Primer Examen Métodos Numéricos

Carlos Giovanny Encinia González

21 de octubre 2021

Resumen

El siguiente trabajo es un reporte del primer examen parcial, se muestra la aplicación de diferentes métodos vistos en clases.

Desarrollo

Problema 1

Sea $f : \mathbb{R} \to \mathbb{R}$ una función continua. Se dice que una solución $\hat{x}$ de la ecuación $f(x) = 0$ es un **cero de multiplicidad** $m \geq 1$ de f, si para $x = \hat{x}$ podemos escribir $f(x) = (x - \hat{x})^m q(x)$, donde $q(\hat{x}) \neq 0$.

1. Muestra que si $f \in C^1[a, b]$ tiene un cero simple en $\hat{x} \in (a, b)$ entonces $f'(\hat{x}) \neq 0$.

   Nota: los **ceros simples** tienen multiplicidad uno, es decir $m = 1$ en la definición anterior

2. También se cumple el recíproco del inciso (a), es decir: Si $f \in C^1[a, b]$, $\hat{x} \in (a, b)$, $f(\hat{x}) = 0$ y $f'(\hat{x}) \neq 0$ entonces $\hat{x}$ es un cero simple de f.

   Basado en esta afirmación, muestra que si la función $f(x)$ tiene un cero $\hat{x}$ de multiplicidad $m > 1$, entonces $\hat{x}$ es un cero simple de la función

   $$h(x) = \frac{f(x)}{f'(x)}$$

3. Por lo tanto, usando el inciso (b), podemos aplicar el método de Newton a la función $h(x)$ para encontrar ceros de multiplicidad $m > 1$ de la función f (método de Newton modificado). Aplique el método de Newton a la función $h(x)$ y escriba la Regla de actualización del algoritmo de Newton en términos de f y/o derivadas de f (asumimos por tanto que $h(x)$ es diferenciable y podemos calcular $f''(x)$).

   Implemente el método de Newton modificado y úselo para encontrar una solución aproximada de la ecuación

   $$f(x) = 0$$

donde $f(x) = e^{2x-2} - 2x + 1$. Comentario: $\hat{x} = 1$ es un cero de multiplicidad 2 de f.

Nota:

Usar los siguientes criterios de paro en el Algoritmo de Newton modificado:

(a) El error absoluto entre dos posiciones consecutivas, ie,

$$|x_{k+1} - x_k| < tol_x$$

(b) El valor absoluto de la $f(x)$, ie,

$$|f(x_{k+1})| < tol_f$$

Parámetros de Entrada:

(a) Apuntador a la función $f(x)$

(b) Toleracia tol_x

(c) Toleracia tol_f

(d) Punto Inicial x_0

Salida por Pantalla: Donde se muestre la siguiente Tabla: la iteración k, el iterado x_k y $|f(x_k)|$, es decir:

Iteración (k)	x_k	$\|f(x_k)\|$
1		
2		
3		
...		

Respuesta

Para la parte 1, partiremos de:

$$f(x) = (x - \hat{x})^m q(x) \tag{1}$$

Si sabemos que tenemos multiplicidad m=1, entonces nos queda que:

$$f(x) = (x - \hat{x})q(x) \tag{2}$$

Ahora, para seguir con la demostración, debemos de obtener la derivada con respecto a x de la ecuación anterior y entonces, tenemos que:

$$\frac{d[f(x)]}{dx} = \frac{d[(x - \hat{x})q(x)]}{dx} \tag{3}$$

Utilizando la regla de la cadena nos queda que:

$$\frac{d[f(x)]}{dx} = \frac{d[(x-\hat{x})]}{dx} q(x) + (x-\hat{x}) \frac{d[q(x)]}{dx} \tag{4}$$

Realizando las derivadas y cambiando de notación, podemos escribir que:

$$f'(x) = q(x) + (x-\hat{x}) q'(x) \tag{5}$$

Ahora evaluando en $x = \hat{x}$, tenemos que:

$$f'(\hat{x}) = q(\hat{x}) + (\hat{x}-\hat{x}) q'(\hat{x}) \tag{6}$$

Realizando la resta en el segundo termino nos queda que:

$$f'(\hat{x}) = q(\hat{x}) \tag{7}$$

Pero sabemos que $q(\hat{x}) \neq 0$, Por lo que:

$$\boxed{f'(\hat{x}) \neq 0} \tag{8}$$

Para la parte 2, tenemos que:

$$h(x) = \frac{f(x)}{f'(x)} \tag{9}$$

y queremos demostrar que esa h(x) tiene un cero simple, con base a lo que se define en el apartado 2.

Debemos demostrar entonces que $f \in c^1[a,b]$, $\hat{x} \in (a,b)$, $f(\hat{x}) = 0$ y $f'(\hat{x}) \neq 0$

Primero escribamos las formas que deben de tener f(x) y su derivada

$$f(x) = (x-\hat{x})^m q(x) \tag{10}$$

Y su derivada aplicando regla de la cadena es:

$$f'(x) = m(x-\hat{x})^{m-1} q(x) + (x-\hat{x})^m q'(x) \tag{11}$$

Sustituyendo (11) y (10) en (9) teneos que:

$$h(x) = \frac{(x-\hat{x})^m q(x)}{m(x-\hat{x})^{m-1} q(x) + (x-\hat{x})^m q'(x)} \tag{12}$$

Si en el denominador sacamos de factor común $(x-\hat{x})^m$ nos queda que:

$$h(x) = \frac{(x-\hat{x})^m q(x)}{(x-\hat{x})^m [m(x-\hat{x})^{-1} q(x) + q'(x)]} \tag{13}$$

Ahora multiplicando por $(x-\hat{x})^{-1}$ nos queda que:

Ahora si utilizamos la definición de limite para evaluar cuando x tiende a $\hat{x}$ tenemos que:

$$\lim_{x \to \hat{x}} h(x) = \left. \frac{(x-\hat{x})^m q(x)}{(x-\hat{x})^m [m(x-\hat{x})^{-1} q(x) + q'(x)]} \right|_{x=\hat{x}} \tag{14}$$

Eliminando el factor común tenemos que:

$$\lim_{x \to \hat{x}} h(x) = \left. \frac{q(x)}{m(x-\hat{x})^{-1} q(x) + q'(x)} \right|_{x=\hat{x}} \tag{15}$$

Multiplicando el denominador por $(x - \hat{x})$ tenemos que:

$$\lim_{x \to \hat{x}} h(x) = \frac{q(x)(x - \hat{x})}{mq(x) + q'(x)(x - \hat{x})} \tag{16}$$

Y evaluando tenemos que:

$$\lim_{x \to \hat{x}} h(x) = \frac{q(\hat{x})(\hat{x} - \hat{x})}{mq(\hat{x}) + q'(\hat{x})(\hat{x} - \hat{x})} \tag{17}$$

$$\lim_{x \to \hat{x}} h(x) = \frac{0}{mq(\hat{x})} \tag{18}$$

$$\boxed{\lim_{x \to \hat{x}} h(x) = 0} \tag{19}$$

Ahora tenemos que demostrar que $h'(\hat{x}) \neq 0$
Derivando h(x) tenemos que:

$$h'(x) = \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} \tag{20}$$

Para esto primero vamos a encontrar la segunda derivada de f(x), si sabemos que es un polinomio entonces debe de ser continua y derivable para la segunda derivada.

$$f''(x) = m(m+1)(x - \hat{x})^{m-2}q(x) + 2m(x - \hat{x})^{m-1}q'(x) + (x - \hat{x})^m q'(x) \tag{21}$$

Ahora calcularemos $f'(x)^2$

$$f'(x)^2 = m^2(x - \hat{x})^{m-1}(x - \hat{x})^{m-1}q(x)^2 + (x - \hat{x})^m(x - \hat{x})^m q'(x)^2 + \\ 2m(x - \hat{x})^m(x - \hat{x})^{m-1}q(x)q'(x) \tag{22}$$

Ahora sustituyendo en (20) tenemos que:

$$h'(x) = \{m^2(x - \hat{x})^{m-1}(x - \hat{x})^{m-1}q(x)^2 + (x - \hat{x})^m(x - \hat{x})^m q'(x)^2 + \\ 2m(x - \hat{x})^m(x - \hat{x})^{m-1}q(x)q'(x) - \\ (x - \hat{x})^m q(x)[m(m+1)(x - \hat{x})^{m-2}q(x) + 2m(x - \hat{x})^{m-1}q'(x) + (x - \hat{x})^m q'(x)]\} \\ \div \\ \{m^2(x - \hat{x})^{m-1}(x - \hat{x})^{m-1}q(x)^2 + (x - \hat{x})^m(x - \hat{x})^m q'(x)^2 + \\ 2m(x - \hat{x})^m(x - \hat{x})^{m-1}q(x)q'(x)\} \tag{23}$$

Ahora los siguientes cálculos se realizan pensando en:

$$\lim_{x \to \hat{x}} h'(x) \tag{24}$$

Si sacamos como factor común $(x - \hat{x})^m(x - \hat{x})^{m-1}$ nos queda que:

$$h'(x) = \frac{(x-\hat{x})^m(x-\hat{x})^{m-1}}{(x-\hat{x})^m(x-\hat{x})^{m-1}}\{m^2(x-\hat{x})^{-1}q(x)^2+(x-\hat{x})q'(x)^2+2mq(x)q'(x)-$$
$$q(x)[m(m+1)(x-\hat{x})^{-1}q(x)+2mq'(x)+(x-\hat{x})q'(x)]\}$$
$$\div$$
$$\{m^2(x-\hat{x})^{-1}q(x)^2+(x-\hat{x})q'(x)^2+2mq(x)q'(x)\} \quad (25)$$

Ahora si multiplicamos por $(\hat{x}-\hat{x})$, el numerador y denominador nos queda que:

$$h'(x) = \frac{(x-\hat{x})^m(x-\hat{x})^{m-1}(x-\hat{x})}{(x-\hat{x})^m(x-\hat{x})^{m-1}(x-\hat{x})}\{m^2q(x)^2+(x-\hat{x})^2q'(x)^2+$$
$$2mq(x)q'(x)-$$
$$q(x)[m(m+1)q(x)+2mq'(x)+(x-\hat{x})^2q'(x)]\}$$
$$\div$$
$$\{m^2q(x)^2+(x-\hat{x})^2q'(x)^2+2mq(x)q'(x)\} \quad (26)$$

Ahora si tenemos que:

$$\lim_{x\to\hat{x}} h'(x) = \{m^2q(\hat{x})^2+(\hat{x}-\hat{x})^2q'(\hat{x})^2+2mq(\hat{x})q'(\hat{x})-$$
$$q(\hat{x})[m(m+1)q(\hat{x})+2mq'(\hat{x})+(\hat{x}-\hat{x})^2q'(\hat{x})]\}$$
$$\div$$
$$\{m^2q(\hat{x})^2+(\hat{x}-\hat{x})^2q'(\hat{x})^2+2mq(\hat{x})q'(\hat{x})\} \quad (27)$$

Simplificando nos queda que:

$$\lim_{x\to\hat{x}} h'(x) = \{m^2q(\hat{x})^2+2mq(\hat{x})q'(\hat{x})-$$
$$q(\hat{x})[m(m+1)q(\hat{x})+2mq'(\hat{x})]\}$$
$$\div$$
$$\{m^2q(\hat{x})^2+2mq(\hat{x})q'(\hat{x})\} \quad (28)$$

Ahora si sabemos que q(x) es la factorizacion de los binomios que representan los demás ceros de la función entonces sus derivadas deben ser distintas a cero al evaluarlas en $\hat{x}$ y además debe ser continua en su segunda derivada. Sabiendo esto entonces podemos escribir que:

$$\lim_{x\to\hat{x}} h'(x) = 1 - \frac{(m^2+m)q(\hat{x})^2+2mq(\hat{x})q'(\hat{x})}{m^2q(\hat{x})^2+2mq(\hat{x})q'(\hat{x})} \quad (29)$$

Simplificando nos queda que:

$$\lim_{x\to\hat{x}} h'(x) = 2 - \frac{q(\hat{x})}{mq(\hat{x})+2q'(\hat{x})} \quad (30)$$

Entonces si sabemos que q(x) son los demás factores de la función, estos deben de ser distintos de cero, ya que si estos son cero, también deben ser parte de la multiplicidad, entonces tenemos que:

$$\boxed{\lim_{x \to \hat{x}} h'(x) \neq 0} \tag{31}$$

Entonces si $\hat{x} \in (a, b)$ y la función es continua en [a, b], entonces

Existe un cero simple para h(x)

Ahora la rregla de actualizacion deberia estar definida por la h(x), teniendo asi que:

$$x_{i+1} = x_i - \frac{h(x_i)}{h'(x_i)} \tag{32}$$

entonces sustituyendo por f(x) tenemos que:

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{f'(x_i)^2 - f(x_i)f''(x_i)} \tag{33}$$

Entonces el algoritmo quedaria como:

Algorithm 1: Método de Newton modificado

Newton_mod $(floatx_0, float(*function)(float), float(*f_p)(float), float(*f\_pp)(float);$

Input : se tiene que ingresar la función, la derivada de la función y la segunda derivada de la función, además de una aproximación

Output: $eigenvalor, eigenvector$

while $CRITERIA(x\_before, x\_0) > EPSILON$ *and* $fabs(function(x\_0)) > EPSILON$ *and* $i < LIMIT$ **do**

  if *f_p(x) = ZERO* **then**

    break;

  end

  $x\_before = x\_0;$

  $x\_0- = f(x_0)f'(x_0)/f'(x_0)^2 - f(x_0)f''(x_0);$

  $i++;$

end

return(x_0)

Para nuestro caso tenemos que:

$$f(x) = e^{2x-2} - 2x + 1 \tag{34}$$

$$f'(x) = 2e^{2x-2} - 2 \tag{35}$$

$$f''(x) = 4e^{2x-2} \tag{36}$$

Problema 2

Para este método, utilizare la factorización LU, la otra opción es usar Cholesky pero la matriz no es simétrica.

Una vez que tengo la factorizacion LU, para poder encontrar la inversa puedo hacer

$$L * (x_1, y_1, z_1, ...)^T = (1, 0, 0)^T \tag{37}$$

$$L * (x_2, y_2, z_2, ...)^T = (0, 1, 0)^T \tag{38}$$

$$L * (x_3, y_3, z_3, ...)^T = (0, 0, 1)^T \tag{39}$$

$$\vdots \tag{40}$$

y análogo para calcular La inversa de U. Es importante mencionar que los vectores que $(x_i, y_i, z_i, ...)$, pertenecen a las columnas de la inversa. Es decir tendremos que resolver el sistema de ecuaciones 1000 veces para cada matriz (L y U).

Una vez teniendo la inversa tenemos que:

$$A = LU \rightarrow U^{-1}L^{-1}A = I \tag{41}$$

Multiplicando por la inversa de A por la derecha tenemos que:

$$U^{-1}L^{-1} = A^{-1} \tag{42}$$

Por lo que después de calcular la inversa de cada matriz factor de A, debemos realizar su producto punto.

De esta manera calculamos la inversa. Simplemente necesitamos nuestro programas de factorizacion LU, solución de ecuaciones a matrices triangulares y funciones ya programadas que trabajan con álgebra de matrices.

Para el calculo de los valores que solucionan la ecuación. Utilizare el método de Gauss con pivoteo, es un método algo lento pero al ser un método directo no hay tanto error, adamas que parece que los elementos mas grandes están sobre la diagonal de la matriz, así que eso hará que le método encuentre la solución mas rápido.

Problema 3

Para este problema, compare únicamente cuatro métodos, dos de ellos son métodos directos, y los otros dos son métodos iterativos.

Se tuvo que programar la lectura de los archivos ya que cambiaban con respecto a lo que ya se venia manejando, y muchos de los programas tuvieron que ser modificados. El concepto general de los métodos no cambio en nada. En la sección de Resultados se muestra la comparación y algunos de los resultados obtenidos.

También es importante mencionar, que se creo una función para poder revisar si la matriz es diagonal o no. A simple vista se veía que si era diagonal, pero siempre hay que revisar.

Problema 4

Para el problema 4, se genero primero la matriz mediante un pequeño código, y después se utilizo el método de subespacios para poder calcular los eigenpares que son menores.

Debido a que aun seguía teniendo error en memoria para el método de la potencia, me vi obligado a utilizar el método de Jacobi, este tarda mucho en converger pero se pudo llegar a algo, el error que se le pidió fue de 1E-6, con un máximo de 500000 iteraciones, si cumple con estas iteraciones significa que no pudo converger con el error que se le pidió.

En seguida muestro el pseudocódigo para generar la matriz:

Algorithm 2: Crear matriz problema 4

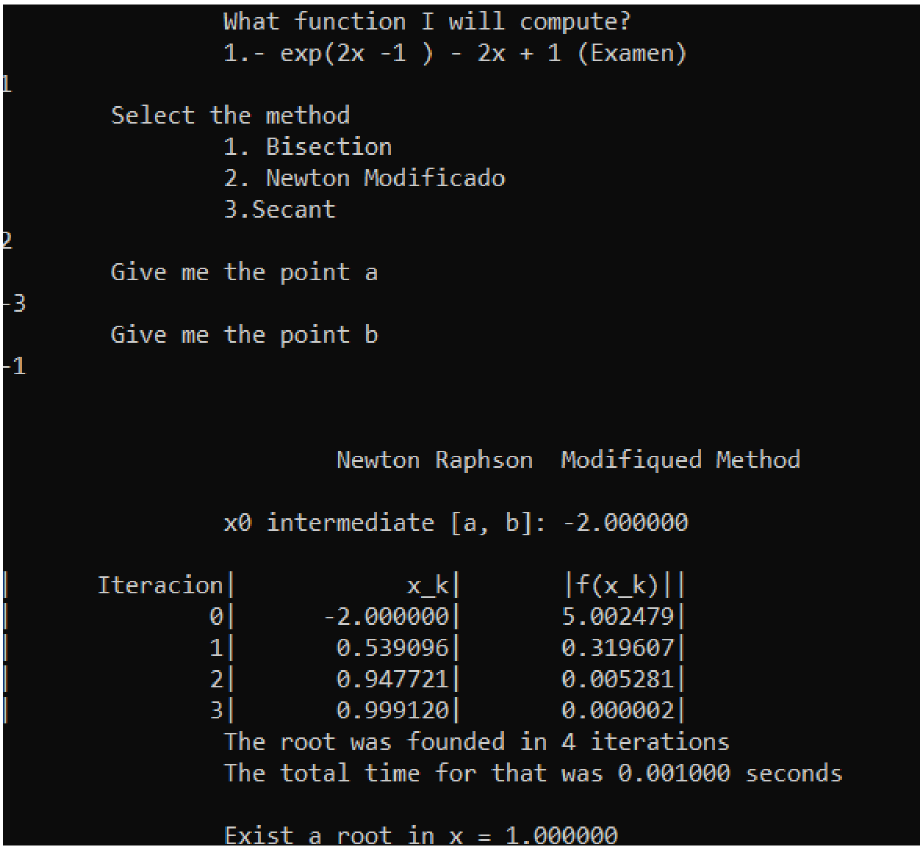
```
crea_matriz ();
Input  : void
Output: eigenvalor, eigenvector
matrix_{0,0} = 40;
matrix_{0,1} = −8;
matrix_{0,2} = −4;
matrix_{1,0} = −8;
matrix_{1,1} = 40;
matrix_{1,2} = −8;
matrix_{1,3} = −4;
for i = 2, i < TAMANIO, i++ do
    k = 0;
    for j = i - 2, j < TAMANIO, j++ do
        if j >= 0 and k < 5 then
            matrix_{i,j} = *(valores + k + +);
        end
    end
end
return matrix; return(x_0)
```

Es importante mencionar que de los dos métodos utilizados, se obtuvieron resultados algo distintos. SE habla mas al respecto en la sección de Resultados.

Resultados

Problema 1

Para el problema 1 tenemos el siguiente resultado:



```
              What function I will compute?
              1.- exp(2x -1 ) - 2x + 1 (Examen)
1
       Select the method
              1. Bisection
              2. Newton Modificado
              3.Secant
2
       Give me the point a
-3
       Give me the point b
-1


                     Newton Raphson  Modifiqued Method

              x0 intermediate [a, b]: -2.000000

|      Iteracion|            x_k|       |f(x_k)||
|              0|      -2.000000|       5.002479|
|              1|       0.539096|       0.319607|
|              2|       0.947721|       0.005281|
|              3|       0.999120|       0.000002|
              The root was founded in 4 iterations
              The total time for that was 0.001000 seconds

              Exist a root in x = 1.000000
```

Problema 2

En seguida se muestran los resultados obtenidos. De cualquier manera, los resultados se encuentran en la carpeta Problema_2_sistemaecuacioens/Resultados/

Figure 1: Parte de la solución al sistema

```
1000 1
 0.001303
 0.000752
-0.000173
 0.000755
 0.000021
 0.000065
-0.000370
 0.000538
-0.000368
-0.000098
 0.000547
 0.000871
 0.000248
 0.001385
 0.000189
 0.000714
 0.001107
 0.000306
 0.000191
-0.000170
 0.000325
 0.001153
 0.000924
-0.000072
 0.000695
-0.000212
 0.000499
 0.000146
 0.000200
```

Figure 2: Parte de la Inversa de A

```
1000 1000
0.00001869993 -0.00000000970 -0.00000001431 -0.00000000646 0.00000000793 -0.00000000917
-0.00000000006 -0.00000000223 -0.00000000285 0.00000000614 -0.00000000817 0.00000000775 0.00000000563
-0.00000000122 -0.00000000614 -0.00000001930 -0.00000001921 -0.00000001979 -0.00000000212
0.00000000231 -0.00000002421 -0.00000000640 -0.00000002214 0.00000000161 -0.00000001292
-0.00000001337 -0.00000000400 -0.00000000514 -0.00000002321 -0.00000002181 0.00000000410
-0.00000000615 -0.00000001328 -0.00000000900 -0.00000002164 0.00000000064 -0.00000000664
-0.00000002424 -0.00000001203 -0.00000002032 0.00000000403 -0.00000001030 -0.00000002489
0.00000000603 -0.00000000686 -0.00000001610 -0.00000000725 0.00000000116 -0.00000001235
-0.00000002224 -0.00000000073 -0.00000002231 -0.00000001778 0.00000000259 0.00000000153 0.00000000056
-0.00000000422 -0.00000001761 0.00000000468 -0.00000001020 -0.00000000657 -0.00000000093
-0.00000001910 -0.00000000586 -0.00000001741 -0.00000001071 -0.00000000646 -0.00000002012
0.00000000537 0.00000000419 -0.00000000163 0.00000000743 -0.00000000993 -0.00000000736 -0.00000000640
-0.00000002362 0.00000000487 -0.00000000349 -0.00000001209 -0.00000001348 -0.00000000900
0.00000000153 0.00000000015 -0.00000000369 -0.00000002377 0.00000000685 -0.00000001386 -0.00000000625
-0.00000000956 0.00000000364 -0.00000000511 0.00000000721 -0.00000001651 0.00000000657 -0.00000002222
-0.00000001909 -0.00000002083 -0.00000001462 -0.00000002408 -0.00000000710 -0.00000000892
-0.00000002398 0.00000000318 -0.00000000647 -0.00000000187 -0.00000001218 -0.00000001955
-0.00000001448 -0.00000002017 -0.00000001847 -0.00000001026 -0.00000001782 -0.00000002292
-0.00000001392 -0.00000001893 -0.00000001212 -0.00000001542 -0.00000002311 0.00000000357
-0.00000002342 -0.00000002283 -0.00000002378 -0.00000000175 -0.00000000289 0.00000000217
-0.00000001327 -0.00000001514 0.00000000525 -0.00000001461 -0.00000000255 0.00000000433
-0.00000000759 -0.00000002383 0.00000000448 -0.00000001899 0.00000000353 -0.00000000990
-0.00000001694 -0.00000001256 -0.00000001023 0.00000000118 0.00000000053 -0.00000001354
-0.00000000815 0.00000000041 -0.00000001995 -0.00000002400 -0.00000000712 -0.00000001624
0.00000000289 -0.00000001059 0.00000000419 -0.00000001230 -0.00000002283 -0.00000001232
-0.00000000382 0.00000000197 0.00000000630 -0.00000000516 -0.00000000649 -0.00000002382
-0.00000000794 -0.00000002432 -0.00000000827 -0.00000002073 -0.00000001237 -0.00000000340
-0.00000002596 -0.00000001770 -0.00000002600 0.00000000404 -0.00000001563 -0.00000001788
-0.00000001176 -0.00000002104 0.00000000136 -0.00000000365 0.00000000133 -0.00000001567 0.00000000815
```

Problema 3

En 3 de los 4 métodos como resultado nos di un vector lleno de 1's, es lo que se esperaba, aunque el vector b, cambia un poco en sus últimos decimales, estos números son muy pequeños, tal vez si en los programas hubiese utilizado un tipo de dato como long float pudiese haber visto los resultados alrededor de 1, dando un respuesta mas precisa.

Gauss: El método de Gauss Pivote total es el que mas tiempo se tardo, esto es debido a que tiene siempre que buscar un elemento grande para poder realizar el pivoteo, con matrices muy grandes el método se va haciendo mas lento, en cuanto a la precisión el resultado obtenido es igual que los demás. Al ser un método directo, no puedo darle una tolerancia de error, y el numero de iteraciones depende del tamaño del archivo, tiene una complejidad cubica, así que es de los peores métodos, pero es eficiente si se quiere evitar errores de punto flotante.

Jacobi: El método de Jacobi fue muy rápido, tal vez es debido a que el vector con el que comienza a iterar es cercano al vector de la solución, pero por lo general este método tiende a ser tardado, no es de mis favoritos, en esta ocasión tardo solo 2 iteraciones para encontrar la respuesta, de igual manera no es el método mas óptimo para poder llegar a una solución, el programa inicializa el vector inicial con un 1 en su primera fila. Tomo unos cuantos segundos en converger.

Gauss Seidel: Este es uno de mis métodos iterativos favoritos, pero en esta ocasión dio un resultado totalmente incorrecto, es posible que sea debido a algún error cuando se comparaba la tolerancia con el error, de cualquier manera dio una respuesta rápida en 2 iteraciones, pero como ya mencione, totalmente equivocada.

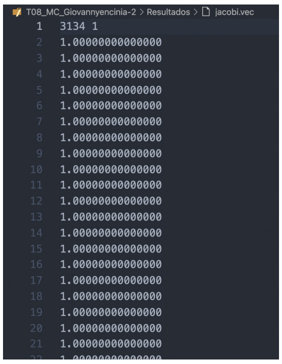
Diagonal: Antes de aplicar este método cree un algoritmo que me dice si en verdad es diagonal la matriz, en esta ocasión así lo fue, así que fácilmente pude aplicar el método de la diagonal, este tiene una complejidad lineal, además de que hace solo una operación importante, que es una división, fue el método mas rápido, es el método directo mas óptimo para este tipo de matrices.

Figure 3: Diagonal

T08_MC_Giovannyencinia-2 > Resultados > diagonal.vec

```
3134 1
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
```

Figure 4: Jacobi



T08_MC_Giovannyencinia-2 > Resultados > jacobi.vec

```
3134 1
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
1.00000000000000
```

Figure 5: Gauss Pivote

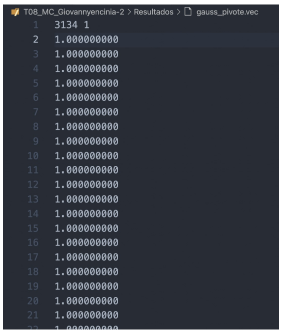
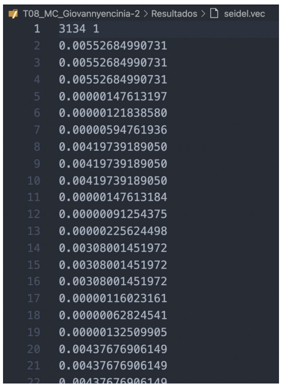


Figure 6: Seidel



Problema 4

Todos los resultados se encuentran en Problema_4_valorespropios/Resultados

Primero calcule los eigenpares menores con el método del subespacio, el cual utiliza la potencia inversa, en seguida muestro el resultado:

Figure 7: Subespacio-potenciaInversa

```
10 1
51.2300216443
51.0880029857
49.5826961557
48.7405689182
47.6411717750
42.7271059352
35.7214545568
28.1634679974
21.6682359164
17.4401797367
```

Debido a que no pude implementar correctamente el método de la potencia junto con el del subespacio, evite usarlo (problemas con la memoria dinámica), entonces utilice el método de Jacobi, aunque es un método lento me pareció que aun podía calcular los eigenpares. Para mi sorpresa, no fue asi. Tarde casi una hora, para una precisión de solo el 1E-6, y aun así no convergió del todo el método, pero aunque le diera mas tiempo no parecía llegar a algo mas preciso, en seguida muestro los resultados para los eigenpares mayores y menores.

Figure 8: Jacobi menores

```
16.022815
16.051438
16.019144
16.012762
16.215507
16.133867
16.056085
16.112727
16.029645
16.014245
```

Figure 9: Jacobi mayores

```
2000 1
51.969905
51.981100
51.949688
51.959609
51.983022
51.953193
51.916051
51.841956
51.978463
51.907118
```

Algo interesante es que aunque los resultado son algo distintos, los valores obtenidos por el método del subespacio parecen estar en lo obtenido en el otro método, me hace pensar que tal vez existan multiplicidades, que hagan que los valores se repitan, mediante el método de Jacobi, solo se obtiene un aproximación, a los eigenvalores, no se pudo lograr diagonalizar toda la matriz pero la gran mayoría de los elementos ya tendían a cero, se realizaron 500 000 iteraciones para poder lograr esto.