Práctica 1. Cálculo Numérico

Leslie B., César B., Gerardo P. y Maximiliano M.

12 de marzo de 2018

1. Implementación

El objetivo de la primera parte del curso es encontrar valores propios y sus vectores propios asociados. Dependiendo el tipo de matriz que tengamos, y de la información que necesitemos, utilizaremos un método diferente o alguna de sus variaciones.

Ejercicio 3.1

Inciso 1

Después de 10 iteraciones del Método de potencia se obtienen el eigenvector y el eigenvalor aproximados:

$$q_{10} = \begin{pmatrix} 0.083519308490898 \\ 0.979587754935313 \\ -0.182845168079429 \end{pmatrix} \text{ y } \lambda_{10} = 8.355251067024421$$

Inciso 2

Según el método *eig* de Matlab, estos deberían ser:

$$q = \begin{pmatrix} 0.083443662459397 \\ 0.979575954647159 \\ -0.182942898939720 \end{pmatrix} \text{ y } \lambda = 8.354544835161549$$

Las diferencias entre el eigenpar aproximado y real son:

$$\Delta\lambda = 0.000706231862862339 \text{ y } \Delta q = \begin{pmatrix} 0.00007564603150084226 \\ 0.00001180028815395140 \\ 0.00009773086029032929 \end{pmatrix}$$

Inciso 3

Las primeras aproximaciones al radio de convergencia son:
0.3823534, 0.3909982, 0.4008846, 0.40579235, 0.40792949, 0.40882466, 0.40919442, 0.40934633, 0.40940861

Inciso 4

Vemos que los radios de convergencia estimados convergen a $\left|\frac{\lambda_2}{\lambda_1}\right| = 0.40945181$.

Explicación

El eigenvector y eigenvalor resultante se asemejan mucho a los dados por Matlab debido a que la matriz tiene un eigenvalor dominante [1] y que el vector inicial q_0 se escribe como combinación lineal de los eigenvectores (la matriz es semisimple), de manera que el coeficiente que multiplica al eigenvector dominante no es cero. Ésto se refleja en los inicisos 3 y 4 porque la razón práctica converge rápidamente a la razón teórica.

Ejercicio 3.2

Inciso A.1

Después de 10 iteraciones del Método de potencia inversa con shift $\rho = 0$ se obtienen el eigenvector y el eigenvalor aproximados:

$$q_{10} = \begin{pmatrix} 0.992719306520162 \\ 0.104174016598757 \\ 0.060466128764870 \end{pmatrix} \text{ y } \lambda_{10} = 1.226789426141183$$

[1] Los eigenvalores de esta matriz son: $\lambda_1 = 8.35454 > \lambda_2 = 3.42078 > \lambda_3 = 1.22467$

Inciso A.2

Según el método *eig* de Matlab, estos deberían ser:

$$q = \begin{pmatrix} -0.992728312692462 \\ -0.104882397034991 \\ -0.059077745141235 \end{pmatrix} \text{ y } \lambda = 1.224671629151526$$

Las diferencias entre el eigenpar aproximado y real son:

$$\Delta\lambda = 0.002117796989657 \text{ y } \Delta q = \begin{pmatrix} 0.000009006172299 \\ 0.000708380436234 \\ -0.001388383623635 \end{pmatrix}$$

Inciso A.3

Las primeras aproximaciones al radio de convergencia son:
0.96608, 0.853758, 0.678342, 0.495284, 0.4043724, 0.373392, 0.363308, 0.359879, 0.3586767

Inciso A.4

Vemos que los radios de convergencia estimados convergen a $\left|\frac{\lambda_2}{\lambda_1}\right| = 0.358009$.

Explicación

El método, puesto que el shift es $\rho = 0$, aproxima los eigenpares de la inversa. El eigenvector y eigenvalor que aproximamos se asemeja mucho a los dados por *matlab* debido a que la matriz tiene eigenvalores ordenados y entonces la inversa también los tiene. También se debe a que el vector inicial q_0 se escribe como combinación lineal de los eigenvectores (que son los mismos que en la matriz original), es decir, el coeficiente que multiplica al eigenvector asociado al eigenvalor dominante -de la inversa- no es cero.

Inciso B.1

Después de 10 iteraciones del Método de potencia inversa con shift $\rho = 3.3$ se obtienen el eigenvector y el eigenvalor aproximados:

$$q_{10} = \begin{pmatrix} 0.515310726734264 \\ -0.332385611227379 \\ 0.789920667131583 \end{pmatrix} \text{ y } \lambda_{10} = 3.420783535683841$$

Inciso B.2

Según el método *eig* de Matlab, estos deberían ser:

$$q = \begin{pmatrix} 0.515310726732435 \\ -0.332385611228160 \\ 0.789920667132448 \end{pmatrix} \text{ y } \lambda = 3.420783535686916$$

Las diferencias entre el eigenpar aproximado y real son:

$$\Delta\lambda = 0.000000000003074 \text{ y } \Delta q = \begin{pmatrix} -0.00000000000182920 \\ -0.00000000000078048 \\ 0.00000000000086497 \end{pmatrix}$$

Inciso B.3

Las primeras aproximaciones del radio de convergencia son:
0.02487, 0.02083, 0.03243, 0.02111, 0.08558, 0.05055, 0.061824, 1, 1

Inciso B.4

Vemos que los radios de convergencia estimados no tienden a $\left|\frac{\lambda_2}{\lambda_1}\right| = 0.058199$.

Explicación

El eigenvector y eigenvalor resultante se aproximan mucho a los dados por *matlab* debido a que los eigenvalores con shift $\rho = 3.3$ siguen estando ordenados. Más aún, el shift ya es cercano al eigenvalor real, por lo que es eficiente. Lo anterior se observa en el radio de convergencia real, cuyo valor es pequeño -entonces el eigenvalor con shift dominante es mucho

mayor que el siguiente en magnitud-. Por lo mismo, como el eigenpar tiende rápidamente al real, la precisión limitada de la computadora no nos permite calcular correctamente el radio de convergencia en las últimas dos iteraciones, puesto que el cociente $||\frac{q_{i+1}-q}{q_i-q}||$ es prácticamente 1. Igualmente se debe a que el vector inicial de nuevo se escribe como combinación lineal de los eigenvectores, con coeficiente del eigenvector dominante distinto de cero.

Ejercicio 3.3

El método de la potencia inversa requiere 13 iteraciones cuando el shift es $p = 3.6$ y la toleracia relativa es 10^{-15}. Nótese que los eigenvalores con shift se vuelven: $\lambda_1 = 4.7545$, λ_2 = -2.3753, y λ_3 = -0.17920. De este modo, el radio real de convergencia sería: $r = 0.075443$. En teoría, converge muy rápido.

Ejercicio 3.4

Usamos los vectores iniciales:

$$x_0 = \begin{pmatrix} 11 \\ 92 \\ 79 \end{pmatrix}, y_0 = \begin{pmatrix} -61 \\ -21 \\ 99 \end{pmatrix}, z_0 = \begin{pmatrix} 31 \\ 32 \\ -68 \end{pmatrix}$$

donde $((l_1)_i, x_i) \to (\lambda_1, v_1)$, $((l_2)_i, y_i) \to (\lambda_2, v_2)$ y $((l_3)_i, z_i) \to (\lambda_3, v_3)$, con $\text{spec}(A) = \{v_i\}$. Así, resultó:

1.

$$(l_1)_{10} = 8.350301009359356 \sim \lambda_1 = 8.354544835161549$$

$$x_{10} = \begin{pmatrix} 0.083489119137721 \\ 0.979589587999048 \\ -0.182849134724406 \end{pmatrix} \sim v_1 = \begin{pmatrix} 0.083443662459397 \\ 0.979575954647159 \\ -0.182942898939720 \end{pmatrix}$$

2.

$$(l_2)_{10} = 1.213459758272933 \sim \lambda_2 = 1.224671629151526$$

$$y_{10} = \begin{pmatrix} 0.992727769096749 \\ 0.104839700638831 \\ 0.059162603341566 \end{pmatrix} \sim -v_2 = - \begin{pmatrix} -0.9927283126924627 \\ -0.104882397034991 \\ -0.059077745141235 \end{pmatrix}$$

3.

$$(l_3)_{10} = 3.412450413785564 \sim \lambda_2 = 3.420783535686916$$

$$z_{10} = \begin{pmatrix} -0.515302410902657 \\ 0.332388801317836 \\ -0.789924749628977 \end{pmatrix} \sim -v_2 = - \begin{pmatrix} 0.515310726732435 \\ -0.332385611228160 \\ 0.789920667132448 \end{pmatrix}$$

En cuanto a la convergencia de uno de los vectores propios aproximados, q_2 para ser exactos, no se corroboró la convergencia; al contrario, la secuencia dada por $(||y_i - v_2||)_{i=1}^{10}$ no converge cuadráticamente -a cero- como hubiéramos esperado:

$1.946583102868893, 1.944970485340590, 1.998600659570482, 0.040486316552812, 1.999899403881148,$

$0.008503334691331, 1.999998214257678, 1.999999999974186, 0.000009978020190, 1.999999999976001$

Al menos experimentalmente, la norma de la diferencia parece disminuir de manera cuadrática en cada iteración al principio; sin embargo, cuando los vectores se acercan y la diferencia disminuye, el error luego aumenta, con cierta periodicidad. La explicación que ofrecemos es por la precisión limitada de la computadora.

Ejercicio 3.5

Los eigenvalores de la matriz (dados por Matlab) son: λ_3 = -2.7631 y $\lambda_{1,2}$ = 2.8816 $\pm$ 2.7606i. Como $|\lambda_3| < |\lambda_2| = |\lambda_1|$. De aquí resulta que $\frac{1}{|\lambda_3|} > \frac{1}{|\lambda_2|} = \frac{1}{|\lambda_3|}$. Y una vez que nuestro vector inicial se escribe como combinación lineal de los eigenvectores de tal forma que el coeficiente del tercer eigenvector es distinto de cero, el método debe converger al tercero por ser el dominante de la matriz inversa. En efecto, tomando $q = (1, 1, 1)^T$ el método ofrece el valor propio aproximado $\lambda_3 = -2.7631$.

Ejercicio 3.6

Matlab	QR simple	QR dinámico
216.731946542820	216.731946542820	216.731946542819
-126.818277896823	-126.818277896822	0
-40.0301574197931	-40.0301574197930	0
-14.2402443490463	-14.2402443490463	0
-8.49982267007264	-8.49982267007263	-8.49982267007264
-5.23606797749979	-5.23606797749978	-5.23606797749979
-3.76682202913040	-3.76682202913040	-3.76682202913040
-2.75804237380977	-2.75804237380977	-2.75804237380977
-2.17749954289979	-2.17749954289979	-2.17749954289979
-1.74149065909789	-1.74149065909789	-1.74149065909789
-1.45606624643127	-1.45606624643128	-1.45606624643128
-1.23058947320439	-1.23058947320439	-1.23058947320439
-1.07089388885102	-1.07089388885102	-1.07089388885102
-.940919194916109	-.940919194916112	-.940919194916115
-.844042685997342	-.844042685997347	-.844042685997347
-.763932022500211	-.763932022500211	-.763932022500211
-.702278178498369	-.702278178498366	-.702278178498365
-.651115088803799	-.651115088803795	-.651115088803797
-.611149485968293	-.611149485968295	-.611149485968293
-.578379494075265	-.578379494075267	-.578379494075262
-.552973671851485	-.552973671851486	-.552973671851489
-.532961914294642	-.532961914294642	-.532961914294639
-.518255221991408	-.518255221991412	-.518255221991410
-.507979555929290	-.507979555929293	-.507979555929290
-.501985501335008	-.501985501335008	-.501985501335009

Aquí se muestran los valores usando QR simple con 2000 iteraciones y QR dinámico con 20 iteraciones. Como se puede ver, el método QR dinámico no alcanza a calcular todos los eigenvalores. Sin embargo, la aproximación de los demás es muy buena. El método QR simple tarda 1572 iteraciones en parar por el criterio de tolerancia relativo, mientras el QR con shift dinámico tarda 24 iteraciones únicamente.

El QR con shift dinámico vuelve algùn eigenvalor cercano a cero y eso acelera la convergencia (como en el método de la potencia inversa). También, al deflactar la matriz toma solamente la submatriz que nos interesa y eso ahorra muchas operaciones. Finalmente, este método toma una matriz Hessenberg, la cual tiene muchos ceros y evita hacer operaciones. Es por esto que el QR con shift dinámico tarda mucho menos que el QR simple.

Ejercicio 3.7

De acuerdo al script proporcionado, obtuvimos los siguientes resultados utilizando dentro de nuestro método SVD: el método QR con 1000 iteraciones y $tol = 10^{-10}$.

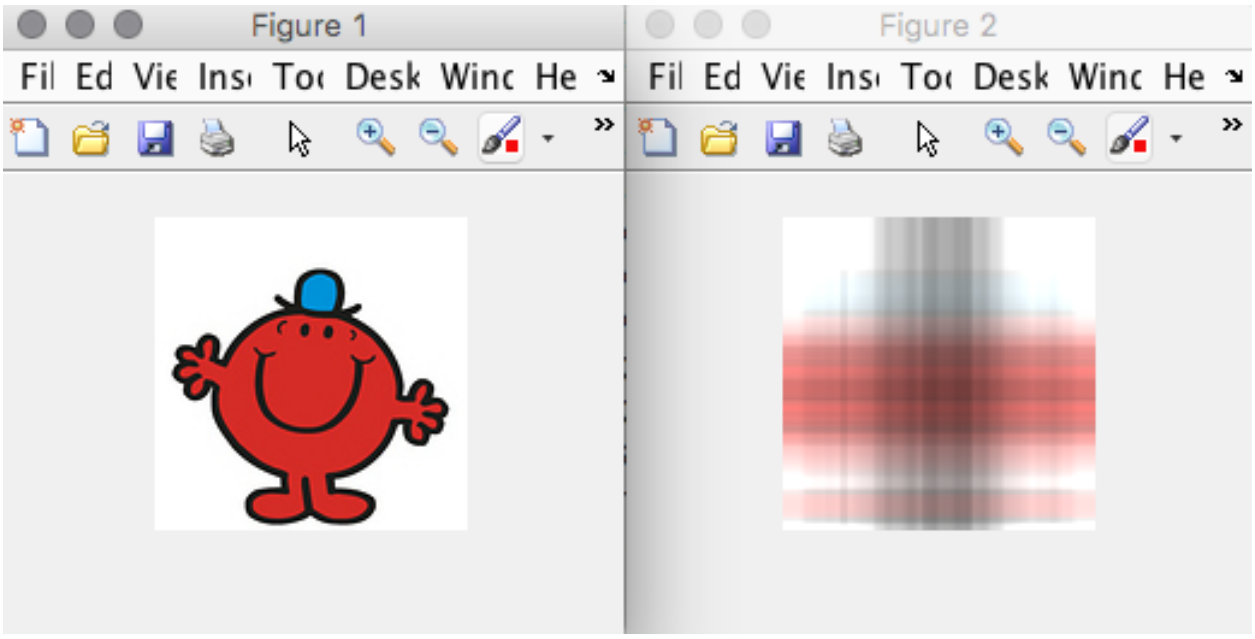

Figura 1: Primera iteración

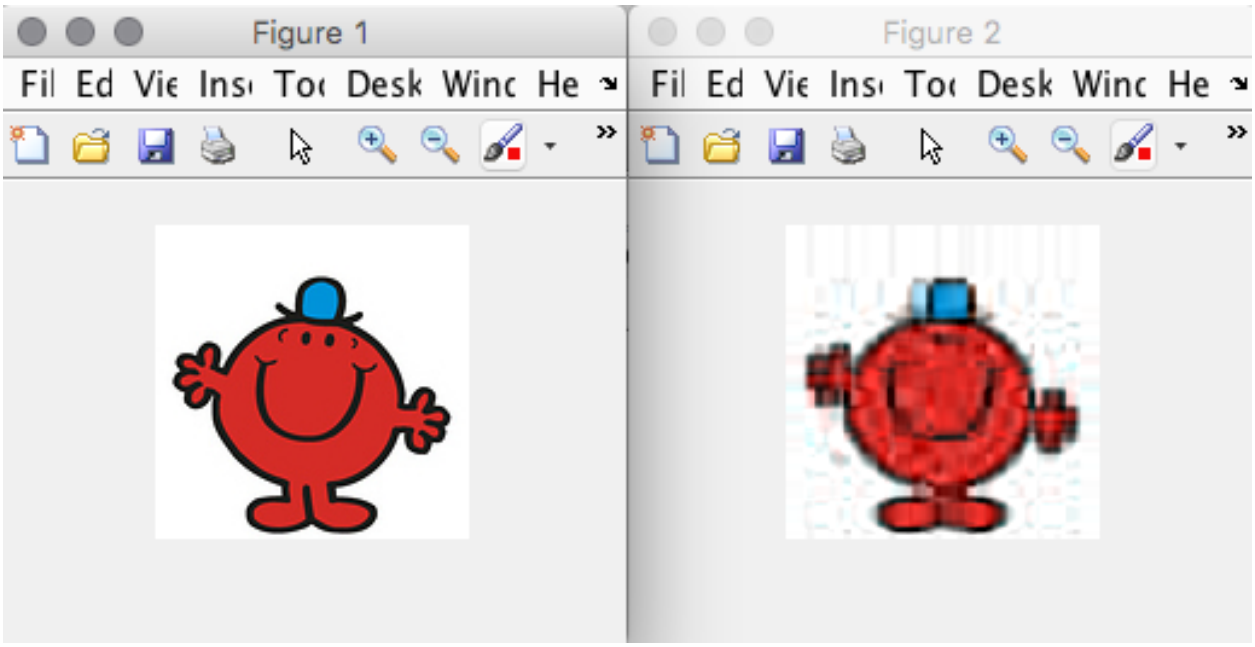

Figura 2: Segunda iteración

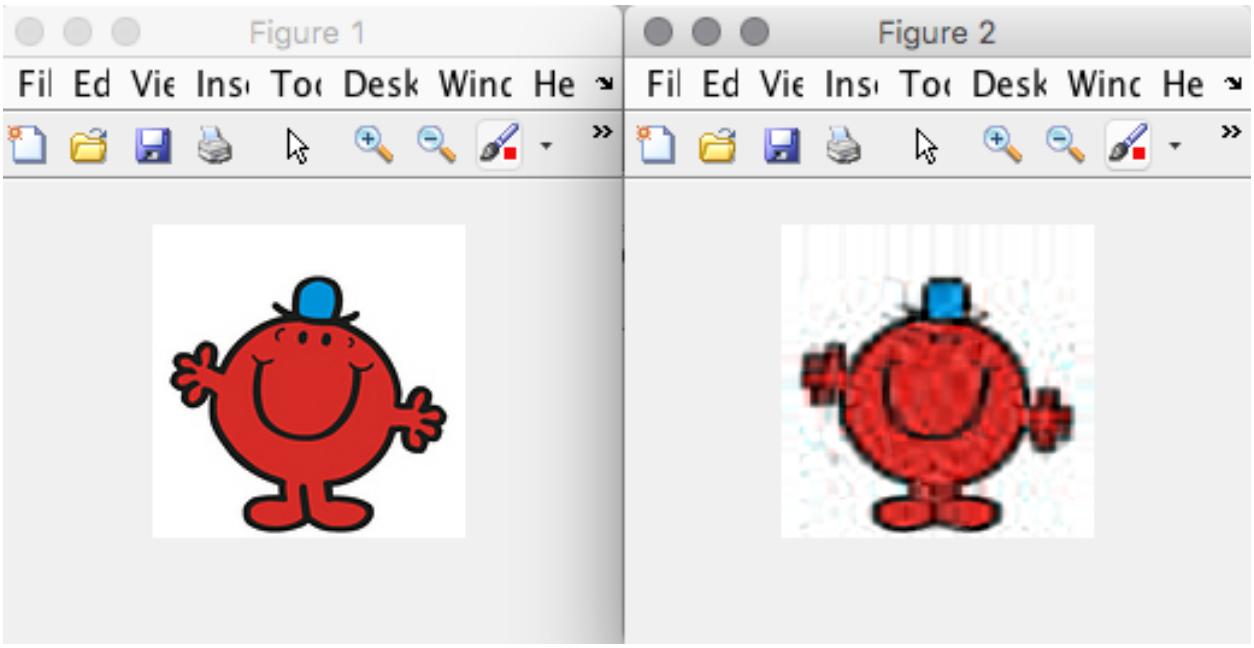

Figura 3: Tercera iteración

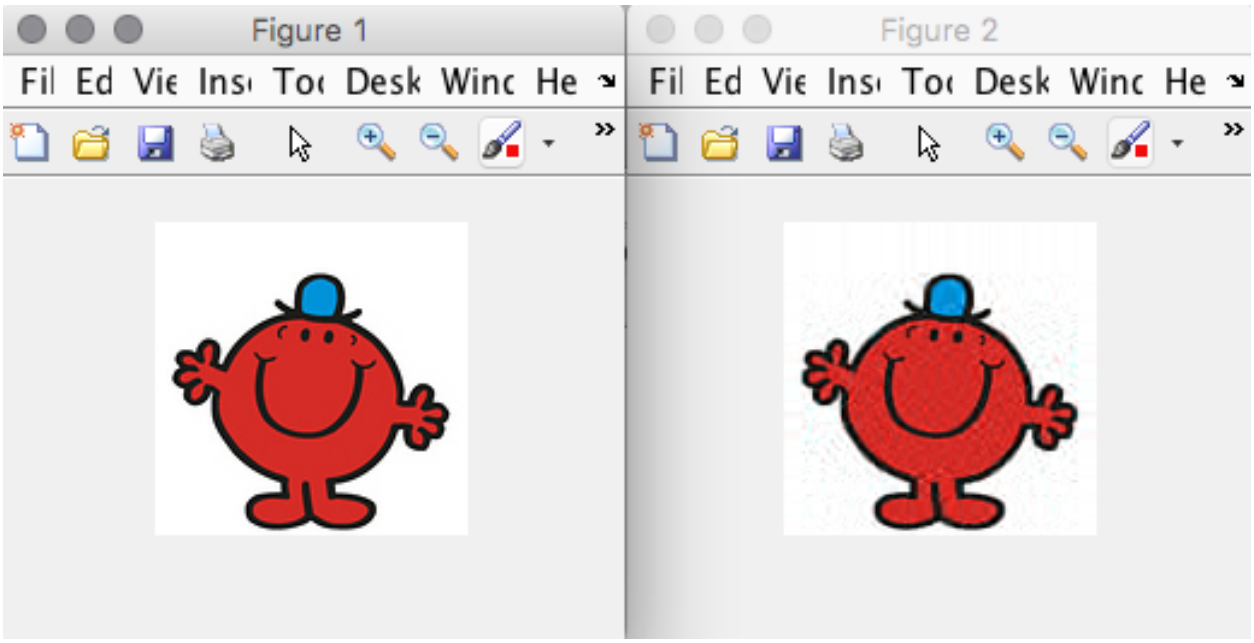

Figura 4: Cuarta iteración