SEGUNDA PRÁCTICA MNT

Rubén Serrano Rodríguez

1.-La factorización QR

Para el código de reflectores y de Gram Schmidt he usado el código dado en las clases de teoría, sin embargo, para el código de GS modificado he usado el código proporcionado en la asignatura del año 2010.

En primer lugar, vamos a comparar los errores producidos en los tres algoritmos frente al número de condición de las matrices generadas. En la *ilustración 1* observamos esta comparación. Lo primero a destacar es que el mejor método (en medida del error) es de los reflectores, pues, el error es casi igual a medida que aumenta el tamaño de la matriz (y por ende el número de condición, pues ya vimos en la anterior práctica que la matriz de Hilbert tiene un número de condición cada vez mayore a medida que aumenta el tamaño de esta) y además es bastante pequeño, del orden de 10^-15. Para el caso del algoritmo de GS modificado, es bastante bueno el error para matrices de un tamaño pequeño, pero mala para matrices grandes. En el último caso, al algoritmo de GS le ocurre lo mismo que ha los reflectores, que el error es casi el mismo a medida que aumenta la dimensión de la matriz, pero en este caso el error es alto, del orden de la unidad, tanto en matrices pequeñas como grandes.

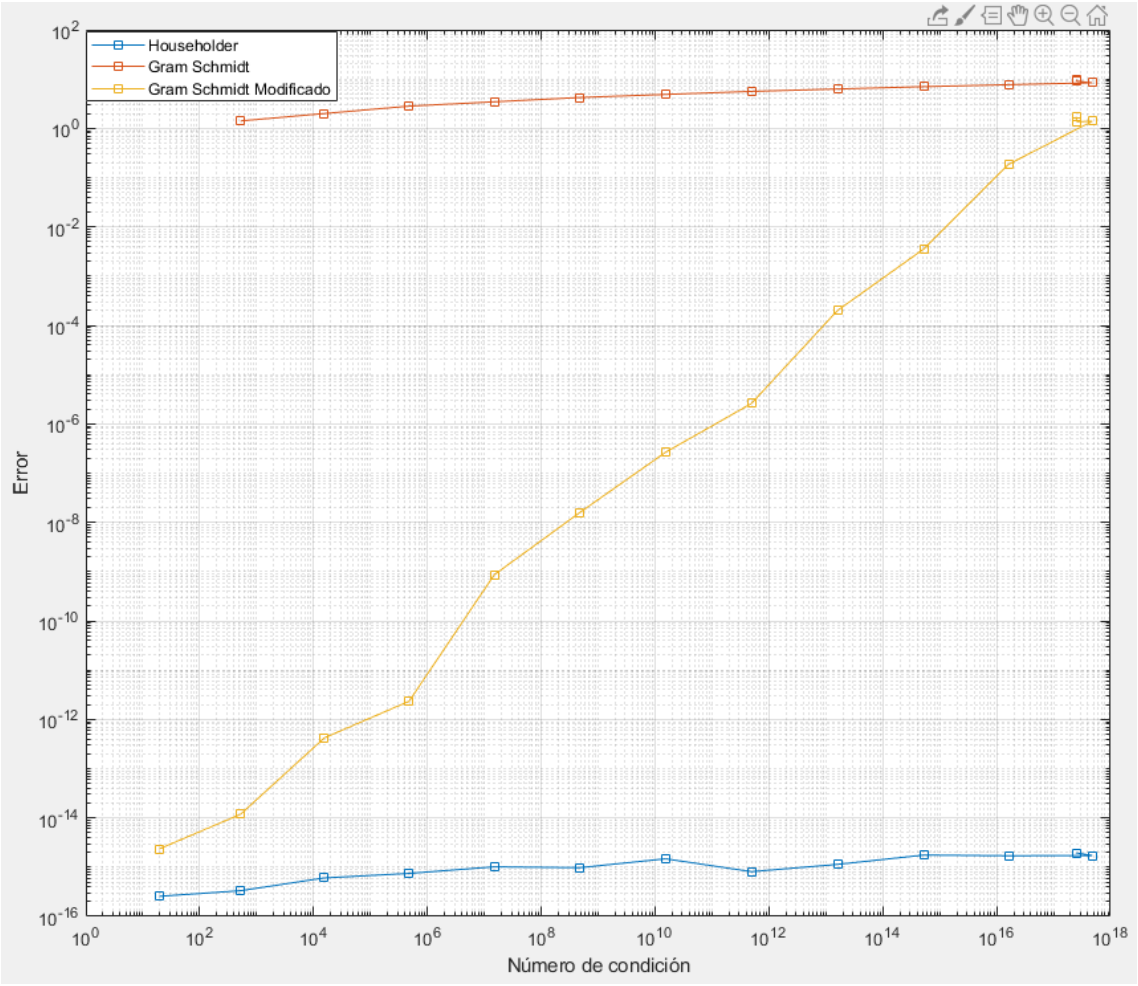

Ilustración 1. Error de las factorizaciones frente al número de condición

Seguidamente realizamos el estudio del coste operativo que conlleva cada factorización frente al tamaño de la matriz a factorizar. Como vemos en la *ilustración 2* ahora el que más atrás se queda es la factorización de los reflectores, pues el tiempo de CPU que tarda en calcularse es mayore que en resto de las factorizaciones. Al contrario que en el caso anterior también, ahora la mejor factorización respecto al tiempo que tarda es la de GS, pues para cualquier tamaño de matriz es menor.

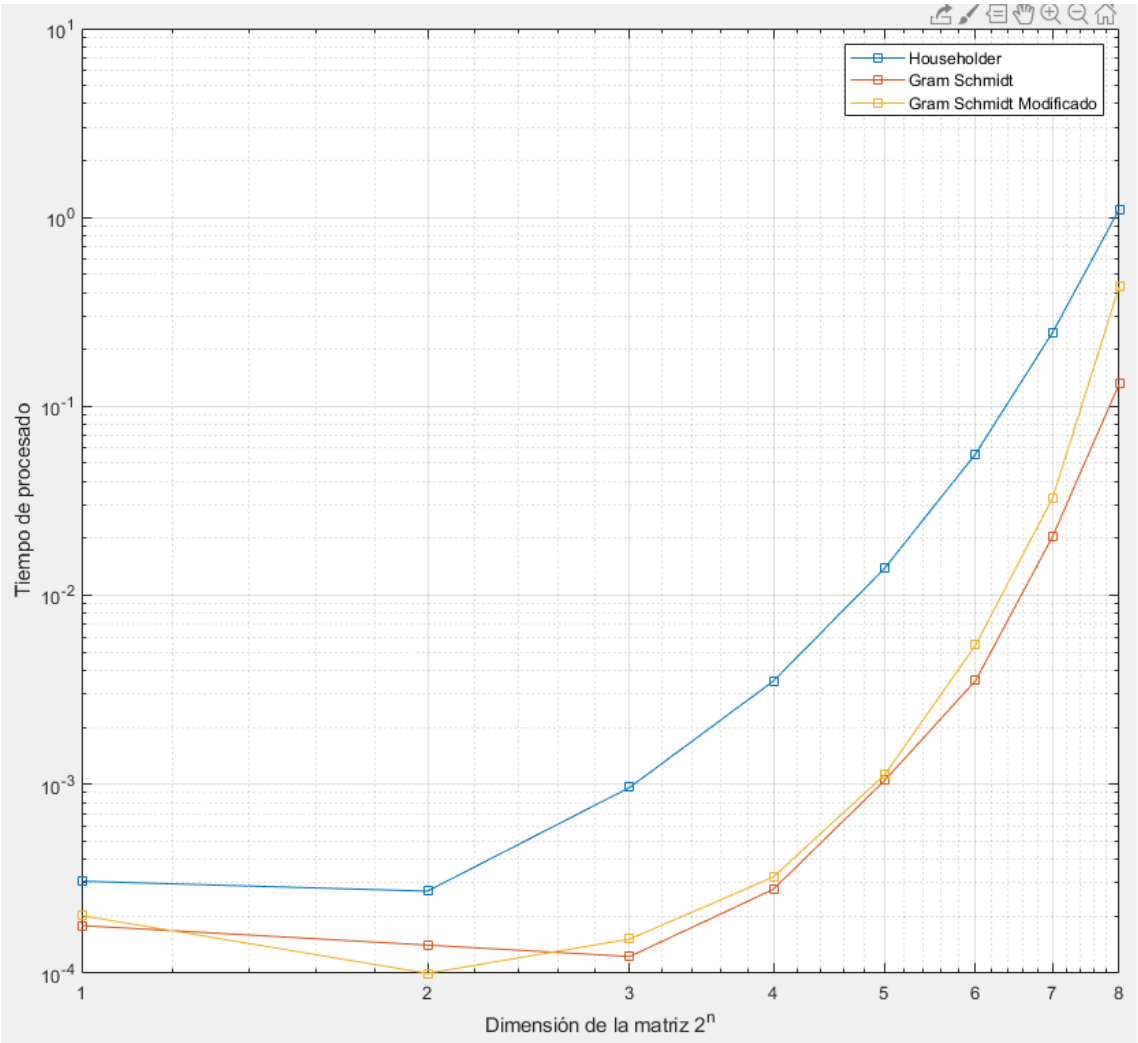

Ilustración 2. Tiempo de CPU en segundos frente a dimensiones de la matriz.

2.-Compresión de imágenes

Primero hemos de precargar una imagen, en mi caso la elegida ha sido “saturn.png”, la cual es una imagen que contiene Matlab. Previa a la imagen de la *ilustración 3* se ha realizado un estudio de la descomposición en valores singulares de la imagen para intentar obtener una imagen semejante, pero con un tamaño menor, es decir, intentar comprimir la imagen. En el caso de nuestra imagen hemos comprobado que para un factor de k = 125, la imagen resultante es bastante similar a la original, por lo que es la elegida para la presentación. Las imágenes anteriores han sido descartadas, pues a simple vista se apreciaba una especia de submuestreo/difuminado de la imagen. Es de utilidad ver la señal diferencia entre ambas imágenes, tercera imagen de la *ilustración 3* donde se ve que es prácticamente negra, es decir, nula. Eso se debe a que la imagen comprimida es bastante parecida a la original.

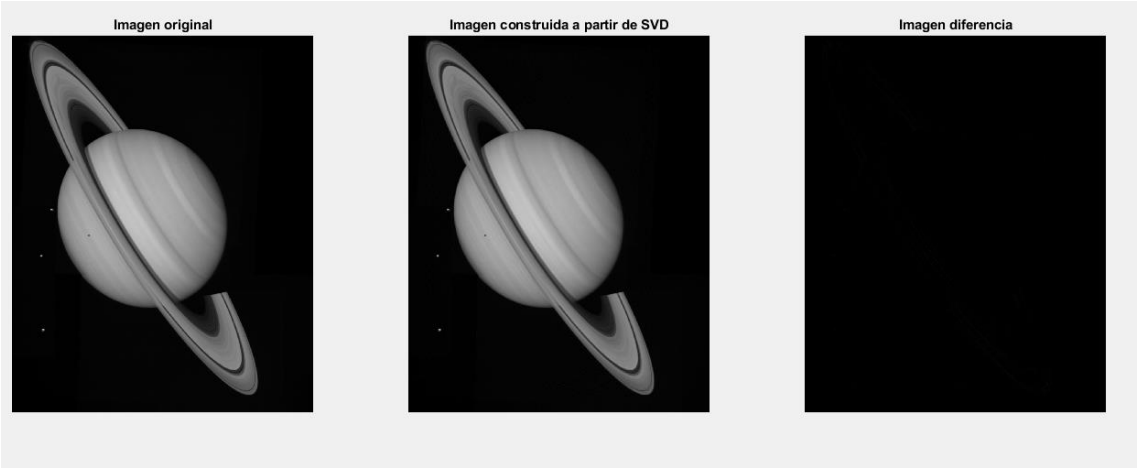

Ilustración 3. Compresión de la imagen saturn.png y representación de la diferencia.

Para observar el efecto de la compresión he decidido poner el ejemplo de la peor compresión posible que he realizado, con un factor k=25, es decir, únicamente las 25 primeras filas de los valores singulares. En este caso como se aprecia en la ilustración 4, la señal diferencia ya no es nula, sino que se ven contornos (y se llega a apreciar incluso el planeta), además de que la señal construida está como difuminada, como decía en el párrafo anterior. Si nos fijamos en la señal diferencia, los contornos que aparecen se deben a los "saltos" bruscos de colores, es decir, a las componentes frecuenciales altas, por lo que nos puede dar a pensar que con factores más altos mejor se aproximan estas componentes frecuenciales.

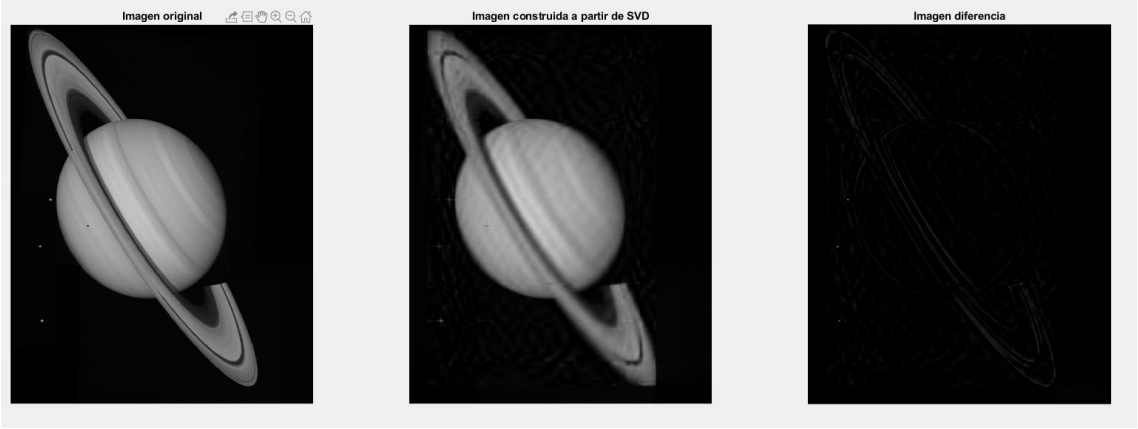

Ilustración 4. Representación con un factor k=25.

Por último, representamos la norma del error producido (la norma de la diferencia entre la imagen original y la imagen comprimida), frente al almacenamiento, es decir, frente al factor de k con la que realizamos la compresión. En la *ilustración 5* observamos que cuanto mayor es el almacenamiento utilizado, es decir, mayor es el factor de k, menor es la medida del error, es decir, la diferencia entre ambas imágenes. Para el factor de k = 125 elegido (punto marcado en la ilustración) el almacenamiento utilizado es de 3200 bytes y norma de 7,17. Podemos pensar que es un error grande, pero esto no es así, ya que los valores de la matriz en escala de grises van del 0 al 255, por lo que el error máximo en es de 7.17, el cual no es tan grande. Así mismo se observa un decrecimiento exponencial del error frente a un crecimiento lineal del almacenamiento, por lo que la convergencia del error es bastante buena, aumentar un poco el tamaño de almacenamiento produce un decrecimiento grande en el error.

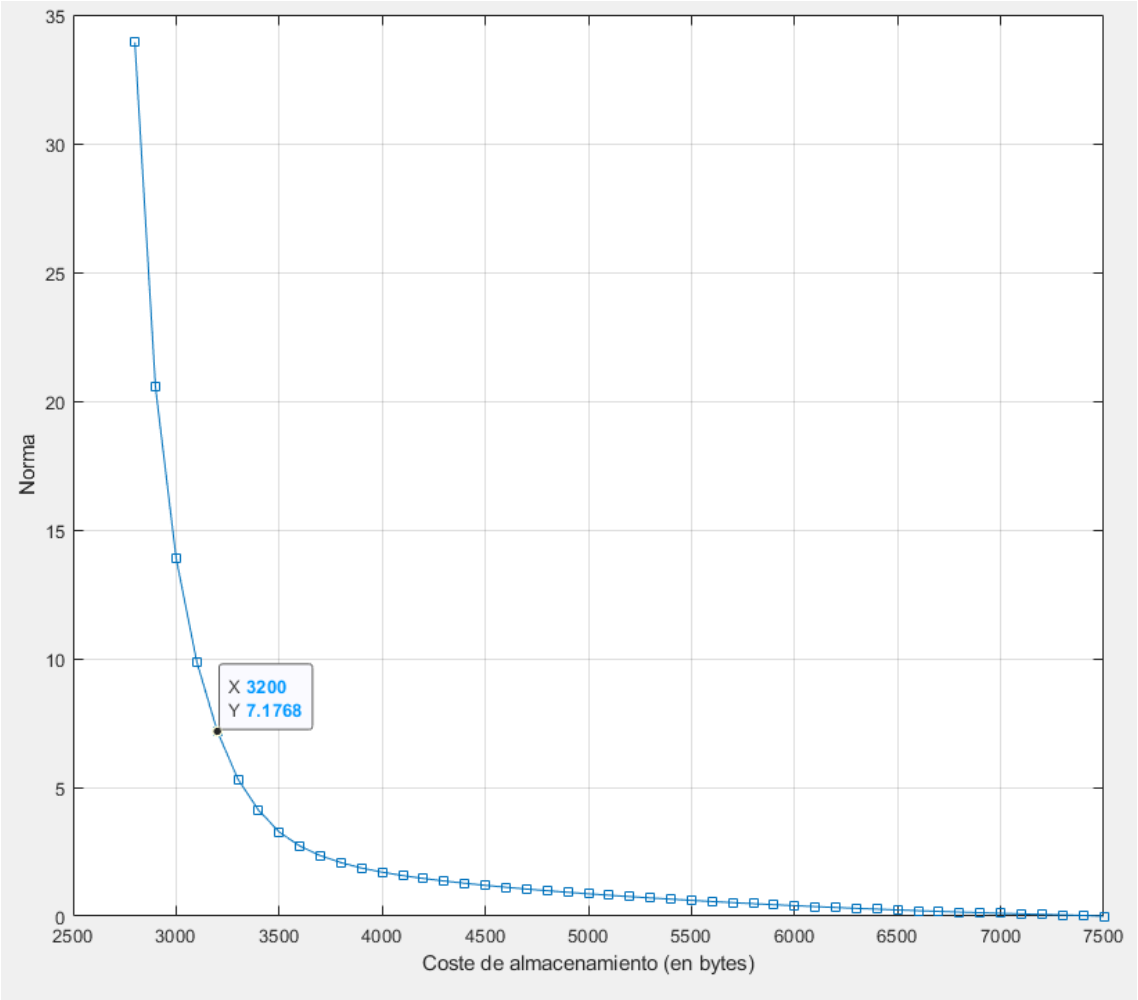

Ilustración 5. Norma del error frente al almacenamiento.

3.-Ejercicios teóricos

1. Sabemos que la matriz A es una matriz semidefinida positiva (valores mayores o iguales que cero). Si realizamos el producto interno de A*A' (con A' = A traspuesta) con un vector cualquiera x vemos que:

   <x,A*A'*x> = x'*A*A'*x = (A'*x)'*A'*x = (norma(A'*x))^2 >= 0

   Es decir, que como es la norma de A traspuesta por el vector x al cuadrado, siempre va a tener valores mayores o iguales a cero A*A'.

2. Como son distintos autovalores y la matriz de autovalores es diag(u1,u2,…,ui,…uj,…,un) entonces el producto interno de ambos vectores es:

   <ui,uj> = (0,0,…,ui,0,...,0) * (0,0,…,uj,0,…,0) = 0 con i distinto de j.

   Como el producto escalar de ambos autovectores son cero, podemos afirmar que son ortogonales.

3. La diferencia es mn – (n+m+1)*k:
   El primer factor es de la matriz original A (m * n)

Para la matriz descompuesta en valores singulares tenemos k valores singulares + n columnas de la matriz U por k filas + m columnas de la matriz V * k filas, por lo que sacando factor común tendremos: (1 + n + m)*k.
Restando ambas dimensiones obtenemos la diferencia de almacenamiento.

4.-Conclusión

En caso de tener que factorizar una matriz, hay que tener cuenta dos aspectos importantes. La medida del error, que no sea muy alta, y el tiempo de cálculo, pues para el primero se observa que el algoritmo de los reflectores es mucho mejor frente al resto, pero a medida que aumenta el tamaño de la matriz, aumenta el tiempo de cálculo (exponencialmente y más rápido que los otros dos), por lo que si tenemos una matriz muy grande puede que sea conveniente usar otros (si la medida del error nos es indiferente).

Hemos apreciado que la descomposición de matrices en valores singulares nos permite obtener una matriz cuyo error es bastante pequeño ahorrando una cantidad de espacio importante como hemos apreciado en el apartado 2 de la práctica, por lo que para la compresión de datos es bastante buena. La única pega que podemos observar en este método es que solo sirve para matrices que sean semidefinidas positivas o definidas positivas. En ámbitos de procesado de imagen es bastante buen método, ya que estas se componen siempre de valores mayores o iguales a 0.