Obtención de Autovalores en una Matriz Grcar utilizando un Algoritmo QR

Pablo I. Chervonagura [1] *Jorge E. Gómez* [2] *Lucas A. Farré* [3] *Khalil Loukili* [4]

[1] ITBA - Instituto Tecnológico de Buenos Aires
pchervon@itba.edu.ar
[2] ITBA - Instituto Tecnológico de Buenos Aires
jogomez@itba.edu.ar
[3] ITBA - Instituto Tecnológico de Buenos Aires
lfarre@itba.edu.ar
[4] INSA Toulouse - Institut National des Sciences Appliquées de Toulouse
kloukili@etud.insa-toulouse.fr

Resumen

Este informe explica los métodos utilizados para la generación de una matriz grcar, y la obtención de sus autovalores a través de un algoritmo QR. Se utilizaron distintos programas que fueron implementados en GNU Octave sin la necesidad de utilizar las funciones principales ya implementadas en su librería. Luego se adjuntan una serie de comparaciones de tiempos y dimensiones de la matriz grcar para poder comparar el alcance del algoritmo.

Palabras claves: Autovalores, matriz grcar, métodos numéricos, algoritmo QR

1 Introducción

El siguiente trabajo consiste en cumplir con el principal objetivo de poder encontrar los autovalores de una matriz grcar utilizando un método numérico implementado en un código simple e intuitivo. Esto también debe cumplirse de forma eficiente y práctica. Debido a estos requerimientos, se decidió modelar el problema en código de GNU Octave por su practicidad y funcionamiento con matrices especialmente.

Una matriz grcar es una matriz de Toeplitz con valores sensibles. Es una matriz cuadrada en la que los elementos de sus diagonales (de izquierda a derecha) son constantes [1]. Los elementos de la subdiagonal son -1, los de la diagonal son 1 y las tres primeras superdiagonales también tienen 1 en su valor. El resto de los elementos son 0. Grcar es también una matriz de Hessenberg, tiene todos ceros por debajo de la primera subdiagonal [2].

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & & & & \\ -1 & 1 & 1 & 1 & 1 & & & \\ & -1 & 1 & 1 & 1 & 1 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & -1 & 1 & 1 & 1 & 1 \\ & & & & -1 & 1 & 1 & 1 \\ & & & & & -1 & 1 & 1 \\ & & & & & & -1 & 1 \end{pmatrix}$$

Figura 1. Matriz grcar de *n x n*

Al querer obtener los autovalores de esta matriz, esto puede convertirse en una tarea tediosa, sobretodo si se trata de una matriz con dimensiones grandes. Por ello, es sumamente práctico contar con una forma rápida de lograr esto. A partir de esto se puede decir que la dificultad principal consiste en escribir un algoritmo eficiente y con una convergencia rápida.

La estructura del informe consiste de tres secciones. En la primera sección se detalla la introducción al trabajo. En la segunda sección se establece la metodología utilizada para resolver el problema, incluyendo el planteo del modelo, las funciones utilizadas con sus explicaciones y los análisis obtenidos de ellas. La tercera sección muestra los resultados obtenidos luego de las implementaciones detalladas en la sección anterior. En la cuarta sección se elaboran las conclusiones que se pueden obtener de los resultados. La quinta sección, la última, consiste en la bibliografía, citando fuentes utilizadas y referenciadas para la elaboración del trabajo y el informe.

2 Metodología

Para poder obtener los autovalores de la matriz grcar, el primer paso planteado fue el de modelar esta matriz ante un valor *n* deseado que indica la dimensión de esta matriz. La función grcar se encarga de esto. Luego, se utilizó un algoritmo *QR* con desplazamiento del coeficiente de *Rayleigh* para lograr una convergencia más rápida que con el algoritmo tradicional. El hecho de que grcar sea una matriz de Hessenberg también contribuye a disminuir la complejidad del algoritmo. La implementación se encuentra en la función *qrrayleigh*. La función *ourqr* fue escrita para evitar utilizar el método provisto por GNU Octave para descomponer una matriz en las matrices *Q* y *R*. Esta sirve como función auxiliar al algoritmo principal. Finalmente, para conseguir los autovalores de la matriz del resultado de *qrrayleigh*, se utiliza la función *eigen*, que recorre su diagonal y obtiene los autovalores finales, sean reales o complejos, y devuelve un vector con ellos.

2.1 Funciones

- ***mainQR***

Función principal a ejecutar en Octave. Ejecuta el algoritmo y muestra el tiempo que tarda en ejecutarse el mismo.

```
function ret = mainQR(n)
	t0 = time();
	if n <= 0
		error("La dimensión de la matriz debe ser un número
positivo.");
	elseif n == 1
		ret = 1;
	else
		ret = eigen(qrrayleigh(grcar(n)));
	endif
	printf("Calculado en %f segundos.\n", time() - t0);
endfunction
```

- ***grcar***

  Función que dado un valor *n*, retorna una matriz grcar de tamaño *n x n*.

```
function ret = grcar(n)
	m = zeros(n, n);
	for i = 1:n
		for j = i:min(i + 3, n)
			m(i,j) = 1;
		endfor
		if i > 1
			m(i, i-1) = -1;
		endif
	endfor
	ret = m;
endfunction
```

- ***ourqr***

Función que dada una matriz cuadrada *A*, retorna su descomposición *QR* utilizando el método Gram-Schmidt. Cumpliendo que $A = QR$, donde *Q* es una matriz ortogonal ($Q^T Q = I$) y *R* es una matriz triangular superior [3].

```
function [q,r] = ourqr(A)
	n = rows(A);
	v = zeros(n,n);
	r = eye(n,n);
	q = eye(n,n);
	for k = 1:n
		v(:,k) = A(:,k);
	endfor
	for k = 1:n
		r(k,k) = norm(v(:,k),2);
		q(:,k) = v(:,k)/r(k,k);
		for j = (k+1):n
			r(k,j) = q(:,k)'*v(:,j);
			v(:,j) = v(:,j) - r(k,j)*q(:,k);
		endfor
	endfor
endfunction
```

- ***qrrayleigh***

Función que dada una matriz *H* (de Hessenberg), la misma es retornada en su forma Schur, que tiene la particularidad de ser una matriz triangular superior que posee los mismos autovalores de *H* y además se encuentran sobre la diagonal principal. En el caso de contener autovalores complejos, estos se encuentran dentro de una matriz de *2 x* 2 dentro de la diagonal.

El algoritmo implementado se denomina Algoritmo Hessenberg QR con desplazamiento del coeficiente de Rayleigh (ver *sigma* en el código) donde se destaca la precisión que se alcanza en la aproximación final de los autovalores [4].

```
function ret = qrrayleigh(H)
    n = rows(H);
    itfactor = 1;
    for k = 1:ceil(n*n*itfactor)
        sigma = H(n, n);
        [Q, R] = ourqr(H - sigma * eye(n));
        H = R*Q + sigma * eye(n);
    endfor
       ret = H;
endfunction
```

- ***qrwilkinson***

Dado a la disconformidad que nos generaba que la cantidad de iteraciones del algoritmo anterior dependiera de la dimensión de la matriz, decidimos buscar una variante, y encontramos otro algoritmo que utiliza el desplazamiento de *Wilkinson* (ver *sigma* en el código) en lugar del de *Rayleigh*. El mismo tiene una cota de error que la definimos en 0.0001, ya que luego de realizar varias pruebas, este valor fue el que mejor se adecuó a la relación precisión-tiempo.

```
function ret = qrwilkinson(H)
   n = rows(H);
   m = n;
   precision = 1e-4;
   while m > 1
      do
```

```
        sigma = max(eigen([H(n-1,n-1),H(n-1,n);H(n,n-
1),H(n,n)]));
        [Q, R] = ourqr(H - sigma * eye(n));
        H = R*Q + sigma * eye(n);
      until(abs(H(m, m-1)) < precision)
      m = m - 1;
   endwhile
   ret = H;
endfunction
```

- ***eigen***

Función que dada una matriz *M* que contiene los autovalores de *A* en la diagonal, retorna los autovalores de *A*. Esta función detecta si estos, son autovalores reales o complejos. Si estos son complejos descompone la matriz en matrices *2 x 2* y calcula los autovalores de estas utilizando el método $\det(F - \lambda I) = 0$. Donde *F* es la matriz de *2 x 2*.

La metodología utilizada para saber si el elemento m_{ii} es un autovalor, o forma parte de una matriz que contiene autovalores complejos, consta de tomar el valor m_{ii+1} y verificar si este es distinto de 0. Si el valor es distinto de 0, m_{ii} forma parte de una matriz F que contiene autovalores complejos, donde:

$$F = \begin{pmatrix} m_{i,i} & m_{i+1,i} \\ m_{i,i+1} & m_{i+1,i+1} \end{pmatrix}$$

```
function ret = eigen(M)
      n = rows(M);
      ret = zeros(n, 1);
      k = 0;
      i = 1;
      while i < n
            if M(i + 1, i) == 0 && k < i
                  ret(k + 1, 1) = M(i, i);
                  k = k + 1;
                  i = i + 1;
            elseif
                  E = roots([1, (-M(i,i)-M(i+1,i+1)),
                  (M(i+1,i+1)*M(i,i)-M(i+1,i)*M(i,i+1))]);
                  ret(k+1, 1) = E(1);
                  ret(k+2, 1) = E(2);
                  k = k + 2;
                  i = i + 2;
            endif
      endwhile
      if i == n
            ret(k+1, 1) = M(i,i);
      endif
```

```
endfunction
```

3 Resultados

Para corroborar la funcionalidad de las funciones utilizadas, realizamos diferentes pruebas con diferentes tamaños de matrices Grcar. De las mismas calculamos los autovalores utilizando nuestras funciones y las funciones que proporciona Octave, y obtuvimos los siguientes resultados:

	Octave		**Rayleigh**		**Wilkinson**	
Dim	**Autovalores**	**Tiempo**	**Autovalores**	**Tiempo**	**Autovalores**	**Tiempo**
2 x 2	1±1i	0.00023 s	1±1i	0.00201 s	1±1i	0.001722 s
3 x 3	1.4534 0.7733± 1.4677i	0.00029 s	1.4534 0.7733± 1.4677i	0.00501 s	1.4534 0.7733± 1.4677i	0.018917 s
5 x 5	1.4448 1.3315± 1.0474i 0.4461± 1.8489i	0.00113 s	1.4448 1.3315± 1.0474i 0.4461± 1.8489i	0.02244 s	1.4448 1.3315± 1.0474i 0.4461± 1.8489i	0.091620 s
10 x 10	0.1980± 2.1293i 0.5648± 1.7599i 1.1281± 1.2781i 1.5266± 0.4024i 1.5825± 1.0195i	0.00071 s	0.1979± 2.1292i 0.5647± 1.7599i 1.1281± 1.2781i 1.5265± 0.4024i 1.5825± 1.0194i	0.22648 s	0.1980± 2.1293i 0.5648± 1.7599i 1.1281± 1.2781i 1.5266± 0.4024i 1.5826± 1.0195i	0.241798 s

50 x 50	0.0772± 2.2568i 0.0970± 2.2371i 0.1297± 2.2044i 0.1754± 2.1593i 0.2338± 2.1022i 0.3046± 2.0339i 0.3875± 1.9554i 0.4823± 1.8679i 0.5885± 1.7729i 0.7057± 1.6724i 0.8334± 1.5689i 0.9708± 1.4659i 1.1161± 1.3678i 1.2661± 1.2806i 1.4139± 1.2116i 1.5460± 1.1667i 1.5965± 0.0934i 1.5994± 0.2788i 1.6053± 0.4598i 1.6150± 0.6327i 1.6287± 0.7930i 1.6407± 1.1437i 1.6463± 0.9341i 1.6648± 1.0467i 1.6743± 1.1183i	0.00608 s	0.0772± 2.2568i 0.0970± 2.2371i 0.1297± 2.2044i 0.1754± 2.1593i 0.2338± 2.1022i 0.3046± 2.0339i 0.3875± 1.9554i 0.4823± 1.8679i 0.5885± 1.7729i 0.7057± 1.6724i 0.8334± 1.5689i 0.9708± 1.4659i 1.1161± 1.3678i 1.2661± 1.2806i 1.4139± 1.2116i 1.5460± 1.1667i 1.5965± 0.0934i 1.5994± 0.2788i 1.6053± 0.4598i 1.6150± 0.6327i 1.6287± 0.7930i 1.6407± 1.1437i 1.6463± 0.9341i 1.6648± 1.0467i 1.6743± 1.1183i	111.908 s	0.0772± 2.2568i 0.0970± 2.2371i 0.1297± 2.2044i 0.1754± 2.1593i 0.2338± 2.1022i 0.3046± 2.0339i 0.3875± 1.9554i 0.4823± 1.8679i 0.5885± 1.7729i 0.7057± 1.6724i 0.8334± 1.5689i 0.9708± 1.4659i 1.1161± 1.3678i 1.2661± 1.2806i 1.4139± 1.2116i 1.5460± 1.1667i 1.5965± 0.0934i 1.5994± 0.2788i 1.6053± 0.4598i 1.6150± 0.6327i 1.6287± 0.7930i 1.6407± 1.1437i 1.6463± 0.9341i 1.6648± 1.0467i 1.6743± 1.1183i	258.175771 s

4 Conclusiones

Llegamos a la conclusión de que para pequeños valores de *n*, al ser matrices de menor tamaño, el algoritmo de *Rayleigh* itera menor cantidad de veces, y por lo tanto el mismo demora menos en ejecutarse. Para matrices de tamaños muy grandes, se puede observar que el mismo llega a demorar más de cien segundos, lo cual nos pareció una duración elevada, aún así, con cuatro decimales de redondeo, se logró tener una precisión muy alta si se compara con el algoritmo que utiliza *Octave*. No obstante, modificando el valor de la constante *itfactor,* en el algoritmo de *Rayleigh* se puede disminuir la precisión del mismo y aumentar su velocidad de respuesta. Su valor se encuentra entre 0 (no inclusive) y 1, esto disminuye la cantidad de iteraciones del algoritmo. Utilizando un valor de 0.5 se puede reducir aproximadamente a la mitad el tiempo de respuesta.

Comparando el algoritmo que utiliza *Rayleigh* con el que utiliza *Wilkinson* se puede notar una gran diferencia en el tiempo de ejecución de los mismos, pero no de precisión, y lamentablemente, el de *Wilkinson* no cumplió nuestras expectativas de mejorar el tiempo.

Si bien ambos algoritmos demoran una gran cantidad de tiempo, estamos conformes con los resultados obtenidos, pues obtuvimos los autovalores correspondientes a la matriz grcar con una excelente precisión.

5 Bibliografía

[1] http://es.wikipedia.org/wiki/Matriz_de_Toeplitz, Wikipedia.

[2] http://es.wikipedia.org/wiki/Matriz_de_Hessenberg, Wikipedia.

[3] http://es.wikipedia.org/wiki/Factorizaci%C3%B3n_QR, Wikipedia.

[4] http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter3.pdf, ETH Zürich, Dr. Peter Arbenz.

[5] http://www.math.iit.edu/~fass/477577_Chapter_11.pdf, Illinois Institute of Technology.