SWDTTI

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 LayerModel Class Reference

Public Member Functions

- void **initialize** ()

  initialize GLL/GRL nodes/weights
- void create_database (double freq, int nlayer, const float ∗vph, const float ∗vpv, const float ∗vsh, const float ∗vsv, const float ∗eta, const float ∗rho, const float ∗thk)

  initalize SEM mesh and create a VTI database from a layered model
- void prepare_matrices (int wavetype)

  prepare M/K/E matrices
- void compute_slegn (double freq, std::vector< double > &c, std::vector< double > &displ) const

  compute Love wave dispersion and eigenfunctions
- void compute_sregn (double freq, std::vector< double > &c, std::vector< double > &displ) const

  compute Love wave dispersion and eigenfunctions
- double compute_love_kl (double freq, double c, const double ∗displ, std::vector< double > &frekl) const

  compute group velocity and kernels for love wave
- double compute_rayl_kl (double freq, double c, const double ∗displ, std::vector< double > &frekl) const

  compute group velocity and kernels for love wave
- void transform_kernels (std::vector< double > &frekl) const

  transform kernels from base to vp/vs/eta/rho

Public Attributes

- int **nspec**
- int **nspec_grl**
- int **nglob**
- std::vector< int > **ibool**
- std::vector< float > **skel**
- std::vector< double > **znodes**
- std::vector< double > **jaco**
- std::vector< double > **z**
- std::vector< double > **xrho**
- std::vector< double > **xA**
- std::vector< double > **xC**
- std::vector< double > **xL**
- std::vector< double > **xF**
- std::vector< double > **xN**

3.1.1 Member Function Documentation

3.1.1.1 compute_love_kl()

```
double LayerModel::compute_love_kl (
            double freq,
            double c,
            const double * displ,
            std::vector< double > & frekl ) const
```

compute group velocity and kernels for love wave

Parameters

freq	current frequency
c	current phase velocity
displ	eigen function, shape(nglob)
frekl	Frechet kernels (N/L/rho) for elastic parameters, shape(3,nspec∗NGLL + NGRL)

Returns

double u group velocity

3.1.1.2 compute_rayl_kl()

```
double LayerModel::compute_rayl_kl (
            double freq,
            double c,
            const double * displ,
            std::vector< double > & frekl ) const
```

compute group velocity and kernels for love wave

Parameters

freq	current frequency
c	current phase velocity
displ	eigen function, shape(nglob ∗ 2)
frekl	Frechet kernels A/C/L/F/rho_kl kernels for elastic parameters, shape(5,nspec∗NGLL + NGRL)

Returns

double u group velocity

3.1.1.3 compute_slegn()

```
void LayerModel::compute_slegn (
            double freq,
```

```
            std::vector< double > & c,
            std::vector< double > & displ ) const
```

compute Love wave dispersion and eigenfunctions

Parameters

freq	current frequency
vmin,vmax	min/max velocity for your model
c	dispersion, shape(nc)
displ	eigen functions(displ at y direction), shape(nc,nglob)

3.1.1.4 compute_sregn()

```
void LayerModel::compute_sregn (
            double freq,
            std::vector< double > & c,
            std::vector< double > & displ ) const
```

compute Love wave dispersion and eigenfunctions

Parameters

freq	current frequency
vmin,vmax	min/max velocity for your model
c	dispersion, shape(nc)
displ	eigen functions(displ at x/z direction), shape(nc,2,nglob)

3.1.1.5 create_database()

```
void LayerModel::create_database (
            double freq,
            int nlayer,
            const float * vph,
            const float * vpv,
            const float * vsh,
            const float * vsv,
            const float * eta,
            const float * rho,
            const float * thk )
```

initalize SEM mesh and create a VTI database from a layered model

Parameters

freq	current frequency
nlayer	# of nlayers
vpv/vph/vsv/vsh/eta/rho	layer model vti parameters, shape(nlayer)

3.1.1.6 prepare_matrices()

```
void LayerModel::prepare_matrices (
            int wavetype )
```

prepare M/K/E matrices

Parameters

wavetype	= 1 for Love = 2 for Rayleigh

3.1.1.7 transform_kernels()

```
void LayerModel::transform_kernels (
            std::vector< double > & frekl ) const
```

transform kernels from base to vp/vs/eta/rho

Parameters

frekl	base Frechet kernels, shape(3/5,nspec∗NGLL+NGRL)

The documentation for this class was generated from the following files:

- swdlayer.hpp
- prepare_vti_matrices.cpp
- sem_vti.cpp
- swdlayer.cpp
- vti_kernels.cpp

3.2 LayerModelTTI Class Reference

Public Member Functions

- void **initialize** ()

  initialize GLL/GRL nodes/weights
- void create_database (double freq, int nlayer, const float ∗vph, const float ∗vpv, const float ∗vsh, const float ∗vsv, const float ∗eta, const float ∗theta0, const float ∗phi0, const float ∗rho, const float ∗thk)

  initalize SEM mesh and create a TTI database from a layered model
- void prepare_matrices (double phi)

  prepare M/K1/K2/E matrices for TTI model
- void compute_egnfun (double freq, double phi, std::vector< double > &c, std::vector< dcmplx > &displ) const

  compute phase velocity and eigen displacements for a given direction
- std::array< double, 2 > compute_kernels (double freq, double c, double phi, const std::vector< dcmplx > &displ, std::vector< double > &frekl) const

  compute group velocity and kernels for tti model
- void **transform_kernels** (std::vector< double > &frekl) const

Public Attributes

- int **nspec**
- int **nspec_grl**
- int **nglob**
- std::vector< int > **ibool**
- std::vector< float > **skel**
- std::vector< double > **znodes**
- std::vector< double > **jaco**
- std::vector< double > **z**
- std::vector< double > **xrho**
- std::vector< double > **xA**
- std::vector< double > **xC**
- std::vector< double > **xL**
- std::vector< double > **xF**
- std::vector< double > **xN**
- std::vector< double > **xT**
- std::vector< double > **xP**

3.2.1 Member Function Documentation

3.2.1.1 compute_egnfun()

```
void LayerModelTTI::compute_egnfun (
            double freq,
            double phi,
            std::vector< double > & c,
            std::vector< dcmplx > & displ ) const
```

compute phase velocity and eigen displacements for a given direction

Parameters

freq	current frequency
phi	current direction, in deg
c	phase velocity
displ	displacement

3.2.1.2 compute_kernels()

```
std::array< double, 2 > LayerModelTTI::compute_kernels (
            double freq,
            double c,
            double phi,
            const std::vector< dcmplx > & displ,
            std::vector< double > & frekl ) const
```

compute group velocity and kernels for tti model

Parameters

freq	current frequency
c	phase velocity at this frequency
phi	azimuthal angle of c
displ	eigen function, shape(nglob ∗ 3)
frekl	Frechet kernels A/C/F/L/N/T/P/rho_kl kernels for elastic parameters, shape(8,nspec∗NGLL + NGRL)

Returns

double u group velocity and it's azimthual angle

3.2.1.3 create_database()

```
void LayerModelTTI::create_database (
            double freq,
            int nlayer,
            const float * vph,
            const float * vpv,
            const float * vsh,
            const float * vsv,
            const float * eta,
            const float * theta0,
            const float * phi0,
            const float * rho,
            const float * thk )
```

initalize SEM mesh and create a TTI database from a layered model

Parameters

freq	current frequency
nlayer	# of nlayers
vpv/vph/vsv/vsh/eta/rho	layer model vti parameters, shape(nlayer)
theta0/phi0	axis direction

3.2.1.4 prepare_matrices()

```
void LayerModelTTI::prepare_matrices (
            double phi )
```

prepare M/K1/K2/E matrices for TTI model

Parameters

phi	polar angle of k vector, in deg

The documentation for this class was generated from the following files:

- swdlayertti.hpp
- prepare_tti_matrices.cpp
- sem_tti.cpp
- swdlayertti.cpp
- tti_kernels.cpp

Chapter 4

File Documentation

4.1 quadrature.hpp

```
00001 #ifndef SWD_QUADRATURE
00002 #define SWD_QUADRATURE
00003 #include <cmath>
00004
00005 //GLL
00006 void gauss_legendre_lobatto(double* knots, double* weights, int length);
00007 void lagrange_poly(double xi,int nctrl,const double *xctrl,
00008                 double *h,double*  hprime);
00009
00010 // GRL
00011 void gauss_radau_laguerre(double *xgrl,double *wgrl,size_t length);
00012 double laguerre_func(size_t n, double x);
00013
00014 #endif
```

4.2 swdlayer.hpp

```
00001 #ifndef SWD_LAYER_MODEL
00002 #define SWD_LAYER_MODEL
00003
00004 #include <complex>
00005 #include <vector>
00006 #include <array>
00007
00008 class LayerModel  {
00009
00010 typedef std::complex<double> dcmplx;
00011
00012 private:
00013     // GLL/GRL nodes and weights
00014     static const int NGLL = 7, NGRL = 20;
00015     std::array<double,NGLL> xgll,wgll;
00016     std::array<double,NGRL> xgrl,wgrl;
00017     std::array<double,NGLL*NGLL> hprimeT,hprime; // hprimeT(i,j) = l'_i(xi_j)
00018     std::array<double,NGRL*NGRL> hprimeT_grl,hprime_grl;
00019
00020     void initialize_nodes();
00021
00022 public:
00023     // SEM Mesh
00024     int nspec,nspec_grl; // # of elements for gll/grl layer
00025     int nglob; // # of unique points
00026     std::vector<int> ibool; // connectivity matrix, shape(nspec * NGLL + NGRL)
00027     std::vector<float> skel;  // skeleton, shape(nspec * 2 + 2)
00028     std::vector<double> znodes; // shape(nspec * NGLL + NGRL)
00029     std::vector<double> jaco; // jacobian for GLL, shape(nspec + 1) dz / dxi
00030     std::vector<double> z; // shape(nglob)
00031
00032     LayerModel(){};
00033     void initialize();
00034
00035 private:
00036     std::vector<int> ilayer_flag; // shape(nspec + 1), return layer flag
00037     std::vector<double> Mmat,Emat,Kmat; // matrices for SEM, om^2 M = k^2 K + E
```

```
00038     double PHASE_VELOC_MIN,PHASE_VELOC_MAX;
00039
00040 public:
00041
00042     // density
00043     std::vector<double> xrho;
00044
00045     // vti Love parameters
00046     std::vector<double> xA,xC,xL,xF,xN; // shape(nspec * NGLL + NGRL)
00047
00048
00049     // VTI model
00050     void create_database(double freq,int nlayer, const float *vph, const float* vpv,
00051                          const float *vsh, const float *vsv, const float *eta,
00052                          const float *rho,const float *thk);
00053
00054     void prepare_matrices(int wavetype);
00055
00056     void compute_slegn(double freq,std::vector<double> &c,
00057                           std::vector<double> &displ) const;
00058     void compute_sregn(double freq,std::vector<double> &c,
00059                          std::vector<double> &displ) const;
00060
00061     double compute_love_kl(double freq,double c,const double *displ, std::vector<double> &frekl)
      const;
00062     double compute_rayl_kl(double freq,double c,const double *displ, std::vector<double> &frekl)
      const;
00063
00064     void transform_kernels(std::vector<double> &frekl) const;
00065
00066 private:
00067     void prepare_matrices_love();
00068     void prepare_matrices_rayl();
00069 };
00070
00071
00072 #endif
```

4.3 swdlayertti.hpp

```
00001 #ifndef SWD_LAYER_TTI_MODEL
00002 #define SWD_LAYER_TTI_MODEL
00003
00004 #include <complex>
00005 #include <vector>
00006 #include <array>
00007
00008 class LayerModelTTI{
00009
00010 typedef std::complex<double> dcmplx;
00011
00012 private:
00013     // GLL/GRL nodes and weights
00014     static const int NGLL = 7, NGRL = 20;
00015     std::array<double,NGLL> xgll,wgll;
00016     std::array<double,NGRL> xgrl,wgrl;
00017     std::array<double,NGLL*NGLL> hprimeT,hprime; // hprimeT(i,j) = l'_i(xi_j)
00018     std::array<double,NGRL*NGRL> hprimeT_grl,hprime_grl;
00019
00020     void initialize_nodes();
00021
00022 public:
00023     // SEM Mesh
00024     int nspec,nspec_grl; // # of elements for gll/grl layer
00025     int nglob; // # of unique points
00026     std::vector<int> ibool; // connectivity matrix, shape(nspec * NGLL + NGRL)
00027     std::vector<float> skel;  // skeleton, shape(nspec * 2 + 2)
00028     std::vector<double> znodes; // shape(nspec * NGLL + NGRL)
00029     std::vector<double> jaco; // jacobian for GLL, shape(nspec + 1) dz / dxi
00030     std::vector<double> z; // shape(nglob)
00031
00032     LayerModelTTI(){};
00033     void initialize();
00034
00035 private:
00036     std::vector<int> ilayer_flag; // shape(nspec + 1), return layer flag
00037     std::vector<dcmplx> Mmat,Emat,K1mat,K2mat; // matrices for SEM,shape(3*nglob,3*nglob) om^2 M = k^2
      K_2 + k K_1 + E
00038     double PHASE_VELOC_MIN,PHASE_VELOC_MAX;
00039
00040 public:
00041
00042     // density
```

```
00043     std::vector<double> xrho;
00044
00045     // tti Love parameters A,C,L,F,N, theta,phi
00046     std::vector<double> xA,xC,xL,xF,xN; // shape(nspec * NGLL + NGRL)
00047     std::vector<double> xT,xP; // theta/phi, shape(nspec *NGLL + NGRL), in rad
00048
00049
00050     // VTI model
00051     void create_database(double freq,int nlayer, const float *vph, const float* vpv,
00052                          const float *vsh, const float *vsv, const float *eta,
00053                          const float *theta0, const float *phi0,
00054                          const float *rho,const float *thk);
00055
00056     void prepare_matrices(double phi);
00057
00058     void compute_egnfun(double freq, double phi, std::vector<double> &c, std::vector<dcmplx> &displ)
      const;
00059     std::array<double,2>
00060     compute_kernels(double freq, double c,double phi,
00061                     const std::vector<dcmplx> &displ,
00062                     std::vector<double> &frekl) const;
00063
00064     void transform_kernels(std::vector<double> &frekl) const;
00065 };
00066
00067 #endif
```

Index