Tarea 9 Métodos Numéricos

Carlos Giovanny Encinia González

3 de octubre 2021

Resumen

En esta tarea se muestran los métodos iterativos para encontrar los eigenvalores y eigenvectores dominante y menores de una matriz. También se muestra la manera de encontrar un numero de eigenvalores sabiendo uno de los eigenvalores dominantes de la matriz.

Desarrollo

Método de la potencia

Primero definiremos el problema de valores y vectores propios.

$$Ax = \lambda x \tag{1}$$

Donde x es llamado vector propio y λ es el valor propio.

Para calcular los valores propios, podemos utilizar el determinante de la siguiente manera.

Primero de la ecuación (1) pasamos el termino con el valor propio hacia la izquierda.

$$Ax - \lambda x = 0 \tag{2}$$

Factorizando nos queda

$$x(A - \lambda I) = 0 \tag{3}$$

Si queremos que la solución no sea trivial debemos hacer que

$$|A - \lambda I| = 0 \tag{4}$$

Ahora si quisiéramos encontrar e valor propio dominante o valor propio con mayor valor absoluto y su correspondiente vector propio podemos utilizar el método de la potencia que es como se describe a continuación.

Primero como hemos dicho antes se tiene que:

$$|\lambda_n| > |\lambda_{n-1}| > \cdots > |\lambda_1| \tag{5}$$

Ahora Denotaremos como x_i al vector propio asociado a λ_i

Proponemos que

$$v_1 = Av_0 \tag{6}$$

Pero v_0 puede ser escrito como una combinacion lineal de los vectores propios de la matriz, entonces tenemos que:

$$v_1 = A\sum_{i=1}^{n} a_i x_i \tag{7}$$

Ingresando A a la sumatoria y aplicando la definición de valor propio nos queda que:

$$v_1 = \sum_{i=1}^{n} a_i \lambda_i x_i \tag{8}$$

Ahora, si multiplicamos y dividimos por el vector dominante λ_n, tenemos que

$$v_1 = \lambda_n \sum_{i=1}^{n} a_i \frac{\lambda_i}{\lambda_n} x_i \tag{9}$$

Ahora si quitamos de la sumatoria el elemento n y reescribimos, tenemos que

$$v_1 = \lambda_n \sum_{i=1}^{n-1} a_i \frac{\lambda_i}{\lambda_n} x_i + a_n x_n \tag{10}$$

Para un vector v_2 tenemos que:

$$v_2 = Av_1 \tag{11}$$

Sustituyendo lo obtenido para v_1 tenemos que:

$$v_2 = A(\lambda_n \sum_{i=1}^{n-1} a_i \frac{\lambda_i}{\lambda_n} x_i + a_n x_n) \tag{12}$$

multiplicando A en cada termino y aplicando la definición de vector propio nos queda que:

$$v_2 = \lambda_n \sum_{i=1}^{n-1} a_i \frac{\lambda_i}{\lambda_n} \lambda_i x_i + a_n \lambda_n x_n \tag{13}$$

Multiplicando por λ_n/λ_n el primer termino y sacando como factor común λ_n nos queda:

$$v_2 = \lambda_n^2 (\sum_{i=1}^{n-1} a_i \left(\frac{\lambda_i}{\lambda_n}\right)^2 x_i + a_n x_n) \tag{14}$$

De aquí podemos generalizar y decir que:

$$v_r = \lambda_n^r (\sum_{i=1}^{n-1} a_i \left(\frac{\lambda_i}{\lambda_n}\right)^r x_i + a_n x_n) \tag{15}$$

Si tenemos valores grandes de r y sabiendo que λ_n es mayor a los demás valores propios podemos hacer:

$$\boxed{v_r \approx \lambda_n^r a_n x_n} \tag{16}$$

y

$$\boxed{v_{r+1} \approx \lambda_n^{r+1} a_n x_n} \tag{17}$$

Para calcular λ_n podemos realizar la siguiente operación:

$$\frac{v_{r+1}^T \cdot v_r}{v_r^T \cdot v_r} \tag{18}$$

Sustituyendo lo obtenido tenemos que:

$$\frac{\lambda_n^{r+1} a_n x_n \lambda_n^r a_n x_n}{\lambda_n^r a_n x_n \lambda_n^r a_n x_n} \tag{19}$$

Eliminando factores comunes nos queda que:

$$\boxed{\lambda_n = \frac{v_{r+1}^T \cdot v_r}{v_r^T \cdot v_r}} \tag{20}$$

Además para evitar que con cada iteración el nuevo vector tenga valores demasiado grandes tendremos que normalizar cada nuevo vector que encontremos.

Entonces con los resultados obtenidos podemos escribir el siguiente pseudocódigo.

Algorithm 1: Método de Potencia

eigen_valor_vector_dominante $(matrix, rows, x0)$;

Input : matrix: matriz que contiene coeficientes ecuación, x0: es el vector que se propone para solucionar, rows: numero de filas

Output: $x1, \lambda$

$ERROR = 1E^{-9}$;

$matrix \leftarrow readMatrix()$;

$x0 \leftarrow creaVectorZeros(rows)$;

$m \leftarrow rows$;

$x0_0 \leftarrow 1/sqrt(n)$;

$iteration \leftarrow 0$;

$condition \leftarrow True$ $\lambda_{old} \leftarrow 0$;

while *condition and iteration* $<$ *LIMIT* **do**

  x1 = productoPunto(matrix, x0);

  $\lambda = productoPunto(x1^T, x0)/productoPunto(x0^T, x0)$;

  if $abs(\lambda_{old} - \lambda) < ERROR$ **then**

    return $x1, \lambda$;

  end

  $\lambda_{old} \leftarrow \lambda$;

  $x0 \leftarrow x1$;

  $x0 \leftarrow normalizar\_vector(x0)$;

  $iteration \leftarrow iteration + 1$;

end

Método de Potencia Inversa

Primero proponemos la siguiente ecuación matricial

$$Av_1 = v_0 \tag{21}$$

Aplicando la inversa de la matriz a cada lado de la ecuación nos queda que:

$$v_1 = A^{-1}v_0 \tag{22}$$

Ahora si sabemos que

$$Ax_i = \lambda_i x_i \tag{23}$$

podemos aplicar de nuevo la inversa y obtener que

$$x_i = \lambda_i A^{-1} x_i \tag{24}$$

pasando x_i a la izquierda y cambiando de sentido la ecuación, tenemos que:

$$A^{-1}x_i = \frac{x_i}{\lambda_i} \tag{25}$$

Volviendo de nuevo a la ecuación propuesta, sabemos que v_0 se puede escribir como una combinación lineal de los eigenvectores de la matriz y entonces tenemos que:

$$v_1 = A^{-1}\sum_{i=1}^{n} a_i x_i \tag{26}$$

Observamos que podemos utilizar la expresión () para simplificar y obtenemos que

$$v_1 = \sum_{i}^{n} \frac{a_i x_i}{\lambda_i} \tag{27}$$

Si multiplicamos y dividimos por λ_1, tenemos que

$$v_1 = \frac{1}{\lambda_1}\sum_{i}^{n} a_i x_i \frac{\lambda_1}{\lambda_i} \tag{28}$$

Ahora si queremos un vector v_2, entonces tendremos que:

$$v_2 = A^{-1}v_1 \tag{29}$$

Sustituyendo los resultados obtenidos anteriormente tenemos que:

$$v_2 = A^{-1}\left(\frac{1}{\lambda_1}\sum_{i=1}^{n} a_i \frac{\lambda_1}{\lambda_i} x_i\right) \tag{30}$$

Multiplicando por A^{-1} y usando la ecuación (), tenemos que:

$$v_2 = \frac{1}{\lambda_1}\sum_{i=1}^{n} a_i \frac{\lambda_1}{\lambda_i}\frac{x_i}{\lambda_i} \tag{31}$$

De nuevo multiplicando denominador y numerados por λ_1 y agrupando factores comunes tenemos que:

$$v_2 = \frac{1}{\lambda_1^2}\sum_{i=1}^{n} a_i \left(\frac{\lambda_1}{\lambda_i}\right)^2 x_i \tag{32}$$

si ahora generalizamos para v_r, tenemos que

$$v_r = \frac{1}{\lambda_1^r}\sum_{i=1}^{n} a_i \left(\frac{\lambda_1}{\lambda_i}\right)^r x_i \tag{33}$$

si expandimos la sumatoria obtenemos

$$v_r = \frac{1}{\lambda_1^r}(a_1 x_1 + a_2\left(\frac{\lambda_1}{\lambda_2}\right)^r x_2 + \cdots + a_n\left(\frac{\lambda_1}{\lambda_n}\right)^r x_n) \tag{34}$$

Los términos mayores a i=1, tienden a cero cuando r es muy grande, ya que el numerador es el lambda mínimo, entonces podemos escribir que:

$$\boxed{v_r = \frac{1}{\lambda_1^r}a_1 x_1} \tag{35}$$

y para un vector siguiente a r tenemos que:

$$\boxed{v_{r+1} = \frac{1}{\lambda_1^{r+1}}a_1 x_1} \tag{36}$$

Para encontrar el eigenvalor mínimo podemos hacer:

$$\frac{v_r^T \cdot v_r}{v_{r+1}^T \cdot v_r} \tag{37}$$

sustituyendo lo obtenido tenemos que:

$$\frac{\frac{1}{\lambda_1^r}a_1 x_1 \frac{1}{\lambda_1^r}a_1 x_1}{\frac{1}{\lambda_1^{r+1}}a_1 x_1 \frac{1}{\lambda_1^r}a_1 x_1} \tag{38}$$

Eliminando los factores iguales tenemos que:

$$\boxed{\lambda_1 = \frac{v_r^T \cdot v_r}{v_{r+1}^T \cdot v_r}} \tag{39}$$

Con todo lo anterior podemos escribir el siguiente pseudocódigo:

Algorithm 2: Método de Potencia Inversa

eigen_valor_vector_dominante $(matrix, rows, x0)$;
Input : matrix: matriz que contiene coeficientes ecuación, x0: es el vector que se propone para solucionar, rows: numero de filas
Output: $x1, \lambda$
$ERROR = 1E^{-9}$;
$matrix \leftarrow readMatrix()$;
$x0 \leftarrow creaVectorZeros(rows)$;
$m \leftarrow rows$;
$x0_0 \leftarrow 1/sqrt(n)$;
$iteration \leftarrow 0$;
$condition \leftarrow True$;
$\lambda_{old} \leftarrow 1E40$;
while *condition and iteration* $<$ *LIMIT* **do**
    x1 = SolveLU(matrix, x0);
    $\lambda = productoPunto(x1^T, x0)/productoPunto(x0^T, x0)$;
    if *abs(*$\lambda_{old} - \lambda$*)* $<$ *ERROR* **then**
        return $x1, \lambda$;
    end
    $\lambda_{old} \leftarrow \lambda$;
    $x0 \leftarrow x1$;
    $x0 \leftarrow normalizar\_vector(x0)$;
    $iteration \leftarrow iteration + 1$;
end

Método potencia para m λ mayores

Para poder aplicar este método primero debemos asegurarnos que la matriz con la que vamos a trabajar debe ser simétrica.

Ahora partimos del supuesto de que conocemos el eigenvalor dominante λ_n y entonces podemos escribir que un vector cualquiera puede ser escrito como una combinación de los eigenvectores:

$$v = \sum_{i=1}^{n-1} a_i x_i + a_n x_n \tag{40}$$

Multiplicando por l traspuesta del eigenvector dominante tenemos que:

$$x_n^T v = x_n^T \sum_{i=1}^{n-1} a_i x_i + a_n x_n^T x_n \tag{41}$$

Como la matriz es simétrica los eigenvectores son ortogonales y la sumatoria al tener elementos distintos a x_n estos se vuelven 0 y nos queda que:

$$x_n^T v = a_n \tag{42}$$

Para calcular el eigenvalor n-1, simplemente debemos de quitar el componente asociado al eigenvalor dominante y tenemos que

$$\hat{v} = v - a_n x_n \tag{43}$$

Y sustituyendo el valor encontrado de a_n tenemos que:

$$\hat{v_0} = v - x_n^T v x_n \tag{44}$$

Donde $\hat{v}$ es el vector el cual utilizaremos en el algoritmo de la potencia para calcular el n-1 eigenvector dominante.

Siguiendo esta lógica podemos encontrar los m valores dominantes, siempre restando el eigenvalor dominante anterior.

en general tendríamos que:

$$\hat{v_0} = v_0 - \sum_{i=n}^{m} x_i^T v_o x_i \tag{45}$$

Con i=n disminuyendo hasta m

En seguida se muestra el pseudocódigo.

Algorithm 3: Método de Potencia deflación

```
eigen_valor_vector_dominante (matrix, rows, x0);
Input  : matrix: matriz que contiene coeficientes ecuación, x0: es el
         vector que se propone para solucionar, rows: numero de filas
Output: x1, λ
i ← 1 for i:m do
    ERROR = 1E^-9;
    matrix ← readMatrix();
    x0 ← creaVectorZeros(rows);
    m ← rows;
    x0_0 ← 1/sqrt(n);
    iteration ← 0;
    condition ← True λ_old ← 0;
    while condition and iteration < LIMIT do
        x1 = productoPunto(matrix, x0);
        λ = productoPunto(x1^T, x0)/productoPunto(x0^T, x0);
        if abs(λ_old − λ) < ERROR then
            xn_i, vector_lambda_i ← x1, λ;
            continue;
        end
        λ_old ← λ;
        x0 ← x1;
        x0 ← normalizar_vector(x0);
        if i ≠ 0 then
            for k ← 0, k < i, k++ do
                an = dot_vector(x0, xn_k, m);
                for j ← 0, j < m, j++ do
                    x0_j ← x0_j - an * xn_{k,j};
                end
            end
        end
        iteration ← iteration + 1;
    end
end
```

M eigenvalores menores con el método de la inversa

Para poder aplicar este método primero debemos asegurarnos que la matriz con la que vamos a trabajar debe ser simétrica.

Ahora partimos del supuesto de que conocemos el eigenvalor menor λ_1 y entonces podemos escribir que un vector cualquiera puede ser escrito como una combinación de los eigenvectores:

$$v = \sum_{i=2}^{n} a_i x_i + a_1 x_1 \tag{46}$$

Multiplicando por l traspuesta del eigenvector dominante tenemos que:

$$x_1^T v = x_1^T \sum_{i=2}^{n} a_i x_i + a_1 x_1^T x_1 \tag{47}$$

Como la matriz es simétrica los eigenvectores son ortogonales y la sumatoria al tener elementos distintos a x_1 estos se vuelven 0 y nos queda que:

$$x_1^T v = a_1 \tag{48}$$

Para calcular el eigenvalor 2, simplemente debemos de quitar el componente asociado al eigenvalor dominante y tenemos que

$$\hat{v} = v - a_1 x_1 \tag{49}$$

Y sustituyendo el valor encontrado de a_1 tenemos que:

$$\hat{v_0} = v - x_1^T v x_1 \tag{50}$$

Donde $\hat{v}$ es el vector el cual utilizaremos en el algoritmo de la potencia para calcular el i eigenvector dominante.

Siguiendo esta lógica podemos encontrar los n valores dominantes, siempre restando el eigenvalor dominante anterior.

en general tendríamos que:

$$\hat{v_0} = v_0 - \sum_{i=1}^{n} x_i^T v_0 x_i \tag{51}$$

Con i=1 aumentando hasta hasta n

En seguida se muestra el pseudocódigo.

Algorithm 4: M eigenvalores menores

eigen_valor_vector_dominante $(matrix, rows, x0)$;
Input : matrix: matriz que contiene coeficientes ecuación, x0: es el vector que se propone para solucionar, rows: numero de filas
Output: $x1, \lambda$

$i \leftarrow 1$ **for** *i:m* **do**
    $ERROR = 1E^{-9}$;
    $matrix \leftarrow readMatrix()$;
    $x0 \leftarrow creaVectorZeros(rows)$;
    $m \leftarrow rows$;
    $x0_0 \leftarrow 1/sqrt(n)$;
    $iteration \leftarrow 0$;
    $condition \leftarrow True$ $\lambda_{old} \leftarrow 0$;
    while *condition and iteration* $<$ *LIMIT* **do**
        x1 = SolveLU(matrix, x0);
        $\lambda = productoPunto(x1^T, x0)/productoPunto(x0^T, x0)$;
        if $abs(\lambda_{old} - \lambda) < ERROR$ **then**
            xn_i, $vector\_lambda_i \leftarrow x1, \lambda$;
            continue;
        end
        $\lambda_{old} \leftarrow \lambda$;
        $x0 \leftarrow x1$;
        $x0 \leftarrow normalizar\_vector(x0)$;
        if $i \neq 0$ **then**
            for $k \leftarrow 0, k < i, k++$ **do**
                an = dot_vector(x0, xn_k, m);
                for $j \leftarrow 0, j < m, j++$ **do**
                    $x0_j \leftarrow x0_j$ - an * $xn_{k,j}$;
                end
            end
        end
        $iteration \leftarrow iteration + 1$;
    end
end

Resultados

Metodo de la potencia e Inversa

Figure 1: Matriz de 3x3

```
==============================================================================
|                                Matriz  3x  3                               |
==============================================================================
|Eigen Dominante                                                             |
------------------------------------------------------------------------------
Iteraciones: 37
Eigenvalor: 10.034796
Eigenvector:
x0: -0.269927 x1: -0.977985 x2: 9.983377
------------------------------------------------------------------------------
|Eigen Menor                                                                 |
------------------------------------------------------------------------------
Iteraciones: 15
Eigenvalor: 2.991343
Eigenvalor
x0: 0.334027 x1: 0.009075 x2: 0.009920
==============================================================================
```

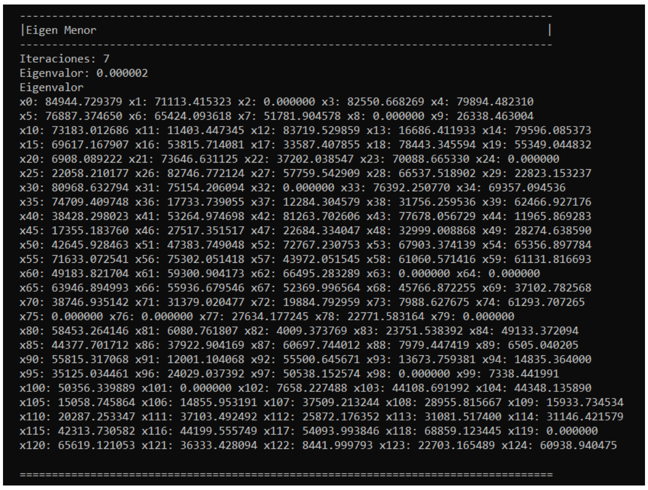
Figure 2: Matriz de 50x50

```
==============================================================================
|                                Matriz 50x 50                               |
==============================================================================
|Eigen Dominante                                                             |
------------------------------------------------------------------------------
Iteraciones: 605
Eigenvalor: 500.001543
Eigenvector:
x0: 0.008834 x1: 0.043285 x2: 0.056450 x3: 0.109045 x4: 0.050010
x5: 0.038664 x6: 0.086008 x7: 0.048686 x8: 0.074086 x9: 0.040892
x10: 0.074451 x11: 0.133104 x12: 0.021049 x13: 0.028015 x14: 0.132889
x15: 0.004210 x16: 0.117507 x17: 0.116089 x18: 0.043179 x19: 0.161682
x20: 0.081283 x21: 0.002856 x22: 0.072070 x23: 0.130338 x24: 0.157877
x25: 0.084478 x26: 0.042222 x27: 0.047144 x28: 0.124842 x29: 0.152326
x30: 0.112853 x31: 0.110345 x32: 0.187246 x33: 0.118602 x34: 0.216441
x35: 0.169339 x36: 0.204037 x37: 0.307582 x38: 0.193361 x39: 0.129275
x40: 0.058324 x41: 0.286511 x42: 0.478469 x43: 0.280062 x44: 0.597825
x45: 0.734740 x46: 0.884681 x47: 1.105093 x48: 3.969892 x49: 499.981909

------------------------------------------------------------------------------
|Eigen Menor                                                                 |
------------------------------------------------------------------------------
Iteraciones: 17
Eigenvalor: 9.998050
Eigenvalor
x0: 0.100015 x1: -0.000831 x2: -0.000188 x3: -0.000257 x4: -0.000197
x5: -0.000180 x6: -0.000030 x7: -0.000046 x8: -0.000094 x9: -0.000030
x10: -0.000055 x11: -0.000042 x12: -0.000052 x13: -0.000027 x14: -0.000035
x15: -0.000062 x16: -0.000056 x17: -0.000037 x18: -0.000039 x19: -0.000007
x20: -0.000030 x21: -0.000000 x22: -0.000010 x23: -0.000005 x24: -0.000033
x25: -0.000006 x26: -0.000015 x27: -0.000004 x28: -0.000003 x29: -0.000034
x30: -0.000007 x31: -0.000016 x32: -0.000026 x33: -0.000018 x34: -0.000008
x35: -0.000018 x36: -0.000014 x37: -0.000013 x38: -0.000025 x39: -0.000007
x40: -0.000019 x41: -0.000013 x42: -0.000018 x43: -0.000009 x44: -0.000020
x45: -0.000006 x46: -0.000007 x47: -0.000017 x48: -0.000019 x49: -0.000001

==============================================================================
```

Figure 3: Matriz de 125x125

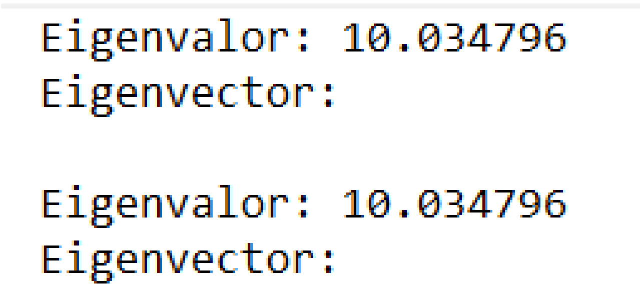


```
--------------------------------------------------------------------------------
|Eigen Menor                                                                   |
--------------------------------------------------------------------------------
Iteraciones: 7
Eigenvalor: 0.000002
Eigenvalor
x0: 84944.729379 x1: 71113.415323 x2: 0.000000 x3: 82550.668269 x4: 79894.482310
x5: 76887.374650 x6: 65424.093618 x7: 51781.904578 x8: 0.000000 x9: 26338.463004
x10: 73183.012686 x11: 11403.447345 x12: 83719.529859 x13: 16686.411933 x14: 79596.085373
x15: 69617.167907 x16: 53815.714081 x17: 33587.407855 x18: 78443.345594 x19: 55349.044832
x20: 6908.089222 x21: 73646.631125 x22: 37202.038547 x23: 70088.665330 x24: 0.000000
x25: 22058.210177 x26: 82746.772124 x27: 57759.542909 x28: 66537.518902 x29: 22823.153237
x30: 80968.632794 x31: 75154.206094 x32: 0.000000 x33: 76392.250770 x34: 69357.094536
x35: 74709.409748 x36: 17733.739055 x37: 12284.304579 x38: 31756.259536 x39: 62466.927176
x40: 38428.298023 x41: 53264.974698 x42: 81263.702606 x43: 77678.056729 x44: 11965.869283
x45: 17355.183760 x46: 27517.351517 x47: 22684.334047 x48: 32999.008868 x49: 28274.638590
x50: 42645.928463 x51: 47383.749048 x52: 72767.230753 x53: 67903.374139 x54: 65356.897784
x55: 71633.072541 x56: 75302.051418 x57: 43972.051545 x58: 61060.571416 x59: 61131.816693
x60: 49183.821704 x61: 59300.904173 x62: 66495.283289 x63: 0.000000 x64: 0.000000
x65: 63946.894993 x66: 55936.679546 x67: 52369.996564 x68: 45766.872255 x69: 37102.782568
x70: 38746.935142 x71: 31379.020477 x72: 19884.792959 x73: 7988.627675 x74: 61293.707265
x75: 0.000000 x76: 0.000000 x77: 27634.177245 x78: 22771.583164 x79: 0.000000
x80: 58453.264146 x81: 6080.761807 x82: 4009.373769 x83: 23751.538392 x84: 49133.372094
x85: 44377.701712 x86: 37922.904169 x87: 60697.744012 x88: 7979.447419 x89: 6505.040205
x90: 55815.317068 x91: 12001.104068 x92: 55500.645671 x93: 13673.759381 x94: 14835.364000
x95: 35125.034461 x96: 24029.037392 x97: 50538.152574 x98: 0.000000 x99: 7338.441991
x100: 50356.339889 x101: 0.000000 x102: 7658.227488 x103: 44108.691992 x104: 44348.135890
x105: 15058.745864 x106: 14855.953191 x107: 37509.213244 x108: 28955.815667 x109: 15933.734534
x110: 20287.253347 x111: 37103.492492 x112: 25872.176352 x113: 31081.517400 x114: 31146.421579
x115: 42313.730582 x116: 44199.555749 x117: 54093.993846 x118: 68859.123445 x119: 0.000000
x120: 65619.121053 x121: 36333.428094 x122: 8441.999793 x123: 22703.165489 x124: 60938.940475

================================================================================
```

M mayores

Figure 4: Matriz de 125x125



```
Eigenvalor: 10.034796
Eigenvector:

Eigenvalor: 10.034796
Eigenvector:
```

Conclusión

En general aunque el método de la potencia es mas rápido, pude observar que necesita de mas iteraciones para llegar a una solución, en el programa pido que se tenga un error de 1E-12, el método de la inversa se realizan menos iteraciones. Puedo decir que ambos métodos en general son rápidos.

Para el algoritmo de los m mayores no logre que funcionara mi algoritmo, lo realicé de diversas maneras, pero al parecer el error que seguía teniendo era con la manera en que llamaba a los apuntadores en C, al final lo dejé al tercer intento, el cuál muestra los valores repetidos, es decir pareciera que no quita las contribuciones de los eigenvalores anteriormente encontrados. La primera forma en que escribí el código mostraba, los valores correctos pero estaban desfasados. Por ejemplo, si quería encontrar el penúltimo eigenvalor dominante debía correr el programa unas 3 o más veces según el tamaño de la matriz y de nuevo, pienso que el problema se relacionaba con cosas técnicas del lenguaje de programación usado.