r_i}$$

where q_i is the atomic charge (in e) and r_i is the distance to the observation point r. The potential is scaled to units of V.

Author

Nathan Baker

Parameters

thee Vgreen object

npos The number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

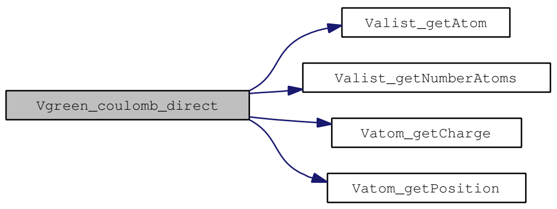
z The npos z-coordinates

val The npos values

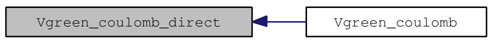
Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.2.3 VEXTERNC int Vgreen_coulombD (Vgreen $*$ *thee*, int *npos*, double $*$ *x*, double $*$ *y*, double $*$ *z*, double $*$ *pot*, double $*$ *gradx*, double $*$ *grady*, double $*$ *gradz*)

Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using either direct summation or H. E. Johnston/R. Krasny FMM library (if available).

Returns the field $\nabla\phi$ defined by

$$\nabla\phi(r) = \sum_i \frac{q_i}{r_i}$$

where q_i is the atomic charge (in e) and r_i is the distance to the observation point r. The field is scaled to units of V/Å.

Author

Nathan Baker

Parameters

thee Vgreen object

npos The number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

z The npos z-coordinates

pot The npos potential values

gradx The npos gradient x-components

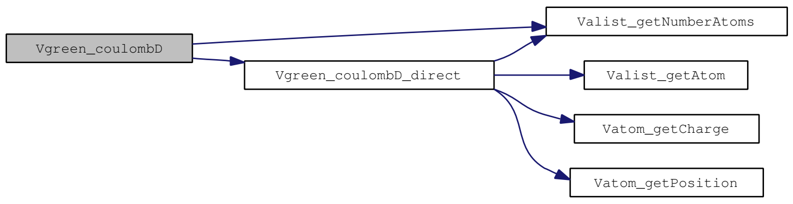
grady The npos gradient y-components

gradz The npos gradient z-components

Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



8.14.2.4 VEXTERNC int Vgreen_coulombD_direct (Vgreen ∗ *thee*, int *npos*, double ∗ *x*, double ∗ *y*, double ∗ *z*, double ∗ *pot*, double ∗ *gradx*, double ∗ *grady*, double ∗ *gradz*)

Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

Returns the field $\nabla\phi$ defined by

$$\nabla\phi(r) = \sum_i \frac{q_i}{r_i}$$

where q_i is the atomic charge (in e) and r_i is the distance to the observation point r. The field is scaled to units of V/Å.

Author

Nathan Baker

Parameters

thee Vgreen object

npos The number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

z The npos z-coordinates

pot The npos potential values

gradx The npos gradient x-components

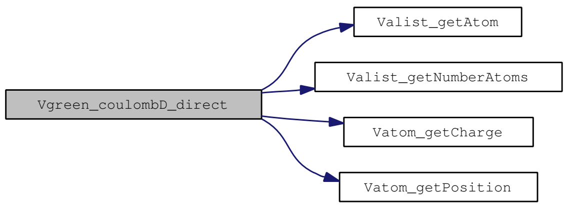
grady The npos gradient y-components

gradz The npos gradient z-components

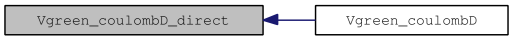
Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.2.5 VEXTERNC Vgreen∗ Vgreen_ctor (Valist ∗ *alist*)

Construct the Green's function oracle.

Author

Nathan Baker

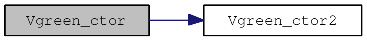
Parameters

alist Atom (charge) list associated with object

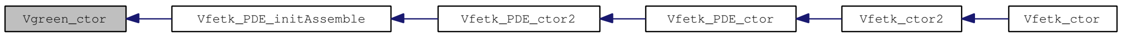
Returns

Pointer to newly allocated Green's function oracle

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.2.6 VEXTERNC int Vgreen_ctor2 (Vgreen ∗ *thee*, Valist ∗ *alist*)

FORTRAN stub to construct the Green's function oracle.

Author

Nathan Baker

Parameters

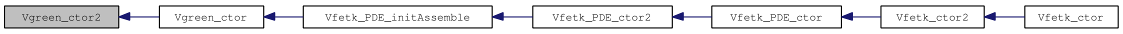
thee Pointer to memory allocated for object

alist Atom (charge) list associated with object

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.14.2.7 VEXTERNC void Vgreen_dtor (Vgreen ∗∗ *thee*)

Destruct the Green's function oracle.

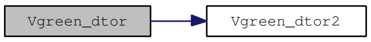
Author

Nathan Baker

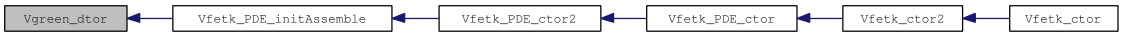
Parameters

thee Pointer to memory location for object

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.2.8 VEXTERNC void Vgreen_dtor2 (Vgreen ∗ *thee*)

FORTRAN stub to destruct the Green's function oracle.

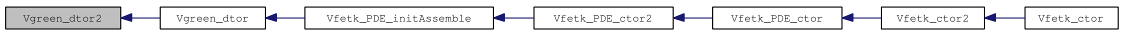
Author

Nathan Baker

Parameters

thee Pointer to object

Here is the caller graph for this function:



8.14.2.9 VEXTERNC Valist∗ Vgreen_getValist (Vgreen ∗ *thee*)

Get the atom list associated with this Green's function object.

Author

Nathan Baker

Parameters

thee Vgreen object

Returns

Pointer to Valist object associated with this Green's function object

8.14.2.10 VEXTERNC int Vgreen_helmholtz (Vgreen ∗ *thee*, int *npos*, double ∗ *x*, double ∗ *y*, double ∗ *z*, double ∗ *val*, double *kappa*)

Get the Green's function for Helmholtz's equation integrated over the atomic point charges.

Returns the potential ϕ defined by

$$\phi(r) = \sum_i \frac{q_i e^{-\kappa r_i}}{r_i}$$

where κ is the inverse screening length (in Å) q_i is the atomic charge (in e), and r_i r_i is the distance from atom i to the observation point r. The potential is scaled to units of V.

Author

Nathan Baker

Bug

Not implemented yet

Note

Not implemented yet

Parameters

thee Vgreen object

npos Number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

z The npos z-coordinates

val The npos values

kappa The value of κ (see above)

Returns

1 if successful, 0 otherwise

8.14.2.11 VEXTERNC int Vgreen_helmholtzD (Vgreen $*$ *thee*, int *npos*, double $*$ *x*, double $*$ *y*, double $*$ *z*, double $*$ *gradx*, double $*$ *grady*, double $*$ *gradz*, double *kappa*)

Get the gradient of Green's function for Helmholtz's equation integrated over the atomic point charges.

Returns the field $\nabla\phi$ defined by

$$\nabla\phi(r) = \nabla\sum_i \frac{q_i e^{-\kappa r_i}}{r_i}$$

where κ is the inverse screening length (in Å). q_i is the atomic charge (in e), and r_i r_i is the distance from atom i to the observation point r. The potential is scaled to units of V/Å.

Author

Nathan Baker

Bug

Not implemented yet

Note

Not implemented yet

Parameters

thee Vgreen object

npos The number of positions to evaluate

x The npos x-coordinates

y The npos y-coordinates

z The npos z-coordinates

gradx The npos gradient x-components

grady The npos gradient y-components

gradz The npos gradient z-components

kappa The value of κ (see above)

Returns

int 1 if sucessful, 0 otherwise

8.14.2.12 VEXTERNC unsigned long int Vgreen_memChk (Vgreen $*$ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

Parameters

thee Vgreen object

Returns

The memory used by this structure and its contents in bytes

8.15 Vhal class

A "class" which consists solely of macro definitions which are used by several other classes.

Files

- file vhal.h

  Contains generic macro definitions for APBS.

Defines

- #define APBS_TIMER_WALL_CLOCK 26

  APBS total execution timer ID.

- #define APBS_TIMER_SETUP 27

  APBS setup timer ID.

- #define APBS_TIMER_SOLVER 28

  APBS solver timer ID.

- #define APBS_TIMER_ENERGY 29

  APBS energy timer ID.

- #define APBS_TIMER_FORCE 30

  APBS force timer ID.

- #define APBS_TIMER_TEMP1 31

  APBS temp timer #1 ID.

- #define APBS_TIMER_TEMP2 32

  APBS temp timer #2 ID.

- #define MAXMOL 5

  The maximum number of molecules that can be involved in a single PBE calculation.

- #define MAXION 10

  The maximum number of ion species that can be involved in a single PBE calculation.

- #define MAXFOCUS 5

The maximum number of times an MG calculation can be focused.

- #define VMGNLEV 4

  Minimum number of levels in a multigrid calculations.

- #define VREDFRAC 0.25

  Maximum reduction of grid spacing during a focusing calculation.

- #define VAPBS_NVS 4

  Number of vertices per simplex (hard-coded to 3D).

- #define VAPBS_DIM 3

  Our dimension.

- #define VAPBS_RIGHT 0

  Face definition for a volume.

- #define VAPBS_FRONT 1

  Face definition for a volume.

- #define VAPBS_UP 2

  Face definition for a volume.

- #define VAPBS_LEFT 3

  Face definition for a volume.

- #define VAPBS_BACK 4

  Face definition for a volume.

- #define VAPBS_DOWN 5

  Face definition for a volume.

- #define VPMGSMALL 1e-12

  A small number used in Vpmg to decide if points are on/off grid-lines or non-zer0 (etc.).

- #define SINH_MIN -85.0

  Used to set the min values acceptable for sinh chopping.

- #define SINH_MAX 85.0

  Used to set the max values acceptable for sinh chopping.

- #define VF77_MANGLE(name, NAME) name

Name-mangling macro for using FORTRAN functions in C code.

- #define VFLOOR(value) floor(value)

  Wrapped floor to fix floating point issues in the Intel compiler.

- #define VEMBED(rctag)

  Allows embedding of RCS ID tags in object files.

Typedefs

- typedef enum eVhal_PBEType Vhal_PBEType

  Declaration of the Vhal_PBEType type as the Vhal_PBEType enum.

- typedef enum eVhal_IPKEYType Vhal_IPKEYType

  Declaration of the Vhal_IPKEYType type as the Vhal_IPKEYType enum.

- typedef enum eVhal_NONLINType Vhal_NONLINType

  Declaration of the Vhal_NONLINType type as the Vhal_NONLINType enum.

- typedef enum eVoutput_Format Voutput_Format

  Declaration of the Voutput_Format type as the VOutput_Format enum.

- typedef enum eVbcfl Vbcfl

  Declare Vbcfl type.

- typedef enum eVsurf_Meth Vsurf_Meth

  Declaration of the Vsurf_Meth type as the Vsurf_Meth enum.

- typedef enum eVchrg_Meth Vchrg_Meth

  Declaration of the Vchrg_Meth type as the Vchrg_Meth enum.

- typedef enum eVchrg_Src Vchrg_Src

  Declaration of the Vchrg_Src type as the Vchrg_Meth enum.

- typedef enum eVdata_Type Vdata_Type

  Declaration of the Vdata_Type type as the Vdata_Type enum.

- typedef enum eVdata_Format Vdata_Format

  Declaration of the Vdata_Format type as the Vdata_Format enum.

Enumerations

- enum eVrc_Codes { **VRC_WARNING** = -1, VRC_FAILURE = 0, VRC_-SUCCESS = 1 }

    Return code enumerations.

- enum eVsol_Meth {

    VSOL_CGMG, **VSOL_Newton**, **VSOL_MG**, **VSOL_CG**,

    VSOL_SOR, **VSOL_RBGS**, **VSOL_WJ**, **VSOL_Richardson**,

    VSOL_CGMGAqua, **VSOL_NewtonAqua** }

    Solution Method enumerations.

- enum eVsurf_Meth {

    VSM_MOL = 0, VSM_MOLSMOOTH = 1, VSM_SPLINE = 2, VSM_-SPLINE3 = 3,

    VSM_SPLINE4 = 4 }

    Types of molecular surface definitions.

- enum eVhal_PBEType {

    PBE_LPBE, PBE_NPBE, PBE_LRPBE, **PBE_NRPBE**,

    PBE_SMPBE }

    Version of PBE to solve.

- enum eVhal_IPKEYType { IPKEY_SMPBE = -2, IPKEY_LPBE, IPKEY_-NPBE }

    Type of ipkey to use for MG methods.

- enum eVhal_NONLINType {

    NONLIN_LPBE = 0, **NONLIN_NPBE**, **NONLIN_SMPBE**, **NONLIN_-LPBEAQUA**,

    NONLIN_NPBEAQUA }

    Type of nonlinear to use for MG methods.

- enum eVoutput_Format { OUTPUT_NULL, OUTPUT_FLAT }

    Output file format.

- enum eVbcfl {

    BCFL_ZERO = 0, BCFL_SDH = 1, BCFL_MDH = 2, BCFL_UNUSED = 3,

    BCFL_FOCUS = 4, BCFL_MEM = 5 }

    Types of boundary conditions.

- enum eVchrg_Meth { VCM_TRIL = 0, VCM_BSPL2 = 1, VCM_BSPL4 = 2 }

  Types of charge discretization methods.

- enum eVchrg_Src { VCM_CHARGE = 0, VCM_PERMANENT = 1, VCM_-INDUCED = 2, VCM_NLINDUCED = 3 }

  Charge source.

- enum eVdata_Type {
  VDT_CHARGE, VDT_POT, VDT_SMOL, VDT_SSPL,
  VDT_VDW, VDT_IVDW, VDT_LAP, VDT_EDENS,
  VDT_NDENS, VDT_QDENS, VDT_DIELX, VDT_DIELY,
  VDT_DIELZ, VDT_KAPPA }

  Types of (scalar) data that can be written out of APBS.

- enum eVdata_Format { VDF_DX = 0, VDF_UHBD = 1, VDF_AVS = 2, VDF_-MCSF = 3 }

  Format of data for APBS I/O.

8.15.1 Detailed Description

A "class" which consists solely of macro definitions which are used by several other classes.

8.15.2 Define Documentation

8.15.2.1 #define VAPBS_BACK 4

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.2 #define VAPBS_DOWN 5

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.3 #define VAPBS_FRONT 1

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.4 #define VAPBS_LEFT 3

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.5 #define VAPBS_RIGHT 0

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.6 #define VAPBS_UP 2

Face definition for a volume.

Note

Consistent with PMG if RIGHT = EAST, BACK = SOUTH

8.15.2.7 #define VEMBED(rctag)

Value:

```
VPRIVATE const char* rctag; \
         static void* use_rcsid=(0 ? &use_rcsid : (void**)&rcsid);
```

Allows embedding of RCS ID tags in object files.

Author

Mike Holst

8.15.2.8 #define VFLOOR(value) floor(value)

Wrapped floor to fix floating point issues in the Intel compiler.

Author

Todd Dolinksy

8.15.3 Enumeration Type Documentation

8.15.3.1 enum eVbcfl

Types of boundary conditions.

Author

Nathan Baker

Enumerator:

BCFL_ZERO Zero Dirichlet boundary conditions

BCFL_SDH Single-sphere Debye-Huckel Dirichlet boundary condition

BCFL_MDH Multiple-sphere Debye-Huckel Dirichlet boundary condition

BCFL_UNUSED Unused boundary condition method (placeholder)

BCFL_FOCUS Focusing Dirichlet boundary condition

BCFL_MEM Focusing membrane boundary condition

8.15.3.2 enum eVchrg_Meth

Types of charge discretization methods.

Author

Nathan Baker

Enumerator:

VCM_TRIL Trilinear interpolation of charge to 8 nearest grid points. The traditional method; not particularly good to use with PBE forces.

VCM_BSPL2 Cubic B-spline across nearest- and next-nearest-neighbors. Mainly for use in grid-sensitive applications (such as force calculations).

VCM_BSPL4 5th order B-spline for AMOEBA permanent multipoles.

8.15.3.3 enum eVchrg_Src

Charge source.

Author

Michael Schnieders

Enumerator:

VCM_CHARGE Partial Charge source distribution

VCM_PERMANENT Permanent Multipole source distribution

VCM_INDUCED Induced Dipole source distribution

VCM_NLINDUCED NL Induced Dipole source distribution

8.15.3.4 enum eVdata_Format

Format of data for APBS I/O.

Author

Nathan Baker

Enumerator:

VDF_DX OpenDX (Data Explorer) format

VDF_UHBD UHBD format

VDF_AVS AVS UCD format

VDF_MCSF FEtk MC Simplex Format (MCSF)

8.15.3.5 enum eVdata_Type

Types of (scalar) data that can be written out of APBS.

Author

Nathan Baker

Enumerator:

VDT_CHARGE Charge distribution (e)

VDT_POT Potential (kT/e)

VDT_SMOL Solvent accessibility defined by molecular/Connolly surface definition (1 = accessible, 0 = inaccessible)

VDT_SSPL Spline-based solvent accessibility (1 = accessible, 0 = inaccessible)

VDT_VDW van der Waals-based accessibility (1 = accessible, 0 = inaccessible)

VDT_IVDW Ion accessibility/inflated van der Waals (1 = accessible, 0 = inaccessible)

VDT_LAP Laplacian of potential (kT/e/A^2)

VDT_EDENS Energy density $\epsilon(\nabla u)^2$, where u is potential (kT/e/A)^2

VDT_NDENS Ion number density $\sum c_i \exp(-q_i u)^2$, where u is potential (output in M)

VDT_QDENS Ion charge density $\sum q_i c_i \exp(-q_i u)^2$, where u is potential (output in $e_c M$)

VDT_DIELX Dielectric x-shifted map as calculated with the currently specified scheme (dimensionless)

VDT_DIELY Dielectric y-shifted map as calculated with the currently specified scheme (dimensionless)

VDT_DIELZ Dielectric y-shifted map as calculated with the currently specified scheme (dimensionless)

VDT_KAPPA Kappa map as calculated with the currently specified scheme ($^{-3}$)

8.15.3.6 enum eVhal_IPKEYType

Type of ipkey to use for MG methods.

Enumerator:

IPKEY_SMPBE SMPBE ipkey

IPKEY_LPBE LPBE ipkey

IPKEY_NPBE NPBE ipkey

8.15.3.7 enum eVhal_PBEType

Version of PBE to solve.

Enumerator:

PBE_LPBE Traditional Poisson-Boltzmann equation, linearized

PBE_NPBE Traditional Poisson-Boltzmann equation, full

PBE_LRPBE Regularized Poisson-Boltzmann equation, linearized

PBE_SMPBE $<$ Regularized Poisson-Boltzmann equation, full SM PBE

8.15.3.8 enum eVoutput_Format

Output file format.

Enumerator:

OUTPUT_NULL	No output
OUTPUT_FLAT	Output in flat-file format

8.15.3.9 enum eVrc_Codes

Return code enumerations.

Author

David Gohara

Note

Note that the enumerated values are opposite the standard for FAILURE and SUCCESS

Enumerator:

VRC_FAILURE	A non-fatal error
VRC_SUCCESS	A fatal error

8.15.3.10 enum eVsol_Meth

Solution Method enumerations.

Author

David Gohara

Note

Note that the enumerated values are opposite the standard for FAILURE and SUCCESS

8.15.3.11 enum eVsurf_Meth

Types of molecular surface definitions.

Author

Nathan Baker

Enumerator:

VSM_MOL Ion accessibility is defined using inflated van der Waals radii, the dielectric coefficient ( ) is defined using the molecular (Conolly) surface definition without smoothing

VSM_MOLSMOOTH As VSM_MOL but with a simple harmonic average smoothing

VSM_SPLINE Spline-based surface definitions. This is primarily for use with force calculations, since it requires substantial reparameterization of radii. This is based on the work of Im et al, Comp. Phys. Comm. 111 , (1998) and uses a cubic spline to define a smoothly varying characteristic function for the surface-based parameters. Ion accessibility is defined using inflated van der Waals radii with the spline function and the dielectric coefficient is defined using the standard van der Waals radii with the spline function.

VSM_SPLINE3 A 5th order polynomial spline is used to create a smoothly varying characteristic function (continuity through 2nd derivatives) for surface based paramters.

VSM_SPLINE4 A 7th order polynomial spline is used to create a smoothly varying characteristic function (continuity through 3rd derivatives) for surface based paramters.

8.16 Vparam class

Reads and assigns charge/radii parameters.

Data Structures

- struct sVparam_AtomData

  AtomData sub-class; stores atom data.

- struct Vparam_ResData

  ResData sub-class; stores residue data.

- struct Vparam

  Reads and assigns charge/radii parameters.

Files

- file vparam.h

  Contains declarations for class Vparam.

- file vparam.c

  Class Vparam methods.

Typedefs

- typedef struct sVparam_AtomData Vparam_AtomData

  Declaration of the Vparam_AtomData class as the sVparam_AtomData structure.

- typedef struct Vparam_ResData Vparam_ResData

  Declaration of the Vparam_ResData class as the Vparam_ResData structure.

- typedef struct Vparam Vparam

  Declaration of the Vparam class as the Vparam structure.

Functions

- VEXTERNC unsigned long int Vparam_memChk (Vparam $*$thee)

  Get number of bytes in this object and its members.

- VEXTERNC Vparam_AtomData ∗ Vparam_AtomData_ctor ()

  Construct the object.

- VEXTERNC int Vparam_AtomData_ctor2 (Vparam_AtomData ∗thee)

  FORTRAN stub to construct the object.

- VEXTERNC void Vparam_AtomData_dtor (Vparam_AtomData ∗∗thee)

  Destroy object.

- VEXTERNC void Vparam_AtomData_dtor2 (Vparam_AtomData ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC void Vparam_AtomData_copyTo (Vparam_AtomData ∗thee, Vparam_AtomData ∗dest)

  Copy current atom object to destination.

- VEXTERNC void Vparam_ResData_copyTo (Vparam_ResData ∗thee, Vparam_ResData ∗dest)

  Copy current residue object to destination.

- VEXTERNC void Vparam_AtomData_copyFrom (Vparam_AtomData ∗thee, Vparam_AtomData ∗src)

  Copy current atom object from another.

- VEXTERNC Vparam_ResData ∗ Vparam_ResData_ctor (Vmem ∗mem)

  Construct the object.

- VEXTERNC int Vparam_ResData_ctor2 (Vparam_ResData ∗thee, Vmem ∗mem)

  FORTRAN stub to construct the object.

- VEXTERNC void Vparam_ResData_dtor (Vparam_ResData ∗∗thee)

  Destroy object.

- VEXTERNC void Vparam_ResData_dtor2 (Vparam_ResData ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC Vparam ∗ Vparam_ctor ()

  Construct the object.

- VEXTERNC int Vparam_ctor2 (Vparam ∗thee)

  FORTRAN stub to construct the object.

- VEXTERNC void Vparam_dtor (Vparam ∗∗thee)

  Destroy object.

- VEXTERNC void Vparam_dtor2 (Vparam ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC Vparam_ResData ∗ Vparam_getResData (Vparam ∗thee, char resName[VMAX_ARGLEN])

  Get residue data.

- VEXTERNC Vparam_AtomData ∗ Vparam_getAtomData (Vparam ∗thee, char resName[VMAX_ARGLEN], char atomName[VMAX_ARGLEN])

  Get atom data.

- VEXTERNC int Vparam_readFlatFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read a flat-file format parameter database.

- VEXTERNC int Vparam_readXMLFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read an XML format parameter database.

- VPRIVATE int readFlatFileLine (Vio ∗sock, Vparam_AtomData ∗atom)

  Read a single line of the flat file database.

- VPRIVATE int readXMLFileAtom (Vio ∗sock, Vparam_AtomData ∗atom)

  Read atom information from an XML file.

Variables

- VPRIVATE char ∗ MCwhiteChars = " =,;\t\n\r"

  Whitespace characters for socket reads.

- VPRIVATE char ∗ MCcommChars = "#%"

  Comment characters for socket reads.

- VPRIVATE char ∗ MCxmlwhiteChars = " =,;\t\n\r<>"

  Whitespace characters for XML socket reads.

8.16.1 Detailed Description

Reads and assigns charge/radii parameters.

8.16.2 Function Documentation

8.16.2.1 VPRIVATE int readFlatFileLine (Vio ∗ *sock*, Vparam_AtomData ∗ *atom*)

Read a single line of the flat file database.

Author

Nathan Baker

Parameters

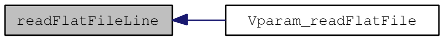
sock Socket ready for reading

atom Atom to hold parsed data

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.16.2.2 VPRIVATE int readXMLFileAtom (Vio ∗ *sock*, Vparam_AtomData ∗ *atom*)

Read atom information from an XML file.

Author

Todd Dolinsky

Parameters

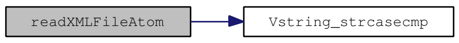
sock Socket ready for reading

atom Atom to hold parsed data

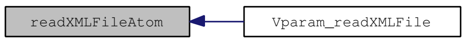
Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.3 VEXTERNC void Vparam_AtomData_copyFrom (Vparam_AtomData $*$ *thee*, Vparam_AtomData $*$ *src*)

Copy current atom object from another.

Author

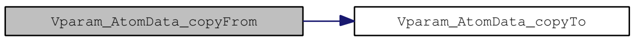
Nathan Baker

Parameters

thee Pointer to destination object

src Pointer to source object

Here is the call graph for this function:



8.16.2.4 VEXTERNC void Vparam_AtomData_copyTo (Vparam_AtomData $*$ *thee*, Vparam_AtomData $*$ *dest*)

Copy current atom object to destination.

Author

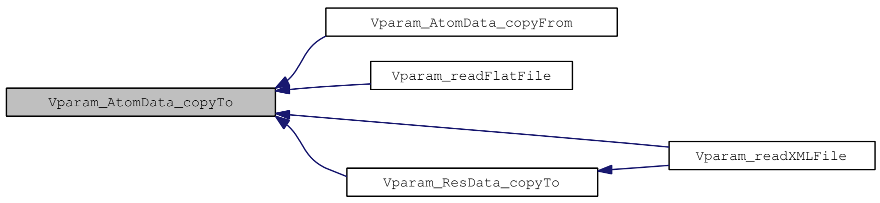
Nathan Baker

Parameters

thee Pointer to source object

dest Pointer to destination object

Here is the caller graph for this function:



8.16.2.5 VEXTERNC Vparam_AtomData∗ Vparam_AtomData_ctor ()

Construct the object.

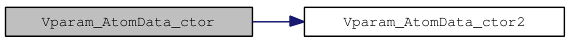
Author

Nathan Baker

Returns

Newly allocated object

Here is the call graph for this function:



8.16.2.6 VEXTERNC int Vparam_AtomData_ctor2 (Vparam_AtomData ∗ *thee*)

FORTRAN stub to construct the object.

Author

Nathan Baker

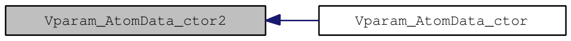
Parameters

thee Allocated memory

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.16.2.7 VEXTERNC void Vparam_AtomData_dtor (Vparam_AtomData ∗∗ *thee*)

Destroy object.

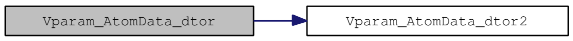
Author

Nathan Baker

Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



8.16.2.8 VEXTERNC void Vparam_AtomData_dtor2 (Vparam_AtomData ∗ *thee*)

FORTRAN stub to destroy object.

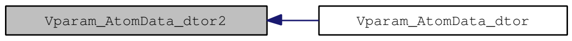
Author

Nathan Baker

Parameters

thee Pointer to object

Here is the caller graph for this function:



8.16.2.9 VEXTERNC Vparam∗ Vparam_ctor ()

Construct the object.

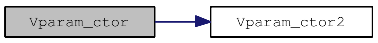
Author

Nathan Baker

Returns

Newly allocated Vparam object

Here is the call graph for this function:



8.16.2.10 VEXTERNC int Vparam_ctor2 (Vparam ∗ *thee*)

FORTRAN stub to construct the object.

Author

Nathan Baker

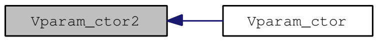
Parameters

thee Allocated Vparam memory

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.16.2.11 VEXTERNC void Vparam_dtor (Vparam ∗∗ *thee*)

Destroy object.

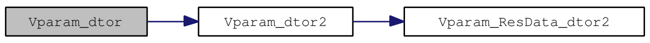
Author

Nathan Baker

Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



8.16.2.12 VEXTERNC void Vparam_dtor2 (Vparam ∗ *thee*)

FORTRAN stub to destroy object.

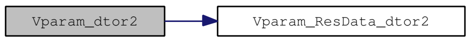
Author

Nathan Baker

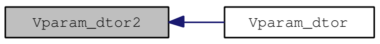
Parameters

thee Pointer to object

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.13 VEXTERNC Vparam_AtomData∗ Vparam_getAtomData (Vparam ∗ *thee*, char *resName*[VMAX_ARGLEN], char *atomName*[VMAX_ARGLEN])

Get atom data.

Author

Nathan Baker

Parameters

thee Vparam object

resName Residue name

atomName Atom name

Returns

Pointer to the desired atom object or VNULL if residue not found

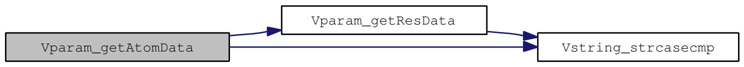
Note

Some method to initialize the database must be called before this method (e.g.,

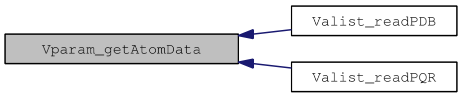
See also

Vparam_readFlatFile)

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.14 VEXTERNC Vparam_ResData∗ Vparam_getResData (Vparam ∗ *thee*, char *resName*[VMAX_ARGLEN])

Get residue data.

Author

Nathan Baker

Parameters

thee Vparam object

resName Residue name

Returns

Pointer to the desired residue object or VNULL if residue not found

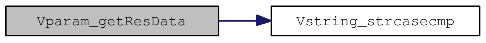
Note

Some method to initialize the database must be called before this method (e.g.,

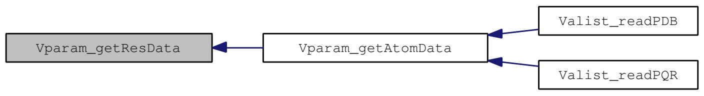
See also

Vparam_readFlatFile)

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.15 VEXTERNC unsigned long int Vparam_memChk (Vparam $*$ *thee*)

Get number of bytes in this object and its members.

Author

Nathan Baker

Parameters

thee Vparam object

Returns

Number of bytes allocated for object

8.16.2.16 VEXTERNC int Vparam_readFlatFile (Vparam $*$ *thee*, const char $*$ *iodev*, const char $*$ *iofmt*, const char $*$ *thost*, const char $*$ *fname*)

Read a flat-file format parameter database.

Author

Nathan Baker

Parameters

thee Vparam object

iodev Input device type (FILE/BUFF/UNIX/INET)

iofmt Input device format (ASCII/XDR)

thost Input hostname (for sockets)

fname Input FILE/BUFF/UNIX/INET name (see note below for format)

Returns

1 if successful, 0 otherwise

Note

The database file should have the following format:

```
 RESIDUE ATOM CHARGE RADIUS EPSILON
```

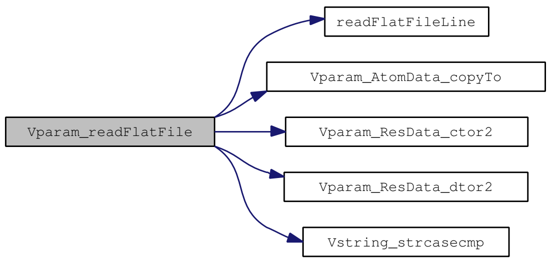
where RESIDUE is the residue name string, ATOM is the atom name string, CHARGE is the charge in e, RADIUS is the van der Waals radius (σ_i) in Å, and EPSILON is the van der Waals well-depth (ϵ_i) in kJ/mol. See the Vparam structure documentation for the precise definitions of σ_i and ϵ_i.

ASCII-format flat files are provided with the APBS source code:

tools/conversion/vparam-amber-parm94.dat AMBER parm94 parameters

tools/conversion/vparam-charmm-par_all27.dat CHARMM par_all27_prot_na parameters

Here is the call graph for this function:



8.16.2.17 VEXTERNC int Vparam_readXMLFile (Vparam ∗ *thee*, const char ∗ *iodev*, const char ∗ *iofmt*, const char ∗ *thost*, const char ∗ *fname*)

Read an XML format parameter database.

Author

Todd Dolinsky

Parameters

thee Vparam object

iodev Input device type (FILE/BUFF/UNIX/INET)

iofmt Input device format (ASCII/XDR)

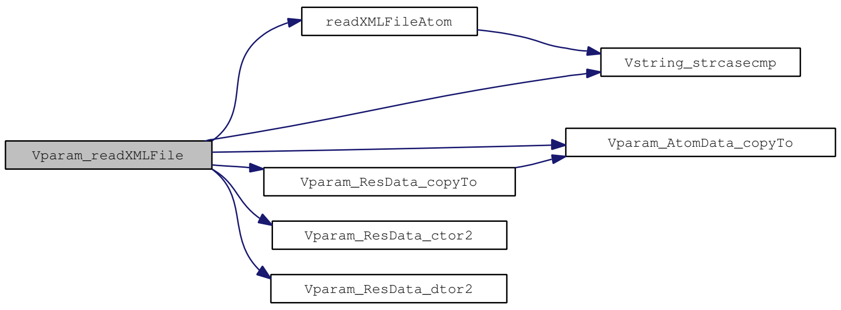
thost Input hostname (for sockets)

fname Input FILE/BUFF/UNIX/INET name

Returns

1 if successful, 0 otherwise

Here is the call graph for this function:



8.16.2.18 VEXTERNC void Vparam_ResData_copyTo (Vparam_ResData ∗ *thee*, Vparam_ResData ∗ *dest*)

Copy current residue object to destination.

Author

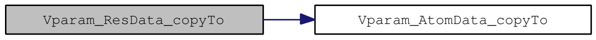
Todd Dolinsky

Parameters

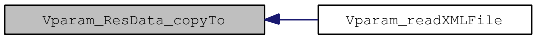
thee Pointer to source object

dest Pointer to destination object

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.19 VEXTERNC Vparam_ResData∗ Vparam_ResData_ctor (Vmem ∗ *mem*)

Construct the object.

Author

Nathan Baker

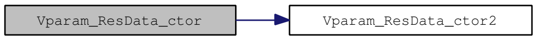
Parameters

mem Memory object of Vparam master class

Returns

Newly allocated object

Here is the call graph for this function:



8.16.2.20 VEXTERNC int Vparam_ResData_ctor2 (Vparam_ResData ∗ *thee*, Vmem ∗ *mem*)

FORTRAN stub to construct the object.

Author

Nathan Baker

Parameters

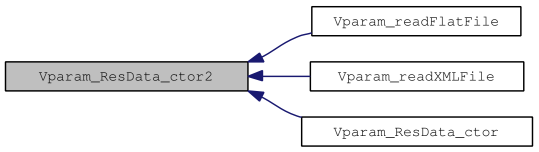
thee Allocated memory

mem Memory object of Vparam master class

Returns

1 if successful, 0 otherwise

Here is the caller graph for this function:



8.16.2.21 VEXTERNC void Vparam_ResData_dtor (Vparam_ResData ** *thee*)

Destroy object.

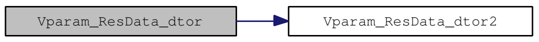
Author

Nathan Baker

Parameters

thee Pointer to memory location of object

Here is the call graph for this function:



8.16.2.22 VEXTERNC void Vparam_ResData_dtor2 (Vparam_ResData * *thee*)

FORTRAN stub to destroy object.

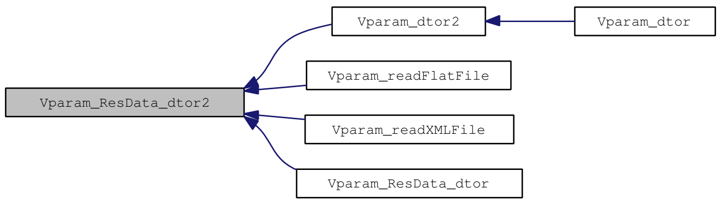
Author

Nathan Baker

Parameters

thee Pointer to object

Here is the caller graph for this function:



8.17 Vpbe class

The Poisson-Boltzmann master class.

Data Structures

- struct sVpbe

  Contains public data members for Vpbe class/module.

Files

- file vpbe.h

  Contains declarations for class Vpbe.

- file vpbe.c

  Class Vpbe methods.

Typedefs

- typedef struct sVpbe Vpbe

  Declaration of the Vpbe class as the Vpbe structure.

Functions

- VEXTERNC Valist ∗ Vpbe_getValist (Vpbe ∗thee)

  Get atom list.

- VEXTERNC Vacc ∗ Vpbe_getVacc (Vpbe ∗thee)

  Get accessibility oracle.

- VEXTERNC double Vpbe_getBulkIonicStrength (Vpbe ∗thee)

  Get bulk ionic strength.

- VEXTERNC double Vpbe_getMaxIonRadius (Vpbe ∗thee)

  Get maximum radius of ion species.

- VEXTERNC double Vpbe_getTemperature (Vpbe ∗thee)

  Get temperature.

- VEXTERNC double Vpbe_getSoluteDiel (Vpbe ∗thee)

  Get solute dielectric constant.

- VEXTERNC double Vpbe_getGamma (Vpbe ∗thee)

  Get apolar coefficient.

- VEXTERNC double Vpbe_getSoluteRadius (Vpbe ∗thee)

  Get sphere radius which bounds biomolecule.

- VEXTERNC double Vpbe_getSoluteXlen (Vpbe ∗thee)

  Get length of solute in x dimension.

- VEXTERNC double Vpbe_getSoluteYlen (Vpbe ∗thee)

  Get length of solute in y dimension.

- VEXTERNC double Vpbe_getSoluteZlen (Vpbe ∗thee)

  Get length of solute in z dimension.

- VEXTERNC double ∗ Vpbe_getSoluteCenter (Vpbe ∗thee)

  Get coordinates of solute center.

- VEXTERNC double Vpbe_getSoluteCharge (Vpbe ∗thee)

  Get total solute charge.

- VEXTERNC double Vpbe_getSolventDiel (Vpbe ∗thee)

  Get solvent dielectric constant.

- VEXTERNC double Vpbe_getSolventRadius (Vpbe ∗thee)

  Get solvent molecule radius.

- VEXTERNC double Vpbe_getXkappa (Vpbe ∗thee)

  Get Debye-Huckel parameter.

- VEXTERNC double Vpbe_getDeblen (Vpbe ∗thee)

  Get Debye-Huckel screening length.

- VEXTERNC double Vpbe_getZkappa2 (Vpbe ∗thee)

  Get modified squared Debye-Huckel parameter.

- VEXTERNC double Vpbe_getZmagic (Vpbe ∗thee)

  Get charge scaling factor.

- VEXTERNC double Vpbe_getzmem (Vpbe ∗thee)

  Get z position of the membrane bottom.

- VEXTERNC double Vpbe_getLmem (Vpbe ∗thee)

  Get length of the membrane (A)
  aauthor Michael Grabe.

- VEXTERNC double Vpbe_getmembraneDiel (Vpbe ∗thee)

  Get membrane dielectric constant.

- VEXTERNC double Vpbe_getmemv (Vpbe ∗thee)

  Get membrane potential (kT).

- VEXTERNC Vpbe ∗ Vpbe_ctor (Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

  Construct Vpbe object.

- VEXTERNC int Vpbe_ctor2 (Vpbe ∗thee, Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

  FORTRAN stub to construct Vpbe objct.

- VEXTERNC int Vpbe_getIons (Vpbe ∗thee, int ∗nion, double ionConc[MAXION], double ionRadii[MAXION], double ionQ[MAXION])

  Get information about the counterion species present.

- VEXTERNC void Vpbe_dtor (Vpbe ∗∗thee)

  Object destructor.

- VEXTERNC void Vpbe_dtor2 (Vpbe ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC double Vpbe_getCoulombEnergy1 (Vpbe ∗thee)

  Calculate coulombic energy of set of charges.

- VEXTERNC unsigned long int Vpbe_memChk (Vpbe ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

8.17.1 Detailed Description

The Poisson-Boltzmann master class. Contains objects and parameters used in every PBE calculation, regardless of method.

8.17.2 Function Documentation

8.17.2.1 VEXTERNC Vpbe$*$ Vpbe_ctor (Valist $*$ *alist*, int *ionNum*, double $*$ *ionConc*, double $*$ *ionRadii*, double $*$ *ionQ*, double *T*, double *soluteDiel*, double *solventDiel*, double *solventRadius*, int *focusFlag*, double *sdens*, double *z_mem*, double *L*, double *membraneDiel*, double *V*)

Construct Vpbe object.

Author

Nathan Baker and Mike Holst and Michael Grabe

Note

This is partially based on some of Mike Holst's PMG code. Here are a few of the original function comments: kappa is defined as follows:

$$\kappa^2 = \frac{8\pi N_A e_c^2 I_s}{1000 \epsilon_w k_B T}$$

where the units are esu$*$esu/erg/mol. To obtain $^{-2}$, we multiply by 10^{-16}. Thus, in $^{-2}$, where k_B and e_c are in gaussian rather than mks units, the proper value for kappa is:

$$\kappa^2 = \frac{8\pi N_A e_c^2 I_s}{1000 \epsilon_w k_b T} \times 10^{-16}$$

and the factor of 10^{-16} results from converting cm^2 to angstroms^2, noting that the 1000 in the denominator has converted m^3 to cm^3, since the ionic strength I_s is assumed to have been provided in moles per liter, which is moles per 1000 cm^3.

Returns

Pointer to newly allocated Vpbe object

Parameters

alist Atom list

ionNum Number of counterion species

ionConc Array containing counterion concentrations (M)

ionRadii Array containing counterion radii (A)

ionQ Array containing counterion charges (e)

T Temperature for Boltzmann distribution (K)

soluteDiel Solute internal dielectric constant

solventDiel Solvent dielectric constant

solventRadius Solvent probe radius for surfaces that use it (A)

focusFlag 1 if focusing operation, 0 otherwise

sdens Vacc sphere density

z_mem Membrane location (A)

L Membrane thickness (A)

membraneDiel Membrane dielectric constant

V Transmembrane potential (V)

8.17.2.2 VEXTERNC int Vpbe_ctor2 (Vpbe ∗ *thee*, Valist ∗ *alist*, int *ionNum*, double ∗ *ionConc*, double ∗ *ionRadii*, double ∗ *ionQ*, double *T*, double *soluteDiel*, double *solventDiel*, double *solventRadius*, int *focusFlag*, double *sdens*, double *z_mem*, double *L*, double *membraneDiel*, double *V*)

FORTRAN stub to construct Vpbe objct.

Author

Nathan Baker and Mike Holst and Michael Grabe

Note

This is partially based on some of Mike Holst's PMG code. Here are a few of the original function comments: kappa is defined as follows:

$$\kappa^2 = \frac{8\pi N_A e_c^2 I_s}{1000 eps_w k_B T}$$

where the units are esu∗esu/erg/mol. To obtain $^{-2}$, we multiply by 10^{-16}. Thus, in $^{-2}$, where k_B and e_c are in gaussian rather than mks units, the proper value for kappa is:

$$\kappa^2 = \frac{8 pi N_A e_c^2 I_s}{1000 eps_w k_b T} \times 10^{-16}$$

and the factor of 10^{-16} results from converting cm^2 to angstroms^2, noting that the 1000 in the denominator has converted m^3 to cm^3, since the ionic strength I_s is assumed to have been provided in moles per liter, which is moles per 1000 cm^3.

Bug

The focusing flag is currently not used!!

Returns

1 if successful, 0 otherwise

Parameters

thee Pointer to memory allocated for Vpbe object

alist Atom list

ionNum Number of counterion species

ionConc Array containing counterion concentrations (M)

ionRadii Array containing counterion radii (A)

ionQ Array containing counterion charges (e)

T Temperature for Boltzmann distribution (K)

soluteDiel Solute internal dielectric constant

solventDiel Solvent dielectric constant

solventRadius Solvent probe radius for surfaces that use it (A)

focusFlag 1 if focusing operation, 0 otherwise

sdens Vacc sphere density

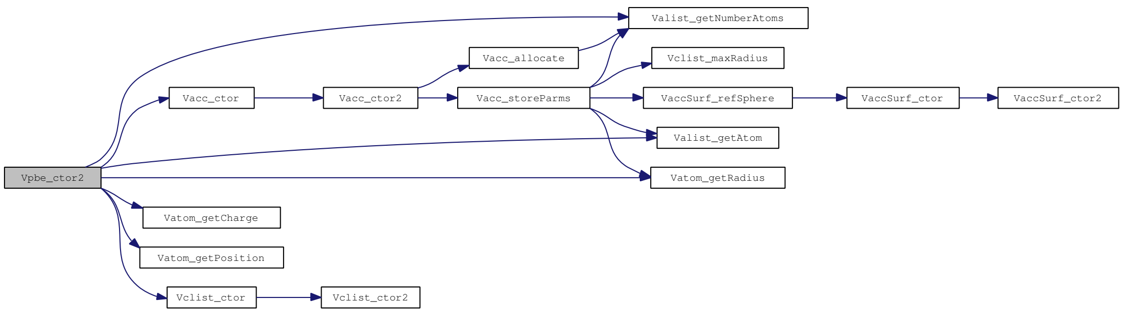
z_mem Membrane location (A)

L Membrane thickness (A)

membraneDiel Membrane dielectric constant

V Transmembrane potential (V)

Here is the call graph for this function:



8.17.2.3 VEXTERNC void Vpbe_dtor (Vpbe ** *thee*)

Object destructor.

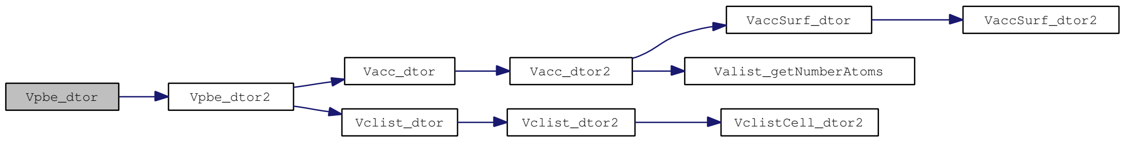
Author

Nathan Baker

Parameters

thee Pointer to memory location of object to be destroyed

Here is the call graph for this function:



8.17.2.4 VEXTERNC void Vpbe_dtor2 (Vpbe * *thee*)

FORTRAN stub object destructor.

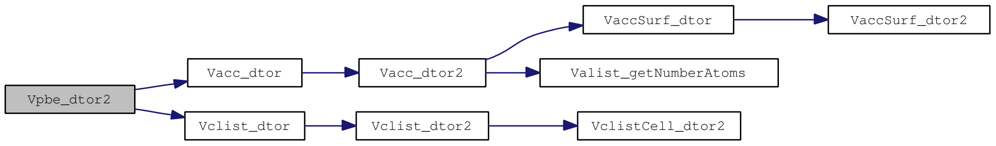
Author

Nathan Baker

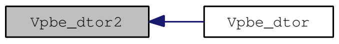
Parameters

thee Pointer to object to be destroyed

Here is the call graph for this function:



Here is the caller graph for this function:



8.17.2.5 VEXTERNC double Vpbe_getBulkIonicStrength (Vpbe ∗ *thee*)

Get bulk ionic strength.

Author

Nathan Baker

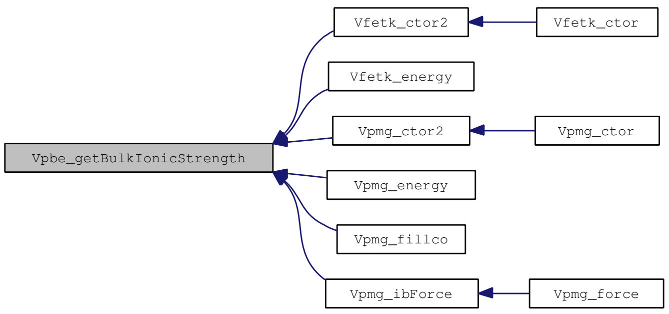
Parameters

thee Vpbe object

Returns

Bulk ionic strength (M)

Here is the caller graph for this function:



8.17.2.6 VEXTERNC double Vpbe_getCoulombEnergy1 (Vpbe ∗ *thee*)

Calculate coulombic energy of set of charges.

Perform an inefficient double sum to calculate the Coulombic energy of a set of charges in a homogeneous dielectric (with permittivity equal to the protein interior) and zero ionic strength. Result is returned in units of k_B T. The sum can be restriction to charges present in simplices of specified color (pcolor); if (color == -1) no restrictions are used.

Author

Nathan Baker

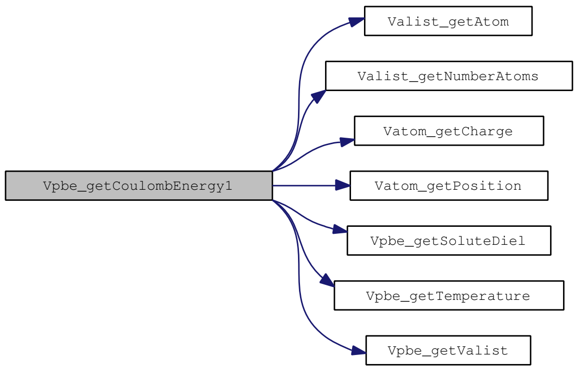
Parameters

thee Vpbe object

Returns

Coulombic energy in units of k_BT.

Here is the call graph for this function:



8.17.2.7 VEXTERNC double Vpbe_getDeblen (Vpbe ∗ *thee*)

Get Debye-Huckel screening length.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Debye-Huckel screening length (Å)

8.17.2.8 VEXTERNC double Vpbe_getGamma (Vpbe ∗ *thee*)

Get apolar coefficient.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Apolar coefficent (kJ/mol/A^2)

8.17.2.9 VEXTERNC int Vpbe_getIons (Vpbe ∗ *thee*, int ∗ *nion*, double *ionConc*[MAXION], double *ionRadii*[MAXION], double *ionQ*[MAXION])

Get information about the counterion species present.

Author

Nathan Baker

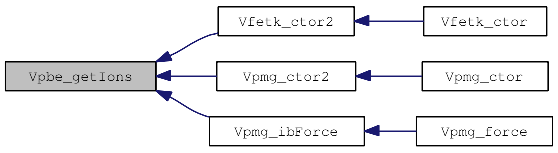
Parameters

thee Pointer to Vpbe object
nion Set to the number of counterion species
ionConc Array to store counterion species' concentrations (M)
ionRadii Array to store counterion species' radii (A)
ionQ Array to store counterion species' charges (e)

Returns

Number of ions

Here is the caller graph for this function:



8.17.2.10 VEXTERNC double Vpbe_getLmem (Vpbe ∗ *thee*)

Get length of the membrane (A)

aauthor Michael Grabe.

Parameters

thee Vpbe object

Returns

Length of the membrane (A)

8.17.2.11 VEXTERNC double Vpbe_getMaxIonRadius (Vpbe $*$ *thee*)

Get maximum radius of ion species.

Author

Nathan Baker

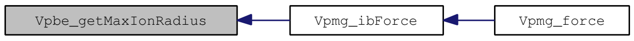
Parameters

thee Vpbe object

Returns

Maximum radius (A)

Here is the caller graph for this function:



8.17.2.12 VEXTERNC double Vpbe_getmembraneDiel (Vpbe $*$ *thee*)

Get membrane dielectric constant.

Author

Michael Grabe

Parameters

thee Vpbe object

Returns

Membrane dielectric constant

8.17.2.13 VEXTERNC double Vpbe_getmemv (Vpbe $*$ *thee*)

Get membrane potential (kT).

Author

Michael Grabe

Parameters

thee Vpbe object

8.17.2.14 VEXTERNC double∗ Vpbe_getSoluteCenter (Vpbe ∗ *thee*)

Get coordinates of solute center.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Pointer to 3∗double array with solute center coordinates (A)

8.17.2.15 VEXTERNC double Vpbe_getSoluteCharge (Vpbe ∗ *thee*)

Get total solute charge.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Total solute charge (e)

8.17.2.16 VEXTERNC double Vpbe_getSoluteDiel (Vpbe ∗ *thee*)

Get solute dielectric constant.

Author

Nathan Baker

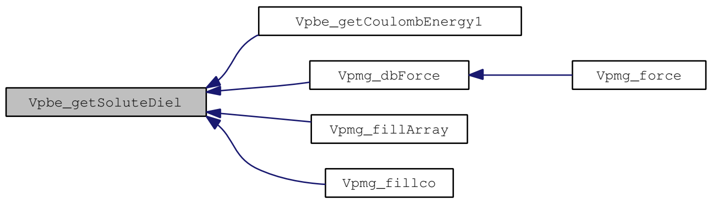
Parameters

thee Vpbe object

Returns

Solute dielectric constant

Here is the caller graph for this function:



8.17.2.17 VEXTERNC double Vpbe_getSoluteRadius (Vpbe ∗ *thee*)

Get sphere radius which bounds biomolecule.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Sphere radius which bounds biomolecule (A)

8.17.2.18 VEXTERNC double Vpbe_getSoluteXlen (Vpbe ∗ *thee*)

Get length of solute in x dimension.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Length of solute in x dimension (A)

8.17.2.19 VEXTERNC double Vpbe_getSoluteYlen (Vpbe ∗ *thee*)

Get length of solute in y dimension.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Length of solute in y dimension (A)

8.17.2.20 VEXTERNC double Vpbe_getSoluteZlen (Vpbe ∗ *thee*)

Get length of solute in z dimension.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Length of solute in z dimension (A)

8.17.2.21 VEXTERNC double Vpbe_getSolventDiel (Vpbe ∗ *thee*)

Get solvent dielectric constant.

Author

Nathan Baker

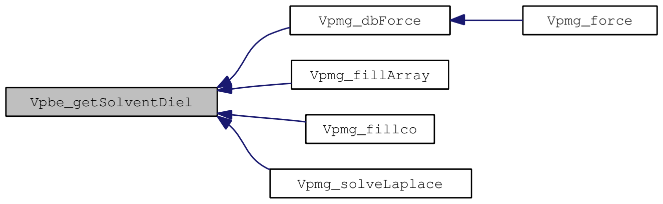
Parameters

thee Vpbe object

Returns

Solvent dielectric constant

Here is the caller graph for this function:



8.17.2.22 VEXTERNC double Vpbe_getSolventRadius (Vpbe * *thee*)

Get solvent molecule radius.

Author

Nathan Baker

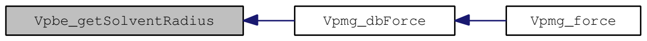
Parameters

thee Vpbe object

Returns

Solvent molecule radius (A)

Here is the caller graph for this function:



8.17.2.23 VEXTERNC double Vpbe_getTemperature (Vpbe * *thee*)

Get temperature.

Author

Nathan Baker

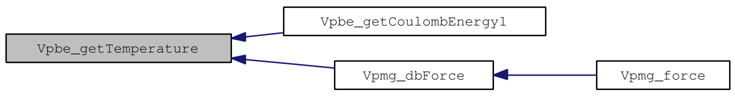
Parameters

thee Vpbe object

Returns

Temperature (K)

Here is the caller graph for this function:



8.17.2.24 VEXTERNC Vacc∗ Vpbe_getVacc (Vpbe ∗ *thee*)

Get accessibility oracle.

Author

Nathan Baker

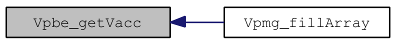
Parameters

thee Vpbe object

Returns

Pointer to internal Vacc object

Here is the caller graph for this function:



8.17.2.25 VEXTERNC Valist∗ Vpbe_getValist (Vpbe ∗ *thee*)

Get atom list.

Author

Nathan Baker

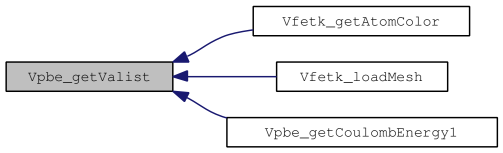
Parameters

thee Vpbe object

Returns

Pointer to internal Valist object

Here is the caller graph for this function:



8.17.2.26 VEXTERNC double Vpbe_getXkappa (Vpbe ∗ *thee*)

Get Debye-Huckel parameter.

Author

Nathan Baker

Parameters

thee Vpbe object

Returns

Bulk Debye-Huckel parameter (Å)

8.17.2.27 VEXTERNC double Vpbe_getZkappa2 (Vpbe ∗ *thee*)

Get modified squared Debye-Huckel parameter.

Author

Nathan Baker

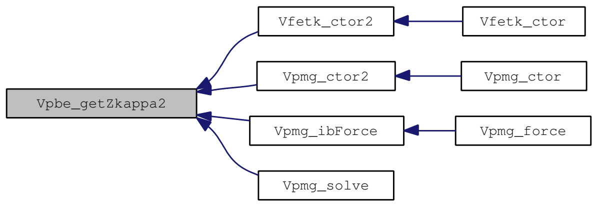
Parameters

thee Vpbe object

Returns

Modified squared Debye-Huckel parameter ($^{-2}$)

Here is the caller graph for this function:



8.17.2.28 VEXTERNC double Vpbe_getZmagic (Vpbe $*$ *thee*)

Get charge scaling factor.

Author

Nathan Baker and Mike Holst

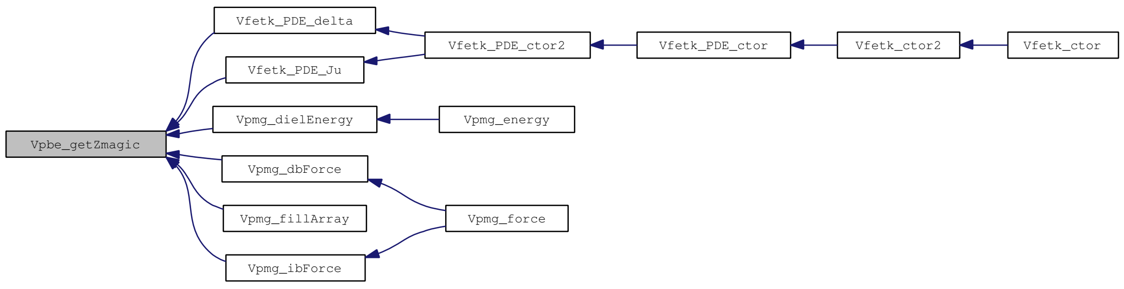
Parameters

thee Vpbe object

Returns

Get factor for scaling charges (in e) to internal units

Here is the caller graph for this function:



8.17.2.29 VEXTERNC double Vpbe_getzmem (Vpbe $*$ *thee*)

Get z position of the membrane bottom.

Author

Michael Grabe

Parameters

thee Vpbe object

Returns

z value of membrane (A)

8.17.2.30 VEXTERNC unsigned long int Vpbe_memChk (Vpbe $*$ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

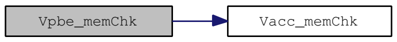
Parameters

thee Vpbe object

Returns

The memory used by this structure and its contents in bytes

Here is the call graph for this function:



8.18 Vstring class

Provides a collection of useful non-ANSI string functions.

Files

- file vstring.h

  Contains declarations for class Vstring.

Functions

- VEXTERNC int Vstring_strcasecmp (const char ∗s1, const char ∗s2)

  Case-insensitive string comparison (BSD standard).

- VEXTERNC int Vstring_isdigit (const char ∗tok)

  A modified sscanf that examines the complete string.

8.18.1 Detailed Description

Provides a collection of useful non-ANSI string functions.

8.18.2 Function Documentation

8.18.2.1 VEXTERNC int Vstring_isdigit (const char ∗ *tok*)

A modified sscanf that examines the complete string.

Author

Todd Dolinsky

Parameters

tok The string to examine

Returns

1 if the entire string is an integer, 0 if otherwise.

8.18.2.2 VEXTERNC int Vstring_strcasecmp (const char ∗ *s1*, const char ∗ *s2*)

Case-insensitive string comparison (BSD standard).

Author

Copyright (c) 1988-1993 The Regents of the University of California. Copyright (c) 1995-1996 Sun Microsystems, Inc.

Note

Copyright (c) 1988-1993 The Regents of the University of California. Copyright (c) 1995-1996 Sun Microsystems, Inc.

Parameters

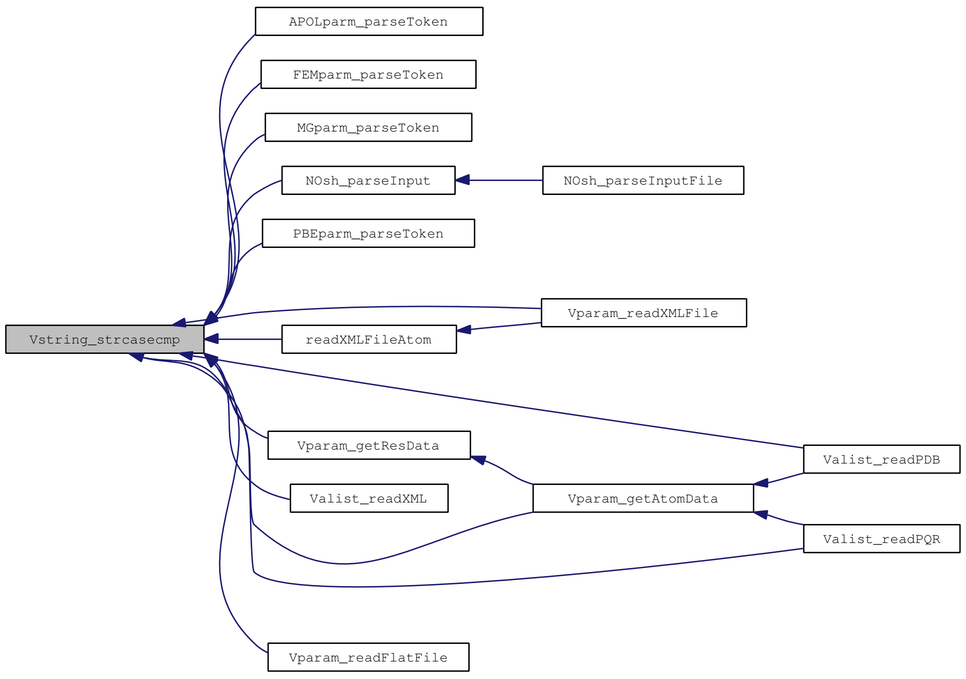
s1 First string for comparison

s2 Second string for comparison

Returns

An integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2. (Source: Linux man pages)

Here is the caller graph for this function:



8.19 Vunit class

Collection of constants and conversion factors.

Files

- file vunit.h

  Contains a collection of useful constants and conversion factors.

Defines

- #define Vunit_J_to_cal 4.1840000e+00

  Multiply by this to convert J to cal.

- #define Vunit_cal_to_J 2.3900574e-01

  Multiply by this to convert cal to J.

- #define Vunit_amu_to_kg 1.6605402e-27

  Multiply by this to convert amu to kg.

- #define Vunit_kg_to_amu 6.0221367e+26

  Multiply by this to convert kg to amu.

- #define Vunit_ec_to_C 1.6021773e-19

  Multiply by this to convert ec to C.

- #define Vunit_C_to_ec 6.2415065e+18

  Multiply by this to convert C to ec.

- #define Vunit_ec 1.6021773e-19

  Charge of an electron in C.

- #define Vunit_kb 1.3806581e-23

  Boltzmann constant.

- #define Vunit_Na 6.0221367e+23

  Avogadro's number.

- #define Vunit_pi VPI

  Pi.

- #define Vunit_eps0 8.8541878e-12

  Vacuum permittivity.

- #define Vunit_esu_ec2A 3.3206364e+02

  ${e_c}^2/$ *in ESU units => kcal/mol*

- #define Vunit_esu_kb 1.9871913e-03

  k_b *in ESU units => kcal/mol*

8.19.1 Detailed Description

Collection of constants and conversion factors.

8.20 Vgrid class

Oracle for Cartesian mesh data.

Data Structures

- struct sVgrid

  Electrostatic potential oracle for Cartesian mesh data.

Files

- file vgrid.h

  Potential oracle for Cartesian mesh data.

- file vgrid.c

  Class Vgrid methods.

Defines

- #define VGRID_DIGITS 6

  Number of decimal places for comparisons and formatting.

Typedefs

- typedef struct sVgrid Vgrid

  Declaration of the Vgrid class as the sVgrid structure.

Functions

- VEXTERNC unsigned long int Vgrid_memChk (Vgrid ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vgrid ∗ Vgrid_ctor (int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Construct Vgrid object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vgrid_ctor2 (Vgrid ∗thee, int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Initialize Vgrid object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vgrid_value (Vgrid ∗thee, double x[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vgrid_dtor (Vgrid ∗∗thee)

  Object destructor.

- VEXTERNC void Vgrid_dtor2 (Vgrid ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vgrid_curvature (Vgrid ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vgrid_gradient (Vgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VEXTERNC void Vgrid_writeUHBD (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in UHBD grid format.

- VEXTERNC void Vgrid_writeDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in OpenDX grid format.

- VEXTERNC int Vgrid_readDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read in data in OpenDX grid format.

- VEXTERNC double Vgrid_integrate (Vgrid ∗thee)

  Get the integral of the data.

- VEXTERNC double Vgrid_normL1 (Vgrid ∗thee)

  Get the L_1 norm of the data. This returns the integral:

  $$\|u\|_{L_1} = \int_\Omega |u(x)| dx$$

  .

- VEXTERNC double Vgrid_normL2 (Vgrid ∗thee)

Get the L_2 norm of the data. This returns the integral:

$$\|u\|_{L_2} = \left(\int_\Omega |u(x)|^2 dx \right)^{1/2}$$

.

- VEXTERNC double Vgrid_normLinf (Vgrid ∗thee)

  Get the L_∞ norm of the data. This returns the integral:

  $$\|u\|_{L_\infty} = \sup_{x \in \Omega} |u(x)|$$

  .

- VEXTERNC double Vgrid_seminormH1 (Vgrid ∗thee)

  Get the H_1 semi-norm of the data. This returns the integral:

  $$|u|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx \right)^{1/2}$$

  .

- VEXTERNC double Vgrid_normH1 (Vgrid ∗thee)

  Get the H_1 norm (or energy norm) of the data. This returns the integral:

  $$\|u\|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx + \int_\Omega |u(x)|^2 dx \right)^{1/2}$$

  .

8.20.1 Detailed Description

Oracle for Cartesian mesh data.

8.20.2 Function Documentation

8.20.2.1 VEXTERNC Vgrid∗ Vgrid_ctor (int *nx*, int *ny*, int *nz*, double *hx*, double *hy*, double *hzed*, double *xmin*, double *ymin*, double *zmin*, double ∗ *data*)

Construct Vgrid object with values obtained from Vpmg_readDX (for example).

Author

Nathan Baker

Parameters

nx Number grid points in x direction

ny Number grid points in y direction

nz Number grid points in z direction

hx Grid spacing in x direction

hy Grid spacing in y direction

hzed Grid spacing in z direction

xmin x coordinate of lower grid corner

ymin y coordinate of lower grid corner

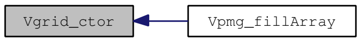
zmin z coordinate of lower grid corner

data nx∗ny∗nz array of data. This can be VNULL if you are planning to read in data later with one of the read routines

Returns

Newly allocated and initialized Vgrid object

Here is the caller graph for this function:



8.20.2.2 VEXTERNC int Vgrid_ctor2 (Vgrid ∗ *thee*, int *nx*, int *ny*, int *nz*, double *hx*, double *hy*, double *hzed*, double *xmin*, double *ymin*, double *zmin*, double ∗ *data*)

Initialize Vgrid object with values obtained from Vpmg_readDX (for example).

Author

Nathan Baker

Parameters

thee Pointer to newly allocated Vgrid object

nx Number grid points in x direction

ny Number grid points in y direction

nz Number grid points in z direction

hx Grid spacing in x direction

hy Grid spacing in y direction

hzed Grid spacing in z direction

xmin x coordinate of lower grid corner

ymin y coordinate of lower grid corner

zmin z coordinate of lower grid corner

data nx∗ny∗nz array of data. This can be VNULL if you are planning to read in data later with one of the read routines

Returns

Newly allocated and initialized Vgrid object

8.20.2.3 VEXTERNC int Vgrid_curvature (Vgrid ∗ *thee*, double *pt*[3], int *cflag*, double ∗ *curv*)

Get second derivative values at a point.

Author

Steve Bond and Nathan Baker

Parameters

thee Pointer to Vgrid object

pt Location to evaluate second derivative

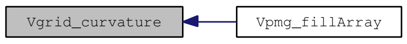
cflag
- 0: Reduced Maximal Curvature
- 1: Mean Curvature (Laplace)
- 2: Gauss Curvature
- 3: True Maximal Curvature

curv Specified curvature value

Returns

1 if successful, 0 if off grid

Here is the caller graph for this function:



8.20.2.4 VEXTERNC void Vgrid_dtor (Vgrid ∗∗ *thee*)

Object destructor.

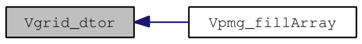
Author

Nathan Baker

Parameters

thee Pointer to memory location of object to be destroyed

Here is the caller graph for this function:



8.20.2.5 VEXTERNC void Vgrid_dtor2 (Vgrid ∗ *thee*)

FORTRAN stub object destructor.

Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

8.20.2.6 VEXTERNC int Vgrid_gradient (Vgrid ∗ *thee*, double *pt*[3], double *grad*[3])

Get first derivative values at a point.

Author

Nathan Baker and Steve Bond

Parameters

thee Pointer to Vgrid object

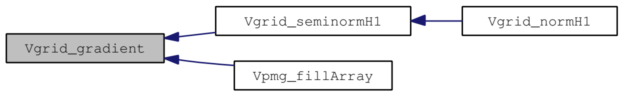
pt Location to evaluate gradient

grad Gradient

Returns

1 if successful, 0 if off grid

Here is the caller graph for this function:



8.20.2.7 VEXTERNC double Vgrid_integrate (Vgrid * *thee*)

Get the integral of the data.

Author

Nathan Baker

Parameters

thee Vgrid object

Returns

Integral of data

8.20.2.8 VEXTERNC unsigned long int Vgrid_memChk (Vgrid * *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

Parameters

thee Vgrid object

Returns

The memory used by this structure and its contents in bytes

8.20.2.9 VEXTERNC double Vgrid_normH1 (Vgrid * *thee*)

Get the H_1 norm (or energy norm) of the data. This returns the integral:

$$\|u\|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx + \int_\Omega |u(x)|^2 dx \right)^{1/2}$$

.

Author

Nathan Baker

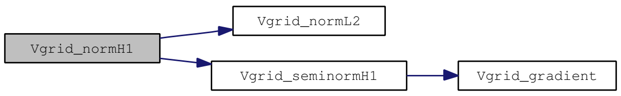
Parameters

thee Vgrid object

Returns

Integral of data

Here is the call graph for this function:



8.20.2.10 VEXTERNC double Vgrid_normL1 (Vgrid $*$ *thee*)

Get the L_1 norm of the data. This returns the integral:

$$\|u\|_{L_1} = \int_\Omega |u(x)| dx$$

.

Author

Nathan Baker

Parameters

thee Vgrid object

Returns

L_1 norm of data

8.20.2.11 VEXTERNC double Vgrid_normL2 (Vgrid $*$ *thee*)

Get the L_2 norm of the data. This returns the integral:

$$\|u\|_{L_2} = \left(\int_\Omega |u(x)|^2 dx \right)^{1/2}$$

.

Author

Nathan Baker

Parameters

thee Vgrid object

Returns

L_2 norm of data

Here is the caller graph for this function:

Vgrid_normL2
Vgrid_normH1

8.20.2.12 VEXTERNC double Vgrid_normLinf (Vgrid $*$ *thee*)

Get the L_∞ norm of the data. This returns the integral:

$$\|u\|_{L_\infty} = \sup_{x \in \Omega} |u(x)|$$

.

Author

Nathan Baker

Parameters

thee Vgrid object

Returns

$L\infty$ norm of data

8.20.2.13 VEXTERNC int Vgrid_readDX (Vgrid $*$ *thee*, const char $*$ *iodev*, const char $*$ *iofmt*, const char $*$ *thost*, const char $*$ *fname*)

Read in data in OpenDX grid format.

Note

All dimension information is given in order: z, y, x

Author

Nathan Baker

Parameters

thee Vgrid object

iodev Input device type (FILE/BUFF/UNIX/INET)

iofmt Input device format (ASCII/XDR)

thost Input hostname (for sockets)

fname Input FILE/BUFF/UNIX/INET name

Returns

1 if sucessful, 0 otherwise

8.20.2.14 VEXTERNC double Vgrid_seminormH1 (Vgrid * *thee*)

Get the H_1 semi-norm of the data. This returns the integral:

$$|u|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx \right)^{1/2}$$

.

Author

Nathan Baker

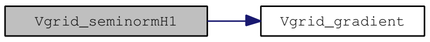
Parameters

thee Vgrid object

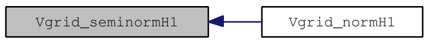
Returns

Integral of data

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.15 VEXTERNC int Vgrid_value (Vgrid * *thee*, double *x*[3], double * *value*)

Get potential value (from mesh or approximation) at a point.

Author

Nathan Baker

Parameters

thee Vgrid obejct

x Point at which to evaluate potential

value Value of data at point x

Returns

1 if successful, 0 if off grid

8.20.2.16 VEXTERNC void Vgrid_writeDX (Vgrid ∗ *thee*, const char ∗ *iodev*, const char ∗ *iofmt*, const char ∗ *thost*, const char ∗ *fname*, char ∗ *title*, double ∗ *pvec*)

Write out the data in OpenDX grid format.

Author

Nathan Baker

Parameters

thee Grid object

iodev Output device type (FILE/BUFF/UNIX/INET)

iofmt Output device format (ASCII/XDR)

thost Output hostname (for sockets)

fname Output FILE/BUFF/UNIX/INET name

title Title to be inserted in grid file

pvec Partition weight (if 1: point in current partition, if 0 point not in current partition if $>$ 0 && $<$ 1 point on/near boundary)

8.20.2.17 VEXTERNC void Vgrid_writeUHBD (Vgrid ∗ *thee*, const char ∗ *iodev*, const char ∗ *iofmt*, const char ∗ *thost*, const char ∗ *fname*, char ∗ *title*, double ∗ *pvec*)

Write out the data in UHBD grid format.

Note

- The mesh spacing should be uniform

- Format changed from 12.6E to 12.5E

Author

Nathan Baker

Parameters

thee Grid object

iodev Output device type (FILE/BUFF/UNIX/INET)

iofmt Output device format (ASCII/XDR)

thost Output hostname (for sockets)

fname Output FILE/BUFF/UNIX/INET name

title Title to be inserted in grid file

pvec Partition weight (if 1: point in current partition, if 0 point not in current partition if $>$ 0 && $<$ 1 point on/near boundary)

Bug

This routine does not respect partition information

8.21 Vmgrid class

Oracle for Cartesian mesh data.

Data Structures

- struct sVmgrid

  Multiresoltion oracle for Cartesian mesh data.

Files

- file vmgrid.h

  Multiresolution oracle for Cartesian mesh data.

- file vmgrid.c

  Class Vmgrid methods.

Defines

- #define VMGRIDMAX 20

  The maximum number of levels in the grid hiearchy.

Typedefs

- typedef struct sVmgrid Vmgrid

  Declaration of the Vmgrid class as the Vgmrid structure.

Functions

- VEXTERNC Vmgrid ∗ Vmgrid_ctor ()

  Construct Vmgrid object.

- VEXTERNC int Vmgrid_ctor2 (Vmgrid ∗thee)

  Initialize Vmgrid object.

- VEXTERNC int Vmgrid_value (Vmgrid ∗thee, double x[3], double ∗value)

Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vmgrid_dtor (Vmgrid ∗∗thee)

  Object destructor.

- VEXTERNC void Vmgrid_dtor2 (Vmgrid ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vmgrid_addGrid (Vmgrid ∗thee, Vgrid ∗grid)

  Add a grid to the hierarchy.

- VEXTERNC int Vmgrid_curvature (Vmgrid ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vmgrid_gradient (Vmgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VEXTERNC Vgrid ∗ Vmgrid_getGridByNum (Vmgrid ∗thee, int num)

  Get specific grid in hiearchy.

- VEXTERNC Vgrid ∗ Vmgrid_getGridByPoint (Vmgrid ∗thee, double pt[3])

  Get grid in hiearchy which contains specified point or VNULL.

8.21.1 Detailed Description

Oracle for Cartesian mesh data.

8.21.2 Function Documentation

8.21.2.1 VEXTERNC int Vmgrid_addGrid (Vmgrid ∗ *thee*, Vgrid ∗ *grid*)

Add a grid to the hierarchy.

Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

grid Grid to be added. As mentioned above, we would prefer to have the finest grid added first, next-finest second, ..., coarsest last -- this is how the grid will be searched when looking up values for points. However, this is not enforced to provide flexibility for cases where the dataset is decomposed into disjoint partitions, etc.

Returns

1 if successful, 0 otherwise

8.21.2.2 VEXTERNC Vmgrid∗ Vmgrid_ctor ()

Construct Vmgrid object.

Author

Nathan Baker

Returns

Newly allocated and initialized Vmgrid object

8.21.2.3 VEXTERNC int Vmgrid_ctor2 (Vmgrid ∗ *thee*)

Initialize Vmgrid object.

Author

Nathan Baker

Parameters

thee Newly allocated Vmgrid object

Returns

Newly allocated and initialized Vmgrid object

8.21.2.4 VEXTERNC int Vmgrid_curvature (Vmgrid ∗ *thee*, double *pt*[3], int *cflag*, double ∗ *curv*)

Get second derivative values at a point.

Author

Nathan Baker (wrapper for Vgrid routine by Steve Bond)

Parameters

thee Pointer to Vmgrid object

pt Location to evaluate second derivative

cflag
- 0: Reduced Maximal Curvature
- 1: Mean Curvature (Laplace)
- 2: Gauss Curvature
- 3: True Maximal Curvature

curv Specified curvature value

Returns

1 if successful, 0 if off grid

8.21.2.5 VEXTERNC void Vmgrid_dtor (Vmgrid ** *thee*)

Object destructor.

Author

Nathan Baker

Parameters

thee Pointer to memory location of object to be destroyed

8.21.2.6 VEXTERNC void Vmgrid_dtor2 (Vmgrid * *thee*)

FORTRAN stub object destructor.

Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

8.21.2.7 VEXTERNC Vgrid* Vmgrid_getGridByNum (Vmgrid * *thee*, int *num*)

Get specific grid in hiearchy.

Author

Nathan Baker

Parameters

thee Pointer to Vmgrid object
num Number of grid in hiearchy

Returns

Pointer to specified grid

8.21.2.8 VEXTERNC Vgrid$*$ Vmgrid_getGridByPoint (Vmgrid $*$ *thee*, double *pt*[3])

Get grid in hiearchy which contains specified point or VNULL.

Author

Nathan Baker

Parameters

thee Pointer to Vmgrid object
pt Point to check

Returns

Pointer to specified grid

8.21.2.9 VEXTERNC int Vmgrid_gradient (Vmgrid $*$ *thee*, double *pt*[3], double *grad*[3])

Get first derivative values at a point.

Author

Nathan Baker and Steve Bond

Parameters

thee Pointer to Vmgrid object
pt Location to evaluate gradient
grad Gradient

Returns

1 if successful, 0 if off grid

8.21.2.10 VEXTERNC int Vmgrid_value (Vmgrid $*$ *thee*, double *x*[3], double $*$ *value*)

Get potential value (from mesh or approximation) at a point.

Author

Nathan Baker

Parameters

thee Vmgrid obejct

x Point at which to evaluate potential

value Value of data at point x

Returns

1 if successful, 0 if off grid

8.22 Vopot class

Potential oracle for Cartesian mesh data.

Data Structures

- struct sVopot

  Electrostatic potential oracle for Cartesian mesh data.

Files

- file vopot.h

  Potential oracle for Cartesian mesh data.

- file vopot.c

  Class Vopot methods.

Typedefs

- typedef struct sVopot Vopot

  Declaration of the Vopot class as the Vopot structure.

Functions

- VEXTERNC Vopot ∗ Vopot_ctor (Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Construct Vopot object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vopot_ctor2 (Vopot ∗thee, Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Initialize Vopot object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vopot_pot (Vopot ∗thee, double x[3], double ∗pot)

  Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vopot_dtor (Vopot ∗∗thee)

  Object destructor.

- VEXTERNC void Vopot_dtor2 (Vopot ∗thee)

FORTRAN stub object destructor.

- VEXTERNC int Vopot_curvature (Vopot ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vopot_gradient (Vopot ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

8.22.1 Detailed Description

Potential oracle for Cartesian mesh data.

8.22.2 Function Documentation

8.22.2.1 VEXTERNC Vopot∗ Vopot_ctor (Vmgrid ∗ *mgrid*, Vpbe ∗ *pbe*, Vbcfl *bcfl*)

Construct Vopot object with values obtained from Vpmg_readDX (for example).

Author

Nathan Baker

Parameters

mgrid Multiple grid object containing potential data (in units kT/e)

pbe Pointer to Vpbe object for parameters

bcfl Boundary condition to use for potential values off the grid

Returns

Newly allocated and initialized Vopot object

8.22.2.2 VEXTERNC int Vopot_ctor2 (Vopot ∗ *thee*, Vmgrid ∗ *mgrid*, Vpbe ∗ *pbe*, Vbcfl *bcfl*)

Initialize Vopot object with values obtained from Vpmg_readDX (for example).

Author

Nathan Baker

Parameters

thee Pointer to newly allocated Vopot object

mgrid Multiple grid object containing potential data (in units kT/e)

pbe Pointer to Vpbe object for parameters

bcfl Boundary condition to use for potential values off the grid

Returns

1 if successful, 0 otherwise

8.22.2.3 VEXTERNC int Vopot_curvature (Vopot ∗ *thee*, double *pt*[3], int *cflag*, double ∗ *curv*)

Get second derivative values at a point.

Author

Nathan Baker

Parameters

thee Pointer to Vopot object

pt Location to evaluate second derivative

cflag
- 0: Reduced Maximal Curvature
- 1: Mean Curvature (Laplace)
- 2: Gauss Curvature
- 3: True Maximal Curvature

curv Set to specified curvature value

Returns

1 if successful, 0 otherwise

8.22.2.4 VEXTERNC void Vopot_dtor (Vopot ∗∗ *thee*)

Object destructor.

Author

Nathan Baker

Parameters

thee Pointer to memory location of object to be destroyed

8.22.2.5 VEXTERNC void Vopot_dtor2 (Vopot $*$ *thee*)

FORTRAN stub object destructor.

Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

8.22.2.6 VEXTERNC int Vopot_gradient (Vopot $*$ *thee*, double *pt*[3], double *grad*[3])

Get first derivative values at a point.

Author

Nathan Baker

Parameters

thee Pointer to Vopot object
pt Location to evaluate gradient
grad Gradient

Returns

1 if successful, 0 otherwise

8.22.2.7 VEXTERNC int Vopot_pot (Vopot $*$ *thee*, double *x*[3], double $*$ *pot*)

Get potential value (from mesh or approximation) at a point.

Author

Nathan Baker

Parameters

thee Vopot obejct
x Point at which to evaluate potential
pot Set to dimensionless potential (units kT/e) at point x

Returns

1 if successful, 0 otherwise

8.23 Vpmg class

A wrapper for Mike Holst's PMG multigrid code.

Data Structures

- struct sVpmg

  Contains public data members for Vpmg class/module.

Files

- file vpmg.h

  Contains declarations for class Vpmg.

- file vpmg.c

  Class Vpmg methods.

Typedefs

- typedef struct sVpmg Vpmg

  Declaration of the Vpmg class as the Vpmg structure.

Functions

- VEXTERNC unsigned long int Vpmg_memChk (Vpmg ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vpmg ∗ Vpmg_ctor (Vpmgp ∗parms, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

  Constructor for the Vpmg class (allocates new memory).

- VEXTERNC int Vpmg_ctor2 (Vpmg ∗thee, Vpmgp ∗parms, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

  FORTRAN stub constructor for the Vpmg class (uses previously-allocated memory).

- VEXTERNC void Vpmg_dtor (Vpmg ∗∗thee)

  Object destructor.

- VEXTERNC void Vpmg_dtor2 (Vpmg ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vpmg_fillco (Vpmg ∗thee, Vsurf_Meth surfMeth, double splineWin, Vchrg_Meth chargeMeth, int useDielXMap, Vgrid ∗dielXMap, int useDielYMap, Vgrid ∗dielYMap, int useDielZMap, Vgrid ∗dielZMap, int useKappaMap, Vgrid ∗kappaMap, int useChargeMap, Vgrid ∗chargeMap)

  Fill the coefficient arrays prior to solving the equation.

- VEXTERNC int Vpmg_solve (Vpmg ∗thee)

  Solve the PBE using PMG.

- VEXTERNC int Vpmg_solveLaplace (Vpmg ∗thee)

  Solve Poisson's equation with a homogeneous Laplacian operator using the solvent dielectric constant. This solution is performed by a sine wave decomposition.

- VEXTERNC double Vpmg_energy (Vpmg ∗thee, int extFlag)

  Get the total electrostatic energy.

- VEXTERNC double Vpmg_qfEnergy (Vpmg ∗thee, int extFlag)

  Get the "fixed charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_qfAtomEnergy (Vpmg ∗thee, Vatom ∗atom)

  Get the per-atom "fixed charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_qmEnergy (Vpmg ∗thee, int extFlag)

  Get the "mobile charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_dielEnergy (Vpmg ∗thee, int extFlag)

  Get the "polarization" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_dielGradNorm (Vpmg ∗thee)

  Get the integral of the gradient of the dielectric function.

- VEXTERNC int Vpmg_force (Vpmg ∗thee, double ∗force, int atomID, Vsurf_-Meth srfm, Vchrg_Meth chgm)

  Calculate the total force on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_qfForce (Vpmg ∗thee, double ∗force, int atomID, Vchrg_Meth chgm)

  Calculate the "charge-field" force on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_dbForce (Vpmg ∗thee, double ∗dbForce, int atomID, Vsurf_Meth srfm)

  Calculate the dielectric boundary forces on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_ibForce (Vpmg ∗thee, double ∗force, int atomID, Vsurf_Meth srfm)

  Calculate the osmotic pressure on the specified atom in units of k_B T/AA.

- VEXTERNC void Vpmg_setPart (Vpmg ∗thee, double lowerCorner[3], double upperCorner[3], int bflags[6])

  Set partition information which restricts the calculation of observables to a (rectangular) subset of the problem domain.

- VEXTERNC void Vpmg_unsetPart (Vpmg ∗thee)

  Remove partition restrictions.

- VEXTERNC int Vpmg_fillArray (Vpmg ∗thee, double ∗vec, Vdata_Type type, double parm, Vhal_PBEType pbetype)

  Fill the specified array with accessibility values.

- VPUBLIC void Vpmg_fieldSpline4 (Vpmg ∗thee, int atomID, double field[3])

  Computes the field at an atomic center using a stencil based on the first derivative of a 5th order B-spline.

- VEXTERNC double Vpmg_qfPermanentMultipoleEnergy (Vpmg ∗thee, int atomID)

  Computes the permanent multipole electrostatic hydration energy (the polarization component of the hydration energy currently computed in TINKER).

- VEXTERNC void Vpmg_qfPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3], double torque[3])

  Computes the q-Phi Force for permanent multipoles based on 5th order B-splines.

- VEXTERNC void Vpmg_ibPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3])

  Compute the ionic boundary force for permanent multipoles.

- VEXTERNC void Vpmg_dbPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3])

  Compute the dielectric boundary force for permanent multipoles.

- VEXTERNC void Vpmg_qfDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3], double torque[3]) *q-Phi direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.*

- VEXTERNC void Vpmg_qfNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3], double torque[3])

  q-Phi direct polarization force between permanent multipoles and non-local induced dipoles based on 5th Order B-Splines. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_ibDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3])

  Ionic boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

- VEXTERNC void Vpmg_ibNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3])

  Ionic boundary direct polarization force between permanent multipoles and non-local induced dipoles based on 5th order Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_dbDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3])

  Dielectric boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

- VEXTERNC void Vpmg_dbNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3])

  Dielectric bounday direct polarization force between permanent multipoles and non-local induced dipoles. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_qfMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

  Mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_ibMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

Ionic boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_dbMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

  Dielectric boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_printColComp (Vpmg ∗thee, char path[72], char title[72], char mxtype[3], int flag)

  Print out a column-compressed sparse matrix in Harwell-Boeing format.

8.23.1 Detailed Description

A wrapper for Mike Holst's PMG multigrid code.

Note

Many of the routines and macros are borrowed from the main.c driver (written by Mike Holst) provided with the PMG code.

8.23.2 Function Documentation

8.23.2.1 VEXTERNC Vpmg∗ Vpmg_ctor (Vpmgp ∗ *parms*, Vpbe ∗ *pbe*, int *focusFlag*, Vpmg ∗ *pmgOLD*, MGparm ∗ *mgparm*, PBEparm_calcEnergy *energyFlag*)

Constructor for the Vpmg class (allocates new memory).

Author

Nathan Baker

Returns

Pointer to newly allocated Vpmg object

Parameters

parms PMG parameter object

pbe PBE-specific variables

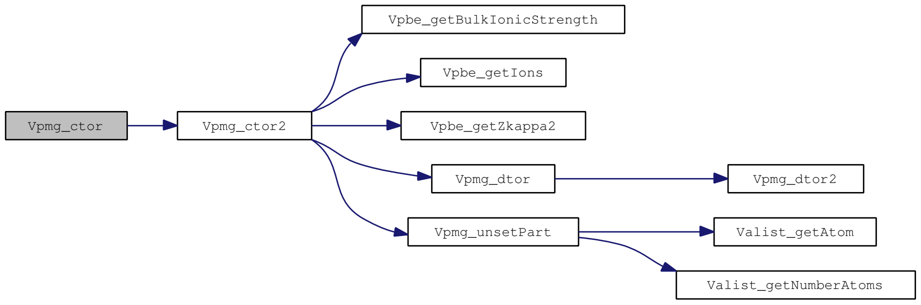
focusFlag 1 for focusing, 0 otherwise

pmgOLD Old Vpmg object to use for boundary conditions

mgparm MGparm parameter object for boundary conditions

energyFlag What types of energies to calculate

Here is the call graph for this function:



8.23.2.2 VEXTERNC int Vpmg_ctor2 (Vpmg $*$ *thee*, Vpmgp $*$ *parms*, Vpbe $*$ *pbe*, int *focusFlag*, Vpmg $*$ *pmgOLD*, MGparm $*$ *mgparm*, PBEparm_calcEnergy *energyFlag*)

FORTRAN stub constructor for the Vpmg class (uses previously-allocated memory).

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Parameters

thee Memory location for object

parms PMG parameter object

pbe PBE-specific variables

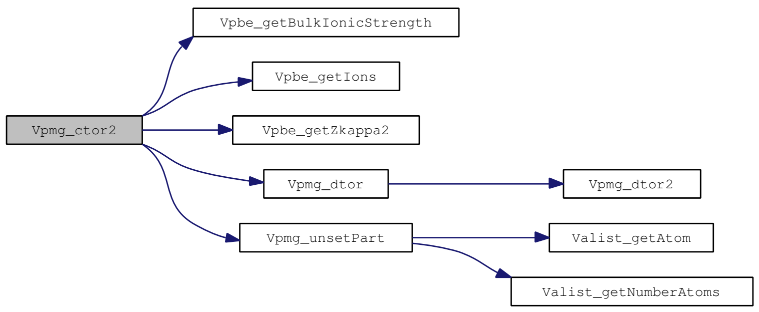
focusFlag 1 for focusing, 0 otherwise

pmgOLD Old Vpmg object to use for boundary conditions (can be VNULL if focusFlag = 0)

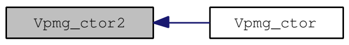
mgparm MGparm parameter object for boundary conditions (can be VNULL if focusFlag = 0)

energyFlag What types of energies to calculate (ignored if focusFlag = 0)

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.2.3 VEXTERNC void Vpmg_dbDirectPolForce (Vpmg ∗ *thee*, Vgrid ∗ *perm*, Vgrid ∗ *induced*, int *atomID*, double *force*[3])

Dielectric boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

induced Induced dipole potential

atomID Atom index

force (returned) force

8.23.2.4 VEXTERNC int Vpmg_dbForce (Vpmg ∗ *thee*, double ∗ *dbForce*, int *atomID*, Vsurf_Meth *srfm*)

Calculate the dielectric boundary forces on the specified atom in units of k_B T/AA.

Author

Nathan Baker

Note

- Using the force evaluation methods of Im et al (Roux group), Comput Phys Commun, 111, 59--75 (1998). However, this gives the whole (self-interactions included) force -- reaction field forces will have to be calculated at higher level.
- No contributions are made from higher levels of focusing.

Returns

1 if successful, 0 otherwise

Parameters

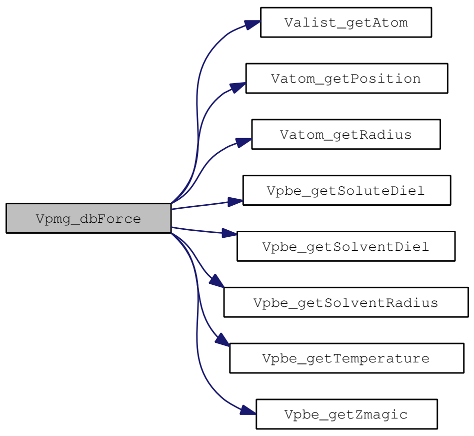
thee Vpmg object

dbForce 3∗sizeof(double) space to hold the dielectric boundary force in units of k_B T/AA

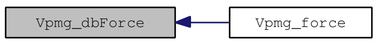
atomID Valist ID of desired atom

srfm Surface discretization method

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.2.5 VEXTERNC void Vpmg_dbMutualPolForce (Vpmg ∗ *thee*, Vgrid ∗ *induced*, Vgrid ∗ *nlInduced*, int *atomID*, double *force*[3])

Dielectric boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

induced Induced dipole potential

nlInduced Non-local induced dipole potential

atomID Atom index

force (returned) force

8.23.2.6 VEXTERNC void Vpmg_dbNLDirectPolForce (Vpmg ∗ *thee*, Vgrid ∗ *perm*, Vgrid ∗ *nlInduced*, int *atomID*, double *force*[3])

Dielectric bounday direct polarization force between permanent multipoles and non-local induced dipoles. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

nlInduced Non-local induced dipole potential

atomID Atom index

force (returned) force

8.23.2.7 VEXTERNC void Vpmg_dbPermanentMultipoleForce (Vpmg $*$ *thee*, int *atomID*, double *force*[3])

Compute the dielectric boundary force for permanent multipoles.

Author

Michael Schnieders

Parameters

thee Vpmg object

atomID Atom index

force (returned) force

8.23.2.8 VEXTERNC double Vpmg_dielEnergy (Vpmg $*$ *thee*, int *extFlag*)

Get the "polarization" contribution to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the mobile charges with the potential:

$$G = \frac{1}{2} \int \epsilon (\nabla u)^2 dx$$

where epsilon is the dielectric parameter and u(x) is the dimensionless electrostatic potential. The energy is scaled to units of k_b T.

Author

Nathan Baker

Note

The value of this observable may be modified by setting restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

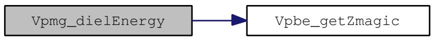
The polarization electrostatic energy in units of k_B T.

Parameters

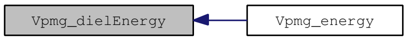
thee Vpmg object

extFlag If this was a focused calculation, include (1 -- for serial calculations) or ignore (0 -- for parallel calculations) energy contributions from outside the focusing domain

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.2.9 VEXTERNC double Vpmg_dielGradNorm (Vpmg $*$ *thee*)

Get the integral of the gradient of the dielectric function.

Using the dielectric map at the finest mesh level, calculate the integral of the norm of the dielectric function gradient routines of Im et al (see Vpmg_dbForce for reference):

$$\int \|\nabla\epsilon\| dx$$

where epsilon is the dielectric parameter. The integral is returned in units of A^2.

Author

Nathan Baker restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

The integral in units of A^2.

Parameters

thee Vpmg object

8.23.2.10 VEXTERNC void Vpmg_dtor (Vpmg $**$ *thee*)

Object destructor.

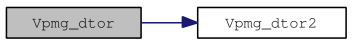
Author

Nathan Baker

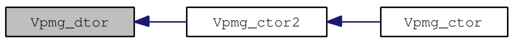
Parameters

thee Pointer to memory location of object to be destroyed

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.2.11 VEXTERNC void Vpmg_dtor2 (Vpmg ∗ *thee*)

FORTRAN stub object destructor.

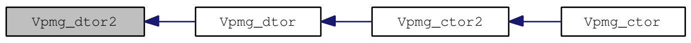
Author

Nathan Baker

Parameters

thee Pointer to object to be destroyed

Here is the caller graph for this function:



8.23.2.12 VEXTERNC double Vpmg_energy (Vpmg ∗ *thee*, int *extFlag*)

Get the total electrostatic energy.

Author

Nathan Baker

Note

The value of this observable may be modified by setting restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

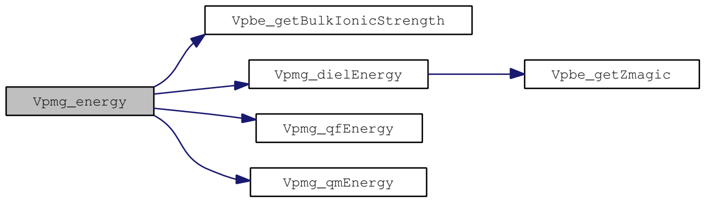
The electrostatic energy in units of k_B T.

Parameters

thee Vpmg object

extFlag If this was a focused calculation, include (1 -- for serial calculations) or ignore (0 -- for parallel calculations) energy contributions from outside the focusing domain

Here is the call graph for this function:



8.23.2.13 VPUBLIC void Vpmg_fieldSpline4 (Vpmg ∗ *thee*, int *atomID*, double *field*[3])

Computes the field at an atomic center using a stencil based on the first derivative of a 5th order B-spline.

Author

Michael Schnieders

Parameters

thee Vpmg object

atomID Atom index

field The (returned) electric field

8.23.2.14 VEXTERNC int Vpmg_fillArray (Vpmg ∗ *thee*, double ∗ *vec*, Vdata_Type *type*, double *parm*, Vhal_PBEType *pbetype*)

Fill the specified array with accessibility values.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

Parameters

thee Vpmg object

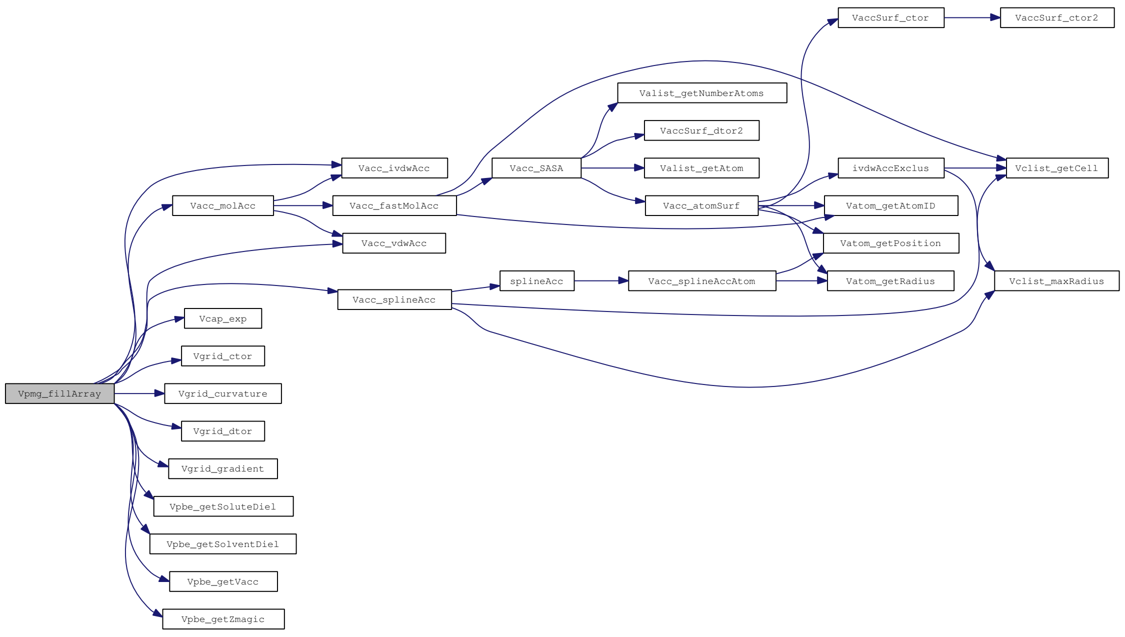
vec A nx$*$ny$*$nz$*$sizeof(double) array to contain the values to be written

type What to write

parm Parameter for data type definition (if needed)

pbetype Parameter for PBE type (if needed)

Here is the call graph for this function:



8.23.2.15 VEXTERNC int Vpmg_fillco (Vpmg $*$ *thee*, Vsurf_Meth *surfMeth*, double *splineWin*, Vchrg_Meth *chargeMeth*, int *useDielXMap*, Vgrid $*$ *dielXMap*, int *useDielYMap*, Vgrid $*$ *dielYMap*, int *useDielZMap*, Vgrid $*$ *dielZMap*, int *useKappaMap*, Vgrid $*$ *kappaMap*, int *useChargeMap*, Vgrid $*$ *chargeMap*)

Fill the coefficient arrays prior to solving the equation.

Author

Nathan Baker

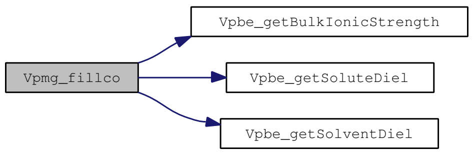
Returns

1 if successful, 0 otherwise

Parameters

thee Vpmg object
surfMeth Surface discretization method
splineWin Spline window (in A) for surfMeth = VSM_SPLINE
chargeMeth Charge discretization method
useDielXMap Boolean to use dielectric map argument
dielXMap External dielectric map
useDielYMap Boolean to use dielectric map argument
dielYMap External dielectric map
useDielZMap Boolean to use dielectric map argument
dielZMap External dielectric map
useKappaMap Boolean to use kappa map argument
kappaMap External kappa map
useChargeMap Boolean to use charge map argument
chargeMap External charge map

Here is the call graph for this function:



8.23.2.16 VEXTERNC int Vpmg_force (Vpmg $*$ *thee*, double $*$ *force*, int *atomID*, Vsurf_Meth *srfm*, Vchrg_Meth *chgm*)

Calculate the total force on the specified atom in units of k_B T/AA.

Author

Nathan Baker

Note

- Using the force evaluation methods of Im et al (Roux group), Comput Phys Commun, 111, 59--75 (1998). However, this gives the whole (self-interactions included) force -- reaction field forces will have to be calculated at higher level.

- No contributions are made from higher levels of focusing.

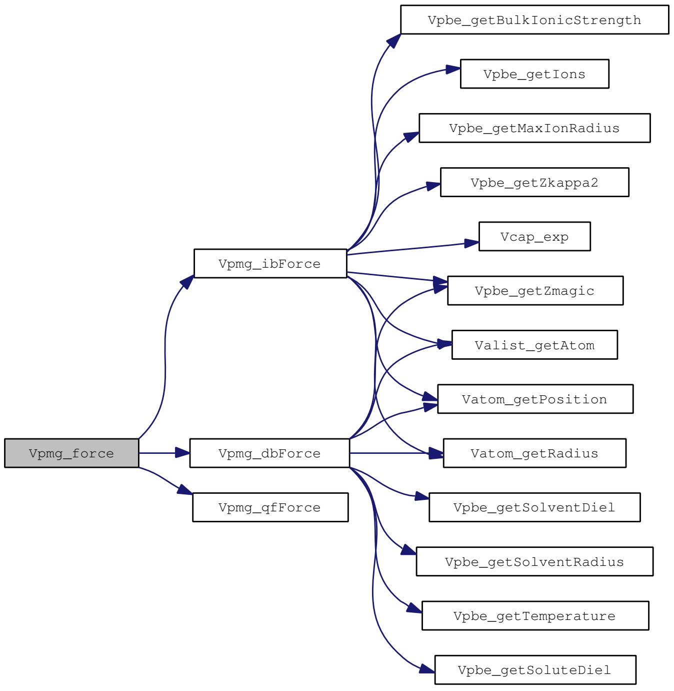
Returns

1 if successful, 0 otherwise

Parameters

thee Vpmg object
force 3∗sizeof(double) space to hold the force in units of k_B T/AA
atomID Valist ID of desired atom
srfm Surface discretization method
chgm Charge discretization method

Here is the call graph for this function:



8.23.2.17 VEXTERNC void Vpmg_ibDirectPolForce (Vpmg ∗ *thee*, Vgrid ∗ *perm*, Vgrid ∗ *induced*, int *atomID*, double *force*[3])

Ionic boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

induced Induced dipole potential

atomID Atom index

force (returned) force

8.23.2.18 VEXTERNC int Vpmg_ibForce (Vpmg ∗ *thee*, double ∗ *force*, int *atomID*, Vsurf_Meth *srfm*)

Calculate the osmotic pressure on the specified atom in units of k_B T/AA.

Author

Nathan Baker

Note

- Using the force evaluation methods of Im et al (Roux group), Comput Phys Commun, 111, 59--75 (1998). However, this gives the whole (self-interactions included) force -- reaction field forces will have to be calculated at higher level.
- No contributions are made from higher levels of focusing.

Returns

1 if successful, 0 otherwise

Parameters

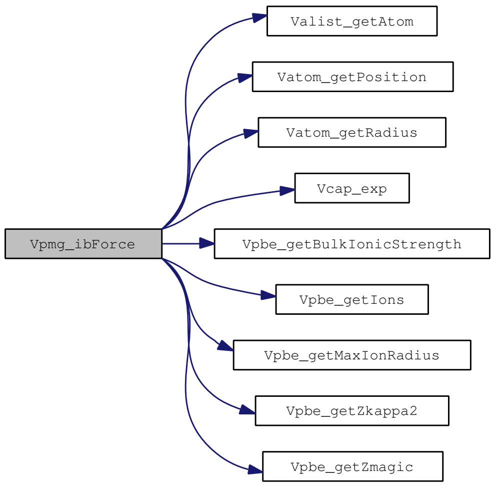
thee Vpmg object

force 3∗sizeof(double) space to hold the boundary force in units of k_B T/AA

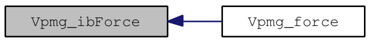
atomID Valist ID of desired atom

srfm Surface discretization method

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.2.19 VEXTERNC void Vpmg_ibMutualPolForce (Vpmg ∗ *thee*, Vgrid ∗ *induced*, Vgrid ∗ *nlInduced*, int *atomID*, double *force*[3])

Ionic boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

induced Induced dipole potential

nlInduced Non-local induced dipole potential

atomID Atom index

force (returned) force

8.23.2.20 VEXTERNC void Vpmg_ibNLDirectPolForce (Vpmg $*$ *thee*, Vgrid $*$ *perm*, Vgrid $*$ *nlInduced*, int *atomID*, double *force*[3])

Ionic boundary direct polarization force between permanent multipoles and non-local induced dipoles based on 5th order Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

nlInduced Induced dipole potential

atomID Atom index

force (returned) force

8.23.2.21 VEXTERNC void Vpmg_ibPermanentMultipoleForce (Vpmg $*$ *thee*, int *atomID*, double *force*[3])

Compute the ionic boundary force for permanent multipoles.

Author

Michael Schnieders

Parameters

thee Vpmg object

atomID Atom index

force (returned) force

8.23.2.22 VEXTERNC unsigned long int Vpmg_memChk (Vpmg $*$ *thee*)

Return the memory used by this structure (and its contents) in bytes.

Author

Nathan Baker

Returns

The memory used by this structure and its contents in bytes

Parameters

thee Object for memory check

8.23.2.23 VEXTERNC void Vpmg_printColComp (Vpmg ∗ *thee*, char *path*[72], char *title*[72], char *mxtype*[3], int *flag*)

Print out a column-compressed sparse matrix in Harwell-Boeing format.

Author

Nathan Baker

Bug

Can this path variable be replaced with a Vio socket?

Parameters

thee Vpmg object

path The file to which the matrix is to be written

title The title of the matrix

mxtype The type of REAL-valued matrix, a 3-character string of the form "R_A" where the '_' can be one of:

- S: symmetric matrix
- U: unsymmetric matrix
- H: Hermitian matrix
- Z: skew-symmetric matrix
- R: rectangular matrix

flag The operator to compress:

- 0: Poisson operator
- 1: Linearization of the full Poisson-Boltzmann operator around the current solution

8.23.2.24 VEXTERNC double Vpmg_qfAtomEnergy (Vpmg ∗ *thee*, Vatom ∗ *atom*)

Get the per-atom "fixed charge" contribution to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the fixed charges with the potential:

$$G = qu(r),$$

where q$ is the charge and r is the location of the atom of interest. The result is returned in units of k_B T. Clearly, no self-interaction terms are removed. A factor a 1/2 has to be included to convert this to a real energy.

Author

Nathan Baker

Note

The value of this observable may be modified by setting restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

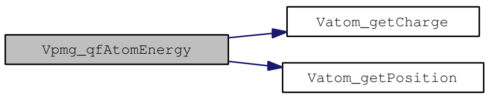
The fixed charge electrostatic energy in units of k_B T.

Parameters

thee The Vpmg object

atom The atom for energy calculations

Here is the call graph for this function:



8.23.2.25 VEXTERNC void Vpmg_qfDirectPolForce (Vpmg ∗ *thee*, Vgrid ∗ *perm*, Vgrid ∗ *induced*, int *atomID*, double *force*[3], double *torque*[3])

q-Phi direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

induced Induced dipole potential

atomID Atom index

force (returned) force

torque (returned) torque

8.23.2.26 VEXTERNC double Vpmg_qfEnergy (Vpmg $*$ *thee*, int *extFlag*)

Get the "fixed charge" contribution to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the fixed charges with the potential:

$$G = \sum_i q_i u(r_i)$$

and return the result in units of k_B T. Clearly, no self-interaction terms are removed. A factor a 1/2 has to be included to convert this to a real energy.

Author

Nathan Baker

Note

The value of this observable may be modified by setting restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

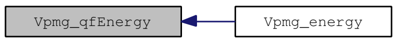
The fixed charge electrostatic energy in units of k_B T.

Parameters

thee Vpmg object

extFlag If this was a focused calculation, include (1 -- for serial calculations) or ignore (0 -- for parallel calculations) energy contributions from outside the focusing domain

Here is the caller graph for this function:



8.23.2.27 VEXTERNC int Vpmg_qfForce (Vpmg ∗ *thee*, double ∗ *force*, int *atomID*, Vchrg_Meth *chgm*)

Calculate the "charge-field" force on the specified atom in units of k_B T/AA.

Author

Nathan Baker

Note

- Using the force evaluation methods of Im et al (Roux group), Comput Phys Commun, 111, 59--75 (1998). However, this gives the whole (self-interactions included) force -- reaction field forces will have to be calculated at higher level.
- No contributions are made from higher levels of focusing.

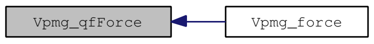
Returns

1 if sucessful, 0 otherwise

Parameters

thee Vpmg object
force 3∗sizeof(double) space to hold the force in units of k_B T/A
atomID Valist ID of desired atom
chgm Charge discretization method

Here is the caller graph for this function:



8.23.2.28 VEXTERNC void Vpmg_qfMutualPolForce (Vpmg ∗ *thee*, Vgrid ∗ *induced*, Vgrid ∗ *nlInduced*, int *atomID*, double *force*[3])

Mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

Author

Michael Schnieders

Parameters

thee Vpmg object

induced Induced dipole potential

nlInduced Non-local induced dipole potential

atomID Atom index

force (returned) force

8.23.2.29 VEXTERNC void Vpmg_qfNLDirectPolForce (Vpmg ∗ *thee*, Vgrid ∗ *perm*, Vgrid ∗ *nlInduced*, int *atomID*, double *force*[3], double *torque*[3])

q-Phi direct polarization force between permanent multipoles and non-local induced dipoles based on 5th Order B-Splines. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

Author

Michael Schnieders

Parameters

thee Vpmg object

perm Permanent multipole potential

nlInduced Non-local induced dipole potential

atomID Atom index

force (returned) force

torque (returned) torque

8.23.2.30 VEXTERNC double Vpmg_qfPermanentMultipoleEnergy (Vpmg ∗ *thee*, int *atomID*)

Computes the permanent multipole electrostatic hydration energy (the polarization component of the hydration energy currently computed in TINKER).

Author

Michael Schnieders

Returns

The permanent multipole electrostatic hydration energy

Parameters

thee Vpmg object

atomID Atom index

8.23.2.31 VEXTERNC void Vpmg_qfPermanentMultipoleForce (Vpmg * *thee*, int *atomID*, double *force*[3], double *torque*[3])

Computes the q-Phi Force for permanent multipoles based on 5th order B-splines.

Author

Michael Schnieders

Parameters

thee Vpmg object

atomID Atom index

force (returned) force

torque (returned) torque

8.23.2.32 VEXTERNC double Vpmg_qmEnergy (Vpmg * *thee*, int *extFlag*)

Get the "mobile charge" contribution to the electrostatic energy.

Using the solution at the finest mesh level, get the electrostatic energy due to the interaction of the mobile charges with the potential:

$$G = \frac{1}{4I_s} \sum_i c_i q_i^2 \int \kappa^2(x) e^{-q_i u(x)} dx$$

for the NPBE and

$$G = \frac{1}{2} \int \overline{\kappa}^2(x) u^2(x) dx$$

for the LPBE. Here i denotes the counterion species, I_s is the bulk ionic strength, kappa^2(x) is the modified Debye-Huckel parameter, c_i is the concentration of species i, q_i is the charge of species i, and u(x) is the dimensionless electrostatic potential. The energy is scaled to units of k_b T.

Author

Nathan Baker

Note

The value of this observable may be modified by setting restrictions on the subdomain over which it is calculated. Such limits can be set via Vpmg_setPart and are generally useful for parallel runs.

Returns

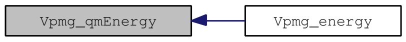
The mobile charge electrostatic energy in units of k_B T.

Parameters

thee Vpmg object

extFlag If this was a focused calculation, include (1 -- for serial calculations) or ignore (0 -- for parallel calculations) energy contributions from outside the focusing domain

Here is the caller graph for this function:



8.23.2.33 VEXTERNC void Vpmg_setPart (Vpmg $*$ *thee*, double *lowerCorner*[3], double *upperCorner*[3], int *bflags*[6])

Set partition information which restricts the calculation of observables to a (rectangular) subset of the problem domain.

Author

Nathan Baker

Parameters

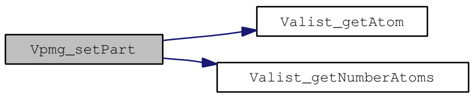
thee Vpmg object

lowerCorner Partition lower corner

upperCorner Partition upper corner

bflags Booleans indicating whether a particular processor is on the boundary with another partition. 0 if the face is not bounded (next to) another partition, and 1 otherwise.

Here is the call graph for this function:



8.23.2.34 VEXTERNC int Vpmg_solve (Vpmg $*$ *thee*)

Solve the PBE using PMG.

Author

Nathan Baker

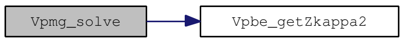
Returns

1 if successful, 0 otherwise

Parameters

thee Vpmg object

Here is the call graph for this function:



8.23.2.35 VEXTERNC int Vpmg_solveLaplace (Vpmg ∗ *thee*)

Solve Poisson's equation with a homogeneous Laplacian operator using the solvent dielectric constant. This solution is performed by a sine wave decomposition.

Author

Nathan Baker

Returns

1 if successful, 0 otherwise

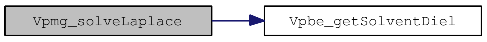
Note

This function is really only for testing purposes as the PMG multigrid solver can solve the homogeneous system much more quickly. Perhaps we should implement an FFT version at some point...

Parameters

thee Vpmg object

Here is the call graph for this function:



8.23.2.36 VEXTERNC void Vpmg_unsetPart (Vpmg * *thee*)

Remove partition restrictions.

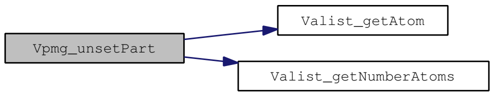
Author

Nathan Baker

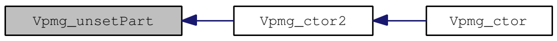
Parameters

thee Vpmg object

Here is the call graph for this function:



Here is the caller graph for this function:



8.24 Vpmgp class

Parameter structure for Mike Holst's PMGP code.

Data Structures

- struct sVpmgp

  Contains public data members for Vpmgp class/module.

Files

- file vpmgp.h

  Contains declarations for class Vpmgp.

- file vpmgp.c

  Class Vpmgp methods.

Typedefs

- typedef struct sVpmgp Vpmgp

  Declaration of the Vpmgp class as the sVpmgp structure.

Functions

- VEXTERNC Vpmgp ∗ Vpmgp_ctor (MGparm ∗mgparm)

  Construct PMG parameter object and initialize to default values.

- VEXTERNC int Vpmgp_ctor2 (Vpmgp ∗thee, MGparm ∗mgparm)

  FORTRAN stub to construct PMG parameter object and initialize to default values.

- VEXTERNC void Vpmgp_dtor (Vpmgp ∗∗thee)

  Object destructor.

- VEXTERNC void Vpmgp_dtor2 (Vpmgp ∗thee)

  FORTRAN stub for object destructor.

8.24.1 Detailed Description

Parameter structure for Mike Holst's PMGP code.

Note

Variables and many default values taken directly from PMG

8.24.2 Function Documentation

8.24.2.1 VEXTERNC Vpmgp∗ Vpmgp_ctor (MGparm ∗ *mgparm*)

Construct PMG parameter object and initialize to default values.

Author

Nathan Baker

Parameters

mgparm MGParm object containing parameters to be used in setup

Returns

Newly allocated and initialized Vpmgp object

8.24.2.2 VEXTERNC int Vpmgp_ctor2 (Vpmgp ∗ *thee*, MGparm ∗ *mgparm*)

FORTRAN stub to construct PMG parameter object and initialize to default values.

Author

Nathan Baker

Parameters

thee Newly allocated PMG object

mgparm MGParm object containing parameters to be used in setup

Returns

1 if successful, 0 otherwise

8.24.2.3 VEXTERNC void Vpmgp_dtor (Vpmgp ** *thee*)

Object destructor.

Author

Nathan Baker

Parameters

thee Pointer to memory location for Vpmgp object

8.24.2.4 VEXTERNC void Vpmgp_dtor2 (Vpmgp * *thee*)

FORTRAN stub for object destructor.

Author

Nathan Baker

Parameters

thee Pointer to Vpmgp object

Chapter 9

Data Structure Documentation

9.1 sAPOLparm Struct Reference

Parameter structure for APOL-specific variables from input files.

```
#include <apolparm.h>
```

Data Fields

- int parsed
- double grid [3]
- int setgrid
- int molid
- int setmolid
- double bconc
- int setbconc
- double sdens
- int setsdens
- double dpos
- int setdpos
- double press
- int setpress
- Vsurf_Meth srfm
- int setsrfm
- double srad
- int setsrad
- double swin
- int setswin

- double temp
- int settemp
- double gamma
- int setgamma
- APOLparm_calcEnergy calcenergy
- int setcalcenergy
- APOLparm_calcForce calcforce
- int setcalcforce
- double watsigma
- double watepsilon
- double sasa
- double sav
- double wcaEnergy
- double totForce [3]
- int setwat

9.1.1 Detailed Description

Parameter structure for APOL-specific variables from input files.

Author

David Gohara

9.1.2 Field Documentation

9.1.2.1 double bconc

Vacc sphere density

9.1.2.2 APOLparm_calcEnergy calcenergy

Energy calculation flag

9.1.2.3 APOLparm_calcForce calcforce

Atomic forces calculation

9.1.2.4 double dpos

Atom position offset

9.1.2.5 double gamma

Surface tension for apolar energies/forces (in kJ/mol/A^2)

9.1.2.6 double grid[3]

Grid spacing

9.1.2.7 int molid

Molecule ID to perform calculation on

9.1.2.8 int parsed

Flag: Has this structure been filled with anything other than the default values? (0 = no, 1 = yes)

9.1.2.9 double press

Solvent pressure

9.1.2.10 double sasa

Solvent accessible surface area for this calculation

9.1.2.11 double sav

Solvent accessible volume for this calculation

9.1.2.12 double sdens

Vacc sphere density

9.1.2.13 int setbconc

Flag,

See also

bconc

9.1.2.14 int setcalcenergy

Flag,

See also

calcenergy

9.1.2.15 int setcalcforce

Flag,

See also

calcforce

9.1.2.16 int setdpos

Flag,

See also

dpos

9.1.2.17 int setgamma

Flag,

See also

gamma

9.1.2.18 int setgrid

Flag,

See also

grid

9.1.2.19 int setmolid

Flag,

See also

molid

9.1.2.20 int setpress

Flag,

See also

press

9.1.2.21 int setsdens

Flag,

See also

sdens

9.1.2.22 int setsrad

Flag,

See also

srad

9.1.2.23 int setsrfm

Flag,

See also

srfm

9.1.2.24 int setswin

Flag,

See also

swin

9.1.2.25 int settemp

Flag,

See also

temp

9.1.2.26 int setwat

Boolean for determining if a water parameter is supplied. Yes = 1, No = 0

9.1.2.27 double srad

Solvent radius

9.1.2.28 Vsurf_Meth srfm

Surface calculation method

9.1.2.29 double swin

Cubic spline window

9.1.2.30 double temp

Temperature (in K)

9.1.2.31 double totForce[3]

Total forces on x, y, z

9.1.2.32 double watepsilon

Water oxygen Lennard-Jones well depth (kJ/mol)

9.1.2.33 double watsigma

Water oxygen Lennard-Jones radius (A)

9.1.2.34 double wcaEnergy

wcaEnergy

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

- src/generic/apbs/apolparm.h

9.2 sFEMparm Struct Reference

Parameter structure for FEM-specific variables from input files.

```
#include <femparm.h>
```

Data Fields

- int parsed
- FEMparm_CalcType type
- int settype
- double glen [3]
- int setglen
- double etol
- int setetol
- FEMparm_EtolType ekey
- int setekey
- FEMparm_EstType akeyPRE
- int setakeyPRE
- FEMparm_EstType akeySOLVE
- int setakeySOLVE
- int targetNum
- int settargetNum
- double targetRes
- int settargetRes
- int maxsolve
- int setmaxsolve
- int maxvert
- int setmaxvert
- int pkey
- int useMesh
- int meshID

9.2.1 Detailed Description

Parameter structure for FEM-specific variables from input files.

Author

Nathan Baker

9.2.2 Field Documentation

9.2.2.1 FEMparm_EstType akeyPRE

Adaptive refinment error estimator method for pre-solution refine. Note, this should either be FRT_UNIF or FRT_GEOM.

9.2.2.2 FEMparm_EstType akeySOLVE

Adaptive refinment error estimator method for a posteriori solution-based refinement.

9.2.2.3 FEMparm_EtolType ekey

Adaptive refinment interpretation of error tolerance

9.2.2.4 double etol

Error tolerance for refinement; interpretation depends on the adaptive refinement method chosen

9.2.2.5 double glen[3]

Domain side lengths (in Å)

9.2.2.6 int maxsolve

Maximum number of solve-estimate-refine cycles

9.2.2.7 int maxvert

Maximum number of vertices in mesh (ignored if less than zero)

9.2.2.8 int meshID

External finite element mesh ID (if used)

9.2.2.9 int parsed

Flag: Has this structure been filled with anything other than ∗ the default values? (0 = no, 1 = yes)

9.2.2.10 int pkey

Boolean sets the pkey type for going into AM_Refine pkey = 0 for non-HB based methods pkey = 1 for HB based methods

9.2.2.11 int setakeyPRE

Boolean

9.2.2.12 int setakeySOLVE

Boolean

9.2.2.13 int setekey

Boolean

9.2.2.14 int setetol

Boolean

9.2.2.15 int setglen

Boolean

9.2.2.16 int setmaxsolve

Boolean

9.2.2.17 int setmaxvert

Boolean

9.2.2.18 int settargetNum

Boolean

9.2.2.19 int settargetRes

Boolean

9.2.2.20 int settype

Boolean

9.2.2.21 int targetNum

Initial mesh will continue to be marked and refined with the method specified by akeyPRE until the mesh contains this many vertices or until targetRes is reached.

9.2.2.22 double targetRes

Initial mesh will continue to be marked and refined with the method specified by akeyPRE until the mesh contains no markable simplices with longest edges above this size or until targetNum is reached.

9.2.2.23 FEMparm_CalcType type

Calculation type

9.2.2.24 int useMesh

Indicates whether we use external finite element mesh

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

- src/generic/apbs/femparm.h

9.3 sMGparm Struct Reference

Parameter structure for MG-specific variables from input files.

```
#include <mgparm.h>
```

Data Fields

- MGparm_CalcType type
- int parsed
- int dime [3]
- int setdime
- Vchrg_Meth chgm
- int setchgm
- Vchrg_Src chgs
- int nlev
- int setnlev
- double etol
- int setetol
- double grid [3]
- int setgrid
- double glen [3]
- int setglen
- MGparm_CentMeth cmeth
- double center [3]
- int centmol
- int setgcent
- double cglen [3]
- int setcglen
- double fglen [3]
- int setfglen
- MGparm_CentMeth ccmeth
- double ccenter [3]
- int ccentmol
- int setcgcent
- MGparm_CentMeth fcmeth
- double fcenter [3]
- int fcentmol
- int setfgcent
- double partDisjCenter [3]
- double partDisjLength [3]
- int partDisjOwnSide [6]
- int pdime [3]

- int setpdime
- int proc_rank
- int setrank
- int proc_size
- int setsize
- double ofrac
- int setofrac
- int async
- int setasync
- int nonlintype
- int setnonlintype
- int method
- int setmethod
- int useAqua
- int setUseAqua

9.3.1 Detailed Description

Parameter structure for MG-specific variables from input files.

Author

Nathan Baker and Todd Dolinsky

Note

If you add/delete/change something in this class, the member functions -- especially MGparm_copy -- must be modified accordingly

9.3.2 Field Documentation

9.3.2.1 int async

Processor ID for asynchronous calculation

9.3.2.2 double ccenter[3]

Coarse grid center.

9.3.2.3 int ccentmol

Particular molecule on which we want to center the grid. This should be the appropriate index in an array of molecules, not the positive definite integer specified by the user.

9.3.2.4 MGparm_CentMeth ccmeth

Coarse grid centering method

9.3.2.5 double center[3]

Grid center. If ispart = 0, then this is only meaningful if cmeth = 0. However, if ispart = 1 and cmeth = MCM_PNT, then this is the center of the non-disjoint (overlapping) partition. If ispart = 1 and cmeth = MCM_MOL, then this is the vector that must be added to the center of the molecule to give the center of the non-disjoint partition.

9.3.2.6 int centmol

Particular molecule on which we want to center the grid. This should be the appropriate index in an array of molecules, not the positive definite integer specified by the user.

9.3.2.7 double cglen[3]

Coarse grid side lengths

9.3.2.8 Vchrg_Meth chgm

Charge discretization method

9.3.2.9 Vchrg_Src chgs

Charge source (Charge, Multipole, Induced Dipole, NL Induced

9.3.2.10 MGparm_CentMeth cmeth

Centering method

9.3.2.11 int dime[3]

Grid dimensions

9.3.2.12 double etol

User-defined error tolerance

9.3.2.13 double fcenter[3]

Fine grid center.

9.3.2.14 int fcentmol

Particular molecule on which we want to center the grid. This should be the appropriate index in an array of molecules, not the positive definite integer specified by the user.

9.3.2.15 MGparm_CentMeth fcmeth

Fine grid centering method

9.3.2.16 double fglen[3]

Fine grid side lengths

9.3.2.17 double glen[3]

Grid side lengths.

9.3.2.18 double grid[3]

Grid spacings

9.3.2.19 int method

Solver Method

9.3.2.20 int nlev

Levels in multigrid hierarchy

Deprecated

Just ignored now

9.3.2.21 int nonlintype

Linearity Type Method to be used

9.3.2.22 double ofrac

Overlap fraction between procs

9.3.2.23 int parsed

Has this structure been filled? (0 = no, 1 = yes)

9.3.2.24 double partDisjCenter[3]

This gives the center of the disjoint partitions

9.3.2.25 double partDisjLength[3]

This gives the lengths of the disjoint partitions

9.3.2.26 int partDisjOwnSide[6]

Tells whether the boundary points are ours (1) or not (0)

9.3.2.27 int pdime[3]

Grid of processors to be used in calculation

9.3.2.28 int proc_rank

Rank of this processor

9.3.2.29 int proc_size

Total number of processors

9.3.2.30 int setasync

Flag,

See also

asynch

9.3.2.31 int setcgcent

Flag,

See also

ccmeth

9.3.2.32 int setcglen

Flag,

See also

cglen

9.3.2.33 int setchgm

Flag,

See also

chgm

9.3.2.34 int setdime

Flag,

See also

dime

9.3.2.35 int setetol

Flag,

See also

etol

9.3.2.36 int setfgcent

Flag,

See also

fcmeth

9.3.2.37 int setfglen

Flag,

See also

fglen

9.3.2.38 int setgcent

Flag,

See also

cmeth

9.3.2.39 int setglen

Flag,

See also

glen

9.3.2.40 int setgrid

Flag,

See also

grid

9.3.2.41 int setmethod

Flag,

See also

method

9.3.2.42 int setnlev

Flag,

See also

nlev

9.3.2.43 int setnonlintype

Flag,

See also

nonlintype

9.3.2.44 int setofrac

Flag,

See also

ofrac

9.3.2.45 int setpdime

Flag,

See also

pdime

9.3.2.46 int setrank

Flag,

See also

proc_rank

9.3.2.47 int setsize

Flag,

See also

proc_size

9.3.2.48 int setUseAqua

Flag,

See also

useAqua

9.3.2.49 MGparm_CalcType type

What type of MG calculation?

9.3.2.50 int useAqua

Enable use of lpbe/aqua

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

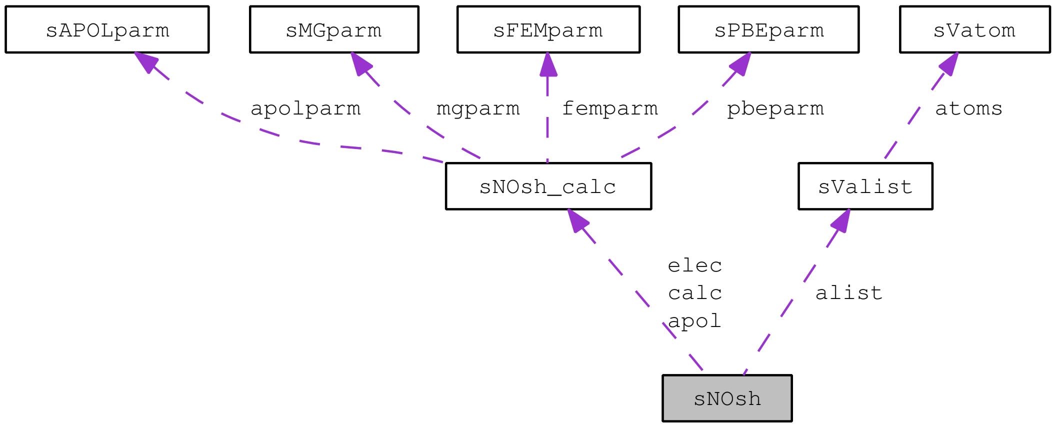
- src/generic/apbs/mgparm.h

9.4 sNOsh Struct Reference

Class for parsing fixed format input files.

`#include <nosh.h>`

Collaboration diagram for sNOsh:



Data Fields

- NOsh_calc ∗ calc [NOSH_MAXCALC]
- int ncalc
- NOsh_calc ∗ elec [NOSH_MAXCALC]
- int nelec
- NOsh_calc ∗ apol [NOSH_MAXCALC]
- int napol
- int ispara
- int proc_rank
- int proc_size
- int bogus
- int elec2calc [NOSH_MAXCALC]
- int apol2calc [NOSH_MAXCALC]
- int nmol
- char molpath [NOSH_MAXMOL][VMAX_ARGLEN]
- NOsh_MolFormat molfmt [NOSH_MAXMOL]
- Valist ∗ alist [NOSH_MAXMOL]
- int gotparm
- char parmpath [VMAX_ARGLEN]
- NOsh_ParmFormat parmfmt

- int ndiel
- char dielXpath [NOSH_MAXMOL][VMAX_ARGLEN]
- char dielYpath [NOSH_MAXMOL][VMAX_ARGLEN]
- char dielZpath [NOSH_MAXMOL][VMAX_ARGLEN]
- Vdata_Format dielfmt [NOSH_MAXMOL]
- int nkappa
- char kappapath [NOSH_MAXMOL][VMAX_ARGLEN]
- Vdata_Format kappafmt [NOSH_MAXMOL]
- int ncharge
- char chargepath [NOSH_MAXMOL][VMAX_ARGLEN]
- Vdata_Format chargefmt [NOSH_MAXMOL]
- int nmesh
- char meshpath [NOSH_MAXMOL][VMAX_ARGLEN]
- Vdata_Format meshfmt [NOSH_MAXMOL]
- int nprint
- NOsh_PrintType printwhat [NOSH_MAXPRINT]
- int printnarg [NOSH_MAXPRINT]
- int printcalc [NOSH_MAXPRINT][NOSH_MAXPOP]
- int printop [NOSH_MAXPRINT][NOSH_MAXPOP]
- int parsed
- char elecname [NOSH_MAXCALC][VMAX_ARGLEN]
- char apolname [NOSH_MAXCALC][VMAX_ARGLEN]

9.4.1 Detailed Description

Class for parsing fixed format input files.

Author

Nathan Baker

9.4.2 Field Documentation

9.4.2.1 Valist∗ alist[NOSH_MAXMOL]

Molecules for calculation (can be used in setting mesh centers

9.4.2.2 NOsh_calc∗ apol[NOSH_MAXCALC]

The array of calculation objects corresponding to APOLAR statements read in the input file. Compare to sNOsh::calc

9.4.2.3 int apol2calc[NOSH_MAXCALC]

(see elec2calc)

9.4.2.4 char apolname[NOSH_MAXCALC][VMAX_ARGLEN]

Optional user-specified name for APOLAR statement

9.4.2.5 int bogus

A flag which tells routines using NOsh that this particular NOsh is broken -- useful for parallel focusing calculations where the user gave us too many processors (1 => ignore this NOsh; 0 => this NOsh is OK)

9.4.2.6 NOsh_calc* calc[NOSH_MAXCALC]

The array of calculation objects corresponding to actual calculations performed by the code. Compare to sNOsh::elec

9.4.2.7 Vdata_Format chargefmt[NOSH_MAXMOL]

Charge maps fileformats

9.4.2.8 char chargepath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to charge map files

9.4.2.9 Vdata_Format dielfmt[NOSH_MAXMOL]

Dielectric maps file formats

9.4.2.10 char dielXpath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to x-shifted dielectric map files

9.4.2.11 char dielYpath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to y-shifted dielectric map files

9.4.2.12 char dielZpath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to z-shifted dielectric map files

9.4.2.13 NOsh_calc* elec[NOSH_MAXCALC]

The array of calculation objects corresponding to ELEC statements read in the input file. Compare to sNOsh::calc

9.4.2.14 int elec2calc[NOSH_MAXCALC]

A mapping between ELEC statements which appear in the input file and calc objects stored above. Since we allow both normal and focused multigrid, there isn't a 1-to-1 correspondence between ELEC statements and actual calcualtions. This can really confuse operations which work on specific calculations further down the road (like PRINT). Therefore this array is the initial point of entry for any calculation-specific operation. It points to a specific entry in the calc array.

9.4.2.15 char elecname[NOSH_MAXCALC][VMAX_ARGLEN]

Optional user-specified name for ELEC statement

9.4.2.16 int gotparm

Either have (1) or don't have (0) parm

9.4.2.17 int ispara

1 => is a parallel calculation, 0 => is not

9.4.2.18 Vdata_Format kappafmt[NOSH_MAXMOL]

Kappa maps file formats

9.4.2.19 char kappapath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to kappa map files

9.4.2.20 Vdata_Format meshfmt[NOSH_MAXMOL]

Mesh fileformats

9.4.2.21 char meshpath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to mesh files

9.4.2.22 NOsh_MolFormat molfmt[NOSH_MAXMOL]

Mol files formats

9.4.2.23 char molpath[NOSH_MAXMOL][VMAX_ARGLEN]

Paths to mol files

9.4.2.24 int napol

The number of apolar statements in the input file and in the apolar array

9.4.2.25 int ncalc

The number of calculations in the calc array

9.4.2.26 int ncharge

Number of charge maps

9.4.2.27 int ndiel

Number of dielectric maps

9.4.2.28 int nelec

The number of elec statements in the input file and in the elec array

9.4.2.29 int nkappa

Number of kappa maps

9.4.2.30 int nmesh

Number of meshes

9.4.2.31 int nmol

Number of molecules

9.4.2.32 int nprint

How many print sections?

9.4.2.33 NOsh_ParmFormat parmfmt

Parm file format

9.4.2.34 char parmpath[VMAX_ARGLEN]

Paths to parm file

9.4.2.35 int parsed

Have we parsed an input file yet?

9.4.2.36 int printcalc[NOSH_MAXPRINT][NOSH_MAXPOP]

ELEC id (see elec2calc)

9.4.2.37 int printnarg[NOSH_MAXPRINT]

How many arguments in energy list

9.4.2.38 int printop[NOSH_MAXPRINT][NOSH_MAXPOP]

Operation id (0 = add, 1 = subtract)

9.4.2.39 NOsh_PrintType printwhat[NOSH_MAXPRINT]

What do we print:

- 0 = energy,
- 1 = force

9.4.2.40 int proc_rank

Processor rank in parallel calculation

9.4.2.41 int proc_size

Number of processors in parallel calculation

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

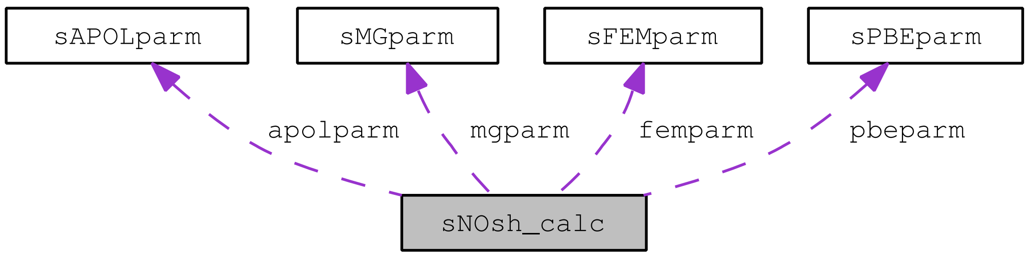
- src/generic/apbs/nosh.h

9.5 sNOsh_calc Struct Reference

Calculation class for use when parsing fixed format input files.

```
#include <nosh.h>
```

Collaboration diagram for sNOsh_calc:



Data Fields

- MGparm ∗ mgparm
- FEMparm ∗ femparm
- PBEparm ∗ pbeparm
- APOLparm ∗ apolparm
- NOsh_CalcType calctype

9.5.1 Detailed Description

Calculation class for use when parsing fixed format input files.

Author

Nathan Baker

9.5.2 Field Documentation

9.5.2.1 APOLparm∗ apolparm

Non-polar parameters

9.5.2.2 NOsh_CalcType calctype

Calculation type

9.5.2.3 FEMparm$*$ femparm

Finite element parameters

9.5.2.4 MGparm$*$ mgparm

Multigrid parameters

9.5.2.5 PBEparm$*$ pbeparm

Generic PBE parameters

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

- src/generic/apbs/nosh.h

9.6 sPBEparm Struct Reference

Parameter structure for PBE variables from input files.

```
#include <pbeparm.h>
```

Data Fields

- int molid
- int setmolid
- int useDielMap
- int dielMapID
- int useKappaMap
- int kappaMapID
- int useChargeMap
- int chargeMapID
- Vhal_PBEType pbetype
- int setpbetype
- Vbcfl bcfl
- int setbcfl
- int nion
- int setnion
- double ionq [MAXION]
- double ionc [MAXION]
- double ionr [MAXION]
- int setion [MAXION]
- double pdie
- int setpdie
- double sdens
- int setsdens
- double sdie
- int setsdie
- Vsurf_Meth srfm
- int setsrfm
- double srad
- int setsrad
- double swin
- int setswin
- double temp
- int settemp
- double smsize
- int setsmsize
- double smvolume

- int setsmvolume
- PBEparm_calcEnergy calcenergy
- int setcalcenergy
- PBEparm_calcForce calcforce
- int setcalcforce
- double zmem
- int setzmem
- double Lmem
- int setLmem
- double mdie
- int setmdie
- double memv
- int setmemv
- int numwrite
- char writestem [PBEPARM_MAXWRITE][VMAX_ARGLEN]
- Vdata_Type writetype [PBEPARM_MAXWRITE]
- Vdata_Format writefmt [PBEPARM_MAXWRITE]
- int writemat
- int setwritemat
- char writematstem [VMAX_ARGLEN]
- int writematflag
- int parsed

9.6.1 Detailed Description

Parameter structure for PBE variables from input files.

Author

Nathan Baker

Note

If you add/delete/change something in this class, the member functions -- especially PBEparm_copy -- must be modified accordingly

9.6.2 Field Documentation

9.6.2.1 Vbcfl bcfl

Boundary condition method

9.6.2.2 PBEparm_calcEnergy calcenergy

Energy calculation flag

9.6.2.3 PBEparm_calcForce calcforce

Atomic forces calculation

9.6.2.4 int chargeMapID

Charge distribution map ID (if used)

9.6.2.5 int dielMapID

Dielectric map ID (if used)

9.6.2.6 double ionc[MAXION]

Counterion concentrations (in M)

9.6.2.7 double ionq[MAXION]

Counterion charges (in e)

9.6.2.8 double ionr[MAXION]

Counterion radii (in A)

9.6.2.9 int kappaMapID

Kappa map ID (if used)

9.6.2.10 double Lmem

membrane width

9.6.2.11 double mdie

membrane dielectric constant

9.6.2.12 double memv

Membrane potential

9.6.2.13 int molid

Molecule ID to perform calculation on

9.6.2.14 int nion

Number of counterion species

9.6.2.15 int numwrite

Number of write statements encountered

9.6.2.16 int parsed

Has this been filled with anything other than the default values?

9.6.2.17 Vhal_PBEType pbetype

Which version of the PBE are we solving?

9.6.2.18 double pdie

Solute dielectric

9.6.2.19 double sdens

Vacc sphere density

9.6.2.20 double sdie

Solvent dielectric

9.6.2.21 int setbcfl

Flag,

See also

bcfl

9.6.2.22 int setcalcenergy

Flag,

See also

calcenergy

9.6.2.23 int setcalcforce

Flag,

See also

calcforce

9.6.2.24 int setion[MAXION]

Flag,

See also

ionq

9.6.2.25 int setLmem

Flag

9.6.2.26 int setmdie

Flag

9.6.2.27 int setmemv

Flag

9.6.2.28 int setmolid

Flag,

See also

molid

9.6.2.29 int setnion

Flag,

See also

nion

9.6.2.30 int setpbetype

Flag,

See also

pbetype

9.6.2.31 int setpdie

Flag,

See also

pdie

9.6.2.32 int setsdens

Flag,

See also

sdens

9.6.2.33 int setsdie

Flag,

See also

sdie

9.6.2.34 int setsmsize

Flag,

See also

temp

9.6.2.35 int setsmvolume

Flag,

See also

temp

9.6.2.36 int setsrad

Flag,

See also

srad

9.6.2.37 int setsrfm

Flag,

See also

srfm

9.6.2.38 int setswin

Flag,

See also

swin

9.6.2.39 int settemp

Flag,

See also

temp

9.6.2.40 int setwritemat

Flag,

See also

writemat

9.6.2.41 int setzmem

Flag

9.6.2.42 double smsize

SMPBE size

9.6.2.43 double smvolume

SMPBE size

9.6.2.44 double srad

Solvent radius

9.6.2.45 Vsurf_Meth srfm

Surface calculation method

9.6.2.46 double swin

Cubic spline window

9.6.2.47 double temp

Temperature (in K)

9.6.2.48 int useChargeMap

Indicates whether we use an external charge distribution map

9.6.2.49 int useDielMap

Indicates whether we use external dielectric maps (note plural)

9.6.2.50 int useKappaMap

Indicates whether we use an external kappa map

9.6.2.51 Vdata_Format writefmt[PBEPARM_MAXWRITE]

File format to write data in

9.6.2.52 int writemat

Write out the operator matrix?

- 0 => no
- 1 => yes

9.6.2.53 int writematflag

What matrix should we write:

- 0 => Poisson (differential operator)
- 1 => Poisson-Boltzmann operator linearized around solution (if applicable)

9.6.2.54 char writematstem[VMAX_ARGLEN]

File stem to write mat

9.6.2.55 char writestem[PBEPARM_MAXWRITE][VMAX_ARGLEN]

File stem to write data to

9.6.2.56 Vdata_Type writetype[PBEPARM_MAXWRITE]

What data to write

9.6.2.57 double zmem

z value of membrane bottom

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

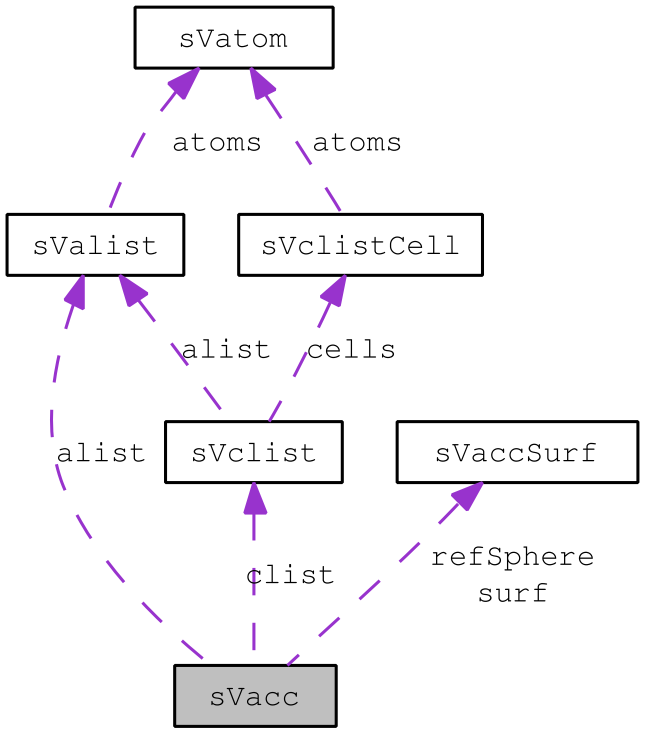
- src/generic/apbs/pbeparm.h

9.7 sVacc Struct Reference

Oracle for solvent- and ion-accessibility around a biomolecule.

`#include <vacc.h>`

Collaboration diagram for sVacc:



Data Fields

- Vmem ∗ mem
- Valist ∗ alist
- Vclist ∗ clist
- int ∗ atomFlags
- VaccSurf ∗ refSphere
- VaccSurf ∗∗ surf
- Vset acc
- double surf_density

9.7.1 Detailed Description

Oracle for solvent- and ion-accessibility around a biomolecule.

Author

Nathan Baker

9.7.2 Field Documentation

9.7.2.1 Vset acc

An integer array (to be treated as bitfields) of Vset type with length equal to the number of vertices in the mesh

9.7.2.2 Valist* alist

Valist structure for list of atoms

9.7.2.3 int* atomFlags

Array of boolean flags of length Valist_getNumberAtoms(thee->alist) to prevent double-counting atoms during calculations

9.7.2.4 Vclist* clist

Vclist structure for atom cell list

9.7.2.5 Vmem* mem

Memory management object for this class

9.7.2.6 VaccSurf* refSphere

Reference sphere for SASA calculations

9.7.2.7 VaccSurf** surf

Array of surface points for each atom; is not initialized until needed (test against VNULL to determine initialization state)

9.7.2.8 double surf_density

Minimum solvent accessible surface point density (in pts/A^2)

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

- src/generic/apbs/vacc.h

9.8 sVaccSurf Struct Reference

Surface object list of per-atom surface points.

`#include <vacc.h>`

Data Fields

- Vmem ∗ mem
- double ∗ xpts
- double ∗ ypts
- double ∗ zpts
- char ∗ bpts
- double area
- int npts
- double probe_radius

9.8.1 Detailed Description

Surface object list of per-atom surface points.

Author

Nathan Baker

9.8.2 Field Documentation

9.8.2.1 double area

Area spanned by these points

9.8.2.2 char∗ bpts

Array of booleans indicating whether a point is (1) or is not (0) part of the surface

9.8.2.3 Vmem∗ mem

Memory object

9.8.2.4 int npts

Length of thee->xpts, ypts, zpts arrays

9.8.2.5 double probe_radius

Probe radius (A) with which this surface was constructed

9.8.2.6 double∗ xpts

Array of point x-locations

9.8.2.7 double∗ ypts

Array of point y-locations

9.8.2.8 double∗ zpts

Array of point z-locations

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

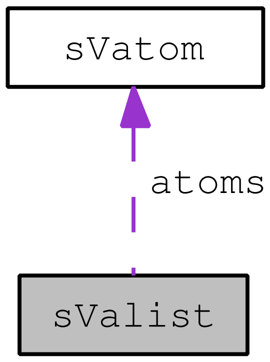
- src/generic/apbs/vacc.h

9.9 sValist Struct Reference

Container class for list of atom objects.

`#include <valist.h>`

Collaboration diagram for sValist:



Data Fields

- int number
- double center [3]
- double mincrd [3]
- double maxcrd [3]
- double maxrad
- double charge
- Vatom ∗ atoms
- Vmem ∗ vmem

9.9.1 Detailed Description

Container class for list of atom objects.

Author

Nathan Baker

9.9.2 Field Documentation

9.9.2.1 Vatom∗ atoms

Atom list

9.9.2.2 double center[3]

Molecule center (xmin - xmax)/2, etc.

9.9.2.3 double charge

Net charge

9.9.2.4 double maxcrd[3]

Maximum coordinates

9.9.2.5 double maxrad

Maximum radius

9.9.2.6 double mincrd[3]

Minimum coordinates

9.9.2.7 int number

Number of atoms in list

9.9.2.8 Vmem∗ vmem

Memory management object

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

- src/generic/apbs/valist.h

9.10 sVatom Struct Reference

Contains public data members for Vatom class/module.

```
#include <vatom.h>
```

Data Fields

- double position [3]
- double radius
- double charge
- double partID
- double epsilon
- int id
- char resName [VMAX_RECLEN]
- char atomName [VMAX_RECLEN]

9.10.1 Detailed Description

Contains public data members for Vatom class/module.

Author

Nathan Baker, David Gohara, Mike Schneiders

9.10.2 Field Documentation

9.10.2.1 char atomName[VMAX_RECLEN]

Atom name from PDB/PDR file

9.10.2.2 double charge

Atomic charge

9.10.2.3 double epsilon

Epsilon value for WCA calculations

9.10.2.4 int id

Atomic ID; this should be a unique non-negative integer assigned based on the index of the atom in a Valist atom array

9.10.2.5 double partID

Partition value for assigning atoms to particular processors and/or partitions

9.10.2.6 double position[3]

Atomic position

9.10.2.7 double radius

Atomic radius

9.10.2.8 char resName[VMAX_RECLEN]

Residue name from PDB/PQR file

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

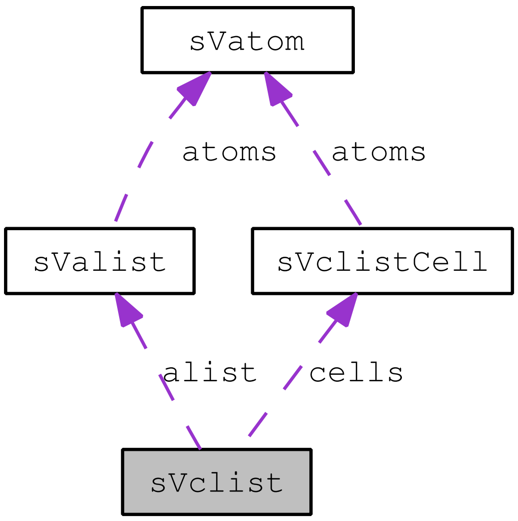
- src/generic/apbs/vatom.h

9.11 sVclist Struct Reference

Atom cell list.

`#include <vclist.h>`

Collaboration diagram for sVclist:



Data Fields

- Vmem ∗ vmem
- Valist ∗ alist
- Vclist_DomainMode mode
- int npts [VAPBS_DIM]
- int n
- double max_radius
- VclistCell ∗ cells
- double lower_corner [VAPBS_DIM]
- double upper_corner [VAPBS_DIM]
- double spacs [VAPBS_DIM]

9.11.1 Detailed Description

Atom cell list.

Author

Nathan Baker

9.11.2 Field Documentation

9.11.2.1 Valist∗ alist

Original Valist structure for list of atoms

9.11.2.2 VclistCell∗ cells

Cell array of length thee->n

9.11.2.3 double lower_corner[VAPBS_DIM]

Hash table grid corner

9.11.2.4 double max_radius

Maximum probe radius

9.11.2.5 Vclist_DomainMode mode

How the cell list was constructed

9.11.2.6 int n

n = nx∗nz∗ny

9.11.2.7 int npts[VAPBS_DIM]

Hash table grid dimensions

9.11.2.8 double spacs[VAPBS_DIM]

Hash table grid spacings

9.11.2.9 double upper_corner[VAPBS_DIM]

Hash table grid corner

9.11.2.10 Vmem∗ vmem

Memory management object for this class

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

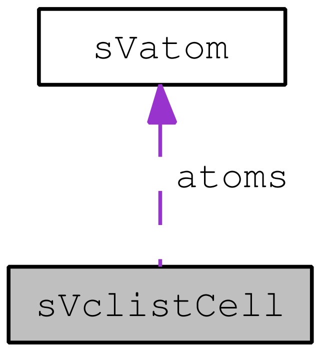
- src/generic/apbs/vclist.h

9.12 sVclistCell Struct Reference

Atom cell list cell.

`#include <vclist.h>`

Collaboration diagram for sVclistCell:



Data Fields

- Vatom ∗∗ atoms
- int natoms

9.12.1 Detailed Description

Atom cell list cell.

Author

Nathan Baker

9.12.2 Field Documentation

9.12.2.1 Vatom∗∗ atoms

Array of atom objects associated with this cell

9.12.2.2 int natoms

Length of thee->atoms array

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

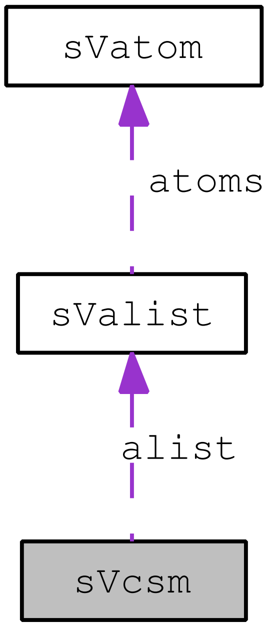
- src/generic/apbs/vclist.h

9.13 sVcsm Struct Reference

Charge-simplex map class.

```
#include <vcsm.h>
```

Collaboration diagram for sVcsm:



Data Fields

- Valist ∗ alist
- int natom
- Gem ∗ gm
- int ∗∗ sqm
- int ∗ nsqm
- int nsimp
- int msimp
- int ∗∗ qsm
- int ∗ nqsm
- int initFlag
- Vmem ∗ vmem

9.13.1 Detailed Description

Charge-simplex map class.

Author

Nathan Baker

9.13.2 Field Documentation

9.13.2.1 Valist∗ alist

Atom (charge) list

9.13.2.2 Gem∗ gm

Grid manager (container class for master vertex and simplex lists as well as prolongation operator for updating after refinement)

9.13.2.3 int initFlag

Indicates whether the maps have been initialized yet

9.13.2.4 int msimp

The maximum number of entries that can be accomodated by sqm or nsqm -- saves on realloc's

9.13.2.5 int natom

Size of thee->alist; redundant, but useful for convenience

9.13.2.6 int∗ nqsm

The length of the simplex lists in thee->qsm

9.13.2.7 int nsimp

The _currently used) length of sqm, nsqm -- may not always be up-to-date with Gem

9.13.2.8 int* nsqm

The length of the charge lists in thee->sqm

9.13.2.9 int** qsm

The inverse of sqm; the list of simplices associated with a given charge

9.13.2.10 int** sqm

The map which gives the list charges associated with each simplex in gm->simplices. The indices of the first dimension are associated with the simplex ID's in Vgm. Each charge list (second dimension) contains entries corresponding to indicies in thee->alist with lengths given in thee->nsqm

9.13.2.11 Vmem* vmem

Memory management object

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

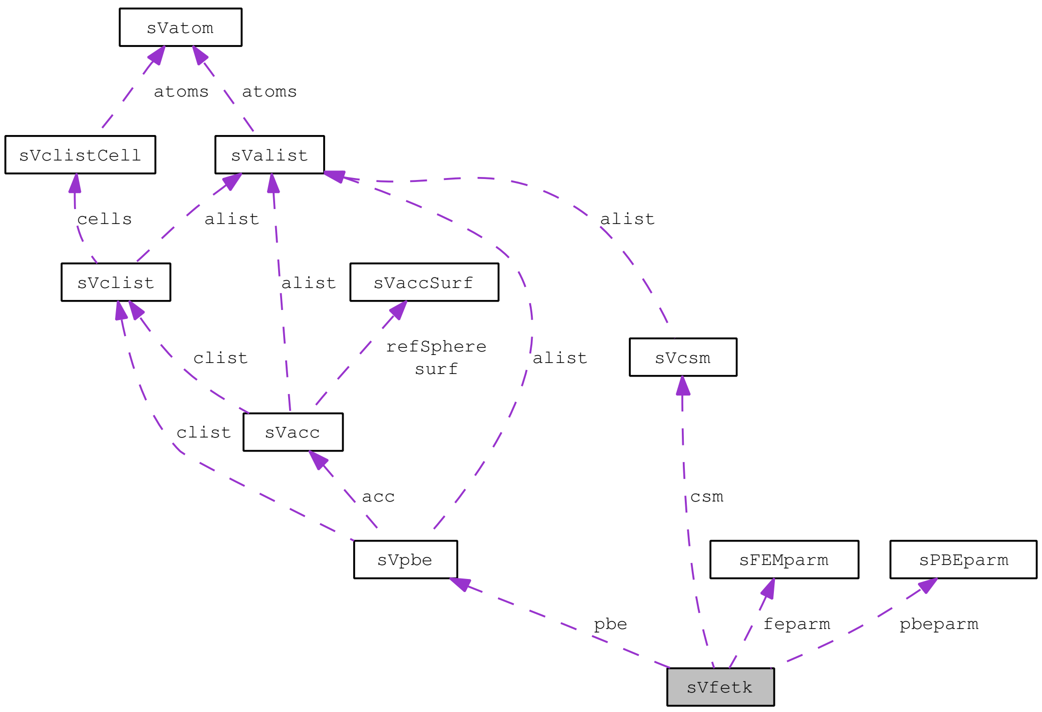
- src/fem/apbs/vcsm.h

9.14 sVfetk Struct Reference

Contains public data members for Vfetk class/module.

`#include <vfetk.h>`

Collaboration diagram for sVfetk:



Data Fields

- Vmem ∗ vmem
- Gem ∗ gm
- AM ∗ am
- Aprx ∗ aprx
- PDE ∗ pde
- Vpbe ∗ pbe
- Vcsm ∗ csm
- Vfetk_LsolvType lkey
- int lmax
- double ltol
- Vfetk_NsolvType nkey

- int nmax
- double ntol
- Vfetk_GuessType gues
- Vfetk_PrecType lprec
- int pjac
- PBEparm ∗ pbeparm
- FEMparm ∗ feparm
- Vhal_PBEType type
- int level

9.14.1 Detailed Description

Contains public data members for Vfetk class/module.

Author

Nathan Baker Many of the routines and macros are borrowed from the main.c driver (written by Mike Holst) provided with the PMG code.

9.14.2 Field Documentation

9.14.2.1 AM∗ am

Multilevel algebra manager.

9.14.2.2 Aprx∗ aprx

Approximation manager.

9.14.2.3 Vcsm∗ csm

Charge-simplex map

9.14.2.4 FEMparm∗ feparm

FEM-specific parameters

9.14.2.5 Gem∗ gm

Grid manager (container class for master vertex and simplex lists as well as prolongation operator for updating after refinement).

9.14.2.6 Vfetk_GuessType gues

Initial guess method

9.14.2.7 int level

Refinement level (starts at 0)

9.14.2.8 Vfetk_LsolvType lkey

Linear solver method

9.14.2.9 int lmax

Maximum number of linear solver iterations

9.14.2.10 Vfetk_PrecType lprec

Linear preconditioner

9.14.2.11 double ltol

Residual tolerance for linear solver

9.14.2.12 Vfetk_NsolvType nkey

Nonlinear solver method

9.14.2.13 int nmax

Maximum number of nonlinear solver iterations

9.14.2.14 double ntol

Residual tolerance for nonlinear solver

9.14.2.15 Vpbe$*$ pbe

Poisson-Boltzmann object

9.14.2.16 PBEparm∗ pbeparm

Generic PB parameters

9.14.2.17 PDE∗ pde

FEtk PDE object

9.14.2.18 int pjac

Flag to print the jacobians (usually set this to -1, please)

9.14.2.19 Vhal_PBEType type

Version of PBE to solve

9.14.2.20 Vmem∗ vmem

Memory management object

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

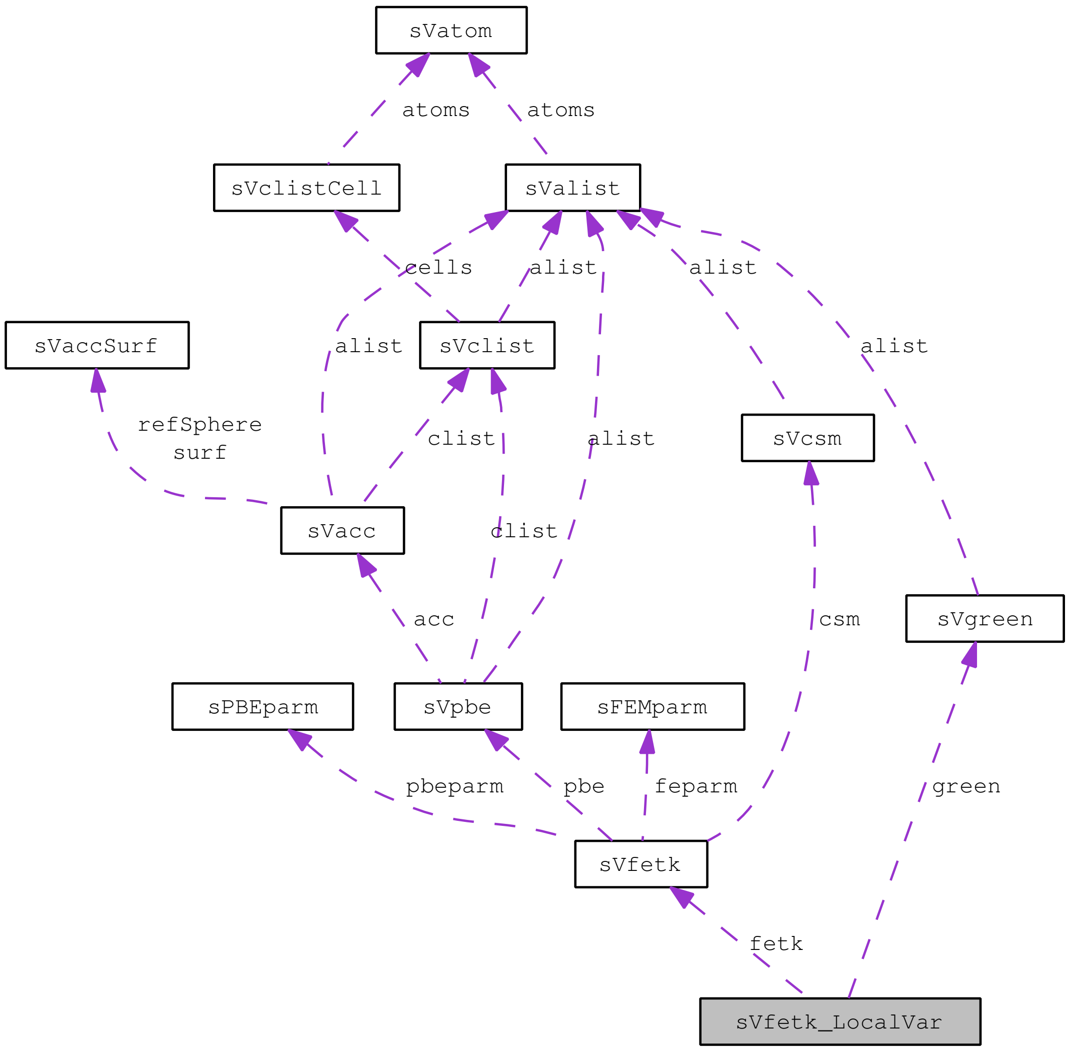
- src/fem/apbs/vfetk.h

9.15 sVfetk_LocalVar Struct Reference

Vfetk LocalVar subclass.

`#include <vfetk.h>`

Collaboration diagram for sVfetk_LocalVar:



Data Fields

- double nvec [VAPBS_DIM]

- double vx [4][VAPBS_DIM]
- double xq [VAPBS_DIM]
- double U [MAXV]
- double dU [MAXV][VAPBS_DIM]
- double W
- double dW [VAPBS_DIM]
- double d2W
- int sType
- int fType
- double diel
- double ionacc
- double A
- double F
- double B
- double DB
- double jumpDiel
- Vfetk ∗ fetk
- Vgreen ∗ green
- int initGreen
- SS ∗ simp
- VV ∗ verts [4]
- int nverts
- double ionConc [MAXION]
- double ionQ [MAXION]
- double ionRadii [MAXION]
- double zkappa2
- double zks2
- double ionstr
- int nion
- double Fu_v
- double DFu_wv
- double delta
- double u_D
- double u_T

9.15.1 Detailed Description

Vfetk LocalVar subclass.

Author

Nathan Baker Contains variables used when solving the PDE with FEtk

9.15.2 Field Documentation

9.15.2.1 double A

Second-order differential term

9.15.2.2 double B

Entire ionic strength term

9.15.2.3 double d2W

Coulomb regularization term Laplacia

9.15.2.4 double DB

Entire ionic strength term derivative

9.15.2.5 double delta

Store delta value

9.15.2.6 double DFu_wv

Store DFu_wv value

9.15.2.7 double diel

Dielectric value

9.15.2.8 double dU[MAXV][VAPBS_DIM]

Solution gradient

9.15.2.9 double dW[VAPBS_DIM]

Coulomb regularization term gradient

9.15.2.10 double F

RHS characteristic function value

9.15.2.11 Vfetk∗ fetk

Pointer to the VFETK object

9.15.2.12 int fType

Face type

9.15.2.13 double Fu_v

Store Fu_v value

9.15.2.14 Vgreen∗ green

Pointer to a Green's function object

9.15.2.15 int initGreen

Boolean to designate whether Green's function has been initialized

9.15.2.16 double ionacc

Ion accessibility value

9.15.2.17 double ionConc[MAXION]

Counterion species' concentrations

9.15.2.18 double ionQ[MAXION]

Counterion species' valencies

9.15.2.19 double ionRadii[MAXION]

Counterion species' radii

9.15.2.20 double ionstr

Ionic strength parameters (M)

9.15.2.21 double jumpDiel

Dielectric value on one side of a simplex face

9.15.2.22 int nion

Number of ion species

9.15.2.23 double nvec[VAPBS_DIM]

Normal vector for a simplex face

9.15.2.24 int nverts

number of vertices in the simplex

9.15.2.25 SS∗ simp

Pointer to the latest simplex object; set in initElement() and delta()

9.15.2.26 int sType

Simplex type

9.15.2.27 double U[MAXV]

Solution value

9.15.2.28 double u_D

Store Dirichlet value

9.15.2.29 double u_T

Store true value

9.15.2.30 VV∗ verts[4]

Pointer to the latest vertices; set in initElement

9.15.2.31 double vx[4][VAPBS_DIM]

Vertex coordinates

9.15.2.32 double W

Coulomb regularization term scalar value

9.15.2.33 double xq[VAPBS_DIM]

Quadrature pt

9.15.2.34 double zkappa2

Ionic strength parameters

9.15.2.35 double zks2

Ionic strength parameters

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

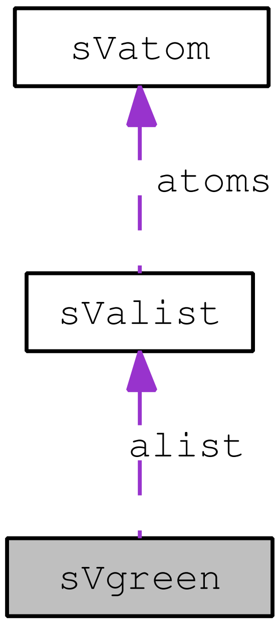
- src/fem/apbs/vfetk.h

9.16 sVgreen Struct Reference

Contains public data members for Vgreen class/module.

`#include <vgreen.h>`

Collaboration diagram for sVgreen:



Data Fields

- Valist ∗ alist
- Vmem ∗ vmem
- double ∗ xp
- double ∗ yp
- double ∗ zp
- double ∗ qp
- int np

9.16.1 Detailed Description

Contains public data members for Vgreen class/module.

Author

Nathan Baker

9.16.2 Field Documentation

9.16.2.1 Valist∗ alist

Atom (charge) list for Green's function

9.16.2.2 int np

Set to size of above arrays

9.16.2.3 double∗ qp

Array of particle charges for use with treecode routines

9.16.2.4 Vmem∗ vmem

Memory management object

9.16.2.5 double∗ xp

Array of particle x-coordinates for use with treecode routines

9.16.2.6 double∗ yp

Array of particle y-coordinates for use with treecode routines

9.16.2.7 double∗ zp

Array of particle z-coordinates for use with treecode routines

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

- src/generic/apbs/vgreen.h

9.17 sVgrid Struct Reference

Electrostatic potential oracle for Cartesian mesh data.

```
#include <vgrid.h>
```

Data Fields

- int nx
- int ny
- int nz
- double hx
- double hy
- double hzed
- double xmin
- double ymin
- double zmin
- double xmax
- double ymax
- double zmax
- double ∗ data
- int readdata
- int ctordata
- Vmem ∗ mem

9.17.1 Detailed Description

Electrostatic potential oracle for Cartesian mesh data.

Author

Nathan Baker

9.17.2 Field Documentation

9.17.2.1 int ctordata

flag indicating whether data was included at construction

9.17.2.2 double∗ data

nx∗ny∗nz array of data

9.17.2.3 double hx

Grid spacing in x direction

9.17.2.4 double hy

Grid spacing in y direction

9.17.2.5 double hzed

Grid spacing in z direction

9.17.2.6 Vmem∗ mem

Memory manager object

9.17.2.7 int nx

Number grid points in x direction

9.17.2.8 int ny

Number grid points in y direction

9.17.2.9 int nz

Number grid points in z direction

9.17.2.10 int readdata

flag indicating whether data was read from file

9.17.2.11 double xmax

x coordinate of upper grid corner

9.17.2.12 double xmin

x coordinate of lower grid corner

9.17.2.13 double ymax

y coordinate of upper grid corner

9.17.2.14 double ymin

y coordinate of lower grid corner

9.17.2.15 double zmax

z coordinate of upper grid corner

9.17.2.16 double zmin

z coordinate of lower grid corner

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

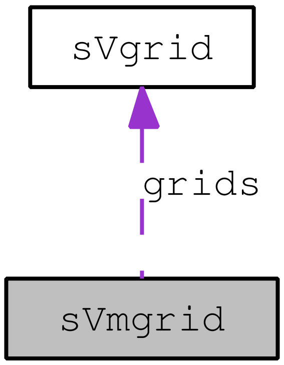
- src/mg/apbs/vgrid.h

9.18 sVmgrid Struct Reference

Multiresoltion oracle for Cartesian mesh data.

```
#include <vmgrid.h>
```

Collaboration diagram for sVmgrid:



Data Fields

- int ngrids
- Vgrid ∗ grids [VMGRIDMAX]

9.18.1 Detailed Description

Multiresoltion oracle for Cartesian mesh data.

Author

Nathan Baker

9.18.2 Field Documentation

9.18.2.1 Vgrid∗ grids[VMGRIDMAX]

Grids in hiearchy. Our convention will be to have the finest grid first, however, this will not be enforced as it may be useful to search multiple grids for parallel datasets, etc.

9.18.2.2 int ngrids

Number of grids in hiearchy

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

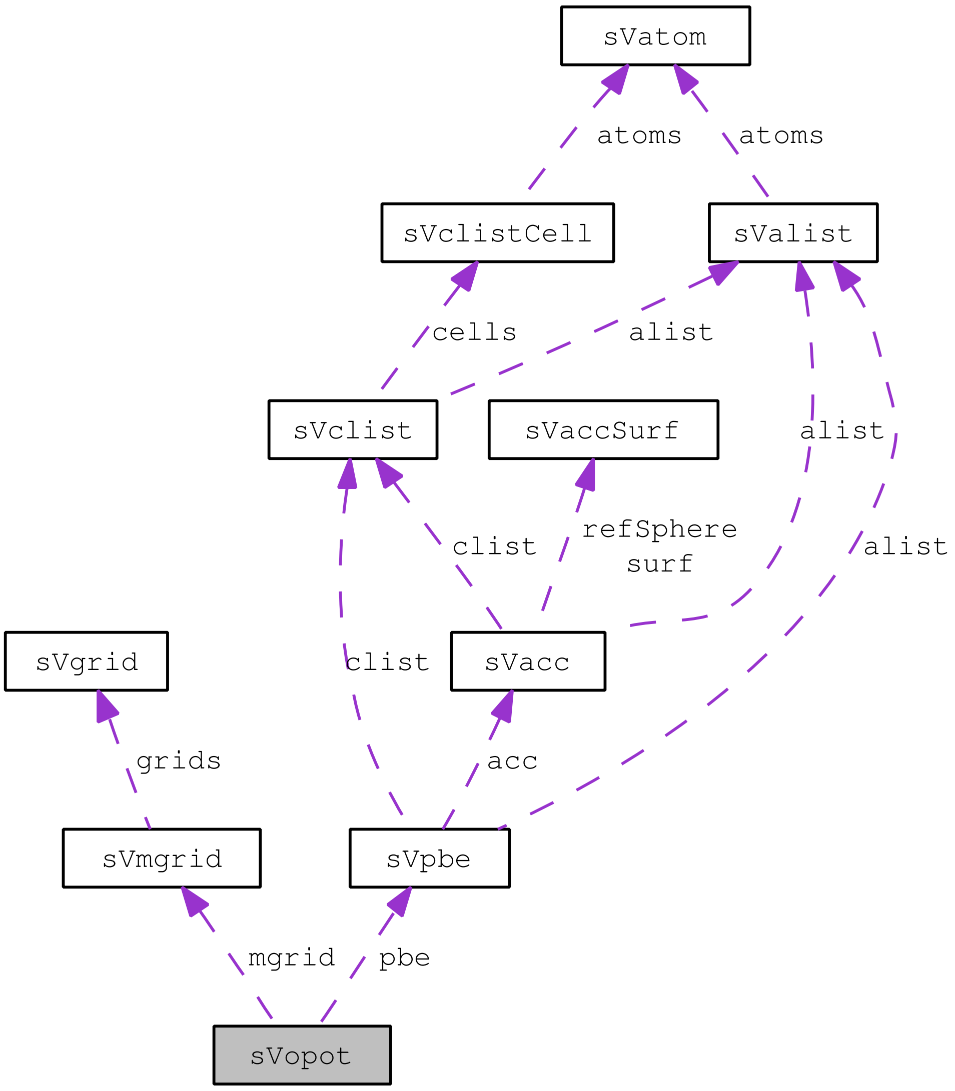
- src/mg/apbs/vmgrid.h

9.19 sVopot Struct Reference

Electrostatic potential oracle for Cartesian mesh data.

`#include <vopot.h>`

Collaboration diagram for sVopot:



Data Fields

- Vmgrid $*$ mgrid
- Vpbe $*$ pbe
- Vbcfl bcfl

9.19.1 Detailed Description

Electrostatic potential oracle for Cartesian mesh data.

Author

Nathan Baker

9.19.2 Field Documentation

9.19.2.1 Vbcfl bcfl

Boundary condition flag for returning potential values at points off the grid.

9.19.2.2 Vmgrid$*$ mgrid

Multiple grid object containing potential data (in units kT/e)

9.19.2.3 Vpbe$*$ pbe

Pointer to PBE object

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

- src/mg/apbs/vopot.h

9.20 sVparam_AtomData Struct Reference

AtomData sub-class; stores atom data.

```
#include <vparam.h>
```

Data Fields

- char atomName [VMAX_ARGLEN]
- char resName [VMAX_ARGLEN]
- double charge
- double radius
- double epsilon

9.20.1 Detailed Description

AtomData sub-class; stores atom data.

Author

Nathan Baker

Note

The epsilon and radius members of this class refer use the following formula for calculating the van der Waals energy of atom i interacting with atom j:

$$V_{ij}(r_{ij}) = \epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right]$$

where $\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j}$ is the well-depth (in the desired energy units), r_{ij} is the distance between atoms i and j, and $\sigma_{ij} = \sigma_i + \sigma_j$ is the sum of the van der Waals radii.

9.20.2 Field Documentation

9.20.2.1 char atomName[VMAX_ARGLEN]

Atom name

9.20.2.2 double charge

Atom charge (in e)

9.20.2.3 double epsilon

Atom VdW well depth (ϵ_i above; in kJ/mol)

9.20.2.4 double radius

Atom VdW radius (σ_i above; in Å)

9.20.2.5 char resName[VMAX_ARGLEN]

Residue name

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

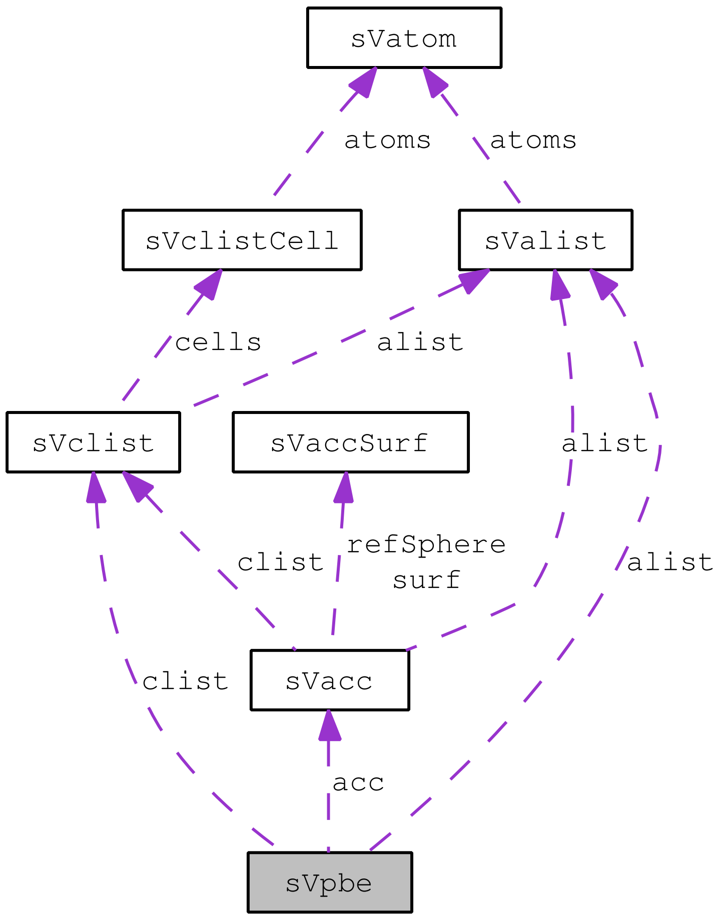
- src/generic/apbs/vparam.h

9.21 sVpbe Struct Reference

Contains public data members for Vpbe class/module.

`#include <vpbe.h>`

Collaboration diagram for sVpbe:



Data Fields

- Vmem ∗ vmem
- Valist ∗ alist
- Vclist ∗ clist
- Vacc ∗ acc
- double T

- double soluteDiel
- double solventDiel
- double solventRadius
- double bulkIonicStrength
- double maxIonRadius
- int numIon
- double ionConc [MAXION]
- double ionRadii [MAXION]
- double ionQ [MAXION]
- double xkappa
- double deblen
- double zkappa2
- double zmagic
- double soluteCenter [3]
- double soluteRadius
- double soluteXlen
- double soluteYlen
- double soluteZlen
- double soluteCharge
- double smvolume
- double smsize
- int ipkey
- int paramFlag
- double z_mem
- double L
- double membraneDiel
- double V
- int param2Flag

9.21.1 Detailed Description

Contains public data members for Vpbe class/module.

Author

Nathan Baker

9.21.2 Field Documentation

9.21.2.1 Vacc* acc

Accessibility object

9.21.2.2 Valist∗ alist

Atom (charge) list

9.21.2.3 double bulkIonicStrength

Bulk ionic strength (M)

9.21.2.4 Vclist∗ clist

Atom location cell list

9.21.2.5 double deblen

Debye length (bulk)

9.21.2.6 double ionConc[MAXION]

Concentration (M) of each species

9.21.2.7 double ionQ[MAXION]

Charge (e) of each species

9.21.2.8 double ionRadii[MAXION]

Ionic radius (A) of each species

9.21.2.9 int ipkey

PBE calculation type (this is a cached copy it should not be used directly in code)

9.21.2.10 double L

Length of the membrane (A)

9.21.2.11 double maxIonRadius

Max ion radius (A; used for calculating accessiblity and defining volumes for ionic strength coeffcients)

9.21.2.12 double membraneDiel

Membrane dielectric constant

9.21.2.13 int numIon

Total number of ion species

9.21.2.14 int param2Flag

Check to see if bcfl=3 parms have been set

9.21.2.15 int paramFlag

Check to see if the parameters have been set

9.21.2.16 double smsize

Size-Modified PBE size

9.21.2.17 double smvolume

Size-Modified PBE relative volume

9.21.2.18 double soluteCenter[3]

Center of solute molecule (A)

9.21.2.19 double soluteCharge

Charge of solute molecule (e)

9.21.2.20 double soluteDiel

Solute dielectric constant (unitless)

9.21.2.21 double soluteRadius

Radius of solute molecule (A)

9.21.2.22 double soluteXlen

Solute length in x-direction

9.21.2.23 double soluteYlen

Solute length in y-direction

9.21.2.24 double soluteZlen

Solute length in z-direction

9.21.2.25 double solventDiel

Solvent dielectric constant (unitless)

9.21.2.26 double solventRadius

Solvent probe radius (angstroms) for accessibility; determining defining volumes for the dielectric coefficient

9.21.2.27 double T

Temperature (K)

9.21.2.28 double V

Membrane potential

9.21.2.29 Vmem* vmem

Memory management object

9.21.2.30 double xkappa

Debye-Huckel parameter (bulk)

9.21.2.31 double z_mem

Z value of the botton of the membrane (A)

9.21.2.32 double zkappa2

Square of modified Debye-Huckel parameter (bulk)

9.21.2.33 double zmagic

Delta function scaling parameter

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

- src/generic/apbs/vpbe.h

9.22 sVpee Struct Reference

Contains public data members for Vpee class/module.

```
#include <vpee.h>
```

Data Fields

- Gem $*$ gm
- int localPartID
- double localPartCenter [3]
- double localPartRadius
- int killFlag
- double killParam
- Vmem $*$ mem

9.22.1 Detailed Description

Contains public data members for Vpee class/module.

Author

Nathan Baker

9.22.2 Field Documentation

9.22.2.1 Gem$*$ gm

Grid manager

9.22.2.2 int killFlag

A flag indicating the method we're using to artificially decrease the error esimate outside the local partition

9.22.2.3 double killParam

A parameter for the error estimate attenuation method

9.22.2.4 double localPartCenter[3]

The coordinates of the center of the local partition

9.22.2.5 int localPartID

The local partition ID: i.e. the partition whose boundary simplices we're keeping track of

9.22.2.6 double localPartRadius

The radius of the circle/sphere which circumscribes the local partition

9.22.2.7 Vmem∗ mem

Memory manager

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

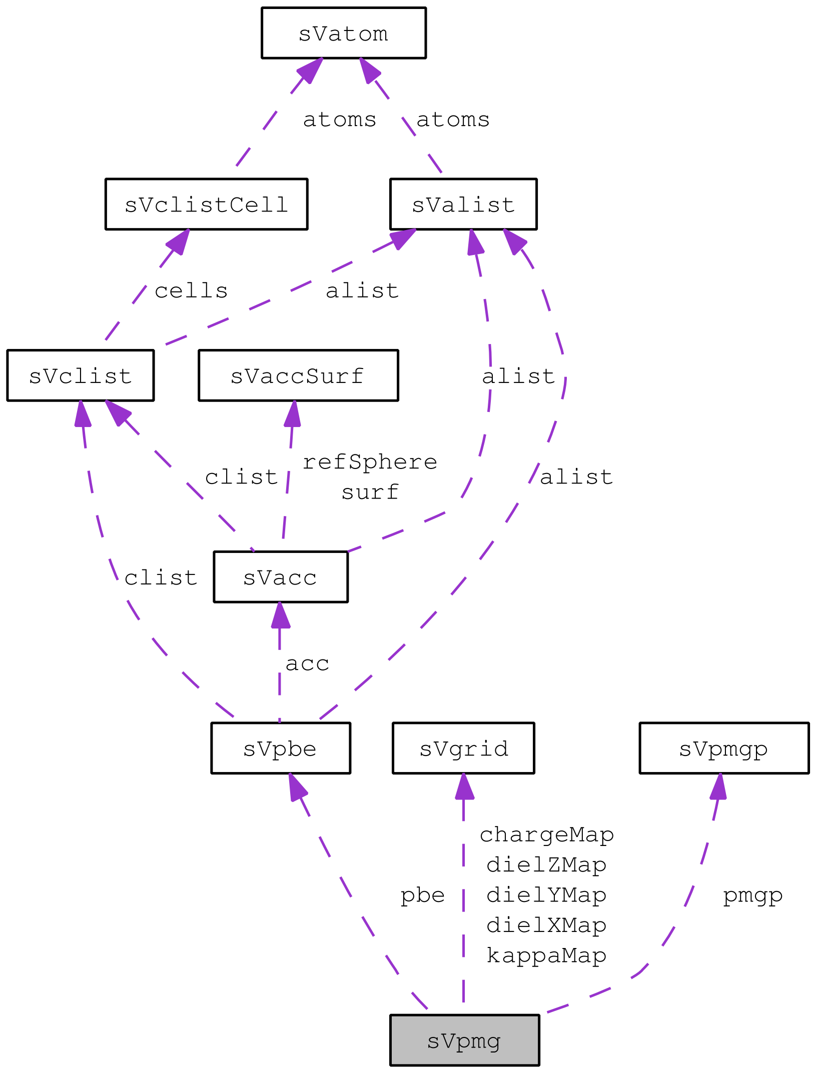
- src/fem/apbs/vpee.h

9.23 sVpmg Struct Reference

Contains public data members for Vpmg class/module.

`#include <vpmg.h>`

Collaboration diagram for sVpmg:



Data Fields

- Vmem ∗ vmem
- Vpmgp ∗ pmgp
- Vpbe ∗ pbe
- double ∗ epsx
- double ∗ epsy
- double ∗ epsz
- double ∗ kappa
- double ∗ charge
- int ∗ iparm
- double ∗ rparm
- int ∗ iwork
- double ∗ rwork
- double ∗ a1cf
- double ∗ a2cf
- double ∗ a3cf
- double ∗ ccf
- double ∗ fcf
- double ∗ tcf
- double ∗ u
- double ∗ xf
- double ∗ yf
- double ∗ zf
- double ∗ gxcf
- double ∗ gycf
- double ∗ gzcf
- double ∗ pvec
- double extDiEnergy
- double extQmEnergy
- double extQfEnergy
- double extNpEnergy
- Vsurf_Meth surfMeth
- double splineWin
- Vchrg_Meth chargeMeth
- Vchrg_Src chargeSrc
- int filled
- int useDielXMap
- Vgrid ∗ dielXMap
- int useDielYMap
- Vgrid ∗ dielYMap
- int useDielZMap
- Vgrid ∗ dielZMap

- int useKappaMap
- Vgrid ∗ kappaMap
- int useChargeMap
- Vgrid ∗ chargeMap

9.23.1 Detailed Description

Contains public data members for Vpmg class/module.

Author

Nathan Baker Many of the routines and macros are borrowed from the main.c driver (written by Mike Holst) provided with the PMG code.

9.23.2 Field Documentation

9.23.2.1 double∗ a1cf

Operator coefficient values (a11) -- this array can be overwritten

9.23.2.2 double∗ a2cf

Operator coefficient values (a22) -- this array can be overwritten

9.23.2.3 double∗ a3cf

Operator coefficient values (a33) -- this array can be overwritten

9.23.2.4 double∗ ccf

Helmholtz term -- this array can be overwritten

9.23.2.5 double∗ charge

Charge map

9.23.2.6 Vgrid∗ chargeMap

External charge distribution map

9.23.2.7 Vchrg_Meth chargeMeth

Charge discretization method

9.23.2.8 Vchrg_Src chargeSrc

Charge source

9.23.2.9 Vgrid* dielXMap

External x-shifted dielectric map

9.23.2.10 Vgrid* dielYMap

External y-shifted dielectric map

9.23.2.11 Vgrid* dielZMap

External z-shifted dielectric map

9.23.2.12 double* epsx

X-shifted dielectric map

9.23.2.13 double* epsy

Y-shifted dielectric map

9.23.2.14 double* epsz

Y-shifted dielectric map

9.23.2.15 double extDiEnergy

Stores contributions to the dielectric energy from regions outside the problem domain

9.23.2.16 double extNpEnergy

Stores contributions to the apolar energy from regions outside the problem domain

9.23.2.17 double extQfEnergy

Stores contributions to the fixed charge energy from regions outside the problem domain

9.23.2.18 double extQmEnergy

Stores contributions to the mobile ion energy from regions outside the problem domain

9.23.2.19 double* fcf

Right-hand side -- this array can be overwritten

9.23.2.20 int filled

Indicates whether Vpmg_fillco has been called

9.23.2.21 double* gxcf

Boundary conditions for x faces

9.23.2.22 double* gycf

Boundary conditions for y faces

9.23.2.23 double* gzcf

Boundary conditions for z faces

9.23.2.24 int* iparm

Passing int parameters to FORTRAN

9.23.2.25 int* iwork

Work array

9.23.2.26 double∗ kappa

Ion accessibility map (0 <= kappa(x) <= 1)

9.23.2.27 Vgrid∗ kappaMap

External kappa map

9.23.2.28 Vpbe∗ pbe

Information about the PBE system

9.23.2.29 Vpmgp∗ pmgp

Parameters

9.23.2.30 double∗ pvec

Partition mask array

9.23.2.31 double∗ rparm

Passing real parameters to FORTRAN

9.23.2.32 double∗ rwork

Work array

9.23.2.33 double splineWin

Spline window parm for surf defs

9.23.2.34 Vsurf_Meth surfMeth

Surface definition method

9.23.2.35 double∗ tcf

True solution

9.23.2.36 double∗ u

Solution

9.23.2.37 int useChargeMap

Indicates whether Vpmg_fillco was called with an external charge distribution map

9.23.2.38 int useDielXMap

Indicates whether Vpmg_fillco was called with an external x-shifted dielectric map

9.23.2.39 int useDielYMap

Indicates whether Vpmg_fillco was called with an external y-shifted dielectric map

9.23.2.40 int useDielZMap

Indicates whether Vpmg_fillco was called with an external z-shifted dielectric map

9.23.2.41 int useKappaMap

Indicates whether Vpmg_fillco was called with an external kappa map

9.23.2.42 Vmem∗ vmem

Memory management object for this class

9.23.2.43 double∗ xf

Mesh point x coordinates

9.23.2.44 double∗ yf

Mesh point y coordinates

9.23.2.45 double∗ zf

Mesh point z coordinates

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

- src/mg/apbs/vpmg.h

9.24 sVpmgp Struct Reference

Contains public data members for Vpmgp class/module.

```
#include <vpmgp.h>
```

Data Fields

- int nx
- int ny
- int nz
- int nlev
- double hx
- double hy
- double hzed
- int nonlin
- int nxc
- int nyc
- int nzc
- int nf
- int nc
- int narrc
- int n_rpc
- int n_iz
- int n_ipc
- int nrwk
- int niwk
- int narr
- int ipkey
- double xcent
- double ycent
- double zcent
- double errtol
- int itmax
- int istop
- int iinfo
- Vbcfl bcfl
- int key
- int iperf
- int meth
- int mgkey
- int nu1
- int nu2

- int mgsmoo
- int mgprol
- int mgcoar
- int mgsolv
- int mgdisc
- double omegal
- double omegan
- int irite
- int ipcon
- double xlen
- double ylen
- double zlen
- double xmin
- double ymin
- double zmin
- double xmax
- double ymax
- double zmax

9.24.1 Detailed Description

Contains public data members for Vpmgp class/module.

Author

Nathan Baker

Bug

Value ipcon does not currently allow for preconditioning in PMG

9.24.2 Field Documentation

9.24.2.1 Vbcfl bcfl

Boundary condition method [default = BCFL_SDH]

9.24.2.2 double errtol

Desired error tolerance [default = 1e-9]

9.24.2.3 double hx

Grid x spacings [no default]

9.24.2.4 double hy

Grid y spacings [no default]

9.24.2.5 double hzed

Grid z spacings [no default]

9.24.2.6 int iinfo

Runtime status messages [default = 1]

- 0: none
- 1: some
- 2: lots
- 3: more

9.24.2.7 int ipcon

Preconditioning method [default = 3]

- 0: diagonal
- 1: ICCG
- 2: ICCGDW
- 3: MICCGDW
- 4: none

9.24.2.8 int iperf

Analysis of the operator [default = 0]

- 0: no

- 1: condition number
- 2: spectral radius
- 3: cond. number & spectral radius

9.24.2.9 int ipkey

Toggles nonlinearity (set by nonlin)

- -2: Size-Modified PBE
- -1: Linearized PBE
- 0: Nonlinear PBE with capped sinh term [default]
- $>$1: Polynomial approximation to sinh, note that ipkey must be odd

9.24.2.10 int irite

FORTRAN output unit [default = 8]

9.24.2.11 int istop

Stopping criterion [default = 1]

- 0: residual
- 1: relative residual
- 2: diff
- 3: errc
- 4: errd
- 5: aerrd

9.24.2.12 int itmax

Maximum number of iters [default = 100]

9.24.2.13 int key

Print solution to file [default = 0]

- 0: no
- 1: yes

9.24.2.14 int meth

Solution method [default = 2]

- 0: conjugate gradient multigrid
- 1: newton
- 2: multigrid
- 3: conjugate gradient
- 4: sucessive overrelaxation
- 5: red-black gauss-seidel
- 6: weighted jacobi
- 7: richardson
- 8: conjugate gradient multigrid aqua
- 9: newton aqua

9.24.2.15 int mgcoar

Coarsening method [default = 2]

- 0: standard
- 1: harmonic
- 2: galerkin

9.24.2.16 int mgdisc

Discretization method [default = 0]

- 0: finite volume
- 1: finite element

9.24.2.17 int mgkey

Multigrid method [default = 0]

- 0: variable v-cycle
- 1: nested iteration

9.24.2.18 int mgprol

Prolongation method [default = 0]

- 0: trilinear
- 1: operator-based
- 2: mod. operator-based

9.24.2.19 int mgsmoo

Smoothing method [default = 1]

- 0: weighted jacobi
- 1: gauss-seidel
- 2: SOR
- 3: richardson
- 4: cghs

9.24.2.20 int mgsolv

Coarse equation solve method [default = 1]

- 0: cghs
- 1: banded linpack

9.24.2.21 int n_ipc

Integer info work array required storage

9.24.2.22 int n_iz

Integer storage parameter (index max)

9.24.2.23 int n_rpc

Real info work array required storage

9.24.2.24 int narr

Array work storage

9.24.2.25 int narrc

Size of vector on coarse level

9.24.2.26 int nc

Number of coarse grid unknowns

9.24.2.27 int nf

Number of fine grid unknowns

9.24.2.28 int niwk

Integer work storage

9.24.2.29 int nlev

Number of mesh levels [no default]

9.24.2.30 int nonlin

Problem type [no default]

- 0: linear
- 1: nonlinear
- 2: linear then nonlinear

9.24.2.31 int nrwk

Real work storage

9.24.2.32 int nu1

Number of pre-smoothings [default = 2]

9.24.2.33 int nu2

Number of post-smoothings [default = 2]

9.24.2.34 int nx

Grid x dimensions [no default]

9.24.2.35 int nxc

Coarse level grid x dimensions

9.24.2.36 int ny

Grid y dimensions [no default]

9.24.2.37 int nyc

Coarse level grid y dimensions

9.24.2.38 int nz

Grid z dimensions [no default]

9.24.2.39 int nzc

Coarse level grid z dimensions

9.24.2.40 double omegal

Linear relax parameter [default = 8e-1]

9.24.2.41 double omegan

Nonlin relax parameter [default = 9e-1]

9.24.2.42 double xcent

Grid x center [0]

9.24.2.43 double xlen

Domain x length

9.24.2.44 double xmax

Domain upper x corner

9.24.2.45 double xmin

Domain lower x corner

9.24.2.46 double ycent

Grid y center [0]

9.24.2.47 double ylen

Domain y length

9.24.2.48 double ymax

Domain upper y corner

9.24.2.49 double ymin

Domain lower y corner

9.24.2.50 double zcent

Grid z center [0]

9.24.2.51 double zlen

Domain z length

9.24.2.52 double zmax

Domain upper z corner

9.24.2.53 double zmin

Domain lower z corner

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

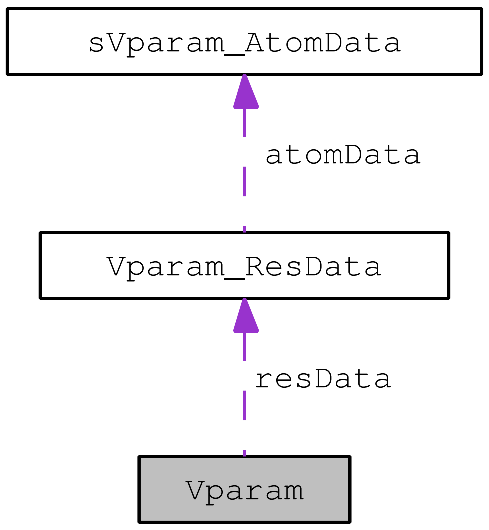
- src/mg/apbs/vpmgp.h

9.25 Vparam Struct Reference

Reads and assigns charge/radii parameters.

`#include <vparam.h>`

Collaboration diagram for Vparam:



Data Fields

- Vmem ∗ vmem
- int nResData
- Vparam_ResData ∗ resData

9.25.1 Detailed Description

Reads and assigns charge/radii parameters.

Author

Nathan Baker

9.25.2 Field Documentation

9.25.2.1 int nResData

Number of Vparam_ResData objects associated with this object

9.25.2.2 Vparam_ResData∗ resData

Array of nResData Vparam_ResData objects

9.25.2.3 Vmem∗ vmem

Memory management object for this class

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

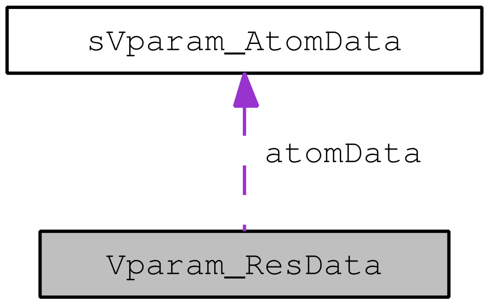
- src/generic/apbs/vparam.h

9.26 Vparam_ResData Struct Reference

ResData sub-class; stores residue data.

`#include <vparam.h>`

Collaboration diagram for Vparam_ResData:



Data Fields

- Vmem ∗ vmem
- char name [VMAX_ARGLEN]
- int nAtomData
- Vparam_AtomData ∗ atomData

9.26.1 Detailed Description

ResData sub-class; stores residue data.

Author

Nathan Baker

9.26.2 Field Documentation

9.26.2.1 Vparam_AtomData∗ atomData

Array of Vparam_AtomData natom objects

9.26.2.2 char name[VMAX_ARGLEN]

Residue name

9.26.2.3 int nAtomData

Number of Vparam_AtomData objects associated with this object

9.26.2.4 Vmem∗ vmem

Pointer to memory manager from Vparam master class

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

- src/generic/apbs/vparam.h

Chapter 10

File Documentation

10.1 doc/license/LICENSE.h File Reference

APBS license.

10.1.1 Detailed Description

APBS license.

Author

Nathan Baker

Version

Id

LICENSE.h 1552 2010-02-10 17:46:27Z yhuang01

Attention

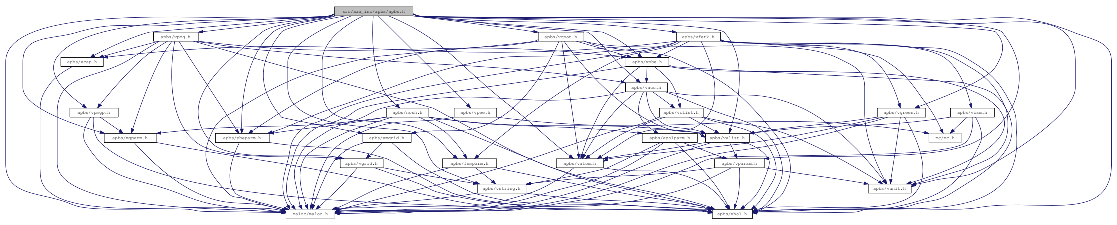
```
 *
 * APBS -- Adaptive Poisson-Boltzmann Solver
 *
 * Nathan A. Baker (baker@biochem.wustl.edu)
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
```

10.2 src/aaa_inc/apbs/apbs.h File Reference

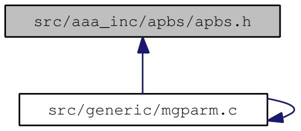
Top-level header for APBS.

```
#include "maloc/maloc.h"
#include "apbs/femparm.h"
#include "apbs/mgparm.h"
#include "apbs/nosh.h"
#include "apbs/pbeparm.h"
#include "apbs/vacc.h"
#include "apbs/valist.h"
#include "apbs/vatom.h"
#include "apbs/vcap.h"
#include "apbs/vgreen.h"
#include "apbs/vhal.h"
#include "apbs/vpbe.h"
#include "apbs/vstring.h"
#include "apbs/vunit.h"
#include "apbs/vparam.h"
#include "apbs/vgrid.h"
#include "apbs/vmgrid.h"
#include "apbs/vopot.h"
#include "apbs/vpmg.h"
#include "apbs/vpmgp.h"
#include "apbs/vfetk.h"
#include "apbs/vpee.h"
```

Include dependency graph for apbs.h:



This graph shows which files directly or indirectly include this file:



10.2.1 Detailed Description

Top-level header for APBS.

Version

Id

apbs.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

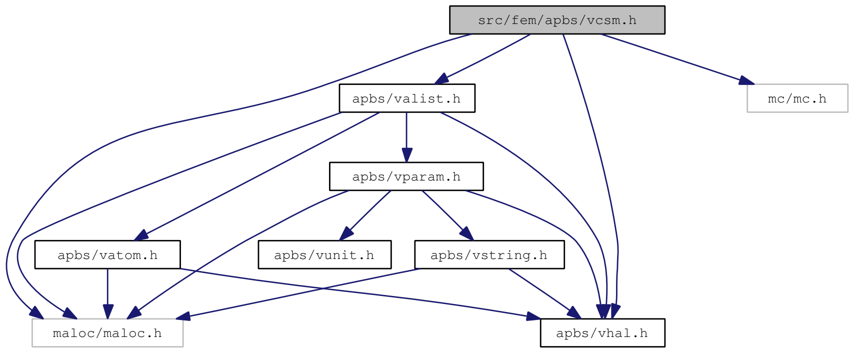
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
```

10.3 src/fem/apbs/vcsm.h File Reference

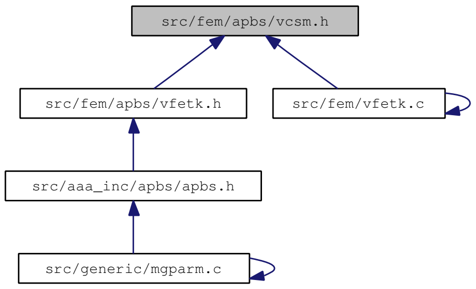
Contains declarations for the Vcsm class.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/valist.h"
#include "mc/mc.h"
```

Include dependency graph for vcsm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVcsm

  Charge-simplex map class.

Typedefs

- typedef struct sVcsm Vcsm

  Declaration of the Vcsm class as the Vcsm structure.

Functions

- VEXTERNC void Gem_setExternalUpdateFunction (Gem ∗thee, void(∗externalUpdate)(SS ∗∗simps, int num))

  External function for FEtk Gem class to use during mesh refinement.

- VEXTERNC Valist ∗ Vcsm_getValist (Vcsm ∗thee)

  Get atom list.

- VEXTERNC int Vcsm_getNumberAtoms (Vcsm ∗thee, int isimp)

  Get number of atoms associated with a simplex.

- VEXTERNC Vatom ∗ Vcsm_getAtom (Vcsm ∗thee, int iatom, int isimp)

  Get particular atom associated with a simplex.

- VEXTERNC int Vcsm_getAtomIndex (Vcsm ∗thee, int iatom, int isimp)

  Get ID of particular atom in a simplex.

- VEXTERNC int Vcsm_getNumberSimplices (Vcsm ∗thee, int iatom)

  Get number of simplices associated with an atom.

- VEXTERNC SS ∗ Vcsm_getSimplex (Vcsm ∗thee, int isimp, int iatom)

  Get particular simplex associated with an atom.

- VEXTERNC int Vcsm_getSimplexIndex (Vcsm ∗thee, int isimp, int iatom)

  Get index particular simplex associated with an atom.

- VEXTERNC unsigned long int Vcsm_memChk (Vcsm ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vcsm ∗ Vcsm_ctor (Valist ∗alist, Gem ∗gm)

  Construct Vcsm object.

- VEXTERNC int Vcsm_ctor2 (Vcsm ∗thee, Valist ∗alist, Gem ∗gm)

  FORTRAN stub to construct Vcsm object.

- VEXTERNC void Vcsm_dtor (Vcsm ∗∗thee)

  Destroy Vcsm object.

- VEXTERNC void Vcsm_dtor2 (Vcsm ∗thee)

  FORTRAN stub to destroy Vcsm object.

- VEXTERNC void Vcsm_init (Vcsm ∗thee)

  Initialize charge-simplex map with mesh and atom data.

- VEXTERNC int Vcsm_update (Vcsm ∗thee, SS ∗∗simps, int num)

  Update the charge-simplex and simplex-charge maps after refinement.

10.3.1 Detailed Description

Contains declarations for the Vcsm class.

Version

Id

vcsm.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

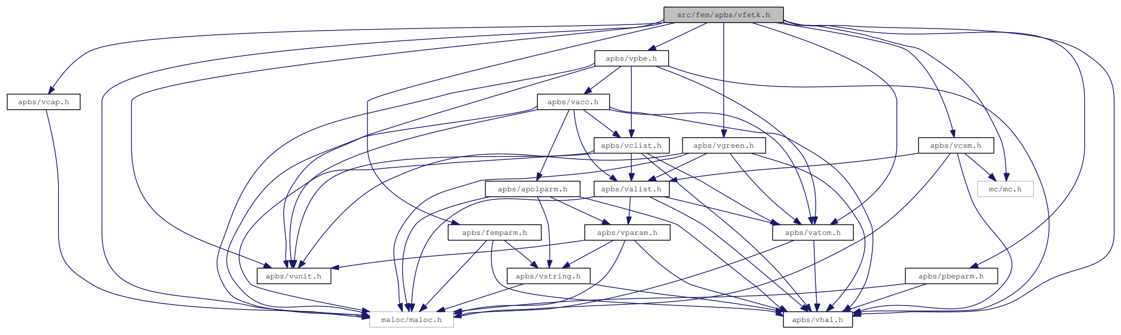
Attention

10.4 src/fem/apbs/vfetk.h File Reference

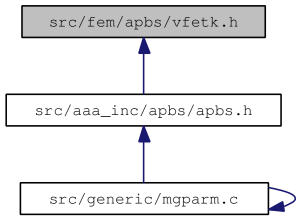
Contains declarations for class Vfetk.

```
#include "maloc/maloc.h"
#include "mc/mc.h"
#include "apbs/vhal.h"
#include "apbs/vatom.h"
#include "apbs/vcsm.h"
#include "apbs/vpbe.h"
#include "apbs/vunit.h"
#include "apbs/vgreen.h"
#include "apbs/vcap.h"
#include "apbs/pbeparm.h"
#include "apbs/femparm.h"
```

Include dependency graph for vfetk.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVfetk

  Contains public data members for Vfetk class/module.

- struct sVfetk_LocalVar

  Vfetk LocalVar subclass.

Typedefs

- typedef enum eVfetk_LsolvType Vfetk_LsolvType

  Declare FEMparm_LsolvType type.

- typedef enum eVfetk_MeshLoad Vfetk_MeshLoad

  Declare FEMparm_GuessType type.

- typedef enum eVfetk_NsolvType Vfetk_NsolvType

  Declare FEMparm_NsolvType type.

- typedef enum eVfetk_GuessType Vfetk_GuessType

  Declare FEMparm_GuessType type.

- typedef enum eVfetk_PrecType Vfetk_PrecType

  Declare FEMparm_GuessType type.

- typedef struct sVfetk Vfetk

  Declaration of the Vfetk class as the Vfetk structure.

- typedef struct sVfetk_LocalVar Vfetk_LocalVar

  Declaration of the Vfetk_LocalVar subclass as the Vfetk_LocalVar structure.

Enumerations

- enum eVfetk_LsolvType { VLT_SLU = 0, VLT_MG = 1, VLT_CG = 2, VLT_-BCG = 3 }

  Linear solver type.

- enum eVfetk_MeshLoad { VML_DIRICUBE, VML_NEUMCUBE, VML_-EXTERNAL }

  Mesh loading operation.

- enum eVfetk_NsolvType { VNT_NEW = 0, VNT_INC = 1, VNT_ARC = 2 }

  Non-linear solver type.

- enum eVfetk_GuessType { VGT_ZERO = 0, VGT_DIRI = 1, VGT_PREV = 2 }

  Initial guess type.

- enum eVfetk_PrecType { VPT_IDEN = 0, VPT_DIAG = 1, VPT_MG = 2 }

  Preconditioner type.

Functions

- VEXTERNC Gem ∗ Vfetk_getGem (Vfetk ∗thee)

  Get a pointer to the Gem (grid manager) object.

- VEXTERNC AM ∗ Vfetk_getAM (Vfetk ∗thee)

  Get a pointer to the AM (algebra manager) object.

- VEXTERNC Vpbe ∗ Vfetk_getVpbe (Vfetk ∗thee)

  Get a pointer to the Vpbe (PBE manager) object.

- VEXTERNC Vcsm ∗ Vfetk_getVcsm (Vfetk ∗thee)

  Get a pointer to the Vcsm (charge-simplex map) object.

- VEXTERNC int Vfetk_getAtomColor (Vfetk ∗thee, int iatom)

  Get the partition information for a particular atom.

- VEXTERNC Vfetk ∗ Vfetk_ctor (Vpbe ∗pbe, Vhal_PBEType type)

  Constructor for Vfetk object.

- VEXTERNC int Vfetk_ctor2 (Vfetk ∗thee, Vpbe ∗pbe, Vhal_PBEType type)

  FORTRAN stub constructor for Vfetk object.

- VEXTERNC void Vfetk_dtor (Vfetk ∗∗thee)

  Object destructor.

- VEXTERNC void Vfetk_dtor2 (Vfetk ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC double ∗ Vfetk_getSolution (Vfetk ∗thee, int ∗length)

Create an array containing the solution (electrostatic potential in units of $k_B T/e$) at the finest mesh level.

- VEXTERNC void Vfetk_setParameters (Vfetk ∗thee, PBEparm ∗pbeparm, FEMparm ∗feparm)

  Set the parameter objects.

- VEXTERNC double Vfetk_energy (Vfetk ∗thee, int color, int nonlin)

  Return the total electrostatic energy.

- VEXTERNC double Vfetk_dqmEnergy (Vfetk ∗thee, int color)

  Get the "mobile charge" and "polarization" contributions to the electrostatic energy.

- VEXTERNC double Vfetk_qfEnergy (Vfetk ∗thee, int color)

  Get the "fixed charge" contribution to the electrostatic energy.

- VEXTERNC unsigned long int Vfetk_memChk (Vfetk ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC void Vfetk_setAtomColors (Vfetk ∗thee)

  Transfer color (partition ID) information frmo a partitioned mesh to the atoms.

- VEXTERNC void Bmat_printHB (Bmat ∗thee, char ∗fname)

  Writes a Bmat to disk in Harwell-Boeing sparse matrix format.

- VEXTERNC Vrc_Codes Vfetk_genCube (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType)

  Construct a rectangular mesh (in the current Vfetk object).

- VEXTERNC Vrc_Codes Vfetk_loadMesh (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType, Vio ∗sock)

  Loads a mesh into the Vfetk (and associated) object(s).

- VEXTERNC PDE ∗ Vfetk_PDE_ctor (Vfetk ∗fetk)

  Constructs the FEtk PDE object.

- VEXTERNC int Vfetk_PDE_ctor2 (PDE ∗thee, Vfetk ∗fetk)

  Intializes the FEtk PDE object.

- VEXTERNC void Vfetk_PDE_dtor (PDE ∗∗thee)

  Destroys FEtk PDE object.

- VEXTERNC void Vfetk_PDE_dtor2 (PDE ∗thee)

FORTRAN stub: destroys FEtk PDE object.

- VEXTERNC void Vfetk_PDE_initAssemble (PDE ∗thee, int ip[ ], double rp[ ])

  Do once-per-assembly initialization.

- VEXTERNC void Vfetk_PDE_initElement (PDE ∗thee, int elementType, int chart, double tvx[ ][VAPBS_DIM], void ∗data)

  Do once-per-element initialization.

- VEXTERNC void Vfetk_PDE_initFace (PDE ∗thee, int faceType, int chart, double tnvec[ ])

  Do once-per-face initialization.

- VEXTERNC void Vfetk_PDE_initPoint (PDE ∗thee, int pointType, int chart, double txq[ ], double tU[ ], double tdU[ ][VAPBS_DIM])

  Do once-per-point initialization.

- VEXTERNC void Vfetk_PDE_Fu (PDE ∗thee, int key, double F[ ])

  Evaluate strong form of PBE. For interior points, this is:

  $$-\nabla \cdot \epsilon \nabla u + b(u) - f$$

  where $b(u)$ is the (possibly nonlinear) mobile ion term and f is the source charge distribution term (for PBE) or the induced surface charge distribution (for RPBE). For an interior-boundary (simplex face) point, this is:

  $$[\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0+} - [\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0-}$$

  where $n(x)$ is the normal to the simplex face and the term represents the jump in dielectric displacement across the face. There is no outer-boundary contribution for this problem.

- VEXTERNC double Vfetk_PDE_Fu_v (PDE ∗thee, int key, double V[ ], double dV[ ][VAPBS_DIM])

  This is the weak form of the PBE; i.e. the strong form integrated with a test function to give:

  $$\int_\Omega [\epsilon \nabla u \cdot \nabla v + b(u)v - fv]\, dx$$

  where $b(u)$ denotes the mobile ion term.

- VEXTERNC double Vfetk_PDE_DFu_wv (PDE ∗thee, int key, double W[ ], double dW[ ][VAPBS_DIM], double V[ ], double dV[ ][VAPBS_DIM])

  This is the linearization of the weak form of the PBE; e.g., for use in a Newton iteration. This is the functional linearization of the strong form integrated with a test function to give:

  $$\int_\Omega \left[\epsilon \nabla w \cdot \nabla v + b'(u)wv - fv\right] dx$$

where $b'(u)$ denotes the functional derivation of the mobile ion term.

- VEXTERNC void Vfetk_PDE_delta (PDE ∗thee, int type, int chart, double txq[ ], void ∗user, double F[ ])

  Evaluate a (discretized) delta function source term at the given point.

- VEXTERNC void Vfetk_PDE_u_D (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the Dirichlet boundary condition at the given point.

- VEXTERNC void Vfetk_PDE_u_T (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the "true solution" at the given point for comparison with the numerical solution.

- VEXTERNC void Vfetk_PDE_bisectEdge (int dim, int dimII, int edgeType, int chart[ ], double vx[ ][VAPBS_DIM])

  Define the way manifold edges are bisected.

- VEXTERNC void Vfetk_PDE_mapBoundary (int dim, int dimII, int vertexType, int chart, double vx[VAPBS_DIM])

  Map a boundary point to some pre-defined shape.

- VEXTERNC int Vfetk_PDE_markSimplex (int dim, int dimII, int simplexType, int faceType[VAPBS_NVS], int vertexType[VAPBS_NVS], int chart[ ], double vx[ ][VAPBS_DIM], void ∗simplex)

  User-defined error estimator -- in our case, a geometry-based refinement method; forcing simplex refinement at the dielectric boundary and (for non-regularized PBE) the charges.

- VEXTERNC void Vfetk_PDE_oneChart (int dim, int dimII, int objType, int chart[ ], double vx[ ][VAPBS_DIM], int dimV)

  Unify the chart for different coordinate systems -- a no-op for us.

- VEXTERNC double Vfetk_PDE_Ju (PDE ∗thee, int key)

  Energy functional. This returns the energy (less delta function terms) in the form:

$$c^{-1}/2\int(\epsilon(\nabla u)^2+\kappa^2(coshu-1))dx$$

  for a 1:1 electrolyte where c is the output from Vpbe_getZmagic.

- VEXTERNC void Vfetk_externalUpdateFunction (SS ∗∗simps, int num)

  External hook to simplex subdivision routines in Gem. Called each time a simplex is subdivided (we use it to update the charge-simplex map).

- VEXTERNC int Vfetk_PDE_simplexBasisInit (int key, int dim, int comp, int ∗ndof, int dof[ ])

  Initialize the bases for the trial or the test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VEXTERNC void Vfetk_PDE_simplexBasisForm (int key, int dim, int comp, int pdkey, double xq[ ], double basis[ ])

  Evaluate the bases for the trial or test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VEXTERNC void Vfetk_readMesh (Vfetk ∗thee, int skey, Vio ∗sock)

  Read in mesh and initialize associated internal structures.

- VEXTERNC void Vfetk_dumpLocalVar ()

  Debugging routine to print out local variables used by PDE object.

- VEXTERNC int Vfetk_fillArray (Vfetk ∗thee, Bvec ∗vec, Vdata_Type type)

  Fill an array with the specified data.

- VEXTERNC int Vfetk_write (Vfetk ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, Bvec ∗vec, Vdata_Format format)

  Write out data.

- VEXTERNC Vrc_Codes Vfetk_loadGem (Vfetk ∗thee, Gem ∗gm)

  Load a Gem geometry manager object into Vfetk.

10.4.1 Detailed Description

Contains declarations for class Vfetk.

Version

Id

vfetk.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

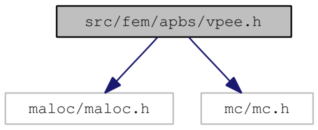
10.5 src/fem/apbs/vpee.h File Reference

Contains declarations for class Vpee.

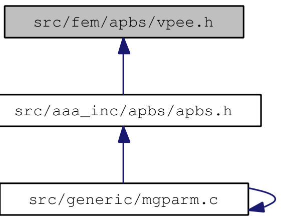
`#include "maloc/maloc.h"`

`#include "mc/mc.h"`

Include dependency graph for vpee.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVpee

  Contains public data members for Vpee class/module.

Typedefs

- typedef struct sVpee Vpee

  Declaration of the Vpee class as the Vpee structure.

Functions

- VEXTERNC Vpee ∗ Vpee_ctor (Gem ∗gm, int localPartID, int killFlag, double killParam)

  Construct the Vpee object.

- VEXTERNC int Vpee_ctor2 (Vpee ∗thee, Gem ∗gm, int localPartID, int killFlag, double killParam)

  FORTRAN stub to construct the Vpee object.

- VEXTERNC void Vpee_dtor (Vpee ∗∗thee)

  Object destructor.

- VEXTERNC void Vpee_dtor2 (Vpee ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vpee_markRefine (Vpee ∗thee, AM ∗am, int level, int akey, int rcol, double etol, int bkey)

  Mark simplices for refinement based on attenuated error estimates.

- VEXTERNC int Vpee_numSS (Vpee ∗thee)

  Returns the number of simplices in the local partition.

10.5.1 Detailed Description

Contains declarations for class Vpee.

Version

Id

vpee.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

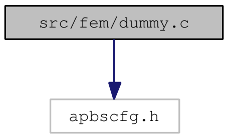
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
```

10.6 src/fem/dummy.c File Reference

Give libtool something to do.

`#include "apbscfg.h"`

Include dependency graph for dummy.c:



Functions

- int **APBSFEM_dummy** (int i)

10.6.1 Detailed Description

Give libtool something to do.

Author

Nathan Baker

Version

Id

dummy.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

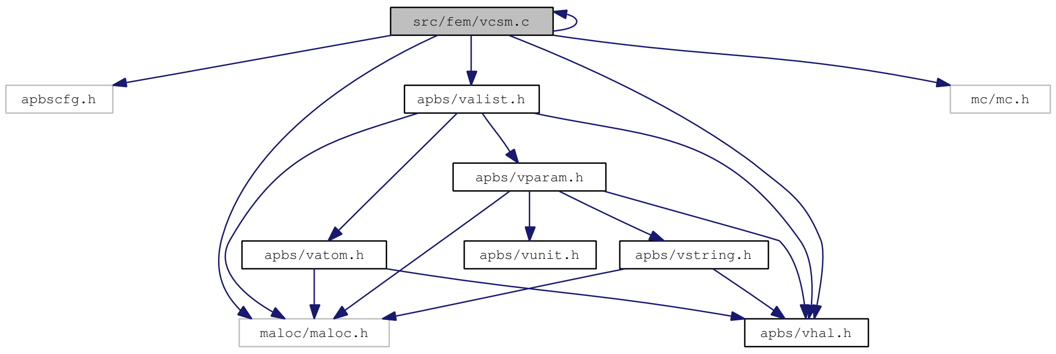
```
 *
 * APBS -- Adaptive Poisson-Boltzmann Solver
 *
 * Nathan A. Baker (baker@biochem.wustl.edu)
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
```

10.7 src/fem/vcsm.c File Reference

Class Vcsm methods.

```
#include "apbscfg.h"
#include "apbs/vcsm.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/valist.h"
#include "mc/mc.h"
```

Include dependency graph for vcsm.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC Valist ∗ Vcsm_getValist (Vcsm ∗thee)

  Get atom list.

- VPUBLIC int Vcsm_getNumberAtoms (Vcsm ∗thee, int isimp)

  Get number of atoms associated with a simplex.

- VPUBLIC Vatom ∗ Vcsm_getAtom (Vcsm ∗thee, int iatom, int isimp)

  Get particular atom associated with a simplex.

- VPUBLIC int Vcsm_getAtomIndex (Vcsm ∗thee, int iatom, int isimp)

  Get ID of particular atom in a simplex.

- VPUBLIC int Vcsm_getNumberSimplices (Vcsm ∗thee, int iatom)

  Get number of simplices associated with an atom.

- VPUBLIC SS ∗ Vcsm_getSimplex (Vcsm ∗thee, int isimp, int iatom)

  Get particular simplex associated with an atom.

- VPUBLIC int Vcsm_getSimplexIndex (Vcsm ∗thee, int isimp, int iatom)

  Get index particular simplex associated with an atom.

- VPUBLIC unsigned long int Vcsm_memChk (Vcsm ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC Vcsm ∗ Vcsm_ctor (Valist ∗alist, Gem ∗gm)

  Construct Vcsm object.

- VPUBLIC int Vcsm_ctor2 (Vcsm ∗thee, Valist ∗alist, Gem ∗gm)

  FORTRAN stub to construct Vcsm object.

- VPUBLIC void Vcsm_init (Vcsm ∗thee)

  Initialize charge-simplex map with mesh and atom data.

- VPUBLIC void Vcsm_dtor (Vcsm ∗∗thee)

  Destroy Vcsm object.

- VPUBLIC void Vcsm_dtor2 (Vcsm ∗thee)

  FORTRAN stub to destroy Vcsm object.

- VPUBLIC int Vcsm_update (Vcsm ∗thee, SS ∗∗simps, int num)

  Update the charge-simplex and simplex-charge maps after refinement.

10.7.1 Detailed Description

Class Vcsm methods.

Author

Nathan Baker

Version

Id

vcsm.c 1552 2010-02-10 17:46:27Z yhuang01

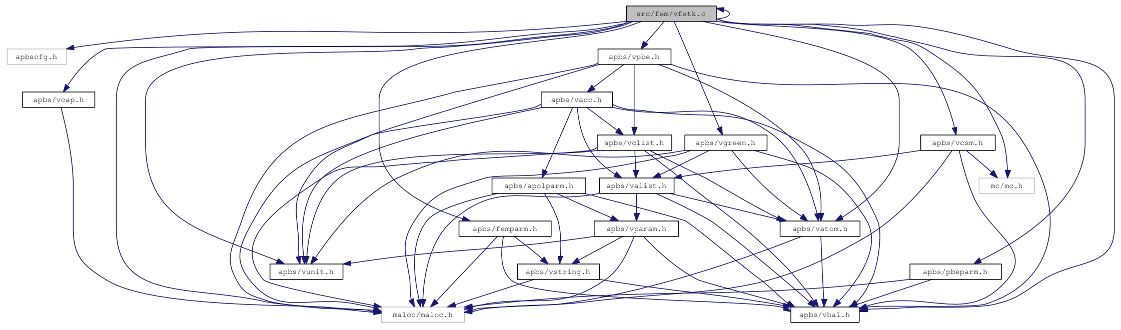
Attention

10.8 src/fem/vfetk.c File Reference

Class Vfetk methods.

```
#include "apbscfg.h"
#include "apbs/vfetk.h"
#include "maloc/maloc.h"
#include "mc/mc.h"
#include "apbs/vhal.h"
#include "apbs/vatom.h"
#include "apbs/vcsm.h"
#include "apbs/vpbe.h"
#include "apbs/vunit.h"
#include "apbs/vgreen.h"
#include "apbs/vcap.h"
#include "apbs/pbeparm.h"
#include "apbs/femparm.h"
```

Include dependency graph for vfetk.c:



This graph shows which files directly or indirectly include this file:

Defines

- #define **VMAXLOCALCOLORSDONTREUSETHISVARIABLE** 1024
- #define VRINGMAX 1000

  Maximum number of simplices in a simplex ring.

- #define VATOMMAX 1000000

  Maximum number of atoms associated with a vertex.

Functions

- VPRIVATE double **Vfetk_qfEnergyAtom** (Vfetk ∗thee, int iatom, int color, double ∗sol)
- VPRIVATE double **diel** ()
- VPRIVATE double **ionacc** ()
- VPRIVATE double **smooth** (int nverts, double dist[VAPBS_NVS], double coeff[VAPBS_NVS], int meth)
- VPRIVATE double **debye_U** (Vpbe ∗pbe, int d, double x[ ])
- VPRIVATE double **debye_Udiff** (Vpbe ∗pbe, int d, double x[ ])
- VPRIVATE void **coulomb** (Vpbe ∗pbe, int d, double x[ ], double eps, double ∗U, double dU[ ], double ∗d2U)
- VPRIVATE void **init_2DP1** (int dimIS[ ], int ∗ndof, int dof[ ], double c[ ][VMAXP], double cx[ ][VMAXP], double cy[ ][VMAXP], double cz[ ][VMAXP])
- VPRIVATE void **init_3DP1** (int dimIS[ ], int ∗ndof, int dof[ ], double c[ ][VMAXP], double cx[ ][VMAXP], double cy[ ][VMAXP], double cz[ ][VMAXP])
- VPRIVATE void **setCoef** (int numP, double c[ ][VMAXP], double cx[ ][VMAXP], double cy[ ][VMAXP], double cz[ ][VMAXP], int ic[ ][VMAXP], int icx[ ][VMAXP], int icy[ ][VMAXP], int icz[ ][VMAXP])
- VPRIVATE void **polyEval** (int numP, double p[ ], double c[ ][VMAXP], double xv[ ])
- VPUBLIC Gem ∗ Vfetk_getGem (Vfetk ∗thee)

  Get a pointer to the Gem (grid manager) object.

- VPUBLIC AM ∗ Vfetk_getAM (Vfetk ∗thee)

  Get a pointer to the AM (algebra manager) object.

- VPUBLIC Vpbe ∗ Vfetk_getVpbe (Vfetk ∗thee)

  Get a pointer to the Vpbe (PBE manager) object.

- VPUBLIC Vcsm ∗ Vfetk_getVcsm (Vfetk ∗thee)

Get a pointer to the Vcsm (charge-simplex map) object.

- VPUBLIC int Vfetk_getAtomColor (Vfetk ∗thee, int iatom)

  Get the partition information for a particular atom.

- VPUBLIC Vfetk ∗ Vfetk_ctor (Vpbe ∗pbe, Vhal_PBEType type)

  Constructor for Vfetk object.

- VPUBLIC int Vfetk_ctor2 (Vfetk ∗thee, Vpbe ∗pbe, Vhal_PBEType type)

  FORTRAN stub constructor for Vfetk object.

- VPUBLIC void Vfetk_setParameters (Vfetk ∗thee, PBEparm ∗pbeparm, FEMparm ∗feparm)

  Set the parameter objects.

- VPUBLIC void Vfetk_dtor (Vfetk ∗∗thee)

  Object destructor.

- VPUBLIC void Vfetk_dtor2 (Vfetk ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC double ∗ Vfetk_getSolution (Vfetk ∗thee, int ∗length)

  Create an array containing the solution (electrostatic potential in units of $k_B T/e$) at the finest mesh level.

- VPUBLIC double Vfetk_energy (Vfetk ∗thee, int color, int nonlin)

  Return the total electrostatic energy.

- VPUBLIC double Vfetk_qfEnergy (Vfetk ∗thee, int color)

  Get the "fixed charge" contribution to the electrostatic energy.

- VPUBLIC double Vfetk_dqmEnergy (Vfetk ∗thee, int color)

  Get the "mobile charge" and "polarization" contributions to the electrostatic energy.

- VPUBLIC void Vfetk_setAtomColors (Vfetk ∗thee)

  Transfer color (partition ID) information frmo a partitioned mesh to the atoms.

- VPUBLIC unsigned long int Vfetk_memChk (Vfetk ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC Vrc_Codes Vfetk_genCube (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType)

  Construct a rectangular mesh (in the current Vfetk object).

- VPUBLIC Vrc_Codes Vfetk_loadMesh (Vfetk ∗thee, double center[3], double length[3], Vfetk_MeshLoad meshType, Vio ∗sock)

  Loads a mesh into the Vfetk (and associated) object(s).

- VPUBLIC void Bmat_printHB (Bmat ∗thee, char ∗fname)

  Writes a Bmat to disk in Harwell-Boeing sparse matrix format.

- VPUBLIC PDE ∗ Vfetk_PDE_ctor (Vfetk ∗fetk)

  Constructs the FEtk PDE object.

- VPUBLIC int Vfetk_PDE_ctor2 (PDE ∗thee, Vfetk ∗fetk)

  Intializes the FEtk PDE object.

- VPUBLIC void Vfetk_PDE_dtor (PDE ∗∗thee)

  Destroys FEtk PDE object.

- VPUBLIC void Vfetk_PDE_dtor2 (PDE ∗thee)

  FORTRAN stub: destroys FEtk PDE object.

- VPUBLIC void Vfetk_PDE_initAssemble (PDE ∗thee, int ip[ ], double rp[ ])

  Do once-per-assembly initialization.

- VPUBLIC void **Vfetk_PDE_initElement** (PDE ∗thee, int elementType, int chart, double tvx[ ][3], void ∗data)
- VPUBLIC void Vfetk_PDE_initFace (PDE ∗thee, int faceType, int chart, double tnvec[ ])

  Do once-per-face initialization.

- VPUBLIC void **Vfetk_PDE_initPoint** (PDE ∗thee, int pointType, int chart, double txq[ ], double tU[ ], double tdU[ ][3])
- VPUBLIC void Vfetk_PDE_Fu (PDE ∗thee, int key, double F[ ])

  Evaluate strong form of PBE. For interior points, this is:

  $$-\nabla \cdot \epsilon \nabla u + b(u) - f$$

  where $b(u)$ is the (possibly nonlinear) mobile ion term and f is the source charge distribution term (for PBE) or the induced surface charge distribution (for RPBE). For an interior-boundary (simplex face) point, this is:

  $$[\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^+} - [\epsilon(x)\nabla u(x) \cdot n(x)]_{x=0^-}$$

  where $n(x)$ is the normal to the simplex face and the term represents the jump in dielectric displacement across the face. There is no outer-boundary contribution for this problem.

- VPUBLIC double Vfetk_PDE_Fu_v (PDE ∗thee, int key, double V[ ], double dV[ ][VAPBS_DIM])

  This is the weak form of the PBE; i.e. the strong form integrated with a test function to give:

$$\int_{\Omega} [\epsilon \nabla u \cdot \nabla v + b(u)v - fv]\, dx$$

  where $b(u)$ denotes the mobile ion term.

- VPUBLIC double **Vfetk_PDE_DFu_wv** (PDE ∗thee, int key, double W[ ], double dW[ ][VAPBS_DIM], double V[ ], double dV[ ][3])
- VPUBLIC void Vfetk_PDE_delta (PDE ∗thee, int type, int chart, double txq[ ], void ∗user, double F[ ])

  Evaluate a (discretized) delta function source term at the given point.

- VPUBLIC void Vfetk_PDE_u_D (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the Dirichlet boundary condition at the given point.

- VPUBLIC void Vfetk_PDE_u_T (PDE ∗thee, int type, int chart, double txq[ ], double F[ ])

  Evaluate the "true solution" at the given point for comparison with the numerical solution.

- VPUBLIC void **Vfetk_PDE_bisectEdge** (int dim, int dimII, int edgeType, int chart[ ], double vx[ ][3])
- VPUBLIC void **Vfetk_PDE_mapBoundary** (int dim, int dimII, int vertexType, int chart, double vx[3])
- VPUBLIC int **Vfetk_PDE_markSimplex** (int dim, int dimII, int simplexType, int faceType[VAPBS_NVS], int vertexType[VAPBS_NVS], int chart[ ], double vx[ ][3], void ∗simplex)
- VPUBLIC void **Vfetk_PDE_oneChart** (int dim, int dimII, int objType, int chart[ ], double vx[ ][3], int dimV)
- VPUBLIC double Vfetk_PDE_Ju (PDE ∗thee, int key)

  Energy functional. This returns the energy (less delta function terms) in the form:

$$c^{-1}/2 \int (\epsilon(\nabla u)^2 + \kappa^2(cosh u - 1))dx$$

  for a 1:1 electrolyte where c is the output from Vpbe_getZmagic.

- VPUBLIC void Vfetk_externalUpdateFunction (SS ∗∗simps, int num)

  External hook to simplex subdivision routines in Gem. Called each time a simplex is subdivided (we use it to update the charge-simplex map).

- VPUBLIC int Vfetk_PDE_simplexBasisInit (int key, int dim, int comp, int ∗ndof, int dof[ ])

    Initialize the bases for the trial or the test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VPUBLIC void Vfetk_PDE_simplexBasisForm (int key, int dim, int comp, int pdkey, double xq[ ], double basis[ ])

    Evaluate the bases for the trial or test space, for a particular component of the system, at all quadrature points on the master simplex element.

- VPUBLIC void Vfetk_dumpLocalVar ()

    Debugging routine to print out local variables used by PDE object.

- VPUBLIC int Vfetk_fillArray (Vfetk ∗thee, Bvec ∗vec, Vdata_Type type)

    Fill an array with the specified data.

- VPUBLIC int Vfetk_write (Vfetk ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, Bvec ∗vec, Vdata_Format format)

    Write out data.

Variables

- VPRIVATE Vfetk_LocalVar **var**
- VPRIVATE char ∗ **diriCubeString**
- VPRIVATE char ∗ **neumCubeString**
- VPRIVATE int **dim_2DP1** = 3
- VPRIVATE int **lgr_2DP1** [3][VMAXP]
- VPRIVATE int **lgr_2DP1x** [3][VMAXP]
- VPRIVATE int **lgr_2DP1y** [3][VMAXP]
- VPRIVATE int **lgr_2DP1z** [3][VMAXP]
- VPRIVATE int **dim_3DP1** = VAPBS_NVS
- VPRIVATE int **lgr_3DP1** [VAPBS_NVS][VMAXP]
- VPRIVATE int **lgr_3DP1x** [VAPBS_NVS][VMAXP]
- VPRIVATE int **lgr_3DP1y** [VAPBS_NVS][VMAXP]
- VPRIVATE int **lgr_3DP1z** [VAPBS_NVS][VMAXP]
- VPRIVATE const int **P_DEG** = 1
- VPRIVATE int **numP**
- VPRIVATE double **c** [VMAXP][VMAXP]
- VPRIVATE double **cx** [VMAXP][VMAXP]
- VPRIVATE double **cy** [VMAXP][VMAXP]
- VPRIVATE double **cz** [VMAXP][VMAXP]

10.8.1 Detailed Description

Class Vfetk methods.

Author

Nathan Baker

Version

Id

vfetk.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

10.8.2 Variable Documentation

10.8.2.1 VPRIVATE int lgr_2DP1[3][VMAXP]

Initial value:

```
 {


{  2, -2, -2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 }
}
```

10.8.2.2 VPRIVATE int lgr_2DP1x[3][VMAXP]

Initial value:

```
 {


{ -2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  2,  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,  0 }
}
```

10.8.2.3 VPRIVATE int lgr_2DP1y[3][VMAXP]

Initial value:

```
 {


{ -2,  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,  0 },
{  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 }
}
```

10.8.2.4 VPRIVATE int lgr_2DP1z[3][VMAXP]

Initial value:

```
 {


{  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,  0,  0 },
{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 }
}
```

10.8.2.5 VPRIVATE int lgr_3DP1[VAPBS_NVS][VMAXP]

Initial value:

```
 {


{  2, -2, -2, -2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 }
}
```

10.8.2.6 VPRIVATE int lgr_3DP1x[VAPBS_NVS][VMAXP]

Initial value:

```
 {


{ -2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  2,  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,  0 },
{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 }
}
```

10.8.2.7 VPRIVATE int lgr_3DP1y[VAPBS_NVS][VMAXP]

Initial value:

```
 {


{ -2,  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,  0 },
{  2,  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,  0 }
}
```

10.8.2.8 VPRIVATE int lgr_3DP1z[VAPBS_NVS][VMAXP]

Initial value:

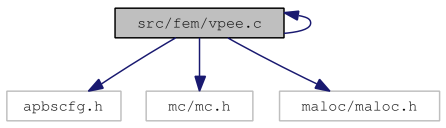
```
 {


{ -2,  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,  0 },
{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
{  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 },
}
```

10.9 src/fem/vpee.c File Reference

Class Vpee methods.

```
#include "apbscfg.h"
#include "mc/mc.h"
#include "apbs/vpee.h"
#include "maloc/maloc.h"
```

Include dependency graph for vpee.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPRIVATE int **Vpee_userDefined** (Vpee ∗thee, SS ∗sm)
- VPRIVATE int **Vpee_ourSimp** (Vpee ∗thee, SS ∗sm, int rcol)
- VEXTERNC double **Aprx_estNonlinResid** (Aprx ∗thee, SS ∗sm, Bvec ∗u, Bvec ∗ud, Bvec ∗f)
- VEXTERNC double **Aprx_estLocalProblem** (Aprx ∗thee, SS ∗sm, Bvec ∗u, Bvec ∗ud, Bvec ∗f)
- VEXTERNC double **Aprx_estDualProblem** (Aprx ∗thee, SS ∗sm, Bvec ∗u, Bvec ∗ud, Bvec ∗f)
- VPUBLIC Vpee ∗ Vpee_ctor (Gem ∗gm, int localPartID, int killFlag, double killParam)

  Construct the Vpee object.

- VPUBLIC int Vpee_ctor2 (Vpee ∗thee, Gem ∗gm, int localPartID, int killFlag, double killParam)

  FORTRAN stub to construct the Vpee object.

- VPUBLIC void Vpee_dtor (Vpee ∗∗thee)

  Object destructor.

- VPUBLIC void Vpee_dtor2 (Vpee ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC int Vpee_markRefine (Vpee ∗thee, AM ∗am, int level, int akey, int rcol, double etol, int bkey)

  Mark simplices for refinement based on attenuated error estimates.

- VPUBLIC int Vpee_numSS (Vpee ∗thee)

  Returns the number of simplices in the local partition.

10.9.1 Detailed Description

Class Vpee methods.

Author

Nathan Baker

Version

Id

vpee.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

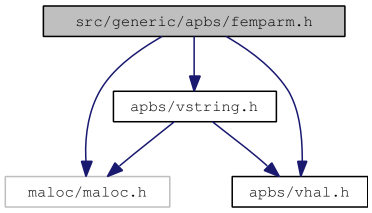
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
*
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* -  Redistributions of source code must retain the above copyright notice, this
* list of conditions and the following disclaimer.
```

10.10 src/generic/apbs/femparm.h File Reference

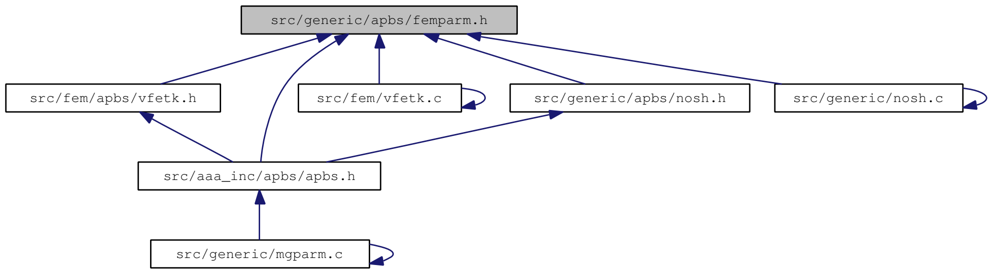
Contains declarations for class APOLparm.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vstring.h"
```

Include dependency graph for femparm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sFEMparm

  Parameter structure for FEM-specific variables from input files.

Typedefs

- typedef enum eFEMparm_EtolType FEMparm_EtolType

  Declare FEparm_EtolType type.

- typedef enum eFEMparm_EstType FEMparm_EstType

Declare FEMparm_EstType type.

- typedef enum eFEMparm_CalcType FEMparm_CalcType

  Declare FEMparm_CalcType type.

- typedef struct sFEMparm FEMparm

  Declaration of the FEMparm class as the FEMparm structure.

Enumerations

- enum eFEMparm_EtolType { FET_SIMP = 0, FET_GLOB = 1, FET_FRAC = 2 }

  Adaptive refinment error estimate tolerance key.

- enum eFEMparm_EstType {
  FRT_UNIF = 0, FRT_GEOM = 1, FRT_RESI = 2, FRT_DUAL = 3,
  FRT_LOCA = 4 }

  Adaptive refinment error estimator method.

- enum eFEMparm_CalcType { FCT_MANUAL, FCT_NONE }

  Calculation type.

Functions

- VEXTERNC FEMparm ∗ FEMparm_ctor (FEMparm_CalcType type)

  Construct FEMparm.

- VEXTERNC int FEMparm_ctor2 (FEMparm ∗thee, FEMparm_CalcType type)

  FORTRAN stub to construct FEMparm.

- VEXTERNC void FEMparm_dtor (FEMparm ∗∗thee)

  Object destructor.

- VEXTERNC void FEMparm_dtor2 (FEMparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC int FEMparm_check (FEMparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void FEMparm_copy (FEMparm ∗thee, FEMparm ∗source)

  Copy target object into thee.

- VEXTERNC Vrc_Codes FEMparm_parseToken (FEMparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

10.10.1 Detailed Description

Contains declarations for class APOLparm. Contains declarations for class FEMparm.

Version

Id

apolparm.h 1564 2010-03-07 14:04:14Z sobolevnrm

Author

Nathan A. Baker

Attention

Version

Id

femparm.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

```
* list of conditions and the following disclaimer.
*
* - Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
*
* - Neither the name of Washington University in St. Louis nor the names of its
* contributors may be used to endorse or promote products derived from this
* software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
* A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*
```

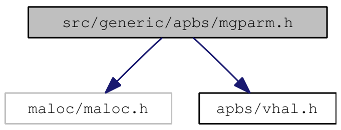
10.11 src/generic/apbs/mgparm.h File Reference

Contains declarations for class MGparm.

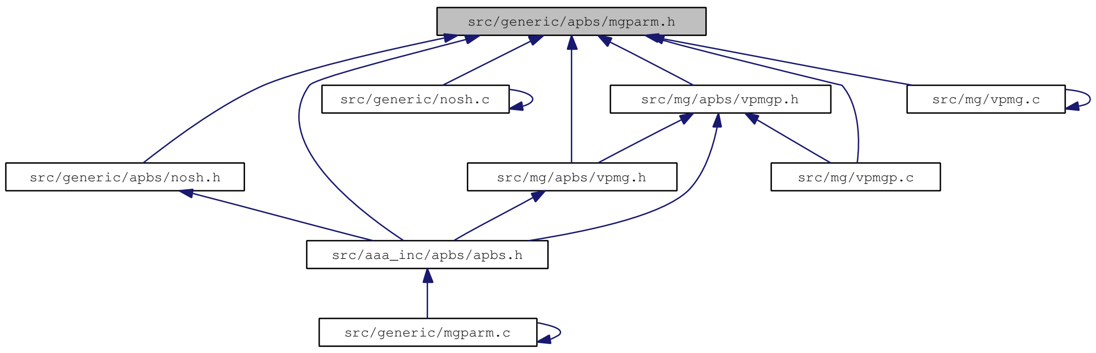
```
#include "maloc/maloc.h"
```

```
#include "apbs/vhal.h"
```

Include dependency graph for mgparm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sMGparm

  Parameter structure for MG-specific variables from input files.

Typedefs

- typedef enum eMGparm_CalcType MGparm_CalcType

  Declare MGparm_CalcType type.

- typedef enum eMGparm_CentMeth MGparm_CentMeth

  Declare MGparm_CentMeth type.

- typedef struct sMGparm MGparm

  Declaration of the MGparm class as the MGparm structure.

Enumerations

- enum eMGparm_CalcType {

  MCT_MANUAL = 0, MCT_AUTO = 1, MCT_PARALLEL = 2, MCT_-DUMMY = 3,

  MCT_NONE = 4 }

  Calculation type.

- enum eMGparm_CentMeth { MCM_POINT = 0, MCM_MOLECULE = 1, MCM_FOCUS = 2 }

  Centering method.

Functions

- VEXTERNC int MGparm_getNx (MGparm ∗thee)

  Get number of grid points in x direction.

- VEXTERNC int MGparm_getNy (MGparm ∗thee)

  Get number of grid points in y direction.

- VEXTERNC int MGparm_getNz (MGparm ∗thee)

  Get number of grid points in z direction.

- VEXTERNC double MGparm_getHx (MGparm ∗thee)

  Get grid spacing in x direction (Å).

- VEXTERNC double MGparm_getHy (MGparm ∗thee)

  Get grid spacing in y direction (Å).

- VEXTERNC double MGparm_getHz (MGparm ∗thee)

  Get grid spacing in z direction (Å).

- VEXTERNC void MGparm_setCenterX (MGparm ∗thee, double x)

  Set center x-coordinate.

- VEXTERNC void MGparm_setCenterY (MGparm ∗thee, double y)

Set center y-coordinate.

- VEXTERNC void MGparm_setCenterZ (MGparm ∗thee, double z)

  Set center z-coordinate.

- VEXTERNC double MGparm_getCenterX (MGparm ∗thee)

  Get center x-coordinate.

- VEXTERNC double MGparm_getCenterY (MGparm ∗thee)

  Get center y-coordinate.

- VEXTERNC double MGparm_getCenterZ (MGparm ∗thee)

  Get center z-coordinate.

- VEXTERNC MGparm ∗ MGparm_ctor (MGparm_CalcType type)

  Construct MGparm object.

- VEXTERNC Vrc_Codes MGparm_ctor2 (MGparm ∗thee, MGparm_CalcType type)

  FORTRAN stub to construct MGparm object.

- VEXTERNC void MGparm_dtor (MGparm ∗∗thee)

  Object destructor.

- VEXTERNC void MGparm_dtor2 (MGparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC Vrc_Codes MGparm_check (MGparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void MGparm_copy (MGparm ∗thee, MGparm ∗parm)

  Copy MGparm object into thee.

- VEXTERNC Vrc_Codes MGparm_parseToken (MGparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

10.11.1 Detailed Description

Contains declarations for class MGparm.

Version

Id

mgparm.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

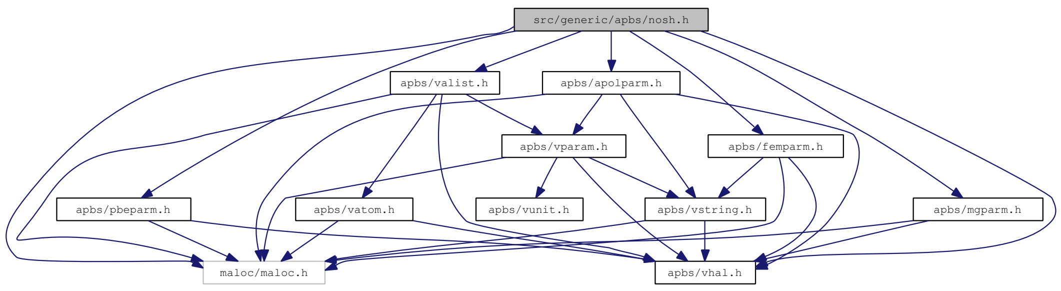
Attention

10.12 src/generic/apbs/nosh.h File Reference

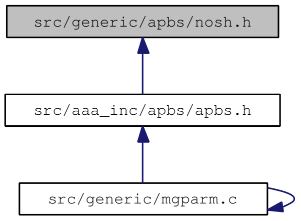
Contains declarations for class NOsh.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/pbeparm.h"
#include "apbs/mgparm.h"
#include "apbs/femparm.h"
#include "apbs/apolparm.h"
#include "apbs/valist.h"
```

Include dependency graph for nosh.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sNOsh_calc

  Calculation class for use when parsing fixed format input files.

- struct sNOsh

Class for parsing fixed format input files.

Defines

- #define NOSH_MAXMOL 20

  Maximum number of molecules in a run.

- #define NOSH_MAXCALC 20

  Maximum number of calculations in a run.

- #define NOSH_MAXPRINT 20

  Maximum number of PRINT statements in a run.

- #define NOSH_MAXPOP 20

  Maximum number of operations in a PRINT statement.

Typedefs

- typedef enum eNOsh_MolFormat NOsh_MolFormat

  Declare NOsh_MolFormat type.

- typedef enum eNOsh_CalcType NOsh_CalcType

  Declare NOsh_CalcType type.

- typedef enum eNOsh_ParmFormat NOsh_ParmFormat

  Declare NOsh_ParmFormat type.

- typedef enum eNOsh_PrintType NOsh_PrintType

  Declare NOsh_PrintType type.

- typedef struct sNOsh_calc NOsh_calc

  Declaration of the NOsh_calc class as the NOsh_calc structure.

- typedef struct sNOsh NOsh

  Declaration of the NOsh class as the NOsh structure.

Enumerations

- enum eNOsh_MolFormat { NMF_PQR = 0, NMF_PDB = 1, NMF_XML = 2 }

  Molecule file format types.

- enum eNOsh_CalcType { NCT_MG = 0, NCT_FEM = 1, NCT_APOL = 2 }

  NOsh calculation types.

- enum eNOsh_ParmFormat { NPF_FLAT = 0, NPF_XML = 1 }

  Parameter file format types.

- enum eNOsh_PrintType {

  NPT_ENERGY = 0, NPT_FORCE = 1, NPT_ELECENERGY, NPT_-ELECFORCE,

  NPT_APOLENERGY, NPT_APOLFORCE }

  NOsh print types.

Functions

- VEXTERNC char ∗ NOsh_getMolpath (NOsh ∗thee, int imol)

  Returns path to specified molecule.

- VEXTERNC char ∗ NOsh_getDielXpath (NOsh ∗thee, int imap)

  Returns path to specified x-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getDielYpath (NOsh ∗thee, int imap)

  Returns path to specified y-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getDielZpath (NOsh ∗thee, int imap)

  Returns path to specified z-shifted dielectric map.

- VEXTERNC char ∗ NOsh_getKappapath (NOsh ∗thee, int imap)

  Returns path to specified kappa map.

- VEXTERNC char ∗ NOsh_getChargepath (NOsh ∗thee, int imap)

  Returns path to specified charge distribution map.

- VEXTERNC NOsh_calc ∗ NOsh_getCalc (NOsh ∗thee, int icalc)

  Returns specified calculation object.

- VEXTERNC int NOsh_getDielfmt (NOsh ∗thee, int imap)

  Returns format of specified dielectric map.

- VEXTERNC int NOsh_getKappafmt (NOsh ∗thee, int imap)

  Returns format of specified kappa map.

- VEXTERNC int NOsh_getChargefmt (NOsh ∗thee, int imap)

  Returns format of specified charge map.

- VEXTERNC NOsh_PrintType NOsh_printWhat (NOsh ∗thee, int iprint)

  Return an integer ID of the observable to print (.

- VEXTERNC char ∗ NOsh_elecname (NOsh ∗thee, int ielec)

  Return an integer mapping of an ELEC statement to a calculation ID (.

- VEXTERNC int NOsh_elec2calc (NOsh ∗thee, int icalc)

  Return the name of an elec statement.

- VEXTERNC int NOsh_apol2calc (NOsh ∗thee, int icalc)

  Return the name of an apol statement.

- VEXTERNC int NOsh_printNarg (NOsh ∗thee, int iprint)

  Return number of arguments to PRINT statement (.

- VEXTERNC int NOsh_printOp (NOsh ∗thee, int iprint, int iarg)

  Return integer ID for specified operation (.

- VEXTERNC int NOsh_printCalc (NOsh ∗thee, int iprint, int iarg)

  Return calculation ID for specified PRINT statement (.

- VEXTERNC NOsh ∗ NOsh_ctor (int rank, int size)

  Construct NOsh.

- VEXTERNC NOsh_calc ∗ NOsh_calc_ctor (NOsh_CalcType calcType)

  Construct NOsh_calc.

- VEXTERNC int NOsh_calc_copy (NOsh_calc ∗thee, NOsh_calc ∗source)

  Copy NOsh_calc object into thee.

- VEXTERNC void NOsh_calc_dtor (NOsh_calc ∗∗thee)

  Object destructor.

- VEXTERNC int NOsh_ctor2 (NOsh ∗thee, int rank, int size)

  FORTRAN stub to construct NOsh.

- VEXTERNC void NOsh_dtor (NOsh ∗∗thee)

  Object destructor.

- VEXTERNC void NOsh_dtor2 (NOsh ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC int NOsh_parseInput (NOsh ∗thee, Vio ∗sock)

  Parse an input file from a socket.

- VEXTERNC int NOsh_parseInputFile (NOsh ∗thee, char ∗filename)

  Parse an input file only from a file.

- VEXTERNC int NOsh_setupElecCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of electrostatics calculations.

- VEXTERNC int NOsh_setupApolCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of non-polar calculations.

10.12.1 Detailed Description

Contains declarations for class NOsh.

Version

Id

nosh.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
```

```
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * -  Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * - Neither the name of Washington University in St. Louis nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

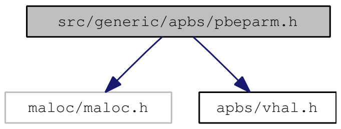
10.13 src/generic/apbs/pbeparm.h File Reference

Contains declarations for class PBEparm.

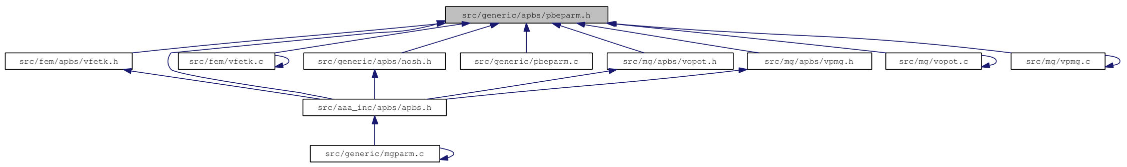
`#include "maloc/maloc.h"`

`#include "apbs/vhal.h"`

Include dependency graph for pbeparm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sPBEparm

    Parameter structure for PBE variables from input files.

Defines

- #define PBEPARM_MAXWRITE 20

    Number of things that can be written out in a single calculation.

Typedefs

- typedef enum ePBEparm_calcEnergy PBEparm_calcEnergy

    Define ePBEparm_calcEnergy enumeration as PBEparm_calcEnergy.

- typedef enum ePBEparm_calcForce PBEparm_calcForce

    Define ePBEparm_calcForce enumeration as PBEparm_calcForce.

- typedef struct sPBEparm PBEparm

  Declaration of the PBEparm class as the PBEparm structure.

Enumerations

- enum ePBEparm_calcEnergy { PCE_NO = 0, PCE_TOTAL = 1, PCE_COMPS = 2 }

  Define energy calculation enumeration.

- enum ePBEparm_calcForce { PCF_NO = 0, PCF_TOTAL = 1, PCF_COMPS = 2 }

  Define force calculation enumeration.

Functions

- VEXTERNC double PBEparm_getIonCharge (PBEparm ∗thee, int iion)

  Get charge (e) of specified ion species.

- VEXTERNC double PBEparm_getIonConc (PBEparm ∗thee, int iion)

  Get concentration (M) of specified ion species.

- VEXTERNC double PBEparm_getIonRadius (PBEparm ∗thee, int iion)

  Get radius (A) of specified ion species.

- VEXTERNC PBEparm ∗ PBEparm_ctor ()

  Construct PBEparm object.

- VEXTERNC int PBEparm_ctor2 (PBEparm ∗thee)

  FORTRAN stub to construct PBEparm object.

- VEXTERNC void PBEparm_dtor (PBEparm ∗∗thee)

  Object destructor.

- VEXTERNC void PBEparm_dtor2 (PBEparm ∗thee)

  FORTRAN stub for object destructor.

- VEXTERNC int PBEparm_check (PBEparm ∗thee)

  Consistency check for parameter values stored in object.

- VEXTERNC void PBEparm_copy (PBEparm ∗thee, PBEparm ∗parm)

  Copy PBEparm object into thee.

- VEXTERNC int PBEparm_parseToken (PBEparm ∗thee, char tok[VMAX_-BUFSIZE], Vio ∗sock)

  Parse a keyword from an input file.

10.13.1 Detailed Description

Contains declarations for class PBEparm.

Version

Id

pbeparm.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

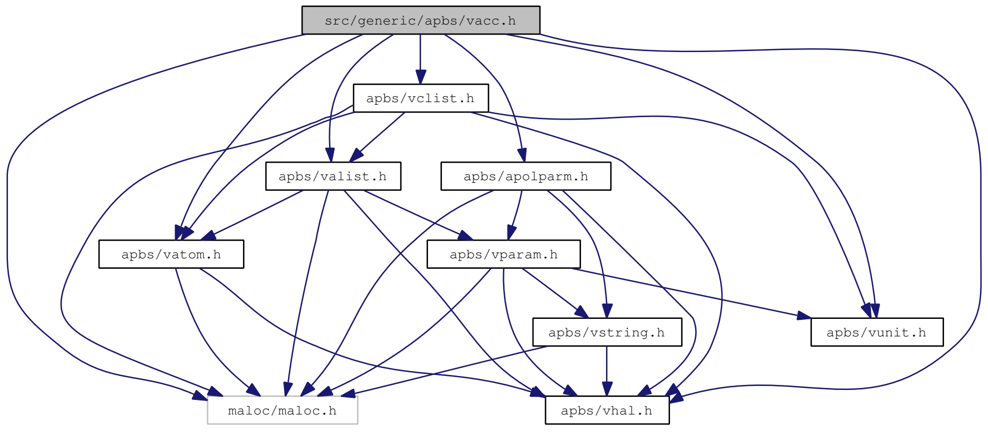
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
*
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* -  Redistributions of source code must retain the above copyright notice, this
* list of conditions and the following disclaimer.
*
* - Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
```

10.14 src/generic/apbs/vacc.h File Reference

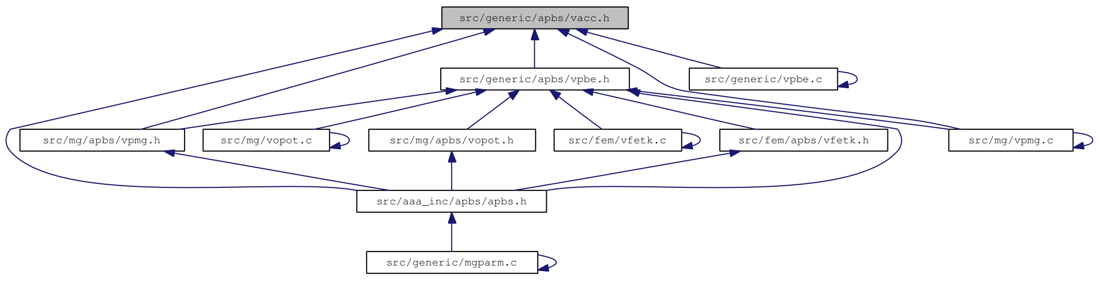
Contains declarations for class Vacc.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/valist.h"
#include "apbs/vclist.h"
#include "apbs/vatom.h"
#include "apbs/vunit.h"
#include "apbs/apolparm.h"
```

Include dependency graph for vacc.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVaccSurf

  Surface object list of per-atom surface points.

- struct sVacc

  Oracle for solvent- and ion-accessibility around a biomolecule.

Typedefs

- typedef struct sVaccSurf VaccSurf

  Declaration of the VaccSurf class as the VaccSurf structure.

- typedef struct sVacc Vacc

  Declaration of the Vacc class as the Vacc structure.

Functions

- VEXTERNC unsigned long int Vacc_memChk (Vacc ∗thee)

  Get number of bytes in this object and its members.

- VEXTERNC VaccSurf ∗ VaccSurf_ctor (Vmem ∗mem, double probe_radius, int nsphere)

  Allocate and construct the surface object; do not assign surface points to positions.

- VEXTERNC int VaccSurf_ctor2 (VaccSurf ∗thee, Vmem ∗mem, double probe_-radius, int nsphere)

  Construct the surface object using previously allocated memory; do not assign surface points to positions.

- VEXTERNC void VaccSurf_dtor (VaccSurf ∗∗thee)

  Destroy the surface object and free its memory.

- VEXTERNC void VaccSurf_dtor2 (VaccSurf ∗thee)

  Destroy the surface object.

- VEXTERNC VaccSurf ∗ VaccSurf_refSphere (Vmem ∗mem, int npts)

  Set up an array of points for a reference sphere of unit radius.

- VEXTERNC VaccSurf ∗ Vacc_atomSurf (Vacc ∗thee, Vatom ∗atom, VaccSurf ∗ref, double probe_radius)

Set up an array of points corresponding to the SAS due to a particular atom.

- VEXTERNC Vacc ∗ Vacc_ctor (Valist ∗alist, Vclist ∗clist, double surf_-density)

  Construct the accessibility object.

- VEXTERNC int Vacc_ctor2 (Vacc ∗thee, Valist ∗alist, Vclist ∗clist, double surf_density)

  FORTRAN stub to construct the accessibility object.

- VEXTERNC void Vacc_dtor (Vacc ∗∗thee)

  Destroy object.

- VEXTERNC void Vacc_dtor2 (Vacc ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC double Vacc_vdwAcc (Vacc ∗thee, double center[VAPBS_-DIM])

  Report van der Waals accessibility.

- VEXTERNC double Vacc_ivdwAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report inflated van der Waals accessibility.

- VEXTERNC double Vacc_molAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report molecular accessibility.

- VEXTERNC double Vacc_fastMolAcc (Vacc ∗thee, double center[VAPBS_-DIM], double radius)

  Report molecular accessibility quickly.

- VEXTERNC double Vacc_splineAcc (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad)

  Report spline-based accessibility.

- VEXTERNC void Vacc_splineAccGrad (Vacc ∗thee, double center[VAPBS_-DIM], double win, double infrad, double ∗grad)

  Report gradient of spline-based accessibility.

- VEXTERNC double Vacc_splineAccAtom (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom)

  Report spline-based accessibility for a given atom.

- VEXTERNC void Vacc_splineAccGradAtomUnnorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom (see Vpmg_splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by the accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm4 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 4th order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC void Vacc_splineAccGradAtomNorm3 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗force)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 3rd order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VEXTERNC double Vacc_SASA (Vacc ∗thee, double radius)

  Build the solvent accessible surface (SAS) and calculate the solvent accessible surface area.

- VEXTERNC double Vacc_totalSASA (Vacc ∗thee, double radius)

  Return the total solvent accessible surface area (SASA).

- VEXTERNC double Vacc_atomSASA (Vacc ∗thee, double radius, Vatom ∗atom)

  Return the atomic solvent accessible surface area (SASA).

- VEXTERNC VaccSurf ∗ Vacc_atomSASPoints (Vacc ∗thee, double radius, Vatom ∗atom)

  Get the set of points for this atom's solvent-accessible surface.

- VEXTERNC void Vacc_atomdSAV (Vacc ∗thee, double radius, Vatom ∗atom, double ∗dSA)

Get the derivatve of solvent accessible volume.

- VEXTERNC void Vacc_atomdSASA (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA)

  Get the derivatve of solvent accessible area.

- VEXTERNC void Vacc_totalAtomdSASA (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA)

  Testing purposes only.

- VEXTERNC void Vacc_totalAtomdSAV (Vacc ∗thee, double dpos, double radius, Vatom ∗atom, double ∗dSA, Vclist ∗clist)

  Total solvent accessible volume.

- VEXTERNC double Vacc_totalSAV (Vacc ∗thee, Vclist ∗clist, APOLparm ∗apolparm, double radius)

  Return the total solvent accessible volume (SAV).

- VPUBLIC int Vacc_wcaEnergy (Vacc ∗thee, APOLparm ∗apolparm, Valist ∗alist, Vclist ∗clist)

  Return the WCA integral energy.

- VPUBLIC int Vacc_wcaForceAtom (Vacc ∗thee, APOLparm ∗apolparm, Vclist ∗clist, Vatom ∗atom, double ∗force)

  Return the WCA integral force.

10.14.1 Detailed Description

Contains declarations for class Vacc.

Version

Id

vacc.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

∗

```
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
*
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* -  Redistributions of source code must retain the above copyright notice, this
* list of conditions and the following disclaimer.
*
* - Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
*
* - Neither the name of Washington University in St. Louis nor the names of its
* contributors may be used to endorse or promote products derived from this
* software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
* A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*
```

10.15 src/generic/apbs/valist.h File Reference

Contains declarations for class Valist.

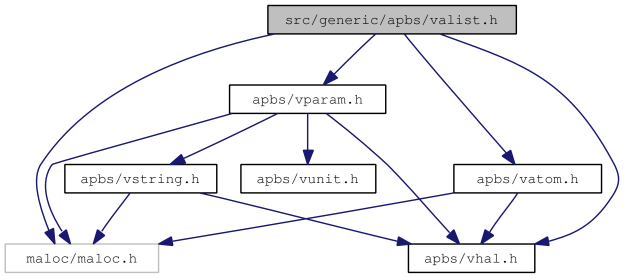
`#include "maloc/maloc.h"`

`#include "apbs/vhal.h"`

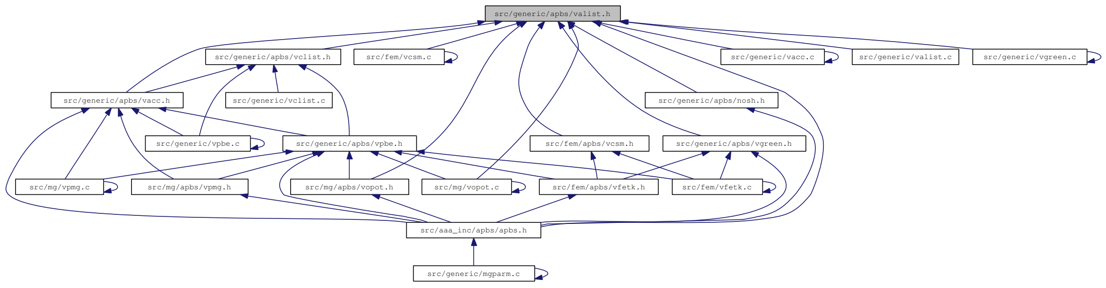
`#include "apbs/vatom.h"`

`#include "apbs/vparam.h"`

Include dependency graph for valist.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sValist

  Container class for list of atom objects.

Typedefs

- typedef struct sValist Valist

Declaration of the Valist class as the Valist structure.

Functions

- VEXTERNC Vatom ∗ Valist_getAtomList (Valist ∗thee)

  Get actual array of atom objects from the list.

- VEXTERNC double Valist_getCenterX (Valist ∗thee)

  Get x-coordinate of molecule center.

- VEXTERNC double Valist_getCenterY (Valist ∗thee)

  Get y-coordinate of molecule center.

- VEXTERNC double Valist_getCenterZ (Valist ∗thee)

  Get z-coordinate of molecule center.

- VEXTERNC int Valist_getNumberAtoms (Valist ∗thee)

  Get number of atoms in the list.

- VEXTERNC Vatom ∗ Valist_getAtom (Valist ∗thee, int i)

  Get pointer to particular atom in list.

- VEXTERNC unsigned long int Valist_memChk (Valist ∗thee)

  Get total memory allocated for this object and its members.

- VEXTERNC Valist ∗ Valist_ctor ()

  Construct the atom list object.

- VEXTERNC Vrc_Codes Valist_ctor2 (Valist ∗thee)

  FORTRAN stub to construct the atom list object.

- VEXTERNC void Valist_dtor (Valist ∗∗thee)

  Destroys atom list object.

- VEXTERNC void Valist_dtor2 (Valist ∗thee)

  FORTRAN stub to destroy atom list object.

- VEXTERNC Vrc_Codes Valist_readPQR (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from a PQR file.

- VEXTERNC Vrc_Codes Valist_readPDB (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from a PDB file.

- VEXTERNC Vrc_Codes Valist_readXML (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from an XML file.

- VEXTERNC Vrc_Codes Valist_getStatistics (Valist ∗thee)

  Load up Valist with various statistics.

10.15.1 Detailed Description

Contains declarations for class Valist.

Version

Id

valist.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
*
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* -  Redistributions of source code must retain the above copyright notice, this
```

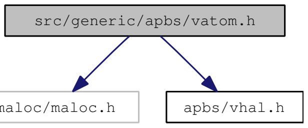
10.16 src/generic/apbs/vatom.h File Reference

Contains declarations for class Vatom.

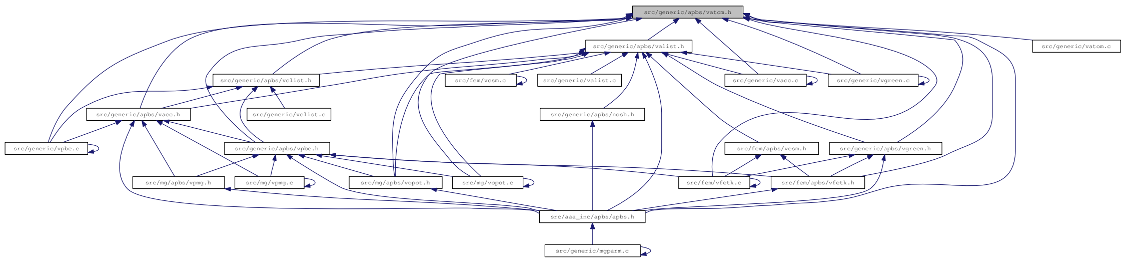
`#include "maloc/maloc.h"`

`#include "apbs/vhal.h"`

Include dependency graph for vatom.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVatom

  Contains public data members for Vatom class/module.

Defines

- #define VMAX_RECLEN 64

  Residue name length.

Typedefs

- typedef struct sVatom Vatom

  Declaration of the Vatom class as the Vatom structure.

Functions

- VEXTERNC double ∗ Vatom_getPosition (Vatom ∗thee)

  Get atomic position.

- VEXTERNC void Vatom_setRadius (Vatom ∗thee, double radius)

  Set atomic radius.

- VEXTERNC double Vatom_getRadius (Vatom ∗thee)

  Get atomic position.

- VEXTERNC void Vatom_setPartID (Vatom ∗thee, int partID)

  Set partition ID.

- VEXTERNC double Vatom_getPartID (Vatom ∗thee)

  Get partition ID.

- VEXTERNC void Vatom_setAtomID (Vatom ∗thee, int id)

  Set atom ID.

- VEXTERNC double Vatom_getAtomID (Vatom ∗thee)

  Get atom ID.

- VEXTERNC void Vatom_setCharge (Vatom ∗thee, double charge)

  Set atomic charge.

- VEXTERNC double Vatom_getCharge (Vatom ∗thee)

  Get atomic charge.

- VEXTERNC void Vatom_setEpsilon (Vatom ∗thee, double epsilon)

  Set atomic epsilon.

- VEXTERNC double Vatom_getEpsilon (Vatom ∗thee)

  Get atomic epsilon.

- VEXTERNC unsigned long int Vatom_memChk (Vatom ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC void Vatom_setResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Set residue name.

- VEXTERNC void Vatom_setAtomName (Vatom ∗thee, char atomName[VMAX_RECLEN])

  Set atom name.

- VEXTERNC void Vatom_getResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Retrieve residue name.

- VEXTERNC void Vatom_getAtomName (Vatom ∗thee, char atomName[VMAX_RECLEN])

  Retrieve atom name.

- VEXTERNC Vatom ∗ Vatom_ctor ()

  Constructor for the Vatom class.

- VEXTERNC int Vatom_ctor2 (Vatom ∗thee)

  FORTRAN stub constructor for the Vatom class.

- VEXTERNC void Vatom_dtor (Vatom ∗∗thee)

  Object destructor.

- VEXTERNC void Vatom_dtor2 (Vatom ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC void Vatom_setPosition (Vatom ∗thee, double position[3])

  Set the atomic position.

- VEXTERNC void Vatom_copyTo (Vatom ∗thee, Vatom ∗dest)

  Copy information to another atom.

- VEXTERNC void Vatom_copyFrom (Vatom ∗thee, Vatom ∗src)

  Copy information to another atom.

10.16.1 Detailed Description

Contains declarations for class Vatom.

Version

Id

vatom.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

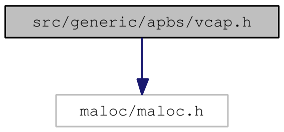
Attention

10.17 src/generic/apbs/vcap.h File Reference

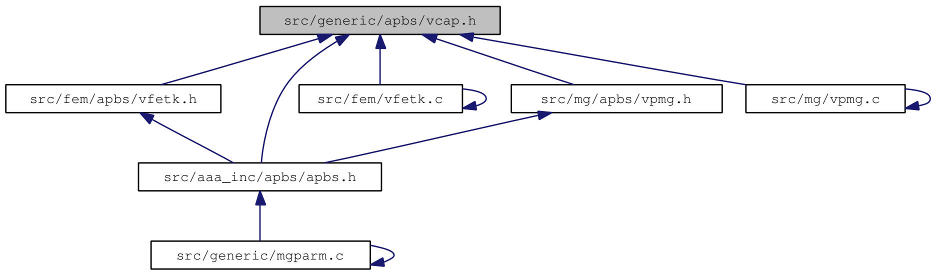
Contains declarations for class Vcap.

`#include "maloc/maloc.h"`

Include dependency graph for vcap.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define EXPMAX 85.00

  Maximum argument for exp(), sinh(), or cosh().

- #define EXPMIN -85.00

  Minimum argument for exp(), sinh(), or cosh().

Functions

- VEXTERNC double Vcap_exp (double x, int ∗ichop)

  Provide a capped exp() function.

- VEXTERNC double Vcap_sinh (double x, int ∗ichop)

  Provide a capped sinh() function.

- VEXTERNC double Vcap_cosh (double x, int ∗ichop)

  Provide a capped cosh() function.

10.17.1 Detailed Description

Contains declarations for class Vcap.

Version

Id

vcap.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

10.18 src/generic/apbs/vclist.h File Reference

Contains declarations for class Vclist.

`#include "maloc/maloc.h"`

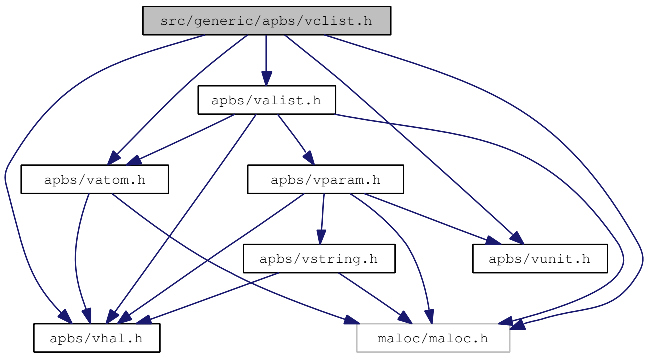
`#include "apbs/vhal.h"`

`#include "apbs/valist.h"`

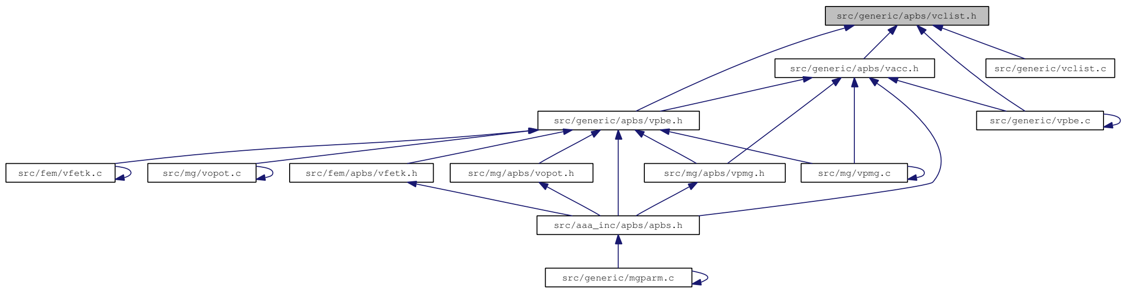
`#include "apbs/vatom.h"`

`#include "apbs/vunit.h"`

Include dependency graph for vclist.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVclistCell

  Atom cell list cell.

- struct sVclist

  Atom cell list.

Typedefs

- typedef enum eVclist_DomainMode Vclist_DomainMode

  Declaration of Vclist_DomainMode enumeration type.

- typedef struct sVclistCell VclistCell

  Declaration of the VclistCell class as the VclistCell structure.

- typedef struct sVclist Vclist

  Declaration of the Vclist class as the Vclist structure.

Enumerations

- enum eVclist_DomainMode { CLIST_AUTO_DOMAIN, CLIST_MANUAL_-DOMAIN }

  Atom cell list domain setup mode.

Functions

- VEXTERNC unsigned long int Vclist_memChk (Vclist ∗thee)

  Get number of bytes in this object and its members.

- VEXTERNC double Vclist_maxRadius (Vclist ∗thee)

  Get the max probe radius value (in A) the cell list was constructed with.

- VEXTERNC Vclist ∗ Vclist_ctor (Valist ∗alist, double max_radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  Construct the cell list object.

- VEXTERNC Vrc_Codes Vclist_ctor2 (Vclist ∗thee, Valist ∗alist, double max_-radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  FORTRAN stub to construct the cell list object.

- VEXTERNC void Vclist_dtor (Vclist ∗∗thee)

Destroy object.

- VEXTERNC void Vclist_dtor2 (Vclist ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC VclistCell ∗ Vclist_getCell (Vclist ∗thee, double position[VAPBS_DIM])

  Return cell corresponding to specified position or return VNULL.

- VEXTERNC VclistCell ∗ VclistCell_ctor (int natoms)

  Allocate and construct a cell list cell object.

- VEXTERNC Vrc_Codes VclistCell_ctor2 (VclistCell ∗thee, int natoms)

  Construct a cell list object.

- VEXTERNC void VclistCell_dtor (VclistCell ∗∗thee)

  Destroy object.

- VEXTERNC void VclistCell_dtor2 (VclistCell ∗thee)

  FORTRAN stub to destroy object.

10.18.1 Detailed Description

Contains declarations for class Vclist.

Version

Id

vclist.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
```

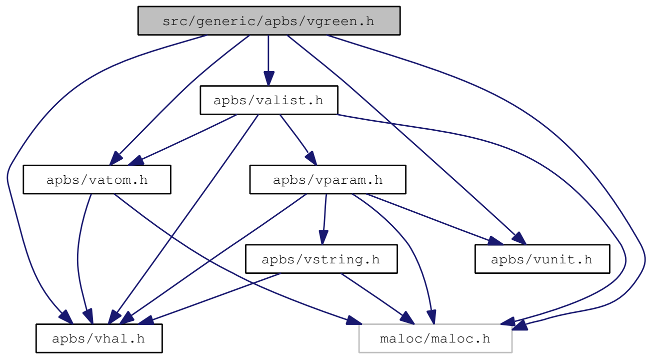
```
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * -  Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * - Neither the name of Washington University in St. Louis nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

10.19 src/generic/apbs/vgreen.h File Reference

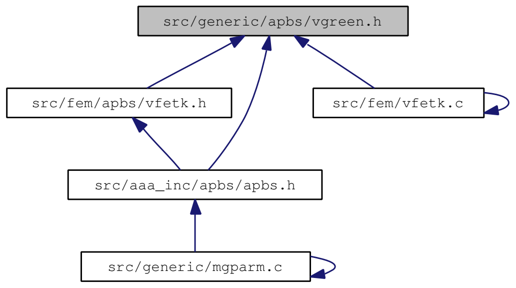
Contains declarations for class Vgreen.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vunit.h"
#include "apbs/vatom.h"
#include "apbs/valist.h"
```

Include dependency graph for vgreen.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVgreen

  Contains public data members for Vgreen class/module.

Typedefs

- typedef struct sVgreen Vgreen

  Declaration of the Vgreen class as the Vgreen structure.

Functions

- VEXTERNC Valist ∗ Vgreen_getValist (Vgreen ∗thee)

  Get the atom list associated with this Green's function object.

- VEXTERNC unsigned long int Vgreen_memChk (Vgreen ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vgreen ∗ Vgreen_ctor (Valist ∗alist)

  Construct the Green's function oracle.

- VEXTERNC int Vgreen_ctor2 (Vgreen ∗thee, Valist ∗alist)

  FORTRAN stub to construct the Green's function oracle.

- VEXTERNC void Vgreen_dtor (Vgreen ∗∗thee)

  Destruct the Green's function oracle.

- VEXTERNC void Vgreen_dtor2 (Vgreen ∗thee)

  FORTRAN stub to destruct the Green's function oracle.

- VEXTERNC int Vgreen_helmholtz (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val, double kappa)

  Get the Green's function for Helmholtz's equation integrated over the atomic point charges.

- VEXTERNC int Vgreen_helmholtzD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗gradx, double ∗grady, double ∗gradz, double kappa)

  Get the gradient of Green's function for Helmholtz's equation integrated over the atomic point charges.

- VEXTERNC int Vgreen_coulomb_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

  Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VEXTERNC int Vgreen_coulomb (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation or H. E. Johnston, R. Krasny FMM library (if available).

- VEXTERNC int Vgreen_coulombD_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VEXTERNC int Vgreen_coulombD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using either direct summation or H. E. Johnston/R. Krasny FMM library (if available).

10.19.1 Detailed Description

Contains declarations for class Vgreen.

Version

Id

vgreen.h 1552 2010-02-10 17:46:27Z yhuang01

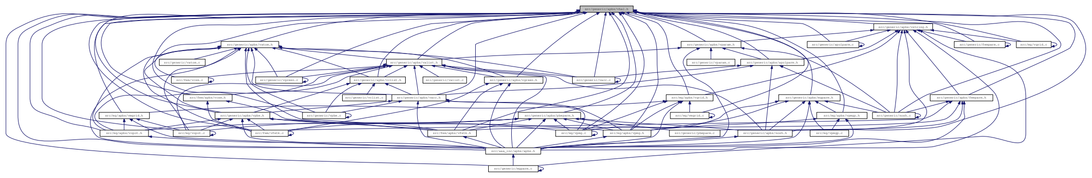
Author

Nathan A. Baker

10.20 src/generic/apbs/vhal.h File Reference

Contains generic macro definitions for APBS.

This graph shows which files directly or indirectly include this file:



Defines

- #define APBS_TIMER_WALL_CLOCK 26

  APBS total execution timer ID.

- #define APBS_TIMER_SETUP 27

  APBS setup timer ID.

- #define APBS_TIMER_SOLVER 28

  APBS solver timer ID.

- #define APBS_TIMER_ENERGY 29

  APBS energy timer ID.

- #define APBS_TIMER_FORCE 30

  APBS force timer ID.

- #define APBS_TIMER_TEMP1 31

  APBS temp timer #1 ID.

- #define APBS_TIMER_TEMP2 32

  APBS temp timer #2 ID.

- #define MAXMOL 5

  The maximum number of molecules that can be involved in a single PBE calculation.

- #define MAXION 10

  The maximum number of ion species that can be involved in a single PBE calculation.

- #define MAXFOCUS 5

The maximum number of times an MG calculation can be focused.

- #define VMGNLEV 4

  Minimum number of levels in a multigrid calculations.

- #define VREDFRAC 0.25

  Maximum reduction of grid spacing during a focusing calculation.

- #define VAPBS_NVS 4

  Number of vertices per simplex (hard-coded to 3D).

- #define VAPBS_DIM 3

  Our dimension.

- #define VAPBS_RIGHT 0

  Face definition for a volume.

- #define VAPBS_FRONT 1

  Face definition for a volume.

- #define VAPBS_UP 2

  Face definition for a volume.

- #define VAPBS_LEFT 3

  Face definition for a volume.

- #define VAPBS_BACK 4

  Face definition for a volume.

- #define VAPBS_DOWN 5

  Face definition for a volume.

- #define VPMGSMALL 1e-12

  A small number used in Vpmg to decide if points are on/off grid-lines or non-zer0 (etc.).

- #define SINH_MIN -85.0

  Used to set the min values acceptable for sinh chopping.

- #define SINH_MAX 85.0

  Used to set the max values acceptable for sinh chopping.

- #define VF77_MANGLE(name, NAME) name

Name-mangling macro for using FORTRAN functions in C code.

- #define VFLOOR(value) floor(value)

  Wrapped floor to fix floating point issues in the Intel compiler.

- #define VEMBED(rctag)

  Allows embedding of RCS ID tags in object files.

Typedefs

- typedef enum eVrc_Codes **Vrc_Codes**
- typedef enum eVsol_Meth **Vsol_Meth**
- typedef enum eVsurf_Meth Vsurf_Meth

  Declaration of the Vsurf_Meth type as the Vsurf_Meth enum.

- typedef enum eVhal_PBEType Vhal_PBEType

  Declaration of the Vhal_PBEType type as the Vhal_PBEType enum.

- typedef enum eVhal_IPKEYType Vhal_IPKEYType

  Declaration of the Vhal_IPKEYType type as the Vhal_IPKEYType enum.

- typedef enum eVhal_NONLINType Vhal_NONLINType

  Declaration of the Vhal_NONLINType type as the Vhal_NONLINType enum.

- typedef enum eVoutput_Format Voutput_Format

  Declaration of the Voutput_Format type as the VOutput_Format enum.

- typedef enum eVbcfl Vbcfl

  Declare Vbcfl type.

- typedef enum eVchrg_Meth Vchrg_Meth

  Declaration of the Vchrg_Meth type as the Vchrg_Meth enum.

- typedef enum eVchrg_Src Vchrg_Src

  Declaration of the Vchrg_Src type as the Vchrg_Meth enum.

- typedef enum eVdata_Type Vdata_Type

  Declaration of the Vdata_Type type as the Vdata_Type enum.

- typedef enum eVdata_Format Vdata_Format

  Declaration of the Vdata_Format type as the Vdata_Format enum.

Enumerations

- enum eVrc_Codes { **VRC_WARNING** = -1, VRC_FAILURE = 0, VRC_-SUCCESS = 1 }

  Return code enumerations.

- enum eVsol_Meth {

  VSOL_CGMG, **VSOL_Newton**, **VSOL_MG**, **VSOL_CG**,

  VSOL_SOR, **VSOL_RBGS**, **VSOL_WJ**, **VSOL_Richardson**,

  VSOL_CGMGAqua, **VSOL_NewtonAqua** }

  Solution Method enumerations.

- enum eVsurf_Meth {

  VSM_MOL = 0, VSM_MOLSMOOTH = 1, VSM_SPLINE = 2, VSM_-SPLINE3 = 3,

  VSM_SPLINE4 = 4 }

  Types of molecular surface definitions.

- enum eVhal_PBEType {

  PBE_LPBE, PBE_NPBE, PBE_LRPBE, **PBE_NRPBE**,

  PBE_SMPBE }

  Version of PBE to solve.

- enum eVhal_IPKEYType { IPKEY_SMPBE = -2, IPKEY_LPBE, IPKEY_-NPBE }

  Type of ipkey to use for MG methods.

- enum eVhal_NONLINType {

  NONLIN_LPBE = 0, **NONLIN_NPBE**, **NONLIN_SMPBE**, **NONLIN_-LPBEAQUA**,

  NONLIN_NPBEAQUA }

  Type of nonlinear to use for MG methods.

- enum eVoutput_Format { OUTPUT_NULL, OUTPUT_FLAT }

  Output file format.

- enum eVbcfl {

  BCFL_ZERO = 0, BCFL_SDH = 1, BCFL_MDH = 2, BCFL_UNUSED = 3,

  BCFL_FOCUS = 4, BCFL_MEM = 5 }

  Types of boundary conditions.

- enum eVchrg_Meth { VCM_TRIL = 0, VCM_BSPL2 = 1, VCM_BSPL4 = 2 }

  Types of charge discretization methods.

- enum eVchrg_Src { VCM_CHARGE = 0, VCM_PERMANENT = 1, VCM_-INDUCED = 2, VCM_NLINDUCED = 3 }

  Charge source.

- enum eVdata_Type {
  VDT_CHARGE, VDT_POT, VDT_SMOL, VDT_SSPL,
  VDT_VDW, VDT_IVDW, VDT_LAP, VDT_EDENS,
  VDT_NDENS, VDT_QDENS, VDT_DIELX, VDT_DIELY,
  VDT_DIELZ, VDT_KAPPA }

  Types of (scalar) data that can be written out of APBS.

- enum eVdata_Format { VDF_DX = 0, VDF_UHBD = 1, VDF_AVS = 2, VDF_-MCSF = 3 }

  Format of data for APBS I/O.

10.20.1 Detailed Description

Contains generic macro definitions for APBS.

Version

Id

vhal.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

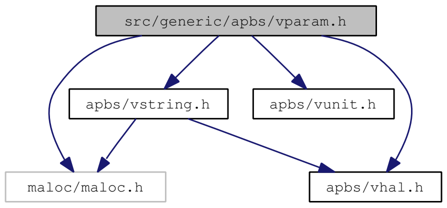
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
```

10.21 src/generic/apbs/vparam.h File Reference

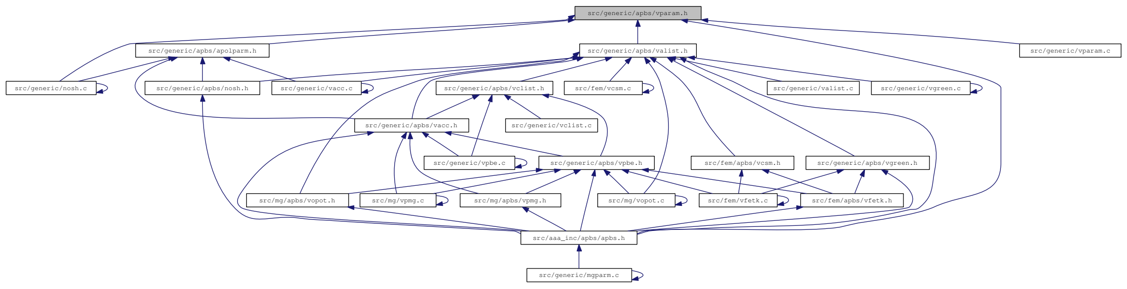
Contains declarations for class Vparam.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vunit.h"
#include "apbs/vstring.h"
```

Include dependency graph for vparam.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVparam_AtomData

  AtomData sub-class; stores atom data.

- struct Vparam_ResData

  ResData sub-class; stores residue data.

- struct Vparam

  Reads and assigns charge/radii parameters.

Typedefs

- typedef struct sVparam_AtomData Vparam_AtomData

    Declaration of the Vparam_AtomData class as the sVparam_AtomData structure.

- typedef struct Vparam_ResData Vparam_ResData

    Declaration of the Vparam_ResData class as the Vparam_ResData structure.

- typedef struct Vparam Vparam

    Declaration of the Vparam class as the Vparam structure.

Functions

- VEXTERNC unsigned long int Vparam_memChk (Vparam ∗thee)

    Get number of bytes in this object and its members.

- VEXTERNC Vparam_AtomData ∗ Vparam_AtomData_ctor ()

    Construct the object.

- VEXTERNC int Vparam_AtomData_ctor2 (Vparam_AtomData ∗thee)

    FORTRAN stub to construct the object.

- VEXTERNC void Vparam_AtomData_dtor (Vparam_AtomData ∗∗thee)

    Destroy object.

- VEXTERNC void Vparam_AtomData_dtor2 (Vparam_AtomData ∗thee)

    FORTRAN stub to destroy object.

- VEXTERNC void Vparam_AtomData_copyTo (Vparam_AtomData ∗thee, Vparam_AtomData ∗dest)

    Copy current atom object to destination.

- VEXTERNC void Vparam_ResData_copyTo (Vparam_ResData ∗thee, Vparam_ResData ∗dest)

    Copy current residue object to destination.

- VEXTERNC void Vparam_AtomData_copyFrom (Vparam_AtomData ∗thee, Vparam_AtomData ∗src)

    Copy current atom object from another.

- VEXTERNC Vparam_ResData ∗ Vparam_ResData_ctor (Vmem ∗mem)

Construct the object.

- VEXTERNC int Vparam_ResData_ctor2 (Vparam_ResData ∗thee, Vmem ∗mem)

  FORTRAN stub to construct the object.

- VEXTERNC void Vparam_ResData_dtor (Vparam_ResData ∗∗thee)

  Destroy object.

- VEXTERNC void Vparam_ResData_dtor2 (Vparam_ResData ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC Vparam ∗ Vparam_ctor ()

  Construct the object.

- VEXTERNC int Vparam_ctor2 (Vparam ∗thee)

  FORTRAN stub to construct the object.

- VEXTERNC void Vparam_dtor (Vparam ∗∗thee)

  Destroy object.

- VEXTERNC void Vparam_dtor2 (Vparam ∗thee)

  FORTRAN stub to destroy object.

- VEXTERNC Vparam_ResData ∗ Vparam_getResData (Vparam ∗thee, char resName[VMAX_ARGLEN])

  Get residue data.

- VEXTERNC Vparam_AtomData ∗ Vparam_getAtomData (Vparam ∗thee, char resName[VMAX_ARGLEN], char atomName[VMAX_ARGLEN])

  Get atom data.

- VEXTERNC int Vparam_readFlatFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read a flat-file format parameter database.

- VEXTERNC int Vparam_readXMLFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read an XML format parameter database.

10.21.1 Detailed Description

Contains declarations for class Vparam.

Version

Id

vparam.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Attention

```
* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*
```

10.22 src/generic/apbs/vpbe.h File Reference

Contains declarations for class Vpbe.

`#include "maloc/maloc.h"`

`#include "apbs/vhal.h"`

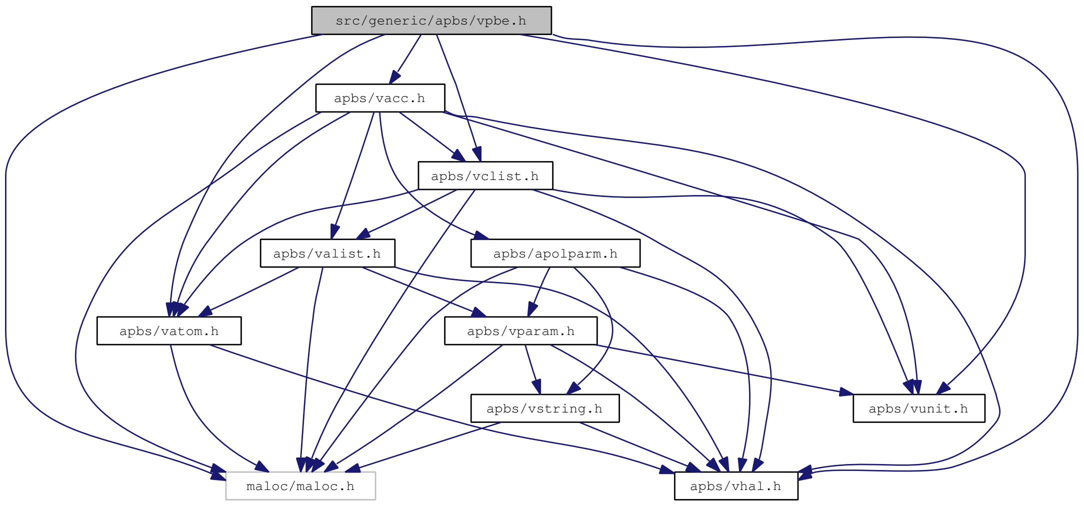
`#include "apbs/vunit.h"`

`#include "apbs/vatom.h"`

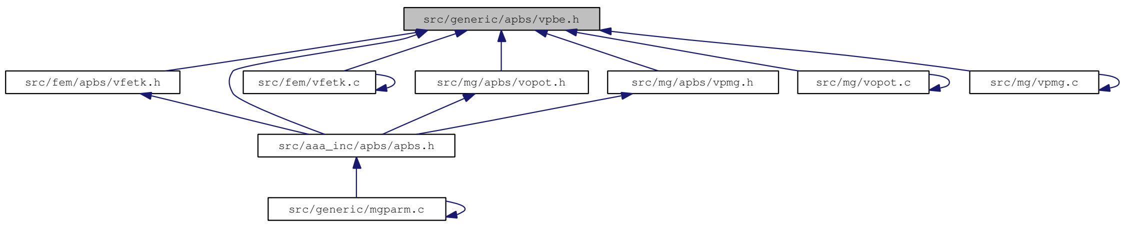
`#include "apbs/vacc.h"`

`#include "apbs/vclist.h"`

Include dependency graph for vpbe.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVpbe

  Contains public data members for Vpbe class/module.

Typedefs

- typedef struct sVpbe Vpbe

  Declaration of the Vpbe class as the Vpbe structure.

Functions

- VEXTERNC Valist ∗ Vpbe_getValist (Vpbe ∗thee)

  Get atom list.
- VEXTERNC Vacc ∗ Vpbe_getVacc (Vpbe ∗thee)

  Get accessibility oracle.
- VEXTERNC double Vpbe_getBulkIonicStrength (Vpbe ∗thee)

  Get bulk ionic strength.
- VEXTERNC double Vpbe_getMaxIonRadius (Vpbe ∗thee)

  Get maximum radius of ion species.
- VEXTERNC double Vpbe_getTemperature (Vpbe ∗thee)

  Get temperature.
- VEXTERNC double Vpbe_getSoluteDiel (Vpbe ∗thee)

  Get solute dielectric constant.
- VEXTERNC double Vpbe_getGamma (Vpbe ∗thee)

  Get apolar coefficient.
- VEXTERNC double Vpbe_getSoluteRadius (Vpbe ∗thee)

  Get sphere radius which bounds biomolecule.
- VEXTERNC double Vpbe_getSoluteXlen (Vpbe ∗thee)

  Get length of solute in x dimension.
- VEXTERNC double Vpbe_getSoluteYlen (Vpbe ∗thee)

Get length of solute in y dimension.

- VEXTERNC double Vpbe_getSoluteZlen (Vpbe ∗thee)

  Get length of solute in z dimension.

- VEXTERNC double ∗ Vpbe_getSoluteCenter (Vpbe ∗thee)

  Get coordinates of solute center.

- VEXTERNC double Vpbe_getSoluteCharge (Vpbe ∗thee)

  Get total solute charge.

- VEXTERNC double Vpbe_getSolventDiel (Vpbe ∗thee)

  Get solvent dielectric constant.

- VEXTERNC double Vpbe_getSolventRadius (Vpbe ∗thee)

  Get solvent molecule radius.

- VEXTERNC double Vpbe_getXkappa (Vpbe ∗thee)

  Get Debye-Huckel parameter.

- VEXTERNC double Vpbe_getDeblen (Vpbe ∗thee)

  Get Debye-Huckel screening length.

- VEXTERNC double Vpbe_getZkappa2 (Vpbe ∗thee)

  Get modified squared Debye-Huckel parameter.

- VEXTERNC double Vpbe_getZmagic (Vpbe ∗thee)

  Get charge scaling factor.

- VEXTERNC double Vpbe_getzmem (Vpbe ∗thee)

  Get z position of the membrane bottom.

- VEXTERNC double Vpbe_getLmem (Vpbe ∗thee)

  Get length of the membrane (A)
  aauthor Michael Grabe.

- VEXTERNC double Vpbe_getmembraneDiel (Vpbe ∗thee)

  Get membrane dielectric constant.

- VEXTERNC double Vpbe_getmemv (Vpbe ∗thee)

  Get membrane potential (kT).

- VEXTERNC Vpbe ∗ Vpbe_ctor (Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

    Construct Vpbe object.

- VEXTERNC int Vpbe_ctor2 (Vpbe ∗thee, Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

    FORTRAN stub to construct Vpbe objct.

- VEXTERNC int Vpbe_getIons (Vpbe ∗thee, int ∗nion, double ionConc[MAXION], double ionRadii[MAXION], double ionQ[MAXION])

    Get information about the counterion species present.

- VEXTERNC void Vpbe_dtor (Vpbe ∗∗thee)

    Object destructor.

- VEXTERNC void Vpbe_dtor2 (Vpbe ∗thee)

    FORTRAN stub object destructor.

- VEXTERNC double Vpbe_getCoulombEnergy1 (Vpbe ∗thee)

    Calculate coulombic energy of set of charges.

- VEXTERNC unsigned long int Vpbe_memChk (Vpbe ∗thee)

    Return the memory used by this structure (and its contents) in bytes.

10.22.1 Detailed Description

Contains declarations for class Vpbe.

Version

Id

vpbe.h 1563 2010-03-07 14:03:26Z sobolevnrm

Author

Nathan A. Baker

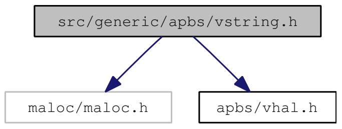
Attention

10.23 src/generic/apbs/vstring.h File Reference

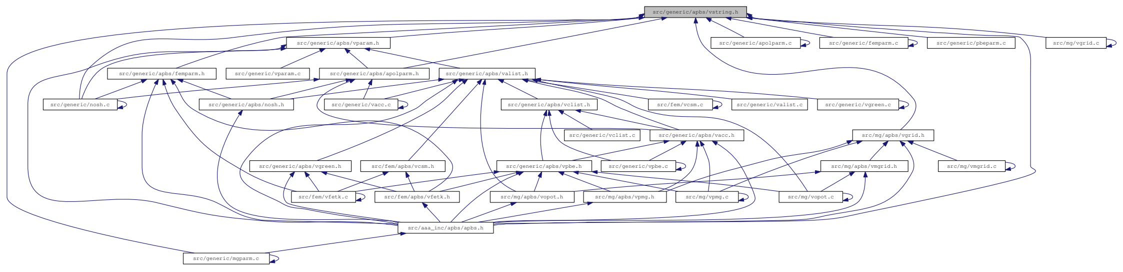
Contains declarations for class Vstring.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
```

Include dependency graph for vstring.h:



This graph shows which files directly or indirectly include this file:



Functions

- VEXTERNC int Vstring_strcasecmp (const char ∗s1, const char ∗s2)

  Case-insensitive string comparison (BSD standard).

- VEXTERNC int Vstring_isdigit (const char ∗tok)

  A modified sscanf that examines the complete string.

10.23.1 Detailed Description

Contains declarations for class Vstring.

Version

Id

vstring.h 1552 2010-02-10 17:46:27Z yhuang01

Author

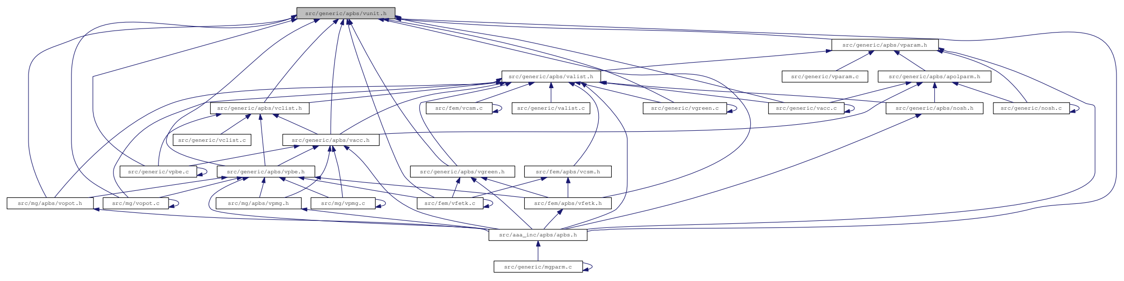
Nathan A. Baker

Attention

10.24 src/generic/apbs/vunit.h File Reference

Contains a collection of useful constants and conversion factors.

This graph shows which files directly or indirectly include this file:



Defines

- #define Vunit_J_to_cal 4.1840000e+00

  Multiply by this to convert J to cal.

- #define Vunit_cal_to_J 2.3900574e-01

  Multiply by this to convert cal to J.

- #define Vunit_amu_to_kg 1.6605402e-27

  Multiply by this to convert amu to kg.

- #define Vunit_kg_to_amu 6.0221367e+26

  Multiply by this to convert kg to amu.

- #define Vunit_ec_to_C 1.6021773e-19

  Multiply by this to convert ec to C.

- #define Vunit_C_to_ec 6.2415065e+18

  Multiply by this to convert C to ec.

- #define Vunit_ec 1.6021773e-19

  Charge of an electron in C.

- #define Vunit_kb 1.3806581e-23

  Boltzmann constant.

- #define Vunit_Na 6.0221367e+23

  Avogadro's number.

- #define Vunit_pi VPI

  Pi.

- #define Vunit_eps0 8.8541878e-12

  Vacuum permittivity.

- #define Vunit_esu_ec2A 3.3206364e+02

  ${e_c}^2/$ *in ESU units => kcal/mol*

- #define Vunit_esu_kb 1.9871913e-03

  k_b *in ESU units => kcal/mol*

10.24.1 Detailed Description

Contains a collection of useful constants and conversion factors.

Author

Nathan Baker
Nathan A. Baker

Version

Id

vunit.h 1552 2010-02-10 17:46:27Z yhuang01

Attention

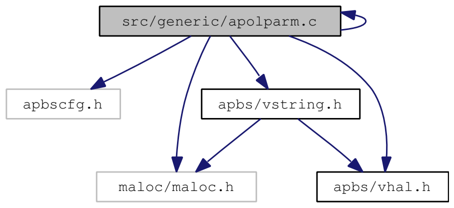
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
```

10.25 src/generic/apolparm.c File Reference

Class APOLparm methods.

```
#include "apbscfg.h"
#include "apbs/apolparm.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vstring.h"
```

Include dependency graph for apolparm.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC APOLparm * APOLparm_ctor ()

  Construct APOLparm.

- VPUBLIC Vrc_Codes APOLparm_ctor2 (APOLparm *thee)

  FORTRAN stub to construct APOLparm.

- VPUBLIC void APOLparm_copy (APOLparm *thee, APOLparm *source)

  Copy target object into thee.

- VPUBLIC void APOLparm_dtor (APOLparm **thee)

  Object destructor.

- VPUBLIC void APOLparm_dtor2 (APOLparm *thee)

  FORTRAN stub for object destructor.

- VPUBLIC Vrc_Codes APOLparm_check (APOLparm ∗thee)

  Consistency check for parameter values stored in object.

- VPRIVATE Vrc_Codes **APOLparm_parseGRID** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseMOL** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseSRFM** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseSRAD** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseSWIN** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseTEMP** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseGAMMA** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseCALCENERGY** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseCALCFORCE** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseBCONC** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseSDENS** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parseDPOS** (APOLparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **APOLparm_parsePRESS** (APOLparm ∗thee, Vio ∗sock)
- VPUBLIC Vrc_Codes APOLparm_parseToken (APOLparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

10.25.1 Detailed Description

Class APOLparm methods.

Author

David Gohara

Version

Id

apolparm.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

10.26 src/generic/femparm.c File Reference

Class FEMparm methods.

```
#include "apbscfg.h"
```

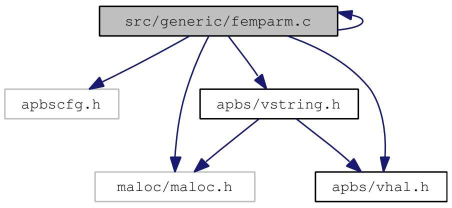
```
#include "apbs/femparm.h"
```

```
#include "maloc/maloc.h"
```

```
#include "apbs/vhal.h"
```

```
#include "apbs/vstring.h"
```

Include dependency graph for femparm.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC FEMparm ∗ FEMparm_ctor (FEMparm_CalcType type)

  Construct FEMparm.

- VPUBLIC int FEMparm_ctor2 (FEMparm ∗thee, FEMparm_CalcType type)

  FORTRAN stub to construct FEMparm.

- VPUBLIC void FEMparm_copy (FEMparm ∗thee, FEMparm ∗source)

  Copy target object into thee.

- VPUBLIC void FEMparm_dtor (FEMparm ∗∗thee)

  Object destructor.

- VPUBLIC void FEMparm_dtor2 (FEMparm ∗thee)

  FORTRAN stub for object destructor.

- VPUBLIC int FEMparm_check (FEMparm ∗thee)

    Consistency check for parameter values stored in object.

- VPRIVATE Vrc_Codes **FEMparm_parseDOMAINLENGTH** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseETOL** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseEKEY** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseAKEYPRE** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseAKEYSOLVE** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseTARGETNUM** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseTARGETRES** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseMAXSOLVE** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseMAXVERT** (FEMparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **FEMparm_parseUSEMESH** (FEMparm ∗thee, Vio ∗sock)
- VPUBLIC Vrc_Codes FEMparm_parseToken (FEMparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

    Parse an MG keyword from an input file.

10.26.1 Detailed Description

Class FEMparm methods.

Author

Nathan Baker

Version

Id

femparm.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
```

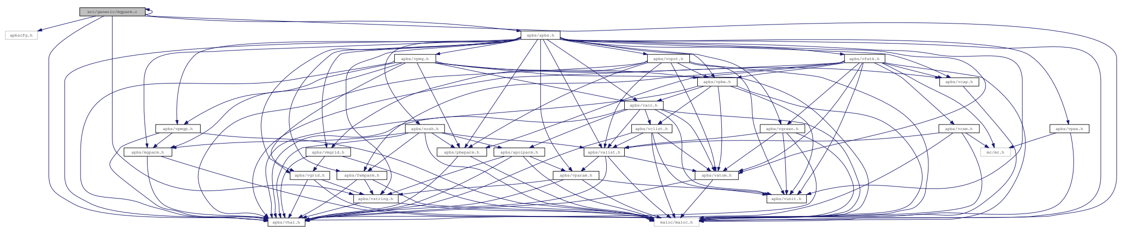
```
 *
 * Nathan A. Baker (baker@biochem.wustl.edu)
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * -  Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * - Neither the name of Washington University in St. Louis nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

10.27 src/generic/mgparm.c File Reference

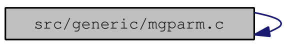
Class MGparm methods.

```
#include "apbscfg.h"
#include "apbs/apbs.h"
#include "apbs/vhal.h"
#include "apbs/mgparm.h"
#include "maloc/maloc.h"
#include "apbs/vstring.h"
```

Include dependency graph for mgparm.c:



This graph shows which files directly or indirectly include this file:



Functions

- VPUBLIC void MGparm_setCenterX (MGparm ∗thee, double x)

  Set center x-coordinate.

- VPUBLIC void MGparm_setCenterY (MGparm ∗thee, double y)

  Set center y-coordinate.

- VPUBLIC void MGparm_setCenterZ (MGparm ∗thee, double z)

  Set center z-coordinate.

- VPUBLIC double MGparm_getCenterX (MGparm ∗thee)

  Get center x-coordinate.

- VPUBLIC double MGparm_getCenterY (MGparm ∗thee)

  Get center y-coordinate.

- VPUBLIC double MGparm_getCenterZ (MGparm ∗thee)

  Get center z-coordinate.

- VPUBLIC int MGparm_getNx (MGparm ∗thee)

  Get number of grid points in x direction.

- VPUBLIC int MGparm_getNy (MGparm ∗thee)

  Get number of grid points in y direction.

- VPUBLIC int MGparm_getNz (MGparm ∗thee)

  Get number of grid points in z direction.

- VPUBLIC double MGparm_getHx (MGparm ∗thee)

  Get grid spacing in x direction (Å).

- VPUBLIC double MGparm_getHy (MGparm ∗thee)

  Get grid spacing in y direction (Å).

- VPUBLIC double MGparm_getHz (MGparm ∗thee)

  Get grid spacing in z direction (Å).

- VPUBLIC MGparm ∗ MGparm_ctor (MGparm_CalcType type)

  Construct MGparm object.

- VPUBLIC Vrc_Codes MGparm_ctor2 (MGparm ∗thee, MGparm_CalcType type)

  FORTRAN stub to construct MGparm object.

- VPUBLIC void MGparm_dtor (MGparm ∗∗thee)

  Object destructor.

- VPUBLIC void MGparm_dtor2 (MGparm ∗thee)

  FORTRAN stub for object destructor.

- VPUBLIC Vrc_Codes MGparm_check (MGparm ∗thee)

  Consistency check for parameter values stored in object.

- VPUBLIC void MGparm_copy (MGparm ∗thee, MGparm ∗parm)

  Copy MGparm object into thee.

- VPRIVATE Vrc_Codes **MGparm_parseDIME** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseCHGM** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseNLEV** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseETOL** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseGRID** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseGLEN** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseGAMMA** (MGparm ∗thee, Vio ∗sock)

- VPRIVATE Vrc_Codes **MGparm_parseGCENT** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseCGLEN** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseFGLEN** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseCGCENT** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseFGCENT** (MGparm ∗thee, Vio ∗sock)

- VPRIVATE Vrc_Codes **MGparm_parsePDIME** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseOFRAC** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseASYNC** (MGparm ∗thee, Vio ∗sock)
- VPRIVATE Vrc_Codes **MGparm_parseUSEAQUA** (MGparm ∗thee, Vio ∗sock)
- VPUBLIC Vrc_Codes MGparm_parseToken (MGparm ∗thee, char tok[VMAX_BUFSIZE], Vio ∗sock)

  Parse an MG keyword from an input file.

10.27.1 Detailed Description

Class MGparm methods.

Author

Nathan Baker

Version

Id

mgparm.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

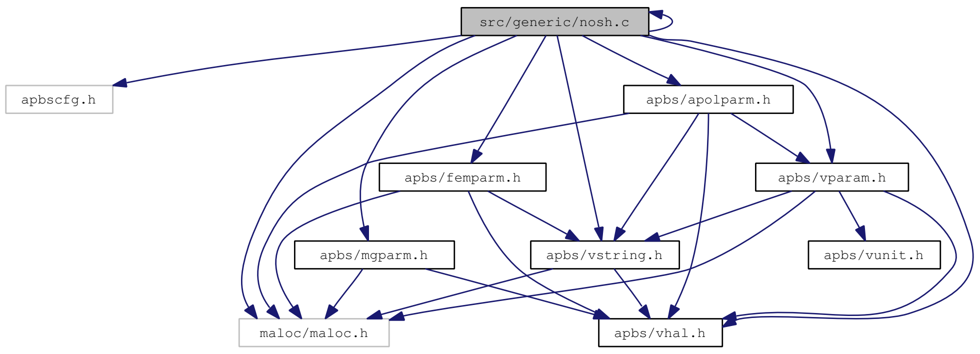
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
```

10.28 src/generic/nosh.c File Reference

Class NOsh methods.

```
#include "apbscfg.h"
#include "apbs/nosh.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/mgparm.h"
#include "apbs/femparm.h"
#include "apbs/apolparm.h"
#include "apbs/vparam.h"
#include "apbs/vstring.h"
```

Include dependency graph for nosh.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPRIVATE int **NOsh_parseREAD** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parsePRINT** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseELEC** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseAPOLAR** (NOsh ∗thee, Vio ∗sock)
- VEXTERNC int **NOsh_parseFEM** (NOsh ∗thee, Vio ∗sock, NOsh_calc ∗elec)

- VEXTERNC int **NOsh_parseMG** (NOsh ∗thee, Vio ∗sock, NOsh_calc ∗elec)
- VEXTERNC int **NOsh_parseAPOL** (NOsh ∗thee, Vio ∗sock, NOsh_calc ∗elec)

- VPRIVATE int **NOsh_setupCalcMG** (NOsh ∗thee, NOsh_calc ∗elec)
- VPRIVATE int **NOsh_setupCalcMGAUTO** (NOsh ∗thee, NOsh_calc ∗elec)
- VPRIVATE int **NOsh_setupCalcMGMANUAL** (NOsh ∗thee, NOsh_calc ∗elec)
- VPRIVATE int **NOsh_setupCalcMGPARA** (NOsh ∗thee, NOsh_calc ∗elec)
- VPRIVATE int **NOsh_setupCalcFEM** (NOsh ∗thee, NOsh_calc ∗elec)
- VPRIVATE int **NOsh_setupCalcFEMANUAL** (NOsh ∗thee, NOsh_calc ∗elec)

- VPRIVATE int **NOsh_setupCalcAPOL** (NOsh ∗thee, NOsh_calc ∗elec)
- VPUBLIC char ∗ NOsh_getMolpath (NOsh ∗thee, int imol)

  Returns path to specified molecule.

- VPUBLIC char ∗ NOsh_getDielXpath (NOsh ∗thee, int imol)

  Returns path to specified x-shifted dielectric map.

- VPUBLIC char ∗ NOsh_getDielYpath (NOsh ∗thee, int imol)

  Returns path to specified y-shifted dielectric map.

- VPUBLIC char ∗ NOsh_getDielZpath (NOsh ∗thee, int imol)

  Returns path to specified z-shifted dielectric map.

- VPUBLIC char ∗ NOsh_getKappapath (NOsh ∗thee, int imol)

  Returns path to specified kappa map.

- VPUBLIC char ∗ NOsh_getChargepath (NOsh ∗thee, int imol)

  Returns path to specified charge distribution map.

- VPUBLIC NOsh_calc ∗ NOsh_getCalc (NOsh ∗thee, int icalc)

  Returns specified calculation object.

- VPUBLIC int NOsh_getDielfmt (NOsh ∗thee, int i)

  Returns format of specified dielectric map.

- VPUBLIC int NOsh_getKappafmt (NOsh ∗thee, int i)

  Returns format of specified kappa map.

- VPUBLIC int NOsh_getChargefmt (NOsh ∗thee, int i)

  Returns format of specified charge map.

- VPUBLIC NOsh_PrintType NOsh_printWhat (NOsh ∗thee, int iprint)

Return an integer ID of the observable to print (.

- VPUBLIC int NOsh_printNarg (NOsh ∗thee, int iprint)

  Return number of arguments to PRINT statement (.

- VPUBLIC int NOsh_elec2calc (NOsh ∗thee, int icalc)

  Return the name of an elec statement.

- VPUBLIC int NOsh_apol2calc (NOsh ∗thee, int icalc)

  Return the name of an apol statement.

- VPUBLIC char ∗ NOsh_elecname (NOsh ∗thee, int ielec)

  Return an integer mapping of an ELEC statement to a calculation ID (.

- VPUBLIC int NOsh_printOp (NOsh ∗thee, int iprint, int iarg)

  Return integer ID for specified operation (.

- VPUBLIC int NOsh_printCalc (NOsh ∗thee, int iprint, int iarg)

  Return calculation ID for specified PRINT statement (.

- VPUBLIC NOsh ∗ NOsh_ctor (int rank, int size)

  Construct NOsh.

- VPUBLIC int NOsh_ctor2 (NOsh ∗thee, int rank, int size)

  FORTRAN stub to construct NOsh.

- VPUBLIC void NOsh_dtor (NOsh ∗∗thee)

  Object destructor.

- VPUBLIC void NOsh_dtor2 (NOsh ∗thee)

  FORTRAN stub for object destructor.

- VPUBLIC NOsh_calc ∗ NOsh_calc_ctor (NOsh_CalcType calctype)

  Construct NOsh_calc.

- VPUBLIC void NOsh_calc_dtor (NOsh_calc ∗∗thee)

  Object destructor.

- VPUBLIC int NOsh_calc_copy (NOsh_calc ∗thee, NOsh_calc ∗source)

  Copy NOsh_calc object into thee.

- VPUBLIC int NOsh_parseInputFile (NOsh ∗thee, char ∗filename)

Parse an input file only from a file.

- VPUBLIC int NOsh_parseInput (NOsh ∗thee, Vio ∗sock)

  Parse an input file from a socket.

- VPRIVATE int **NOsh_parseREAD_MOL** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseREAD_PARM** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseREAD_DIEL** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseREAD_KAPPA** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseREAD_CHARGE** (NOsh ∗thee, Vio ∗sock)
- VPRIVATE int **NOsh_parseREAD_MESH** (NOsh ∗thee, Vio ∗sock)
- VPUBLIC int NOsh_setupElecCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of electrostatics calculations.

- VPUBLIC int NOsh_setupApolCalc (NOsh ∗thee, Valist ∗alist[NOSH_-MAXMOL])

  Setup the series of non-polar calculations.

10.28.1 Detailed Description

Class NOsh methods.

Author

Nathan Baker

Version

Id

nosh.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

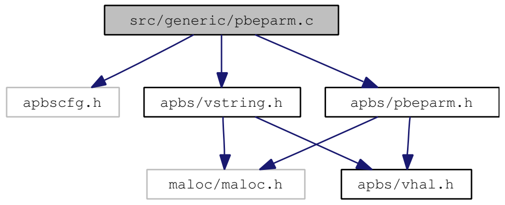
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
```

10.29 src/generic/pbeparm.c File Reference

Class PBEparm methods.

```
#include "apbscfg.h"
#include "apbs/pbeparm.h"
#include "apbs/vstring.h"
```

Include dependency graph for pbeparm.c:



Functions

- VPUBLIC double PBEparm_getIonCharge (PBEparm ∗thee, int i)

    Get charge (e) of specified ion species.

- VPUBLIC double PBEparm_getIonConc (PBEparm ∗thee, int i)

    Get concentration (M) of specified ion species.

- VPUBLIC double PBEparm_getIonRadius (PBEparm ∗thee, int i)

    Get radius (A) of specified ion species.

- VPUBLIC double **PBEparm_getzmem** (PBEparm ∗thee)
- VPUBLIC double **PBEparm_getLmem** (PBEparm ∗thee)
- VPUBLIC double **PBEparm_getmembraneDiel** (PBEparm ∗thee)
- VPUBLIC double **PBEparm_getmemv** (PBEparm ∗thee)
- VPUBLIC PBEparm ∗ PBEparm_ctor ()

    Construct PBEparm object.

- VPUBLIC int PBEparm_ctor2 (PBEparm ∗thee)

    FORTRAN stub to construct PBEparm object.

- VPUBLIC void PBEparm_dtor (PBEparm ∗∗thee)

    Object destructor.

- VPUBLIC void PBEparm_dtor2 (PBEparm ∗thee)

FORTRAN stub for object destructor.

- VPUBLIC int PBEparm_check (PBEparm ∗thee)

  Consistency check for parameter values stored in object.

- VPUBLIC void PBEparm_copy (PBEparm ∗thee, PBEparm ∗parm)

  Copy PBEparm object into thee.

- VPRIVATE int **PBEparm_parseLPBE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseNPBE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseMOL** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseLRPBE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseNRPBE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSMPBE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseBCFL** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseION** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parsePDIE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSDENS** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSDIE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSRFM** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSRAD** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseSWIN** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseTEMP** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseUSEMAP** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseCALCENERGY** (PBEparm ∗thee, Vio ∗sock)

- VPRIVATE int **PBEparm_parseCALCFORCE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseZMEM** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseLMEM** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseMDIE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseMEMV** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseWRITE** (PBEparm ∗thee, Vio ∗sock)
- VPRIVATE int **PBEparm_parseWRITEMAT** (PBEparm ∗thee, Vio ∗sock)
- VPUBLIC int PBEparm_parseToken (PBEparm ∗thee, char tok[VMAX_-BUFSIZE], Vio ∗sock)

  Parse a keyword from an input file.

10.29.1 Detailed Description

Class PBEparm methods.

Author

Nathan Baker

Version

Id

pbeparm.c 1552 2010-02-10 17:46:27Z yhuang01

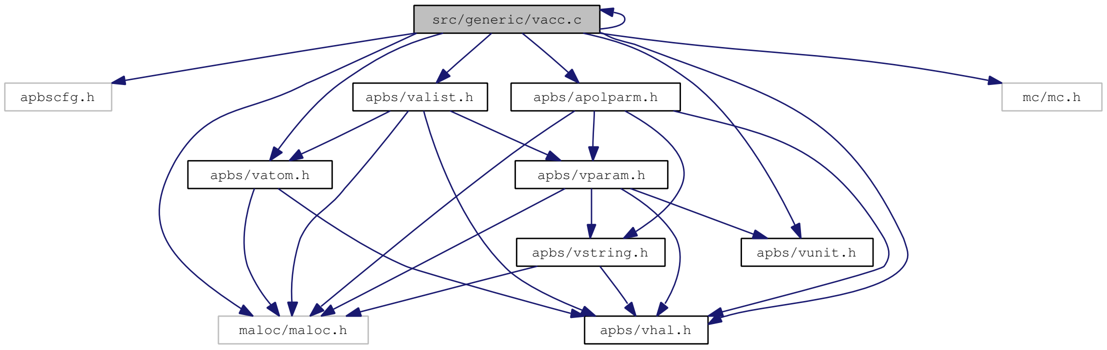
Attention

10.30 src/generic/vacc.c File Reference

Class Vacc methods.

```
#include "apbscfg.h"
#include "apbs/vacc.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/valist.h"
#include "apbs/vatom.h"
#include "apbs/vunit.h"
#include "apbs/apolparm.h"
#include "mc/mc.h"
```

Include dependency graph for vacc.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC unsigned long int Vacc_memChk (Vacc ∗thee)

  Get number of bytes in this object and its members.

- VPRIVATE int ivdwAccExclus (Vacc ∗thee, double center[3], double radius, int atomID)

Determines if a point is within the union of the spheres centered at the atomic centers with radii equal to the sum of their van der Waals radii and the probe radius. Does not include contributions from the specified atom.

- VPUBLIC Vacc ∗ Vacc_ctor (Valist ∗alist, Vclist ∗clist, double surf_density)

  Construct the accessibility object.

- VPRIVATE int Vacc_storeParms (Vacc ∗thee, Valist ∗alist, Vclist ∗clist, double surf_density)
- VPRIVATE int Vacc_allocate (Vacc ∗thee)
- VPUBLIC int Vacc_ctor2 (Vacc ∗thee, Valist ∗alist, Vclist ∗clist, double surf_-density)

  FORTRAN stub to construct the accessibility object.

- VPUBLIC void Vacc_dtor (Vacc ∗∗thee)

  Destroy object.

- VPUBLIC void Vacc_dtor2 (Vacc ∗thee)

  FORTRAN stub to destroy object.

- VPUBLIC double **Vacc_vdwAcc** (Vacc ∗thee, double center[3])
- VPUBLIC double **Vacc_ivdwAcc** (Vacc ∗thee, double center[3], double radius)
- VPUBLIC void Vacc_splineAccGradAtomNorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗grad)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by the accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VPUBLIC void Vacc_splineAccGradAtomUnnorm (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗grad)

  Report gradient of spline-based accessibility with respect to a particular atom (see Vpmg_splineAccAtom).

- VPUBLIC double Vacc_splineAccAtom (Vacc ∗thee, double center[VAPBS_-DIM], double win, double infrad, Vatom ∗atom)

  Report spline-based accessibility for a given atom.

- VPRIVATE double splineAcc (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, VclistCell ∗cell)

  Fast spline-based surface computation subroutine.

- VPUBLIC double Vacc_splineAcc (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad)

Report spline-based accessibility.

- VPUBLIC void Vacc_splineAccGrad (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, double ∗grad)

  Report gradient of spline-based accessibility.

- VPUBLIC double Vacc_molAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report molecular accessibility.

- VPUBLIC double Vacc_fastMolAcc (Vacc ∗thee, double center[VAPBS_DIM], double radius)

  Report molecular accessibility quickly.

- VPUBLIC void **Vacc_writeGMV** (Vacc ∗thee, double radius, int meth, Gem ∗gm, char ∗iodev, char ∗iofmt, char ∗iohost, char ∗iofile)
- VPUBLIC double Vacc_SASA (Vacc ∗thee, double radius)

  Build the solvent accessible surface (SAS) and calculate the solvent accessible surface area.

- VPUBLIC double Vacc_totalSASA (Vacc ∗thee, double radius)

  Return the total solvent accessible surface area (SASA).

- VPUBLIC double Vacc_atomSASA (Vacc ∗thee, double radius, Vatom ∗atom)

  Return the atomic solvent accessible surface area (SASA).

- VPUBLIC VaccSurf ∗ VaccSurf_ctor (Vmem ∗mem, double probe_radius, int nsphere)

  Allocate and construct the surface object; do not assign surface points to positions.

- VPUBLIC int VaccSurf_ctor2 (VaccSurf ∗thee, Vmem ∗mem, double probe_-radius, int nsphere)

  Construct the surface object using previously allocated memory; do not assign surface points to positions.

- VPUBLIC void VaccSurf_dtor (VaccSurf ∗∗thee)

  Destroy the surface object and free its memory.

- VPUBLIC void VaccSurf_dtor2 (VaccSurf ∗thee)

  Destroy the surface object.

- VPUBLIC VaccSurf ∗ Vacc_atomSurf (Vacc ∗thee, Vatom ∗atom, VaccSurf ∗ref, double prad)

Set up an array of points corresponding to the SAS due to a particular atom.

- VPUBLIC VaccSurf ∗ VaccSurf_refSphere (Vmem ∗mem, int npts)

  Set up an array of points for a reference sphere of unit radius.

- VPUBLIC VaccSurf ∗ Vacc_atomSASPoints (Vacc ∗thee, double radius, Vatom ∗atom)

  Get the set of points for this atom's solvent-accessible surface.

- VPUBLIC void Vacc_splineAccGradAtomNorm4 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗grad)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 4th order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VPUBLIC void Vacc_splineAccGradAtomNorm3 (Vacc ∗thee, double center[VAPBS_DIM], double win, double infrad, Vatom ∗atom, double ∗grad)

  Report gradient of spline-based accessibility with respect to a particular atom normalized by a 3rd order accessibility value due to that atom at that point (see Vpmg_-splineAccAtom).

- VPUBLIC void Vacc_atomdSAV (Vacc ∗thee, double srad, Vatom ∗atom, double ∗dSA)

  Get the derivatve of solvent accessible volume.

- VPRIVATE double **Vacc_SASAPos** (Vacc ∗thee, double radius)
- VPRIVATE double **Vacc_atomSASAPos** (Vacc ∗thee, double radius, Vatom ∗atom, int mode)
- VPUBLIC void Vacc_atomdSASA (Vacc ∗thee, double dpos, double srad, Vatom ∗atom, double ∗dSA)

  Get the derivatve of solvent accessible area.

- VPUBLIC void Vacc_totalAtomdSASA (Vacc ∗thee, double dpos, double srad, Vatom ∗atom, double ∗dSA)

  Testing purposes only.

- VPUBLIC void Vacc_totalAtomdSAV (Vacc ∗thee, double dpos, double srad, Vatom ∗atom, double ∗dSA, Vclist ∗clist)

  Total solvent accessible volume.

- VPUBLIC double Vacc_totalSAV (Vacc ∗thee, Vclist ∗clist, APOLparm ∗apolparm, double radius)

Return the total solvent accessible volume (SAV).

- int **Vacc_wcaEnergyAtom** (Vacc ∗thee, APOLparm ∗apolparm, Valist ∗alist, Vclist ∗clist, int iatom, double ∗value)
- VPUBLIC int Vacc_wcaEnergy (Vacc ∗acc, APOLparm ∗apolparm, Valist ∗alist, Vclist ∗clist)

  Return the WCA integral energy.
- VPUBLIC int Vacc_wcaForceAtom (Vacc ∗thee, APOLparm ∗apolparm, Vclist ∗clist, Vatom ∗atom, double ∗force)

  Return the WCA integral force.

10.30.1 Detailed Description

Class Vacc methods.

Author

Nathan Baker

Version

Id

vacc.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
 *
 * APBS -- Adaptive Poisson-Boltzmann Solver
 *
 * Nathan A. Baker (baker@biochem.wustl.edu)
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
```

10.30.2 Function Documentation

10.30.2.1 VPRIVATE int ivdwAccExclus (Vacc ∗ *thee*, double *center*[3], double *radius*, int *atomID*)

Determines if a point is within the union of the spheres centered at the atomic centers with radii equal to the sum of their van der Waals radii and the probe radius. Does not include contributions from the specified atom.

Returns

1 if accessible (outside the inflated van der Waals radius), 0 otherwise

Author

Nathan Baker

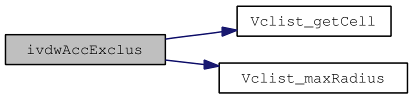
Parameters

center Accessibility object

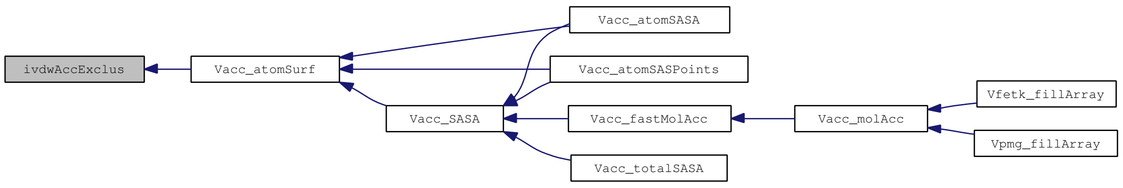
radius Position to test

atomID Radius of probe ID of atom to ignore

Here is the call graph for this function:



Here is the caller graph for this function:



10.30.2.2 VPRIVATE double splineAcc (Vacc $*$ *thee*, double *center*[VAPBS_DIM], double *win*, double *infrad*, VclistCell $*$ *cell*)

Fast spline-based surface computation subroutine.

Returns

Spline value

Author

Todd Dolinsky and Nathan Baker

Parameters

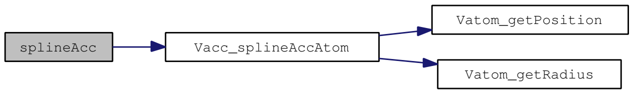
center Accessibility object

win Point at which the acc is to be evaluated

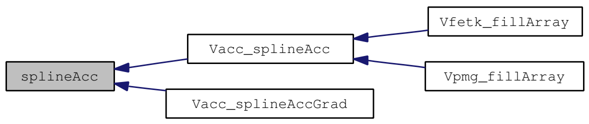
infrad Spline window

cell Radius to inflate atomic radius Cell of atom objects

Here is the call graph for this function:



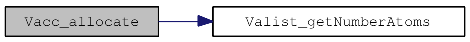
Here is the caller graph for this function:



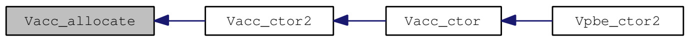
10.30.2.3 VPRIVATE int Vacc_allocate (Vacc ∗ *thee*)

Allocate (and clear) space for storage

Here is the call graph for this function:



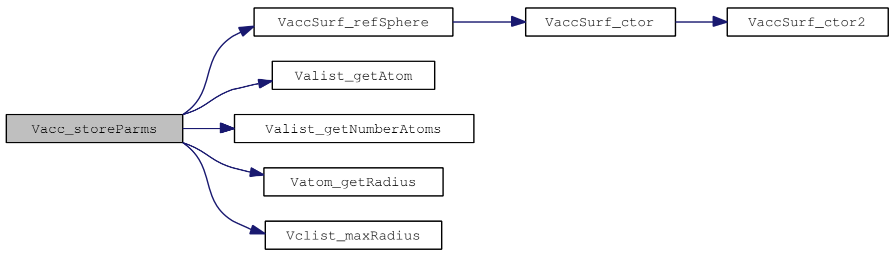
Here is the caller graph for this function:



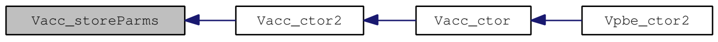
10.30.2.4 VPRIVATE int Vacc_storeParms (Vacc ∗ *thee*, Valist ∗ *alist*, Vclist ∗ *clist*, double *surf_density*)

Check and store parameters passed to constructor

Here is the call graph for this function:



Here is the caller graph for this function:

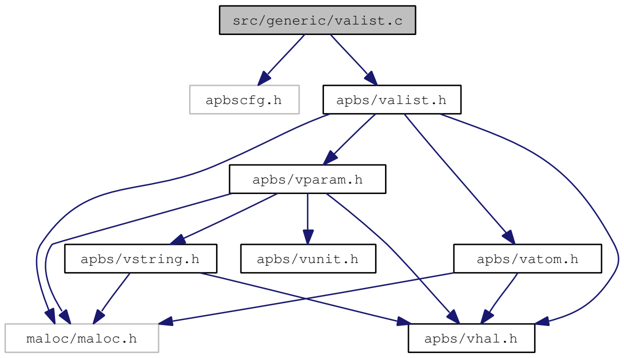


10.31 src/generic/valist.c File Reference

Class Valist methods.

```
#include "apbscfg.h"
#include "apbs/valist.h"
```

Include dependency graph for valist.c:



Functions

- VPUBLIC double Valist_getCenterX (Valist ∗thee)

    Get x-coordinate of molecule center.

- VPUBLIC double Valist_getCenterY (Valist ∗thee)

    Get y-coordinate of molecule center.

- VPUBLIC double Valist_getCenterZ (Valist ∗thee)

    Get z-coordinate of molecule center.

- VPUBLIC Vatom ∗ Valist_getAtomList (Valist ∗thee)

    Get actual array of atom objects from the list.

- VPUBLIC int Valist_getNumberAtoms (Valist ∗thee)

    Get number of atoms in the list.

- VPUBLIC Vatom ∗ Valist_getAtom (Valist ∗thee, int i)

    Get pointer to particular atom in list.

- VPUBLIC unsigned long int Valist_memChk (Valist ∗thee)

    Get total memory allocated for this object and its members.

- VPUBLIC Valist ∗ Valist_ctor ()

  Construct the atom list object.

- VPUBLIC Vrc_Codes Valist_ctor2 (Valist ∗thee)

  FORTRAN stub to construct the atom list object.

- VPUBLIC void Valist_dtor (Valist ∗∗thee)

  Destroys atom list object.

- VPUBLIC void Valist_dtor2 (Valist ∗thee)

  FORTRAN stub to destroy atom list object.

- VPRIVATE Vrc_Codes **Valist_readPDBSerial** (Valist ∗thee, Vio ∗sock, int ∗serial)
- VPRIVATE Vrc_Codes **Valist_readPDBAtomName** (Valist ∗thee, Vio ∗sock, char atomName[VMAX_ARGLEN])
- VPRIVATE Vrc_Codes **Valist_readPDBResidueName** (Valist ∗thee, Vio ∗sock, char resName[VMAX_ARGLEN])
- VPRIVATE Vrc_Codes **Valist_readPDBResidueNumber** (Valist ∗thee, Vio ∗sock, int ∗resSeq)
- VPRIVATE Vrc_Codes **Valist_readPDBAtomCoord** (Valist ∗thee, Vio ∗sock, double ∗coord)
- VPRIVATE Vrc_Codes **Valist_readPDBChargeRadius** (Valist ∗thee, Vio ∗sock, double ∗charge, double ∗radius)
- VPRIVATE Vrc_Codes **Valist_readPDB_throughXYZ** (Valist ∗thee, Vio ∗sock, int ∗serial, char atomName[VMAX_ARGLEN], char resName[VMAX_-ARGLEN], int ∗resSeq, double ∗x, double ∗y, double ∗z)
- VPRIVATE Vatom ∗ **Valist_getAtomStorage** (Valist ∗thee, Vatom ∗∗plist, int ∗pnlist, int ∗pnatoms)
- VPRIVATE Vrc_Codes **Valist_setAtomArray** (Valist ∗thee, Vatom ∗∗plist, int nlist, int natoms)
- VPUBLIC Vrc_Codes Valist_readPDB (Valist ∗thee, Vparam ∗param, Vio ∗sock)

  Fill atom list with information from a PDB file.

- VPUBLIC Vrc_Codes Valist_readPQR (Valist ∗thee, Vparam ∗params, Vio ∗sock)

  Fill atom list with information from a PQR file.

- VPUBLIC Vrc_Codes Valist_readXML (Valist ∗thee, Vparam ∗params, Vio ∗sock)

  Fill atom list with information from an XML file.

- VPUBLIC Vrc_Codes Valist_getStatistics (Valist ∗thee)

  Load up Valist with various statistics.

Variables

- VPRIVATE char ∗ **Valist_whiteChars** = " \t\r\n"
- VPRIVATE char ∗ **Valist_commChars** = "#%"
- VPRIVATE char ∗ **Valist_xmlwhiteChars** = " \t\r\n<>"

10.31.1 Detailed Description

Class Valist methods.

Author

Nathan Baker

Version

Id

valist.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

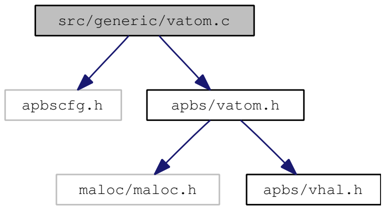
10.32 src/generic/vatom.c File Reference

Class Vatom methods.

`#include "apbscfg.h"`

`#include "apbs/vatom.h"`

Include dependency graph for vatom.c:



Functions

- VPUBLIC double ∗ Vatom_getPosition (Vatom ∗thee)

  Get atomic position.

- VPUBLIC double Vatom_getPartID (Vatom ∗thee)

  Get partition ID.

- VPUBLIC void Vatom_setPartID (Vatom ∗thee, int partID)

  Set partition ID.

- VPUBLIC double Vatom_getAtomID (Vatom ∗thee)

  Get atom ID.

- VPUBLIC void Vatom_setAtomID (Vatom ∗thee, int atomID)

  Set atom ID.

- VPUBLIC void Vatom_setRadius (Vatom ∗thee, double radius)

  Set atomic radius.

- VPUBLIC double Vatom_getRadius (Vatom ∗thee)

  Get atomic position.

- VPUBLIC void Vatom_setCharge (Vatom ∗thee, double charge)

  Set atomic charge.

- VPUBLIC double Vatom_getCharge (Vatom ∗thee)

  Get atomic charge.

- VPUBLIC unsigned long int Vatom_memChk (Vatom ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC Vatom ∗ Vatom_ctor ()

  Constructor for the Vatom class.

- VPUBLIC int Vatom_ctor2 (Vatom ∗thee)

  FORTRAN stub constructor for the Vatom class.

- VPUBLIC void Vatom_dtor (Vatom ∗∗thee)

  Object destructor.

- VPUBLIC void Vatom_dtor2 (Vatom ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC void Vatom_setPosition (Vatom ∗thee, double position[3])

  Set the atomic position.

- VPUBLIC void Vatom_copyTo (Vatom ∗thee, Vatom ∗dest)

  Copy information to another atom.

- VPUBLIC void Vatom_copyFrom (Vatom ∗thee, Vatom ∗src)

  Copy information to another atom.

- VPUBLIC void Vatom_setResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Set residue name.

- VPUBLIC void Vatom_getResName (Vatom ∗thee, char resName[VMAX_-RECLEN])

  Retrieve residue name.

- VPUBLIC void Vatom_setAtomName (Vatom ∗thee, char atomName[VMAX_-RECLEN])

  Set atom name.

- VPUBLIC void Vatom_getAtomName (Vatom ∗thee, char atomName[VMAX_-RECLEN])

  Retrieve atom name.

10.32.1 Detailed Description

Class Vatom methods.

Author

Nathan Baker

Version

Id

vatom.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*
```

10.33 src/generic/vcap.c File Reference

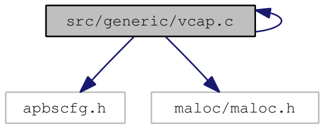
Class Vcap methods.

```
#include "apbscfg.h"
```

```
#include "apbs/vcap.h"
```

```
#include "maloc/maloc.h"
```

Include dependency graph for vcap.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC double Vcap_exp (double x, int ∗ichop)

  Provide a capped exp() function.

- VPUBLIC double Vcap_sinh (double x, int ∗ichop)

  Provide a capped sinh() function.

- VPUBLIC double Vcap_cosh (double x, int ∗ichop)

  Provide a capped cosh() function.

10.33.1 Detailed Description

Class Vcap methods.

Author

Nathan Baker

Version

Id

vcap.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

10.34 src/generic/vclist.c File Reference

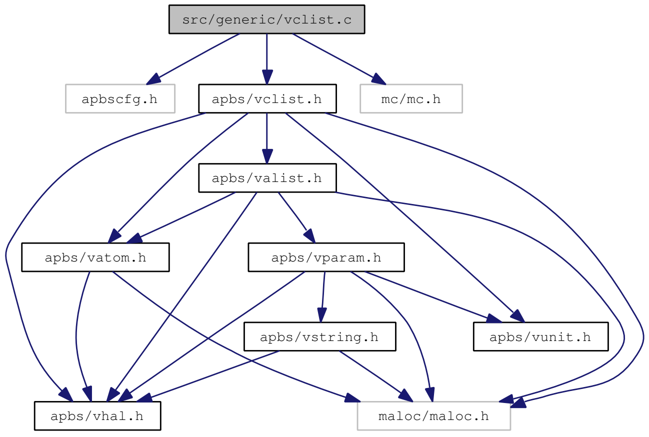
Class Vclist methods.

`#include "apbscfg.h"`

`#include "apbs/vclist.h"`

`#include "mc/mc.h"`

Include dependency graph for vclist.c:



Defines

- #define **VCLIST_INFLATE** 1.42

Functions

- VPUBLIC unsigned long int Vclist_memChk (Vclist ∗thee)

  Get number of bytes in this object and its members.

- VPUBLIC double Vclist_maxRadius (Vclist ∗thee)

  Get the max probe radius value (in A) the cell list was constructed with.

- VPUBLIC Vclist ∗ Vclist_ctor (Valist ∗alist, double max_radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  Construct the cell list object.

- VPRIVATE void **Vclist_getMolDims** (Vclist ∗thee, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM], double ∗r_max)
- VPRIVATE Vrc_Codes **Vclist_setupGrid** (Vclist ∗thee)
- VPRIVATE Vrc_Codes **Vclist_storeParms** (Vclist ∗thee, Valist ∗alist, double max_radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])
- VPRIVATE void **Vclist_gridSpan** (Vclist ∗thee, Vatom ∗atom, int imin[VAPBS_DIM], int imax[VAPBS_DIM])
- VPRIVATE int **Vclist_arrayIndex** (Vclist ∗thee, int i, int j, int k)
- VPRIVATE Vrc_Codes **Vclist_assignAtoms** (Vclist ∗thee)
- VPUBLIC Vrc_Codes Vclist_ctor2 (Vclist ∗thee, Valist ∗alist, double max_-radius, int npts[VAPBS_DIM], Vclist_DomainMode mode, double lower_-corner[VAPBS_DIM], double upper_corner[VAPBS_DIM])

  FORTRAN stub to construct the cell list object.
- VPUBLIC void Vclist_dtor (Vclist ∗∗thee)

  Destroy object.
- VPUBLIC void Vclist_dtor2 (Vclist ∗thee)

  FORTRAN stub to destroy object.
- VPUBLIC VclistCell ∗ Vclist_getCell (Vclist ∗thee, double pos[VAPBS_-DIM])

  Return cell corresponding to specified position or return VNULL.
- VPUBLIC VclistCell ∗ VclistCell_ctor (int natoms)

  Allocate and construct a cell list cell object.
- VPUBLIC Vrc_Codes VclistCell_ctor2 (VclistCell ∗thee, int natoms)

  Construct a cell list object.
- VPUBLIC void VclistCell_dtor (VclistCell ∗∗thee)

  Destroy object.
- VPUBLIC void VclistCell_dtor2 (VclistCell ∗thee)

  FORTRAN stub to destroy object.

10.34.1 Detailed Description

Class Vclist methods.

Author

Nathan Baker

Version

Id

vclist.c 1552 2010-02-10 17:46:27Z yhuang01

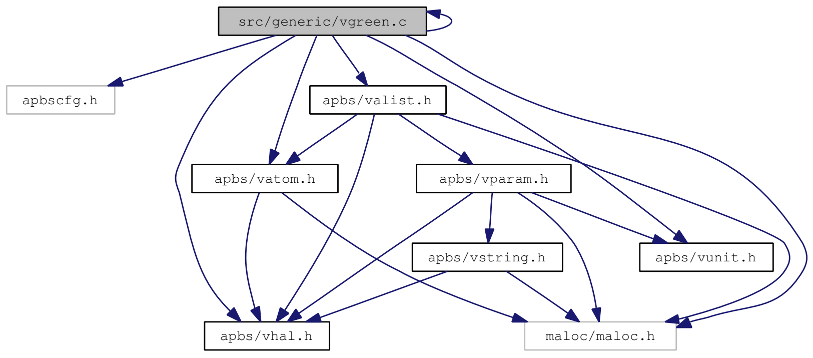
Attention

10.35 src/generic/vgreen.c File Reference

Class Vgreen methods.

```
#include "apbscfg.h"
#include "apbs/vgreen.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vunit.h"
#include "apbs/vatom.h"
#include "apbs/valist.h"
```

Include dependency graph for vgreen.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPRIVATE int **treesetup** (Vgreen ∗thee)
- VPRIVATE int **treecleanup** (Vgreen ∗thee)
- VPRIVATE int **treecalc** (Vgreen ∗thee, double ∗xtar, double ∗ytar, double ∗ztar, double ∗qtar, int numtars, double ∗tpengtar, double ∗x, double ∗y, double ∗z, double ∗q, int numpars, double ∗fx, double ∗fy, double ∗fz, int iflag, int farrdim, int arrdim)
- VPUBLIC Valist ∗ Vgreen_getValist (Vgreen ∗thee)

    Get the atom list associated with this Green's function object.

- VPUBLIC unsigned long int Vgreen_memChk (Vgreen ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC Vgreen ∗ Vgreen_ctor (Valist ∗alist)

  Construct the Green's function oracle.

- VPUBLIC int Vgreen_ctor2 (Vgreen ∗thee, Valist ∗alist)

  FORTRAN stub to construct the Green's function oracle.

- VPUBLIC void Vgreen_dtor (Vgreen ∗∗thee)

  Destruct the Green's function oracle.

- VPUBLIC void Vgreen_dtor2 (Vgreen ∗thee)

  FORTRAN stub to destruct the Green's function oracle.

- VPUBLIC int Vgreen_helmholtz (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val, double kappa)

  Get the Green's function for Helmholtz's equation integrated over the atomic point charges.

- VPUBLIC int Vgreen_helmholtzD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗gradx, double ∗grady, double ∗gradz, double kappa)

  Get the gradient of Green's function for Helmholtz's equation integrated over the atomic point charges.

- VPUBLIC int Vgreen_coulomb_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

  Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VPUBLIC int Vgreen_coulomb (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗val)

  Get the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation or H. E. Johnston, R. Krasny FMM library (if available).

- VPUBLIC int Vgreen_coulombD_direct (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using direct summation.

- VPUBLIC int Vgreen_coulombD (Vgreen ∗thee, int npos, double ∗x, double ∗y, double ∗z, double ∗pot, double ∗gradx, double ∗grady, double ∗gradz)

  Get gradient of the Coulomb's Law Green's function (solution to Laplace's equation) integrated over the atomic point charges using either direct summation or H. E. Johnston/R. Krasny FMM library (if available).

10.35.1 Detailed Description

Class Vgreen methods.

Author

Nathan Baker

Version

Id

vgreen.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

10.36 src/generic/vparam.c File Reference

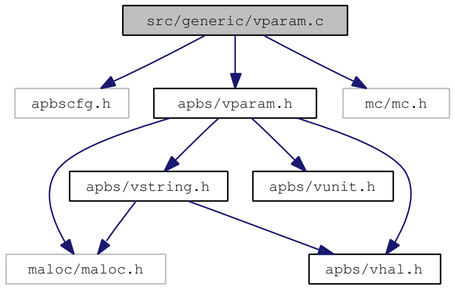
Class Vparam methods.

`#include "apbscfg.h"`

`#include "apbs/vparam.h"`

`#include "mc/mc.h"`

Include dependency graph for vparam.c:



Functions

- VPRIVATE int readFlatFileLine (Vio ∗sock, Vparam_AtomData ∗atom)

    Read a single line of the flat file database.

- VPRIVATE int readXMLFileAtom (Vio ∗sock, Vparam_AtomData ∗atom)

    Read atom information from an XML file.

- VPUBLIC unsigned long int Vparam_memChk (Vparam ∗thee)

    Get number of bytes in this object and its members.

- VPUBLIC Vparam_AtomData ∗ Vparam_AtomData_ctor ()

    Construct the object.

- VPUBLIC int Vparam_AtomData_ctor2 (Vparam_AtomData ∗thee)

    FORTRAN stub to construct the object.

- VPUBLIC void Vparam_AtomData_dtor (Vparam_AtomData ∗∗thee)

    Destroy object.

- VPUBLIC void Vparam_AtomData_dtor2 (Vparam_AtomData ∗thee)

    FORTRAN stub to destroy object.

- VPUBLIC Vparam_ResData ∗ Vparam_ResData_ctor (Vmem ∗mem)

  Construct the object.

- VPUBLIC int Vparam_ResData_ctor2 (Vparam_ResData ∗thee, Vmem ∗mem)

  FORTRAN stub to construct the object.

- VPUBLIC void Vparam_ResData_dtor (Vparam_ResData ∗∗thee)

  Destroy object.

- VPUBLIC void Vparam_ResData_dtor2 (Vparam_ResData ∗thee)

  FORTRAN stub to destroy object.

- VPUBLIC Vparam ∗ Vparam_ctor ()

  Construct the object.

- VPUBLIC int Vparam_ctor2 (Vparam ∗thee)

  FORTRAN stub to construct the object.

- VPUBLIC void Vparam_dtor (Vparam ∗∗thee)

  Destroy object.

- VPUBLIC void Vparam_dtor2 (Vparam ∗thee)

  FORTRAN stub to destroy object.

- VPUBLIC Vparam_ResData ∗ Vparam_getResData (Vparam ∗thee, char resName[VMAX_ARGLEN])

  Get residue data.

- VPUBLIC Vparam_AtomData ∗ Vparam_getAtomData (Vparam ∗thee, char resName[VMAX_ARGLEN], char atomName[VMAX_ARGLEN])

  Get atom data.

- VPUBLIC int Vparam_readXMLFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read an XML format parameter database.

- VPUBLIC int Vparam_readFlatFile (Vparam ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read a flat-file format parameter database.

- VEXTERNC void Vparam_AtomData_copyTo (Vparam_AtomData ∗thee, Vparam_AtomData ∗dest)

  Copy current atom object to destination.

- VEXTERNC void Vparam_ResData_copyTo (Vparam_ResData ∗thee, Vparam_ResData ∗dest)

  Copy current residue object to destination.

- VEXTERNC void Vparam_AtomData_copyFrom (Vparam_AtomData ∗thee, Vparam_AtomData ∗src)

  Copy current atom object from another.

Variables

- VPRIVATE char ∗ MCwhiteChars = " =,;\t\n\r"

  Whitespace characters for socket reads.

- VPRIVATE char ∗ MCcommChars = "#%"

  Comment characters for socket reads.

- VPRIVATE char ∗ MCxmlwhiteChars = " =,;\t\n\r<>"

  Whitespace characters for XML socket reads.

10.36.1 Detailed Description

Class Vparam methods.

Author

Nathan Baker

Version

Id

vparam.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
```

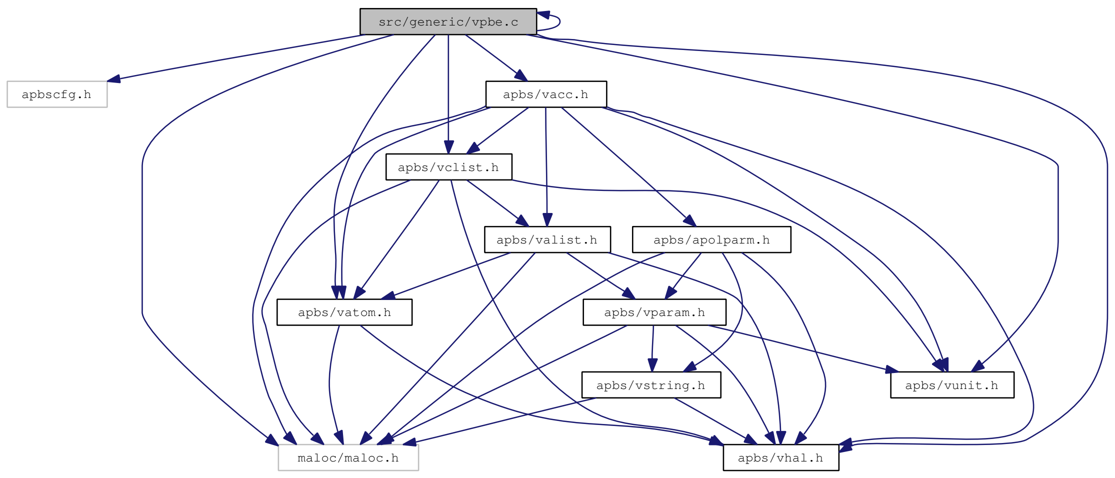
```
 *
 * Nathan A. Baker (baker@biochem.wustl.edu)
 * Dept. of Biochemistry and Molecular Biophysics
 * Center for Computational Biology
 * Washington University in St. Louis
 *
 * Additional contributing authors listed in the code documentation.
 *
 * Copyright (c) 2002-2010, Washington University in St. Louis.
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * -  Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * - Neither the name of Washington University in St. Louis nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

10.37 src/generic/vpbe.c File Reference

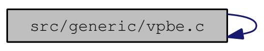
Class Vpbe methods.

```
#include "apbscfg.h"
#include "apbs/vpbe.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vunit.h"
#include "apbs/vatom.h"
#include "apbs/vacc.h"
#include "apbs/vclist.h"
```

Include dependency graph for vpbe.c:



This graph shows which files directly or indirectly include this file:



Defines

- #define **MAX_SPLINE_WINDOW** 0.5

Functions

- VPUBLIC Valist ∗ Vpbe_getValist (Vpbe ∗thee)

  Get atom list.

- VPUBLIC Vacc ∗ Vpbe_getVacc (Vpbe ∗thee)

  Get accessibility oracle.

- VPUBLIC double Vpbe_getBulkIonicStrength (Vpbe ∗thee)

  Get bulk ionic strength.

- VPUBLIC double Vpbe_getTemperature (Vpbe ∗thee)

  Get temperature.

- VPUBLIC double Vpbe_getSoluteDiel (Vpbe ∗thee)

  Get solute dielectric constant.

- VPUBLIC double ∗ Vpbe_getSoluteCenter (Vpbe ∗thee)

  Get coordinates of solute center.

- VPUBLIC double Vpbe_getSolventDiel (Vpbe ∗thee)

  Get solvent dielectric constant.

- VPUBLIC double Vpbe_getSolventRadius (Vpbe ∗thee)

  Get solvent molecule radius.

- VPUBLIC double Vpbe_getMaxIonRadius (Vpbe ∗thee)

  Get maximum radius of ion species.

- VPUBLIC double Vpbe_getXkappa (Vpbe ∗thee)

  Get Debye-Huckel parameter.

- VPUBLIC double Vpbe_getDeblen (Vpbe ∗thee)

  Get Debye-Huckel screening length.

- VPUBLIC double Vpbe_getZkappa2 (Vpbe ∗thee)

  Get modified squared Debye-Huckel parameter.

- VPUBLIC double Vpbe_getZmagic (Vpbe ∗thee)

  Get charge scaling factor.

- VPUBLIC double Vpbe_getSoluteRadius (Vpbe ∗thee)

  Get sphere radius which bounds biomolecule.

- VPUBLIC double Vpbe_getSoluteXlen (Vpbe ∗thee)

  Get length of solute in x dimension.

- VPUBLIC double Vpbe_getSoluteYlen (Vpbe ∗thee)

  Get length of solute in y dimension.

- VPUBLIC double Vpbe_getSoluteZlen (Vpbe ∗thee)

  Get length of solute in z dimension.

- VPUBLIC double Vpbe_getSoluteCharge (Vpbe ∗thee)

  Get total solute charge.

- VPUBLIC double Vpbe_getzmem (Vpbe ∗thee)

  Get z position of the membrane bottom.

- VPUBLIC double Vpbe_getLmem (Vpbe ∗thee)

  Get length of the membrane (A)
  aauthor Michael Grabe.

- VPUBLIC double Vpbe_getmembraneDiel (Vpbe ∗thee)

  Get membrane dielectric constant.

- VPUBLIC double Vpbe_getmemv (Vpbe ∗thee)

  Get membrane potential (kT).

- VPUBLIC Vpbe ∗ Vpbe_ctor (Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

  Construct Vpbe object.

- VPUBLIC int Vpbe_ctor2 (Vpbe ∗thee, Valist ∗alist, int ionNum, double ∗ionConc, double ∗ionRadii, double ∗ionQ, double T, double soluteDiel, double solventDiel, double solventRadius, int focusFlag, double sdens, double z_mem, double L, double membraneDiel, double V)

  FORTRAN stub to construct Vpbe objct.

- VPUBLIC void Vpbe_dtor (Vpbe ∗∗thee)

  Object destructor.

- VPUBLIC void Vpbe_dtor2 (Vpbe ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC double Vpbe_getCoulombEnergy1 (Vpbe ∗thee)

  Calculate coulombic energy of set of charges.

- VPUBLIC unsigned long int Vpbe_memChk (Vpbe ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC int Vpbe_getIons (Vpbe ∗thee, int ∗nion, double ionConc[MAXION], double ionRadii[MAXION], double ionQ[MAXION])

  Get information about the counterion species present.

10.37.1 Detailed Description

Class Vpbe methods.

Author

Nathan Baker

Version

Id

vpbe.c 1552 2010-02-10 17:46:27Z yhuang01

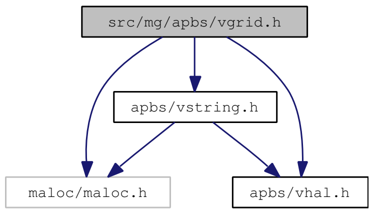
Attention

10.38 src/mg/apbs/vgrid.h File Reference

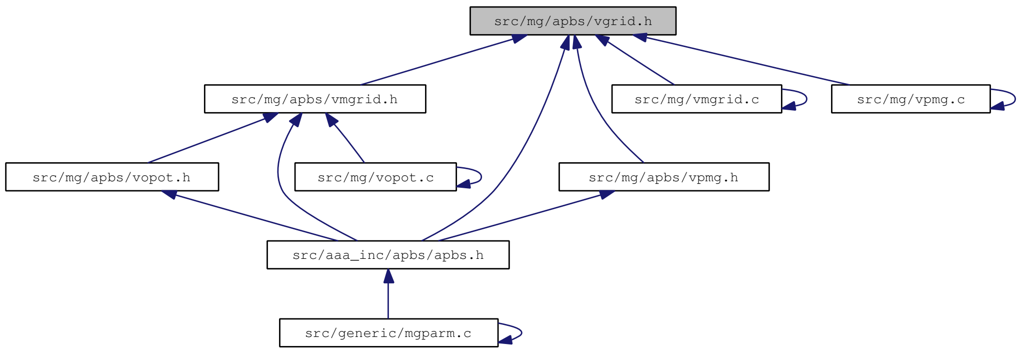
Potential oracle for Cartesian mesh data.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vstring.h"
```

Include dependency graph for vgrid.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVgrid

  Electrostatic potential oracle for Cartesian mesh data.

Defines

- #define VGRID_DIGITS 6

  Number of decimal places for comparisons and formatting.

Typedefs

- typedef struct sVgrid Vgrid

  Declaration of the Vgrid class as the sVgrid structure.

Functions

- VEXTERNC unsigned long int Vgrid_memChk (Vgrid ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vgrid ∗ Vgrid_ctor (int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Construct Vgrid object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vgrid_ctor2 (Vgrid ∗thee, int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Initialize Vgrid object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vgrid_value (Vgrid ∗thee, double x[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vgrid_dtor (Vgrid ∗∗thee)

  Object destructor.

- VEXTERNC void Vgrid_dtor2 (Vgrid ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vgrid_curvature (Vgrid ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vgrid_gradient (Vgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VEXTERNC void Vgrid_writeUHBD (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in UHBD grid format.

- VEXTERNC void Vgrid_writeDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in OpenDX grid format.

- VEXTERNC int Vgrid_readDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read in data in OpenDX grid format.

- VEXTERNC double Vgrid_integrate (Vgrid ∗thee)

  Get the integral of the data.

- VEXTERNC double Vgrid_normL1 (Vgrid ∗thee)

  Get the L_1 norm of the data. This returns the integral:

$$\|u\|_{L_1} = \int_\Omega |u(x)| dx$$

  .

- VEXTERNC double Vgrid_normL2 (Vgrid ∗thee)

  Get the L_2 norm of the data. This returns the integral:

$$\|u\|_{L_2} = \left(\int_\Omega |u(x)|^2 dx \right)^{1/2}$$

  .

- VEXTERNC double Vgrid_normLinf (Vgrid ∗thee)

  Get the L_∞ norm of the data. This returns the integral:

$$\|u\|_{L_\infty} = \sup_{x \in \Omega} |u(x)|$$

  .

- VEXTERNC double Vgrid_seminormH1 (Vgrid ∗thee)

  Get the H_1 semi-norm of the data. This returns the integral:

$$|u|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx \right)^{1/2}$$

  .

- VEXTERNC double Vgrid_normH1 (Vgrid ∗thee)

  Get the H_1 norm (or energy norm) of the data. This returns the integral:

$$\|u\|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx + \int_\Omega |u(x)|^2 dx \right)^{1/2}$$

  .

10.38.1 Detailed Description

Potential oracle for Cartesian mesh data.

Author

Nathan Baker and Steve Bond

Version

Id

vgrid.h 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*
```

10.39 src/mg/apbs/vmgrid.h File Reference

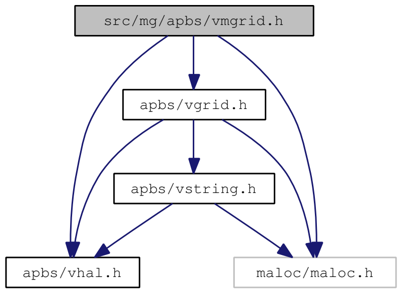
Multiresolution oracle for Cartesian mesh data.

`#include "maloc/maloc.h"`

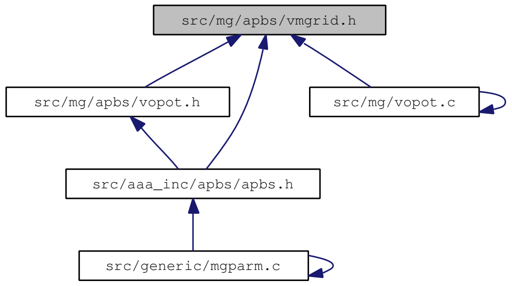
`#include "apbs/vhal.h"`

`#include "apbs/vgrid.h"`

Include dependency graph for vmgrid.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVmgrid

  Multiresoltion oracle for Cartesian mesh data.

Defines

- #define VMGRIDMAX 20

The maximum number of levels in the grid hiearchy.

Typedefs

- typedef struct sVmgrid Vmgrid

  Declaration of the Vmgrid class as the Vgmrid structure.

Functions

- VEXTERNC Vmgrid ∗ Vmgrid_ctor ()

  Construct Vmgrid object.

- VEXTERNC int Vmgrid_ctor2 (Vmgrid ∗thee)

  Initialize Vmgrid object.

- VEXTERNC int Vmgrid_value (Vmgrid ∗thee, double x[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vmgrid_dtor (Vmgrid ∗∗thee)

  Object destructor.

- VEXTERNC void Vmgrid_dtor2 (Vmgrid ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vmgrid_addGrid (Vmgrid ∗thee, Vgrid ∗grid)

  Add a grid to the hierarchy.

- VEXTERNC int Vmgrid_curvature (Vmgrid ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vmgrid_gradient (Vmgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VEXTERNC Vgrid ∗ Vmgrid_getGridByNum (Vmgrid ∗thee, int num)

  Get specific grid in hiearchy.

- VEXTERNC Vgrid ∗ Vmgrid_getGridByPoint (Vmgrid ∗thee, double pt[3])

  Get grid in hiearchy which contains specified point or VNULL.

10.39.1 Detailed Description

Multiresolution oracle for Cartesian mesh data.

Author

Nathan Baker

Version

Id

vmgrid.h 1552 2010-02-10 17:46:27Z yhuang01

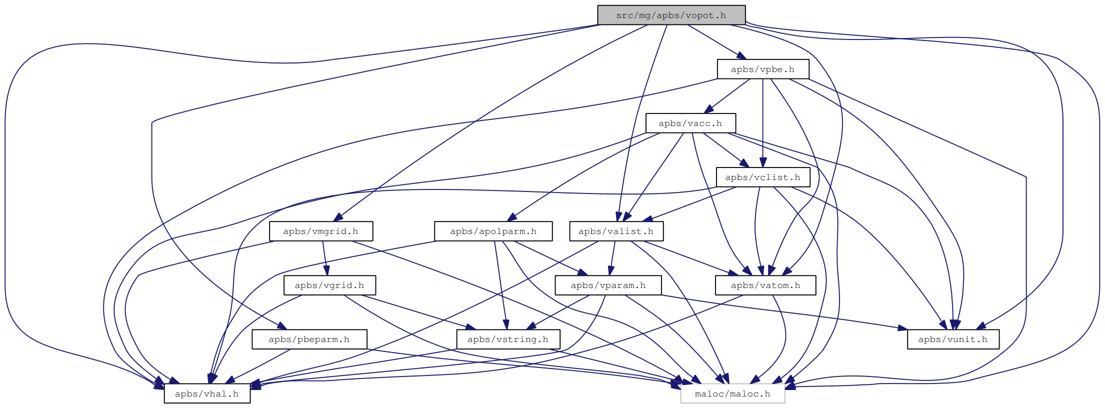
Attention

10.40 src/mg/apbs/vopot.h File Reference

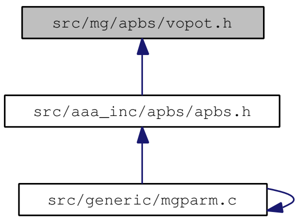
Potential oracle for Cartesian mesh data.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vatom.h"
#include "apbs/valist.h"
#include "apbs/vmgrid.h"
#include "apbs/vunit.h"
#include "apbs/vpbe.h"
#include "apbs/pbeparm.h"
```

Include dependency graph for vopot.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVopot

  Electrostatic potential oracle for Cartesian mesh data.

Typedefs

- typedef struct sVopot Vopot

  Declaration of the Vopot class as the Vopot structure.

Functions

- VEXTERNC Vopot ∗ Vopot_ctor (Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Construct Vopot object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vopot_ctor2 (Vopot ∗thee, Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Initialize Vopot object with values obtained from Vpmg_readDX (for example).

- VEXTERNC int Vopot_pot (Vopot ∗thee, double x[3], double ∗pot)

  Get potential value (from mesh or approximation) at a point.

- VEXTERNC void Vopot_dtor (Vopot ∗∗thee)

  Object destructor.

- VEXTERNC void Vopot_dtor2 (Vopot ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vopot_curvature (Vopot ∗thee, double pt[3], int cflag, double ∗curv)

  Get second derivative values at a point.

- VEXTERNC int Vopot_gradient (Vopot ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

10.40.1 Detailed Description

Potential oracle for Cartesian mesh data.

Author

Nathan Baker

Version

Id

vopot.h 1552 2010-02-10 17:46:27Z yhuang01

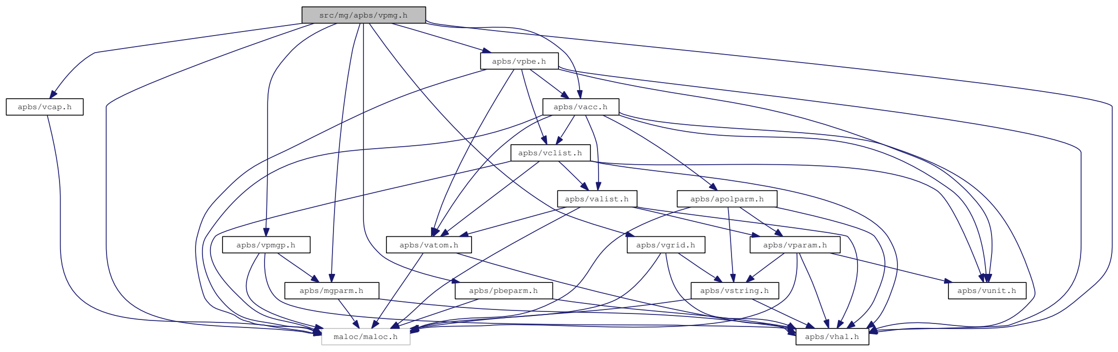
Attention

10.41 src/mg/apbs/vpmg.h File Reference

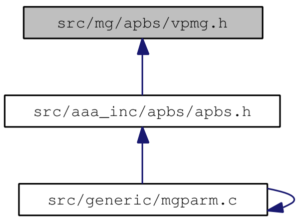
Contains declarations for class Vpmg.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vpmgp.h"
#include "apbs/vacc.h"
#include "apbs/vcap.h"
#include "apbs/vpbe.h"
#include "apbs/vgrid.h"
#include "apbs/mgparm.h"
#include "apbs/pbeparm.h"
```

Include dependency graph for vpmg.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVpmg

  Contains public data members for Vpmg class/module.

Defines

- #define **VPMGMAXPART** 2000

Typedefs

- typedef struct sVpmg Vpmg

  Declaration of the Vpmg class as the Vpmg structure.

Functions

- VEXTERNC unsigned long int Vpmg_memChk (Vpmg ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VEXTERNC Vpmg ∗ Vpmg_ctor (Vpmgp ∗parms, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

  Constructor for the Vpmg class (allocates new memory).

- VEXTERNC int Vpmg_ctor2 (Vpmg ∗thee, Vpmgp ∗parms, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

  FORTRAN stub constructor for the Vpmg class (uses previously-allocated memory).

- VEXTERNC void Vpmg_dtor (Vpmg ∗∗thee)

  Object destructor.

- VEXTERNC void Vpmg_dtor2 (Vpmg ∗thee)

  FORTRAN stub object destructor.

- VEXTERNC int Vpmg_fillco (Vpmg ∗thee, Vsurf_Meth surfMeth, double splineWin, Vchrg_Meth chargeMeth, int useDielXMap, Vgrid ∗dielXMap, int useDielYMap, Vgrid ∗dielYMap, int useDielZMap, Vgrid ∗dielZMap, int useKappaMap, Vgrid ∗kappaMap, int useChargeMap, Vgrid ∗chargeMap)

  Fill the coefficient arrays prior to solving the equation.

- VEXTERNC int Vpmg_solve (Vpmg ∗thee)

  Solve the PBE using PMG.

- VEXTERNC int Vpmg_solveLaplace (Vpmg ∗thee)

  Solve Poisson's equation with a homogeneous Laplacian operator using the solvent dielectric constant. This solution is performed by a sine wave decomposition.

- VEXTERNC double Vpmg_energy (Vpmg ∗thee, int extFlag)

  Get the total electrostatic energy.

- VEXTERNC double Vpmg_qfEnergy (Vpmg ∗thee, int extFlag)

  Get the "fixed charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_qfAtomEnergy (Vpmg ∗thee, Vatom ∗atom)

  Get the per-atom "fixed charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_qmEnergy (Vpmg ∗thee, int extFlag)

  Get the "mobile charge" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_dielEnergy (Vpmg ∗thee, int extFlag)

  Get the "polarization" contribution to the electrostatic energy.

- VEXTERNC double Vpmg_dielGradNorm (Vpmg ∗thee)

  Get the integral of the gradient of the dielectric function.

- VEXTERNC int Vpmg_force (Vpmg ∗thee, double ∗force, int atomID, Vsurf_-Meth srfm, Vchrg_Meth chgm)

  Calculate the total force on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_qfForce (Vpmg ∗thee, double ∗force, int atomID, Vchrg_Meth chgm)

  Calculate the "charge-field" force on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_dbForce (Vpmg ∗thee, double ∗dbForce, int atomID, Vsurf_Meth srfm)

  Calculate the dielectric boundary forces on the specified atom in units of k_B T/AA.

- VEXTERNC int Vpmg_ibForce (Vpmg ∗thee, double ∗force, int atomID, Vsurf_Meth srfm)

  Calculate the osmotic pressure on the specified atom in units of k_B T/AA.

- VEXTERNC void Vpmg_setPart (Vpmg ∗thee, double lowerCorner[3], double upperCorner[3], int bflags[6])

Set partition information which restricts the calculation of observables to a (rectangular) subset of the problem domain.

- VEXTERNC void Vpmg_unsetPart (Vpmg ∗thee)

  Remove partition restrictions.

- VEXTERNC int Vpmg_fillArray (Vpmg ∗thee, double ∗vec, Vdata_Type type, double parm, Vhal_PBEType pbetype)

  Fill the specified array with accessibility values.

- VPUBLIC void Vpmg_fieldSpline4 (Vpmg ∗thee, int atomID, double field[3])

  Computes the field at an atomic center using a stencil based on the first derivative of a 5th order B-spline.

- VEXTERNC double Vpmg_qfPermanentMultipoleEnergy (Vpmg ∗thee, int atomID)

  Computes the permanent multipole electrostatic hydration energy (the polarization component of the hydration energy currently computed in TINKER).

- VEXTERNC void Vpmg_qfPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3], double torque[3])

  Computes the q-Phi Force for permanent multipoles based on 5th order B-splines.

- VEXTERNC void Vpmg_ibPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3])

  Compute the ionic boundary force for permanent multipoles.

- VEXTERNC void Vpmg_dbPermanentMultipoleForce (Vpmg ∗thee, int atomID, double force[3])

  Compute the dielectric boundary force for permanent multipoles.

- VEXTERNC void Vpmg_qfDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3], double torque[3])

  q-Phi direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

- VEXTERNC void Vpmg_qfNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3], double torque[3])

  q-Phi direct polarization force between permanent multipoles and non-local induced dipoles based on 5th Order B-Splines. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_ibDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3])

  Ionic boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

- VEXTERNC void Vpmg_ibNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3])

  Ionic boundary direct polarization force between permanent multipoles and non-local induced dipoles based on 5th order Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_dbDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗induced, int atomID, double force[3])

  Dielectric boundary direct polarization force between permanent multipoles and induced dipoles, which are induced by the sum of the permanent intramolecular field and the permanent reaction field.

- VEXTERNC void Vpmg_dbNLDirectPolForce (Vpmg ∗thee, Vgrid ∗perm, Vgrid ∗nlInduced, int atomID, double force[3])

  Dielectric bounday direct polarization force between permanent multipoles and non-local induced dipoles. Keep in mind that the "non-local" induced dipooles are just a mathematical quantity that result from differentiation of the AMOEBA polarization energy.

- VEXTERNC void Vpmg_qfMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

  Mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_ibMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

  Ionic boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_dbMutualPolForce (Vpmg ∗thee, Vgrid ∗induced, Vgrid ∗nlInduced, int atomID, double force[3])

  Dielectric boundary mutual polarization force for induced dipoles based on 5th order B-Splines. This force arises due to self-consistent convergence of the solute induced dipoles and reaction field.

- VEXTERNC void Vpmg_printColComp (Vpmg ∗thee, char path[72], char title[72], char mxtype[3], int flag)

Print out a column-compressed sparse matrix in Harwell-Boeing format.

10.41.1 Detailed Description

Contains declarations for class Vpmg.

Version

Id

vpmg.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

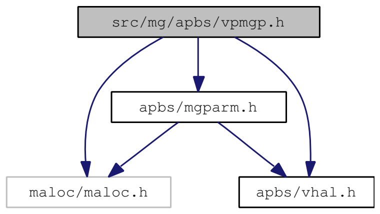
Attention

10.42 src/mg/apbs/vpmgp.h File Reference

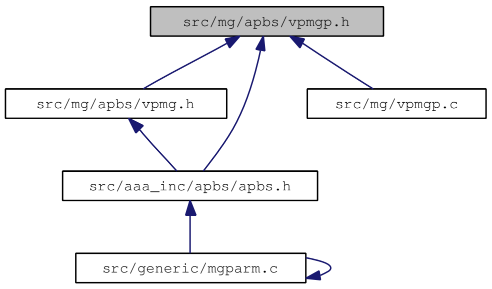
Contains declarations for class Vpmgp.

```
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/mgparm.h"
```

Include dependency graph for vpmgp.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sVpmgp

  Contains public data members for Vpmgp class/module.

Typedefs

- typedef struct sVpmgp Vpmgp

  Declaration of the Vpmgp class as the sVpmgp structure.

Functions

- VEXTERNC Vpmgp ∗ Vpmgp_ctor (MGparm ∗mgparm)

  Construct PMG parameter object and initialize to default values.

- VEXTERNC int Vpmgp_ctor2 (Vpmgp ∗thee, MGparm ∗mgparm)

  FORTRAN stub to construct PMG parameter object and initialize to default values.

- VEXTERNC void Vpmgp_dtor (Vpmgp ∗∗thee)

  Object destructor.

- VEXTERNC void Vpmgp_dtor2 (Vpmgp ∗thee)

  FORTRAN stub for object destructor.

10.42.1 Detailed Description

Contains declarations for class Vpmgp.

Version

Id

vpmgp.h 1552 2010-02-10 17:46:27Z yhuang01

Author

Nathan A. Baker

Note

Variables and many default values taken directly from PMG

Attention

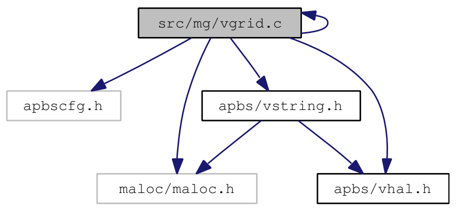
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
```

10.43 src/mg/vgrid.c File Reference

Class Vgrid methods.

```
#include "apbscfg.h"
#include "apbs/vgrid.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vstring.h"
```

Include dependency graph for vgrid.c:



This graph shows which files directly or indirectly include this file:

Defines

- #define **IJK**(i, j, k) (((k)∗(nx)∗(ny))+((j)∗(nx))+(i))

Functions

- VPUBLIC unsigned long int Vgrid_memChk (Vgrid ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC Vgrid ∗ Vgrid_ctor (int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Construct Vgrid object with values obtained from Vpmg_readDX (for example).

- VPUBLIC int Vgrid_ctor2 (Vgrid ∗thee, int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗data)

  Initialize Vgrid object with values obtained from Vpmg_readDX (for example).

- VPUBLIC void Vgrid_dtor (Vgrid ∗∗thee)

  Object destructor.

- VPUBLIC void Vgrid_dtor2 (Vgrid ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC int Vgrid_value (Vgrid ∗thee, double pt[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VPUBLIC int Vgrid_curvature (Vgrid ∗thee, double pt[3], int cflag, double ∗value)

  Get second derivative values at a point.

- VPUBLIC int Vgrid_gradient (Vgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VPUBLIC int Vgrid_readDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname)

  Read in data in OpenDX grid format.

- VPUBLIC void Vgrid_writeDX (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in OpenDX grid format.

- VPUBLIC void Vgrid_writeUHBD (Vgrid ∗thee, const char ∗iodev, const char ∗iofmt, const char ∗thost, const char ∗fname, char ∗title, double ∗pvec)

  Write out the data in UHBD grid format.

- VPUBLIC double Vgrid_integrate (Vgrid ∗thee)

  Get the integral of the data.

- VPUBLIC double Vgrid_normL1 (Vgrid ∗thee)

  Get the L_1 norm of the data. This returns the integral:

  $$\|u\|_{L_1} = \int_\Omega |u(x)| dx$$

  .

- VPUBLIC double Vgrid_normL2 (Vgrid ∗thee)

  Get the L_2 norm of the data. This returns the integral:

  $$\|u\|_{L_2} = \left(\int_\Omega |u(x)|^2 dx \right)^{1/2}$$

  .

- VPUBLIC double Vgrid_seminormH1 (Vgrid *thee)

  Get the H_1 semi-norm of the data. This returns the integral:

$$|u|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx \right)^{1/2}$$

  .

- VPUBLIC double Vgrid_normH1 (Vgrid *thee)

  Get the H_1 norm (or energy norm) of the data. This returns the integral:

$$\|u\|_{H_1} = \left(\int_\Omega |\nabla u(x)|^2 dx + \int_\Omega |u(x)|^2 dx \right)^{1/2}$$

  .

- VPUBLIC double Vgrid_normLinf (Vgrid *thee)

  Get the L_∞ norm of the data. This returns the integral:

$$\|u\|_{L_\infty} = \sup_{x \in \Omega} |u(x)|$$

  .

Variables

- VPRIVATE char * **MCwhiteChars** = " =,;\t\n"
- VPRIVATE char * **MCcommChars** = "#%"
- VPRIVATE double **Vcompare**
- VPRIVATE char **Vprecision** [26]

10.43.1 Detailed Description

Class Vgrid methods.

Author

Nathan Baker

Version

Id

vgrid.c 1552 2010-02-10 17:46:27Z yhuang01

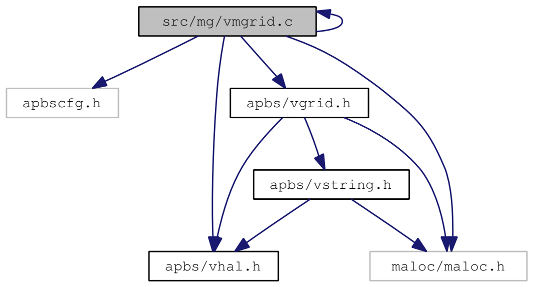
Attention

10.44 src/mg/vmgrid.c File Reference

Class Vmgrid methods.

```
#include "apbscfg.h"
#include "apbs/vmgrid.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vgrid.h"
```

Include dependency graph for vmgrid.c:



This graph shows which files directly or indirectly include this file:

Functions

- VPUBLIC Vmgrid ∗ Vmgrid_ctor ()

  Construct Vmgrid object.

- VPUBLIC int Vmgrid_ctor2 (Vmgrid ∗thee)

  Initialize Vmgrid object.

- VPUBLIC void Vmgrid_dtor (Vmgrid ∗∗thee)

  Object destructor.

- VPUBLIC void Vmgrid_dtor2 (Vmgrid ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC int Vmgrid_value (Vmgrid ∗thee, double pt[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VPUBLIC int Vmgrid_curvature (Vmgrid ∗thee, double pt[3], int cflag, double ∗value)

  Get second derivative values at a point.

- VPUBLIC int Vmgrid_gradient (Vmgrid ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

- VPUBLIC int Vmgrid_addGrid (Vmgrid ∗thee, Vgrid ∗grid)

  Add a grid to the hierarchy.

10.44.1 Detailed Description

Class Vmgrid methods.

Author

Nathan Baker

Version

Id

vmgrid.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

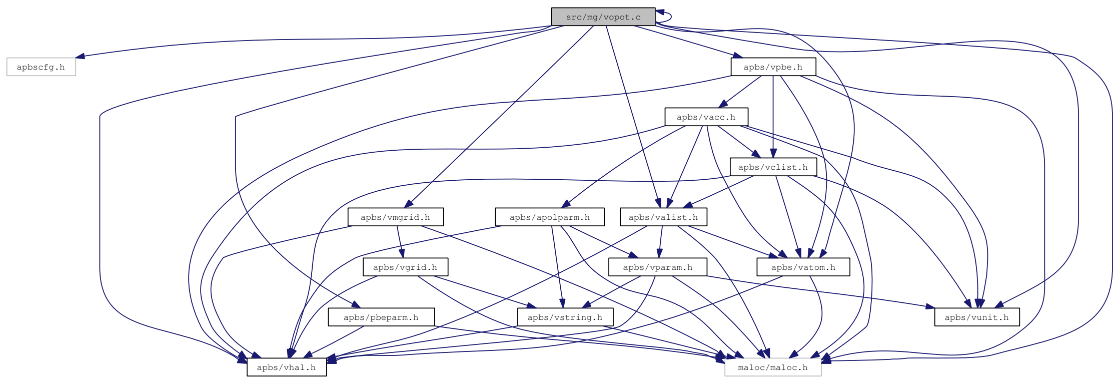
```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
* Portions Copyright (c) 2002-2010.  Nathan A. Baker
* Portions Copyright (c) 1999-2002.  The Regents of the University of California.
* Portions Copyright (c) 1995.  Michael Holst
*
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
```

10.45 src/mg/vopot.c File Reference

Class Vopot methods.

```
#include "apbscfg.h"
#include "apbs/vopot.h"
#include "maloc/maloc.h"
#include "apbs/vhal.h"
#include "apbs/vatom.h"
#include "apbs/valist.h"
#include "apbs/vmgrid.h"
#include "apbs/vunit.h"
#include "apbs/vpbe.h"
#include "apbs/pbeparm.h"
```

Include dependency graph for vopot.c:



This graph shows which files directly or indirectly include this file:

Defines

- #define **IJK**(i, j, k) (((k)∗(nx)∗(ny))+((j)∗(nx))+(i))

Functions

- VPUBLIC Vopot ∗ Vopot_ctor (Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Construct Vopot object with values obtained from Vpmg_readDX (for example).

- VPUBLIC int Vopot_ctor2 (Vopot ∗thee, Vmgrid ∗mgrid, Vpbe ∗pbe, Vbcfl bcfl)

  Initialize Vopot object with values obtained from Vpmg_readDX (for example).

- VPUBLIC void Vopot_dtor (Vopot ∗∗thee)

  Object destructor.

- VPUBLIC void Vopot_dtor2 (Vopot ∗thee)

  FORTRAN stub object destructor.

- VPUBLIC int Vopot_pot (Vopot ∗thee, double pt[3], double ∗value)

  Get potential value (from mesh or approximation) at a point.

- VPUBLIC int Vopot_curvature (Vopot ∗thee, double pt[3], int cflag, double ∗value)

  Get second derivative values at a point.

- VPUBLIC int Vopot_gradient (Vopot ∗thee, double pt[3], double grad[3])

  Get first derivative values at a point.

10.45.1 Detailed Description

Class Vopot methods.

Author

Nathan Baker

Version

Id

vopot.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
```

10.46 src/mg/vpmg.c File Reference

Class Vpmg methods.

`#include "apbscfg.h"`

`#include "vpmg-private.h"`

`#include "apbs/vpmg.h"`

`#include "maloc/maloc.h"`

`#include "apbs/vhal.h"`

`#include "apbs/mgparm.h"`

`#include "apbs/vacc.h"`

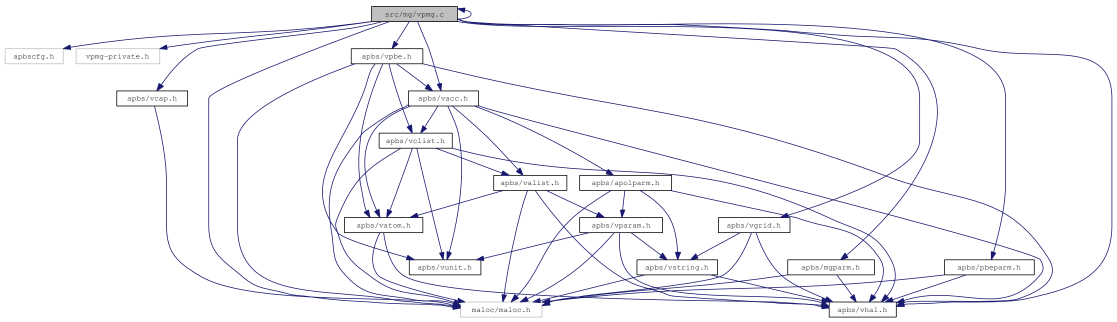
`#include "apbs/vcap.h"`

`#include "apbs/vpbe.h"`

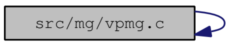
`#include "apbs/vgrid.h"`

`#include "apbs/pbeparm.h"`

Include dependency graph for vpmg.c:



This graph shows which files directly or indirectly include this file:



Functions

- VPUBLIC unsigned long int Vpmg_memChk (Vpmg ∗thee)

  Return the memory used by this structure (and its contents) in bytes.

- VPUBLIC void Vpmg_printColComp (Vpmg ∗thee, char path[72], char title[72], char mxtype[3], int flag)

    Print out a column-compressed sparse matrix in Harwell-Boeing format.

- VPUBLIC Vpmg ∗ Vpmg_ctor (Vpmgp ∗pmgp, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

    Constructor for the Vpmg class (allocates new memory).

- VPUBLIC int Vpmg_ctor2 (Vpmg ∗thee, Vpmgp ∗pmgp, Vpbe ∗pbe, int focusFlag, Vpmg ∗pmgOLD, MGparm ∗mgparm, PBEparm_calcEnergy energyFlag)

    FORTRAN stub constructor for the Vpmg class (uses previously-allocated memory).

- VPUBLIC int Vpmg_solve (Vpmg ∗thee)

    Solve the PBE using PMG.

- VPUBLIC void Vpmg_dtor (Vpmg ∗∗thee)

    Object destructor.

- VPUBLIC void Vpmg_dtor2 (Vpmg ∗thee)

    FORTRAN stub object destructor.

- VPUBLIC void Vpmg_setPart (Vpmg ∗thee, double lowerCorner[3], double upperCorner[3], int bflags[6])

    Set partition information which restricts the calculation of observables to a (rectangular) subset of the problem domain.

- VPUBLIC void Vpmg_unsetPart (Vpmg ∗thee)

    Remove partition restrictions.

- VPUBLIC int Vpmg_fillArray (Vpmg ∗thee, double ∗vec, Vdata_Type type, double parm, Vhal_PBEType pbetype)

    Fill the specified array with accessibility values.

- VPRIVATE double **Vpmg_polarizEnergy** (Vpmg ∗thee, int extFlag)
- VPUBLIC double Vpmg_energy (Vpmg ∗thee, int extFlag)

    Get the total electrostatic energy.

- VPUBLIC double Vpmg_dielEnergy (Vpmg ∗thee, int extFlag)

    Get the "polarization" contribution to the electrostatic energy.

- VPUBLIC double Vpmg_dielGradNorm (Vpmg ∗thee)

    Get the integral of the gradient of the dielectric function.

- VPUBLIC double Vpmg_qmEnergy (Vpmg ∗thee, int extFlag)

  Get the "mobile charge" contribution to the electrostatic energy.

- VPRIVATE double **Vpmg_qmEnergyNONLIN** (Vpmg ∗thee, int extFlag)
- VPUBLIC double **Vpmg_qmEnergySMPBE** (Vpmg ∗thee, int extFlag)
- VPUBLIC double Vpmg_qfEnergy (Vpmg ∗thee, int extFlag)

  Get the "fixed charge" contribution to the electrostatic energy.

- VPRIVATE double **Vpmg_qfEnergyPoint** (Vpmg ∗thee, int extFlag)
- VPUBLIC double Vpmg_qfAtomEnergy (Vpmg ∗thee, Vatom ∗atom)

  Get the per-atom "fixed charge" contribution to the electrostatic energy.

- VPRIVATE double **Vpmg_qfEnergyVolume** (Vpmg ∗thee, int extFlag)
- VPRIVATE void **Vpmg_splineSelect** (int srfm, Vacc ∗acc, double ∗gpos, double win, double infrad, Vatom ∗atom, double ∗force)
- VPRIVATE void **focusFillBound** (Vpmg ∗thee, Vpmg ∗pmgOLD)
- VPRIVATE void **extEnergy** (Vpmg ∗thee, Vpmg ∗pmgOLD, PBEparm_calcEnergy extFlag, double partMin[3], double partMax[3], int bflags[6])
- VPRIVATE double **bcfl1sp** (double size, double ∗apos, double charge, double xkappa, double pre1, double ∗pos)
- VPRIVATE void **bcfl1** (double size, double ∗apos, double charge, double xkappa, double pre1, double ∗gxcf, double ∗gycf, double ∗gzcf, double ∗xf, double ∗yf, double ∗zf, int nx, int ny, int nz)
- VPRIVATE void **bcfl2** (double size, double ∗apos, double charge, double ∗dipole, double ∗quad, double xkappa, double eps_p, double eps_w, double T, double ∗gxcf, double ∗gycf, double ∗gzcf, double ∗xf, double ∗yf, double ∗zf, int nx, int ny, int nz)
- VPRIVATE void **bcCalcOrig** (Vpmg ∗thee)
- VPRIVATE int **gridPointIsValid** (int i, int j, int k, int nx, int ny, int nz)
- VPRIVATE void **packAtoms** (double ∗ax, double ∗ay, double ∗az, double ∗charge, double ∗size, Vpmg ∗thee)
- VPRIVATE void **packUnpack** (int nx, int ny, int nz, int ngrid, double ∗gx, double ∗gy, double ∗gz, double ∗value, Vpmg ∗thee, int pack)
- VPRIVATE void **bcflnew** (Vpmg ∗thee)
- VPRIVATE void **multipolebc** (double r, double kappa, double eps_p, double eps_w, double rad, double tsr[3])
- VPRIVATE void **bcfl_sdh** (Vpmg ∗thee)
- VPRIVATE void **bcfl_mdh** (Vpmg ∗thee)
- VPRIVATE void **bcfl_mem** (double zmem, double L, double eps_m, double eps_w, double V, double xkappa, double ∗gxcf, double ∗gycf, double ∗gzcf, double ∗xf, double ∗yf, double ∗zf, int nx, int ny, int nz)
- VPRIVATE void **bcCalc** (Vpmg ∗thee)

- VPRIVATE void **fillcoCoefMap** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefMol** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefMolIon** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefMolDiel** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefMolDielNoSmooth** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefMolDielSmooth** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefSpline** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoef** (Vpmg ∗thee)
- VPRIVATE Vrc_Codes **fillcoCharge** (Vpmg ∗thee)
- VPRIVATE Vrc_Codes **fillcoChargeMap** (Vpmg ∗thee)
- VPRIVATE void **fillcoChargeSpline1** (Vpmg ∗thee)
- VPRIVATE double **bspline2** (double x)
- VPRIVATE double **dbspline2** (double x)
- VPRIVATE void **fillcoChargeSpline2** (Vpmg ∗thee)
- VPUBLIC int Vpmg_fillco (Vpmg ∗thee, Vsurf_Meth surfMeth, double splineWin, Vchrg_Meth chargeMeth, int useDielXMap, Vgrid ∗dielXMap, int useDielYMap, Vgrid ∗dielYMap, int useDielZMap, Vgrid ∗dielZMap, int useKappaMap, Vgrid ∗kappaMap, int useChargeMap, Vgrid ∗chargeMap)

  Fill the coefficient arrays prior to solving the equation.

- VPUBLIC int Vpmg_force (Vpmg ∗thee, double ∗force, int atomID, Vsurf_Meth srfm, Vchrg_Meth chgm)

  Calculate the total force on the specified atom in units of k_B T/AA.

- VPUBLIC int Vpmg_ibForce (Vpmg ∗thee, double ∗force, int atomID, Vsurf_Meth srfm)

  Calculate the osmotic pressure on the specified atom in units of k_B T/AA.

- VPUBLIC int Vpmg_dbForce (Vpmg ∗thee, double ∗dbForce, int atomID, Vsurf_Meth srfm)

  Calculate the dielectric boundary forces on the specified atom in units of k_B T/AA.

- VPUBLIC int Vpmg_qfForce (Vpmg ∗thee, double ∗force, int atomID, Vchrg_Meth chgm)

  Calculate the "charge-field" force on the specified atom in units of k_B T/AA.

- VPRIVATE void **qfForceSpline1** (Vpmg ∗thee, double ∗force, int atomID)
- VPRIVATE void **qfForceSpline2** (Vpmg ∗thee, double ∗force, int atomID)
- VPRIVATE void **qfForceSpline4** (Vpmg ∗thee, double ∗force, int atomID)
- VPRIVATE void **markFrac** (double rtot, double ∗tpos, int nx, int ny, int nz, double hx, double hy, double hzed, double xmin, double ymin, double zmin, double ∗xarray, double ∗yarray, double ∗zarray)

- VPRIVATE void **markSphere** (double rtot, double ∗tpos, int nx, int ny, int nz, double hx, double hy, double hz, double xmin, double ymin, double zmin, double ∗array, double markVal)
- VPRIVATE void **zlapSolve** (Vpmg ∗thee, double ∗∗solution, double ∗∗source, double ∗∗work1)
- VPUBLIC int Vpmg_solveLaplace (Vpmg ∗thee)

  Solve Poisson's equation with a homogeneous Laplacian operator using the solvent dielectric constant. This solution is performed by a sine wave decomposition.

- VPRIVATE double **VFCHI4** (int i, double f)
- VPRIVATE double **bspline4** (double x)
- VPUBLIC double **dbspline4** (double x)
- VPUBLIC double **d2bspline4** (double x)
- VPUBLIC double **d3bspline4** (double x)
- VPUBLIC void **fillcoPermanentMultipole** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefSpline4** (Vpmg ∗thee)
- VPUBLIC void **fillcoPermanentInduced** (Vpmg ∗thee)
- VPRIVATE void **fillcoCoefSpline3** (Vpmg ∗thee)

10.46.1 Detailed Description

Class Vpmg methods.

Author

Nathan Baker

Version

Id

vpmg.c 1555 2010-02-11 22:25:42Z sdg0919

Attention

```
*
* APBS -- Adaptive Poisson-Boltzmann Solver
*
* Nathan A. Baker (baker@biochem.wustl.edu)
* Dept. of Biochemistry and Molecular Biophysics
* Center for Computational Biology
* Washington University in St. Louis
*
* Additional contributing authors listed in the code documentation.
*
* Copyright (c) 2002-2010, Washington University in St. Louis.
```

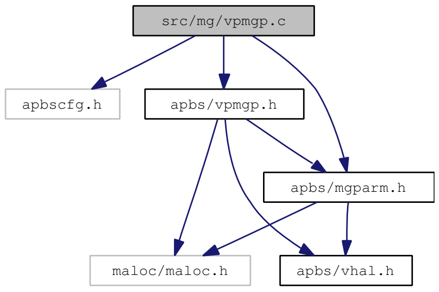
```
 * Portions Copyright (c) 2002-2010.  Nathan A. Baker
 * Portions Copyright (c) 1999-2002.  The Regents of the University of California.
 * Portions Copyright (c) 1995.  Michael Holst
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * -  Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * - Neither the name of Washington University in St. Louis nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
```

10.47 src/mg/vpmgp.c File Reference

Class Vpmgp methods.

```
#include "apbscfg.h"
#include "apbs/vpmgp.h"
#include "apbs/mgparm.h"
```

Include dependency graph for vpmgp.c:



Functions

- VPUBLIC Vpmgp ∗ Vpmgp_ctor (MGparm ∗mgparm)

  Construct PMG parameter object and initialize to default values.

- VPUBLIC int Vpmgp_ctor2 (Vpmgp ∗thee, MGparm ∗mgparm)

  FORTRAN stub to construct PMG parameter object and initialize to default values.

- VPUBLIC void Vpmgp_dtor (Vpmgp ∗∗thee)

  Object destructor.

- VPUBLIC void Vpmgp_dtor2 (Vpmgp ∗thee)

  FORTRAN stub for object destructor.

10.47.1 Detailed Description

Class Vpmgp methods.

Author

Nathan Baker

Version

Id

vpmgp.c 1552 2010-02-10 17:46:27Z yhuang01

Attention

Index