Centro de Investigación en Matemáticas A. C.

Maestría en Ciencias con especialidad en Computación y Matemáticas Industriales

Métodos Numéricos (2019)

Tarea 06 – Eigensolvers

Reporte

Marco Antonio Esquivel Basaldua (MCMESQ)

Los métodos numéricos iterativos consisten en métodos cuya función es aproximarse cada vez más al resultado buscado conforme las iteraciones en el programa van avanzando, entiéndase aquí una iteración como una “vuelta” en un código de programa o parte de él.

En el presente reporte de tarea se muestran y explican los métodos iterativos del método de la potencia inversa para encontrar el mínimo eigenvalor de una matriz cuadrada y su eigenvector asociado; el método de la potencia con deflación para encontrar los k valores propios más grandes; el método de la potencia inversa con deflación para encontrar los k valores propios más chicos; en ambos casos k es un valor entero y positivo; y el método de Jacobi para encontrar todos los valores propios de una matriz cuadrada y sus respectivos vectores propios asociados.

Los códigos utilizados para estas aplicaciones están desarrollados en el lenguaje C y creados y compilados en el entorno de desarrollo *Visual Studio Code*.

1. Método de la potencia inversa

El método de la potencia inversa es un método iterativo que calcula sucesivas aproximaciones al autovalor más chico de una matriz simétrica y su autovector asociado.

Para aplicar este método se tiene una matriz cuadrada y simétrica de nxn. El método comienza por tomar cualquier vector V0 inicializado aleatoriamente, en cada paso i se calcula:

$$V_{i+1} = A^{-1} * V_i$$

$$\lambda_i = \frac{V_{i+1} * V_i}{V_{i+1}^2}$$

Cuando el valor de λ_i converge (es decir que no cambia o cambia extremadamente poco de una iteración a otra) entonces λ_i es el menor autovalor y V_i es el autovector asociado.

Algoritmo:

- *Se inicializa V0 con valores aleatorios*
- *Se inicializa lambda_old a cero*
- *Se define la tolerancia*
- *Mientras no se alcance el máximo de iteraciones*
    - Normalizar V0
    - V1 = A^{-1}*V0
    - Lambda = (V0*V1) / $V1^2$
    - Si la diferencia en valor absoluto entre lambda y lambda_old es menor o igual a una tolerancia
        - lambda es el autovalor más chico
        - V0 es el autovector más chico
        - Se detienen las iteraciones
    - En otro caso
        - lambda_old = lambda
        - V0 = V1
    - End
- End

(Nota: en la implementación de los códigos en C hay que tomar en cuenta que los índices comienzan en 0 y terminan en n-1)

Los resultados obtenidos en la ejecución son mostrados en la terminal.

Una posible mejora es desarrollar la factorización para que los valores de L y U sean guardados sobre la misma matriz A.

2. Método de la potencia con deflación

El método de la potencia con deflación es un método que encuentra los k valores propios más grandes de una matriz cuadrada simétrica y sus vectores propios asociados. Para esto se utiliza una versión extendida del método de la potencia a la cual se le agrega un ciclo más que se repite de acuerdo al valor de k. En este nuevo ciclo se lleva a cabo la deflación que consiste en ir eliminando las proyecciones de los vectores propios ya encontrados del vector de inicio generado para entrar a las iteraciones del método de la potencia.

Algoritmo:

- *Mientras no se hayan calculado los k autovalores y autovectores*
    - *Inicializar V0 con valores aleatorios*
    - *Si se está calculando desde el segundo autovalor en adelante*
        - *Eliminar de V0 las proyecciones de los vectores propios ya encontrados*
    - *Aplicar el método de la potencia*

    - *Guardar autovalor y autovector encontrado*
- *end*

(Nota: en la implementación de los códigos en C hay que tomar en cuenta que los índices comienzan en 0 y terminan en n-1)

Debido a la cantidad de valores generados y a la posible implementación de los mismos en en futuras aplicaciones, los datos generados son guardados en archivos de texto .txt, uno para los valores propios (“Eigenvalues.txt”) y otro para los vectores propios (“Eigenvectos.txt”), estos archivos se guardan en la misma carpeta donde el código de lenguaje C se encuentra. Una forma de optimizar la ejecución del código es llevando a cabo operaciones en cómputo peralelo cuando estas lo permiten, como por ejemplo en la multiplicación matrices y vectores.

3. Método de la potencia inversa con deflación

El método de la potencia inversa con deflación es un método que encuentra los k valores propios más chicos de una matriz cuadrada simétrica y sus vectores propios asociados. Para esto se utiliza una versión extendida del método de la potencia inversa a la cual se le agrega un ciclo más que se repite de acuerdo al valor de k. En este nuevo ciclo se lleva a cabo la deflación que consiste en ir eliminando las proyecciones de los vectores propios ya encontrados del vector de inicio generado para entrar a las iteraciones del método de la potencia inversa.

Algoritmo:

- *Mientras no se hayan calculado los k autovalores y autovectores*
    - *Inicializar V0 con valores aleatorios*
    - *Si se está calculando desde el segundo autovalor en adelante*
        - *Eliminar de V0 las proyecciones de los vectores propios ya encontrados*
    - *Aplicar el método de la potencia inversa*
    - *Guardar autovalor y autovector encontrado*
- *end*

(Nota: en la implementación de los códigos en C hay que tomar en cuenta que los índices comienzan en 0 y terminan en n-1)

Debido a la cantidad de valores generados y a la posible implementación de los mismos en en futuras aplicaciones, los datos generados son guardados en archivos de texto .txt, uno para los valores propios (“Eigenvalues.txt”) y otro para los vectores propios (“Eigenvectos.txt”), estos archivos se guardan en la misma carpeta donde el código de lenguaje C se encuentra. Una forma de optimizar la ejecución del código es llevando a cabo operaciones en cómputo peralelo cuando estas lo permiten, como por ejemplo en la multiplicación matrices y vectores.

4. Método de Jacobi.

El método de Jacobi funciona para todas las matrices simétricas reales. Este método hace uso de matrices de rotación para diagonalizar la matriz de entrada A que es la matriz de la que nos interesa encontrar los valores y vectores propios. Las matrices de rotación se aplican de la siguiente forma en el método:

$$R_n^T \dots R_2^T R_1^T A R_1 R_2 \dots R_n = \Lambda$$

$$\Phi = R_1 R_2 \dots R_n$$

Donde:

A es la matriz interesada en encontrar sus valores y vectores propios

R_n *es cada una de las matrices de rotación necesarias para diagonalizar* A

Λ *es una matriz diagonal cuyas entradas en la diagonal son los valores propios de de* A

Φ *es una matriz cuadrada cuyas colunas son los vectores propios asociados a los valores propios calculados*

Las matrices de rotación son de la forma:

$$\boldsymbol{R} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & \cos\phi & \cdots & \sin\phi & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & -\sin\phi & \cdots & \cos\phi & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow \text{row } i \\ \\ \leftarrow \text{row } j \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & \text{col } i & & \text{col } j & & \end{matrix}$$

Donde i es la fila del valor más grande de A en la iteración n y j es la columna. El valor del ángulo φ se calcula de la siguiente forma:

$$\phi = \frac{1}{2} atan2\left(\frac{2a_{ij}}{a_{ii} - a_{jj}}\right)$$

Algoritmo:

- A es la matriz de interés
- FI es la matriz donde se van a guardar los eigenvectores
- FI es inicializada como la matriz identidad
- toler es una tolerancia cercana a cero que se tomará como criterio de convergencia
- maxIter es el máximo de iteraciones para evitar que el algoritmo se ejecute indefinidamente
- Mientras no se haya alcanzado el máximo de iteraciones:

- Se registran los valores i, j como la posición del valor más grande de A fuera de su diagonal
- Si abs(A[i][j]) <= toler
  - Si se cumple la condición se detienen las iteraciones y se recupera A como valores propios y FI como los vectores propios
  - Si no se cumple la condición:
    - φ = 0.5*Atan2(2*A[i][j] , A[i][i] - A[j][j])
    - for l = 1:n
      - A[l][i] = A[l][i]*cos(φ) + A[l][j]*sin(φ)
      - A[l][j] = -A[l][i]*sin(φ) + A[l][j]*cos(φ)
      - FI[l][i] = Fi[l][i]*cos(φ) + FI[l][j]*sin(φ)
      - A[l][j] = -FI[l][i]*sin(φ) + FI[l][j]*cos(φ)
    - end
    - for m = 1:n
      - A[i][m] = A[i][m]*cos(φ) + A[j][m]*sin(φ)
      - A[j][m] = -A[i][m]*sin(φ) + A[j][m]*cos(φ)
    - end
- end

(Nota: en la implementación de los códigos en C hay que tomar en cuenta que los índices comienzan en 0 y terminan en n-1)

Debido a la cantidad de valores generados y a la posible implementación de los mismos en en futuras aplicaciones, los datos generados son guardados en archivos de texto .txt, uno para los valores propios (“Eigenvalues.txt”) y otro para los vectores propios (“Eigenvectos.txt”), estos archivos se guardan en la misma carpeta donde el código de lenguaje C se encuentra. Una forma de optimizar la ejecución del código es llevando a cabo operaciones en cómputo peralelo cuando estas lo permiten, como por ejemplo en la multiplicación matrices y vectores.