Universidad de Costa Rica
Escuela de Matemáticas
MA-501 Análisis Numérico
Tarea 3

Juan Ignacio Padilla B.
Carné: B55272
Prof. Juan Gabriel Calvo
28 de enero de 2026

Tarea 3

Problema 1.

(Matriz de Vandermonde) Considere el problema de interpolación de Lagrange dado por: Encontrar un polinomio p de grado a lo sumo n cuya gráfica incluya los puntos $\{(x_i, y_i)\}_{i=0}^n$ (el cual sabemos tiene solución única). En este ejercicio plantearemos y resolveremos este problema mediante un sistema de ecuaciones. Escriba

$$p(x) = \sum_{k=0}^{n} a_k x^k$$

El problema de interpolar es equivalente a encontrar los coeficientes $\{a_k\}$ que satisfacen las $n+1$ ecuaciones

$$p(x_i) = \sum_{k=0}^{n} a_k x_i^k = y_i, \quad \forall i = 0, \dots n,$$

el cual se puede escribir matricialmente de la forma $A\boldsymbol{x} = \boldsymbol{b}$, donde

$$\boldsymbol{x} = \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

y A es la matriz de Vandermonde

$$A = \begin{bmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \vdots & & \ddots & \vdots \\ 1 & x_n & \dots & x_n^n \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

En este ejercicio utilizaremos nodos equidistantes en el intervalo $[-1, 1]$, esto es,

$$x_j = -1 + 2\frac{i}{n} \quad (i = 0, \dots, n).$$

a) Demuestre que A es invertible

Solución: Considere

$$A_n = \begin{vmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-2} & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-2} & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-2} & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-2} & x_{n-1}^{n-1} \\ 1 & x_n & x_n^2 & \dots & x_n^{n-2} & x_n^{n-1} \end{vmatrix}$$

Podemos considerar primero A_n como función de x_n. En particular, $A_n(x_n)$ es un polinomio de grado $n-1$, pues expandiendo la última fila se tiene

$$A_n(x_n) = \begin{vmatrix} x_1 & x_1^2 & \dots & x_1^{n-2} & x_1^{n-1} \\ x_2 & x_2^2 & \dots & x_2^{n-2} & x_2^{n-1} \\ x_3 & x_3^2 & \dots & x_3^{n-2} & x_3^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-2} & x_{n-1}^{n-1} \end{vmatrix} + x_n \begin{vmatrix} 1 & x_1^2 & \dots & x_1^{n-2} & x_1^{n-1} \\ 1 & x_2^2 & \dots & x_2^{n-2} & x_2^{n-1} \\ 1 & x_3^2 & \dots & x_3^{n-2} & x_3^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1}^2 & \dots & x_{n-1}^{n-2} & x_{n-1}^{n-1} \end{vmatrix} + \dots$$

$$+ x_n^{n-1} \begin{vmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-2} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-2} \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-2} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-2} \end{vmatrix}$$

Observe que si evaluamos $A_n(x)$ en cualquiera de $x_1, x_2, \dots, x_n$ se obtiene que todos los determinantes tendrán una fila repetida, por lo que $A_n(x_i) = 0$ para $i = 1, \dots, n-1$. Esto nos dice que

$$\det(A) = A_{n-1}(x_n - x_1)(x_n - x_2) \cdots (x_n - x_{n-1})$$

Pues por la expansión anterior, el coeficiente principal es A_{n-1}. Podemos entonces repetir este argumento, considerando el polinomio $A_j(x_j)$ sucesivamente para $x_{n-1}, \dots, x_1$, para obtener

$$\begin{aligned} \det(A) &= \prod_{1 \le i < n} (x_n - x_i) A_{n-1} \\ &= \prod_{1 \le i < n} (x_n - x_i) \prod_{1 \le i < n-1} (x_{n-1} - x_i) A_{n-2} \\ &= \cdots \\ &= \prod_{1 \le i < j \le n} (x_j - x_i) \neq 0 \end{aligned}$$

Lo cual demuestra que A es invertible

b) Para $n = 40$, construya la matriz A. Calcule el número de condición de A, su rango y su determinante. ¿Es la matriz invertible? Sugerencia: puede utilizar los comandos `vander, rank, det, cond` de `MATLAB`.

Solución: Al generar la matriz A para los nodos equidistantes x_j, con $n = 40$, se obtiene numéricamente

- **Número de condición:** $3{,}4987 \times 10^{17}$
- **Rango:** 36
- **Determinante:** $-3{,}6836 \times 10^{-243}$

Numéricamente, la matriz de Vandermonde en este caso no será invertible.

c) Para resolver el sistema de ecuaciones $A\boldsymbol{x} = \boldsymbol{b}$, basta ejecutar el comando $\tilde{\boldsymbol{x}} =$ `A \b`. Considere $f(x) = (1 + 25x^2)^{-1}$. Estime $||f(\texttt{xx}) - p(\texttt{xx})||_\infty$ donde xx = `linspace(-1,1,1e3)`

Solución: Se consideró el polinomio con coeficientes $a_i = \{(A^{-1})b\}_{i+1} (i = 0, \ldots, 40)$, el cual se evaluó numéricamente. Al calcular el error se obtuvo

$$||f(\texttt{xx}) - p(\texttt{xx})||_\infty \approx 1{,}9270 \times 10^5$$

El cual es enorme. Resolver el problema de interpolación por medio de "fuerza bruta" va a resultar en pésimos resultados para matrices con número de condición alto.

d) Calcule la descomposición en valores singulares de A mediante el comando

$$\texttt{[U,S,V]} = \texttt{svd}(A).$$

De esta manera,

$$\tilde{x} = VS^{-1}U^T b.$$

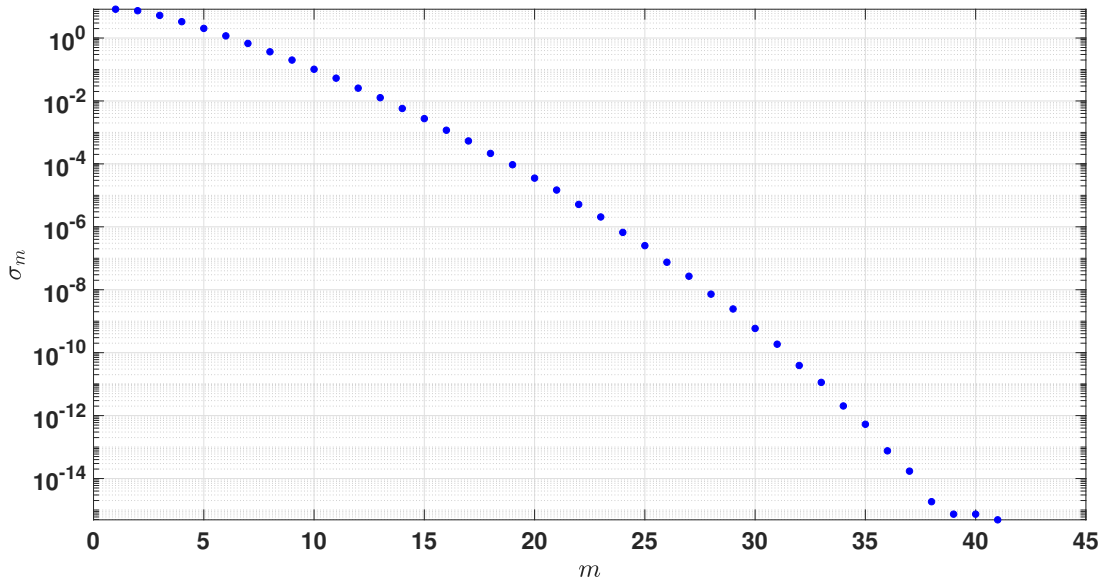
Estime nuevamente $||f(\texttt{xx}) - p(\texttt{xx})||_\infty$.¿Existe alguna mejora en la aproximación de la solución?. Plotee los valores singulares de A en una escala semilogarítmica. Comente el comportamiento. ¿Cuál es el rango numérico de la matriz? Sugerencia: la función `diag` puede ser de utilidad.

Solución: Esta vez se obtiene que

$$||f(\texttt{xx}) - p(\texttt{xx})||_\infty \approx 7{,}2127 \times 10^3$$

El cual es mejor, pero sigue siendo un error enorme. Los valores singulares de A se presentan en la gráfica 1.

Gráfica 1



Se puede ver que los últimos valores se acercan a la precisión de la máquina. Además, el rango numérico de S sigue siendo

Rango: 36.

e) Considere una matriz con descomposición en valores singulares $A = USV^T$. La *pseudoinversa* de A se define como

$$A^+ = VS^+U^T$$

donde

$$S^+ = \begin{bmatrix} 1/\sigma_1 & & & \\ & 1/\sigma_2 & & \\ & & \ddots & \\ & & & 1/\sigma_p \end{bmatrix}$$

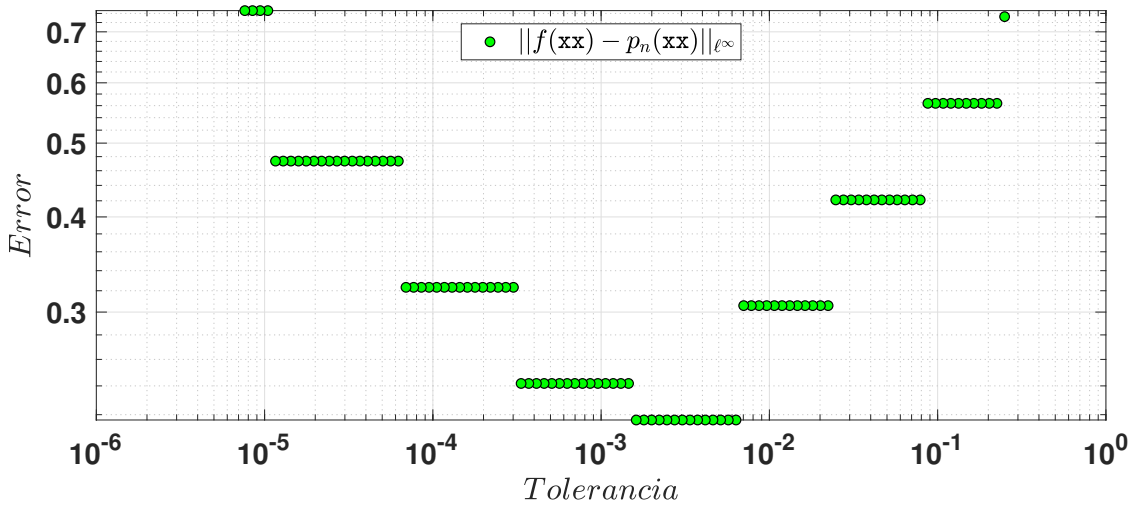
es la matriz diagonal obtenida de invertir los valores singulares positivos de S. Defina un vector de tolerancias `tol = 10.^(-linspace(2,17))`Para cada entrada `j` de `tol`, defina S_j como la matriz obtenida a partir de S, reemplazando los valores singulares menores que `tol(j)`$*\sigma_1$ por cero. Calcule

$$\texttt{x\_svd} = (VS_j^+U^T)b$$

y calcule la norma $||f(\texttt{xx}) - p(\texttt{xx})||_\infty$. Note que estamos aproximando la pseudoinversa con solamente algunos valores singulares de A. Grafique el error en la función del vector de tolerancias. Comente sus resultados y explique el comportamiento del gráfico.

Solución: Al aplicar la aproximación solamente usando algunos valores singulares de A, se obtiene el error en función de la tolerancia elegida, y se resume en la Gráfica 2

Gráfica 2



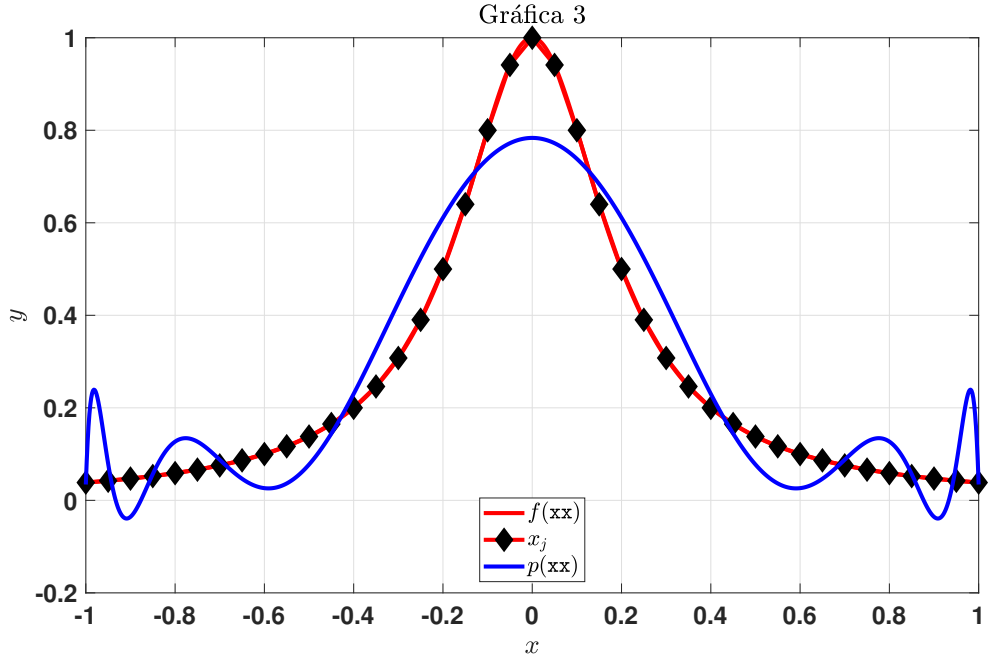
Se puede observar que al disminuir la tolerancia, el error **crece rápidamente**. La razón de esto es la siguiente: al minimizar la tolerancia, se escogen más valores singulares, por lo que la pseudoinversa de la matriz A se acerca numéricamente a la inversa de A. Sin embargo, como vimos, al calcular A^{-1} numéricamente para resolver sistemas, los resultados tienen errores muy grandes. Por lo tanto, al minimizar la tolerancia demasiado, el error va a crecer. Entonces, en este caso, es mejor aproximar la solución utilizando menos valores singulares. **Nota:** Los menores errores en la Gráfica 2 valen aproximadamente 0,01.

f) Para un valor de tolerancia que minimice el error del inciso anterior, plotee en un mismo gráfico $f(\texttt{xx})$, $p(\texttt{xx})$ y los nodos de interpolación. ¿Qué tan buena es la aproximación?

Solución: El valor de tolerancia que minimiza el error del inciso anterior es $t \approx 0{,}0025$ (esto se calculó en el script adjunto). Al graficar f y $p(\texttt{xx})$ con 1000 nodos equidistantes se obtiene la Gráfica 3

Se observa que la aproximación no es muy buena, además, se presentan oscilaciones cerca de los extremos, por lo cual se concluye que este método presenta el fenómeno de Runge.

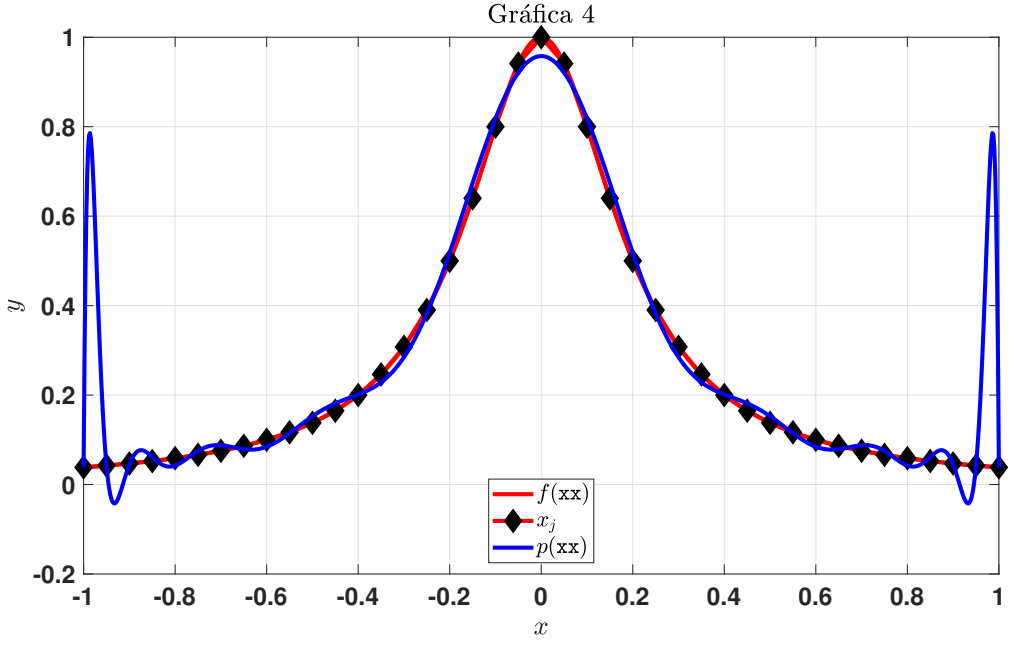
Gráfica 3



g) Ahora utilice `tol = 1e-5`. Plotee en un mismo gráfico $f(\texttt{xx})$, $p(\texttt{xx})$, y los nodos de interpolación. ¿Se observa el fenómeno de Runge? Explique el porqué.

Solución: Para la tolerancia especificada, se repite el inciso anterior, para obtener la Gráfica 4.

Gráfica 4



Si bien la aproximación parece ser mejor a simple vista, en este caso se presenta un mayor fenómeno de Runge, lo que aumenta el error $||f(\texttt{xx}) - p(\texttt{xx})||_\infty$. La razón de esto es que, una vez más, al disminuir la tolerancia, se escogen más valores singulares de A, y la aproximación final de a_i presenta errores considerables, debido al número alto de condición. Según la norma infinito, esta aproximación es **peor** que la anterior.

Problema 2.

(Compresión de imágenes) En este ejercicio utilizamos la descomposición en valores singulares para comprimir imágenes de una manera muy básica. Una imagen de dimensión $m \times n$ consiste de mn pixeles (la resolución se refiere al número de pixeles utilizados). En una imagen a color, cada pixel contiene tres valores enteros $(r_p, g_p, b_p) \in [0, 255]^3$, que corresponden a los componentes rojo, verde y azul. De esta forma, la imagen es guardad como una matriz de tamaño $m \times n \times 3$.

a) (Leyendo la imagen) Considere la imagen `'photo.bmp'`, suministrada. Para leer la imagen ejecute el comando `RGB=imread('photo.jpg')` (*image read*).¿Cuántos pixeles tiene la imagen guardada en `RGB`?¿Qué tipo de entradas tiene dicha matriz?. Sugerengia: imprima el comando `whos RGB` y comente el resultado.

Solución: La matriz `RBG` se guardó como una matriz $3648 \times 2736 \times 3$, la cual almacena tipo de dato `uint8` lo cual corresponde con números enteros entre 0 y 255.

b) (Convirtiendo la imagen a formato doble) Para poder utilizar la DVS, convierta la matriz `RGB` a formato doble mediante el comando `RGB =im2double(RGB)`.

Solución: Se ejecutó la acción en el script `Ejercicio2.m`.

c) (Comprimiendo la imagen) Calcule la descomposición en valores singulares de `RGB`$= USV^T$ (para cada matriz color rojo, verde y azul). Calcule la aproximación de rango 5 dada por

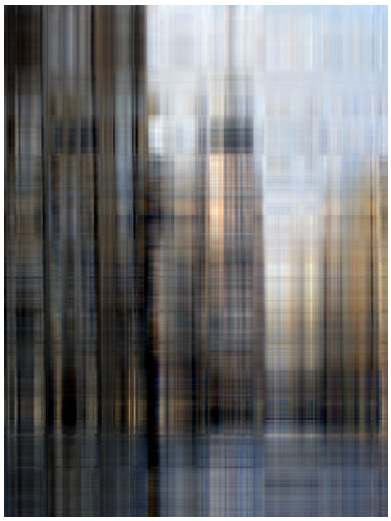
$$A_5 = \sum_{i=1}^{5} \sigma_i u_i v_i^T$$

En una misma ventana (utilizando el comando subplot) muestre la imagen original, la imagen guardada en A_5 y el error $|\texttt{RGB} - A_5|$. Comente sus resultados. Sugerencia: el comando `imshow(RGB)` muestra la imagen dada en `RGB`.

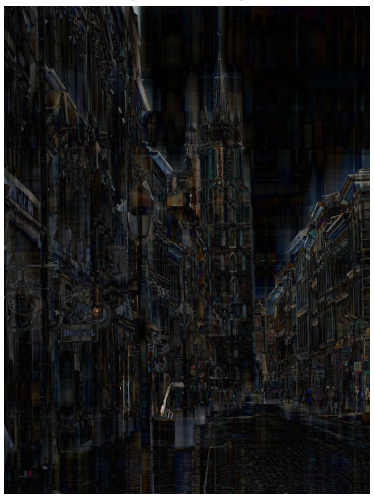
Solución: Al ejecutar la aproximación $A5$, se obtienen las siguientes imágenes 1.

RGB

A5



$|RGB - A5|$



La imagen producida por A_5 se puede ver muy distorsionada, aunque por lo menos preserva la la iluminación y coloración general, lo cual es bastante bueno para solo tomar 5 valores singulares de 2736.

La imagen generada por el error captura los pixeles que A_5 no aproxima correctamente, la mayoría pertenece a regiones de la foto con muchos cambios de colores, algo que es difícil de capturar con pocas entradas numéricas.

d) Para cada matriz `RGB(:,:,i)` determine el numero r_i de valores singulares σ_k tales que $\sigma_k > 0{,}01\sigma_1$ y tome $R_1 = \max r_i$. Para este valor de R_1 calcule la mejor aproximación A_{R_1} de rango R_1 de la matriz `RGB`. Repita para $\sigma_k > 0{,}005\sigma_1$, denote este nuevo rango R_2 y la respectiva aproximación como A_{R_2}. Dibuje en una misma ventana ambas aproximaciones. Comente las diferencias entre ambas imágenes. Sugerencia: el comando `diag(S)` extrae los elementos de la diagonal de la matriz `S`.

Solución: Se adjuntó el script que realiza las instrucciones en `MATLAB`. Se adjunta la comparación entre las imágenes generadas por A_{R_1} y A_{R_2} (página 8).

Se puede observar que ambas imágenes se asemejan a la foto original. Sin embargo, al agrandar la imagen ligeramente, se puede ver que A_{R_1} pierde detalles finos, y se le ve pixeleada bajo aumento. Mientras tanto, A_{R_2} preserva mayor cantidad de detalles, y solo se pixelea si se agranda mucho la imagen. La cantidad de valores singulares para A_{R_1} y A_{R_2} son 75 y 172, respectivamente. (Las imágenes se pueden ver mejor corriendo el script).

A_{R_1}

A_{R_2}

e) Finalmente guarde la imagen A_{R_2}, como archivos `.bmp` y `.jpg` mediante el comando `imwrite(A_R2,'photo_comp.bmp')` y `imwrite(A_R2,'photo_comp.jpg')`. Compare el tamaño de estos dos archivos con el tamaño original de la foto inicial. (Note que el formato `.jpg` por defecto comprime la imagen y está optimizado para tal fin).

Solución: Se generaron las imágenes en los formatos `.jpg` y `.bmp` (se adjuntan en los archivos). El tamaño original de la imagen era de 29242 Kb. Se puede observar que el tamaño de la imagen en formato `.bmp` es el mismo, de lo cual se concluye que este formato de imagen simplemente cuenta los pixeles (el tamaño de la matriz) que se usó, sin importar que esta tenga muchos ceros o sea de rango mucho menor al número de filas que ésta tenga. Mientras tanto, el tamaño de la imagen en formato `.jpg` es de tan solo 989 Kb, un 3 % del tamaño original. Claramente si lo que se desea es eficiencia de memoria, el formato .jpg es el indicado.

Problema 3.

(Estabilidad de la factorización QR) En las notas se discuten tres algoritmos para calcular la factorización QR de una matriz $A \in \mathbb{C}^{m\times n}$, los cuales presentamos a continuación:

Data: Matriz $A \in \mathbb{C}^{m\times n}$
Result: Matriz unitaria Q y triangular superior R tales que $A = QR$.
for $j = 1 : n$ **do**
  $\boldsymbol{v}_j = \boldsymbol{a}_j$;
  for $i = 1 : j - 1$ **do**
    $r_{ij} = \boldsymbol{q}_i^* \boldsymbol{a}_j$;
    $\boldsymbol{v}_j = \boldsymbol{v}_j - r_{ij}\boldsymbol{q}_i$;
  end for
  $r_j j = ||\boldsymbol{v}_j||_2$;
  $\boldsymbol{q}_j = \boldsymbol{v}_j / r_{jj}$;
end for

Algoritmo 1: Ortogonalización de Gram-Schmidt (inestable).

Data: Matriz $A \in \mathbb{C}^{m\times n}$
Result: Matriz unitaria Q y triangular superior R tales que $A = QR$.
$V = A$;
for $j = 1 : n$ **do**
  $r_{ii} = ||\boldsymbol{v}_i||_2$;
  $\boldsymbol{q}_i = \boldsymbol{v}_i / r_{ii}$;
  for $j = i + 1 : n$ **do**
    $r_{ij} = \boldsymbol{q}_i^* \boldsymbol{v}_j$;
    $\boldsymbol{v}_j = \boldsymbol{v}_j - r_{ij}\boldsymbol{q}_i$;
  end for
end for

Algoritmo 2: Ortogonalización de Gram-Schmidt (estable).

Data: Matriz $A \in \mathbb{C}^{m\times n}$
Result: Matriz unitaria Q y triangular superior R tales que $A = QR$.
for $j = 1 : n$ **do**
  $\boldsymbol{x} = A_{j:m,j}$
  $\boldsymbol{v}_j = \text{sign}(x_1)||\boldsymbol{x}||_2 \boldsymbol{e}_1 + \boldsymbol{x}$;
  $\boldsymbol{v}_j = \boldsymbol{v}_j \backslash ||\boldsymbol{v}_j||_2$;
  $A_{j:m,j:n} = A_{j:m,j:n} - 2\boldsymbol{v}_j(\boldsymbol{v}_j^* A_{j:m,j:n})$;
end for
$R = A$;
$Q = I$;
for $j = n : -1 : 1$ **do**
  $Q_{j:m,j:n} = Q_{j:m,j:n} - 2\boldsymbol{v}_j(\boldsymbol{v}_j^* A_{j:m,j:n})$
end for

Algoritmo 3: Triangularización de Householder

a) Escriba tres funciones que calculen cada uno de los algoritmos.

Solución: Se adjuntan las tres funciones `GS_inest,GS_est,Householder` en los archivos.

b) Defina la matriz A mediante las siguientes líneas:

```
m=80;
[U,~] = qr(randn(m));
[V,~] = qr(randn(m));
S=diag(2.^(-1:-1:-80));
A=U*S*V;
```

Solución: Se definió la matriz en `MATLAB`

c) Calcule la factorización QR de A utilizando los tres algoritmos mencionados. Para cada uno calcule $||A - QR||_2$ y $||Q'Q - I||_2$. Comente sus resultados.¿Cuál algoritmo es mejor?¿Compare sus resultados con el comando de `MATLAB` $[q, r] =$ `qr`(A)

Solución: Se resumen los resultados en la siguiente tabla

Tabla 1. Error en uso de algoritmos para factorización QR

Algoritmo	**Error promedio** $\|\|A - QR\|\|_2$	**Error promedio** $\|\|Q^*Q - I\|\|_2$
Gram-Schmidt Inestable	$6{,}6472 \times 10^{-17}$	52,012
Gram-Schmidt Estable	$6{,}61492 \times 10^{-17}$	0,99528
Householder	$1{,}61738 \times 10^{-16}$	$1{,}90284 \times 10^{-15}$
`qr`	$1{,}573642 \times 10^{-16}$	$1{,}98785 \times 10^{-15}$

Nota: Para calcular el error promedio se corrió cada script 5 veces. Puede observarse que el error promedio de la factorización es muy bajo en todos los casos, siendo ligeramente mejor para los métodos de Gram-Schmidt. Sin embargo, los primeros dos métodos no producen una matriz Q del todo unitaria. La Q obtenida del primer método está lejos de ser unitaria, mientras que la obtenida en el segundo método presenta un error aún considerable. Mientras tanto, el método de Householder y el método del software, presentan muy buenas aproximaciones de matrices unitarias. El código de `MATLAB` no permite examinar el comando `qr`, pero se presume que éste coincide con el de Householder. En resumen, el mejor método pareciera ser el de **Householder.**