Tarea 7-Métodos numéricos

1st Daniel Vallejo Aldana
Maestría en Ciencias de la Computación
Centro de Investigación en Matemáticas
daniel.vallejo@cimat.mx

***Resumen*—El presente trabajo se divide en dos partes, la primera borda métodos de encontrar valores y vectores propios para una matriz A usando las técnicas de iteración en subespacio y de iteración en subespacio inverso. Así mismo se implementa el método de Raleigh para el cálculo de un vector propio y su valor propio asociado. La segunda parte de este trabajo se refiera a métodos de solución de sistemas de ecuaciones, en este trabajo se abordan los métodos de descomposición QR, Gradiente conjugado y gradiente conjugado pre condicionado.**

***Index Terms*—Iteración en Subespacio, Iteración en Subespacio Inverso, Método de Raleigh para valores y vectores propios, QR, Gradiente Conjugado, Gradiente Conjugado Pre condicionado**

I. Introducción

El cálculo de valores y vectores propios es un problema principal en métodos numéricos [1]. Como se ha abordado en las tareas anteriores ,es posible encontrar muchas aplicaciones que involucren el cálculo de los vectores y los valores propios, tales como la solución al problema de la ecuación de calor. En este trabajo se implementan y evaluan tres nuevos métodos para calcular los valores propios y los vectores propios de una matriz. Dichos métodos son la iteración en subespacio conocida como método de Krylov, el método de Krylov inverso y el método del cociente de Raleigh.

Al igual que el cálculo de vectores y valores propios, solucionar sistemas de ecuaciones es de vital importancia en muchos problemas hoy en día. En el presente trabajo se implementan y analizan tres métodos nuevos de solución de ecuaciones, el método QR, el método de Gradiente Conjugado y el Método de Gradiente conjugado pre condicionado.

II. Método/Algoritmo

Debido a la extensión de la tarea y a la diversidad temática de la misma, dividiremos los algoritmos de dicha tarea en dos secciones principales, la primera referente a métodos para encontrar los valores propios de una matriz A (Iteración en subespacio, Iteración en Subespacio Inverso y el método de Raleigh) y la segunda sección referente a métodos para solución de ecuaciones de la forma $A\mathbf{x} = \mathbf{b}$ (QR, gradiente conjugado y gradiente conjugado precondicionado).

II-A. Métodos para encontrar valores y vectores propios de una matriz A

II-A1. Iteración en Subespacio: El método de Iteración en subespacio, también conocido como el método de Krylov se basa en encontrar un subconjunto de m pares de vectores y valores asociados de una matriz A, tal que $m \ll M$ con M el número de columnas de la matriz A. Lo anterior por medio de una transformación de la matriz A, en una matriz W cuya dimensión es de $m \times m$, mediante una matriz de rotación V cuya dimensión es de $n \times m$. Lo anterior lo podemos expresar de la siguiente forma.

$$W_{k+1} = V_k^T A V_k$$

Cuya matriz V_0 corresponde a la matriz compuesta por las m primeras columnas de la matriz A, en este caso podemos ver que V_0 puede ser escrito de la siguiente manera

$$V_0 = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{m-1}]$$

No obstante, antes de realizar el producto $W_{k+1} = V_k^T A V_k$, debemos asegurarnos que las columnas de V_i sean ortonormales entre si, para eso, llevamos a cabo el proceso de ortonormalización de Gram-Schmidt descrito a continuación para una colección de vectores $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_i\}$

$$\begin{aligned}
\hat{\mathbf{v}}_0 &= \mathbf{v}_0 & \mathbf{e}_0 &= \frac{\hat{\mathbf{v}}_0}{\|\hat{\mathbf{v}}_0\|} \\
\hat{\mathbf{v}}_1 &= \mathbf{v}_1 - \frac{< \mathbf{v}_0, \mathbf{v}_1 >}{\|\mathbf{v}_0\|^2}\mathbf{v}_0 & \mathbf{e}_1 &= \frac{\hat{\mathbf{v}}_1}{\|\hat{\mathbf{v}}_1\|} \\
&\vdots \\
\hat{\mathbf{v}}_i &= \mathbf{v}_i - \sum_{j=0}^{i-1} \frac{< \mathbf{v}_j, \mathbf{v}_i >}{\|\mathbf{v}_j\|^2}\mathbf{v}_j & \mathbf{e}_i &= \frac{\hat{\mathbf{v}}_i}{\|\hat{\mathbf{v}}_i\|}
\end{aligned}$$

Lo anterior aplicado a la matriz V_i nos da una matriz de vectores ortonormales a la cual denotaremos como $\hat{V}_i = [\mathbf{e}_0, \ldots, \mathbf{e}_{m-1}]$

Una vez que tenemos las columnas de V_k ortonormali-

zadas, procedemos entonces a calcular $W_k = \hat{V}_k^T A \hat{V}_k$, a dicha matriz W_k le aplicaremos el método de Jacobi (En el caso del presente trabajo usamos pocas iteraciones, ya que la mayor parte del algoritmo se carga hacia el método de la potencia), lo anterior nos define un producto de la forma

$$W_{k+1} = R_{k+1}^T \Psi_k R_{k+1}$$

Donde Ψ_k contiene los valores propios en la diagonal a medida que el método converge. La actualización de la matriz V_{k+1} se compone del producto

$$V_{k+1} = \hat{V}_k R_{k+1}$$

Y repetimos nuevamente el proceso hasta convergencia. En este caso la condición de paro del algoritmo es

$$\sum_{i=0}^{m-1} |\Psi_{k+1}^{i,i} - \Psi_k^{i,i}| < TOL$$

Es decir, que los valores propios no cambien demasiado en la siguiente iteración. La secuencia de pasos para el método de iteración en subespacio para una matriz A, es la siguiente

1. Tomar $V_0 = [\mathbf{a}_0, \ldots, \mathbf{a}_0]$
2. Ortonormalizar V_0 y obtener $\hat{V}_0$
3. Inicializar $i = 1$
4. Realizar $W_i = \hat{V}_{i-1}^T A \hat{V}_{i-1}$
5. Obtener $W_i = R_i^T \Psi_i R_i$
6. Checar convergencia
7. Si no hay convergencia actualizar $V_i = \hat{V}_{i-1} R_i$
8. Volver al paso 4 hasta convergencia, actualizar i

A continuación, mostramos el pseudocódigo del algoritmo de iteración en subespacio, como las funciones de orthonormalización de Gram-Schmidt y el Método de Jacobi ya fueron descritas en trabajos anteriores, se asumirá que se conoce el funcionamiento de dichos algoritmos para interpretar el pseudocódigo

II-A2. Método de subespacio para encontrar los valores propios más pequeños: Una modificación del método de iteración en subespacio es modificar la ecuación $W_k = \hat{V}_0^T A \hat{V}_0$ para poder calcular los valores propios más pequeños de la matriz A y sus correspondientes vectores propios. En este caso tenemos una ecuación de la forma

$$W_k = \hat{V}_0^T A^{-1} \hat{V}_0$$

No obstante, calcular la inversa de una matriz A, puede resultar bastante costoso. Por lo anterior haremos uso del método de factorización $A = LDU$ donde L y U, son matrices triangulares inferiores y superiores respectivamente y D es una matriz diagonal, notemos que si $T \in Mat_{n\times n}(\mathbb{R})$ es una matriz triangular cuyas

Algorithm 1 Subspace-Iteration

Require: A, m, TOL, $MaxiterJacobi$
Ensure: $V \in Mat_{n\times m}(\mathbb{R})$ $\Psi \in Mat_{m\times m}(\mathbb{R})$ tal que los elementos de la diagonal son los m primeros valores propios de A.
Inicializar $V_0 = [\mathbf{a}_0, \ldots, \mathbf{a}_{m-1}]$
$\epsilon = \infty$
$\Psi^{tmp} = \mathbf{0}$
while $\epsilon > TOL$ **do**
$\hat{V}_0 = Orthonormalize(V_0)$
$W = \hat{V}_0^T A \hat{V}_0$
$\Psi, R = JacobiMethod(W)$
$\epsilon = \sum_{i=0}^{m-1} |\Psi_{i,i} - \Psi_{i,i}^{tmp}|$
$V_0 = V_0 R$
$\Psi = \Psi^{tmp}$
end while
return $\hat{V}_0, \Psi$

entradas de la diagonal son 1's entonces tenemos lo siguiente

$$\begin{bmatrix} 1 & \ldots & 0 & 0 \\ a_{2,1} & 1\ldots & 0 & 0 \\ \vdots & & & 0 \\ a_{n,1} & \ldots & a_{n,n-1} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & \ldots & 0 & 0 \\ -a_{2,1} & 1\ldots & 0 & 0 \\ \vdots & & & 0 \\ -a_{n,1} & \ldots & -a_{n,n-1} & 1 \end{bmatrix}$$

Es decir que la inversa de una matriz triangular cuyas entradas en la diagonal sean 1's es la matriz tal que $a_{i,j} = -a_{i,j}, i \neq j$ e $i < j$ o $i > j$ según sea el caso.

Usando lo anterior, vemos que $A^{-1} = (LDU)^{-1} = U^{-1}D^{-1}L^{-1}$, por lo que sustituyendo lo anterior en el algoritmo previamente definido para iteración en subespacio, obtenemos que la actualización de W_k es de la siguiente forma

$$W_k = \hat{V}_0^T U^{-1} D^{-1} L^{-1} \hat{V}_0$$

Como la matriz A^{-1} no cambia a lo largo de las iteraciones, solo hay que calcular el producto encadenado de matrices $U^{-1}D^{-1}L^{-1}$ una sola vez.

Lo anterior nos deja un pseudocódigo muy similar al descrito anteriormente. A continuación se muestra el pseudocódigo del método de iteración en subespacio para los m valores más pequeños.

II-A3. Método de Raleigh para calcular un valor y vector propio dada una solución inicial: El método de Raleigh es un método iterativo para encontrar un vector propio y su correspondiente valor propio utilizando el coeficiente de Raleigh definido por

$$\mu = \frac{\mathbf{v}^T A \mathbf{v}}{\mathbf{v}^T \mathbf{v}}$$

Algorithm 2 Inverse-Subspace-Iteration

Require: A, m, TOL, $MaxiterJacobi$
Ensure: $V \in Mat_{n\times m}(\mathbb{R})$ $\Psi \in Mat_{m\times m}(\mathbb{R})$ tal que los elementos de la diagonal son los m primeros valores propios de A.
Inicializar
$L, D, U = LDUFACTORIZATION(A)$
$B = U^{-1}D^{-1}L^{-1}$
$V_0 = [\mathbf{b}_0, \ldots, \mathbf{b}_{m-1}]$
$\epsilon = \infty$
$\Psi^{tmp} = \mathbf{0}$
while $\epsilon > TOL$ **do**
$\hat{V}_0 = Orthonormalize(V_0)$
$W = \hat{V}_0^T B \hat{V}_0$
$\Psi, R = JacobiMethod(W)$
$\epsilon = \sum_{i=0}^{m-1} |\Psi_{i,i} - \Psi_{i,i}^{tmp}|$
$V_0 = V_0 R$
$\Psi = \Psi^{tmp}$
end while
return $\hat{V}_0, \Psi$

Donde μ es un valor propio de la matriz A. En el método de Raleigh se da una aproximación inicial al vector propio $\mathbf{v}_0$ y al valor propio asociado μ_0.

Primero resolvemos el sistema de ecuaciones

$$(A - \mu_i I)\mathbf{v}_{i+1} = \mathbf{v}_i$$

Calculamos entonces la actualización del valor propio μ_{i+1}

$$\mu_{i+1} = \mu_i + \frac{1}{\mathbf{v}_{i+1}^T \mathbf{v}_i}$$

En este caso consideramos la condición de paro como

$$\frac{\left\| \mathbf{v}_1 - \frac{<\mathbf{v}_1, \mathbf{v}_0>}{\|\mathbf{v}_0\|}\mathbf{v}_0 \right\|}{\|\mathbf{v}_0\|} < TOL$$

Lo anterior implica que no exista variación entre los vectores propios que se calculan entre una iteración y la siguiente. En este caso estamos considerando $\|\mathbf{v}_i\| = 1$

El pseudocódigo para el método de Raleigh se muestra a continuación

Nótese que este método depende mucho de las condiciones iniciales que se le den al vector inicial y al valor propio inicial, una aproximación a solucionar este problema son los círculos de Greshgorin.

Algorithm 3 Raleigh-Method

Require: A,$\mathbf{v}_0$,μ,$Maxiter$,TOL
Ensure: $\mathbf{v}, \mu$ Un vector propio y su valor propio
$\epsilon = \infty$
while $\epsilon > TOL$ **do**
$Solve((A - \mu I)\mathbf{v}_1 = \mathbf{v}_0)$
$\mu = \mu + \frac{1}{\mathbf{v}_1^T \mathbf{v}_0}$
$\epsilon = \frac{\left\| \mathbf{v}_1 - \frac{<\mathbf{v}_1, \mathbf{v}_0>}{\|\mathbf{v}_0\|}\mathbf{v}_0 \right\|}{\|\mathbf{v}_0\|}$
$\mathbf{v}_0 = \mathbf{v}_1$
end while
return $\mathbf{v}_1, \mu$

II-B. Métodos de solución de sistemas de ecuaciones de la forma $A\boldsymbol{x} = \boldsymbol{b}$

II-B1. Factorización QR: La factorización QR busca descomponer una matriz A en el producto de dos matrices Q y R, tales que $QQ^T = I$ y R es una matriz triangular superior. En el caso de la matriz Q, recordemos el proceso de ortonormalización de la matriz A mediante Gram-Schmidt, dicho procedimiento transforma un conjunto de vectores $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ en un conjunto $\{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$ ortonormales.

Definimos entonces $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$, vemos que $Q^T Q = QQ^T = I$. En el caso de la Matriz R notemos que para $i \leq j$ se tiene

$$r_{i,j} = \mathbf{q}_i \mathbf{a}_j$$

De esta forma para resolver el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ que se convierte en $R\mathbf{x} = Q^T\mathbf{b}$, el cual puede ser resuelto con sustitución hacia atrás, método ya implementado.

En este caso la matriz R, tiene la siguiente forma

$$\begin{bmatrix} < \mathbf{q}_1, \mathbf{a}_1 > & \ldots < \mathbf{q}_1, \mathbf{a}_n > \\ \vdots & \\ 0 & < \mathbf{q}_n, \mathbf{a}_n > \end{bmatrix}$$

El pseudocódigo para el algoritmo de descomposición QR es el siguiente

Algorithm 4 Inner-Product

Require: $\mathbf{v}_1\mathbf{v}_2$
Ensure: σ el producto interior de dos vectores
$\sigma = 0$
for i in $size(\mathbf{v}_1)$ **do**
$\sigma = \sigma + v_1^i * v_2^i$
end for
return σ

II-B2. Método de Gradiente Conjugado: Notemos que dos vectores $\mathbf{u}$ y $\mathbf{v}$ son conjugados repsecto a una matriz A, si $\mathbf{u}^T A\mathbf{v} = 0$. Consideremos un conjunto

Algorithm 5 Decompose QR

Require: A
Ensure: QR tales que $QR = A$
  $Q = Orthonormalize(A)$
  $R \in Mat_{n\times n}(\mathbb{R})$ Inicializada con cero
  for i in 1 to n **do**
    for j in i to n **do**
      $R_{i,j} = InnerProduct(\mathbf{q}_i, \mathbf{a}_j)$
    end for
  end for
  return Q, R

Algorithm 6 Solve using QR

Require: A,**b**
Ensure: **x** tal que $\|A\mathbf{x} - \mathbf{b}\| = 0$
  Q,R=QRDecompose(A)
  $\mathbf{b} = Q^T\mathbf{b}$
  $\mathbf{x} = BackwardSubstitution(R, \mathbf{b})$

de vectores $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, decimos que este conjunto está compuesto de vectores mutuamente conjugados si $\forall i, ji \neq j$ se tiene que $\mathbf{p}_i A\mathbf{p}_j = 0$. Por lo anterior decimos que $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ es una base de $\mathbb{R}^n$ y, por lo tanto, podemos expresar la solución del sistema de ecuaciones **x** respecto a esta base. De lo anterior vemos que el coeficiente α_k está dado de la siguiente manera

$$\alpha_k = \frac{\mathbf{p}_k\mathbf{b}}{\mathbf{p}_k^T A\mathbf{p}_k}$$

Consideremos entonces un residuo de la forma $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$, el residuo de la k-esima iteración, notemos que el criterio de paro para el método de gradiente conjugado es cuando $\|\mathbf{r}_k\| < TOL$ para alguna tolerancia. Usando este residuo podemos calcular el vector $\mathbf{p}_k$ mediante la siguiente actualización

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{p}_i A\mathbf{r}_k}{\mathbf{p}_i A\mathbf{p}_i}\mathbf{p}_i$$

De esta forma vamos construyendo los vectores conjugados $\mathbf{p}_i$ y a su vez construyendo la solución **x**. En este caso actualizamos la solución mediante la siguiente ecuación de actualización

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$$

Donde

$$\alpha_k = \frac{\mathbf{p}_k\mathbf{r}_k}{\mathbf{p}_k^T A\mathbf{p}_k}$$

Lo anterior nos define el siguiente algoritmo de gradiente conjugado descrito en pseudocódigo.

II-B3. Gradiente conjugado pre condicionado: El gradiente conjugado pre condicionado es una variante del gradiente conjugado en donde consideramos una matriz

Algorithm 7 Conjugate-Gradient

Require: A, **b**, $\mathbf{x}_0$,TOL
Ensure: **x** tal que $A\mathbf{x} = \mathbf{b}$
  $\mathbf{r}_0 = b - Ax_0$
  $\mathbf{p}_0 = \mathbf{r}_0$
  $\epsilon = \|r_0\|$
  while $\epsilon > TOL$ **do**
    $\alpha_k = \frac{r_k r_k}{p_k^T A p_k}$
    $x_{k+1} = \alpha_k p_k$
    $r_k = r_k - \alpha_k A p_k$
    $\epsilon = \|r_{k+1}\|$
    if $\epsilon < TOL$ **then**
      return x_{k+1}
    end if
    $\beta_{k+1} = \frac{r_{k+1}r_{k+1}}{r_k r_k}$
    $p_{k+1} = r_{k+1} + \beta_k p_k$
  end while

M, que utilizamos como pre condicionador para acelerar la convergencia del método. La matriz M debe de ser una matriz simétrica, positiva definida y no debe de cambiar a lo largo de las iteraciones. En este trabajo utilizamos el pre condicionador de Jacobi M cuyas entradas de la matriz son $a_{i,i}$ si $i = j$ y 0 en otro caso. El algoritmo de gradiente conjugado pre condicionado es el siguiente.

Algorithm 8 Pre-Conditioned-Conjugate-Gradient

Require: A, **b**, $\mathbf{x}_0$,TOL
Ensure: **x** tal que $A\mathbf{x} = \mathbf{b}$
  $\mathbf{r}_0 = b - Ax_0$
  $z_0 = M^{-1}r_0$
  $\mathbf{p}_0 = \mathbf{z}_0$
  $\epsilon = \|r_0\|$
  while $\epsilon > TOL$ **do**
    $\alpha_k = \frac{r_k z_k}{p_k^T A p_k}$
    $x_{k+1} = \alpha_k p_k$
    $r_k = r_k - \alpha_k A p_k$
    $\epsilon = \|r_{k+1}\|$
    if $\epsilon < TOL$ **then**
      return x_{k+1}
    end if
    $z_{k+1} = M^{-1}r_{k+1}$
    $\beta_{k+1} = \frac{r_{k+1}z_{k+1}}{r_k z_k}$
    $p_{k+1} = z_{k+1} + \beta_k p_k$
  end while

III. Resultados

En la sección de resultados reportaremos solamente los valores propios obtenidos por los métodos tanto de iteración en subespacio como iteración en subespacio inverso, al ser los vectores propios de alta dimensionalidad solo nos limitaremos a mostrar los valores propios

obtenidos por los métodos, la prueba de que las implementaciones realizadas en este trabajo funcionan se encuentran en el Apéndice A que contiene las capturas de pantalla para cada uno de los ejercicios.

III-A. Cálculo de Valores y vectores propios

Los $n = 10$ valores propios más grandes y más pequeños respectivamente se muestran en la siguiente tabla

Indice	Valor λ_i
1	9.99805
2	19.9988
3	29.9989
4	39.9998
5	49.9984
6	59.9998
7	69.9991
8	79.9997
9	89.9999
10	99.9994
41	410
42	420
43	430
44	440
45	450
46	460.001
47	470.001
48	480.001
49	490.001
50	500.002

Cuadro I
VALORES PROPIOS PARA UNA MATRIZ DE 50×50 OBTENIDOS CON LAS VARIACIONES DE ITERACIÓN EN SUBESPACIO

III-B. Solución de sistemas de ecuaciones

Al igual que en la sección anterior, en el apéndice A se muestra la funcionalidad de los métodos. Para los métodos de gradiente conjugado y gradiente conjugado pre condicionado utilizamos un vector inicial de unos.

Mostramos a continuación la solución al sistema de ecuaciones de 3×3. Las soluciones a los sistemas de 3×3 y 125×125 se adjuntan en el archivo comprimido de esta tarea.

Variable	Valor
x_1	3
x_2	-2.5
x_3	7

Cuadro II
SOLUCIONES AL SISTEMA DE 3×3

IV. CONCLUSIONES

Debido a la diversidad temática de la presente tarea dividiremos las conclusiones en dos partes. Respecto a los métodos de obtención de valores y vectores propios vistos en el presente trabajo, podemos ver que son más eficientes que los métodos previamente implementados. Sin embargo, en el caso de la iteración en subespacio inversa, al hacer una multiplicación encadenada de matrices se acarrea error de forma que hay un error mayor al momento de calcular el error $\|Ax - \lambda x\|$. No obstante, ambos métodos cumplen su funcionalidad para encontrar de forma precisa los correspondientes m valores y vectores propios.

Comparativo de tiempos de ejecución	
Método	Tiempo
Iteración en Subespacio	**0.53**
Iteración en Subespacio Inversa	4.74
Raleigh	$5{,}14e^{-5}$

Cuadro III
TIEMPOS DE EJECUCIÓN

Notemos que de los métodos de iteración en subespacio el más eficiente es el usado para calcular los m valores propios más grandes, tiene sentido,ya que dicho método no involucra factorización de matrices ni producto encadenado de matrices

En cuanto a la solución de sistemas de ecuaciones con los métodos de la descomposición QR y los métodos de gradiente conjugado, tuvieron un desempeño similar en cuanto a error y en cuanto a tiempo de ejecución. El tiempo de ejecución de los métodos se muestra a continuación.

Comparativo de tiempos de ejecución	
Método	Tiempo
QR	0.0671
Gradiente Conjugado	0.04
Gradiente conjugado prec ondicionado	**0.024**

Cuadro IV
TIEMPOS DE EJECUCIÓN

En la Tabla IV podemos ver los tiempos de ejecución de los diferentes métodos, vemos que el método de gradiente conjugado pre condicionado es el más eficiente en tiempo de los métodos implementados. Esto demuestra que la matriz pre condicionante si aumenta la velocidad de convergencia tal y como fue enunciado en la sección de metodología.

REFERENCIAS

[1] Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis*. Cengage learning, 2015.

V. Apéndice A

V-A. Problema 1-Iteración en subespacio

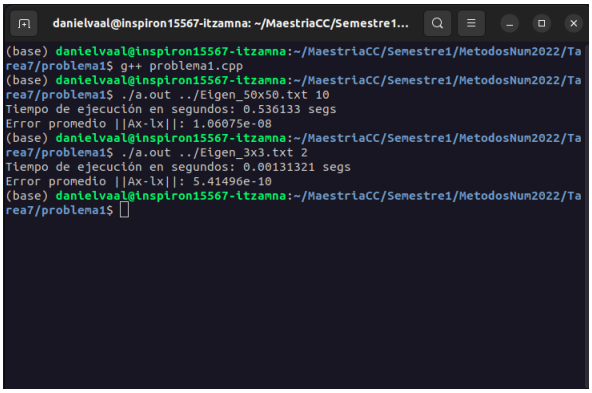

Figura 1. Salida del problema 1 en consola

V-B. Problema 2-Iteración en subespacio inversa

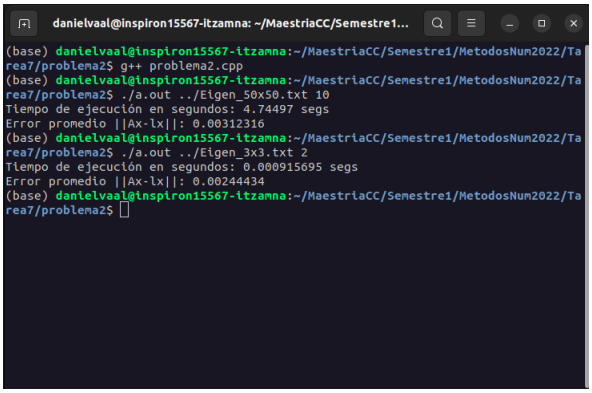

Figura 2. Salida del problema 2 en consola

V-C. Problema 3-Cociente de Raleigh

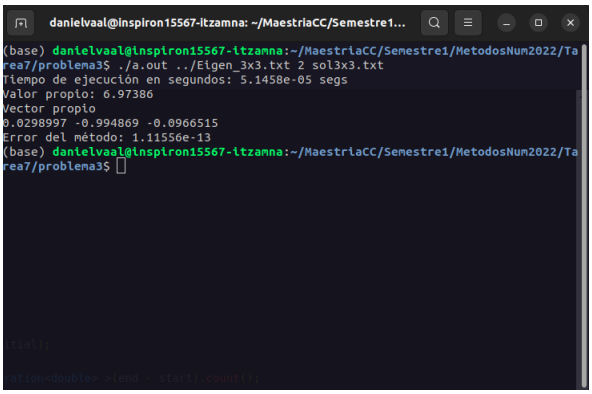

Figura 3. Salida del problema 3 en consola

V-D. Problema 4-QR

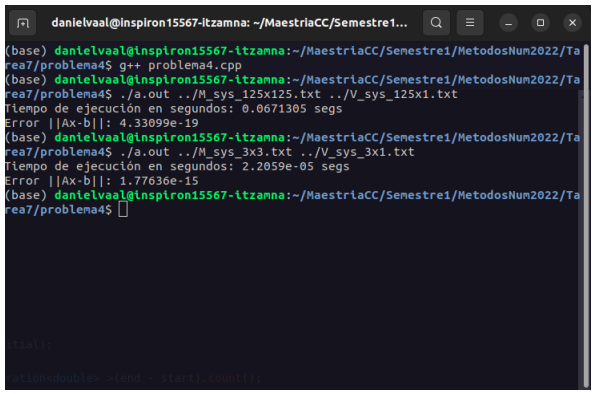

Figura 4. Salida del problema 4 en consola

V-E. Problema 5-Gradiente Conjugado

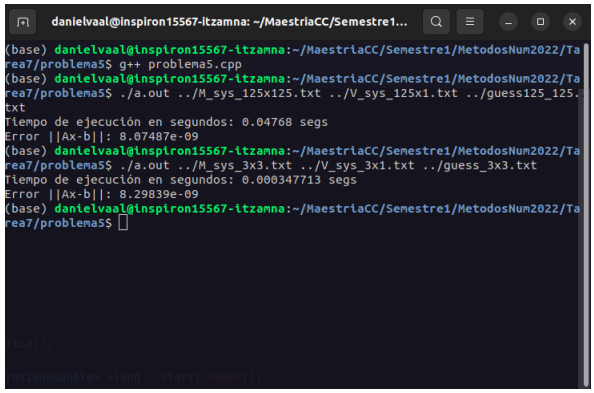

Figura 5. Salida del problema 5 en consola

V-F. Problema 6-Gradiente conjugado pre condicionado

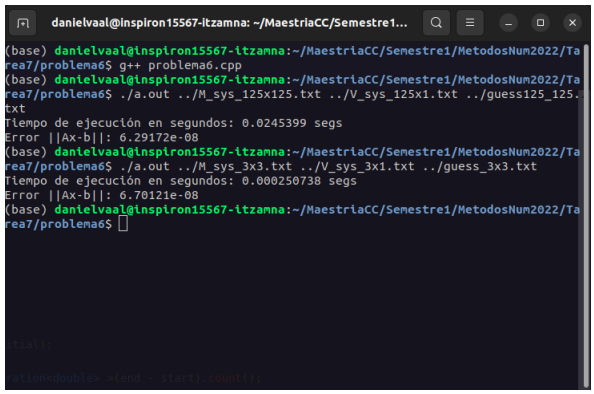

Figura 6. Salida del problema 6 en consola