CIMAT

Centro de Investigación en Matemáticas, A.C.
Métodos Numéricos

Tarea 07

José Miguel Saavedra Aguilar

Resumen

En esta tarea se explora el método de Iteración en subespacios y el método del cociente de Rayleigh para encontrar los eigenpares de una matriz. También se estudia la factorización QR de una matriz y el método de gradiente conjugado para sistemas lineales.

1. Metodología

1.1. Método de Iteración en Subespacios

Sea $A \in \mathbb{R}^{n\times n}$ simétrica. El teorema de Factorización Espectral nos indica que existen $Q, \lambda \in \mathbb{R}^{n\times n}$ con Q ortogonal y Λ diagonal tales que $A = Q\Lambda Q^\top$. Las columnas de Q son los eigenvectores y los elementos de la diagonal de Λ son sus respectivos eigenvalores. En el método de iteración en subespacios buscamos encontrar $p < n$ eigenpares de A. Sea Φ^n una matriz ortogonal que aproxima p eigenvectores de A. Definimos $K^n = {\Phi^n}^\top A\Phi^n$ y obtenemos Z^n, D^n tales que $K^n = Z^n D^n {Z^n}^\top$, esto es, la descomposición espectral de K^n. Posteriormente, $\tilde{\Phi}^n = \Phi^n Z^n$, para finalmente tener $\Phi^{n+1} = f(\tilde{\Phi}^n)$, donde f es una función de actualización de $\tilde{\Phi}^n$.

Para obtener los eigenpares absolutamente más grandes, la función de actualización f será una iteración del método de la potencia generalizado para los p vectores propios más grandes de A. De forma similar, si deseamos obtener los eigenpares absolutamente más pequeños, la función de actualización f será una iteración del método de la potencia inversa generalizado para los p vectores propios más pequeños de A.

Si las columnas de Φ^n son efectivamente eigenvectores de A, K^n será diagonal, $Z^n = \mathbf{I}(p)$ y $\tilde{\Phi}^n = \Phi^n$.

1.2. Método del cociente de Rayleigh

Sea $A \in \mathbb{R}^{n\times n}$ y sea λ^0, v^0 una aproximación de λ, v un eigenpar de A. El método de Rayleigh consiste en actualizar v^n resolviendo el sistema:

$$(A - \lambda^n \mathbf{I})v^{n+1} = v^n$$

Si $Av^{n+1} \approx \lambda^{n+1} v^{n+1}$, tenemos que

$$\begin{aligned} {v^{n+1}}^\top (A - \lambda^n \mathbf{I}) v^{n+1} &= {v^{n+1}}^\top v^n \\ {v^{n+1}}^\top (\lambda^{n+1} - \lambda^n) v^{n+1} &= {v^{n+1}}^\top v^n \\ (\lambda^{n+1} - \lambda^n) &= \frac{{v^{n+1}}^\top v^n}{{v^{n+1}}^\top v^{n+1}} \end{aligned}$$

De esta forma, $\lambda^{n+1} = \lambda^n + \frac{{v^{n+1}}^\top v^n}{{v^{n+1}}^\top v^{n+1}}$.

1.3. Factorización QR de una matriz

Sea $A \in \mathbb{R}^{n \times n}$. La factorización QR de una matriz consiste en encontrar $Q, R \in \mathbb{R}^{n \times n}$ tales que $A = QR$, $Q^\top Q = \mathbf{I}$ y R es triangular superior. Notamos $A_{:,1} = QR_{:,1}$, además $R_{:,1} = \begin{bmatrix} R_{1,1} & 0 & \dots & 0 \end{bmatrix}^\top$, por lo que:

$$A_{:,1} = R_{1,1} q_1$$

donde $q_i = Q_{:,i}$ es la i-ésima columna de Q. Como Q es ortogonal, $Q^\top A = R$, de forma que para $j < i$,

$$R_{i,j} = q_i^\top A_{:,j}$$

Así mismo, $A_{:,i} = R_{i,i} q_i + \sum_{j=1}^{i-1} R_{j,i} q_j$, de forma que

$$\begin{aligned} q_i^\top A_{:,i} &= q_i^\top R_{i,i} q_i + q_i^\top \left(\sum_{j=1}^{i-1} R_{j,i} q_j \right) q_i^\top \\ q_i^\top A_{:,i} &= R_{i,i} \\ \tilde{q}_i &= A_{:,i} - \sum_{j=1}^{i-1} R_{j,i} q_j \\ q_i &= \frac{1}{\|\tilde{q}_i\|_2} \tilde{q}_i \\ R_{i,i} &= \|\tilde{q}_i\|_2 \end{aligned}$$

Una vez teniendo $A = QR$, podemos resolver el sistema $Ax = b$ resolviendo $Rx = Q^\top b$, recordando que R es triangular superior.

1.4. Gradiente Conjugado

Sea $A \in \mathbb{R}^{n \times n}$ simétrica y positiva definida, $b \in \mathbb{R}^n$ y considere el sistema $Ax = b$. El problema

$$x^* = \underset{x \in \mathbb{R}^n}{\text{arg mín}} \quad \phi(x) := \frac{1}{2} x^\top A x - b^\top x \tag{1}$$

es equivalente a resolver el sistema $Ax = b$, esto pues $\nabla\phi(x^*) = Ax^* - b = 0$ y $\nabla^2\phi(x^*) = A$. En el método de gradiente conjugado buscamos un conjunto de vectores A conjugados $\{p_0, \ldots, p_n\}$, pues de esta forma es posible resolver (1) en a lo más n pasos. Esto es posible si tenemos $x_0 \approx x^*$ y $\{p_0, \ldots, p_n\}$ con $p_i^\top A p_j = 0$ para $i \neq j$, definiendo:

$$r_k = Ax_k - b \tag{2}$$

$$\alpha_k = -\frac{r_k^\top p_k}{p_k^\top A p_k} \tag{3}$$

$$x_{k+1} = x_k + \alpha_k p_k \tag{4}$$

de esta forma $x_n = x^*$, ver [1].

En el método de gradiente conjugado buscamos construir las direcciones de descenso p_k como una combinación lineal de la dirección de máximo descenso, $\nabla\phi(x_k) = r_k$ y p_{k-1}, de forma que:

$$p_k = -r_k + \beta_k p_{k-1} \tag{5}$$

de forma que p_k, p_{k-1} sean A conjugados, de forma que $0 = p_{k-1}^\top A p_k = -p_{k-1}^\top A r_k + \beta_k p_{k-1}^\top A p_{k-1}$, teniendo así:

$$\beta_k = \frac{r_k^\top A p_{k-1}}{p_{k-1}^\top A p_{k-1}} \tag{6}$$

Es posible verificar [1] que α_k y β_k pueden ser calculadas de la forma:

$$\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k} \tag{7}$$

$$\beta_{k+1} = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k} \tag{8}$$

1.4.1. Precondicionamiento

Es posible acelerar el método del gradiente conjugado si tenemos información de la distribución de los eigenvalores de A . Esto se consigue con un proceso llamado *precondicionamiento*, que consiste en cambiar x a $\hat{x}$ tal que $\hat{x} = Cx$, con C no singular. De esta forma, el problema (1) se vuelve

$$\phi(\hat{x}) = \frac{1}{2}\hat{x}^\top \left(C^{-\top} A C^{-1}\right)\hat{x} - \left(C^{-\top} b\right)^\top \hat{x} \tag{9}$$

Entonces, aplicamos el gradiente conjugado a $\hat{x}$ y resulta que el algoritmo depende de $M = C^\top C$. El algoritmo queda:

$$\alpha_k = \frac{r_k^\top y_k}{p_k^\top A p_k} \tag{10}$$

$$M y_{k+1} = r_{k+1} \tag{11}$$

$$\beta_k = \frac{r_{k+1}^\top y_{k+1}}{r_k^\top y_k} \tag{12}$$

$$p_{k+1} = -y_{k+1} + \beta_{k+1} p_k \tag{13}$$

El caso $M = \text{diag}(A)$ se conoce como precondicionador de Jacobi.

2. Pseudocódigo

Algoritmo 1: Método de la Iteración en Subespacios para los p eigenvalores más grandes

Entrada: $A, \Phi^0, tol, maxIter$
Salida: Φ, Λ

$\tilde{\Phi}^0 \leftarrow A\Phi^0$;
$\Phi^1 \leftarrow \text{ortogonalizar}(\tilde{\Phi}^0)$;
$K^1 \leftarrow {\Phi^1}^\top A\Phi^1$;
$i \leftarrow 1$;
mientras $i < maxIter$ ***y*** $\frac{\left\|\Phi^i_{:,1} - \Phi^{i-1}_{:,1}\right\|}{\left\|\Phi^i_{:,1}\right\|} > tol$ **hacer**
  $Z^i \leftarrow \text{metodoJacobi}(K^i)$;
  $\tilde{\Phi}^i \leftarrow A\Phi^i Z^i$;
  $\Phi^{i+1} \leftarrow \text{ortogonalizar}(\tilde{\Phi}^i)$;
  $K^{i+1} \leftarrow {\Phi^{i+1}}^\top A\Phi^{i+1}$;
  $i \leftarrow i + 1$;
fin
$\Phi \leftarrow \Phi^i$;
$\Lambda \leftarrow \text{diag}(K^i)$;

Algoritmo 2: Método de la Iteración en Subespacios para los p eigenvalores más chicos

Entrada: $A, \Phi^0, tol, maxIter$
Salida: Φ, Λ

resolver $A\tilde{\Phi}^0 = \Phi^0$;
$\Phi^1 \leftarrow \text{ortogonalizar}(\tilde{\Phi}^0)$;
$K^1 \leftarrow {\Phi^1}^\top A\Phi^1$;
$i \leftarrow 1$;
mientras $i < maxIter$ ***y*** $\frac{\left\|\Phi^i_{:,1} - \Phi^{i-1}_{:,1}\right\|}{\left\|\Phi^i_{:,1}\right\|} > tol$ **hacer**
  $Z^i \leftarrow \text{metodoJacobi}(K^i)$;
  resolver $A\tilde{\Phi}^i = \Phi^i Z^i$;
  $\Phi^{i+1} \leftarrow \text{ortogonalizar}(\tilde{\Phi}^i)$;
  $K^{i+1} \leftarrow {\Phi^{i+1}}^\top A\Phi^{i+1}$;
  $i \leftarrow i + 1$;
fin
$\Phi \leftarrow \Phi^i$;
$\Lambda \leftarrow \text{diag}(K^i)$;

Algoritmo 3: Método del cociente de Rayleigh

Entrada: $A, v^0, \lambda^0, tol, maxIter$
Salida: v, λ

$i \leftarrow 0$;
mientras $i < maxIter$ ***y*** $\left|\lambda^i - \lambda^{i-1}\right| > tol$ **hacer**
  resolver $(A - \lambda^i \mathbf{I})v^{i+1} = v^i$;
  $\lambda^{i+1} = \lambda^i + \frac{v^{i+1\top} v^i}{v^{i+1\top} v^{i+1}}$;
  $i \leftarrow i + 1$;
fin
$v \leftarrow v^i$;
$\lambda \leftarrow \lambda^i$;

Algoritmo 4: Factorización QR de una matriz

Entrada: A
Salida: Q, R

para $i = 1 : n$ **hacer**
  para $j = 1 : i - 1$ **hacer**
    $R_{i,j} \leftarrow Q_{:,i}^\top A_{:,j}$;
  fin
  $\tilde{q}_i \leftarrow A_{:,i} - \sum_{j=1}^{i-1} R_{j,i} Q_{:,j}$;
  $Q_{:,i} \leftarrow \frac{1}{\|\tilde{q}_i\|_2} \tilde{q}_i$;
  $R_{i,i} \leftarrow \|\tilde{q}_i\|_2$;
fin

Algoritmo 5: Gradiente Conjugado

Entrada: $A, b, x^0, tol, maxIter$
Salida: x

$r^0 \leftarrow Ax^0 - b$;
$p^0 \leftarrow -r^0$;
$i \leftarrow 0$;
mientras $i < maxIter$ ***y*** $\left\|r^i\right\| > tol$ **hacer**
  $\alpha^i \leftarrow -\frac{r^{i\top} r^i}{p^{i\top} A p^i}$;
  $x^{i+1} \leftarrow x^i + \alpha^i p^i$;
  $r^{i+1} \leftarrow Ax^i - b$;
  $\beta^{i+1} \leftarrow \frac{r^{i+1\top} r^{i+1}}{r^{i\top} r^i}$;
  $p^{i+1} = -r^{i+1} + \beta^{i+1} p^i$;
  $i \leftarrow i + 1$;
fin
$x \leftarrow x^i$

Algoritmo 6: Gradiente Conjugado Precondicionado

Entrada: $A, b, x^0, M, tol, maxIter$
Salida: x

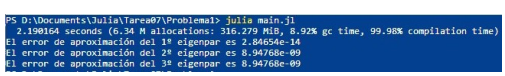
$r^0 \leftarrow Ax^0 - b$;
resolver $My^0 = r^0$;
$p^0 \leftarrow -r^0$;
$i \leftarrow 0$;
mientras $i < maxIter$ ***y*** $\left\|r^i\right\| > tol$ **hacer**
  $\alpha^i \leftarrow -\frac{{r^i}^\top y^i}{{p^i}^\top A p^i}$;
  $x^{i+1} \leftarrow x^i + \alpha^i p^i$;
  $r^{i+1} \leftarrow Ax^i - b$;
  resolver $My^{i+1} = r^{i+1}$;
  $\beta^{i+1} \leftarrow \frac{{r^{i+1}}^\top y^{i+1}}{{r^i}^\top y^i}$;
  $p^{i+1} = -y^{i+1} + \beta^{i+1} p^i$;
  $i \leftarrow i + 1$;
fin
$x \leftarrow x^i$

3. Algoritmos

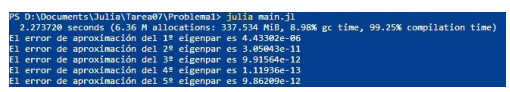
Para la ejecución de los algoritmos, debemos ejecutar

```
julia main.jl
```

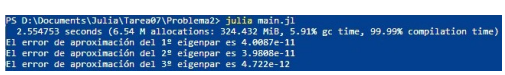
desde la carpeta del problema. A continuación se muestra la ejecución de los algoritmos para los problemas de la tarea.



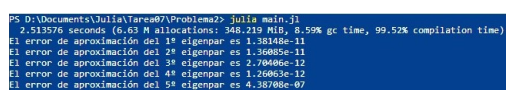
(a) Ejecución del método de Iteración en Subespacios de E_3 para los 3 eigenpares más grandes



(b) Ejecución del método de Iteración en Subespacios de E_{50} para los 5 eigenpares más grandes



(a) Ejecución del método de Iteración en Subespacios de E_3 para los 3 eigenpares más chicos



(b) Ejecución del método de Iteración en Subespacios de E_{50} para los 5 eigenpares más chicos

```
PS D:\Documents\Julia\Tarea07\Problema3> julia main.jl
  0.738736 seconds (2.36 M allocations: 121.173 MiB, 3.80% gc time, 99.97% compilation time)
El error de aproximación del 1º eigenpar es 5.88785e-17
El error de aproximación del 2º eigenpar es 2.77556e-17
El error de aproximación del 3º eigenpar es 1.77982e-15
```

(a) Ejecución del método del Cociente de Rayleigh de E_3 para todos los eigenpares

```
PS D:\Documents\Julia\Tarea07\Problema3> julia main.jl
  1.086005 seconds (2.98 M allocations: 507.947 MiB, 18.31% gc time, 69.61% compilation time)
El error de aproximación del 1º eigenpar es 1.78651e-15
El error de aproximación del 2º eigenpar es 1.59801e-16
El error de aproximación del 3º eigenpar es 2.18361e-16
El error de aproximación del 4º eigenpar es 1.42118e-14
El error de aproximación del 5º eigenpar es 2.84224e-14
El error de aproximación del 6º eigenpar es 7.10688e-15
El error de aproximación del 7º eigenpar es 1.8573e-16
El error de aproximación del 8º eigenpar es 1.42123e-14
El error de aproximación del 9º eigenpar es 4.26334e-14
El error de aproximación del 10º eigenpar es 2.03974e-16
El error de aproximación del 11º eigenpar es 2.84231e-14
El error de aproximación del 12º eigenpar es 4.26523e-16
El error de aproximación del 13º eigenpar es 2.84311e-14
El error de aproximación del 14º eigenpar es 8.52659e-14
El error de aproximación del 15º eigenpar es 2.84244e-14
El error de aproximación del 16º eigenpar es 2.84268e-14
El error de aproximación del 17º eigenpar es 2.84245e-14
El error de aproximación del 18º eigenpar es 8.09234e-16
El error de aproximación del 19º eigenpar es 2.8426e-14
El error de aproximación del 20º eigenpar es 2.84271e-14
El error de aproximación del 21º eigenpar es 5.68449e-14
El error de aproximación del 22º eigenpar es 8.52683e-14
El error de aproximación del 23º eigenpar es 2.8425e-14
El error de aproximación del 24º eigenpar es 5.68445e-14
El error de aproximación del 25º eigenpar es 4.44022e-16
El error de aproximación del 26º eigenpar es 5.63214e-16
El error de aproximación del 27º eigenpar es 1.13687e-13
El error de aproximación del 28º eigenpar es 5.68486e-14
El error de aproximación del 29º eigenpar es 5.68466e-14
El error de aproximación del 30º eigenpar es 1.13688e-13
El error de aproximación del 31º eigenpar es 1.70531e-13
El error de aproximación del 32º eigenpar es 1.00421e-15
El error de aproximación del 33º eigenpar es 5.68447e-14
El error de aproximación del 34º eigenpar es 5.68441e-14
El error de aproximación del 35º eigenpar es 5.82156e-16
El error de aproximación del 36º eigenpar es 5.68551e-14
El error de aproximación del 37º eigenpar es 1.70531e-13
El error de aproximación del 38º eigenpar es 1.13691e-13
El error de aproximación del 39º eigenpar es 5.68479e-14
El error de aproximación del 40º eigenpar es 5.68466e-14
El error de aproximación del 41º eigenpar es 5.68466e-14
El error de aproximación del 42º eigenpar es 1.13688e-13
El error de aproximación del 43º eigenpar es 5.68462e-14
El error de aproximación del 44º eigenpar es 4.91838e-16
El error de aproximación del 45º eigenpar es 1.13688e-13
El error de aproximación del 46º eigenpar es 1.03642e-15
El error de aproximación del 47º eigenpar es 4.29237e-16
El error de aproximación del 48º eigenpar es 9.25406e-16
El error de aproximación del 49º eigenpar es 5.68445e-14
El error de aproximación del 50º eigenpar es 3.35036e-16
```

(b) Ejecución del método del cociente de Rayleigh de E_{50} para todos los eigenpares

```
PS D:\Documents\Julia\Tarea07\Problema4> julia main.jl
  0.244122 seconds (670.08 k allocations: 35.108 MiB, 2.75% gc time, 99.98% compilation time)
El error relativo de aproximación es 1.7729e-15
```

(a) Solución de $A_3x = b_3$ por factorización QR

```
PS D:\Documents\Julia\Tarea07\Problema4> julia main.jl
  0.237974 seconds (670.45 k allocations: 35.554 MiB, 2.89% gc time, 99.08% compilation time)
El error relativo de aproximación es 1.27004e-19
```

(b) Solución de $A_{125}x = b_{125}$ por factorización QR

```
PS D:\Documents\Julia\Tarea07\Problema5> julia main.jl
  0.151958 seconds (577.91 k allocations: 30.211 MiB, 5.33% gc time, 99.97% compilation time)
El error relativo de aproximación es 8.82405e-12
```

(a) Solución de $A_3x = b_3$ por Gradiente Conjugado

```
PS D:\Documents\Julia\Tarea07\Problema5> julia main.jl
  0.137389 seconds (577.91 k allocations: 30.216 MiB, 5.03% gc time, 99.25% compilation time)
El error relativo de aproximación es 2.2507e-11
```

(b) Solución de $A_{125}x = b_{125}$ por Gradiente Conjugado

```
PS D:\Documents\Julia\Tarea07\Problema6> julia main.jl
  0.181230 seconds (594.13 k allocations: 31.031 MiB, 4.27% gc time, 99.96% compilation time)
El error relativo de aproximación es 9.52616e-11
```

(a) Solución de $A_3x = b_3$ por Gradiente Conjugado con Precondicionamiento de Jacobi

```
PS D:\Documents\Julia\Tarea07\Problema6> julia main.jl
  0.164877 seconds (594.13 k allocations: 31.038 MiB, 4.07% gc time, 99.45% compilation time)
El error relativo de aproximación es 1.53242e-11
```

(b) Solución de $A_{125}x = b_{125}$ por Gradiente Conjugado con Precondicionamiento de Jacobi

4. Resultados

Se probaron los métodos de Iteración en Subespacios y del cociente de Rayleigh para las matrices proporcionadas E_3 =`Eigen_3x3.txt` y E_{50} =`Eigen_50x50.txt`. En el método de Iteración en Subespacios se consideraron $p = 3$ para E_3 y $p = 5$ para E_{50}. La tolerancia para los métodos de Iteración en Subespacios es de 10^{-8} y 100000 iteraciones, mientras que para el cociente de Rayleigh es de 10^{-10} y 100 iteraciones.

Algoritmo	máx $\|E_3 v - \lambda\|$	**Tiempo**	máx $\|E_{50} v - \lambda\|$	**Tiempo**
1	8.94768×10^{-9}	2.190164	4.43302×10^{-6}	2.27372
2	4.0087×10^{-11}	2.554753	4.38708×10^{-7}	2.513576
3	1.77982×10^{-15}	0.738736	1.70531×10^{-13}	1.086005

Tabla 1: Resultados para los algoritmos para encontrar eigenpares

De forma similar, se probaron los métodos de factorización QR y gradiente conjugado, con y sin precondicionamiento, para los sistemas $A_3 x = b_3$ y $A_{125} x = b_{125}$, donde A_3 =`M_sys_3x3.txt`, b_3 =`V_sys_3x1.txt`, A_{125} =`M_sys_125x125.txt` y b_{125} =`V_sys_125x1.txt`. Para los métodos de Gradiente Conjugado, se consideran $3m$ iteraciones y tolerancia de 10^{-10}.

Algoritmo	$A_3 x = b_3$	**Tiempo**	$A_{125} x = b_{125}$	**Tiempo**
4	1.7729×10^{-15}	0.244122	1.2700×10^{-19}	0.237974
5	8.8241×10^{-12}	0.151958	2.2507×10^{-11}	0.137389
6	9.5262×10^{-11}	0.18123	1.5324×10^{-11}	0.164877

Tabla 2: Resultados para los algoritmos para resolver sistemas lineales

5. Conclusiones

Notamos que para obtener los eigenpares de A_i el mejor método es el del cociente de Rayleigh en el caso de tener una aproximación de los eigenpares, en otro caso, los métodos de iteración en subespacios son efectivos para obtener algunos eigenpares.
Para la solución de sistemas lineales, el método de gradiente conjugado es bueno, especialmente si n es grande, pues no requiere mucha memoria, lo que se refleja en el tiempo de ejecución. La desventaja es que este método solamente funciona para A simétrica y positiva definida.

Referencias

[1] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer, Jul. 2006.