Tarea 4:
Métodos iterativos y método Cholesky

Resumen

En este trabajo se menciona un método para la factorización de matrices, y otros métodos para la solución de sistemas de ecuaciones lineales.

1. Introducción

Existen distintos métodos para factorizar una matriz: método de Crout, método de Doolittle y método de descomposición LDU, etc. Sin embargo, existen otros con los que el tiempo y las operaciones realizadas en la factorización, se disminuyen. Uno de estos métodos, es el método de Cholesky, el cual descompone una matriz A (definida positiva) en una matriz L y su transpuesta, es decir, con encontrar una matriz, se tiene la factorización completa. Además, esta matriz no tiene términos en la triangular superior, en otras palabras, la matriz A se factoriza en una matriz triangular inferior y su transpuesta.

Existen otros tipo de matrices, llamados matrices banda, donde los valores no nulos están cerca de la diagonal principal, y una forma de resolver sistemas de ecuaciones con matrices de esta forma, es utilizando el método de Cholesky, que sólo necesita un ciclo para poder descomponer la matriz. También, para resolver sistemas de ecuaciones se pueden utilizar métodos iterativos, los cuales comienzan con una aproximación inicial y van sustituyendo el resultado hasta que se se detiene el método porque el error es mínimo, o tiene las iteraciones suficientes.

2. Metodología

2.1. Cholesky

Sea A una matriz definida positiva, cuadrada:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} L_{00} & 0 & 0 & 0 \\ L_{10} & L_{11} & 0 & 0 \\ L_{20} & L_{21} & L_{22} & 0 \\ L_{30} & L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{00} & L_{01} & L_{02} & L_{03} \\ 0 & L_{11} & L_{12} & L_{13} \\ 0 & 0 & L_{22} & L_{23} \\ 0 & 0 & 0 & L_{33} \end{pmatrix}, \tag{1}$$

de donde se deducen:

para $i < j$: (2)

$$L_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{kj}}{L_{jj}},$$

para $i = j$, (3)

$$L_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}.$$

(4)

Algorithm 1 Método Cholesky

Entrada: Dimensión n de la matriz A, elementos de la matriz A, suma, suma2, suma3
Salida: Elementos de la matriz L

Hacer $l_{00} = \sqrt{a_{00}}$
for $j = 1;\ j < N; j++$ **do**
  $l_{j0} = a_{j0/l_{00}}$
  for $i = 1;\ i <= j;\ i++$ **do**
    suma = 0
    if $(i == j)$ **then**
      for $k = 0;\ k <= i-1;\ k++$ **do**
        suma = suma + l_{ik}^2
      $l_{ii} = a_{ii} - suma$
    else
      for $k = 0;\ k <= i-1;\ k++$ **do**
        suma = suma + $l_{jk}l_{ik}$
      $l_{ji} = (a_{ji} - suma)/l_{ii}$
return l_{ij}

2.2. Adaptar Cholesky a una matriz tridiagonal

Una de las ventajas de una matriz tridiagonal, es que es posible reducir el número de cálculos que se hacen para una factorización común, o incluso la forma de llenar los datos. En este caso, es una matriz tridiagonal porque sólo tiene valores en la diagonal y en sus vecinas, y esto se podía escribir en tres vectores, y con eso se puede llenar la matriz. De la misma forma, al hacer la factorización LL^T, solo se necesita un ciclo.

$$\begin{pmatrix} \alpha_0 & \beta_0 & 0 & 0 \\ \beta_0 & \alpha_1 & \beta_1 & 0 \\ 0 & \beta_1 & \alpha_2 & \beta_2 \\ 0 & 0 & \beta_2 & \alpha_3 \end{pmatrix} = \begin{pmatrix} L_{00}^2 & 0 & 0 & 0 \\ L_{10} & L_{11} & 0 & 0 \\ 0 & L_{21} & L_{22} & 0 \\ 0 & 0 & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{00} & L_{10} & 0 & 0 \\ 0 & L_{11} & L_{21} & 0 \\ 0 & 0 & L_{22} & L_{32} \\ 0 & 0 & 0 & L_{33} \end{pmatrix}, \tag{5}$$

$$\begin{pmatrix} \alpha_0 & \beta_0 & 0 & 0 \\ \beta_0 & \alpha_1 & \beta_1 & 0 \\ 0 & \beta_1 & \alpha_2 & \beta_2 \\ 0 & 0 & \beta_2 & \alpha_3 \end{pmatrix} = \begin{pmatrix} L_{00} & L_{00}L_{10} & 0 & 0 \\ 0 & L_{10}^2 + L_{11}^2 & L_{11}L_{21} & 0 \\ 0 & 0 & L_{21}^2 + L_{22}^2 & L_{22}L_{32} \\ 0 & 0 & 0 & L_{31}^2 + L_{33}^2 \end{pmatrix}. \tag{6}$$

$$L_{00} = \alpha_0, \qquad L_{11} = \sqrt{\alpha_1 - L_{10}^2}, \qquad L_{22} = \sqrt{\alpha_2 - L_{21}^2},$$

$$L_{10} = \frac{\beta_0}{L_{00}}, \qquad L_{21} = \frac{\beta_1}{L_{11}}, \qquad L_{32} = \frac{\beta_2}{L_{22}},$$

de donde se deduce una forma general para encontrar su factorización:

$$L_{i+1,i} = \frac{\beta_i}{L_{ii}} \tag{7}$$

$$L_{i+1,i+1} = \sqrt{\alpha_{i+1} - L_{i+1,i}^2}. \tag{8}$$

Algorithm 2 Método Cholesky para una matriz tridiagonal

Entrada: Matriz tridiagonal M, matriz vacia L, número filas: N
Salida: Elementos de la matriz L

$L[0][0] = \sqrt{\alpha[0]}$
for $i = 0; i < N; i++$ **do**
  if $(i + 1 < N)$ **then**
    $L[i+1][i] = \frac{\beta[i]}{L[i][i]}$
    $L[i+1][i+1] = \sqrt{\alpha[i+1] - L[i+1][i]^2}$
return l_{ij}

2.3. Aplicación del método de Cholesky a una matriz tridiagonal

Sea un problema elíptico unidimensional, del cual conocemos el valor de su segunda derivada y las condiciones de frontera:

$$\begin{cases} \frac{\mathrm{d}^2\mathrm{f}}{\mathrm{d}x^2} = 2 & 0 \leq x \leq 1, \\ f(0) = 0, & f(1) = 2. \end{cases} \tag{9}$$

Sabemos que podemos aproximar la derivada de una función por diferencias finitas, y se tendrá:

$$\frac{d^2f}{dx^2} = \frac{f_i - 2f_{i-1} + f_{i-2}}{h^2}, \tag{10}$$

y los términos que nos interesan son los coeficientes dentro de la derivada, los números 1, 2 y 1. Si multiplicamos por un negativo, tendríamos -1, 2, -1 y podríamos hacer una matriz tridiagonal para buscar la solución de la función con base en la segunda derivada y se tendría una matriz de esta forma:

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \frac{d^2f_0}{dx^2} + f_0 \\ \frac{d^2f_1}{dx^2} \\ \frac{d^2f_2}{dx^2} \\ \frac{d^2f_3}{dx^2} \\ \frac{d^2f_4}{dx^2} + f_4 \end{pmatrix} \tag{11}$$

este es el caso para 4 dimensiones, sin embargo, se tendría que hacer el mismor razonamiento para n dimensiones, como la segunda derivada está siendo dividida por h^2, se debe de pasar ese término al otro lado de la ecuación, es decir:

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = h^2 \begin{pmatrix} \frac{d^2f_0}{dx^2} + f_0 \\ \frac{d^2f_1}{dx^2} \\ \frac{d^2f_2}{dx^2} \\ \frac{d^2f_3}{dx^2} \\ \frac{d^2f_4}{dx^2} + f_4 \end{pmatrix}$$

La razón para quitar el signo negativo de la diagonal principal fue porque la factorización de la matriz triadigonal aplica raíces a los términos de la digonal principal.

2.4. Método Jacobi

El método de Jacobi está dentro de los métodos iterativos para la solución de sistemas de ecuaciones, estos métodos comienzan con una aproximación inicial, y a cada iteración se va mejorando la aproximación hasta cierto número de iteraciones o que la norma del vector sea mínima.

$$x_i = \frac{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j}{a_{ii}} \tag{12}$$

Algorithm 3 Método Jacobi

Entrada: n: cantidad de filas, matriz A[n][n], b[n], VV[n], número iteraciones, valor
Salida: Solución x

1: VV: vector viejo, VN: vector nuevo
2: Inicializar vector viejo: $VV[i] = b[i]/a[i][i]$
3: **Mientras no se supere el número de iteraciones:**
4: **for** $i = 0; i < N; i++$ **do**
5: suma = 0
6: **for** $j = 0; j < N; j++$ **do**
7: **if** $i \neq j$ **then**
8: suma += $a[i][j] * VV[j]$
9: $VN[i] = \frac{b[i]-suma}{a[i][i]}$
10: Calcular la tolerancia
11: **if** $TOL < valor$ **then**
12: termina el programa: break;
13: **Else** Reemplaza vector viejo, con vector nuevo
14: Aumenta +1 el número de iteraciones
15: **return** solución x

2.5. Método de Gauss Seidel

A diferencia del método Jacobi, Gauss Seidel sólo depende de un vector, se inicializa con cualquier valor (excepto vector de ceros).

Algorithm 4 Método Gauss-Seidel

Entrada: n: cantidad de filas, matriz A[n][n], b[n], número iteraciones
Salida: Solución x

1: VN: vector nuevo
2: Inicializar vector viejo: $VN[i] = 1{,}0$
3: **Mientras no se supere el número de iteraciones:**
4: **for** $i = 0; i < N; i++$ **do**
5: suma = 0
6: **for** $j = 0; j < N; j++$ **do**
7: **if** $i \neq j$ **then**
8: suma += $a[i][j] * VN[j]$
9: $VN[i] = \frac{b[i]-suma}{a[i][i]}$
10: Aumenta +1 el número de iteraciones
11: **return** solución x

3. Resultados

3.1. Método de Cholesky

Se utilizaron distintas matrices para probar el método, lo único que se le pide al usuario es que escriba la cantidad de filas de la matriz y el nombre del archivo.

```
Matriz a descomponer:
  4.000000    -1.000000    1.000000
  -1.000000    4.250000    2.750000
  1.000000    2.750000    3.500000

Resultado y traspuesta
  2.000000    0.000000    0.000000
  -0.500000    2.000000    0.000000
  0.500000    1.500000    1.000000

  2.000000    -0.500000    0.500000
  0.000000    2.000000    1.500000
  0.000000    0.000000    1.000000
 Tiempo en segundos: 0.000000.
```

Figura 1: Para una matriz de 3x3.

```
Matriz a descomponer:
  10.000000    -1.000000    2.000000    0.000000
  -1.000000    11.000000    -1.000000    3.000000
  2.000000    -1.000000    10.000000    -1.000000
  0.000000    3.000000    -1.000000    8.000000

Resultado y traspuesta
  3.162278    0.000000    0.000000    0.000000
  -0.316228    3.301515    0.000000    0.000000
  0.632456    -0.242313    3.088897    0.000000
  0.000000    0.908674    -0.252458    2.666567

  3.162278    -0.316228    0.632456    0.000000
  0.000000    3.301515    -0.242313    0.908674
  0.000000    0.000000    3.088897    -0.252458
  0.000000    0.000000    0.000000    2.666567
 Tiempo en segundos: 0.000000.
```

Figura 2: Para una matriz de 4x4.

3.2. Aplicación del método de Cholesky a una matriz tridiagonal

Se resolvió el sistema de la ec. 11 aplicando las condiciones de frontera, esto se hizo para distinta cantidad de nodos, N = 10, 50, 100 para poder observar cómo mejoraba su precisión a la solución exacta. En la fig. 3 se muestran los resultados.

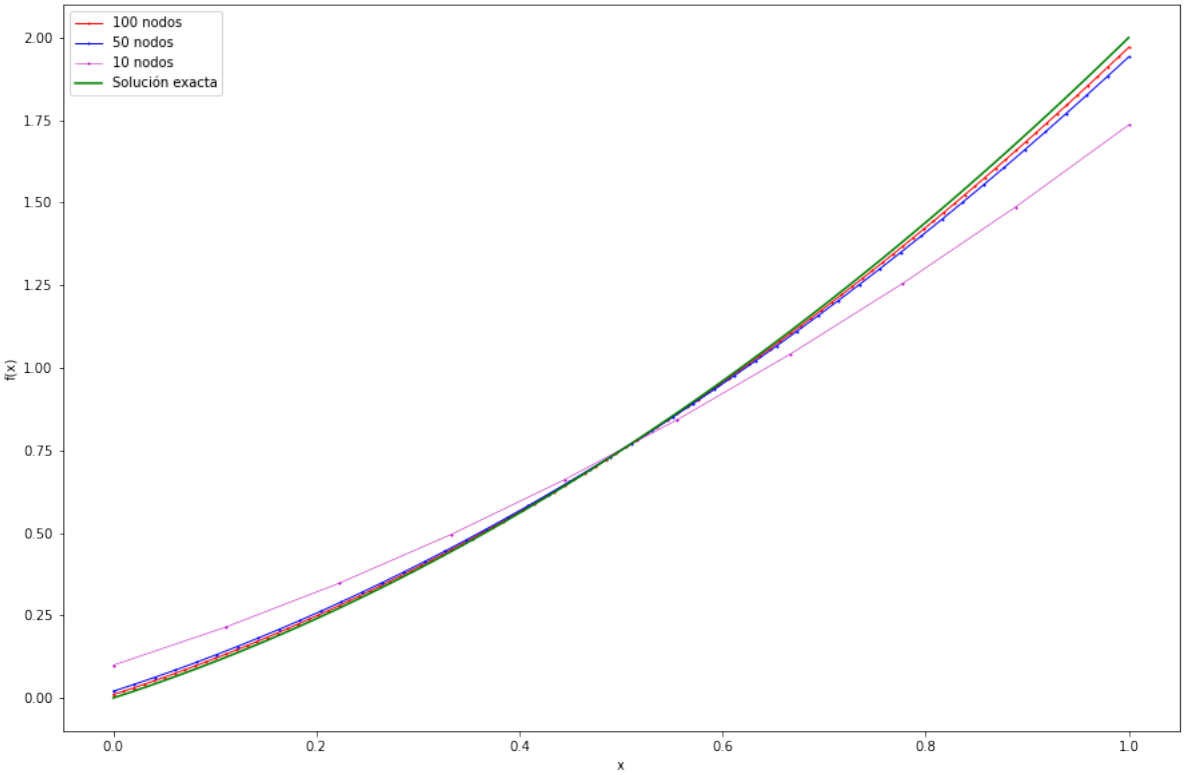


Figura 3: Aplicación de la factorización de Cholesky a un caso específico de matriz.

Con los tiempos de ejecución:

```
0.099174
0.214876
0.347107
0.495868
0.661157
0.842975
1.041322
1.256198
1.487603
1.735537
Segundos: 0.002000
```

(a) 10 nodos

```
1.450211
1.501730
1.554018
1.607074
1.660900
1.715494
1.770857
1.826990
1.883891
1.941561
Segundos: 0.006000
```

(b) 50 nodos

```
1.712773
1.740614
1.768650
1.796883
1.825311
1.853936
1.882757
1.911773
1.940986
1.970395
Segundos: 0.020000
```

(c) 100 nodos

Figura 4: Método de Cholesky con sustitución hacia atrás y hacia adelante.

3.3. Método de Jacobi

Se utilizó el sistema para comprobar los resultados de Jacobi:

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & 1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

y se obtuvo de resultado:

```
0.999942
2.000085
-1.000068
1.000109
Tiempo en segundos: 0.000000.
```

Figura 5: Resultados utilizando Jacobi

que tiene como solución exacta x = (1, 2, -1, 1).

3.4. Método Gauss Seidel

Se utilizó el mismo sistema de ecuaciones que el del método Jacobi y se obtuvo:

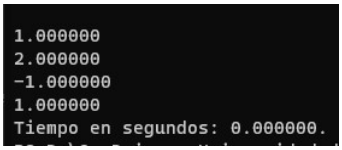


Figura 6: Resultados al ejecutar 10 veces para Gauss Seidel

3.5. Considerando 10,000 nodos

Se consideró el caso en el que se tuviera una matriz de 10000 por 10000, con 20 iteraciones como máximo para Gauss Seidel y Jacobi, y se compararon con los resultados obtenidos utilizando el método de Cholesky.

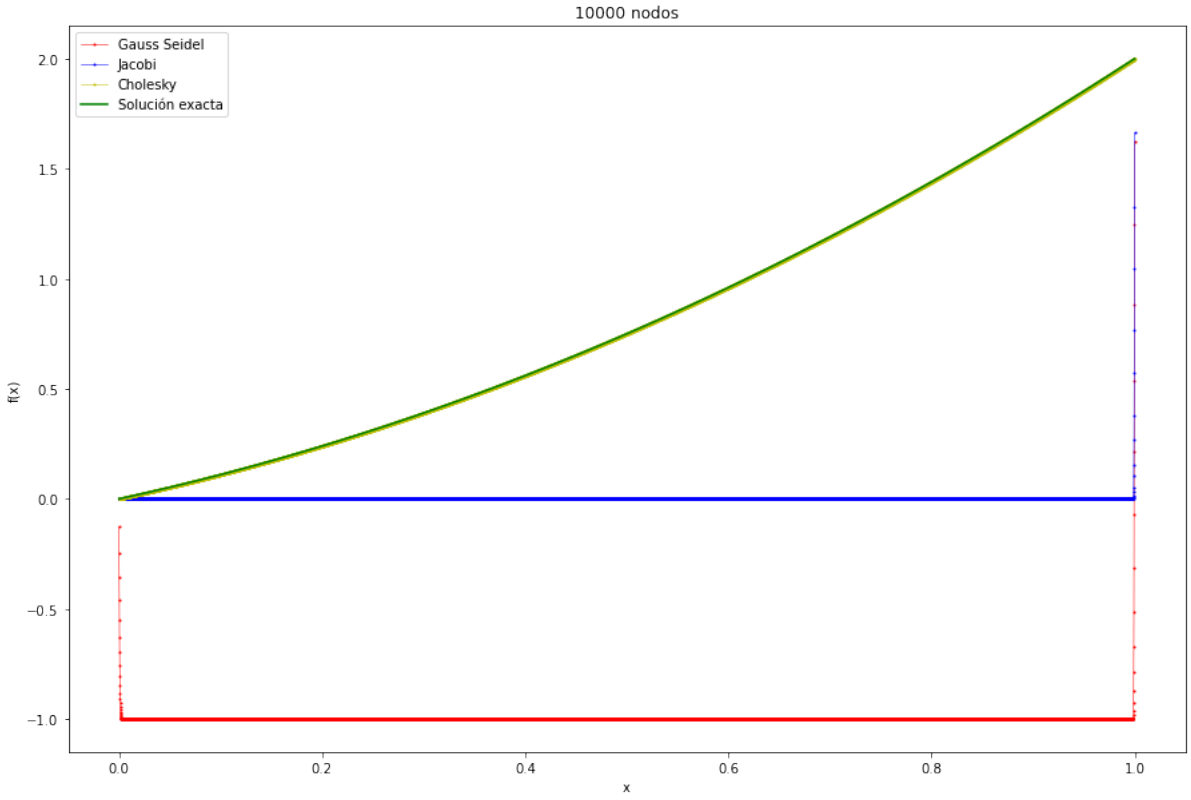


Figura 7: Ejecución con 10000 nodos con los tres métodos

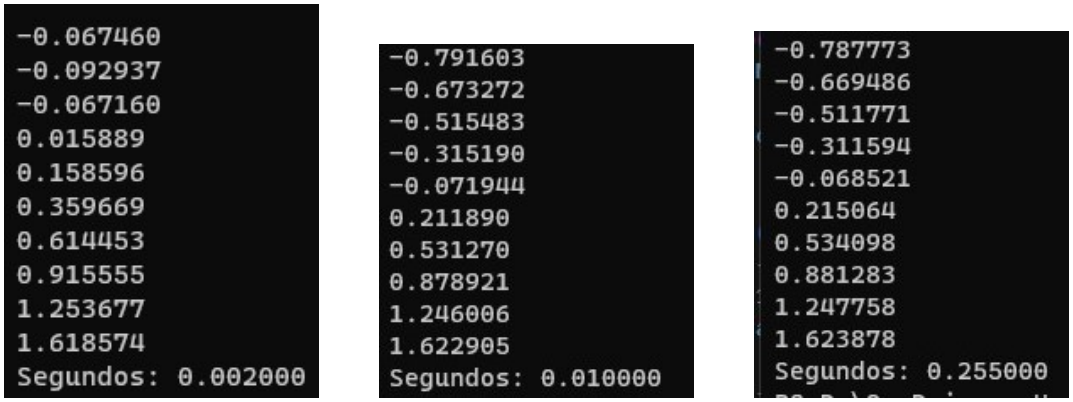
3.6. Comparación entre los tres métodos para distintos nodos.

En el Cuadro 1. se muestran los resultados de los errores promedio entre la solución analítica conocida y los obtenidos con los métodos, como se observa, la factorización con Cholesky mostró mejores resultados, ya que el error de los otros métodos aumentó en lugar de disminuir.

n	Gauss	Jacobi	Método analítico (Cholesky)
10	0.1629	0.1295	0.015
100	3.1338	0.8668	0.0001
1000	3.6407	1.0156	1.53E-06

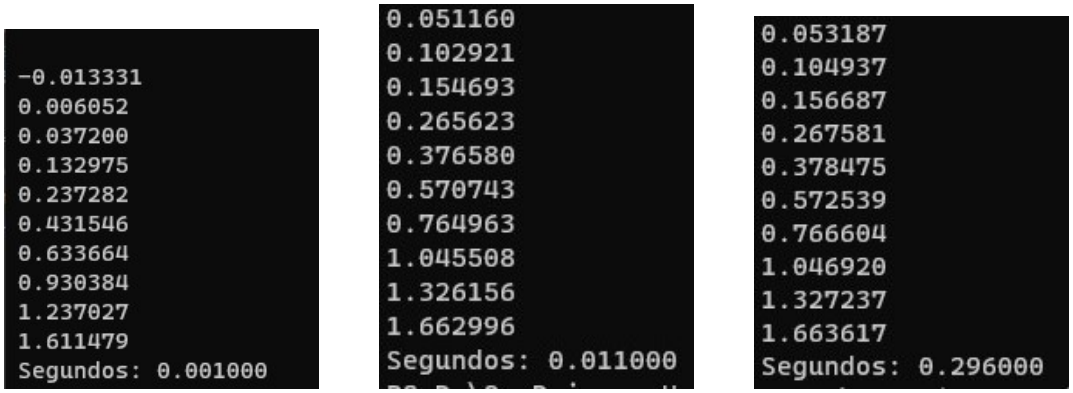
Cuadro 1: Errores promedio

Además, se muestran los tiempos de ejecución:



(a) 10 nodos (b) 100 nodos (c) 1000 nodos

Figura 8: Método Gauss-Seidel



(a) 10 nodos (b) 100 nodos (c) 1000 nodos

Figura 9: Método Jacobi

4. Conclusiones

Los métodos iterativos mostraron mejores resultados en tiempo en comparación con la factorización LU. Sin embargo, aunque convergieron al final, tuvieron una mayor diferencia de resultados entre las iteraciones.

En este caso, ambos métodos iterativos iban a converger porque la matriz que se estudió es diagonalmente dominante, ya que el término de la diagonal es mayor en todas las filas que se generen. Además, al compararlo con el método de Cholesky junto con la sustitución hacia atrás, se mostraron mejores resultados. Pero al hacer el mismo análisis para una matriz de 10 nodos (fig. 10), los resultados no difieren tanto, y se muestra su convergencia al valor de f(x) en el último punto.

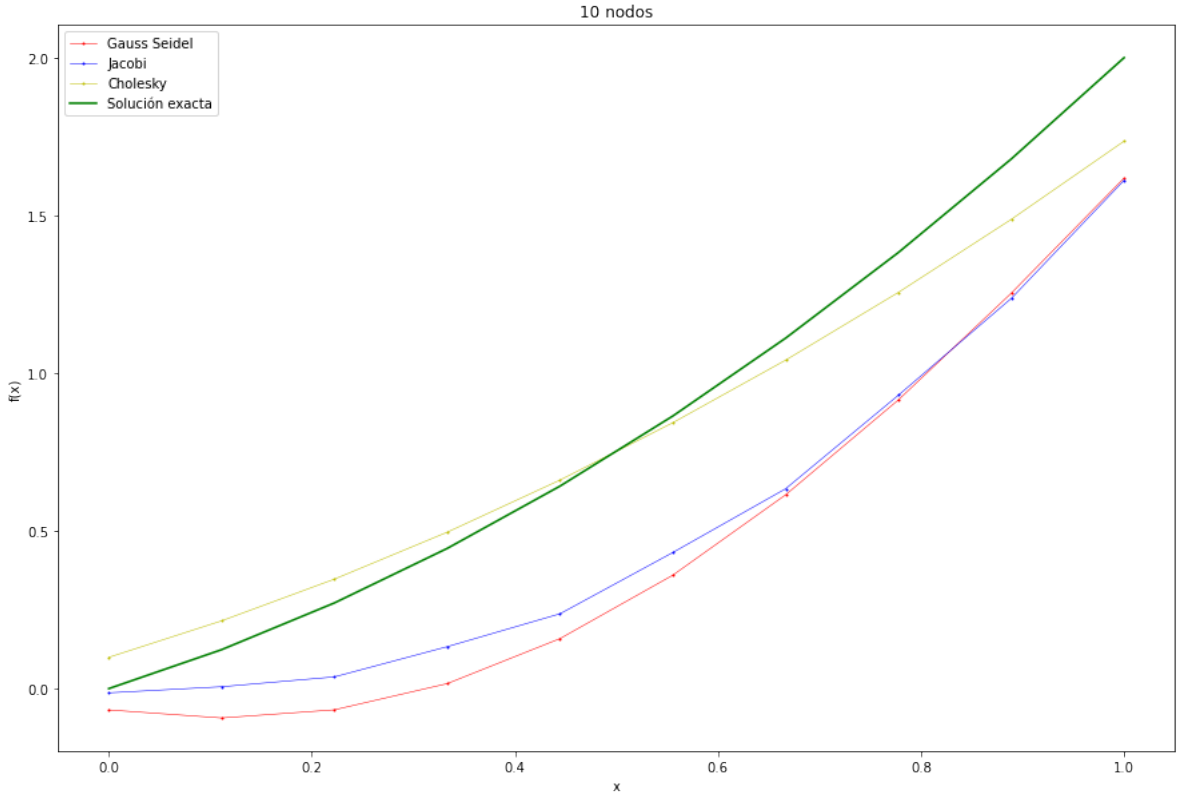


Figura 10: Resultados para 10 nodos.