Álgebra Lineal Computacional

2do Cuatrimestre 2025

Laboratorio N° 5: Descomposición QR.

En este laboratorio generaremos funciones para calcular la factorización QR de una matriz $A \in R^{n\times n}$. La factorización QR se compone de una matriz ortogonal Q tal que $Q^tQ = I$, la matriz identidad y R es una matriz triangular superior. De esta forma, obtenemos una forma de calcular la solución del problema $Ax = b$ como $Rx = Q^tb$. El desafío entonces es encontrar la matriz Q tal que $Q^tA = R$. En este taller, nos enfocaremos en dos de ellas, aunque existen otras en el mercado. En todos los casos, el problema se trata sobre encontrar un conjunto de vectores $\{q_i\}_{i=1}^n$ tal que $q_i^tq_j = \delta_{ij}$ ($\delta_{ij} = 1$ si $i = j$ y 0 en otro caso).

Ejercicio 1 (Gram-Schmidt)**.** Uno de los métodos más intuitivos es el de Gram-Schmidt. Se basa en ortonormalizar un conjunto de vectores de forma iterativa hasta tener. Es útil para este algoritmo pensar a la matriz $A \in \mathbb{R}^{n\times n}$ en términos de sus columnas $a_1, \ldots, a_n \in \mathbb{R}^n$. Notemos a las columnas de Q como $q_1, \ldots, q_n$ y a los elementos de R como r_{ij}. Entonces, empezando con $q_1 = a_1/||a_1||_2$ y $r_{11} = ||a_1||_2$ podemos calcular a los elementos de Q y R a través del siguiente algoritmo:

Algoritmo 1: Factorización QR por Gram-Schmidt

Input: Matriz $A \in \mathbb{R}^{n\times n}$
Output: Matrices Q con columnas q_j, y R con elementos r_{ij} tal que $A = QR$.

$q_1 \leftarrow a_1/||a_1||_2$;
$r_{11} = ||a_1||_2$
for $j = 2$ **to** n **do**
  $\tilde{q}_j \leftarrow a_j$
  for $k = 1$ **to** $j - 1$ **do**
    $r_{kj} \leftarrow q_k^t\tilde{q}_j$
    $\tilde{q}_j \leftarrow \tilde{q}_j - r_{kj}q_k$
  end
  $r_{jj} \leftarrow ||\tilde{q}_j||_2$
  $q_j \leftarrow \tilde{q}_j/r_{jj}$
end

a) Implementen el algoritmo de Gram Schmidt, y cuenten la cantidad de operaciones necesarias para el calculo de Q y R. Diferencien sumas, restas, multiplicaciones y divisiones del resto de las operaciones. Comparen con la factorización LU para distintos n. ¿Cuánto más costoso es QR?

b) Comparen la factorización LU con la QR obtenida por el algoritmo descrito anteriormente. Para esto consideren el siguiente esquema:

   1. Genere una matriz al azar $A \in \mathbb{R}^{n\times n}$ tal que $|A_{ij}| \leq 1$ $\forall i, j$.

2. Calcule su factorización LU empleando el método generado en el L04. En caso de no ser posible, descarte la matriz y vuelva a 1.
3. Calcule su factoización QR usando el algoritmo de GS.
4. Calcule la inversa de la matriz A con cada método, resolviendo el problema $LUA^{-1}_{:,i} = e_i$ y $QRA^{-1}_{:,i} = e_i$, donde $A^{-1}_{:,i}$ es la columna i-esima de A^{-1} y e_i es el i-esimo canónico. Llamamos a la matriz obtenida mediante LU A^{-1}_{LU} y a la obtenida mediante QR, A^{-1}_{QR}
5. Calcule el producto $E_{LU} = AA^{-1}_{LU}$ y $E_{QR} = AA^{-1}_{QR}$. Usando la función normaExacta desarrollada en el L03, compute $\epsilon_{LU} = ||E_{LU} - I_n||_\infty$ y $\epsilon_{QR} = ||E_{QR} - I_n||_\infty$, con I_n la matriz identidad de $n \times n$.

Para $n = 2, 5, 10, 20, 50, 100$, aplique el esquema de testeo propuesto a 100 matrices generadas al azar de $n \times n$ y calcule valor medio de ϵ_{LU} y ϵ_{QR} en función de n. Grafique para cada caso un histograma mostrando la distribución de los errores.

c) El algoritmo de Gram Schmidt descrito anteriormente falla si las columnas de A son linealmente dependientes. Sin embargo, esto puede resolverse guardando en Q únicamente aquellos vectores que tengan $||\tilde{q}_j||_2 > 0$. El número de vectores en cuestión, r es igual al rango de la matriz A. Modifique el algoritmo anterior de forma tal que construya $\hat{Q} \in \mathbb{R}^{nxr}$ y $\hat{R} \in \mathbb{R}^{rxn}$. $\tilde{Q}$ debe mantener únicamente aquellos vectores $\tilde{q}_j$ con norma mayor a una tolerancia `tol=1e-12` (parámetro default de la función) y la matriz $\hat{R}$ debe retener únicamente las primeras r filas de R.

Ejercicio 2 (Householder)**.** A continuación, se describe el algoritmo para la construcción de las matrices Q y R empleando reflexiones de Householder, para matrices $A \in \mathbb{R}^{m\times n}$:

Algoritmo 2: Factorización QR por reflexiones de Householder

Input: Matriz $A \in \mathbb{R}^{m\times n}$ con $m \geq n$, tolerancia tol
Output: Matriz ortogonal $Q \in \mathbb{R}^{m\times m}$ y $R \in \mathbb{R}^{m\times n}$ triangular superior, tal que $A = QR$.

$R \leftarrow A$;
$Q \leftarrow I_m$
for $k = 1$ **to** n **do**
  $x \leftarrow R_{k:m,\,k}$
  $\alpha \leftarrow -\operatorname{sign}(x_1)\,\|x\|_2$
  $u \leftarrow x - \alpha e_1^{(m-k+1)}$
  if $||u||_2 > tol$ **then**
    $u \leftarrow u/\|u\|_2$
    $H_k \leftarrow I_{m-k+1} - 2\,uu^T$
    $\tilde{H}_k \leftarrow \begin{pmatrix} I_{k-1} & 0 \\ 0 & H_k \end{pmatrix} \in \mathbb{R}^{m\times m}$
    $R \leftarrow \tilde{H}_k R$;
    $Q \leftarrow Q\tilde{H}_k^T$
  end
end

En el Algoritmo 2, la matriz I_m corresponde a la matriz identidad de $m \times m$, el vector e_1^l corresponde al primer canónico de $\mathbb{R}^l$ y el vector $R_{k:m,k}$ corresponde a los elementos k a m de la columna k de la matriz R (y por lo tanto, $\in \mathbb{R}^{m-k}$)

a) Implemente la factorización QR empleando el método de reflexiones de Householder descrito en el Algoritmo 2.

b) Repita la comparación realizada en el punto 1b), incluyendo ahora LU y QR mediante Gram-Schmidt y Householder. Incorpore al análisis la comparación entre Q_{GS} (calculada con Gram Schmidt) y Q_{RH} (calculada con reflectores de Householder). Para esto, compute $\epsilon_{GS} = ||Q_{GS}^t Q_{GS} - I||_\infty$ y $\epsilon_{RH} = ||Q_{RH}^t Q_{RH} - I||_\infty$, y repita el análisis hecho para los ϵ del punto 1b).

c) Implemente una función *wrapper* `calculaQR(A,tol,metodo='RH')` que permita calcular la factorización QR de una matriz A empleando la versión del punto 1c) de Gram-Schmidt al indicar `metodo='GS'` o las reflexiones de Householder al indicar `metodo='RH'`.

Ejercicios extra

Ejercicio 3 (Núcleo de una transformación lineal). Sea $R \in \mathbb{R}^{m\times n}, m \geq n$ una matriz triangular superior con r filas distintas de cero y tal que el primer elemento distinto de cero de cada fila se encuentra en la diagonal. Podemos entonces escribir a la matriz R como

$$R = \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}$$

con $R_1 \in \mathbb{R}^{r\times r}$ y $R_2 \in \mathbb{R}^{r\times n-r}$, y con $m-r$ filas de ceros en la parte inferior. Los elementos del núcleo están generados entonces por las columnas de la matriz (*¡pruebenlo!*):

$$x = \begin{pmatrix} -R_1^{-1}R_2 \\ I_{n-r} \end{pmatrix}$$

Implementen una función `nucleo(A,tol=1e-12,metodo=['RH','LU'])` que reciba una matriz $A \in \mathbb{R}^{m\times n}$, calcule su factorización QR usando reflectores de Householder o su factorización LU usando el método del L04. Luego, la función debe identificar como 0 a los elementos de R (o U) menores a `tol`, y calcular los generadores del núcleo de la matriz R (o U) y por lo tanto de A si las filas de R (o U) se anulan a partir de la fila r (*¿por qué los núcleos de R, U y A son los mismos?*).

Ejercicio 4 (Imagen de una transformación lineal). En relación directa con el ejercicio anterior, se puede obtener un conjunto de generadores de la imagen de A fácilmente cuando R (o U) tiene filas iguales a cero a partir de la fila r. Implementen una función `imagen(A,tol=1e-12,metodo=['RH','LU'])` que reciba una matriz $A \in \mathbb{R}^{m\times n}$, calcule su factorización QR usando reflectores de Householder (o su factorización LU usando el método del L04). Luego, la función debe identificar como 0 a los elementos de R (o U) menores a `tol`, y calcular los generadores del núcleo de la matriz R (o U) si las filas de R (o U) se anulan a partir de la fila r.

Módulo ALC

Para el módulo ALC, deben programar las siguientes funciones. Tests para las mismas acompañan en el archivo `L05.py`.

```python
def QR_con_GS(A,tol=1e-12,retorna_nops=False):
    """
    A una matriz de n x n
    tol la tolerancia con la que se filtran elementos nulos en R
    retorna_nops permite (opcionalmente) retornar el numero de operaciones realizado
    retorna matrices Q y R calculadas con Gram Schmidt (y como tercer argumento opcional, el numero de operaciones).
    Si la matriz A no es de n x n, debe retornar None
    """

def QR_con_HH(A,tol=1e-12):
    """
    A una matriz de m x n (m>=n)
    tol la tolerancia con la que se filtran elementos nulos en R
    retorna matrices Q y R calculadas con reflexiones de Householder
    Si la matriz A no cumple m>=n, debe retornar None
    """

def calculaQR(A,metodo='RH',tol=1e-12):
    """
    A una matriz de n x n
    tol la tolerancia con la que se filtran elementos nulos en R
    metodo = ['RH','GS'] usa reflectores de Householder (RH) o Gram Schmidt (GS) para realizar la factorizacion
    retorna matrices Q y R calculadas con Gram Schmidt (y como tercer argumento opcional, el numero de operaciones)
    Si el metodo no esta entre las opciones, retorna None
    """
```

Nota: no está permitodo el uso de la multiplicación matricial de numpy (`@`, `np.matmul`, etc)