Plan del Proyecto

1.Objetivos

ID	Objetivos	Estado del Objetivo
1	Realizar pruebas Benchmark	*Iniciadas*
2	Realizar cambios	*Iniciado*
3	Validar el sistema	*Iniciado*
4	Revisar documentación del sistema	*Pendiente*

Cuadro 1: Lista de objetivos a completar.

2.Tareas

ID	Funcionalidad o Actividad		Responsable	Fecha de Entrega	Estado (Iniciado/ Terminado/ Pendiente
1	*Definir pruebas Benchmark*				
	1.1	Obtener problemas con solución analítica	Administrador del proyecto	*2-11-2016*	Iniciado
	1.2	Entender las pruebas a realizar	Desarrollador	*4-11-2016*	Iniciado
2	*Realizar Cambios*				
	2.1	Realizar los cambios necesarios al sistema para poder iniciar y aplicar las pruebas Benchmark	Desarrollador	*11-11-2016*	Pendiente
3	*Validación*				
	3.1	Aplicar el sistema a la pregunta inicial a resolver	Administrador, Desarrollador	*22-11-2016*	Pendiente
4	*Documentación*				
	4.1	Verificar que la documentación esté completa y disponible	Desarrollador	*20-11-2016*	Pendiente

3.Cambios

ID	Producto	Descripción	Solicitante	Estado
1	CVFEM	Modificación de la funcionalidad que construye la matriz y la del vector b de la ecuación $\underline{\underline{A}}\underline{x} = \underline{b}$	Desarrollador	Realizado
2	Visualizador	Construcción de una función para la graficación de una superficie	Administrador, Desarrollador	Realizado.
3	CVFEM	Cálculo de la funcionalidad de permeabilidad aleatoria	Desarrollador	Realizado.

4.Riesgos

Nombre	Descripción	Plan de contingencia	Impacto	Estado del riesgo
Fecha de entrega	La fecha de conclusión del proyecto esta próxima y aún falta mucho por realizar.	Solicitar una extensión de tiempo	Alto	Identificado
Reuniones	Reuniones con el Administrador y experto del tema, deben ser más frecuentes.	Constante comunicación por otros medios: E-mail, Teléfono, etc.	Alto	Controlado

Especificación de Requerimientos

1.Descripción general del problema

El sistema debe ser capaz de leer un dominio en dos dimensiones construido con la herramienta gmsh según la triangulación de Delaunay. Las especificaciones del archivo estan en el apéndice B. Con lo cual debe ser capaz de almacenar los nodos y los elementos triángulares, los nodos deben ser ordenados respecto a su coordenada y y después a su coordenada x.
Se tiene que construir un sistema de ecuaciones donde la matriz generada debe considerar condiciones de frontera y no debe de almacenar 0 's. El sistema de ecuaciones es resuelto con métodos numéricos de álgebra lineal. La solución final debe ser almacenada en un archivo con el siguiente formato, coordenada x, coordenada y y solución en ese punto, finalmente debe de poder graficarse la solución obtenida por lo cual se utiliza una herramienta externa.

2.Requerimientos funcionales

2.1. Tratamiento del Dominio de interés

- Leer el contenido de un archivo gmsh.
- Almacenar las coordenadas x, y de cada nodo del dominio.
- Almacenar los nodos que forman un elemento triangular.
- Reordenar los nodos, primero por su coordenada y y después por su coordenada x, por ejemplo:

Coordenada x	Coordenada y
0	0
0.5	0
1	0
0.2	0.1
0.5	0.4
0.6	0.5
0.3	0.7

- Reordenar los elementos según la nueva numeración de los nodos.
- Distinguir los nodos que están en el contorno del dominio y aquellos que están en el interior.
- Distinguir los nodos de contorno en dos tipos:
  - Neumann.
  - Dirichlet
- Almacenar el valor de cada nodo de frontera.

- Identificar para cada nodo que elementos son su soporte; es decir que elementos contienen a ese nodo como uno de sus vértices.
- Identificar los nodos vecinos de cada nodo.
- Cálcular el área de cada elemento triangular.

2.2. CVFEM

- Debe ser capaz de construir un sistema de ecuaciones de la forma $\underline{\underline{A}}\underline{x}=\underline{b}$.
- La matriz $\underline{\underline{A}}$ debe ser construida a partir de los coeficientes de transmisiblidad según la teoría y la discretización realizada.
- El vector $\underline{b}$ se debe construir a partir de la teoría y la discretización del vector.
- Debe de ser capaz de calcular la permeabilidad en distintos subdominos según lo descrito en el problema.

2.3. Almacenamiento de un vector.

- Debe almacenar en memoria los componentes de un vector en números de tipo flotante de doble precisión (double)..
- Se definen las operaciones vector-vector
  - Suma.
  - Resta.
  - Producto punto.
- Se definen las operaciones con el esquema de almacenamiento; Matriz por vector.

- Cálculo de una norma euclidiana $||V|| = \sqrt{{v_1}^2 + {v_2}^2 ... + {v_n}^2}$
- Cálculo del rms: $V_{rms}=\frac{\sqrt{{v_1}^2+{v_2}^2...+{v_n}^2}}{n}$

2.3. Almacenamiento de una matriz.

- Almacenamiento de los componentes de una matriz.
- Inicializar una matriz con ceros.
- Acceso a los valores de la matriz.
- Una iteración del algoritmo Jacobi;
- Una iteración del algoritmo Gauss Seidel.

2.4. Álgebra lineal.

- Implementar los métodos para sistemas de ecuaciones lineales.
  - Jacobi.
  - Gauss.
  - SOR.
  - Gradiente Conjugado.
  - BICGSTAB.
- Implementar precondicionadores de la matriz.
  - Jacobi.
  - MILU.
  - ILU.
  - ICHOL.

2.5. Visualizador.

- Graficación de la solución final usando la biblioteca matplotlib.
- Solve:
  - Devuelve la solución final en un archivo de texto según un visualizador externo.
  - Calcula la distribución del error para los problemas de tipo benchmark usando $|error = |exacta_{i,j} - calculada_{i,j}|$. Es decir se calcula para cada nodo su error.

3.Requerimientos no funcionales

- El rendimiento es un factor importante.
- Debe permitir un mantenimiento sencillo.
- Minimización del uso de memoria en el almacenamiento de la matriz.
- Utilizar técnicas de programación genérica y metaprogramación.

4.Restricciones de construcción

El lenguaje de programación debe ser C++

Arquitectura y Diseño Detallado de Software

1.Normatividad Interna

- Los atributos inician con $_(Palabra(s)clave(s))$.
- Los atributos privados terminan con P mayúscula.
- Palabra clave inicia con minúscula, si son 2 palabras la primera con minúscula y la segunda con mayúscula.
- Nombre de las clases con mayúscula.
- Nombre de la clase = Nombre del archivo = Palabra identificadora.
- Toda clase se escribe en 2 archivos: hpp y cpp.
- Los comentarios se hacen por línea y se deja un espacio en blanco antes de iniciar la escritura de los comentarios.
- Al inicio de cada archivo se describe generalmente lo que hace la clase.
- Nombre del autor.
- Descripción del programa de manera general.
- Los métodos inician con minúscula y las palabras clave terminan con P mayúscula.
- Toda clase debe tener un constructor vacío y el constructor copia.

- Los parámetros variables de los métodos inician con minúsculas.

2.Arquitectura de Software

Se presenta una descripción abstracta (Ver Fig. 1) de la arquitectura propuesta. Donde cada bloque de la arquitectura, tiene una tarea específica, con base en los requerimientos presentados anteriormente. Ésto permite que el mantenimiento sea sencillo, pues se tendrá bien especificado quién hace qué y cómo lo hace.

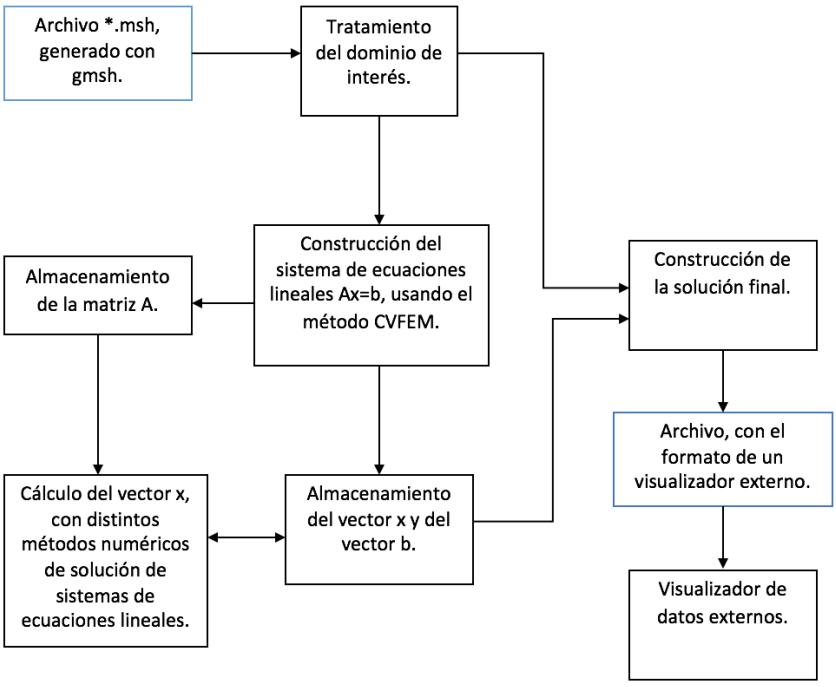

Figura 1: Diagrama de bloques de la arquitectura para SYSG.

Los datos pasan de un sistema a otro en la dirección de la flecha.

Los bloques acceden a la información de otro bloque por medio de una interfaz, como resultado, los bloques deben estar débilmente acoplados. Por lo tanto si se llega a la conclusión que un bloque no satisface los requerimientos, puede ser reemplazado por otro fácilmente, siempre y cuando la interfaz se mantenga.

3. Descripción de Componentes de Software

Tratamiento del dominio de interés

Se transforma en la clase mesh, en este caso sus funcionalidades principales son:

- Almacenar los nodos y los elementos de la malla.
- Reordenar los nodos en forma ascendente.
- Identificar los nodos frontera y los nodos interiores.
- Identificar los elementos soporte de cada nodo.
- Dado un nodo, identificar los nodos vecinos.
- Calcular el área de cada elemento triángular.
- Almacenamiento de las condiciones de frontera de cada nodo.

Construcción del sistema de ecuaciones con CVFEM

Se transforma en la clase CVFEM, su objetivo principal es formar el sistema de ecuaciones $\underline{\underline{A}}\underline{x} = \underline{b}$, aplicando la expresión discreta a cada vértice de la malla, tomando en cuenta las condiciones de frontera. Sus operaciones más importantes son:

- Cálculo de los coeficientes de las funciones de forma.
- Cálculo de los coeficientes de transmisibilidad.
- Construcción de la matriz $\underline{A}$.
- Cálculo de regiones de distinta permeabilidad.
- Cálculo del vector b.

Almacenamiento de la matriz

La matriz que se construye con el CVFEM, es dispersa y no estructurada, por lo tanto muchos de sus componentes son cero y aquellos que no son cero se presentan en posiciones irregulares. Los componentes que son cero no son significativos para obtener la aproximación de la solución final, por lo tanto, almacenarlos constituye una pérdida de memoria, conviene entonces usar estrategias de almacenamiento que permitan almacenar y operar sólo la información relevante. Existen distintos esquemas de almacenamiento para matrices dispersas en éste trabajo se eligieron COO y CSR.

COO

Esta estrategia es la más natural, almacena el valor y las coordenadas i y j de cada componente no cero en tres arreglos de nombre: data, row y col. La ventaja de éste esquema con relación a otros es que la inserción es constante, sin embargo, la búsqueda de un elemento es una operación ineficiente cuya complejidad es $O(nnz)$, donde nnz es el número de elementos no cero, debido a que los elementos pueden estar desordenados.

CSR

Esta estrategia, almacena los componentes por renglón en tres arreglos: data, col e irow, en éstos se almacenan los datos, el índice de columna y los índices de inicio y fin de cada renglón. La ventaja de éste es que la complejidad de buscar un elemento dentro de la matriz es $O(\log_2 n)$, donde n es el número máximo de elementos en un renglón.
Sabiendo todo esto y con el fin de minimizar el tiempo de procesamiento y el almacenamiento, es utilizado el esquema COO para la construcción de la matriz y una vez formado se convierte al esquema CSR.

Almacenamiento de Vectores

Se transforma en la clase Vector, su objetivo principal, es almacenar y operar la información de los vectores definidos por CVFEM. Sus operaciones principales son:

- Almacenamiento de un vector dado.
- Cálculo de la norma euclidiana.
- Cálculo de la media cuadrática rms.

Sistemas de ecuaciones

Se implementa una librería de métodos iterativos, los cuales son: Gauss Seidel, Jacobi, SOR, Gradiente conjugado(CG) y BICGSTAB. Además con el fin, de mejorar la matriz son implementados precondicionadores los cuales son: Jacobi, ILU, MILU y ICHOL. Pueden ser usados por BICGSTAB Y CG.

Construcción de la solución final

Una vez que se tiene el resultado aproximado del problema en cada nodo, se procede a generar un archivo de salida, para ser leído por un visualizador externo. En éste caso, el bloque se transforma en la clase Solve, que genera un archivo de texto con las coordenadas de cada nodo de la malla y su valor aproximado.

Visualizador de datos externos

Se eligió como visualizador externo la biblioteca Matplotlib, para la visualización de los datos, para tal motivo se desarrolla un programa en python que permita su uso.

Reporte de Seguimiento

Proyecto: *[Simulación de flujo de medios porosos usando CVFE]* **Periodo a Reportar:** 1-11-2016.] al [30-11-2016]

1.Reporte de Actividades

Integrante	**Actividad u Objetivo**	**Tiempo Total**	**Estado**	**Observaciones**
Desarrollador	Pruebas Benchmark	45 hr	Finalizado	La obtención de las pruebas de parte del administrador tomó dos horas, el entendimiento y realización de las pruebas tomó mucho tiempo, fueron 3 pruebas a realizar, y en cada prueba se realizaron los cambios necesarios para el buen funcionamiento del sistema.
Desarrollador	Cambios	40 hr	Finalizado	Los cambios realizados fueron realizados en conjunto con las pruebas Benchmark, se realizaron cambios en el paquete del mallado, CVFEM.
Desarrollador	Validación	40 hr	Pendiente	La validación se encuentra pendiente, debido a los cambios que se deben realizar.
Desarrollador	Documentación	16 hr	Pendiente	La documentación esta siendo revisada y actualizada, aún falta por definir completamente ciertos documentos.
Total de Horas trabajadas		141		

2.Tareas finalizadas

Actividad y/o tarea	**Integrante**	**Fecha de entrega**	**Fecha real**
Obtener la definición del problema benchmark	Desarrollador	*[2-11-2016]*	*[2-11-2016]*
Entendimiento de las pruebas	Desarrollador	*[4-11-2016]*	*[16-11-2016]*
Realizar cambios	Desarrollador	*[11-11-2016]*	*[16-11-2016]*
Validación	Desarrollador	*[22-11-2016]*	Pendiente
Documentación	Equipo de trabajo	*[22-11-2016]*	Pendiente

3.Productos

ID	**Producto**	**Estado**
1	Tratamiento del Dominio	Terminado
2	CVFEM	Terminado
3	Almacenamiento de la Matriz $\underline{\underline{A}}$	Terminado
4	Calculo del vector $\underline{x}$	Terminado
5	Almacenamiento del vector $\underline{x}$ y de $\underline{b}$	Terminado
6	Visualizador	Iniciado

4.Cambios

ID	Producto	Descripción	Solicitante	Estado
1	Tratamiento del Dominio	Se realizaron los siguientes cambios: ▪ Reordenamiento de nodos. ▪ Distinción entre nodos frontera e interiores. ▪ Tratamiento de nodos Neumann y Dirichlet.	Desarrollador	Realizado
2	CVFEM	Se realizaron los siguientes cambios: ▪ Análisis de permeabilidad. ▪ Distinción de subdominios. ▪ Realización de tipos de matrices para nodos tipo Neumann y Dirichlet.	Desarrollador	Realizado

5.Riesgos

Nombre	**Descripción**	**Plan de contingencia**	**Impacto**	**Estado del riesgo**
Tiempo para realizar cambios	El tiempo de entrega y el tiempo para la realización de los cambios se visualiza es insuficiente.	Solicitar una extensión de tiempo	Alto	Identificado
Cambios que puedan comprometer el funcionamiento del sistema	Al momento de realizar los cambios el sistema pueda dejar de funcionar.	Tener las versiones del código disponibles en caso de que no se halle la solución al problema de manera inmediata.	Alto	Controlado

6.Resumen

Tareas	**Cambios**	**Riesgos**
A tiempo: 4	Solicitados: 5	Encontrados 2
Retrasadas: 2	Rechazados: 0	Resueltos 1
Adelantadas:0	Realizados: 5	Postergados 1
Postergadas: 2	Postergados: 0	

Cierre del Plan del ProyectoNo.

1.Resumen

Productos	**Cambios**
A tiempo: 5	Solicitados: 0
Retrasados: 0	Rechazados: 0
Adelantados: 0	Realizados:7
Postergados: 1	Postergados:0

Riesgos	**Actividades u Objetivos**
Encontrados: 2	Postergados: 2
Resueltos: 2	Alcanzados: 2

2.Reporte de productos.

Nombre del producto	**Cambios**		**Estado del producto**
	Encontrados	**Corregidos**	
Tratamiento del Dominio	3	3	*[Terminado*
CVFEM	3	3	*Terminado*
Almacenamiento de la matriz $\underline{\underline{A}}$	0	0	*[Terminado*
Cálculo del vector $\underline{x}$	0	0	*[Terminado*
Almacenamiento del vector $\underline{x}$ y del vector $\underline{b}$	0	0	*Terminado*

3.Retroalimentación.
Tareas por terminar.

- Terminar la validación del sistema.
- Iniciar el cierre del proyecto.

4.Comentarios
Se tiene el sistema en un estado próximo a finalizar, faltan por realizar ciertas modificaciones al sistema relacionados con la validación, una vez realizadas deben de ser aprobadas por el Administrador.El trabajo a realizar es complejo y en ésta parte del proyecto se necesita de la experiencia y apoyo del experto en el área. Aún no se tiene la certeza del tiempo que se necesita para realizar estas modificaciones pero se trabaja para que todo el sistema esté listo para antes de diciembre del 2016, esperando presentar el trabajo para evaluación en el mes de enero del 2017.

Pruebas de Paquete

La clase Mesh es la encargada del tratamiento del dominio que previamente debió ser construido y discretizado con la herramienta gmsh.

Datos de entrada

Se requiere los nodos que forman el dominio rectangular y el archivo *.msh que contiene la discretización.

Flujo esperado

Dado que se está trabajando con geometrías rectangulares se debe indicar los 4 nodos que forman el dominio para que el sistema pueda distinguir los nodos que se encuentran en la frontera y los que se encuentran en su interior. Se carga el archivo que contiene la geometría discretizada. Los nodos que se encuentran en la frontera pueden ser Dirichlet o Neumman por lo tanto debe indicarse para cada nodo su tipo y su valor. Se deben obtener los nodos que son incógnitas los cuales son nodos interiores o nodos Neumman, esto en código:

```
// Definimos los vertices del dominio
std::vector<double> vertices;
vertices.push_back(0.0);
vertices.push_back(0.0);
vertices.push_back(1.0);
vertices.push_back(2.0);
// Objeto para el tratamiento de la malla
Mesh mesh("./GMSH/Laplace/mesh10.msh",vertices);
// Cargamos las condiciones de frontera
std::vector<int> border = mesh.bordeNode();
for (int i = 0; i < border.size(); ++i)
{
        if (mesh.coordY(border[i]) == 0
        && (mesh.coordX(border[i]) != 0
        && mesh.coordX(border[i]) != 1))
        {
                mesh.bondaryCondition(0,100);
```

```
                }
                else
                {
                        mesh.bondaryCondition(0,0);
                }
        }
        // Mostramos el valor de las condiciones
        mesh.showBoundaryCondition();
        // Es necesario determinar los nodos incognita
        mesh.nodeIncongnite();
        mesh.showTotal();
```

Resultado esperado

Se espera que los nodos y los elementos sean leídos del archivo *.msh y almacenados en memoria, que se sean numerados según lo indicado en los requerimientos. Además debe ser capaz de reconocer qué nodos son interiores y que nodos son frontera. Y debe indicar que nodos son las incógnitas del problema, para que la siguiente clase se encargue de la construcción del sistema.

Cambios realizados

Sin cambios.

Defectos encontrados

Sin defectos encontrados

Pruebas de Paquete

La clase CVFEM es la encargada de construir la matriz $\underline{\underline{A}}$ y el vector $\underline{b}$, para que posteriormente sea resuelto.

Datos de entrada

Se requiere la información de la malla y contenedores para la matriz $\underline{\underline{A}}$ y el vector $\underline{b}$.

Flujo esperado

Se requiere definir los contenedores para el almacenamiento de la matriz y del vector, esto en código:

```
// Definimos las variables necesarias
CSR A(mesh.nNodeIncongnite());     // Matriz CVFEM
Vector x(mesh.nNodeIncongnite()); // Vector x
Vector b(mesh.nNodeIncongnite()); // Vector b
Solve solver; // Para el trataiento de las soluciones
// Objeto para crear el sistema de ecuaciones
CVFEM cvfem;
cvfem.Ab(A,b,mesh);
```

Resultado esperado

Se espera obtener una matriz dispersa no estructura y una vector.

Cambios realizados

Sin cambios.

Defectos encontrados

Sin defectos encontrados.

Pruebas de Paquete

Se prueba la clase COO Y CSR que implementan esquemas útiles de almacenamiento, se eligió

Datos de entrada

Se requiere una matriz dispersa como la mostrada en la figura

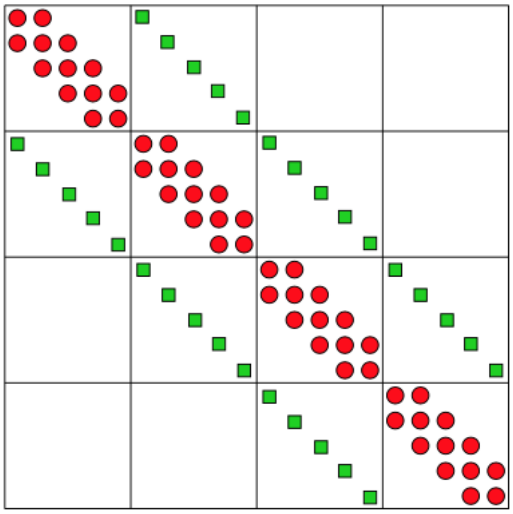

Figura 2: Matriz dispersa

Flujo esperado

Se define el tamaño de la matriz que será almacenada, de manera temporal se almacena en un formato COO, finalmente se transforma la matriz COO a una matriz CSR, esto en código:

```
{
  COO Acoo(n); //matriz temporal
   timer.tic(); //comienza a medir tiempo
   for(int j=1;j<ny;++j)
   {
     for(int i=1;i<nx;++i)
```

```
      {
        if(j>1)
          Acoo.insert(l, l-(nx-1),-1.);
        if(i>1)
          Acoo.insert(l, l-1,-1.);
          Acoo.insert(l, l,4.);
        if(i<nx-1)
          Acoo.insert(l, l+1,-1.);
        if(j<ny-1)
          Acoo.insert(l, l+(ny-1),-1.);
          ++l;
      }
    }
    A.convert(Acoo); //convierte de COO a formato CSR
    Acoo.impMatrix();
  }
//destruye Acoo
```

Resultado esperado

Después de compilar y ejecutar el código anterior el resultado obtenido puede verse en la figura 3. La tabla muestra el tiempo utilizado para la construcción de matrices de distinto tamaño en formato COO y su conversión a CSR.

```
4    -1    0    -1    0    0    0    0    0
-1    4    -1    0    -1    0    0    0    0
0    -1    4    0    0    -1    0    0    0
-1    0    0    4    -1    0    -1    0    0
0    -1    0    -1    4    -1    0    -1    0
0    0    -1    0    -1    4    0    0    -1
0    0    0    -1    0    0    4    -1    0
0    0    0    0    -1    0    -1    4    -1
0    0    0    0    0    -1    0    -1    4
```

Figura 3: Se muestra la matriz almacenada en el formato CSR

Tamaño	Inserción COO [ms]	Conversión CSR [ms]
100x100	0.002	0.002
10000x10000	0.12	0.199
40000x40000	0.584	1.126
90000x90000	1.146	2.018
160000x160000	2.745	3.862
250000x250000	5.185	6.504
360000x360000	6.733	9.288
640000x640000	8.05	16.027
810000x810000	11.155	19.669
1000000x1000000	13.785	28.496

Cuadro 3: Ejecuciones para distintos tamaños de una matriz.

Cambios realizados

Sin cambios.

Defectos encontrados

Sin defectos encontrados

Pruebas de Paquete

Uno de los requerimientos no funcionales, indicaba que la eficiencia era un factor importante. Con éste objetivo en mente se desarrollaron dos implementaciones diferentes para estos métodos, la primera usando sobrecarga de operadores y la segunda utilizando programación genérica y técnicas de metaprogramación.
Se prueba las clases los métodos numéricos: Jacobi, Gauss-Seidel, SOR, Gradiente Conjudado (CG) y el gradiente conjugado estabilizados (BICGSTAB), y para los últimos dos se prueba también los precondicionadores.

Datos de entrada

Se requiere de un sistema de ecuaciones como el mostrado en la figura 8.

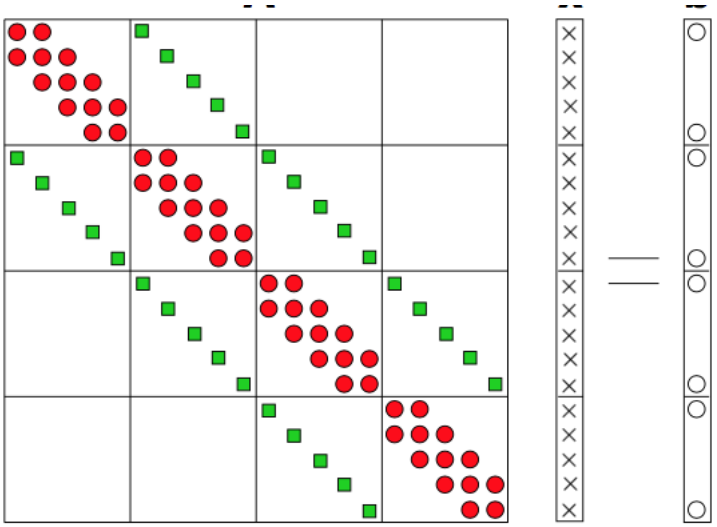

Figura 4: Se muestra la matriz almacenada en el formato CSR

Flujo esperado

Se define el tamaño de la matriz que será almacenada, de manera temporal se almacena en un formato COO, finalmente se transforma la matriz COO a una matriz CSR, esto en código:

```
int ny=nx;
double dx = 1./nx;
double dy = 1./ny;
int n = (nx - 1)*(ny - 1);
//almacenamiento
CSR A(n); //matriz en formato CSR
Vector x(n),b(n),r(n);

{
  COO Acoo(n); //matriz temporal
  timer.tic(); //comienza a medir tiempo
  for(int j=1;j<ny;++j)
  {
    for(int i=1;i<nx;++i)
    {
      if(j>1)
        Acoo.insert(l, l-(nx-1),-1.);
      if(i>1)
        Acoo.insert(l, l-1,-1.);
        Acoo.insert(l, l,4.);
      if(i<nx-1)
        Acoo.insert(l, l+1,-1.);
      if(j<ny-1)
        Acoo.insert(l, l+(ny-1),-1.);
        ++l;
    }
  }
  A.convert(Acoo); //convierte de COO a formato CSR
  Acoo.impMatrix();
}
b = 1.*dx*dx;
x=0.0;
```

Dependiendo de que método se desea utilizar para la solución del sistema se llama.

Resultado esperado

Se espera obtener un vector solución, que tenga un error menor a la tolerancia definida en tiempos cortos, además se espera que la implementación por medio de programación genérica y técnicas de programación sea mucho mejor. En éste caso los se definió una tolerancia de 1e-6 y se se dio un máximo de 2000 iteraciones para cada método, con lo cual Jacobi, Gauss-Seidel y SOR no convergían. Se presenta solo la comparación en velocidad de las dos implementaciones para CG y BICGSTAB precondicionados y no precondicionados en las siguientes grafica:

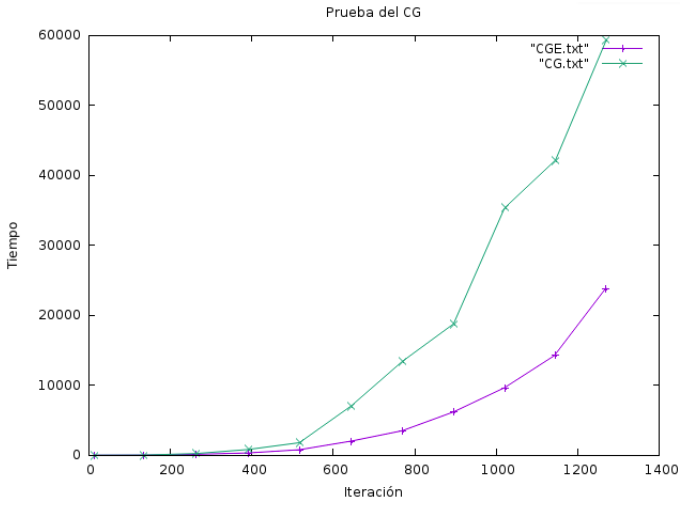

Figura 5: Comparación de tiempo para el algoritmo gradiente conjugado, usando dos implementaciones diferentes.

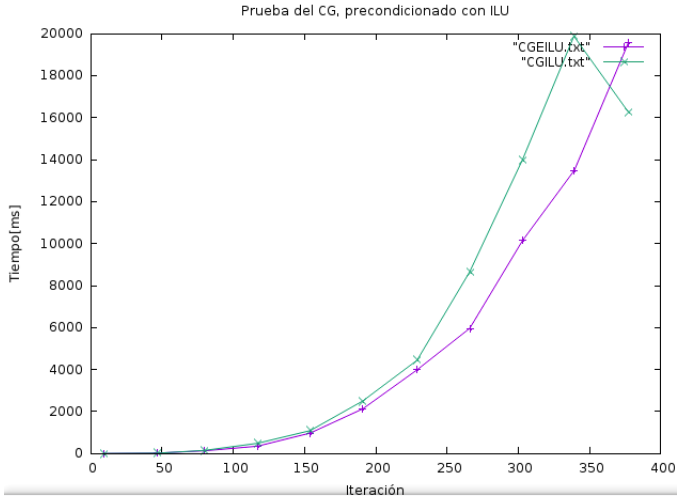

Figura 6: Comparación de tiempo para el algoritmo gradiente conjugado precondicionado con ILU, usando dos implementaciones diferentes.

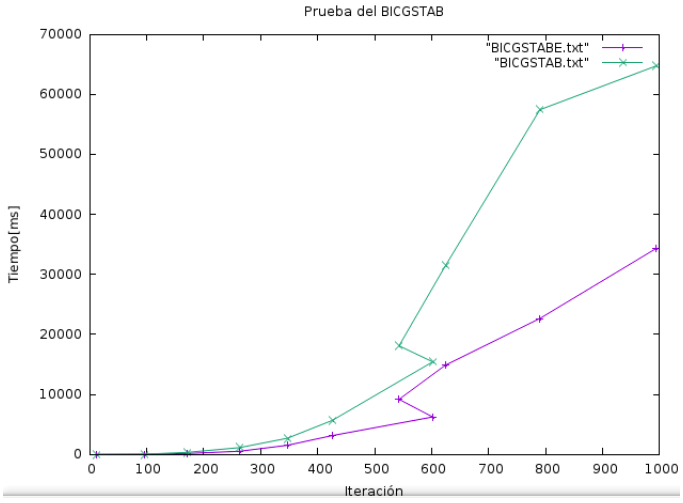

Figura 7: Comparación de tiempo para el algoritmo BICGSTAB, usando dos implementaciones diferentes.

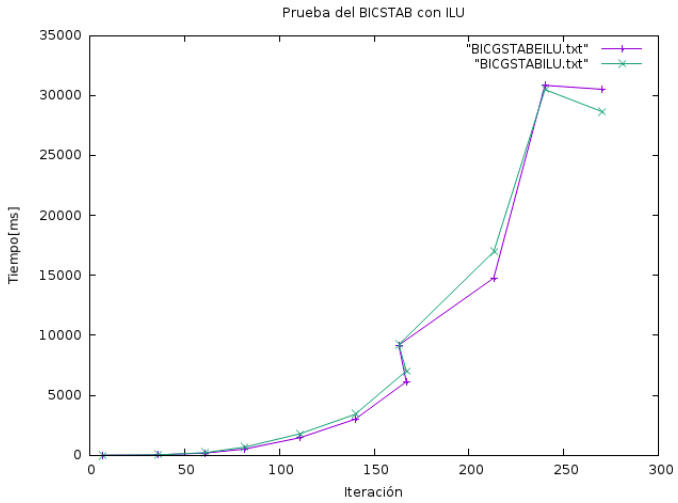

Figura 8: Comparación de tiempo para el algoritmo BICGSTAB precondicionado con ILU, usando dos implementaciones diferentes.

Cada punto de las gráficas representa un sistema donde la matriz tenia un tamaño de: 100x100, 10000x10000, 40000x40000, 90000x90000, 160000x160000, 250000x250000, 360000x360000, 640000x640000, 810000x810000, 1000000x1000000. Nótese que la implementación con programación genérica y metaprogramación es mucho mejor.

Cambios realizados

Sin cambios.

Defectos encontrados

Sin defectos encontrados