Instituto Tecnológico Autónomo de México

Cálculo Numérico

Práctica 1: Valores y vectores propios

Andrés Cruz y Vera
Luis Felipe Landa Lizarralde
Mónica Elizabeth Alba González

23 de semptiembre de 2018

1 Introducción

En este documento se describen los resultados obtenidos en la elaboración de los ejercicios de la práctica 1.

Los métodos fueron implementados como los vistos en clase, a excepción de funciones auxiliares, las cuales son explicadas en el código de estas mismas.

2 Evaluación de resultados

2.1 Ejercicio 1

En este ejercicio comparamos la excatitud del valor propio de la implementación del método de la potencia visto en clase contra el resultado del método *eig* de matlab, dando como resultado un error de 0.008708; es decir, 3 dígitos de exactitud con solamente 10 iteraciones. Además, calculamos las razones

$$\frac{||q_{j+1} - v||_\infty}{||q_j - v||_\infty}$$

en cada iteración del método (esto se puede encontrar en la función *metodo_potencia_c*) y vemos que con cada iteración se va acercando cada vez más al radio de convergencia calculado con los valores de la matriz diagonal que regresa el método *eig*.

Con esto podemos concluir satisfactoriamente que nuestro método está bien implementado, ya que si aumentamos el número de iteraciones máximas aumenta o si modifica la tolerancia de modo que haya más dígitos de precisión, logramos cada vez más una mejor aproximación a los eigenvalores de la matriz.

Notas importantes : El método *metodo_potencia_c* es exáctamente el mismo que el método *metodo_potencia*, solo que en el primero se añade la convergencia, de modo que solo se impriman los resultados del inciso c) al llamar a esta nueva función.

2.2 Ejercicio 2

En este ejercicio usamos la misma matriz que se pide en el ejercicio anterior para aplicar el método de la potencia inversa con shift que vimos en clase. Nuestro método *metodo_potencia_inv* llama al método *metodo_potencia_inv_sin_shift* con la matriz original menos el shift.

Los resultados obtenidos fueron satisfactorios, en tan solo 10 iteraciones sin shift:

- diferencia entre eigenvalor y el valor obtenido con el método: 6.8841e-006

- radio de convergencia calculada con 3 dígitos de precisión

Los resultados obtenidos fueron satisfactorios, en tan solo 10 iteraciones con shift:

- diferencia entre eigenvalor y el valor obtenido con el método: 1.7390e-012
- radio de convergencia calculada con 3 dígitos de precisión

Notas importantes : Los métodos *metodo_potencia_inv* y *metodo_potencia_inv_sin_shift* reciben un parámetro extra: **vecRef** para poder implementar la razón que se pide en el inciso c) sin tener que duplicar el método como se hizo en el ejercicio anterior.

2.3 Ejercicio 3

Para aproximar los vectores propios utilizando el método de la potencia inversa con el cociente de Rayleigh usamos los vectores iniciales:

$q0_1 = (-1, 0.2, 0.5)^T$

$q0_2 = (-0.7, 0.1, 1.5)^T$

$q0_3 = (0.5, -0.99, -0.3)^T$

Cada vector convergió a un vector propio diferente a su vez asociado a un valor propio único. Para observar la velocidad de convergencia implementamos una función llamada *metodo_potencia_invRayleigh_convergencia* que junto con un vector de referencia calculaba el radio de convergencia en cada iteración. Usamos el vector $q0_3$ que correspondió al valor propio más grande para comparar la convergencia. Notamos de forma inmediata que la convergencia era significativamente más rápida pues con el shift dinámico el error porcentual se redujo en varios órdenes de magnitud con solo cuatro iteraciones. Más áun, con un número reducido de iteraciones el error era tan pequeño que el épsilon de la máquina advertía que lo hacía perder precisión.

2.4 Ejercicio 4

Al considerar la matriz A del problema y despúes de verificar que nuestro metodo de la potencia no convergía, aplicamos el comando *eig(A)* de matlab. Descubrimos que la matriz tiene un par conjugado de eigenvalores. Estos eigenvalores corresponden a dos eigenespacios diferentes pero tienen el mismo modulo. Así, se rompe la hipotesis de que hay un eigenvalor dominante, necesario para que el método funcione.

2.5 Ejercicio 5

El método QR simple realizó 162 iteraciones con una tolerancia de 1E-10, mientras que el método QR dinámico realizó 53 iteraciones con la misma tolerancia. Ambos dando errores relativos de orden de 10^{-16}.

Estos métodos están basados en las notas de clase y están mejorados con las notas del libro *Numerical Analysis de Timothy Sauer*[1].

2.6 Ejercicio 6

Para desarrollar la intuicion sobre el comportamiento del sistema de ranking implementamos un metodo *google(A)* que construye la matriz de google a partir de una matriz de adyacencia A dada. Escribimos la matriz de adyacencia correspondiente al caso de 7 nodos y aplicamos nuestro metodo de la potencia a su matriz de google. El eigenpar resultante consistió del eigenvalor 1 y un eigenvector con todas las entradas iguales. Este resultado nos mostró que el modelo daba la misma importancia a las páginas en el escenario planteado por el problema.
Desde una perspectiva teórica, la matriz de google es la transpuesta de lo que usualmente se llama la matriz de transición de una cadena de Markov. Las probabilidades estacionarias asociadas a dicha matriz se obtienen resolviendo un sistema de ecuaciones que involucra la suma ponderada de los elementos de cada columna. De forma equivalente, el mismo vector de probabilidades estacionarias corresponde a un eigenvector de la matriz transpuesta (en este caso la matriz de google).
Una condicion suficiente para que las probabilidades estacionarias sean iguales para cada estado es que la matriz sea *doblemente estocástica*, es decir, que tanto los renglones como las columnas sumen uno. Particularmente, la matriz de google es doblemente estocastica siempre que cada nodo tenga el mismo numero de flechas apuntando a él como flechas salen de él. Por esta razón, en los dos escenarios planteados en el problema el ranking de todas las páginas es el mismo.

2.7 Ejercicio 7

La matriz original que corresponde a la imagen sin comprimir tiene un rango máximo de 1500. Tras implementar el método de descomposición en valores singulares economizado obtuvimos un valor de $r = 187$. Este valor es casi 90% más pequeño que el rango original de la matriz.
En la descomposición SVD de la nueva matriz A_r tenemos un total de $1500 * 187 + 187 + 1500 * 187 = 595969$ datos o valores que aportan información. Esto se debe a que $U \in R^{1500x187}$, $\Sigma \in R^{187x187}$ y $V \in R^{1500x187}$. Este valor se compara con los $1500^2 = 2250000$ datos contenidos en la matriz original: la "economización" de los datos es clara.
La diferencia entre la imagen comprimida y la original es casi imperceptible

utilizando 2000 como el criterio de paro para los valores singulares. Al jugar con este criterio y subiendolo a 4500 nos quedamos con un valor de $r = 73$ que aunque hace la compresión muy clara, sigue siendo aceptable a nuestros ojos. También es importante notar que si usamos 0 como nuestro criterio de paro obtenemos todos los valores propios entonces nuestro método de SVD sirve para descomposiciones totales también. Con estos resultados podemos concluir que nuestra implementación del método SVD es versátil, correcta y tiene aplicaciones en un gran rango de situaciones.

References

[1] T. Sauer. *Numerical Analysis*. Pearson, 2006.