ETRAD

1.0

Generado por Doxygen 1.5.9

Mon Oct 5 21:19:53 2009

Índice general

Capítulo 1

Índice de clases

1.1. Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Capítulo 2

Índice de clases

2.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 3

Documentación de las clases

3.1. Referencia de la Clase Archivo

Clase para manipular archivos.

`#include <Archivo.hpp>`

Métodos públicos

- Archivo ()

  Contructor de la Clase.

- ~Archivo ()

  Destructor de la Clase.

- void Escribe_Vector (Vector *a, unsigned long long nn)

  Escribe un Vector a un archivo (Recordar cambiar a Escribe_Vector).

- void Escribe_MatrizCuadrada (Matriz *a, unsigned long long nodos)

  Escribe una matriz cuadrada a un archivo.

- void Escribe_Matriz (Matriz *a)

  Escribe una matriz a un archivo.

- void SetNombre (char *nombre)

  Asigna un nombre al archivo.

- void SetModo (int modo)

Define el tipo de acceso al archivo.

- char * GetNombre ()

  Retrona Nombre del archivo.

- FILE * GetArchivo ()

  Retorna el archivo.

- unsigned long long Numero_Lineas ()

  Retorna el numero de lineas del archivo.

- void Abrir ()

  Abre el archivo.

- void Cerrar ()

  Cierra el archivo.

- void VerNombre ()

  Muestra el nombre del archivo.

3.1.1. Descripción detallada

Clase para manipular archivos.

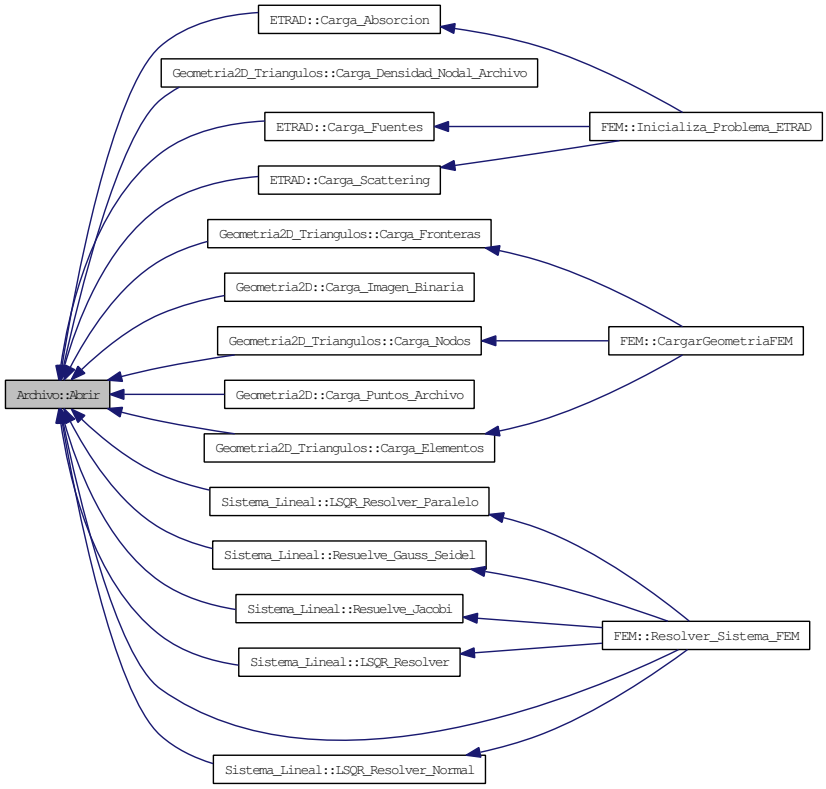
3.1.2. Documentación de las funciones miembro

3.1.2.1. void Archivo::Abrir ()

Abre el archivo.

Abre archivo.

Gráfico de llamadas a esta función:

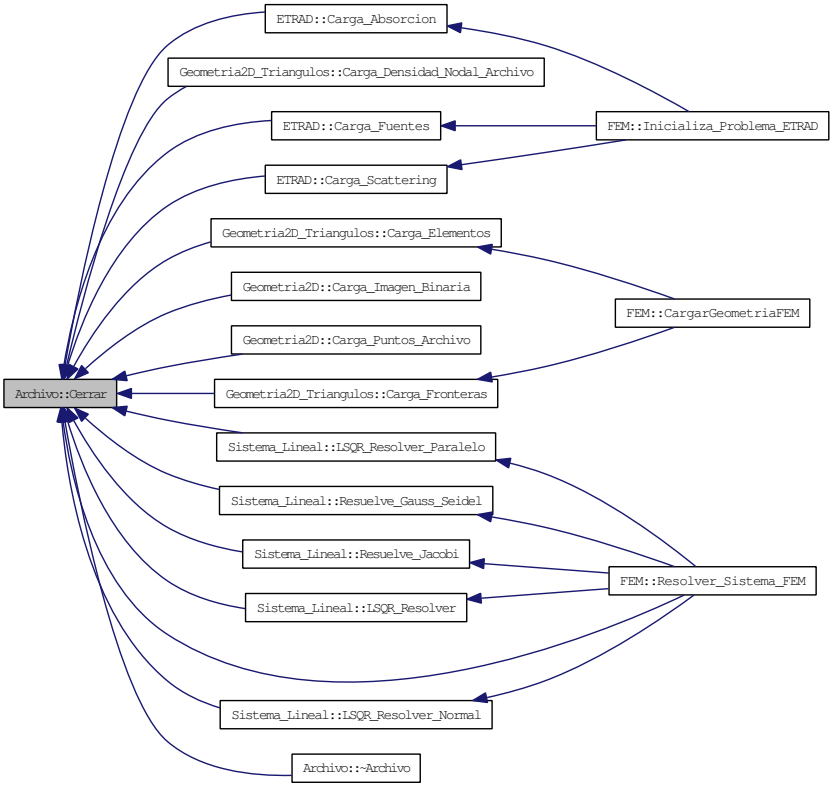


3.1.2.2. void Archivo::Cerrar (void)

Cierra el archivo.

Cierra archivo.

Gráfico de llamadas a esta función:

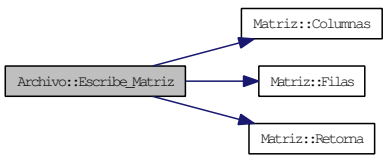


3.1.2.3. void Archivo::Escribe_Matriz (Matriz ∗ *a*)

Escribe una matriz a un archivo.

Escribe una matriz cuadrada a un archivo.

Gráfico de llamadas para esta función:



3.1.2.4. void Archivo::Escribe_Vector (Vector * *a*, unsigned long long *nn*)

Escribe un Vector a un archivo (Recordar cambiar a Escribe_Vector).

Escribe un Vector a un archivo.

Gráfico de llamadas para esta función:

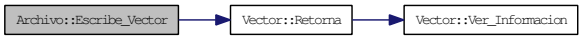
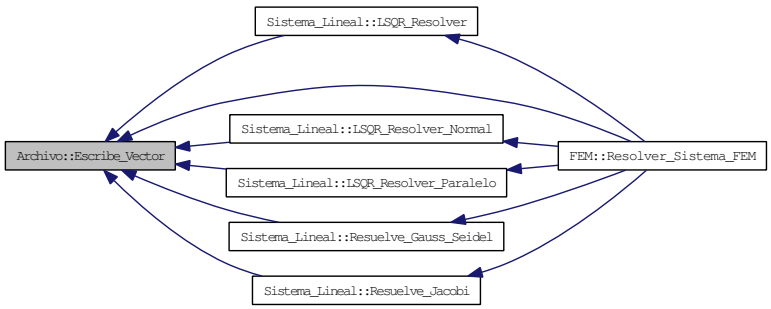


Gráfico de llamadas a esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Archivo.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Archivo.cpp

3.2. Referencia de la Clase Elemento

Clase que define un elemento finito, donde se indica el tipo de elemento y el indice del elemento.

`#include <Elemento.hpp>`

Heredado por Elemento_Triangular.

Métodos públicos

- Elemento ()
  Constructor de la clase.
- ~Elemento ()
  Destructor de la clase.
- void Set_Id_Elemento (unsigned long long id)
  Asigna Id del elemento.
- void Set_Id_Tipo (int tipo)
  Asigna el tipo de elemento.
- unsigned long long Get_Id_Elemento ()
  Retorna el id del elemento.
- int Get_Tipo_Elemento ()
  Retorna el tipo de elemento.

Atributos protegidos

- unsigned long long id_elemento
  Indice del Elemento.
- int tipo_elemento
  1 - Lineal, 2 - Triangular, 3 - Cuadrilatero, 4 - Tetrahedro

3.2.1. Descripción detallada

Clase que define un elemento finito, donde se indica el tipo de elemento y el indice del elemento.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Elemento.hpp

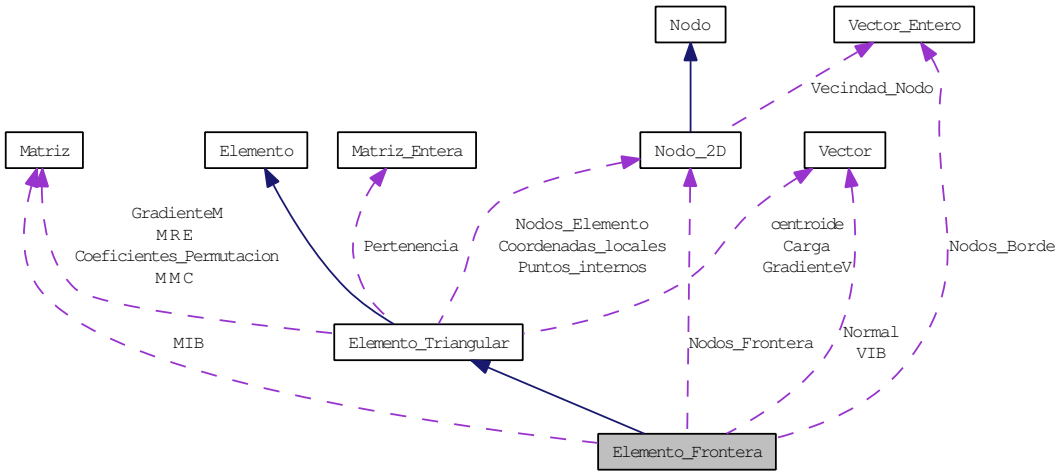
3.3. Referencia de la Clase Elemento_Frontera

Clase que define los elementos que estan en la frontera del dominio, util para un futuro calculo de matrices elementales frontera.

`#include <Elemento_Frontera.hpp>`

Herencias Elemento_Triangular.

Diagrama de colaboración para Elemento_Frontera:



Métodos públicos

- Elemento_Frontera ()

  Constructor de la clase.

- ~Elemento_Frontera ()

  Destructor de la clase (RECORDAR LIBERAR RAM).

- Matriz * Get_MIB ()

  Retorna la matriz borde.

- Vector * Get_VIB ()

  Retorna la vector borde.

- Vector_Entero * Get_pertenencia_Borde ()

  Retorna la matriz pertenencia borde.

- void Set_Vector_Pertenencia_Borde (int col, int valor)

  Asigna un valor al vector de pertenencia en la posicion col.

- int Get_Valor_Nodo_Frontera (int nNodo)

  Retorna el indice global de un nodo frontera.

- Nodo_2D * Get_Nodo_Frontera ()

  Retorna el arreglo Nodo_2D[] que contiene informacion de los nodos de la frontera.

- void Set_Reserva_Memoria_Matriz_Borde (int fil, int col)

  Reserva memoria para las matrices asociadas al borde.

- void Set_Reserva_Memoria_Vector_Borde (int col)

  Reserva memoria para las matrices asociadas al borde.

- void Set_Reserva_Memoria_MIB ()

  Reserva memoria a una matriz 2x2 que contendra informacion del elemento frontera (Variante de alternativa directa).

- void Set_Matriz_Borde (int fil, int col, double valor)

  Asigna un valor a la matriz de borde en fila - columna.

- void Set_Vector_Borde (int col, double valor)

  Asigna un valor al vector borde en columna.

- void Set_Reserva_Memoria_Nodos_Frontera (int NNE)

  Reserva memoria para alamacenar informacion de los nodos frontera.

- void Set_Nodo_Frontera (int Posicion_Nodal, int nodo_global)

  Asigna el indice global a un nodo frontera.

- void Set_Dominio_Frontera (int Dom)

  Asigna el dominio al cual pertenece el nodo.

- int Get_Dominio_Frontera ()

  Retorna el dominio al cual pertenece el nodo frontera.

- void Set_Norma (double nNorma)

  Asigna la norma al elemento frontera.

- double Get_Norma ()

  Asigna la norma al elemento frontera.

- void Set_Vector_Normal (Vector vNormal)

  Reserva memoria y asigna el vector normal a un elemento frontera.

- double Calculo_MBorde_ML (double qi, double nNorma, int peso)

  Calculo de la matriz asociada al borde.

- double Calculo_VBorde_ML (double qi, double nNorma)

  Calculo del Vector asociado al borde.

Atributos protegidos

- Nodo_2D * Nodos_Frontera

  Arreglo de objetos Nodo_2D[ ] que contendran informacion sobre la frontera.

- Vector_Entero * Nodos_Borde

  Indices globales de los nodos de la frontera.

- Matriz * MIB

  Matriz de borde Integral.

- Vector * VIB

  Vector de borde Integral.

- int dominio_frontera

  Dominio de la frontera.

- double R

  Factor experimental de Fressnel (No se usa, pero se deja para una futura implementacion).

- double A_Factor

  Factor asociado a la difusion (No se usa, pero se deja para una futura implementacion).

- double Yn

  Factor asociado a la dispersion (No se usa, pero se deja para una futura implementacion).

- double kScatering

  Constante asociada al scattering.

- double Norma

  Norma - Distancia, usada para el calculo integral entre dos puntos.

- Vector ∗ Normal

  Vector normal.

3.3.1. Descripción detallada

Clase que define los elementos que estan en la frontera del dominio, util para un futuro calculo de matrices elementales frontera.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Elemento_Frontera.hpp

3.4. Referencia de la Clase Elemento_Triangular

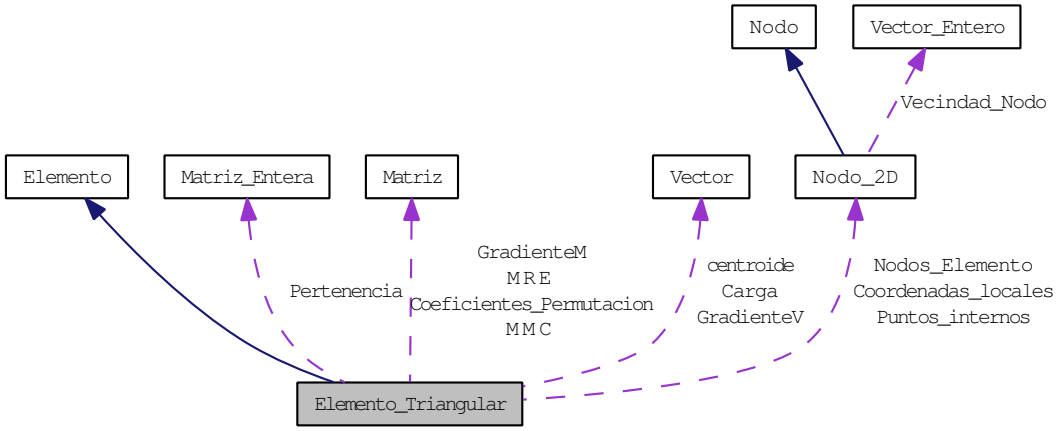
Clase que define un elemento finito triangular lineal de 3 nodos.

`#include <Elemento_Triangular.hpp>`

Herencias Elemento.

Heredado por Elemento_Frontera.

Diagrama de colaboración para Elemento_Triangular:



Métodos públicos

- Elemento_Triangular ()

  Constructor de la CLase.

- ~Elemento_Triangular ()

  Destructor de la Clase.

- void Set_Numero_Nodos_Elemento (int NNE)

  Asigna el numero de nodos del elemento.

- void Set_Reserva_Memoria_Nodos_Triangulo (int NNE)

  Reserva memoria y define un arreglo de objetos Nodo_2D[] que pertenecen al elemento.

- void Set_Reserva_Memoria_Coordenadas_locales (int NNE)

  Reserva memoria y define un arreglo de objetos Nodo_2D[] que pertenecen al elemento en caso de utilizar coordenadas locales.

- void Set_Reserva_Memoria_Matriz_Rigidez_Elemento ()

  Reserva memoria para la matriz de Rigidez del elemento.

- void Set_Reserva_Memoria_Matriz_Masa_Elemento ()

  Reserva memoria para la matriz de masa del elemento.

- void Set_Reserva_Memoria_Matriz_Pertenencia_Elemento ()

  Reserva memoria para la matriz de pertenencia del elemento.

- void **Set_Nodo_Elemento** (int Posicion_Nodal, int nodo_global)
- void Set_Constante_Fuente (double fi)

  Asigna constante Fuente, solo se utiliza para problemas con distribucion de energia constante en todo el dominio.

- void Set_Centroide (double x, double y)

  Reserva memoria para el centroide y asigna el valor del centro geometrico del elemento.

- void Set_Vector_Gradiente (double Bi, double Ci)

  Reserva memoria y asigna el vector gradiente del elemento.

- void Set_Reserva_Memoria_Vector_Carga ()

  Reserva memoria para vector de carga del elemento.

- void Set_Vector_Carga (double fi, double fj, double fk)

  Asigna los valores al vector de carga del elemento.

- void Set_Reserva_Memoria_Permutacion ()

  Reserva memoria para ma matriz de permutacion del elemento.

- void Set_Permutacion (int fil, int col, double valor)

  Asigna un valor a la matriz de permutacion del elemento.

- void Set_Matriz_Rigidez_Elemento (int fil, int col, double valor)

  Asigna un valor a la matriz de rigidez en la posicion fil, col.

- void Set_Matriz_Masa_Elemento (int fil, int col, double valor)

  Asigna un valor a la matriz de masa en la posicion fil, col.

- void Set_Matriz_Pertenencia_Elemento (int fil, int col, int valor)

  Asigna un valor a la matriz de pertenencia en la posicion fil, col.

- void Set_Matriz_Gradiente (double Bi, double Ci)

  Reserva memoria y asigna los valores de la matriz gradiente.

- void Set_Area (double area_elemento)

  Asigna el area del elemento.

- void Set_Dominio_Elemento (int Dom)

  Asigna el dominio al cual pertenece el elemento.

- void Set_Elemento_Fuente (int asignacion)

  Asigna un 1 si es un elemento que contiene a una fuente.

- void Set_kVariable_Elemento (double Valor)

  Asigna constante kVariable del elemento, generamente asociada al comportamiento fisico-geometrico del elemento.

- void Set_sVariable_Elemento (double Valor)

  Asigna constante sVariable del elemento asociada al comportamiento fisico del elemento.

- void Set_Densidad_Elemento (double Valor)

  Asigna la densisad del elemento.

- void Set_mVariable_Elemento (double Valor)

  Asigna constante mVariable del elemento, generamente asociada al comportamiento fisico-masa del elemento.

- void Set_Coordenada_Local_Elemento (int idNodo, double xCord, double yCord)

  Asigna el valor de una coordenada local del elemento.

- unsigned long long Get_Valor_Nodo (int Posicion_Nodal)

  Retorna la numeracion nodal global del nodo local "Posicion_Nodal".

- int Get_Dominio ()

  Retorna el dominio del elemento.

- double Get_Area ()

  retorna el area del elemento

- double Get_f ()

  Retorna el valor de la fuente del elemento.

- double Get_kVariable ()

  Retorna la kVariable del elemento.

- double Get_mVariable ()

  Retorna la mVariable del elemento.

- double Get_sVariable ()

  Retorna la sVariable del elemento.

- Nodo_2D Get_Coordenada_Local (int id)

  Retorna el objeto Nodo_2D Coordenadas_locales[id] que tiene informacion sobre las coordenadas locales del elemento.

- Nodo_2D Get_Nodos_Elemento (int id)

  Retorna el objeto Nodo_2D Nodos_Elemento[id], que tiene informacion sobre los nodos del elemento.

- Matriz * Get_Coeficientes_Permutacion ()

  Retorna la matriz de permutacion del elemento.

- Matriz * Get_Matriz_Rigidez_Elemento ()

  Retorna la matriz de Rigidez del elemento.

- Matriz * Get_Matriz_Masa_Elemento ()

  Retorna la matriz de masa del elemento.

- Matriz_Entera * Get_Matriz_Pertenecia_Elemento ()

  Retorna la matriz de pertenencia del elemento.

- Vector * Get_Centroide ()

  Retorna el vector que contiene el centro geometrico del elemento.

- Vector * Get_GradienteV ()

  Retorna el vector que contiene el gradiente del elemento.

- Vector * Get_CargasV ()

  Retorna el vector de cargas del elemento.

- Matriz * Get_GradienteM ()

  Retorna la matriz gradiente del elemento.

- int Get_Numero_Nodos_Elemento ()

  Retorna el numero de nodos del elemento.

Atributos protegidos

- Nodo_2D * Nodos_Elemento

  Nodos locales del elemento, se utiliza como un vector de objetos Nodo_2D[].

- Nodo_2D * Coordenadas_locales

  Nodos almacenados como coordenadas locales (Solo se utiliza si se quieren utilizar formulacion Isoparametrica o bien en coordenadas locales).

- Nodo_2D * Puntos_internos

  Puntos internos en un elemento (REVISAR POR QUE LO DEFINI).

- int numero_nodos_elemento

  Numero de nodos del elemento.

- int dominio

  Dominio al cual pertenece el elemento.

- int eFuente

  Indica si el elemento contiene a una fuente.

- double eDensidad

  Densidad aproximada del elemento.

- double area

  Area del elemento.

- double f

  Aporte que hace el elemento a la fuente.

- double kVariable

  Variable fisica 1 del elemento asociada al comportamiento fisico-geometrico.

- double mVariable

  Variable fisica 2 del elemento asociado a la fisico-masa.

- double sVariable

  Variable secundaria de dispersion del elemento.

- Vector * Carga

  Vector de Cargas del elemento.

- Vector * GradienteV

Vector Gradiente de un nodo del Elemento.

- Vector * centroide

  Centro geometrico del elemento.

- Matriz * GradienteM

  Matriz Gradientes del elemento.

- Matriz * Coeficientes_Permutacion

  Coeficientes ai,bi,ci de la funcion de forma del elemento.

- Matriz * MRE

  Matriz Rigidez Elemento - Stiffness.

- Matriz * MMC

  Matriz de Masa Elemento.

- Matriz_Entera * Pertenencia

  Matriz de pertenencia global (numeracion nodal global) del Elemento.

3.4.1. Descripción detallada

Clase que define un elemento finito triangular lineal de 3 nodos.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Elemento_-Triangular.hpp

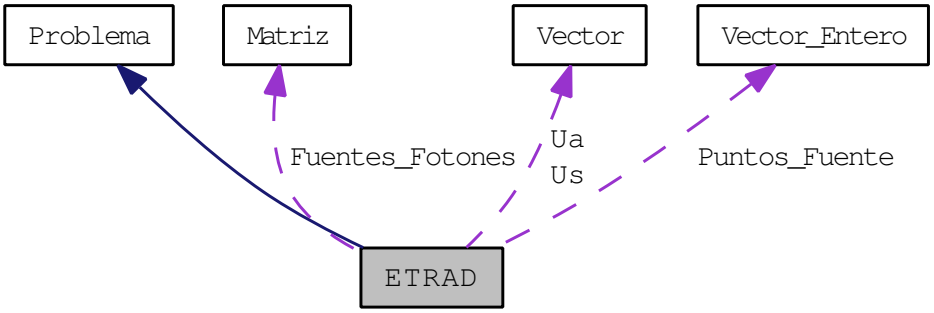
3.5. Referencia de la Clase ETRAD

Clase que define las caracteristicas del problema de difusion de la luz en medios difusos.

`#include <ETRAD.hpp>`

Herencias Problema.

Diagrama de colaboración para ETRAD:



Métodos públicos

- ETRAD (void)

  Construtor de la clase.

- ~ETRAD ()

  Destructor de la clase.

- double Coeficiente_difusion (double UaValor, double UsValor, double fase)

  Retorna el coeficiente de difusion de la ecuacion de difusion de la luz, dado absorcion, scattering y funcion de fase.

- double Coseno_Promedio_funcion_Fase2D (double g, double pScatter)

  Retorna el coseno promedio dado la funcion de fase y el scattering, segun propuesta de la Tesis de Tarvainen 2006 (REVISAR IMPLEMENTACION, NO SE ESTA USANDO, POSIBLE ELIMINACION).

- void Asigna_Funcion_Fase (double fase)

  Asigna el valos de la funcion de fase en la difusion.

- void Carga_Absorcion (char *arch)

  Carga del archivo que contiene los datos de absorcion en los distintos subdominios.

- void Carga_Scattering (char *arch)

  Carga el archivo que contiene los datos de scattering en los distintos subdominios.

- void Carga_Fuentes (char *arch)

  Carga el archivo que contiene los datos de las fuentes.

- void SetNombreArchivoAbsorcion (char *nombre)

  Asigna el nombre del archivo aociado a la absorcion.

- void SetNombreArchivoScattering (char *nombre)

  Asigna el nombre del archivo asociado al scattering.

- void SetNombreArchivoFuentes (char *nombre)

  Asigna el nombre del archivo asociado a las fuentes de fotones.

- char * GetNombreArchivoAbsorcion ()

  Retorna el nombre del archivo asociado a la absorcion.

- char * GetNombreArchivoScattering ()

  Retorna el nombre del archivo asociado al scattering.

- char * GetNombreArchivoFuentes ()

  Retorna el nombre del archivo asociado a las fuentes.

- void Set_Longitud_Vector_Ua (int longitud)

  Reserva memoria para el vector que contendra los datos de absorcion.

- void Set_Longitud_Vector_Us (int longitud)

  Reserva memoria para el vector que contendra los datos de scattering.

- void Set_Valor_Ua (int indice, double valor)

  Asigna el valor de absorcion al vector Ua en la posicion "indice".

- void Set_Valor_Us (int indice, double valor)

  Asigna el valor de scattering al vector Us en la posicion "indice".

- Vector * Get_Vector_Ua ()

  Retorna el vector que contiene los datos de absorcion.

- Vector * Get_Vector_Us ()

  Retorna el vector que contiene los datos de scattering.

- Matriz * Get_Fuentes ()

  Retorna el vector que contiene los datos de scattering.

- unsigned long long Get_Suma_Fuentes ()

  Retorna la suma del numero de fotones provenientes de las fuentes en todo el dominio.

- double Get_Valor_Ua (int id)

  Retorna el valor de absorcion del vector Ua en la posicion "id".

- double Get_Valor_Us (int id)

  Retorna el valor de absorcion del vector Us en la posicion "id".

- double Get_Funcion_Fase ()

  Retorna el valor de la funcion de fase.

Atributos públicos

- unsigned long long **Suma_Fotones_Fuentes**

Atributos protegidos

- Vector * Ua

  Vector que contiene los coeficientes de absorcion en cada subdominio del problema.

- Vector * Us

  Vector que contiene los coeficientes de Dispersion - Scattering en cada subdominio del problema.

- Vector_Entero * Puntos_Fuente

  Puntos fuente (REVISAR IMPLEMENTACION Y USO).

- Matriz * Fuentes_Fotones

  Fuentes y densidades respectivas.

- double cLuz

  Velocidad de la luz en el medio.

- double fFase

  Funcion de fase.

- int tipo_aproximacion

  Tipo de aproximacion del problema, 1-Eliptico (sin calculo temporal asociado) 2-Hiperbolico (con calculo temporal asociado).

- int borde_interno

  Valor 1 si se usa condicion de borde en los subdominios interiores, 0 en caso contrario (REVISAR IMPLEMENTACION).

- int borde_externo

  Valor 1 si se usa condicion de borde distinta de 0 en el borde del dominio (REVISAR IMPLEMENTACION).

- int tiempo

  Numero de epocas en caso de utilizar funcion Hiperbolica (REVISAR IMPLEMENTACION).

- char * NombreArchivoAbsorcion

  Nombre del archivo que contiene los datos de absorsion.

- char * NombreArchivoScattering

  Nombre del archivo que contiene los datos de Scattering.

- char * NombreArchivoFuentes

  Nombre del archivo que contiene los datos de las fuentes de fotones.

3.5.1. Descripción detallada

Clase que define las caracteristicas del problema de difusion de la luz en medios difusos.

3.5.2. Documentación de las funciones miembro

3.5.2.1. double ETRAD::Coseno_Promedio_funcion_Fase2D (double *g*, double *pScatter*)

Retorna el coseno promedio dado la funcion de fase y el scattering, segun propuesta de la Tesis de Tarvainen 2006 (REVISAR IMPLEMENTACION, NO SE ESTA USANDO, POSIBLE ELIMINACION).

Retorna el coseno promedio dado la funcion de fase y el scattering, segun propuesta de la Tesis de Tarvainen 2006.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/ETRAD.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/ETRAD.cpp

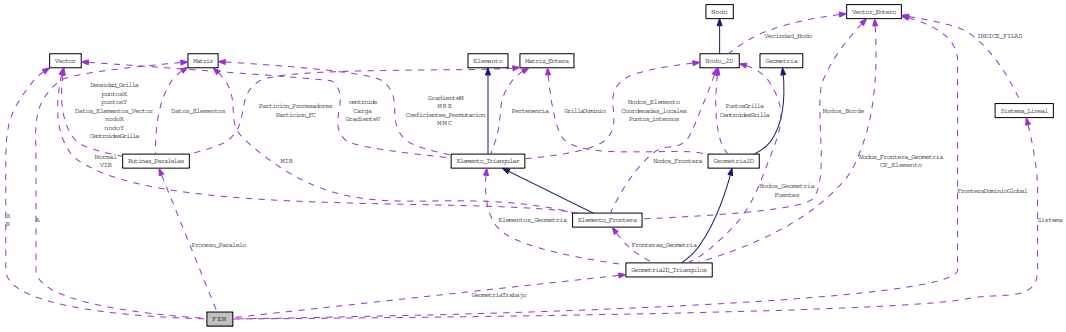
3.6. Referencia de la Clase FEM

Clase que define un procedimiento general del metodo del elemento finito - FEM (Siglas en ingles).

`#include <FEM.hpp>`

Heredado por FEM2D.

Diagrama de colaboración para FEM:



Métodos públicos

- FEM ()

  Constructor de la clase.

- ~FEM ()

  Destructor de la clase.

- void Hashing_FEM (unsigned long long indice, double aporte, unsigned long long i, unsigned long long j)

  Almacena y suma los aportea a una Map que utiliza el Hashing como referencia para cada aporte nodal a la matriz A - se almacenan solo los datos no nulos.

- void Mostrar_Hashing_FEM ()

  Muetra la matriz de tipo Map donde se utiliza un hash como estructura de almacenamiento de la matriz A.

- void Mostrar_Tiempo_Comunicacion ()

  Muestra el tiempo total utilizado en comunicacion.

- void Mostrar_Tiempo_Global ()

  Muestra el tiempo global utilizado en el programa.

- double Retornan_Tiempo_Comunicacion ()

  Retorna el tiempo usado en comunicacion.

- double Retornar_Tiempo_Global ()

  Retorna el tiempo total del programa.

- void SetProcesadores (int Procesadores)

  Alamcena el numero de procesadores.

- int GetProcesadores ()

  Retorna el numero de procesos.

- Geometria2D_Triangulos * Get_Geometria2D_Triangulos ()

  Retorna la geometria de trabajo.

- void SetNombreArchivoX (char *nombre)

  Asigna el nombre al archivo solucion.

- void SetNombreArchivoTiempos (char *nombre)

  Asigna el nombre al archivo tiempos.

- char * GetNombreArchivoTiempos ()

  Retorna el nombre del archivo tiempos.

- char * GetNombreArchivoX ()

  Retorna el nombre del archivo solucion.

- char * GetNombreArchivoGrilla ()

  Retorna el nombre del archivo solucion.

- void SetNombreArchivoNodos (char *nombre)

  Asigna el nombre del archivo grilla.

- void SetNombreArchivoGrilla (char *nombre)

  Asigna el nombre del archivo nodos.

- void SetNombreArchivoElementos (char *nombre)

  Asigna el nombre del archivo elementos.

- void SetNombreArchivoFronteras (char *nombre)

  Asigna el nombre del archivo fronteras.

- char * GetNombreArchivoNodos ()

  Retorna el nombre del archivo nodos.

- char * GetNombreArchivoElementos ()

  Retorna el nombre del archivo elementos.

- char * GetNombreArchivoFronteras ()

  Retorna el nombre del archivo fronteras.

- void SetGeometria (int TipoGeometria)

  Define el tipo de geometria a utilizar.

- void SetTipoCalculo (int TipoCalculo)

  Define el tipo de calculo a utilizar.

- int GetGeometria ()

  Retorna el tipo de geometria utilizado.

- int Get_tipo_solucion_matricial ()

  Retorna el tipo de de solucion matricial 1 - Matriz Completa, 2 - Matriz reducida.

- double Get_sCarga ()

  Retorna la sumas de magnitudes original del vector de cargas.

- void Set_tipo_solucion_matricial (int tipoM)

  Asigna el tipo de solucion matricial.

- unsigned long Get_orden_nodal ()

  Retorna el orden nodal global.

- void Set_orden_nodal (int oGlobal)

  Asigna el orden nodal global.

- void Set_sCarga (double sC)

  Asigna el suma de magnitudes original en vector de cargas.

- void SetNumeroProcesadores (int NumeroProcesadores)

  Define el numero de procesadores a utilizar en el calculo paralelo.

- void Densidad_FEM (void)

Calcula la dendidad global de la solucion FEM, es una suma de densidades nodales por sus funciones de forma nodal (Se utiliza solo como funcion de traza del programa).

- void VerTipoGeometria (void)

  Muestra el tipo de geometria que se esta utilizando.

- void CargarGeometriaFEM ()

  Carga la geometria y calcula las areas de los elementos y matrices de permutacion elemental, que luego son utilizados al momento de generar las matrices de rigidez y de masa.

- void Inicializa_Problema_ETRAD (ETRAD *FisicaProblema, int tipoFase)

  Inicializa problema ETRAD (rECORDAR CAMBIAR IMPLEMNETACION Y DEJARLA EN UNA CLASE SEPARADA, SOLO SE DEJO ACA POR COMODIDAD DEPROGRMACION Y TRAZA).

- void Contruir_Matrices_Problema ()

  Calcula las matrices de rigidez, de masa y vector de carga de los elementos,.

- void Resolver_Sistema_FEM (int metodo, int traza, unsigned long long iteraciones, int tipo_parada)

  Resolucion del sistema de ecuaciones.

- void Establecer_Condicion_Matricial_FEM (int condicion, int metodo)

  Establece el tipo de tratamiento numerico con el cula se trabaja en la resolucion del sistema lineal (Matriz completa o reducida).

- void Ingreso_Datos_Grilla_FEM (Vector *dominio, Vector_Entero *Resolucion, int nProcesos)

  Adquisicion de datos de la grilla a calcular.

- void Calculo_Grilla_FEM ()

  Aproximacion nodal a partir de una grilla de puntos.

- void Carga_Datos_Elementos_Grilla (void)

  Carga datos de calculos elementales a una matriz global para futuro calculo paralelo.

Atributos públicos

- char * NombreArchivoGrilla

  Nombre archivo nodos.

- char * NombreArchivoNodos

  Nombre archivo nodos.

- char * NombreArchivoElementos

  Nombre archivo elementos.

- char * NombreArchivoFronteras

  Nombre archivo fronteras.

- char * NombreArchivoX

  Nombre archivo solucion.

- char * NombreArchivoTiempos

  Nombre archivo tiempos.

- int TIPO_GEOMETRIA

  Tipo de geometria a utilizar.

- int TIPO_CALCULO

  Tipo calculo secuencial o paralelo.

- int NUMERO_PROCESADORES

  Numero de procesadores en caso de utilizar programacion paralela (FALTA EVALUAR IMPLEMENTACION).

- int tipo_solucion_matricial

  Tipo de solucion matricial a utilizar 1-Matriz completa, 2-Matriz reducida.

- unsigned long orden_nodal

  Orden del sistema de ecuaciones.

- unsigned long orden_dirichlet

  Orden del sistema reducido - Condicion borde 0.

- int tipo_vector_cargas

  Tipo vector de cargas, 1-distribucion uniforme 2- Distribucion de Dirac.

- int FronteraExterior

  Indica cual es el dominio que contiene a la frontera global.

- double tTotalComunicacion

Tiempo total utilizado en comunicacion.

- double tTotalGlobal

  Tiempo total de ejecucion global.

- double Tiempo_Inicial_Comunicacion_Grilla

  Tiempos parciales comunicacion grilla.

- double **Tiempo_Parcial_Comunicacion_Grilla**
- double **tTotalComunicacionGrilla**
- double Tiempo_Inicial_Comunicacion_FEM

  Tiempos parciales comunicacion FEM.

- double **Tiempo_Parcial_Comunicacion_FEM**
- double **tTotalComunicacionFEM**
- map< unsigned long long, double > Matriz_A_FEM
- map< unsigned long long, unsigned long long > Mapeo_FEM_Fila

  Mapeo de la informacion de las filas con elementos no nulos.

- map< unsigned long long, unsigned long long > Mapeo_FEM_Columna

  Mapeo de la informacion de las columnas con elementos no nulos.

- map< unsigned long long, unsigned long long > Mapeo_FEM_Dirichlet

  Mapeo de los indice para aproximacion de borde tipo Dirichlet.

- double sCarga

  Suma vector de cargas en magnitud de entrada.

Atributos protegidos

- Matriz * A

  Matriz de riguidez completa del sistema (Matriz stiffness (Fisico-geometria) + Matriz Masa (fisico-masa)).

- Vector * B

  Vector que contendra los valores conocidos del vector de carga global (Vector Fuente).

- Vector * X

  Vector incognita del problema, son los valores desconocidos del problema.

- Vector_Entero * FronteraDominioGlobal

  Vector Fronteras.

- Sistema_Lineal * Sistema

  Declaracion del sistema lineal a resolver.

- int nProcesadores

  Numero de procesadores a utilizar.

- Rutinas_Paralelas * Proceso_Paralelo

  Puntero de las rutinas paralelas.

- Geometria2D_Triangulos * GeometriaTrabajo

  Puntero a Geometria de trabajo a utilizar.

3.6.1. Descripción detallada

Clase que define un procedimiento general del metodo del elemento finito - FEM (Siglas en ingles).

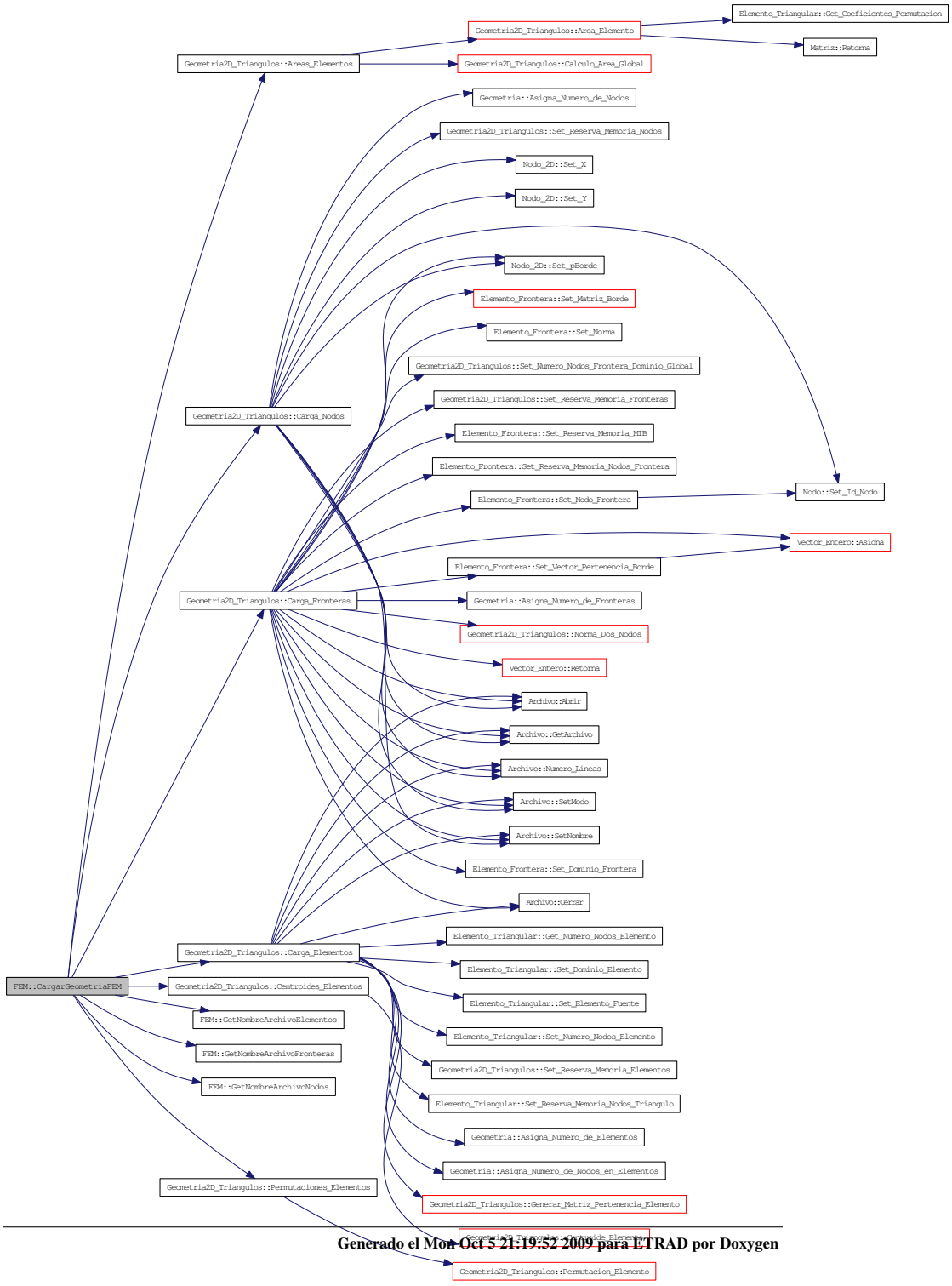
3.6.2. Documentación de las funciones miembro

3.6.2.1. void FEM::CargarGeometriaFEM ()

Carga la geometria y calcula las areas de los elementos y matrices de permutacion elemental, que luego son utilizados al momento de generar las matrices de rigidez y de masa.

Carga de la geometria FEM, en esta instancia se carga en memoria los nodos y elementos internos y fronteras con los respectivos subdominios a los cuales perteneces cada uno, ademas se realizan calculos preliminares como las areas de los elementos y permutaciones elementales

Gráfico de llamadas para esta función:

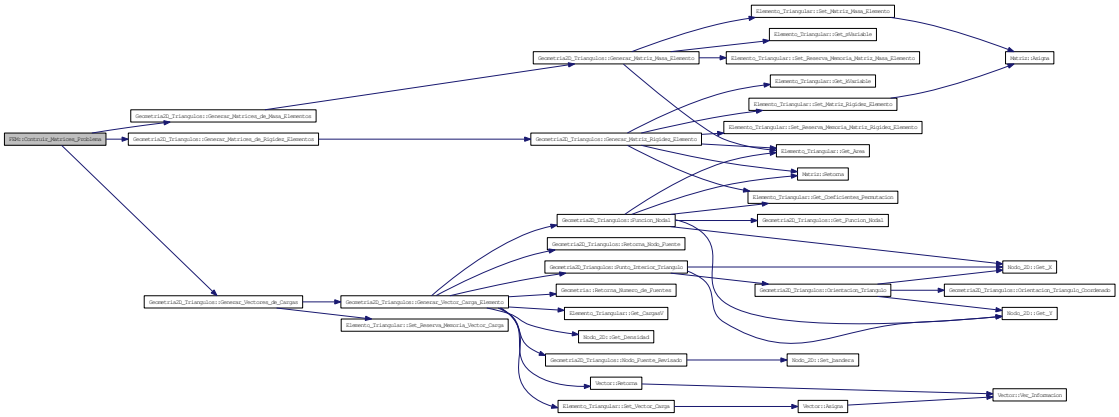


3.6.2.2. void FEM::Contruir_Matrices_Problema ()

Calcula las matrices de rigidez, de masa y vector de carga de los elementos,.

Contruccion de matrices por elemento de Rigidez, masa y vectores de carga.

Gráfico de llamadas para esta función:

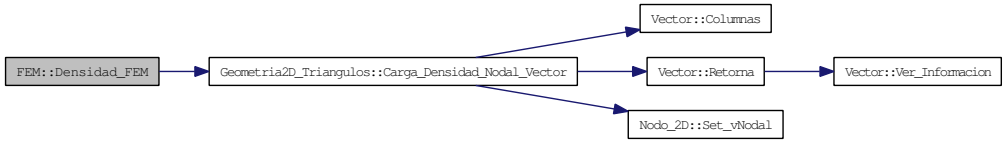


3.6.2.3. void FEM::Densidad_FEM (void)

Calcula la dendidad global de la solucion FEM, es una suma de densidades nodales por sus funciones de forma nodal (Se utiliza solo como funcion de traza del programa).

Carga la densidad obtenida por la FEM en los objetos Nodos de la geometria.

Gráfico de llamadas para esta función:



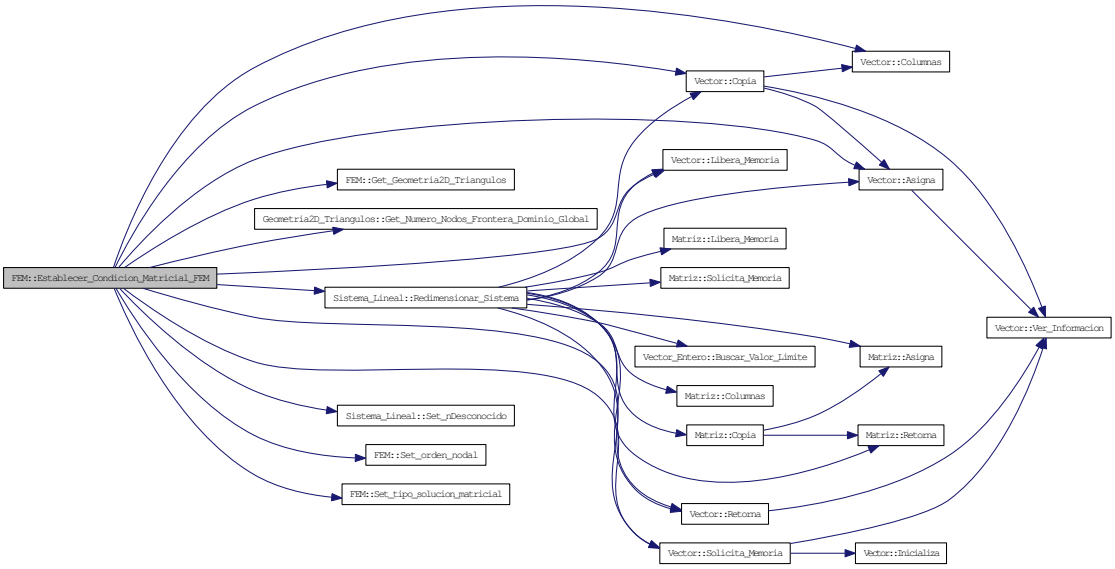
3.6.2.4. void FEM::Establecer_Condicion_Matricial_FEM (int *condicion*, int *metodo*)

Establece el tipo de tratamiento numerico con el cula se trabaja en la resolucion del sistema lineal (Matriz completa o reducida).

Establece condicional de la solucion matricial 1-Matrices normales, 2-Matrices reducidas.

Elimina la informacion de tipo map de columnas, filas y datos, a partir del vector de entrada donde se indica que filas - columnas se quieren eliminar de la matriz

Gráfico de llamadas para esta función:

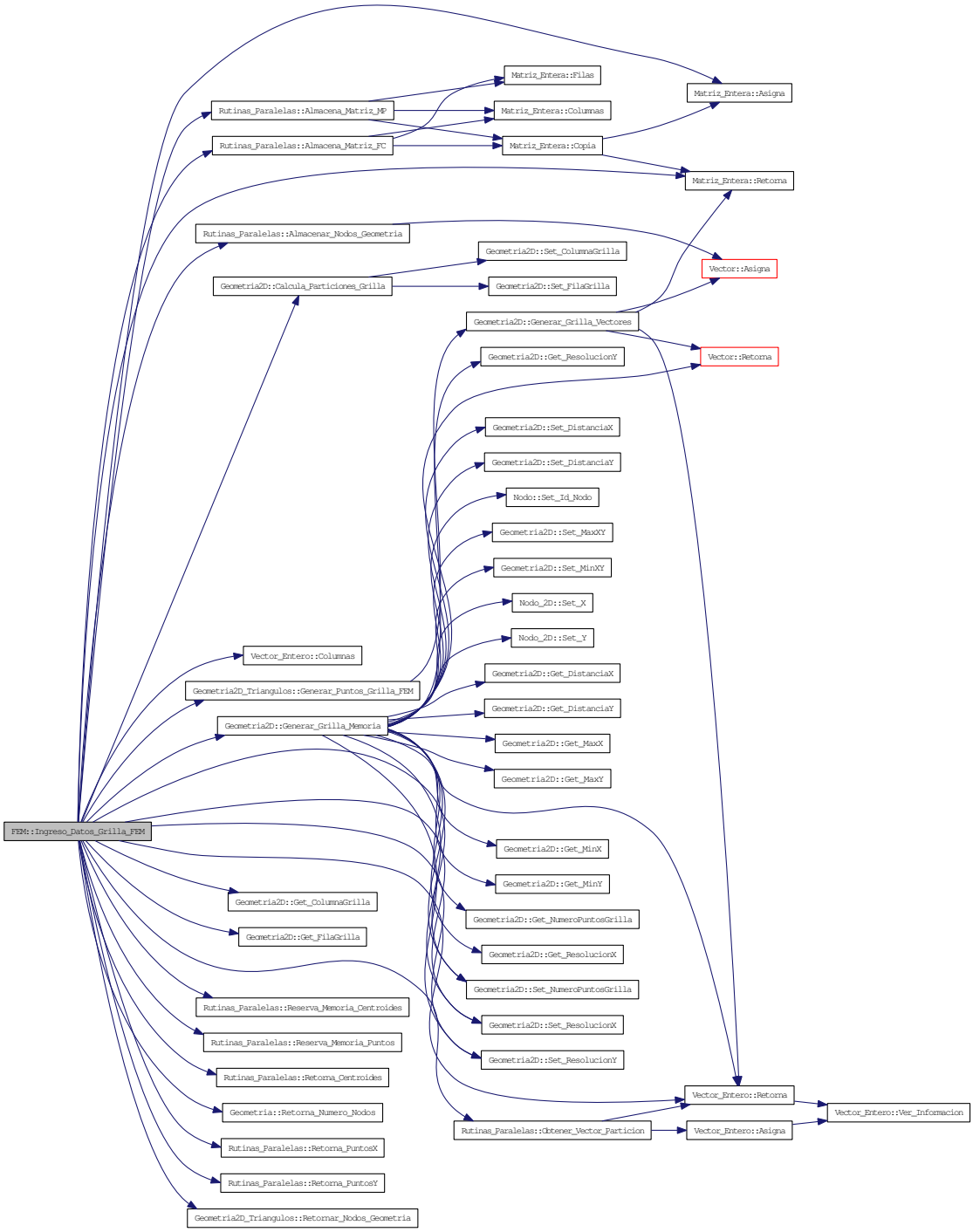


3.6.2.5. void FEM::Ingreso_Datos_Grilla_FEM (Vector * *dominio*, Vector_Entero * *Resolucion*, int *nProcesos*)

Adquisicion de datos de la grilla a calcular.

Calculo de densidades a partir de una grilla de puntos una vez ya realizada una aproximacion nodal por el metodo FEM.

Gráfico de llamadas para esta función:

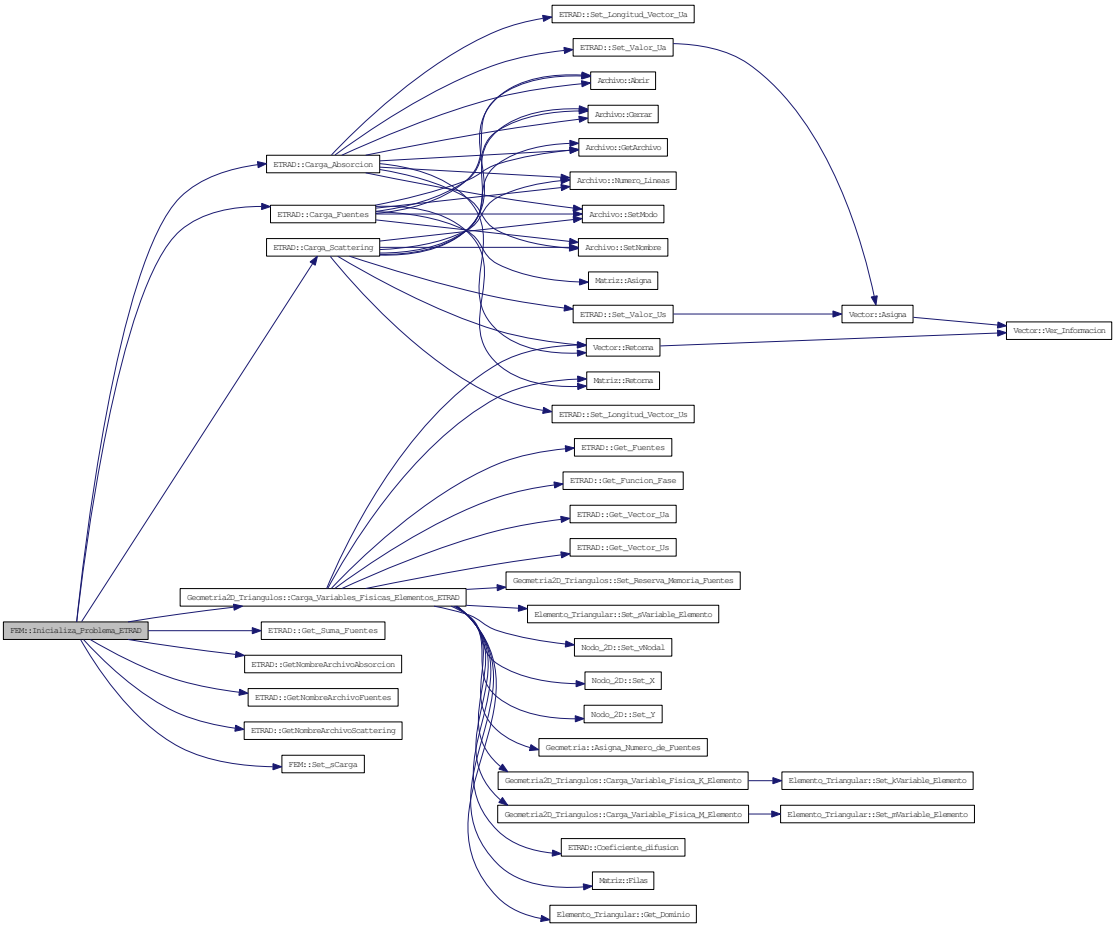


3.6.2.6. void FEM::Inicializa_Problema_ETRAD (ETRAD ∗ *FisicaProblema*, int *tipoFase*)

Inicializa problema ETRAD (rECORDAR CAMBIAR IMPLEMNETACION Y DEJARLA EN UNA CLASE SEPARADA, SOLO SE DEJO ACA POR COMODIDAD DEPROGRMACION Y TRAZA).

Inicializa el problema de difusion de la luz. se cargan en memoria la informacion de la absorcion, scatter y la posicion y densidad de las fuentes de fotones.

Gráfico de llamadas para esta función:



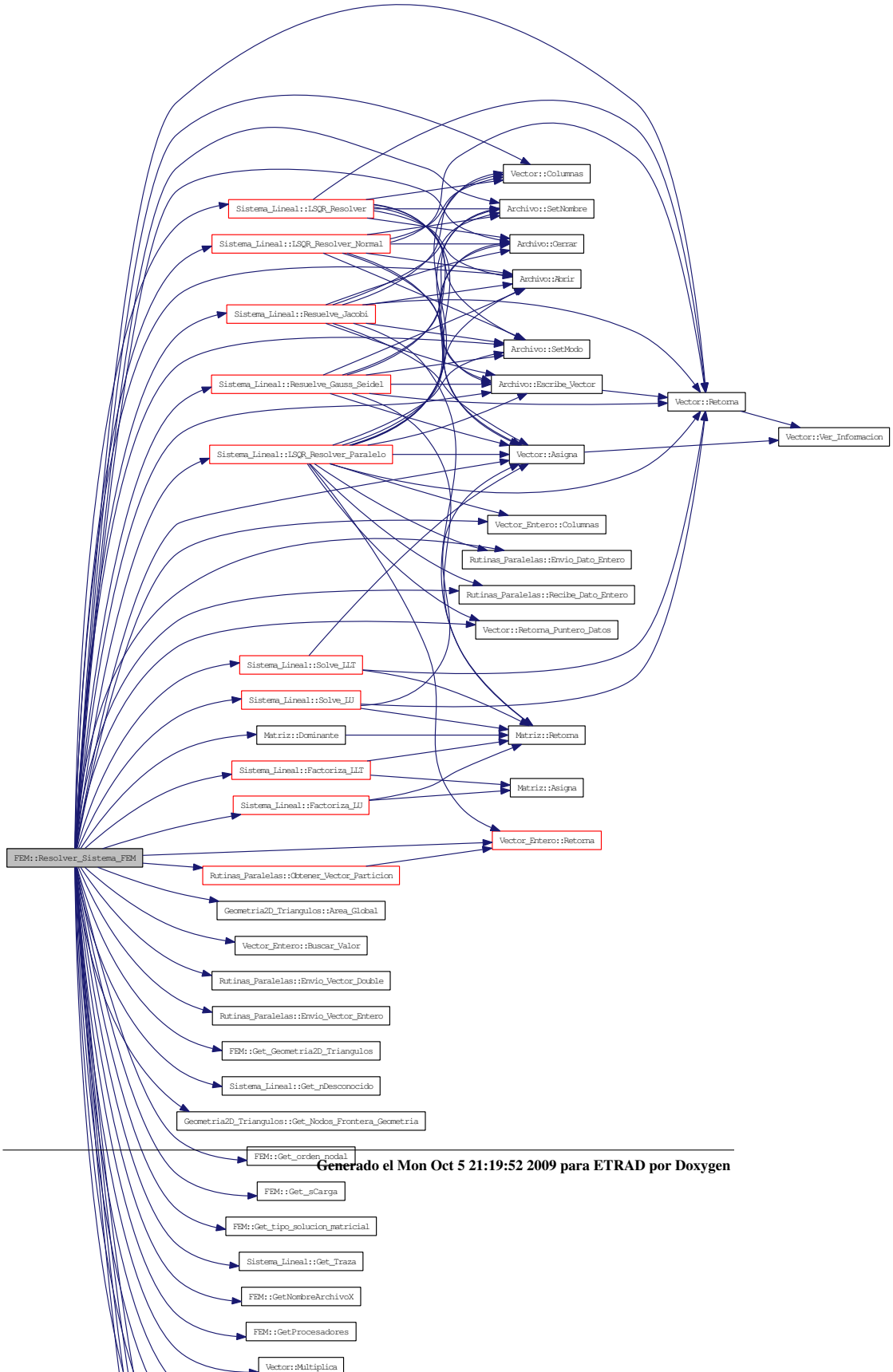
3.6.2.7. void FEM::Resolver_Sistema_FEM (int *metodo*, int *traza*, unsigned long long *iteraciones*, int *tipo_parada*)

Resolucion del sistema de ecuaciones.

Resuelve el sistema de ecuaciones con un metodo especifico: Jacobi / Gauss - Seidel / Factorizacion LU / Factorizacion Cholesky / LSQR (Matriz Completa) / LSQR (Matriz Dispersa en notacion vectorial) (Secuencial - Paralelo).

Almacenamiento de solucion a archivo

Gráfico de llamadas para esta función:



3.6.2.8. void FEM::VerTipoGeometria (void)

Muestra el tipo de geometria que se esta utilizando.

Muestra el tipo de geometria utilizada.

3.6.3. Documentación de los datos miembro

3.6.3.1. map<unsigned long long,double> FEM::Matriz_A_FEM

Mapeo de la Matriz de riguidez del sistema (Matriz stiffness (Fisico-geometria) + Matriz Masa (fisico-masa)) Se utiliza este mapeo para reservar memoria solo para los valoress no nulos de la matriz dispersa

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/FEM.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/FEM.cpp

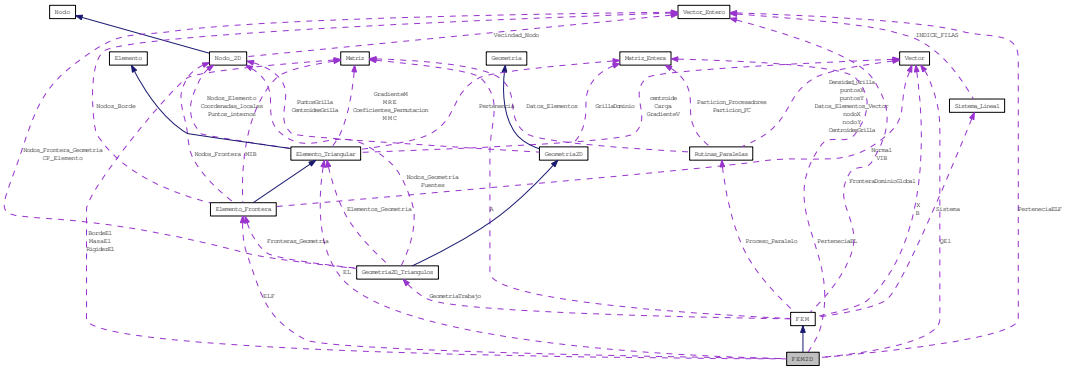
3.7. Referencia de la Clase FEM2D

Clase donde se define un problema FEM de dos dimensiones.

`#include <FEM2D.hpp>`

Herencias FEM.

Diagrama de colaboración para FEM2D:



Métodos públicos

- ~FEM2D ()

  Destructor de la clase.

- void Libera_Memoria_Carga ()

  Libera carga de memoria.

- void Inicializar_FEM2D (int Tipo_resolucion_AX)

  Reserva memoria para el sistema de ecuaciones, segun geometria utilizada.

- void Llena_Matriz_B_FEM ()

  Realiza un llamada al metodo de contruccion de la vector de carga global B segun la geometria utilizada.

- void Llena_Matriz_A_FEM (int borde, int tipo_solucion_matricial)

  Realiza un llamada al metodo de contruccion de la matriz global A segun la geometria utilizada.

- void Set_Suma_Nodal_B_FEM (unsigned long long i, double aNodal)

  Suma un aporte nodal al vector de cargas global B del sistema de ecuaciones.

- void Set_Suma_Nodal_A_FEM (unsigned long long i, unsigned long long j, double aNodal)

  Suma un aporte nodal a la matriz global A del sistema de ecuaciones.

- void Set_Reserva_Memoria_A_FEM (unsigned long long nNodo)

  Reserva memoria para la matriz global A del sistema de ecuaciones.

- void Set_Reserva_Memoria_B_FEM (unsigned long long nNodos)

  Reserva memoria para la matriz global B del sistema de ecuaciones.

- void Llena_Matriz_B_FEM_2D_Triangulos ()

  Construye el vector de cargas global B del sistema de ecuaciones calculada con elementos finitos triangulares.

- void Llena_Matriz_A_FEM_2D_Triangulos (int tipo_AX)

  Construye la matriz global A del sistema de ecuaciones a partir de las matrices de rigidez y de masa, calculada con elementos finitos triangulares.

- void Aporte_Matriz_A_FEM_Borde_2D_Triangulos (int tipo_AX)

  Aporta a la matriz global A del sistema de ecuaciones los calculos del borde global.

Atributos públicos

- Matriz_Entera * **PerteneciaEL**
- Matriz * **RigidezEl**
- Matriz * **MasaEl**
- Elemento_Triangular * **EL**
- Elemento_Frontera * **ELF**
- Vector * **QEl**
- Vector_Entero * **PerteneciaELF**
- Matriz * **BordeEl**

3.7.1. Descripción detallada

Clase donde se define un problema FEM de dos dimensiones.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/FEM2D.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/FEM2D.cpp

3.8. Referencia de la Clase Geometria

Clase base para generar una geometria.

`#include <Geometria.hpp>`

Heredado por Geometria2D.

Métodos públicos

- Geometria (void)

  Constructor de la clase.

- ~Geometria ()

  Destructor de la clase.

- void Asigna_Numero_de_Nodos (int num)

  Retorna el numero de nodos.

- void Asigna_Numero_de_Elementos (int num)

  Asigna el numero de elementos de la geometria.

- void Asigna_Numero_de_Fronteras (int num)

  Asigna el numero de fronteras de la geometria.

- void Asigna_Numero_de_Nodos_en_Elementos (int num)

  Asigna el numero de nodos por elemento de la geometria (En caso de utilizar otros tipos de elementos).

- void Asigna_Numero_de_Fuentes (int num)

  Asigna el numero de fuentes de energia, para problemas de difusion.

- int Retorna_Numero_de_Fuentes (void)

  Retorna el numero de fuentes.

- unsigned long Retorna_Numero_Fronteras (void)

  Retorna el nuemro de fronteras.

- unsigned long Retorna_Numero_Nodos (void)

  Retorna el numero de nodos.

- unsigned long Retorna_Numero_Elementos (void)

  Retorna el numero de elementos.

- int Retorna_nodos_elemento (void)

  Retorna el número de nodos por elemento.

Atributos protegidos

- unsigned long long Numero_Nodos

  Numero de nodos.

- unsigned long long Numero_Elementos

  Numero de elementos.

- int DIM

  Dimension a trabajar.

- int Numero_Nodos_Elemento

  Numero de nodos por elemento.

- unsigned long Numero_Fronteras

  Numero de fronteras.

- int Numero_Fuentes

  Numero de fuentes.

3.8.1. Descripción detallada

Clase base para generar una geometria.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Geometria.hpp

3.9. Referencia de la Clase Geometria2D

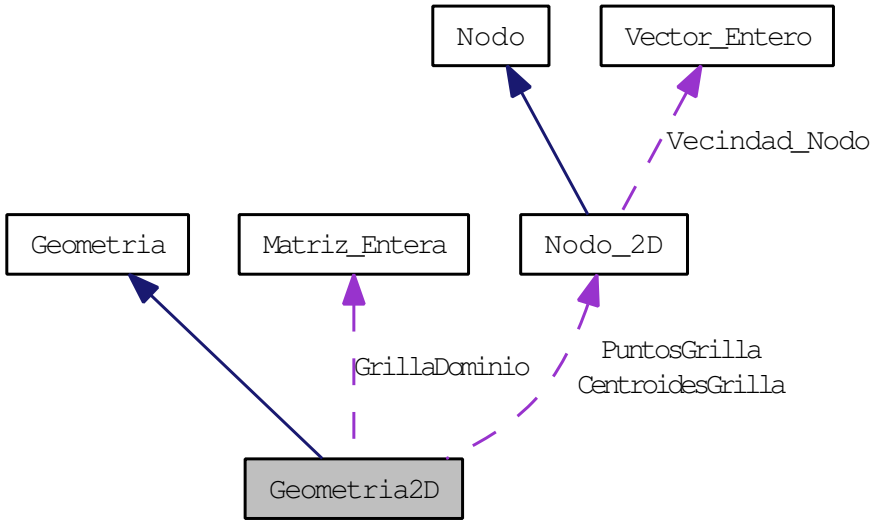
Clase base para generar una geometria en 2D.

`#include <Geometria2D.hpp>`

Herencias Geometria.

Heredado por Geometria2D_Triangulos.

Diagrama de colaboración para Geometria2D:



Métodos públicos

- Geometria2D (void)
  Constructor de la clase.
- ~Geometria2D ()
  Destructor de la clase.
- void Asigna_Tipo_Geometria (int tipoGeometria)
  Asigna el tipo de geometria 2D.
- Matriz_Entera * Get_Grilla ()
  Retorna la pertenencia del dominio de la grilla.
- Nodo_2D Retornar_Punto_Grilla (unsigned long id)

Retorna puntos de la grilla.

- Nodo_2D Retornar_Centroide_Grilla (unsigned long id)

  Retorna los centroides de la grilla.

- Nodo_2D * Retornar_Puntos_Grilla ()

  Retorna puntero de los puntos de la grilla.

- Nodo_2D * Retornar_Centroides_Grilla ()

  Retorna puntero de centroides de la grilla.

- int Retorna_Tipo_Geometria ()

  Retorna el tipo de geometria 2D.

- double Calculo_Area_Diferencial ()

  Calcula el area diferencial por cada nodo a partir de una grilla de entrada.

- void Set_MinXY (double mX, double mY)

  Almacena los minimo del dominio global.

- void Set_MaxXY (double mX, double mY)

  Almacena los maximos del dominio global.

- void Set_NumeroPuntosGrilla (unsigned long long valor)

  Almacena el numero total de puntos de la grilla.

- unsigned long Get_NumeroPuntosGrilla ()

  Retorna el numeo de puntos de la grilla.

- void Set_DistanciaX (double valor)

  Almacena la distancia entre el minimo y el maximo del eje X de la grilla.

- void Set_DistanciaY (double valor)

  Almacena la distancia entre el maximo y el minimo del eje Y de la grilla.

- void Set_FilaGrilla (int valor)

  Asigna el numero de filas a particionar la grilla en proceso paralelo.

- void Set_ColumnaGrilla (int valor)

  Asigna el numero de Columnas a particionar la grilla en el proceso paralelo.

- void Set_ResolucionX (unsigned long long valor)

Asigna la resolucion X de la grilla.

- void Set_ResolucionY (unsigned long long valor)

  Asigna la resolucion Y de la grilla.

- double Get_DistanciaX ()

  Retorna la distancia de X.

- double Get_DistanciaY ()

  Retorna la distancia de Y.

- int Get_FilaGrilla ()

  Retorna el numero de filas de la grilla.

- int Get_ColumnaGrilla ()

  Retorna el numero de columnas de la grilla.

- double Get_MaxX ()

  Retorna el maximo valor del dominio en X.

- double Get_MaxY ()

  Retorna el maximo valor del dominio en Y.

- double Get_MinX ()

  Retorna el minimo valor del dominio en X.

- double Get_MinY ()

  Retorna el minimo valor del dominio en Y.

- int Get_ResolucionX ()

  Retorna la resolucion en X.

- int Get_ResolucionY ()

  Retorna la solucion en Y.

- void Calcula_Particiones_Grilla (int particiones)

  Calcula particiones Grilla.

- void Carga_Imagen_Binaria (char *arch)

  Carga la informacion de la imagen binaria del dominio a calcular densidad.

- int Carga_Puntos_Archivo (char *arch)

Carga la informacion de puntos donde se desea obtener densidad.

- void Generar_Grilla_Vectores (Vector *dominio, Vector_Entero *Resolucion, Matriz_Entera *Dim_Fila_Columna, Matriz_Entera *Bandera_Fila_Columna, Vector *gX, Vector *gY, Vector *CentroGeometrico, int Proceso)

  Genera una grilla de puntos a partir de informacion particionada de la grilla y almacena los centros geometricos de la grilla.

- void Generar_Grilla_Memoria (Vector *dominio, Vector_Entero *Resolucion)

  Genera una grilla para un procesador.

Atributos públicos

- Matriz_Entera * GrillaDominio

  Grilla que almacena datos del dominio de la geometria 1=pertenece al dominio, 0 EOC.

- Nodo_2D * PuntosGrilla

  Puntos a evaluar de la grilla.

- Nodo_2D * CentroidesGrilla

  Centroides de particiones de la grilla, utilizado en la seleccion de elementos que estan contenidos en las subgrillas particionadas para el proceso paralelo.

- unsigned long NumeroPuntosGrilla

  Numero de puntos a calcular de la grilla.

Atributos protegidos

- int tipoGeometria2D

  Tipo de geometria, 2 - Triangulo, 3 Cuadrilatero.

- int nParticiones

  Numero de particiones - Uitlizado en calculo paralelo.

- int FilaGrilla

  Numero Filas particion paralela.

- int ColumnaGrilla

  Numero de columnas particion paralela.

- int ResolucionX

  Resolucion X,Y.

- int **ResolucionY**
- double minX

  Fronteras de la Grilla.

- double **minY**
- double **maxX**
- double **maxY**
- double Area_DIferencial

  Area diferencial.

- int tipoGrilla

  Tipo Grilla aproximacion en vertice = 1, en centro geometrico de 4 vertices que forman un cuadrilatero = 2.

- double distanciaX

  Distancias de la grilla.

- double **distanciaY**

3.9.1. Descripción detallada

Clase base para generar una geometria en 2D.

3.9.2. Documentación de las funciones miembro

3.9.2.1. void Geometria2D::Calcula_Particiones_Grilla (int *particiones*)

Calcula particiones Grilla.

Calcula particiones de la grilla y almacena en memoria el numero de filas y columnas que se utilizara en el proceso paralelo de particion de la grilla.

Gráfico de llamadas para esta función:

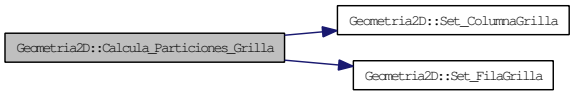


Gráfico de llamadas a esta función:

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Geometria2D.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Geometria2D.cpp

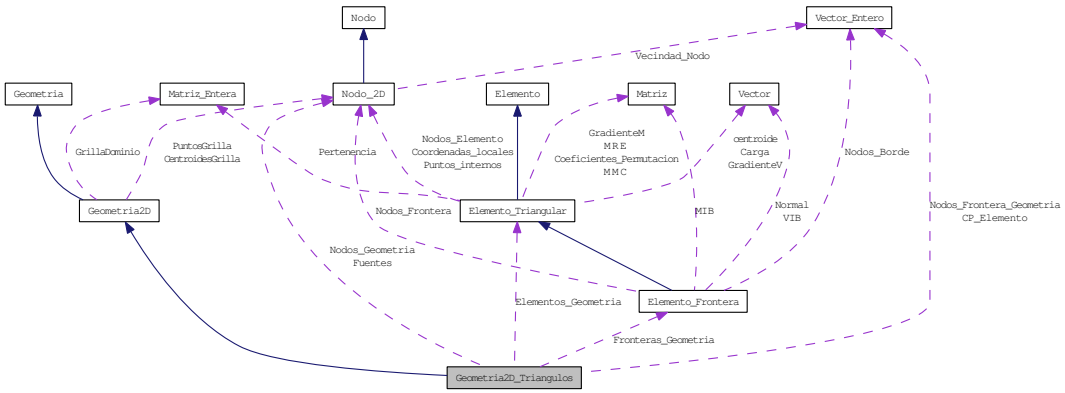
3.10. Referencia de la Clase Geometria2D_Triangulos

Clase para generar una geometria triangular 2D.

`#include <Geometria2D_Triangulos.hpp>`

Herencias Geometria2D.

Diagrama de colaboración para Geometria2D_Triangulos:



Métodos públicos

- Geometria2D_Triangulos (void)

  Constructor de la clase.

- ~Geometria2D_Triangulos ()

  Destructor de la clase.

- void Set_nPuntosInterior (int nPuntos)

  Almacena el numero de puntos interiores.

- unsigned long long **Retorna_nPuntosInterior** ()
- void Calculo_Area_Global ()

  Calculo de area global del dominio.

- void Almacenar_Area_Dominio (double aGlobal)

  Amacena el area global del dominio.

- double Area_Global ()

  Retorna el area del dominio global.

- void Generar_Vectores_de_Cargas ()

  Calcula los vectores de carga de todos los elementos del dominio.

- void Generar_Vector_Carga_Elemento (unsigned long long id)

  Calcula el vector de carga de un elemento "id".

- double Funcion_Nodal (unsigned long long id, int indice, Nodo_2D A1)

  Calcula la funcion de forma del Elemento "id", funcion "indice" y Nodo "A1".

- double Get_Funcion_Nodal (double areaE, double Ai, double Bi, double Ci, double pX, double pY)

  Calcula la funcion de forma con los parametros: Area del elemento, Ai, Bi y Ci (constantes geometricas del elemento) del elemento y un punto pX y pY cualquiera.

- double Permutacion_Nodal (Nodo_2D A1, Nodo_2D A2, Nodo_2D A3, int coeficiente, int indice)

  Retorna la constante geometrica del elemento (Ai, Bi o Ci) indicada por "coeficiente" de la funcion de forma "indice" del elemento, esta es una variante del calculo nodal.

- double Buscar_Mininos_Nodos (unsigned long long id, int eje)

  Busca el valor minimo de los nodos del elemento "id" en el eje X (0) o Y (1), util para obtener coordenadas locales del elemento.

- void Obtener_Coordenadas_Locales_Elemento (unsigned long long id)

  Obtiene y muestra las coordenadas locales del elemento "id".

- int Orientacion_Elemento (unsigned long long id)

  Retorna la orientacion de un elemento.

- int Orientacion_Triangulo (Nodo_2D A1, Nodo_2D A2, Nodo_2D A3)

  Retorna la orientacion de un triangulo dado 3 Nodos.

- int Punto_Interior_Triangulo (unsigned long long id, Nodo_2D Punto)

  Verifica si un punto pertence al interior de un elemento finito triangular id.

- int Orientacion_Triangulo_Coordenado (double x1, double x2, double x3, double y1, double y2, double y3)

  Retorna la orientacion de un triangulo dadas las 3 coordenadas.

- int Punto_Interior_Triangulo_Coordenado (double x1T, double x2T, double x3T, double y1T, double y2T, double y3T, double px, double py)

  Verifica si un punto pertence al interior de un triangulo dadas las tres coordenadas de un triangulo y el punto a evaluar.

- void Conjunto_Elementos_Vecinos_a_Nodo (unsigned long long idNodo)

  Busca los elementos vecinos a un nodo.

- double Distancia_Nodo_Punto (unsigned long long idNodo, double xCord, double yCord)

  Calcula la distancia de un Nodo "idNodo" a un punto cualquiera (x,y) = (xCord,yCord).

- double Distancia_Dos_Puntos (double xCord1, double xCord2, double yCord1, double yCord2)

  Retorna la distancia de dos puntos.

- double Distancia_Centro_Elemento_Punto (double xCord, double yCord, unsigned long long id)

  Calcula la distancia del centro del elemento "id" a un punto (x,y) = (xCord,yCord).

- void Buscar_Minino_Centroide_Triangulo_Fuente (double xCord, double yCord)

  Busca el Triangulo que tiene menor distancia desde su centro geometrico a un punto (x,y) = (xCord,yCord).

- void Set_Reserva_Memoria_Fuentes (int nFuentes)

  Reserva memoria para las fuentes.

- void Set_Reserva_Memoria_Contenedor_Punto_Elemento (unsigned long long nElementosP)

  Reserva memoria vector contenedor de puntos fuente (VER LA FORMA DE CREAR SUBVECTORES DE LARGO VARIABLE SEGUN PERTENENCIAS).

- void Set_Reserva_Memoria_Nodos (unsigned long long nNodos)

  Reserva memoria para el arreglo de objetos Nodo_2D[], que contendran informacion de nodos de la geometria.

- void Set_Reserva_Memoria_Elementos (unsigned long long nElementos)

  Reserva memoria para el arreglo de objetos Elemento_Triangular[], que contendran informacion de los elementos de la geometria.

- void Set_Reserva_Memoria_Fronteras (unsigned long long nCaras)

  Reserva memoria para el arreglo de objetos Elemento_Frontera[], que contendran informacion de los elementos de la geometria que tienen nodos en el borde del dominio.

- void Carga_Densidad_Nodal_Archivo (char ∗arch)

Carga las densidades nodales desde un archivo, cuyos valores fueron obtenidos por el metodo de elementos finitos.

- void Carga_Densidad_Nodal_Vector (Vector *X)

  Funcion que carga las densidades nodales obtenidas por el metodo de elementos finitos.

- void Calcula_Densidades_Puntos (Vector *dGrilla, unsigned long long *Puntos_Dominio)

  Calcula las densidades de una serie de puntos definidos en una grilla - Este proceso utiliza notacion de objetos Nodo 2D y esta implementada para version secuencial.

- void Calcula_Densidades_Puntos_Grilla (int procesador, Vector_Entero *pCentroides, unsigned long long nElementos, Vector *DENSIDADES, double *Entrada, double *px, double *py, double *nx, double *ny, unsigned long long *puntos_internos)

  Calcula densidades a partir de una grilla de puntos - Esta funcion es la utilizada en los procesos paralelos, por eso se utiliza notacion vectorial y de punteros, ademas es mas facil de manipular en el envio y recepcion de mensajes.

- void Carga_Nodos (char *arch)

  Carga la informacion de los Nodos de la geometria desde el archivo en formato Matlab-PDETool.

- void Carga_Elementos (char *arch)

  Carga la informacion de los Elementos de la geometria desde el archivo en formato Matlab-PDETool.

- void Carga_Fronteras (char *arch)

  Carga la informacion de las Fronteras de la geometria desde el archivo en formato Matlab-PDETool.

- void Ver_Nodo (unsigned long long id)

  Muestra el nodo "id" de la geometria.

- void Ver_Nodos (void)

  Muestra todos los nodos de la geometria.

- void Ver_Elemento (unsigned long long id)

  Muestra informacion de elemento "id" de la geometria.

- void Ver_Elementos (void)

  Muestra informacion de todos los elementos de la geometria.

- void Ver_Coordenadas_Elemento (unsigned long long id)

  Muestra las coordenadas (x,y) de un elemento "id" de la geometria.

- void Ver_Coordenadas_Todos_los_Elementos ()

  Muestra las coordenadas (x,y) de todos los elementos de la geometria.

- void Centroide_Elemento (unsigned long long id)

  Calcula el centroide del elemento "id" de la geometria.

- void Centroides_Elementos ()

  Calcula los centroides de todos los elementos de la geometria.

- void Permutacion_Elemento (unsigned long long id)

  Calcula la permutacion en orden natural, asociada a las funciones de forma del elemento "id" de la geometria.

- void Permutaciones_Elementos ()

  Calcula las permutaciones de todos los elementos de la geometria.

- void Area_Elemento (unsigned long long id)

  Calcula el area del elemento "id" de la geometria.

- void Areas_Elementos ()

  Calcula todas las areas de los elementos de la geometria.

- void Ver_Area_Elemento (unsigned long long id)

  Muestra el area del elemento "id".

- void Ver_Areas_Elementos ()

  Muestra las areas de todos los elementos.

- void Generar_Matriz_Rigidez_Elemento (unsigned long long id)

  Genera la matriz de rigidez de un elemento "id" de la geometria.

- void Generar_Matriz_Masa_Elemento (unsigned long long id)

  Genera la matriz de masa de un elemento "id" de la geometria.

- void Vector_Gradiente_Elemento (unsigned long long id, unsigned long long nodo)

  Calcula el vector gradiente del elemento "id" en el nodo "nodo".

- void Matriz_Gradiente_Elemento (unsigned long long id, unsigned long long nodo)

Calcula la matriz gradiente del elemento "id" en el nodo "nodo".

- Vector_Entero * Get_Nodos_Frontera_Geometria ()

  Retorna el vector que contiene los identificadores de los nodos que estan en la frontera (borde) del dominio.

- double Norma_Dos_Nodos (unsigned long long id_nodo1, unsigned long long id_nodo2)

  Retorna la norma entre dos nodos.

- void Generar_Matriz_Pertenencia_Elemento (unsigned long long id)

  Genera la matriz de pertencia del elemento "id".

- void Carga_Variable_Fisica_M_Elemento (unsigned long long id, double Valor)

  Carga la variable fisica asociada a la fisico-masa del elemento "id".

- void Carga_Variable_Fisica_K_Elemento (unsigned long long id, double Valor)

  Carga la variable fisica asociada a la fisico-geometria del elemento "id".

- Nodo_2D * Retornar_Nodos_Geometria ()

  Retorna el arreglo de objetos Nodo_2D[] de la Geometria.

- Nodo_2D Retorna_Nodo_Fuente (int id)

  Retorna el arreglo de objetos Nodo_2D[] que tienen informacion de las fuentes.

- void Nodo_Fuente_Revisado (int id)

  Asigna como nodo revisado.

- Nodo_2D Retornar_Nodos_Especifico (unsigned long long id)

  Retorna un Nodo_2D[] de la geometria.

- Elemento_Triangular * Retornar_Elementos_Geometria ()

  Retorna el arreglo de objetos Elemento_Triangular[] de la geometria.

- Elemento_Frontera * Retornar_Fronteras_Geometria ()

  Retorna el arreglo de objetos Elemento_Frontera[] de la geometria.

- void Carga_Variables_Fisicas_Elementos_ETRAD (ETRAD *TMP)

  Carga variables fisicas del problema - Dependiendo de los tipos de problemas se deben agregar los distintos metodos de calculo de constantes.

- void Set_Numero_Nodos_Frontera_Dominio_Global (unsigned long long num)

  Asigna el numero de nodos frontera en el dominio global.

- unsigned long long Get_Numero_Nodos_Frontera_Dominio_Global ()

  Retorna el numero global de nodos en la frontera.

- void Generar_Matrices_de_Masa_Elementos ()

  Genera las matrices de masa de todos los elementos de la geometria.

- void Generar_Matrices_de_Rigidez_Elementos ()

  Genera las matrices de rigides de todos los elementos de la geometria.

- void Generar_Puntos_Grilla_FEM (Vector *dominio, Vector_Entero *Resolucion, Matriz_Entera *Dim_Fila_Columna, Matriz_Entera *Bandera_Fila_Columna, Vector *gX, Vector *gY, Vector *CentroGeometrico, int Proceso)

  Lamada a generar una grilla y almacenar las coordenadas en vectores.

Atributos públicos

- double fEsfera

  Fase en la fuente.

3.10.1. Descripción detallada

Clase para generar una geometria triangular 2D.

3.10.2. Documentación de las funciones miembro

3.10.2.1. void Geometria2D_Triangulos::Area_Elemento (unsigned long long *id*)

Calcula el area del elemento "id" de la geometria.

Calcula el area de un elemento utilizando la matriz de permutacion del elemento finito trinagular de tres nodos.

Gráfico de llamadas para esta función:

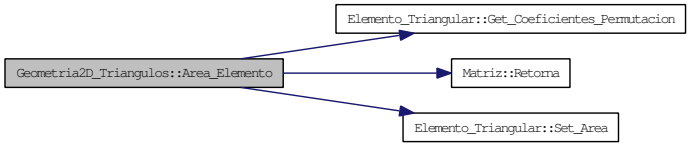
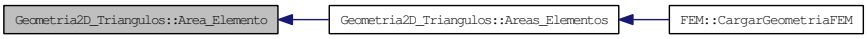


Gráfico de llamadas a esta función:



3.10.2.2. void Geometria2D_Triangulos::Areas_Elementos ()

Calcula todas las areas de los elementos de la geometria.

Calcula todas las areas de los elementos finitos triangulares de tres nodos.

Gráfico de llamadas para esta función:

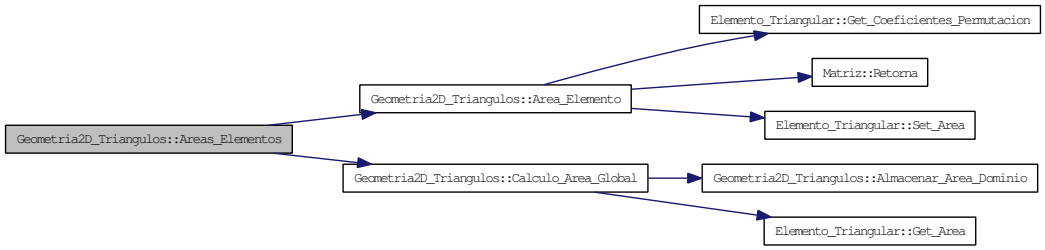


Gráfico de llamadas a esta función:

3.10.2.3. double Geometria2D_Triangulos::Buscar_Mininos_Nodos (unsigned long long *id*, int *eje*)

Busca el valor minimo de los nodos del elemento "id" en el eje X (0) o Y (1), util para obtener coordenadas locales del elemento.

Busca el valor minimo nodal - puede ser usado para normalizar graficas.

Gráfico de llamadas para esta función:

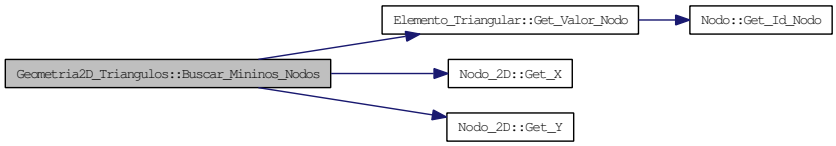


Gráfico de llamadas a esta función:

3.10.2.4. void Geometria2D_Triangulos::Calcula_Densidades_Puntos (Vector * *dGrilla*, unsigned long long * *Puntos_Dominio*)

Calcula las densidades de una serie de puntos definidos en una grilla - Este proceso utiliza notacion de objetos Nodo 2D y esta implementada para version secuencial.

Calcula densidades a partir de una grilla de puntos en notacion utilizando objetos Nodo_2D, solo para calculo secuencial.

Gráfico de llamadas para esta función:

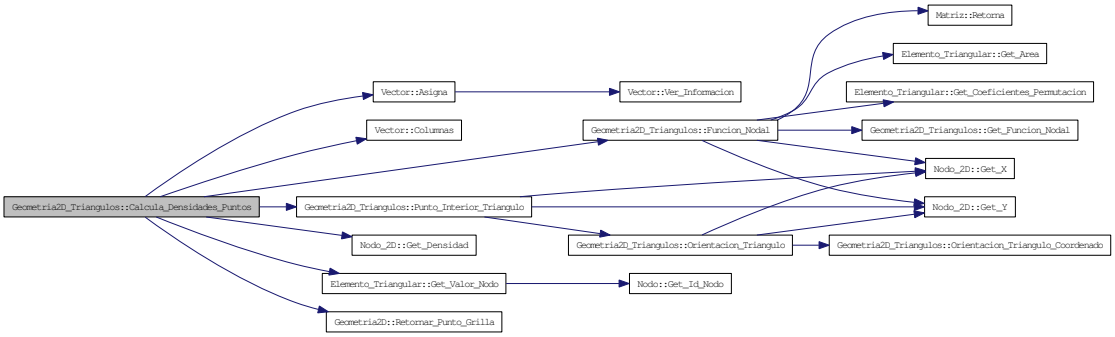


Gráfico de llamadas a esta función:

3.10.2.5. void Geometria2D_Triangulos::Calculo_Area_Global ()

Calculo de area global del dominio.

Calcula el area global a partir de las sumas de las areas de los elementos.

Gráfico de llamadas para esta función:

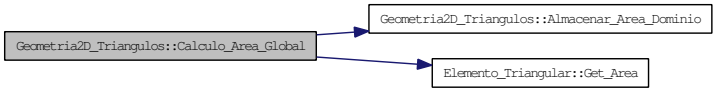
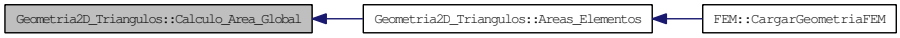


Gráfico de llamadas a esta función:

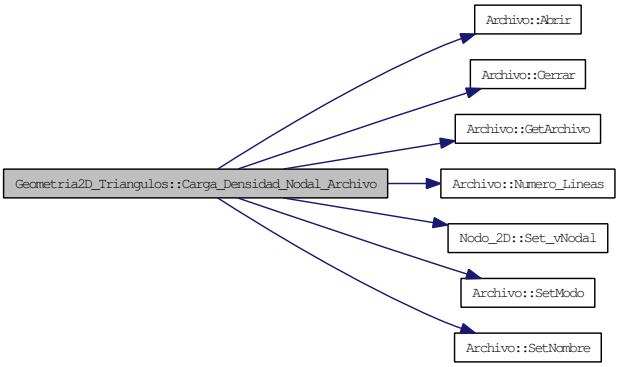


3.10.2.6. void Geometria2D_Triangulos::Carga_Densidad_Nodal_Archivo (char * *arch*)

Carga las densidades nodales desde un archivo, cuyos valores fueron obtenidos por el metodo de elementos finitos.

Carga solucion Nodal desde archivo.

Gráfico de llamadas para esta función:



3.10.2.7. void Geometria2D_Triangulos::Carga_Densidad_Nodal_Vector (Vector * *X*)

Funcion que carga las densidades nodales obtenidas por el metodo de elementos finitos.

Carga solucion Nodal - Desde Vector.

Gráfico de llamadas para esta función:

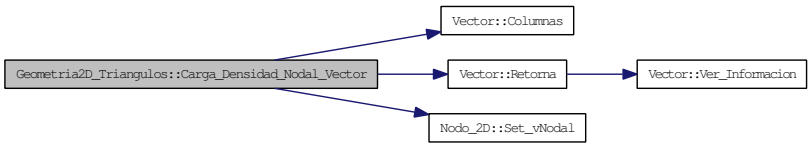


Gráfico de llamadas a esta función:

3.10.2.8. void Geometria2D_Triangulos::Carga_Elementos (char * *arch*)

Carga la informacion de los Elementos de la geometria desde el archivo en formato Matlab-PDETool.

Carga la informacion de los elementos desde un archivo.

Gráfico de llamadas para esta función:

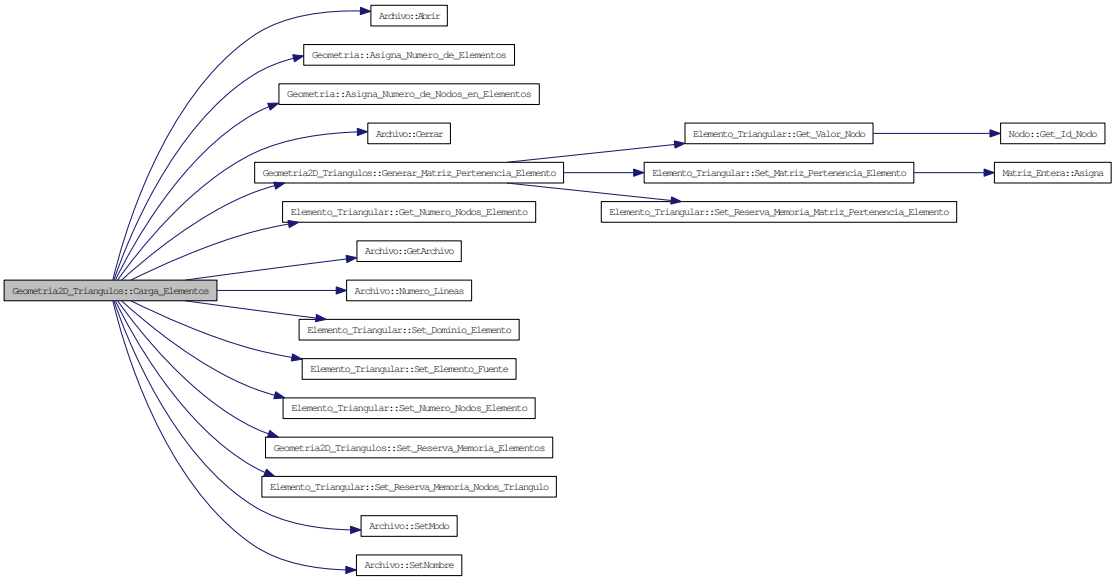


Gráfico de llamadas a esta función:

3.10.2.9. void Geometria2D_Triangulos::Carga_Fronteras (char * *arch*)

Carga la informacion de las Fronteras de la geometria desde el archivo en formato Matlab-PDETool.

Carga la informacion de las fronteras de los dominios desde un archivo.

Gráfico de llamadas para esta función:

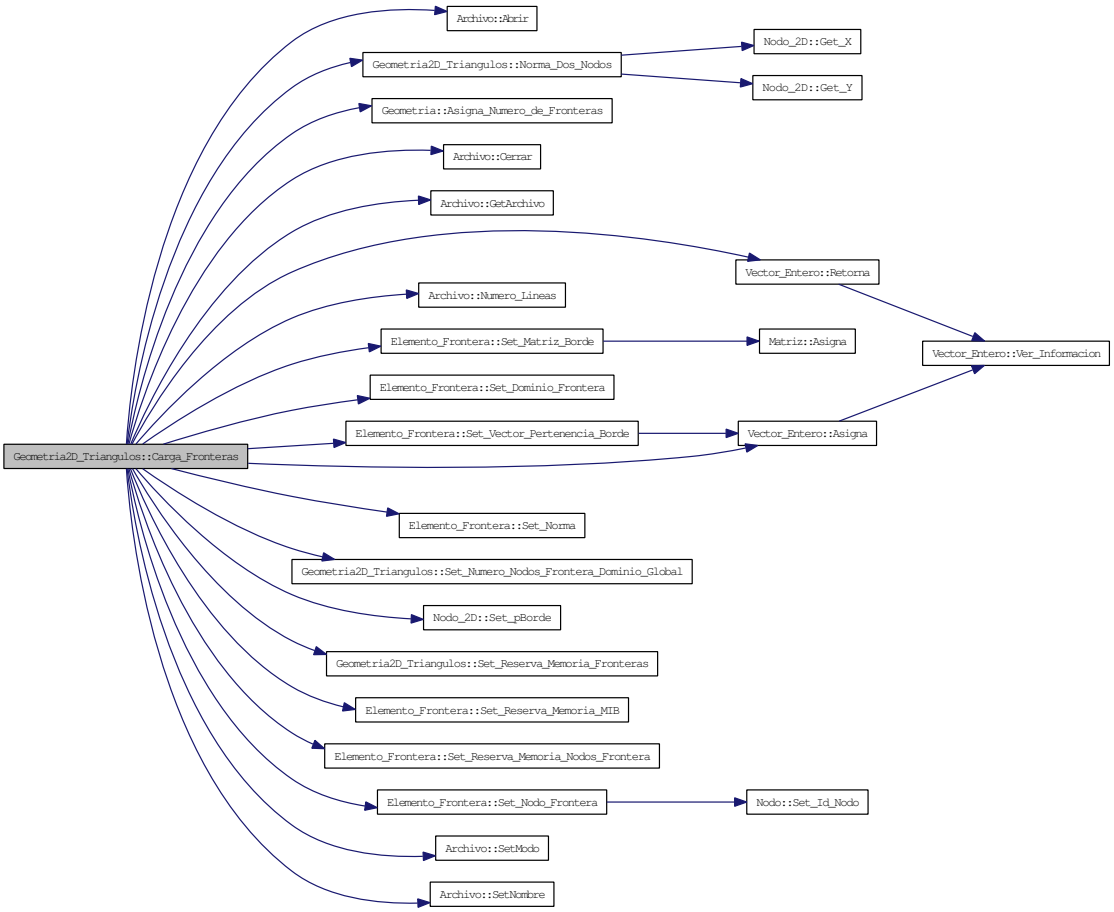


Gráfico de llamadas a esta función:

3.10.2.10. void Geometria2D_Triangulos::Carga_Nodos (char * *arch*)

Carga la informacion de los Nodos de la geometria desde el archivo en formato Matlab-PDETool.

Carga la informacion de de los nodos desde un archivo.

Gráfico de llamadas para esta función:

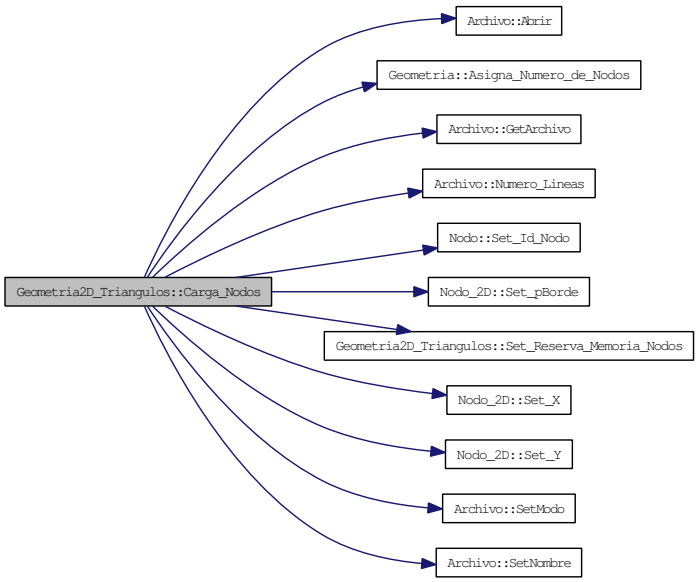


Gráfico de llamadas a esta función:

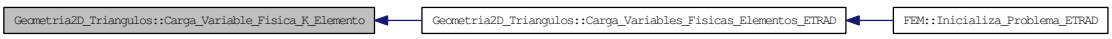
3.10.2.11. void Geometria2D_Triangulos::Carga_Variable_Fisica_K_Elemento (unsigned long long *id*, double *Valor*)

Carga la variable fisica asociada a la fisico-geometria del elemento "id".

Carga el factor fisico asociado a la fisica-geometria , en el caso de la ETR-AD es el scattering.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:



3.10.2.12. void Geometria2D_Triangulos::Carga_Variable_Fisica_M_Elemento (unsigned long long *id*, double *Valor*)

Carga la variable fisica asociada a la fisico-masa del elemento "id".

Carga el factor fisico asociado a la fisica-masa , en el caso de la ETR-AD es absorcion.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

3.10.2.13. void Geometria2D_Triangulos::Carga_Variables_Fisicas_-Elementos_ETRAD (ETRAD ∗ *TMP*)

Carga variables fisicas del problema - Dependiendo de los tipos de problemas se deben agregar los distintos metodos de calculo de constantes.

Metodo que carga la informacion fisica de un problema de difusion del tipo ETR-AD.

Gráfico de llamadas para esta función:

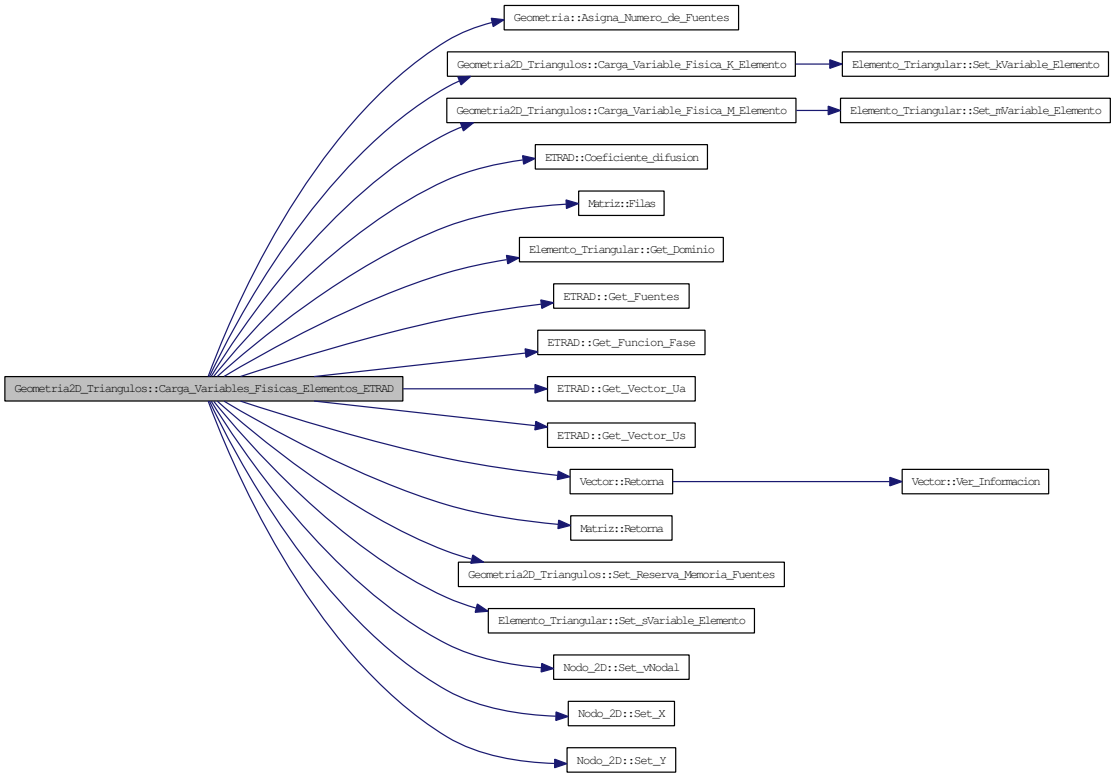


Gráfico de llamadas a esta función:

3.10.2.14. void Geometria2D_Triangulos::Centroide_Elemento (unsigned long long *id*)

Calcula el centroide del elemento "id" de la geometria.

Calcula el centro geometrico de un elemento finito triangular de tres nodos.

Gráfico de llamadas para esta función:

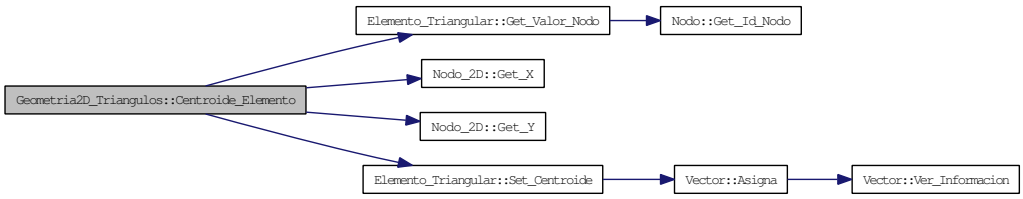
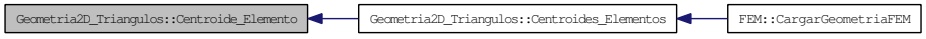


Gráfico de llamadas a esta función:



3.10.2.15. void Geometria2D_Triangulos::Centroides_Elementos ()

Calcula los centroides de todos los elementos de la geometria.

Calcula todos los centros geometricos de todos los elementos finitos triangulares de tres nodos.

Gráfico de llamadas para esta función:

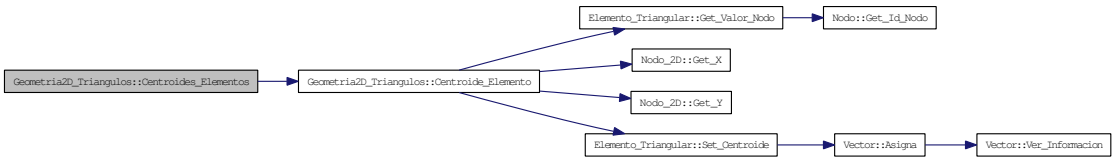


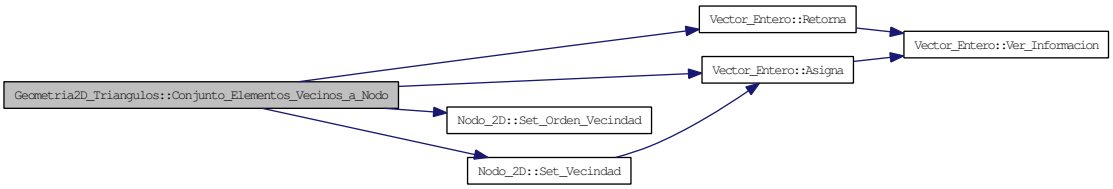
Gráfico de llamadas a esta función:

3.10.2.16. void Geometria2D_Triangulos::Conjunto_-Elementos_Vecinos_a_Nodo (unsigned long long *idNodo*)

Busca los elementos vecinos a un nodo.

Obtiene el conjunto de elementos que contienen a un nodo en comun.

Gráfico de llamadas para esta función:

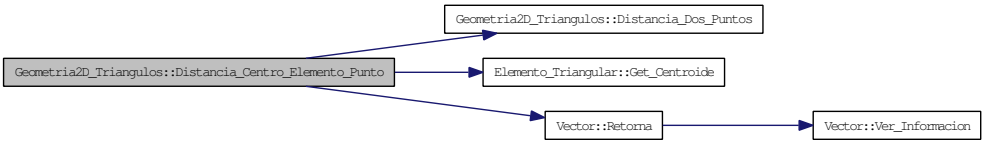


3.10.2.17. double Geometria2D_Triangulos::Distancia_Centro_Elemento_-Punto (double *xCord*, double *yCord*, unsigned long long *id*)

Calcula la distancia del centro del elemento "id" a un punto (x,y) = (xCord,yCord).

Distancia del centro de un elemento a un punto.

Gráfico de llamadas para esta función:

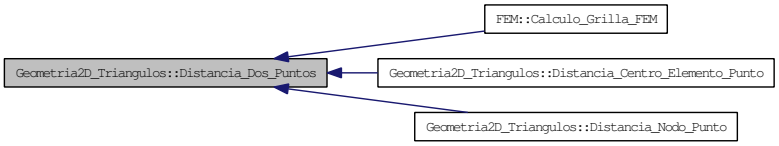


3.10.2.18. double Geometria2D_Triangulos::Distancia_Dos_Puntos (double *xCord1*, double *xCord2*, double *yCord1*, double *yCord2*)

Retorna la distancia de dos puntos.

Calcula la distancia de dos puntos, dada las coordenadas de los dos puntos.

Gráfico de llamadas a esta función:



3.10.2.19. double Geometria2D_Triangulos::Distancia_Nodo_Punto (unsigned long long *idNodo*, double *xCord*, double *yCord*)

Calcula la distancia de un Nodo "idNodo" a un punto cualquiera (x,y) = (xCord,yCord).

Distancia de un nodo a un punto.

Gráfico de llamadas para esta función:

3.10.2.20. double Geometria2D_Triangulos::Funcion_Nodal (unsigned long long *id*, int *indice*, Nodo_2D *A1*)

Calcula la funcion de forma del Elemento "id", funcion "indice" y Nodo "A1".

Retorna la funciona nodal de un elemento finito triangular id, indicando el nodo y un punto cualquiera.

Gráfico de llamadas para esta función:

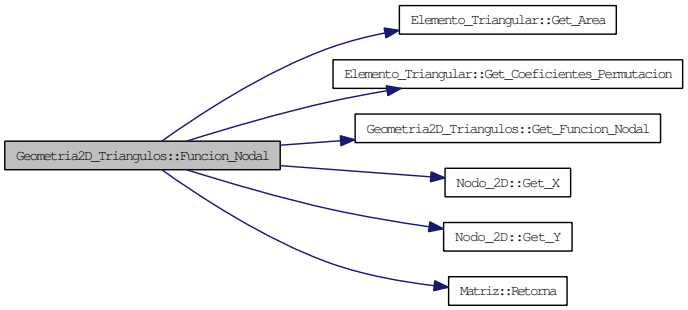


Gráfico de llamadas a esta función:

3.10.2.21. void Geometria2D_Triangulos::Generar_Matrices_de_Masa_-Elementos ()

Genera las matrices de masa de todos los elementos de la geometria.

Genera las matrices de masas de todos los elementos.

Gráfico de llamadas para esta función:

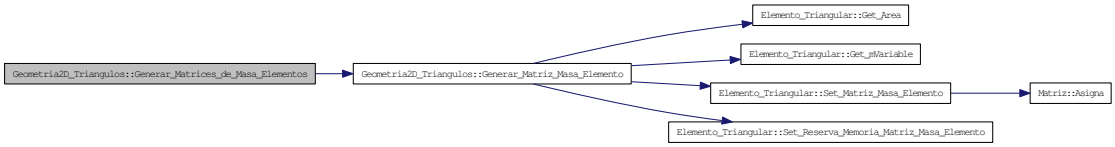


Gráfico de llamadas a esta función:

3.10.2.22. void Geometria2D_Triangulos::Generar_Matrices_de_Rigidez_-Elementos ()

Genera las matrices de rigides de todos los elementos de la geometria.

Genera las matrices de rigidez de todos los elementos.

Gráfico de llamadas para esta función:

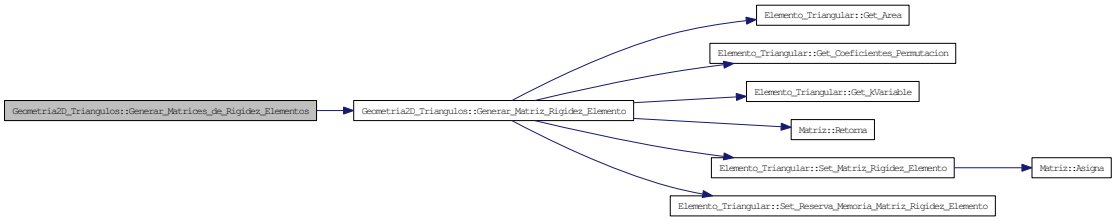
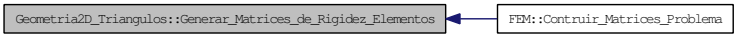


Gráfico de llamadas a esta función:



3.10.2.23. void Geometria2D_Triangulos::Generar_Matriz_Masa_Elemento (unsigned long long *id*)

Genera la matriz de masa de un elemento "id" de la geometria.

Genera la matriz de masa de un elemento finito triangular de tres nodos.

Gráfico de llamadas para esta función:

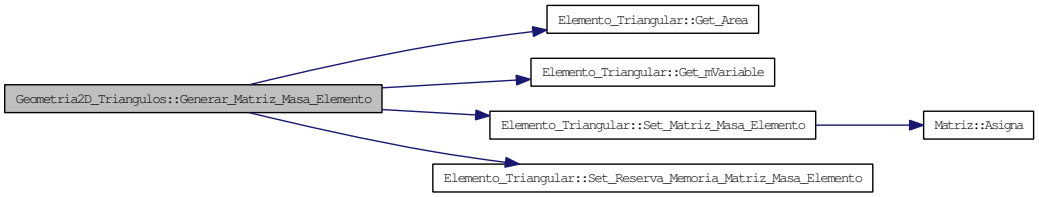
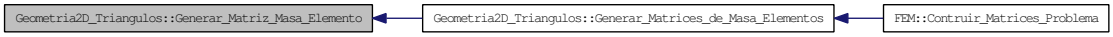


Gráfico de llamadas a esta función:



3.10.2.24. void Geometria2D_Triangulos::Generar_Matriz_-Pertenencia_Elemento (unsigned long long *id*)

Genera la matriz de pertencia del elemento "id".

Genera la matriz de pertenencia de un lemento finito triangular de tres nodos.

Gráfico de llamadas para esta función:

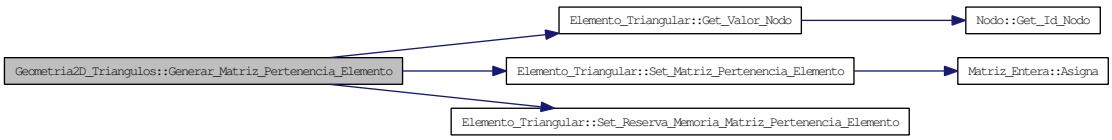


Gráfico de llamadas a esta función:

3.10.2.25. void Geometria2D_Triangulos::Generar_Matriz_Rigidez_Elemento (unsigned long long *id*)

Genera la matriz de rigidez de un elemento "id" de la geometria.

Genera la matriz de rigidez de un elemento finito trinagular de 3 nodos.

Gráfico de llamadas para esta función:

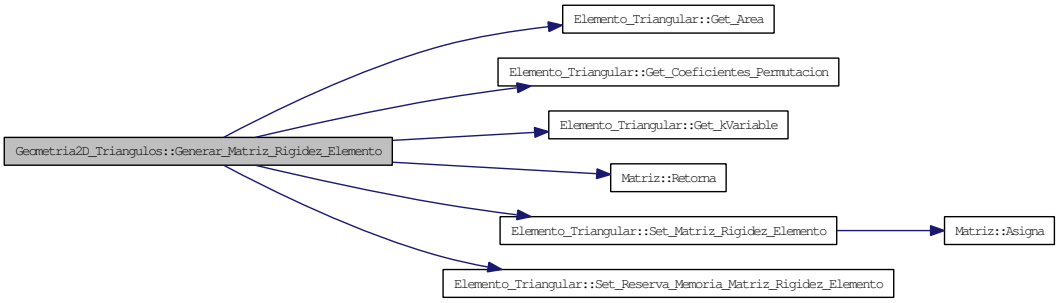
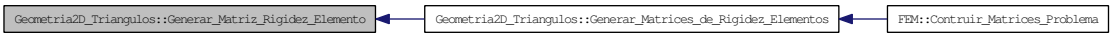


Gráfico de llamadas a esta función:



3.10.2.26. void Geometria2D_Triangulos::Generar_Vector_Carga_Elemento (unsigned long long *id*)

Calcula el vector de carga de un elemento "id".

Genera los vectores de carga del elemento id y con Distribucion de Dirac.

Gráfico de llamadas para esta función:

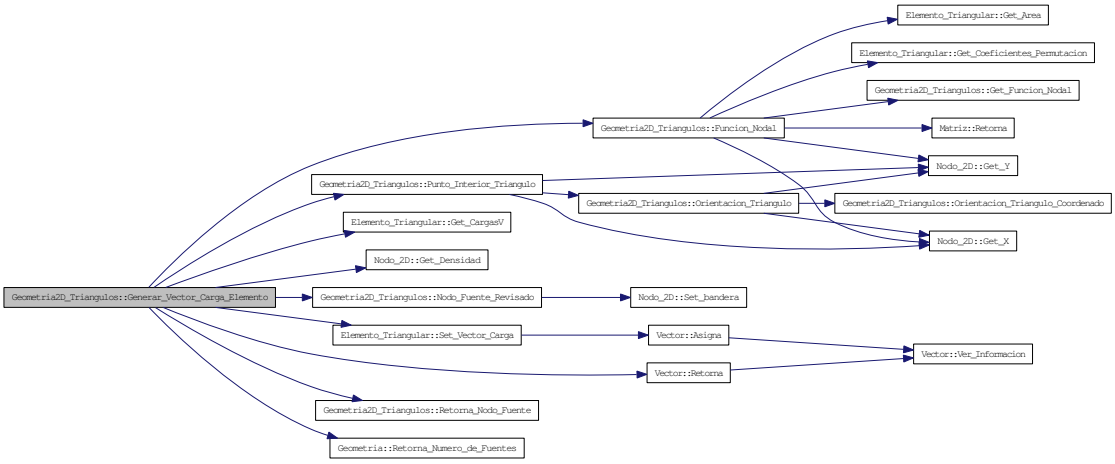
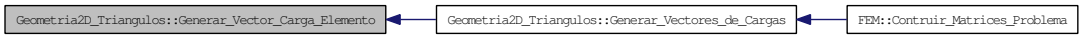


Gráfico de llamadas a esta función:



3.10.2.27. void Geometria2D_Triangulos::Generar_Vectores_de_Cargas ()

Calcula los vectores de carga de todos los elementos del dominio.

Genera los vectores de carga de todos los elementos.

Gráfico de llamadas para esta función:

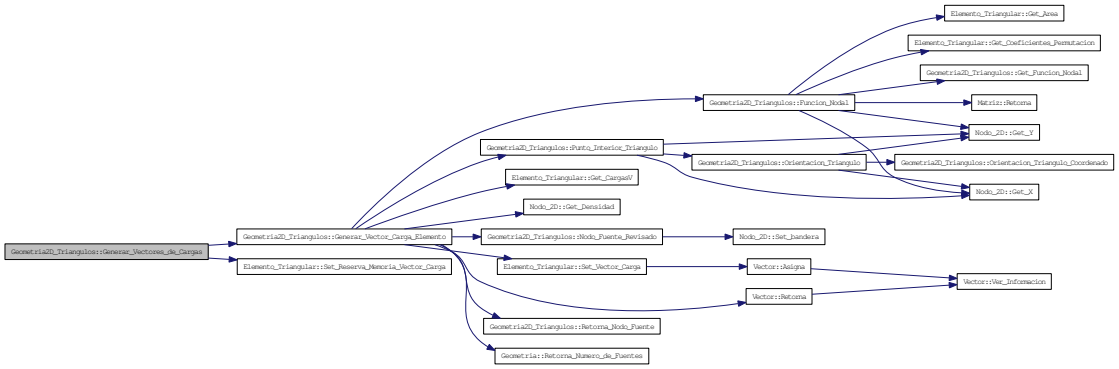


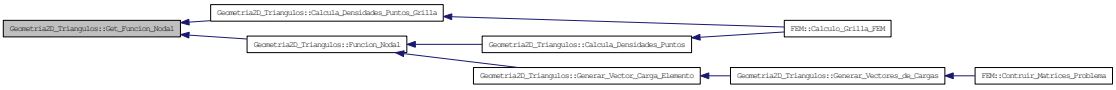
Gráfico de llamadas a esta función:

3.10.2.28. double Geometria2D_Triangulos::Get_Funcion_Nodal (double *areaE*, double *Ai*, double *Bi*, double *Ci*, double *pX*, double *pY*)

Calcula la funcion de forma con los parametros: Area del elemento, Ai, Bi y Ci (constantes geometricas del elemento) del elemento y un punto pX y pY cualquiera.

Retorna la funcion nodal de un elemento finito triangular de e nodos dado: Area, coeficientes ai,bi y ci, ademas de un punto x,y.

Gráfico de llamadas a esta función:

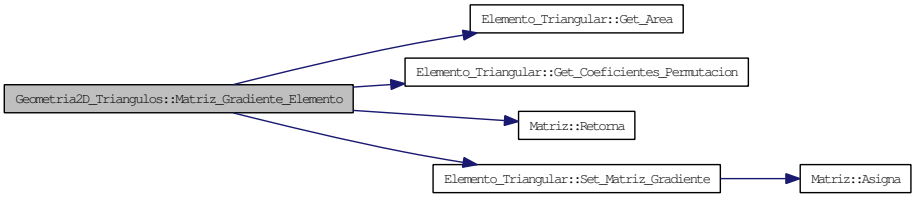


3.10.2.29. void Geometria2D_Triangulos::Matriz_Gradiente_Elemento (unsigned long long *id*, unsigned long long *nodo*)

Calcula la matriz gradiente del elemento "id" en el nodo "nodo".

Calcula la matriz gradiente de un elemento finito triangular de tres nodos.

Gráfico de llamadas para esta función:



3.10.2.30. double Geometria2D_Triangulos::Norma_Dos_Nodos (unsigned long long *id_nodo1*, unsigned long long *id_nodo2*)

Retorna la norma entre dos nodos.

Norma (distancia euclidea) entre dos nodos de la geometria.

Gráfico de llamadas para esta función:

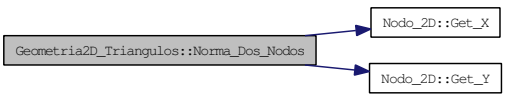
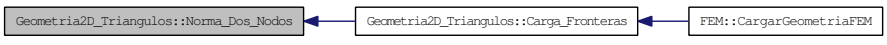


Gráfico de llamadas a esta función:

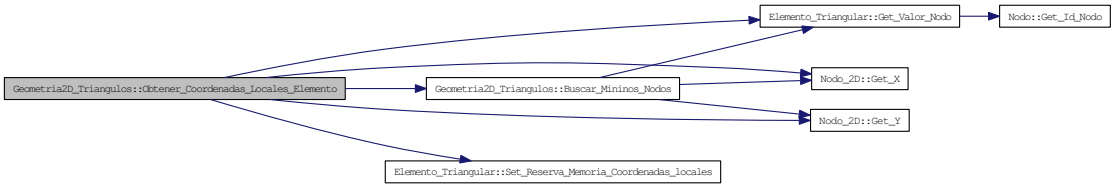


3.10.2.31. void Geometria2D_Triangulos::Obtener_-Coordenadas_Locales_Elemento (unsigned long long *id*)

Obtiene y muestra las coordenadas locales del elemento "id".

Obtiene y muestra las coordenadas locales del elemento id.

Gráfico de llamadas para esta función:

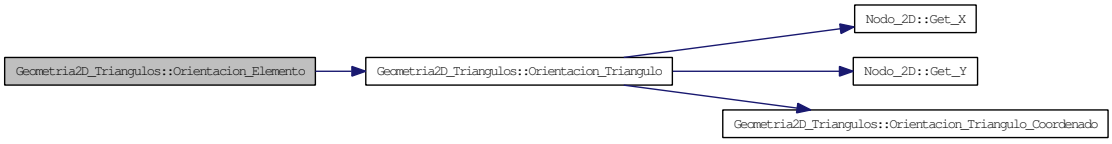


3.10.2.32. int Geometria2D_Triangulos::Orientacion_Elemento (unsigned long long *id*)

Retorna la orientacion de un elemento.

Retorna orientacion de un elemento finito triangular.

Gráfico de llamadas para esta función:



3.10.2.33. int Geometria2D_Triangulos::Orientacion_Triangulo (Nodo_2D *A1*, Nodo_2D *A2*, Nodo_2D *A3*)

Retorna la orientacion de un triangulo dado 3 Nodos.

Llamada a funcion que retorna la orientacion de un triangulo utilizando nodos.

Gráfico de llamadas para esta función:

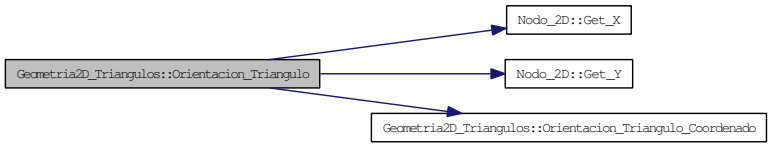
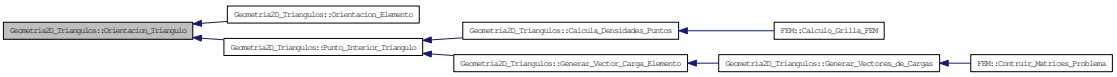


Gráfico de llamadas a esta función:



3.10.2.34. int Geometria2D_Triangulos::Orientacion_Triangulo_Coordenado (double *x1*, double *x2*, double *x3*, double *y1*, double *y2*, double *y3*)

Retorna la orientacion de un triangulo dadas las 3 coordenadas.

Retorna la orientacion de un triangulo, 1 - orientacion positiva (sentido horario) y 0 - orientacion negativa (sentido anti-horario) teniendo como entrada las coordenadas de los tres puntos de los vertices.

Gráfico de llamadas a esta función:

3.10.2.35. void Geometria2D_Triangulos::Permutacion_Elemento (unsigned long long *id*)

Calcula la permutacion en orden natural, asociada a las funciones de forma del elemento "id" de la geometria.

Calculo de permutacion nodal de un elemento finito triangular de tres nodos, donde se obtienen los coeficientes geometricos de las funciones de forma.

Gráfico de llamadas para esta función:

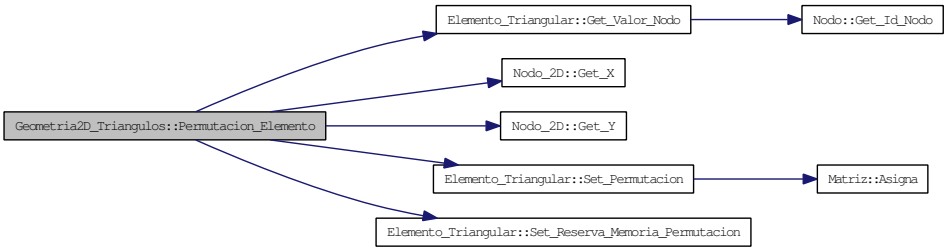
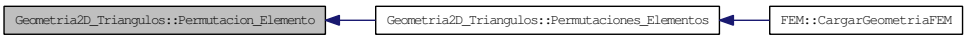


Gráfico de llamadas a esta función:



3.10.2.36. double Geometria2D_Triangulos::Permutacion_Nodal (Nodo_2D *A1*, Nodo_2D *A2*, Nodo_2D *A3*, int *coeficiente*, int *indice*)

Retorna la constante geometrica del elemento (Ai, Bi o Ci) indicada por "coeficiente" de la funcion de forma "indice" del elemento, esta es una variante del calculo nodal.

Funcion alternativa para el calculo de funciones nodales de un elemento finito triangular de tres nodos.

Gráfico de llamadas para esta función:

3.10.2.37. void Geometria2D_Triangulos::Permutaciones_Elementos ()

Calcula las permutaciones de todos los elementos de la geometria.

Calculo de permutacion de todos los elementos finitos triangulares de tres nodos.

Gráfico de llamadas para esta función:

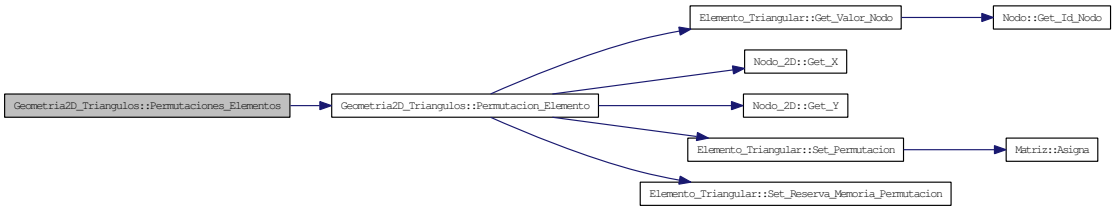


Gráfico de llamadas a esta función:

3.10.2.38. int Geometria2D_Triangulos::Punto_Interior_Triangulo (unsigned long long *id*, Nodo_2D *Punto*)

Verifica si un punto pertence al interior de un elemento finito triangular id.

Verifica si un punto del tipo Nodo_2D se encuentra en el interior de un elemento finito triangular "id".

Gráfico de llamadas para esta función:

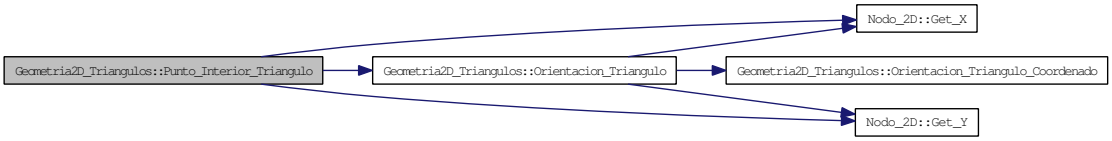
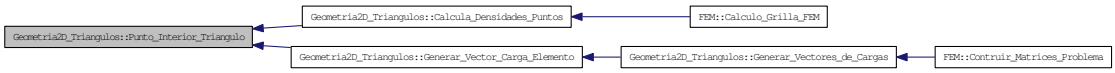


Gráfico de llamadas a esta función:

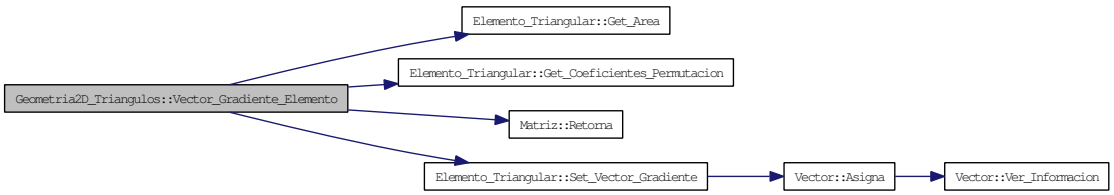


3.10.2.39. void Geometria2D_Triangulos::Vector_Gradiente_Elemento (unsigned long long *id*, unsigned long long *nodo*)

Calcula el vector gradiente del elemento "id" en el nodo "nodo".

Calculo del vector gradiente de un elemento id en un nodo.

Gráfico de llamadas para esta función:



3.10.2.40. void Geometria2D_Triangulos::Ver_Area_Elemento (unsigned long long *id*)

Muestra el area del elemento "id".

Muestra el area de un elmento triangular de tres nodos.

Gráfico de llamadas a esta función:

3.10.2.41. void Geometria2D_Triangulos::Ver_Areas_Elementos ()

Muestra las areas de todos los elementos.

Muestra todas las areas de los elementos triangulares de tres nodos.

Gráfico de llamadas para esta función:

3.10.2.42. void Geometria2D_Triangulos::Ver_Coordenadas_Elemento (unsigned long long *id*)

Muestra las coordenadas (x,y) de un elemento "id" de la geometria.

Muestra las coordenadas de un elemento finito triangular de tres nodos.

Gráfico de llamadas a esta función:

3.10.2.43. void Geometria2D_Triangulos::Ver_Coordenadas_Todos_los_Elementos ()

Muestra las coordenadas (x,y) de todos los elementos de la geometria.

Muestra las cordenadas de todos los elementos finitos triangulares de tres nodos.

Gráfico de llamadas para esta función:

3.10.2.44. void Geometria2D_Triangulos::Ver_Elemento (unsigned long long *id*)

Muestra informacion de elemento "id" de la geometria.

Muestra informacion nodal de un elemento finito triangular de tres nodos.

Gráfico de llamadas a esta función:

3.10.2.45. void Geometria2D_Triangulos::Ver_Elementos (void)

Muestra informacion de todos los elementos de la geometria.

Muestra asignacion nodal de los elementos finitos triangulares de tres nodos.

Gráfico de llamadas para esta función:

3.10.2.46. void Geometria2D_Triangulos::Ver_Nodo (unsigned long long *id*)

Muestra el nodo "id" de la geometria.

Muestra la informacion de un nodo.

Gráfico de llamadas a esta función:

3.10.2.47. void Geometria2D_Triangulos::Ver_Nodos (void)

Muestra todos los nodos de la geometria.

Muestra los nodos de la geometria.

Gráfico de llamadas para esta función:

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Geor Triangulos.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Geor Triangulos.cpp

3.11. Referencia de la Clase Matriz

Clase para el trabajar con matrices de tipo double.

`#include <Matriz.hpp>`

Métodos públicos

- Matriz ()

  Constructor de la clase.

- Matriz (unsigned long long fil, unsigned long long col)

  Constructor sobrecargado de la clase.

- ~Matriz ()

  Destructor de la clase.

- void Visualiza ()

  Muestra la matriz.

- void AsignaNombre (const char *nmb)

  Asigna nombre a la matriz.

- double ** Retorna_Puntero_Datos ()

  Retorna el puntero donde se almacena la matriz.

- char * Nombre (void)

  Retorna el nombre de la matriz.

- void SetFilas (unsigned long long fil)

  Asignar numero de filas.

- void SetColumnas (unsigned long long col)

  Asignar numero de columnas.

- unsigned long long Filas (void)

  Retorna el numero de filas.

- unsigned long long Columnas (void)

  Retorna el numero de columnas.

- void Solicita_Memoria (unsigned long long fil, unsigned long long col)

Asigna memoria.

- void Libera_Memoria ()

  Libera memoria.

- int Matriz_Cuadrada (void)

  Retorna 1 si es matriz cuadrada en caso contrario regresa 0.

- int Misma_Dimension (Matriz *a)

  Revisa si tienen la misma dimension regresando 1 en caso afirmativo y cero en caso contrario.

- void FaltaMemoria (void)

  Visualiza el error de falta de memoria para soportar la matriz.

- void Ver_Informacion (void)

  Visualiza información general de la matriz.

- void Asigna (unsigned long long fila, unsigned long long columna, const double valor)

  Asigna valor en fila, columna.

- double Retorna (unsigned long long fila, unsigned long long columna)

  Retorna valor en fila, columna.

- double Tamano (void)

  Tamaño del vector (aproximado) en Kb.

- void Inicializa (const double valor)

  Inicializa la matriz con el valor val.

- void Visualiza (const int tp)

  Visualiza la matriz.

- void Copia (Matriz *a)

  Copia el contenido de la matriz a la matriz a.

- void Suma (Matriz *a, Matriz *b)

  Suma las matrices A y B.

- void Suma (Matriz *a)

  Suma a la matriz el contenido de la matriz A.

- void Resta (Matriz *a, Matriz *b)

  Resta a la matriz A la matriz B.

- void Resta (Matriz *a)

  Resta a la matriz el contenido de la matriz A.

- void Multiplica (double escalar)

  Multiplica la matriz por el escalar ESC.

- void Multiplica (Matriz *a, Matriz *b)

  Multiplica la matriz A por la matriz B.

- void Multiplica (Matriz *a, Vector *b)

  Multiplica la matriz A por el vector B.

- void Multiplica (Vector *b, Vector *r)

  Multiplica la matriz A por el vector B dejando el Resultado en R.

- int Dominante ()

  Verifica si es estrictamente dominante la matriz.

Atributos protegidos

- double ** M

  Puntero de 2 dimensiones.

- unsigned long long Col

  Numero de columnas.

- unsigned long long Fil

  Numero de filas.

- char * Nmb

  Nombre de la matriz.

3.11.1. Descripción detallada

Clase para el trabajar con matrices de tipo double.

3.11.2. Documentación de las funciones miembro

3.11.2.1. void Matriz::AsignaNombre (const char * *nmb*)

Asigna nombre a la matriz.

Asigna un nombre a la matriz como identificador.

3.11.2.2. void Matriz::Copia (Matriz * *a*)

Copia el contenido de la matriz a la matriz a.

Copia matriz a una matriz nueva.

Gráfico de llamadas para esta función:

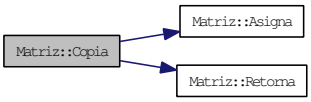


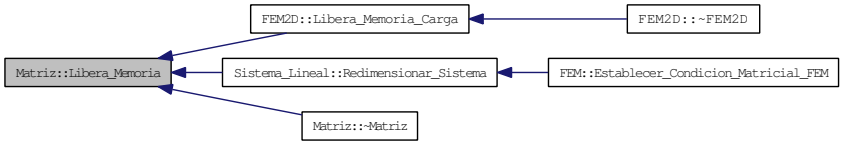
Gráfico de llamadas a esta función:

3.11.2.3. void Matriz::Libera_Memoria ()

Libera memoria.

Libera la memoria solicitada para la matriz.

Gráfico de llamadas a esta función:

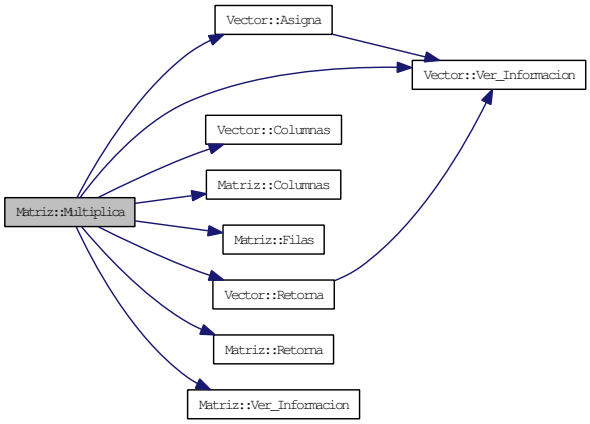


3.11.2.4. void Matriz::Multiplica (Vector * *b*, Vector * *r*)

Multiplica la matriz A por el vector B dejando el Resultado en R.

Multiplica la matriz A por el vector B dejando el Resultado en el vector R.

Gráfico de llamadas para esta función:



3.11.2.5. void Matriz::Multiplica (double *escalar*)

Multiplica la matriz por el escalar ESC.

Multiplica la matriz M por el escalar.

Gráfico de llamadas para esta función:

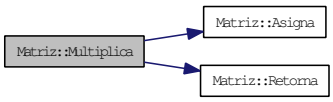
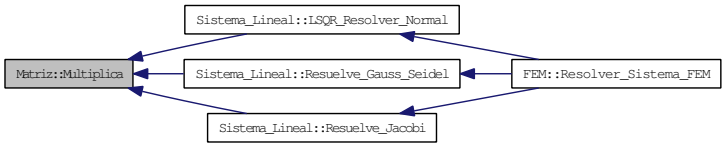


Gráfico de llamadas a esta función:

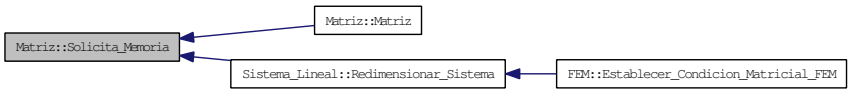


3.11.2.6. void Matriz::Solicita_Memoria (unsigned long long *fil*, unsigned long long *col*)

Asigna memoria.

Solicita memoria para la matriz.

Gráfico de llamadas a esta función:

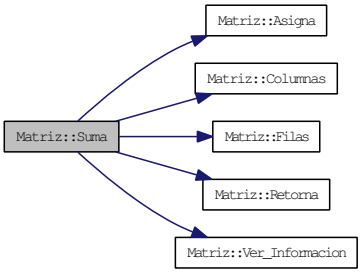


3.11.2.7. void Matriz::Suma (Matriz * *a*)

Suma a la matriz el contenido de la matriz A.

Suma a la matriz el contenido de la matriz a.

Gráfico de llamadas para esta función:

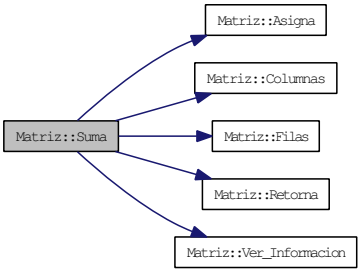


3.11.2.8. void Matriz::Suma (Matriz * *a*, Matriz * *b*)

Suma las matrices A y B.

Suma las matrices a y b.

Gráfico de llamadas para esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

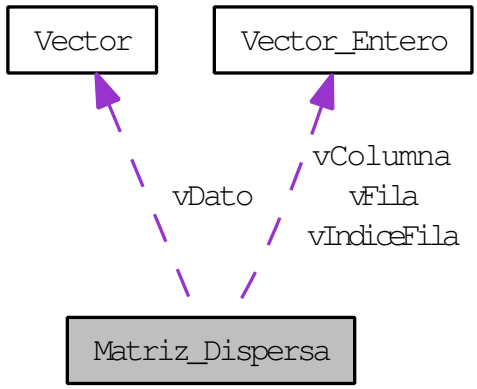
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matriz.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matriz.cpp

3.12. Referencia de la Clase Matriz_Dispersa

Clase para el trabajar con matrices dispersas de tipo double.

`#include <Matriz_Dispersa.hpp>`

Diagrama de colaboración para Matriz_Dispersa:



Métodos públicos

- Matriz_Dispersa ()

  Constructor de la clase.

- ~Matriz_Dispersa ()

  Destructor de la clase.

- Matriz_Dispersa (Vector_Entero *FILAS, Vector_Entero *COLUMNAS, Vector_Entero *INDICES, Vector *DATOS)

  Constructor sobrecargado de la clase, entrada directa de vectores Fila, Columna Indices y Datos y quedan almacenado en los mismos punteros de referencia en memoria.

- Matriz_Dispersa (Matriz *ENTRADA)

  Constructor sobrecargado de la clase con entrada Matriz y almacenamiento en formato CSR.

- Matriz_Dispersa (map< unsigned long long, double > ENTRADA, unsigned long long nFilas, map< unsigned long long, unsigned long long > mFILA, map< unsigned long long, unsigned long long > mCOLUMNA)

  Constructor sobrecargado de la clase con entrada Matriz.

- void Producto_MA_Vector (Vector *x, Vector *Sslida)

  Producto matriz dispersa por un vector.

- void Almacenar_Formato_CSR (Matriz *Entrada)

  Almacena en formato CSR de una entrada de tipo Matriz.

- void Almacenar_Formato_CSR_Hash (map< unsigned long long, double > ENTRADA, unsigned long long nFilas, map< unsigned long long, unsigned long long > mFILA, map< unsigned long long, unsigned long long > mCOLUMNA)

  Almacena en formato CSR de una entrada de tipo map (Hash).

- unsigned long long Conteo_Datos (Matriz *Entrada)

  Retorna el numero de datos de la Matriz.

- double Producto_Punto_Vectores_Disperso (double *x, double *y, unsigned long long length)

  Producto puntos vectores DIsperso.

- Vector * Producto_Matriz_Vector (Vector *x)

  Producto matriz dispersa por vector.

- Vector_Entero * Retorna_Vector_Indices ()

  Retorna Vector_Entero de indices.

- Vector_Entero * Retorna_Vector_vColumna ()

  Retorna el vector con indices de columnas.

- Vector_Entero * Retorna_Vector_vFila ()

  Retorna el vector con indices de filas.

- Vector * Retorna_Vector_vDato ()

  Retorna el vector que contiene los datos de la matriz dispersa.

Atributos públicos

- Vector_Entero * vColumna

  Vector entero que contiene datos de columnas.

- Vector_Entero * vFila

  Vector entero que contiene datos de filas.

- Vector_Entero * vIndiceFila

  Vector Entero que contiene los indices de cada nueva fila.

- Vector * vDato

  Vector flotante que contiene los datos de la matriz dispersa.

- unsigned long long nDatos

  Numero de elementos matriz no ceros.

- unsigned long long nColumnasDatos

  Dimension vector Columnas con datos.

- unsigned long long nIndices

  DImension Vector indices.

3.12.1. Descripción detallada

Clase para el trabajar con matrices dispersas de tipo double.

3.12.2. Documentación de las funciones miembro

3.12.2.1. void Matriz_Dispersa::Almacenar_Formato_CSR (Matriz * *Entrada*)

Almacena en formato CSR de una entrada de tipo Matriz.

Transforma un matriz dispera a formato de matriz CSR.

Gráfico de llamadas para esta función:

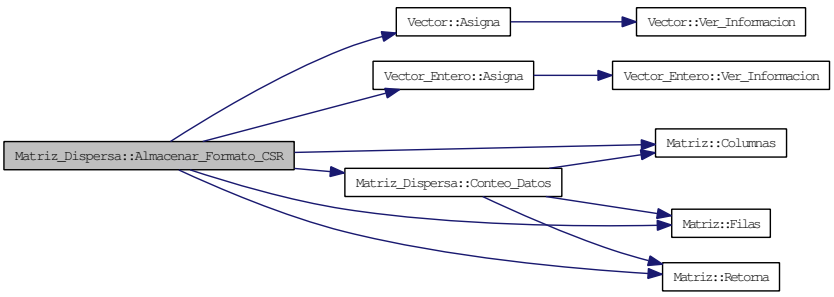


Gráfico de llamadas a esta función:

3.12.2.2. void Matriz_Dispersa::Almacenar_Formato_CSR_Hash (map< unsigned long long, double > *ENTRADA*, unsigned long long *nFilas*, map< unsigned long long, unsigned long long > *mFILA*, map< unsigned long long, unsigned long long > *mCOLUMNA*)

Almacena en formato CSR de una entrada de tipo map (Hash).

Transforma una matriz dispersa con datos de entrada map a formato de matriz CSR.

Gráfico de llamadas para esta función:

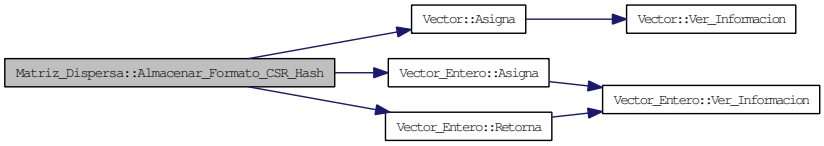


Gráfico de llamadas a esta función:

3.12.2.3. unsigned long long Matriz_Dispersa::Conteo_Datos (Matriz * *Entrada*)

Retorna el numero de datos de la Matriz.

Cuenta elementos no ceros de la matriz.

Gráfico de llamadas para esta función:

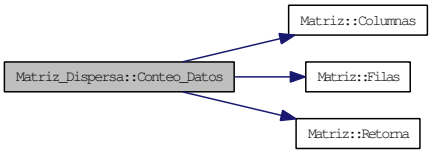
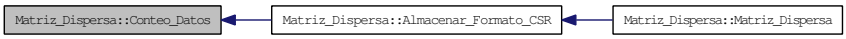


Gráfico de llamadas a esta función:



3.12.2.4. void Matriz_Dispersa::Producto_MA_Vector (Vector * *x*, Vector * *Sslida*)

Producto matriz dispersa por un vector.

Multiplica una matriz dispersa por un vector, guardando en vector salida el resultado.

Gráfico de llamadas para esta función:

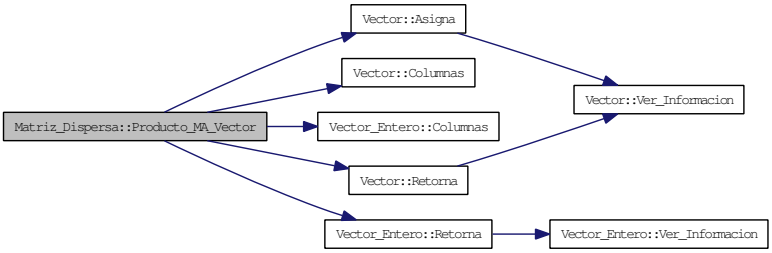
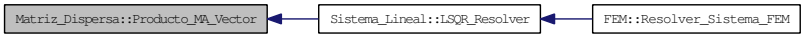


Gráfico de llamadas a esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matr Dispersa.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matr Dispersa.cpp

3.13. Referencia de la Clase Matriz_Entera

Clase para el trabajar con matrices Enteras sin signo.

`#include <Matriz_Entera.hpp>`

Métodos públicos

- Matriz_Entera ()

  Constructor de la clase.

- Matriz_Entera (unsigned long long fil, unsigned long long col)

  Constructor sobrecargado de la matriz.

- ∼Matriz_Entera ()

  Destructor de la clase.

- void Visualiza ()

  Muestra la matriz.

- void AsignaNombre (const char ∗nmb)

  Asigna nombre a la matriz.

- char ∗ Nombre (void)

  Retorna el nombre de la matriz.

- void SetFilas (unsigned long long fil)

  Asignar numero de filas.

- void SetColumnas (unsigned long long col)

  Asignar numero de columnas.

- unsigned long long Filas (void)

  Retorna el numero de filas.

- unsigned long long Columnas (void)

  Retorna el numero de columnas.

- void Solicita_Memoria (unsigned long long fil, unsigned long long col)

  Reserva memoria para la matriz.

- void Libera_Memoria ()

Libera memoria de la matriz.

- int Matriz_Cuadrada (void)

  Retorna 1 si es matriz cuadrada en caso contrario regresa 0.

- int Misma_Dimension (Matriz_Entera *a)

  Revisa si tienen la misma dimension regresando 1 en caso afirmativo y cero en caso contrario.

- void FaltaMemoria (void)

  Visualiza el error de falta de memoria para soportar la matriz.

- void Ver_Informacion (void)

  Visualiza información general de la matriz.

- void Asigna (unsigned long long fila, unsigned long long columna, unsigned long long valor)

  Asigna valor en fila, columna.

- unsigned long long Retorna (unsigned long long fila, unsigned long long columna)

  Retorna valor en fila, columna.

- void Copia (Matriz_Entera *a)

  Copia el contenido de la matriz a la matriz a.

Atributos protegidos

- unsigned long long ** M

  Puntero bidimensional que almacenara los valores de una matriz.

- unsigned long long Col

  Numero de columnas.

- unsigned long long Fil

  Numero de filas.

- char * Nmb

  Nombre de la matriz.

3.13.1. Descripción detallada

Clase para el trabajar con matrices Enteras sin signo.

3.13.2. Documentación de las funciones miembro

3.13.2.1. void Matriz_Entera::AsignaNombre (const char * *nmb*)

Asigna nombre a la matriz.

Copia el contenido de la matriz a la matriz a.

3.13.2.2. void Matriz_Entera::Solicita_Memoria (unsigned long long *fil*, unsigned long long *col*)

Reserva memoria para la matriz.

Solicita memoria para la matriz.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matriz_-Entera.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Matriz_-Entera.cpp

3.14. Referencia de la Clase Nodo

Clase utilizada para definir un nodo de un elemento finito.

`#include <Nodo.hpp>`

Heredado por Nodo_1D, Nodo_2D, y Nodo_3D.

Métodos públicos

- Nodo (void)
  Constructor de la clase Nodo.
- ~Nodo (void)
  Destructor de la clase Nodo.
- void Set_Dimension_Nodo (int dimension)
  Asigna dimension del nodo.
- void Set_Id_Nodo (int id)
  Asigna el identificador del nodo.
- int Get_Dimension_Nodo ()
  Retorna la dimension del nodo.
- unsigned long long Get_Id_Nodo ()
  Retorna el identificador del nodo.

Atributos protegidos

- unsigned long long id_nodo
  Identificador del nodo.
- int dimension_nodo
  Dimension del nodo.

3.14.1. Descripción detallada

Clase utilizada para definir un nodo de un elemento finito.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Nodo.hpp

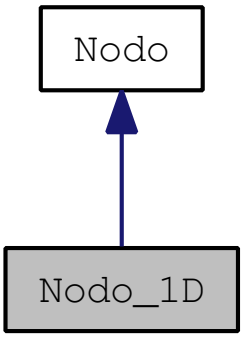
3.15. Referencia de la Clase Nodo_1D

Clase para definir un nodo unidimensional.

`#include <Nodo_1D.hpp>`

Herencias Nodo.

Diagrama de colaboración para Nodo_1D:



Métodos públicos

- Nodo_1D (void)

  Contructor de la clase.

- ~Nodo_1D (void)

  Destructor de la clase.

- void Set_X (double x)

  Asigna un valor flotante al nodo.

- double Get_X ()

  Retorna un valor del nodo.

Atributos protegidos

- double valorX

  Posicion nodal en el eje coordenado.

3.15.1. Descripción detallada

Clase para definir un nodo unidimensional.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Nodo_1D.hpp

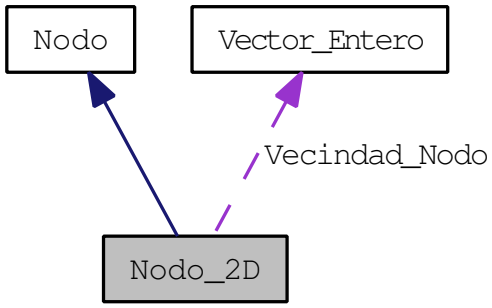
3.16. Referencia de la Clase Nodo_2D

Clase para definir un nodo bidimensional.

`#include <Nodo_2D.hpp>`

Herencias Nodo.

Diagrama de colaboración para Nodo_2D:



Métodos públicos

- Nodo_2D (void)
  Contructor de la clase.
- ~Nodo_2D (void)
  Destructor de la clase.
- void Set_bandera (int valor)
  Asigna el valor de la bandera.
- int Get_bandera ()
  Retorna el valor de la bandera.
- void Set_X (double x)
  Asigna un valor flotante en x.
- void Set_Y (double y)
  Asigna un valor flotante en y.
- void Set_pBorde (int valor)
  Asigna como nodo borde.

- void Set_vNodal (double valor)

  Asigna el valor flotante del nodo.

- void Set_gNodal (double valor)

  Asigna el valor gradiente del nodo.

- void Set_gNodal_Suma (double valor)

  Suma al gradiente un desplazamiento.

- void Set_Orden_Vecindad (int orden)

  Asigna el numero de elementos que contienen al nodo y crea un vector de largo orden.

- void Set_Vecindad (int id, int valor)

  Asigna al vector Vecindad_Nodo en posicion id un indicador de elemento contenedor de nodo (REVISAR).

- Vector_Entero * Get_Vecindad ()

  Retorna los elementos vecinos.

- int Get_Orden_Vecindad ()

  Retorna el numero de elementos que contienen al Nodo.

- int Get_pBorde ()

  Retorna 1 si pertenece a la frontera del dominio el nodo.

- double Get_X ()

  Retorna el valor de x del nodo.

- double Get_Y ()

  Retorna el valor de y del nodo.

- double Get_Densidad ()

  Retorna el valor del nodo.

- double Get_Gradiente ()

  Retorna la gradiente nodal (REVISION).

Atributos protegidos

- double valorX

  Posicion nodal en el eje x.

- double valorY

  Posicion nodal en el eje y.

- double valorNodal

  Valor del nodo.

- double gradienteNodal

  Gradiente nodal.

- int NV

  Numero de elementos que contienen al nodo.

- int pBorde

  Indicador de nodo frontera donde es 1 si es nodo frontera.

- Vector_Entero * Vecindad_Nodo

  Vector que contiene a los elementos que contienen al nodo.

- int bandera_nodo

  Bandera usada para marcar un nodo ya calculado, util para el calculo de fuentes que comparten un mismo nodo.

3.16.1. Descripción detallada

Clase para definir un nodo bidimensional.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Nodo 2D.hpp

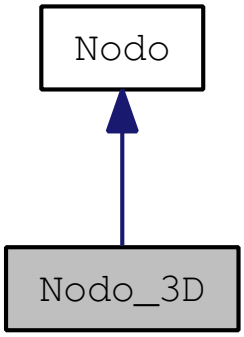
3.17. Referencia de la Clase Nodo_3D

Clase para definir un nodo tridimensional.

`#include <Nodo_3D.hpp>`

Herencias Nodo.

Diagrama de colaboración para Nodo_3D:



Métodos públicos

- Nodo_3D (void)

  Contructor de la clase.

- ~Nodo_3D (void)

  Destructor de la clase.

- void Set_X (double x)

  Asigan un valor de posicion nodal en el eje x.

- void Set_Y (double y)

  Asigan un valor de posicion nodal en el eje y.

- void Set_Z (double z)

  Asigan un valor de posicion nodal en el eje z.

- double Get_X ()

  Retorna el valor de x.

- double Get_Y ()

Retorna el valor de y.

- double Get_Z ()

  Retorna el valor de z.

Atributos protegidos

- double valorX

  Posicion nodal en el eje x.

- double valorY

  Posicion nodal en el eje y.

- double valorZ

  Posicion nodal en el eje z.

3.17.1. Descripción detallada

Clase para definir un nodo tridimensional.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Nodo 3D.hpp

3.18. Referencia de la Clase Problema

Clase que permite definir problemas a solucionar con elementos finitos - Esta es una clase de enlace a futuros problemas a resolver con el metodo FEM.

`#include <Problema.hpp>`

Heredado por ETRAD.

Métodos públicos

- Problema (void)

  Constructor de la clase.

- ~Problema ()

  Destructor de la clase.

Atributos protegidos

- int tipo_problema

  Tipo de problema, 1-Eliptico, 2-Parabolico, 3 - Hiperbolico etc.....

- int tipo_derivacion

  Tipo derivacion de la ecuacion diferencial a resolver, Borde Vectorial 1 - matricial 2.

- int tipo_contribucion_carga

  Tipo constribucion de vector de cargas, Carga Nula 0, Carga constante 1, Carga tipo fuente (Funcion de distribucion de Dirac) 2.

3.18.1. Descripción detallada

Clase que permite definir problemas a solucionar con elementos finitos - Esta es una clase de enlace a futuros problemas a resolver con el metodo FEM.

La documentación para esta clase fue generada a partir del siguiente fichero:

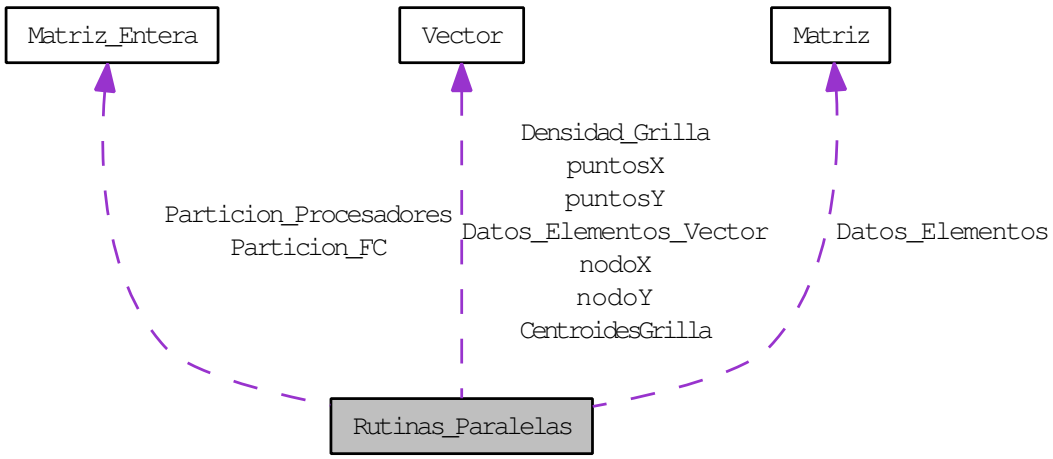
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Problema.hpp

3.19. Referencia de la Clase Rutinas_Paralelas

Clase para trabajar con rutinas paralelas en almacenamiento y calculos.

`#include <Rutinas_Paralelas.hpp>`

Diagrama de colaboración para Rutinas_Paralelas:



Métodos públicos

- Rutinas_Paralelas ()

  Costructor de la clase.

- Rutinas_Paralelas (int Procesadores)

  Costructor sobre cargado de la clase.

- ~Rutinas_Paralelas ()

  Destructor de la clase.

- Matriz * Retorna_Datos_Elementos (void)

  Retorna la matriz que contiene todo los datos de los elementos (id elemento, area, coeficientes de permutacion, aproximaciones nodales y lista de nodos).

- Matriz_Entera * Retorna_FC (void)

  Retorna la matriz que almacena las filas y columnas de las particiones.

- Matriz_Entera * Retorna_MP (void)

Retorna la matriz con los indices de rank de la matriz de particion fila columna.

- Vector * Retorna_Centroides (void)

  Retorna en un vector los centroides de las particiones de la grilla en paralelo.

- Vector * Retorna_PuntosX (void)

  Retorna en un vector los valores X de los puntos de la grilla.

- Vector * Retorna_PuntosY (void)

  Retorna en un vector los valores Y de los puntos de la grilla.

- Vector * Retorna_NodosX (void)

  Retorna en un vector los valores de X de los nodos de la geometria original evaluada en a FEM.

- Vector * Retorna_NodosY (void)

  Retorna en un vector los valores de Y de los nodos de la geometria original evaluada en la FEM.

- Vector * Retorna_Datos_Elementos_Vector (void)

  Retorna el vector de los datos que continenen toda la informacion del calculo FEM.

- void Almacena_Matriz_FC (Matriz_Entera *entrada)

  Almacena la matriz de particiones a partir de una matriz de entrada.

- void Almacena_Matriz_MP (Matriz_Entera *entrada)

  Almacena la matriz de indices de procesadores a partir de un matriz de entrada.

- void Reserva_Memoria_Centroides (unsigned long long dim)

  Reserva memoria para los centroides de la grilla particionada.

- void Reserva_Memoria_Puntos (unsigned long long nNodos)

  Reserva memoria para los puntos de la grilla.

- void Reserva_Memoria_Nodos (unsigned long long nNodos)

  Reserva memoria para la informacion de los en de x e y en los nodos de la FEM.

- void Reserva_Memoria_Datos_Elementos_Vector (unsigned long long nElementos)

  Reserva memoria para los datos de todo el calculo realizado por la aproximacion original de la FEM.

- void Almacenar_Nodos_Geometria (Nodo_2D *GeometriaFEM, unsigned long long nNodos)

  Almacena en punteros los nodos de la geometria FEM para distribucion de informacion entre procesadores.

- void Almacenar_Nodos_Grilla (Nodo_2D *GrillaFEM, unsigned long long nNodos)

  Almacena en punteros los nodos de la grilla para distribucion de informacion entre procesadores.

- void Almacenar_Datos_Elemento (unsigned long long id_fila, unsigned long long id_elemento, unsigned long long nodo1, unsigned long long nodo2, unsigned long long nodo3, double area, double fi1, double fi2, double fi3, Matriz *Coeficientes_PermutacionEL)

  Almacena en una fila especifica de la matriz Datos_Elementos (id elemento, area, coeficientes de permutacion, aproximaciones nodales y lista de nodos).

- void SetProcesadores (int Procesadores)

  Almacena el numero de procesos.

- int GetProcesadores ()

  Retorna el numero de procesos.

- void Producto_Matriz_Dispersa_Vector (Vector *x, Vector *Salida, Vector_Entero *INDICES, Vector_Entero *COLUMNAS, Vector *DATOS)

  Multiplica una matriz dispersa por un vector, guardando en vector salida el resultado.

- void Producto_MatrizT_Dispersa_Vector (Vector *x, Vector *Salida, Vector_Entero *INDICES, Vector_Entero *COLUMNAS, Vector *DATOS, unsigned long long dim, unsigned long long inicio)

  Multiplica una matriz traspuesta dispersa por un vector, guardando en vector salida el resultado.

- void Producto_Matriz_Dispersa_Vector_Seccion (Vector *x, Vector *Salida, Vector_Entero *INDICES, Vector_Entero *COLUMNAS, Vector *DATOS, unsigned long long dim, unsigned long long inicio)

  Multiplica una matriz dispersa por un vector, guardando en una seccion el vector salida el resultado.

- void Producto_Matriz_Dispersa_Vector_Seccion_LSQR (Vector *x, Vector *Salida, Vector_Entero *INDICES, Vector_Entero *COLUMNAS, Vector *DATOS, unsigned long long dim, unsigned long long inicio, double normaIn, double normaOut, Vector *u)

  Multiplica una matriz dispersa por un vector, guardando en una seccion el vector salida el resultado.

- void Obtener_Vector_Particion (int size, unsigned long long orden, Vector_-Entero *vParticion)

  Particiona un vector segun numero de procesadores, almacenando el largo de cada sub vector.

- void Envio_Vector_Entero (int procesador_destino, int TAG, unsigned long long posicion_indice, unsigned long long largo, unsigned long long *vector)

  Envia un mensaje que contiene un vector entero.

- void Envio_Vector_Double (int procesador_destino, int TAG, unsigned long long posicion_indice, unsigned long long largo, double *vector)

  Envia un mensaje que contiene un vector flotante.

- void Envio_Dato_Double (int procesador_destino, int TAG, double Dato)

  Envia un mensaje que contiene un dato flotante.

- void Envio_Dato_Entero (int procesador_destino, int TAG, unsigned long long Dato)

  Envia un mensaje que contiene un dato entero.

- void Recibe_Dato_Double (int procesador_origen, int TAG, double *con)

  Recibe un mensaje que contiene un dato flotante.

- void Recibe_Dato_Entero (int procesador_origen, int TAG, unsigned long long *con)

  Recibe un mensaje que contiene un dato entero.

- void Recibe_Vector_Entero (int procesador_origen, int TAG, unsigned long long largo, unsigned long long *vector, MPI_Status status)

  Recibe un vector de tipo entero long long sin signo.

- void Recibe_Vector_Double (int procesador_origen, int TAG, unsigned long long largo, double *vector, MPI_Status status)

  Recibe un vector de tipo double.

- void Recibe_Vector_Double_Puntero (int procesador_origen, int TAG, unsigned long long posicion_indice, unsigned long long largo, double *vector, MPI_Status status)

  Recibe un vector de tipo double indicando la posicion del indicie desde donde se recibira la informacion.

Atributos públicos

- unsigned long long nDesconocido

  Numero de valores desconocidos de la grilla.

- int nProcesadores

  Numero de procesadores a utilizar.

- Matriz * Datos_Elementos

  Matriz que contiene todo los datos de los elementos (id elemento, area, coeficientes de permutacion, aproximaciones nodales y lista de nodos).

- Vector * **Datos_Elementos_Vector**
- Vector * puntosX

  Vector que contiene datos de las coordenadas X para el calculo paralelo de la grilla FEM.

- Vector * puntosY

  Vector que contiene datos de las ordenadas Y para el calculo paralelo de la grilla FEM.

- Vector * Densidad_Grilla

  Vector que contiene las densidades calculadas en el proceso paralelo del calculo de la grilla.

- Vector * nodoX

  Vector que contiene datos de los nodos en X del dominio FEM.

- Vector * nodoY

  Vector que contiene datos de los nodos en Y del dominio FEM.

- Vector * CentroidesGrilla

  Vector que contiene los centroides de la grilla.

- Matriz_Entera * Particion_FC

  Matriz que contiene informacion sobre la particion paralalela de un matriz.

- Matriz_Entera * Particion_Procesadores

  Matriz que continene informacion sobre indices de procesadores.

- double tFEM

  Tiempo utilizado en comunicacion aproxiamcion FEM.

- double tGrilla

  Tiempo utilizado en comunicacion calculo Grilla.

3.19.1. Descripción detallada

Clase para trabajar con rutinas paralelas en almacenamiento y calculos.

3.19.2. Documentación de las funciones miembro

3.19.2.1. void Rutinas_Paralelas::Envio_Dato_Double (int *procesador_destino*, int *TAG*, double *Dato*)

Envia un mensaje que contiene un dato flotante.

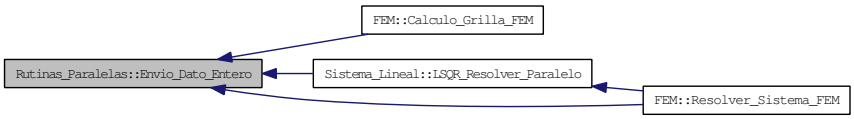
Envia un vector de tipo double.

3.19.2.2. void Rutinas_Paralelas::Envio_Dato_Entero (int *procesador_destino*, int *TAG*, unsigned long long *Dato*)

Envia un mensaje que contiene un dato entero.

Envia un dato de tipo entero long long sin signo.

Gráfico de llamadas a esta función:

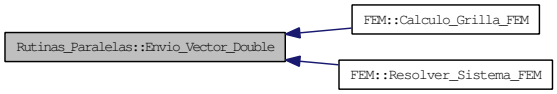


3.19.2.3. void Rutinas_Paralelas::Envio_Vector_Double (int *procesador_destino*, int *TAG*, unsigned long long *posicion_indice*, unsigned long long *largo*, double * *vector*)

Envia un mensaje que contiene un vector flotante.

Recibe un vector entero long long sin signo.

Gráfico de llamadas a esta función:



3.19.2.4. void Rutinas_Paralelas::Envio_Vector_Entero (int *procesador_destino*, int *TAG*, unsigned long long *posicion_indice*, unsigned long long *largo*, unsigned long long * *vector*)

Envia un mensaje que contiene un vector entero.

Envia un vector entero long long sin signo.

Gráfico de llamadas a esta función:

3.19.2.5. void Rutinas_Paralelas::Producto_Matriz_Dispersa_Vector_Seccion (Vector * *x*, Vector * *Salida*, Vector_Entero * *INDICES*, Vector_Entero * *COLUMNAS*, Vector * *DATOS*, unsigned long long *dim*, unsigned long long *inicio*)

Multiplica una matriz dispersa por un vector, guardando en una seccion el vector salida el resultado.

Multiplica una matriz dispersa por una seccion de un vector guardando en vector salida el resultado.

Gráfico de llamadas para esta función:

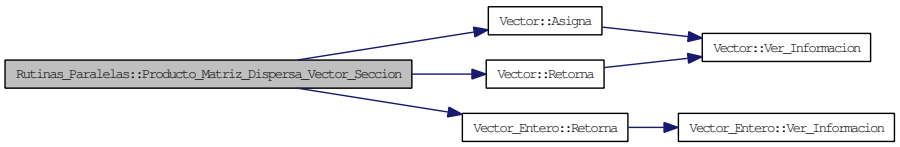
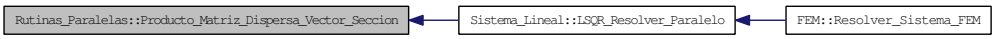


Gráfico de llamadas a esta función:

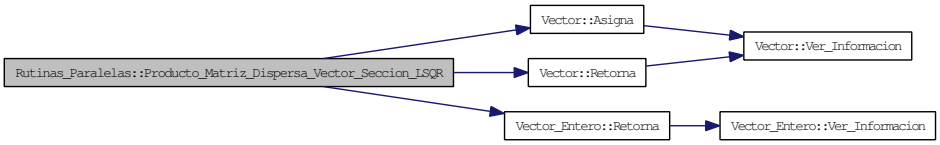


3.19.2.6. void Rutinas_Paralelas::Producto_Matriz_Dispersa_Vector_Seccion_LSQR (Vector * *x*, Vector * *Salida*, Vector_Entero * *INDICES*, Vector_Entero * *COLUMNAS*, Vector * *DATOS*, unsigned long long *dim*, unsigned long long *inicio*, double *normaIn*, double *normaOut*, Vector * *u*)

Multiplica una matriz dispersa por un vector, guardando en una seccion el vector salida el resultado.

Multiplica una matriz dispersa por un vector, guardando en vector salida el resultado.

Gráfico de llamadas para esta función:



3.19.2.7. void Rutinas_Paralelas::Producto_MatrizT_Dispersa_Vector (Vector * *x*, Vector * *Salida*, Vector_Entero * *INDICES*, Vector_Entero * *COLUMNAS*, Vector * *DATOS*, unsigned long long *dim*, unsigned long long *inicio*)

Multiplica una matriz traspuesta dispersa por un vector, guardando en vector salida el resultado.

Multiplica una matriz dispersa traspuesta por un vector guardando en vector salida el resultado.

Gráfico de llamadas para esta función:

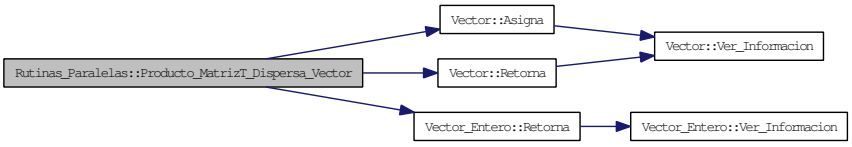
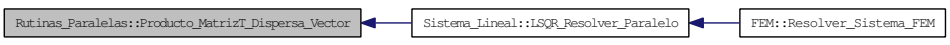


Gráfico de llamadas a esta función:



3.19.2.8. void Rutinas_Paralelas::Recibe_Dato_Double (int *procesador_origen*, int *TAG*, double * *con*)

Recibe un mensaje que contiene un dato flotante.

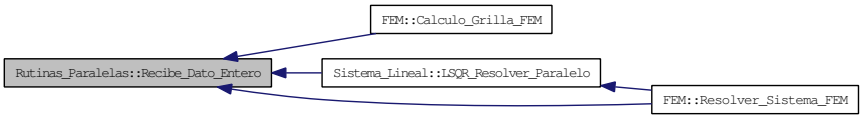
Recibe un dato de tipo double.

3.19.2.9. void Rutinas_Paralelas::Recibe_Dato_Entero (int *procesador_origen*, int *TAG*, unsigned long long * *con*)

Recibe un mensaje que contiene un dato entero.

Recibe un dato de tipo entero long long sin signo.

Gráfico de llamadas a esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

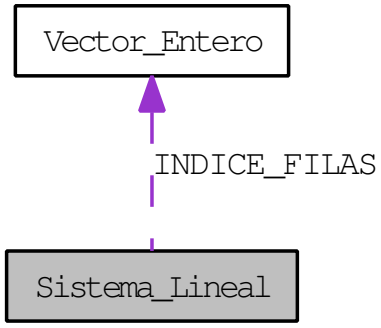
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Rutin Paralelas.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Rutin Paralelas.cpp

3.20. Referencia de la Clase Sistema_Lineal

Clase para trabajar con sistema lineales.

`#include <Sistema_Lineal.hpp>`

Diagrama de colaboración para Sistema_Lineal:



Métodos públicos

- Sistema_Lineal ()

  Costructor de la clase.

- **Sistema_Lineal** (int traza)
- ~Sistema_Lineal ()

  Destructor de la clase.

- void Set_Traza (int traza)

  Asigna numero de iteraciones.

- int Get_Traza ()

  Asigna numero de iteraciones.

- unsigned long long Get_Iteraciones ()

  Asigna numero de iteraciones.

- void Set_Iteraciones (int Iter)

  Asigna numero de procesos - procesadores.

- void SetNP (int procesos)

  Asigna numero de procesos - procesadores.

- int GetNP ()

  Retorna numero de procesos - procesadores.

- void **Asigna_Indice_FIla** (Vector_Entero *ENTRADA)
- unsigned long long Get_nDesconocido ()

  Retorna el numero de incognitas desconocidas.

- void Set_nDesconocido (unsigned long long variables_X)

  Asigna el numeros de incognitas del sistema.

- void **LSRQ_Matriz_Dispersa** (Matriz *A, Matriz_Dispersa *MDA)
- void LSQR_Resolver_Paralelo (Vector *x, Vector *b, unsigned long long *iParticion, Vector_Entero *COLUMNAS, Vector_Entero *INDICES, Vector *DATOS, int Tipo_Parada, int traza)

  Algoritmo LSQR paralelo para matrices dispersas.

- void LSQR_Resolver (Matriz_Dispersa *MDA, Vector *x, Vector *b, int Tipo_Parada, int traza)

  Algoritmo LSQR para matrices dispersas.

- void LSQR_Resolver_Normal (Matriz *A, Vector *x, Vector *b, int Tipo_Parada, int traza)

  Algoritmo LSQR.

- void Redimensionar_Sistema (Matriz *a, Vector *b, Vector_Entero *frontera, unsigned long long DimFrontera)

  Redimensiona un sistema de ecuaciones representados matricialmente, indicando que filas-columnas se desean eliminar.

- void Redimensionar_Sistema_Hashing (map< unsigned long long, double > A_FEM, map< unsigned long long, unsigned long long > mFila, map< unsigned long long, unsigned long long > mColumna, Vector *b, Vector_Entero *frontera, unsigned long DimActual)

  Redimensiona un sistema de ecuaciones representados matricialmente con Hashing , indicando que filas-columnas se desean eliminar.

- void Resuelve_Jacobi (Matriz *A, Vector *x, Vector *b, double ep, int Tipo_Parada, int traza)

  Resuelve Ax=b usando el metodo Jacobi.

- void Resuelve_Gauss_Seidel (Matriz *a, Vector *x, Vector *b, double ep, int Tipo_Parada, int traza)

Resuelve Ax=b usando el metodo Gauss Seidel.

- void Resuelve_Factorizacion_LU (Matriz *a, Vector *x, Vector *b)
  Resuelve el sistema Ax=b usando factorizacion LU.
- void Factoriza_LU (Matriz *a)
  Factoriza la matriz A en L y U dejandolas en la misma matriz.
- void Solve_LU (Matriz *a, Vector *x, Vector *b)
  Resuelve el sistema Ax=LUx = b.
- void Solve_L (Matriz *a, Vector *y, Vector *b)
  Resuelve el sistema LY=B, dados L y B.
- void Solve_LLT (Matriz *a, Vector *x, Vector *b)
  Resuelve un sistema factorizado utilizando Cholesky LLT.
- void Solve_U (Matriz *a, Vector *x, Vector *y)
  Resuelve el sistema UX=Y, dados U y Y.
- void Factoriza_LLT (Matriz *a)
  Factoriza la matriz A en L y LT dejandolas en la misma matriz.

Atributos públicos

- unsigned long long nDesconocido
  Numero de variables desconocidas.
- int NP
  Numero de procesos.
- unsigned long long Iteraciones
  Numero de iteraciones para uso en metodos iterativos.
- int Traza
  Indica si se desea alamcenar en archivo la convergencia del error.
- Vector_Entero * INDICE_FILAS
  Usado en el calculo paralelo para indicar fila inicial global.
- double tSistema
  Tiempo utilizado en resolver sistema.

3.20.1. Descripción detallada

Clase para trabajar con sistema lineales.

3.20.2. Documentación de las funciones miembro

3.20.2.1. void Sistema_Lineal::Factoriza_LLT (Matriz * *a*)

Factoriza la matriz A en L y LT dejandolas en la misma matriz.

Calcula la factorizacion LLT.

Gráfico de llamadas para esta función:

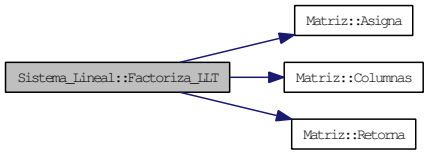


Gráfico de llamadas a esta función:

3.20.2.2. void Sistema_Lineal::LSQR_Resolver (Matriz_Dispersa * *MDA*, Vector * *x*, Vector * *b*, int *Tipo_Parada*, int *traza*)

Algoritmo LSQR para matrices dispersas.

LSQR para un procesador con matrices dispersas.

Gráfico de llamadas para esta función:

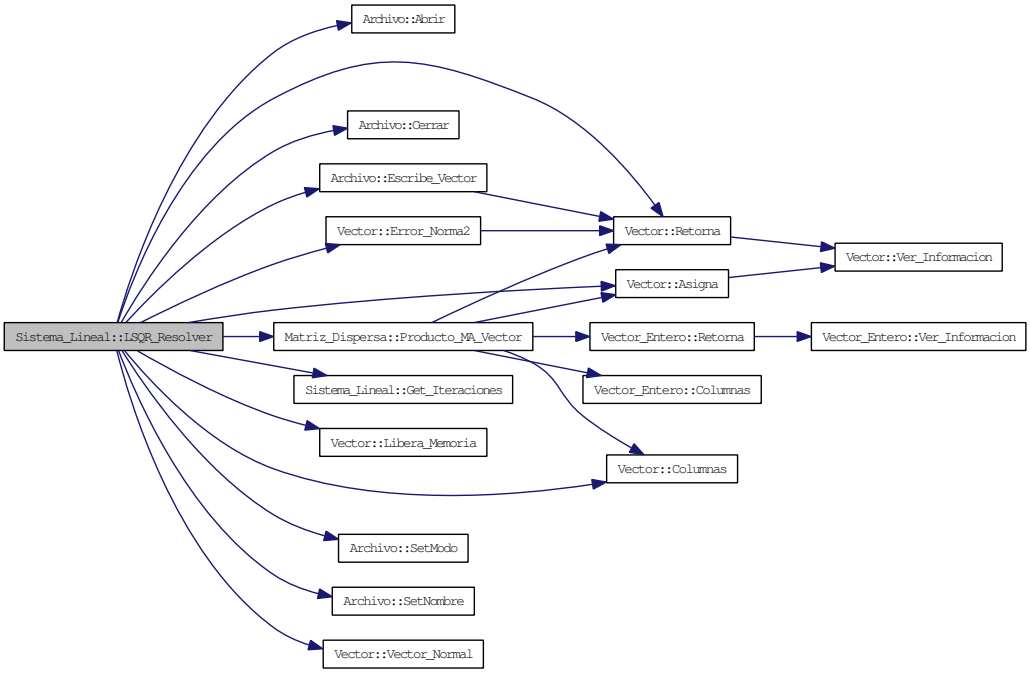


Gráfico de llamadas a esta función:

3.20.2.3. void Sistema_Lineal::LSQR_Resolver_Normal (Matriz * *A*, Vector * *x*, Vector * *b*, int *Tipo_Parada*, int *traza*)

Algoritmo LSQR.

LSQR con matriz tradicional.

Gráfico de llamadas para esta función:

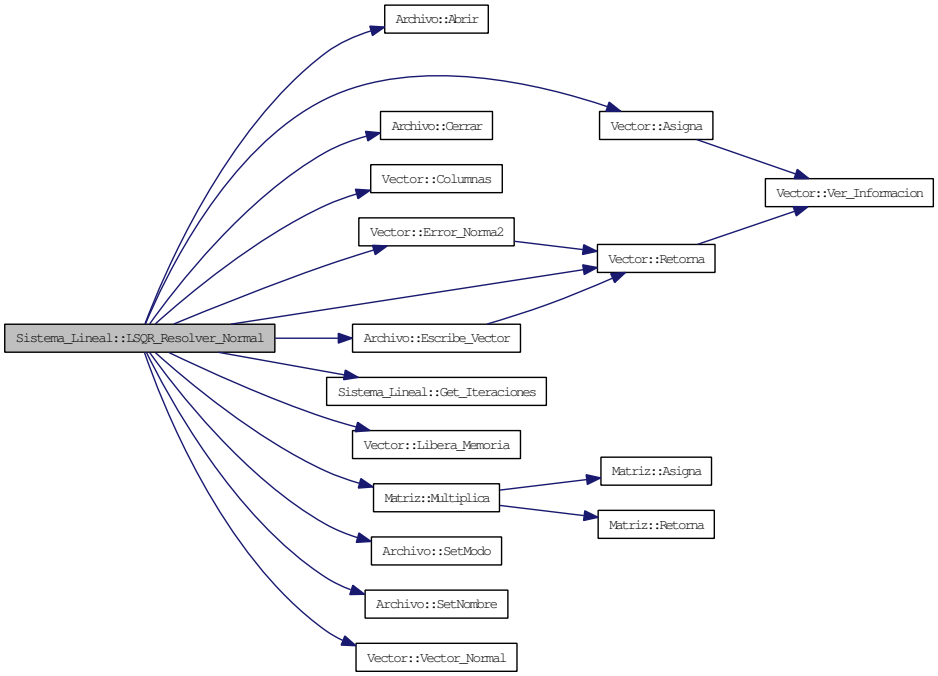


Gráfico de llamadas a esta función:

3.20.2.4. void Sistema_Lineal::LSQR_Resolver_Paralelo (Vector * *x*, Vector * *b*, unsigned long long * *iParticion*, Vector_Entero * *COLUMNAS*, Vector_Entero * *INDICES*, Vector * *DATOS*, int *Tipo_Parada*, int *traza*)

Algoritmo LSQR paralelo para matrices dispersas.

Muestra cada 100 iteraciones un contador en pantalla...

Gráfico de llamadas para esta función:

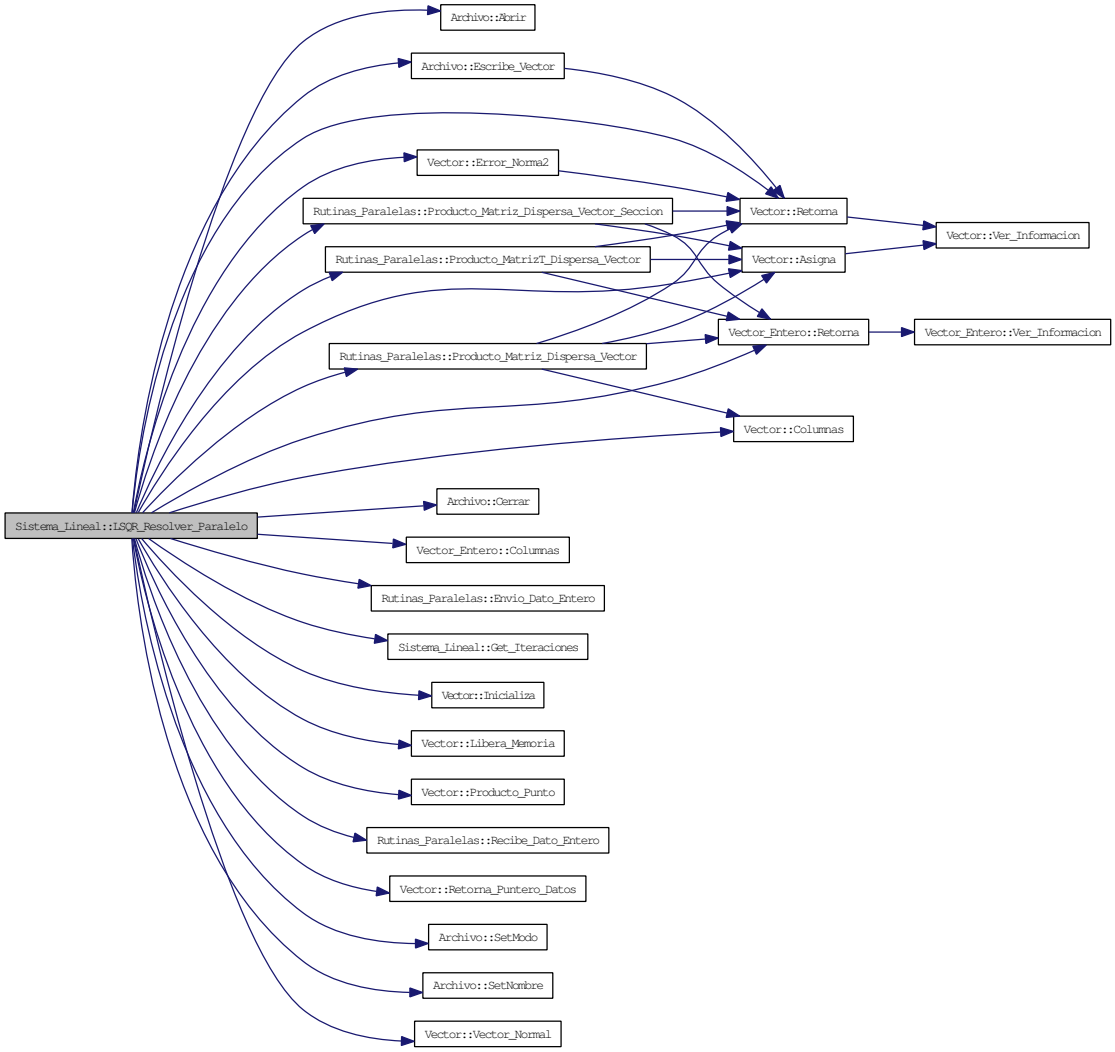


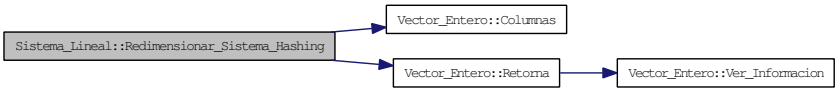
Gráfico de llamadas a esta función:

3.20.2.5. void Sistema_Lineal::Redimensionar_Sistema_Hashing (map< unsigned long long, double > *A_FEM*, map< unsigned long long, unsigned long long > *mFila*, map< unsigned long long, unsigned long long > *mColumna*, Vector * *b*, Vector_Entero * *frontera*, unsigned long *DimActual*)

Redimensiona un sistema de ecuaciones representados matricialmente con Hashing , indicando que filas-columnas se desean eliminar.

Elimina la informacion de tipo map de columnas, filas y datos, a partir del vector de entrada donde se indica que filas - columnas se quieren eliminar de la matriz

Gráfico de llamadas para esta función:



3.20.2.6. void Sistema_Lineal::Resuelve_Jacobi (Matriz * *A*, Vector * *x*, Vector * *b*, double *ep*, int *Tipo_Parada*, int *traza*)

Resuelve Ax=b usando el metodo Jacobi.

Resuelve un sistema de ecuaciones utilizando el metodo Jacobi.

Gráfico de llamadas para esta función:

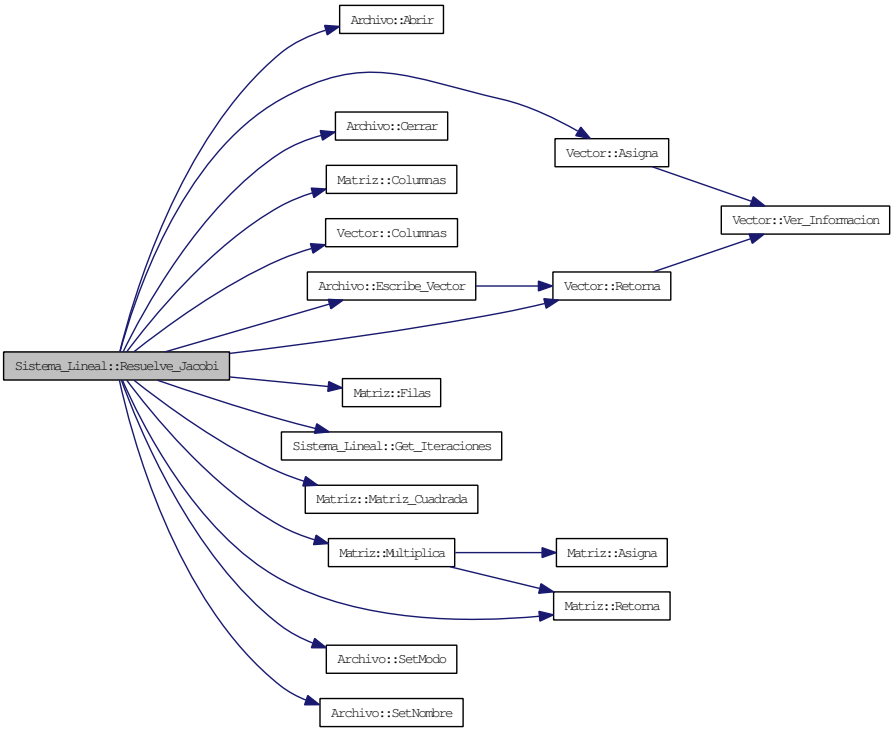


Gráfico de llamadas a esta función:

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Sistema_-Lineal.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Sistema_-Lineal.cpp

3.21. Referencia de la Clase Vector

Clase para el trabajar con vectores de tipo double.

`#include <Vector.hpp>`

Métodos públicos

- Vector (void)
  Constructor de la clase.
- Vector (unsigned long long col)
  Constructor sobrecargado de la clase.
- Vector (unsigned long long col, const double val)
  Constructor sobrecargado de la clase.
- Vector (unsigned long long col, char ∗nmb)
  Constructor sobrecargado de la clase.
- ∼Vector ()
  Destructor de la clase.
- void Libera_Memoria ()
  Libera memoria.
- double ∗ Retorna_Puntero_Datos ()
  Retorna puntero del vector.
- void Solicita_Memoria (unsigned long long col, const double val)
  Solicita la memoria para el vector.
- void AsignaNombre (const char ∗nmb)
  Asigna un nombre al vector.
- char ∗ Nombre (void)
  Retorna el nombre al vector.
- unsigned long long Columnas (void)
  Retorna el numero de columnas.
- void Inicializa (double val)

Inicializa con valores val el vector.

- void Asigna (unsigned long long col, double val)

  Asigna valor de una elemento del vector en columna col.

- double Retorna (unsigned long long col)

  Retorna valor de la columna col del vector.

- void Visualiza ()

  Visualiza el vector.

- void Copia (Vector *a)

  Copia el vector a otro vector a.

- void Convierte (double *a, unsigned long long tam)

  Convierte el arreglo de tamaño tam a vector.

- void Suma (Vector *a, Vector *b)

  Suma los vectores a y b.

- void Suma (Vector *a)

  Suma al vector a.

- void Resta (Vector *a, Vector *b)

  Resta los vectores a menos b.

- void Resta (Vector *a)

  Resta el vector a.

- void Multiplica (double esc)

  Multiplica el vector por el escalar esc.

- double Norma_inf (void)

  Calcula la norma infinito.

- double Producto_Punto ()

  Calcula el producto punto.

- double Producto_Punto (Vector *a)

  Calcula el producto punto.

- double Vector_Normal ()

Calcula el vector Normal.

- double Error_Norma2 (Vector *a, Vector *b)

  Obtiene el error norma 2 entre dos vectores.

- double Tamano (void)

  Tamaño del vector (aproximado) en Kb.

- double Sumatoria_Vector ()

  Sumatoria elementos del vector.

- void Ver_Informacion (void)

  Visualiza información general de la matriz.

- unsigned long long Conteo_Datos ()

  Cuenta el numero de elementos no ceros de un vector.

Atributos protegidos

- unsigned long long Columna

  Numero de columnas.

- char * Nmb

  Nombre del vector.

- double * V

  Puntero al Vector flotante.

3.21.1. Descripción detallada

Clase para el trabajar con vectores de tipo double.

3.21.2. Documentación de las funciones miembro

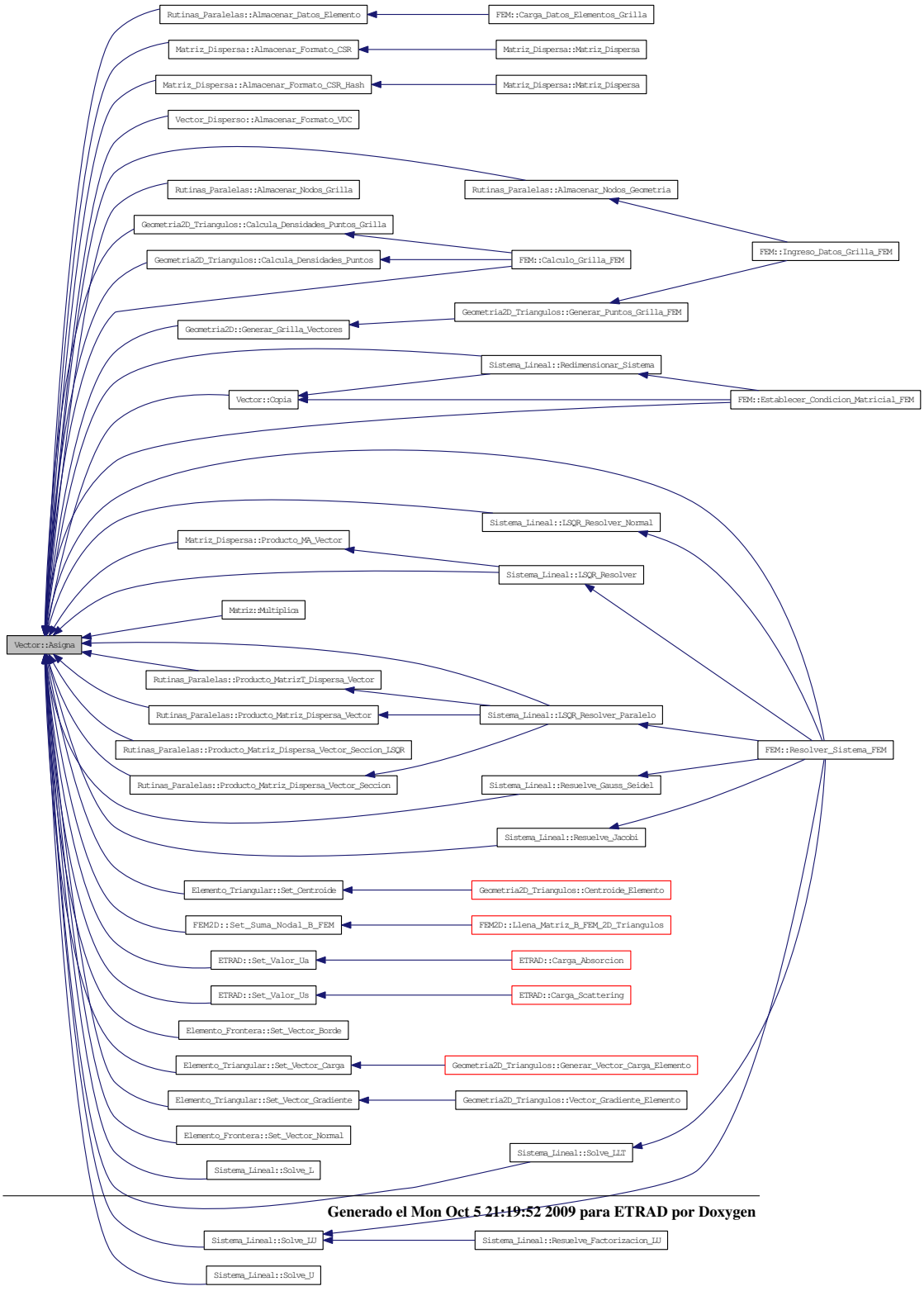
3.21.2.1. void Vector::Asigna (unsigned long long *col*, double *val*)

Asigna valor de una elemento del vector en columna col.

Asigna valor en columa especifica.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

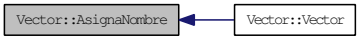


3.21.2.2. void Vector::AsignaNombre (const char * *nmb*)

Asigna un nombre al vector.

Asigna nombre a la matriz.

Gráfico de llamadas a esta función:

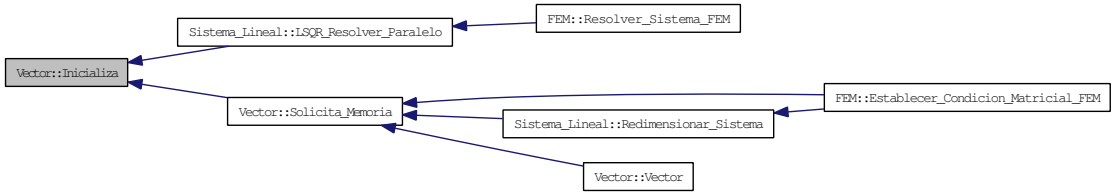


3.21.2.3. void Vector::Inicializa (double *val*)

Inicializa con valores val el vector.

Inicializa a val el vector.

Gráfico de llamadas a esta función:



3.21.2.4. void Vector::Multiplica (double *esc*)

Multiplica el vector por el escalar esc.

Multiplica el vector por el escalar.

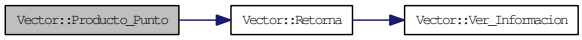
Gráfico de llamadas a esta función:

3.21.2.5. double Vector::Producto_Punto (Vector * *a*)

Calcula el producto punto.

Calcula el producto punto con otro vector.

Gráfico de llamadas para esta función:



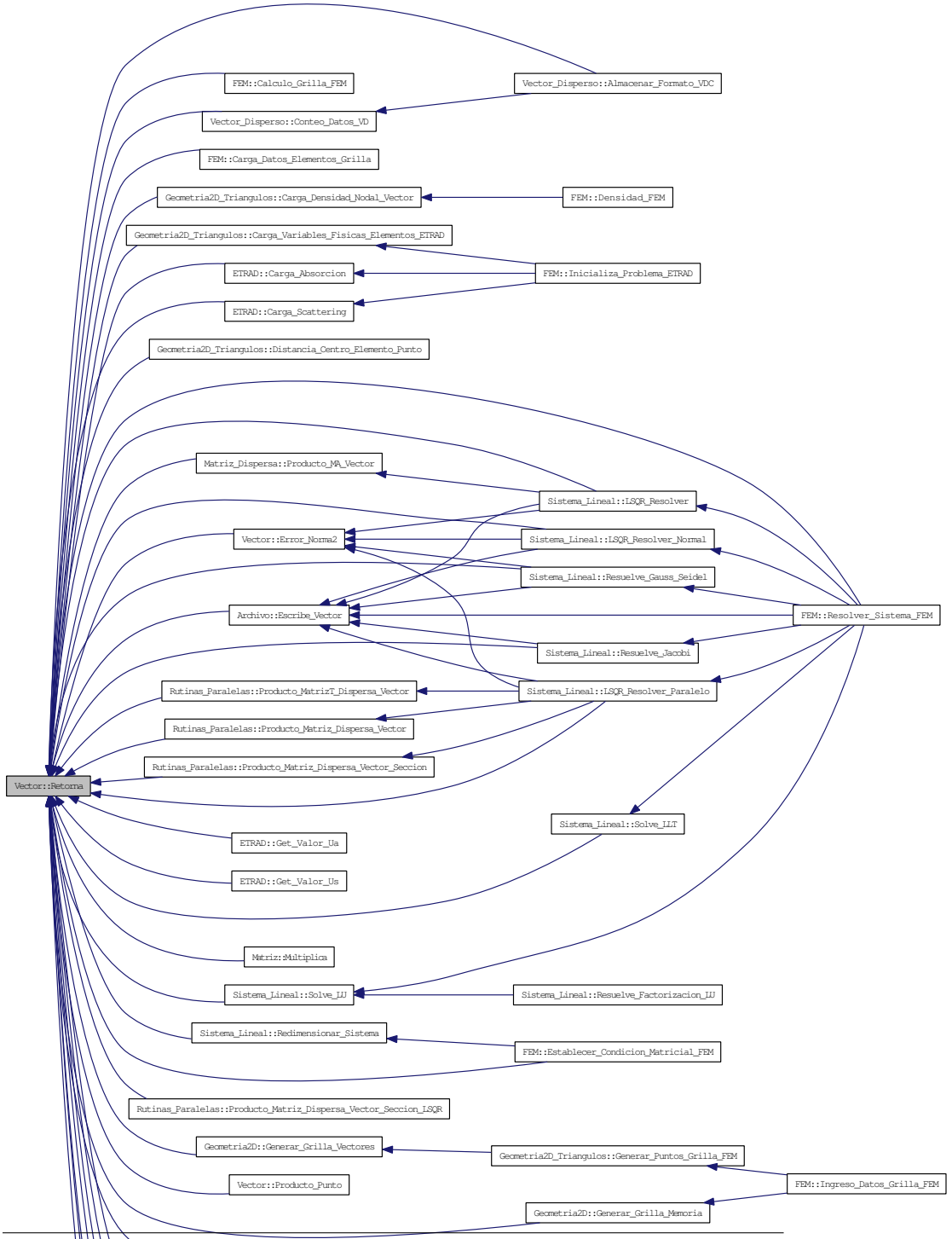
3.21.2.6. double Vector::Retorna (unsigned long long *col*)

Retorna valor de la columna col del vector.

Retorna valor en columna especifica.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:



3.21.2.7. double Vector::Sumatoria_Vector ()

Sumatoria elementos del vector.

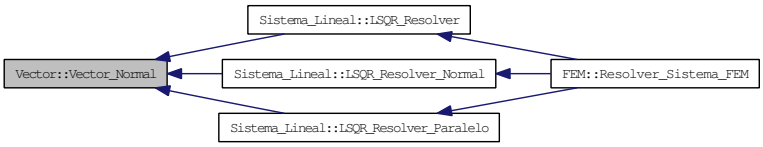
Sumatoria de los elementos de un vector.

3.21.2.8. double Vector::Vector_Normal ()

Calcula el vector Normal.

Calcula el Vector Normal.

Gráfico de llamadas a esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

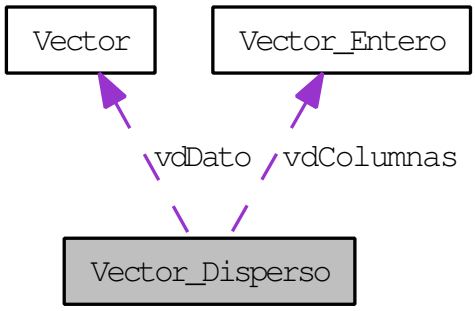
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vector.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vector.cpp

3.22. Referencia de la Clase Vector_Disperso

Clase para el trabajar con vectores dispersos de tipo double.

`#include <Vector_Disperso.hpp>`

Diagrama de colaboración para Vector_Disperso:



Métodos públicos

- Vector_Disperso (void)

  Constructor de la clase.

- Vector_Disperso (unsigned long columnas)

  Constructor sobrecargado de la clase.

- ~Vector_Disperso ()

  Destructor de la clase.

- void Almacenar_Formato_VDC (Vector *ENTRADA)

  Almacena en formato vector - columnas VDC.

- unsigned long Retorna_dimOriginal (void)

  Retorna el numero de columnas del vector origen.

- unsigned long Conteo_Datos_VD (Vector *Entrada)

  Cuenta el numero de elementos no ceros de un vector.

- unsigned long Retorna_nzDatos ()

  Retorna el numero de elementos no ceros del vector disperso.

Atributos protegidos

- Vector_Entero * vdColumnas

  Vector entero que contiene datos de columnas.

- Vector * vdDato

  Vector double que contiene los datos del vector disperso.

- unsigned long nzDatos

  Numero de elementos del vector no ceros.

- unsigned long **dimOriginal**

3.22.1. Descripción detallada

Clase para el trabajar con vectores dispersos de tipo double.

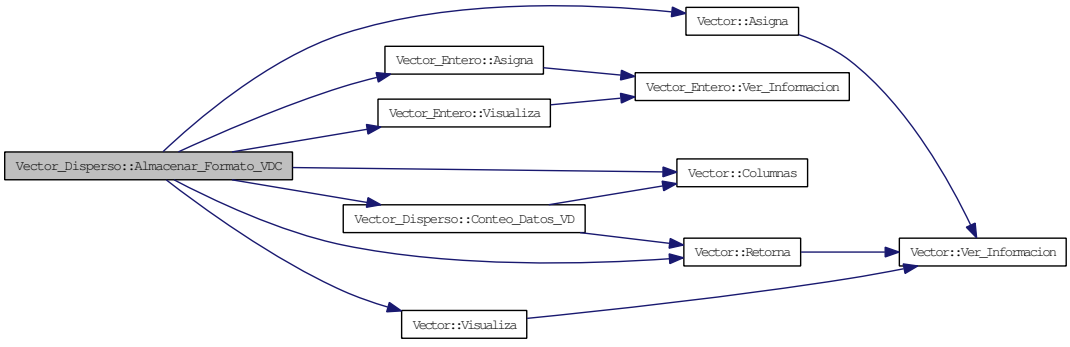
3.22.2. Documentación de las funciones miembro

3.22.2.1. void Vector_Disperso::Almacenar_Formato_VDC (Vector * *ENTRADA*)

Almacena en formato vector - columnas VDC.

Transforma un vector disperso a un formato Columna - Dato.

Gráfico de llamadas para esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vecto Disperso.hpp

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vector_-Disperso.cpp

3.23. Referencia de la Clase Vector_Entero

Clase para el trabajar con vectores enteros long long sin signo.

`#include <Vector_Entero.hpp>`

Métodos públicos

- Vector_Entero ()

  Constructor de la clase.

- Vector_Entero (unsigned long long col)

  Constructor sobrecargado de la clase.

- Vector_Entero (unsigned long long col, int val)

  Constructor sobrecargado de la clase.

- Vector_Entero (unsigned long long col, char ∗nmb)

  Constructor sobrecargado de la clase.

- ∼Vector_Entero ()

  Destructor de la clase.

- unsigned long long ∗ Retorna_Puntero_Datos ()

  Retorna el puntero donde se almacena el vector.

- void **Libera_Memoria** ()
- void AsignaNombre (const char ∗nmb)

  Asigna nombre al vector.

- char ∗ Nombre (void)

  Retorna el nombre al vector.

- unsigned long long Columnas (void)

  Retorna el numero de columnas.

- unsigned long long Buscar_Valor (unsigned long long valor)

  Busca un valor en el vector y retorna 1 en caso verdadero (solo indica si existe, no cuenta).

- unsigned long long Buscar_Valor_Limite (unsigned long long valor, unsigned long long limite)

Busca un valor en el vector hasta un cierto indice y retorna 1 en caso verdadero (solo indica si existe, no cuenta).

- void Inicializa (unsigned long long val)

  Inicializa con val el vector.

- void Asigna (unsigned long long col, unsigned long long val)

  Asigna un valor en posicion col.

- unsigned long long Retorna (unsigned long long col)

  Retorna valor de col.

- void Visualiza ()

  Visualiza el vector.

- void Copia (Vector_Entero *a)

  Copia el vector a otro vector a.

- float Tamano (void)

  Tamaño del vector (aproximado) en Kb.

- unsigned long long Sumatoria_Vector ()

  Sumatoria de datos del vector.

- void Ver_Informacion (void)

  Visualiza información general de la matriz.

Métodos protegidos

- void Solicita_Memoria (unsigned long long col, unsigned long long val)

  Solicita la memoria para el vector.

Atributos protegidos

- unsigned long long Columna

  Numero de columnas.

- char * Nmb

  Nombre del vector.

- unsigned long long * V

  Puntero al Vector entero.

3.23.1. Descripción detallada

Clase para el trabajar con vectores enteros long long sin signo.

3.23.2. Documentación de las funciones miembro

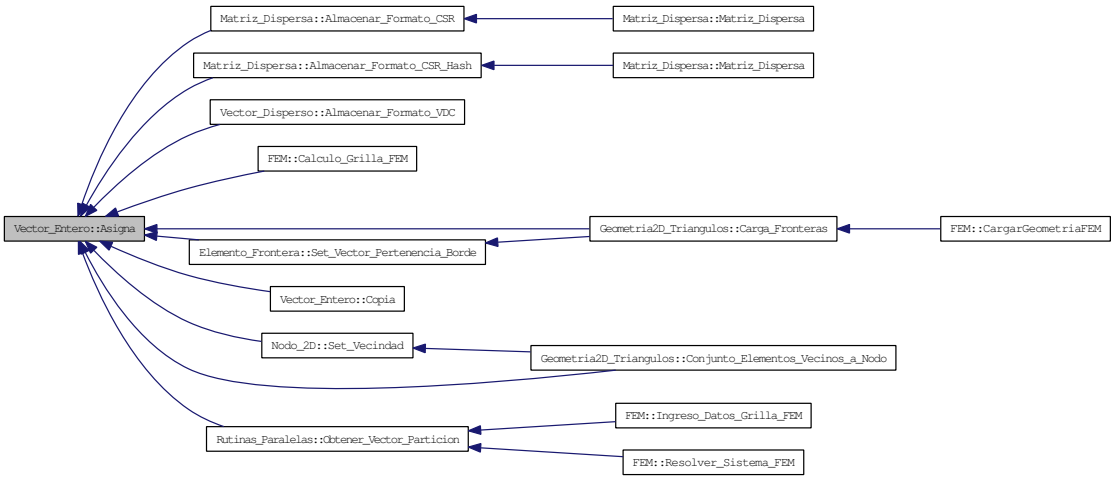
3.23.2.1. void Vector_Entero::Asigna (unsigned long long *col*, unsigned long long *val*)

Asigna un valor en posicion col.

Asigna valor en columna.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:



3.23.2.2. void Vector_Entero::AsignaNombre (const char * *nmb*)

Asigna nombre al vector.

Asigna nombre a la matriz.

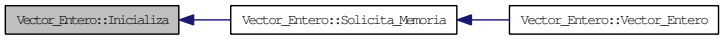
Gráfico de llamadas a esta función:

3.23.2.3. void Vector_Entero::Inicializa (unsigned long long *val*)

Inicializa con val el vector.

Inicializa a val el vector.

Gráfico de llamadas a esta función:



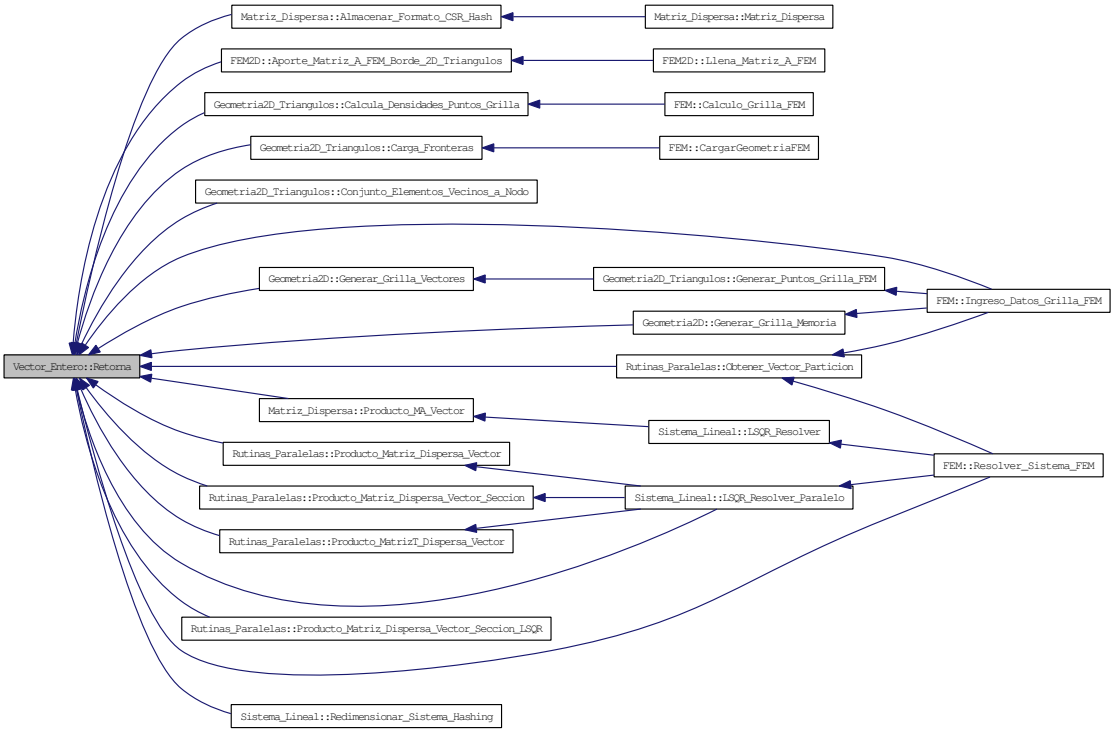
3.23.2.4. unsigned long long Vector_Entero::Retorna (unsigned long long *col*)

Retorna valor de col.

Retorna valor en columna.

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:



3.23.2.5. void Vector_Entero::Solicita_Memoria (unsigned long long *col*, unsigned long long *val*) [protected]

Solicita la memoria para el vector.

Solicita memoria para el vector.

Gráfico de llamadas para esta función:

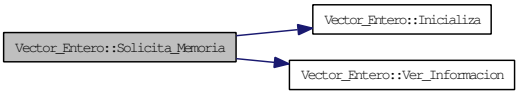
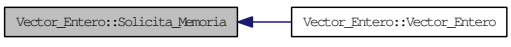


Gráfico de llamadas a esta función:

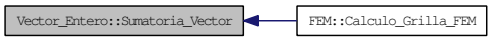


3.23.2.6. unsigned long long Vector_Entero::Sumatoria_Vector ()

Sumatoria de datos del vector.

Sumatoria de los elementos de un vector.

Gráfico de llamadas a esta función:



La documentación para esta clase fue generada a partir de los siguientes ficheros:

- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vector_-Entero.hpp
- C:/Documents and Settings/Oscar/Desktop/USACH/magister/DocumentoTesis/ETRADORD2009/src/Vector_-Entero.cpp

Índice alfabético