UNIVERSIDAD YACHAY TECH, ESCUELA DE CIENCIAS MATEMÁTICAS E INFORMÁTICAS

PROYECTO DE COMPRESIÓN DE IMAGENES UTILIZANDO LA DESCOMPOSICIÓN EN VALORES SINGULARES SVD

Richard Torres[*] Vanessa Hinojosa[**] Diana Pereira [***]
Karen Sánchez[****]

21 de Julio del 2016

1. RESUMEN

En este proyecto se comprimirá la información que conforma una imagen con el fin de conseguir una imagen parecida pero que ocupe menor espacio de almacenamiento. Para ello se utilizó la descomposición SVD, así también el método de deflación y método de potencias. Se pudo observar que se obtiene una imagen apreciable utilizando un bajo número de valores singulares, considerando el tamaño de la matriz que representa la imagen. Se observó que el número de valores singulares necesarios para obtener una imagen clara no es estándar, depende de cada imagen. Sin embargo, para las imágenes tratadas los valores singulares necesarios son muy pocos en relación a todos los que contienen la imagen.

[*] richard.torres@yachaytech.edu.ec
[**] vanessa.hinojosa@yachaytech.edu.ec
[***] diana.pereira@yachaytech.edu.ec
[****] karen.sanchez@yachaytech.edu.ecr

2. PLANTEAMIENTO DEL PROBLEMA

La compresión de imágenes abarca un conjunto de técnicas que se aplican a las imágenes, y es un tema importante debido a que muchas de las imágenes utilizan una gran cantidad de datos, del orden de mega y giga bytes. La compresión permite reducir el volumen de información para almacenar y/o transmitir los datos de las imágenes eficientemente, pero sin una pérdida significativa de la calidad de la imagen.

Para este proyecto se requiere reducir los rasgos redundantes de una imagen para conseguir caracteres relevantes y poder distinguir una imagen. Nos basamos en el recurso de que una imagen puede ser representada por una matriz, por lo que para este proyecto se obtiene la aproximación de la matriz original mediante la descomposición en valores singulares SVD. Debido a que es necesario calcular autovalores y autovectores, el método de las potencias y el método de deflación son ideales para obtener una aproximación con gran precisión.

3. METODOLOGÍA

Para dar solución a este problema se siguió el siguiente esquema:

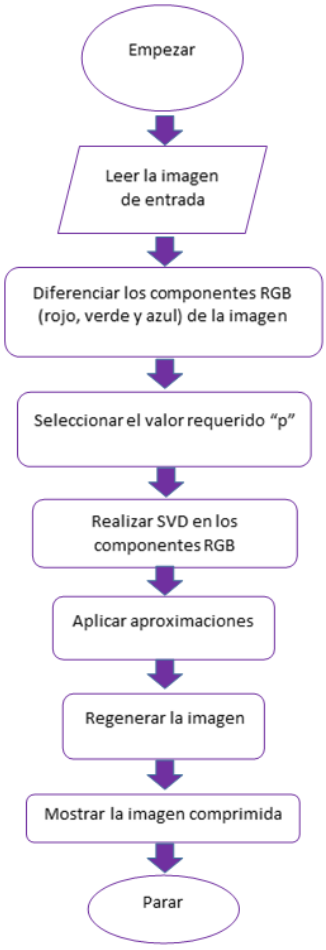


Figura 3.1: Diagrama explicativo del procedimiento realizado.

La descomposición en valores singulares requiere un conocimiento de Álgebra, por lo que es importante utilizar ciertos teoremas correspondientess al tema tratado.

Teorema 1. ***Descomposición en Valores Singulares***

Cada matriz A de dimensión $m \times n$ puede ser factorizada en el producto de U ;una matriz unitaria de orden m, V; matriz unitaria de orden n, y D; matriz diagonal que contiene que contiene los valores singulares $\sigma_1, \sigma_2, ..., \sigma_k$ de A en las entradas de su diagonal.

$$A = UDV^T = (u_1 u_2 ... u_k u_{k+1} ... u_n) \begin{pmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_k & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \\ v_{k+1}^T \\ \vdots \\ v_n^T \end{pmatrix} \tag{3.1}$$

Teorema 2. ***Valores Singulares***

A es una matriz $m \times n$, los valores singulares de A son las raíces cuadradas de los valores propios o autovalores de $A^T A$ y se denotan por $\sigma_1, \sigma_2, ..., \sigma_n$ donde $\sigma_1 \geq \sigma_2 \geq, ..., \sigma_n$. Entonces:

$$\sigma_i = \sqrt{\lambda_i} \tag{3.2}$$

Lema 1. ***Construcción Matriz V***

Para encontrar la matriz V se debe encontrar una base ortonormal $(v_1 v_2 ..... v_n)$ en $\Re^n$ consistente de los vectores propios de una matriz simétrica $n \times n$ $A^T A$, entonces:

$$V = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix} \tag{3.3}$$

Definición 1. ***Construcción Matriz U***

Para construir la matriz ortogonal U se realiza la descomposición QR implementando el método de Gram- Schmidt.

Descomposición *QR La descomposición o factorización QR es la descomposición de una matriz cuadrada V en el producto de una matriz ortogonal Q y una matriz triangular superior R. Uno de los métodos utilizados para la factorización QR es el método de Gram-Schmidt.*

Proceso de Gram- Schmidt *Se considera el procedimiento de Gram- Schmidt con los vectores columnas de la matriz V,*

$$V = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}$$

Luego,

$u_1 = v_1, \qquad e_1 = \frac{u_1}{\|u_1\|}$
$u_2 = v_2 - (v_2 \cdot e_1)\, e_1, \quad e_2 = \frac{u_2}{\|u_2\|}$

Entonces,

$$u_{i+1} = v_{i+1} - (v_{i+1} \cdot e_1)\, e_1 - (v_{i+1} \cdot e_2)\, e_{21} - \cdots - (v_{i+1} \cdot e_i)\, e_i, \quad e_{i+1} = \frac{u_{i+1}}{\|u_{i+1}\|} \tag{3.4}$$

Finalmente tenemos:

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = \underbrace{\begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} v_1 \cdot e_1 & v_2 \cdot e_1 & \cdots & v_n \cdot e_1 \\ 0 & v_2 \cdot e_2 & \cdots & v_n \cdot e_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & v_n \cdot e_n \end{bmatrix}}_{R} = QR \tag{3.5}$$

donde la matriz $Q = U$

Definición 2. ***Valor propio dominante***

Se dice que A tiene valor propio dominante si existe un v.p. λ verificando: $|\lambda_1| > \quad |\lambda_i|, \quad i = 2, 3, ..., n$.

Nuestro próximo objetivo es el cálculo aproximado de λ_1 Para ello, supongamos que la matriz es diagonalizable; es decir, existe una base de vectores propios $\{u_1, \quad u_2, \quad ..., \quad u_n\}$ tales que $Au_i = \lambda_i u_i$

Lema 2. ***Método de la Potencia***

Se describe el método de la potencia para calcular el autopar dominante. Su extensión en el método de la potencia inversa permite encontrar cualquier valor propio siempre que se conozca una aproximación inicial. Algunos esquemas para encontrar valores propios utilizan otros métodos que convergen rápidamente, pero son de precisión limitada. Razón por la cual se utiliza el método de la potencia inversa para refinar los valores numéricos y ganar precisión completa.

Definiciones:

Autovalor dominante
Es el autovalor λ_1 más grande en valor absoluto de la matriz A.

Autovector dominante
Es el autovector v_1 correspondiente al autovalor dominante.

Autovector normalizado
Un autovector se dice normalizado si la coordenada de mayor magnitud es igual a la unidad, es decir igual a uno.

Teorema 3. ***Teorema- Método de la Potencia***

Suponemos una matriz A de $n \times n$ con n autovalores diferentes $\lambda_1, \lambda_2, \lambda_3, .., \lambda_n$ y que estén ordenados en magnitud decreciente $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Si x_0 es seleccionado apropiadamente, entonces las secuencias $x_k = \begin{pmatrix} x_1^{(k)} & x_2^{(k)} & \cdots & x_n^{(k)} \end{pmatrix}^T$ y c_k generado recursivamente por

$$Y_k = AXk \text{ y } Y_{k+1} = \frac{1}{c_{k+1}} X_k, \text{ donde}$$

$$c_{k+1} = x_j^{(k)} \text{ y } x_j^{(k)} = \text{máx}_{1<i<n} \left\{ \left| x_i^{(k)} \right| \right\}$$

pueden converger al autovector v_1 y autovalor λ_1

$$\text{Entonces, } \lim_{k \longrightarrow \infty} x_k = v_k \text{ y } \lim_{k \longrightarrow \infty} c_k = \lambda_k$$

Lema 3. ***Deflación***

Este es un método que es de utilidad si se conoce un valor y vector propio de A de forma que el resto de valores propios se obtienen desde una matriz más sencilla como se indica en el resultado siguiente:

Teorema 4. ***Deflación***

Si λ_1 y u_1 son valor y vector propios de A y v es un vector tal que $v^T.u_1 = 1$; entonces, los valores propios de la matriz $B = A - \lambda_1 (u_1) v^T$ son: $0, \lambda_1, ..., \lambda_n$, donde λ_i es valor propio de A.

Definición 3. ***Método de Hotelling***

Si A es una matriz normal entonces el Teorema de Schur asegura que existe una base ortonormal de autovectores $\{p_1, \quad p_2, \quad ..., \quad p_n\}$; esto es, tal que:

$$p_i^* p_j = \delta_{ij} \quad , \quad \forall \quad i, j = 1, \quad ..., n$$

Se construye entonces la matriz:

$$A_1 = \quad A p_1 p_1^*$$

que verifica:

$$A_1 p_j = 0 \quad , \quad si \quad j = 1 A_1 p_j = \lambda_j p_j \quad , \quad si \quad j \neq 1$$

Por tanto, $Sp(A_1) = \{0, \quad \lambda_2, \quad ..., \lambda_n\}$ Aplicando el método de la potencia iterada a la matriz A_1 se obtiene su autovalor dominante, que es el autovalor intermedio λ_2.

CÓDIGOS ELABORADOS

- Def_potencias.m
- Power_method_Z.m
- compresionproyecto.m
- codigoSVD.m
- Def_Hotelling.m
- gschmidt.m

- main_eig_def.m
- error_relativo_SVD.m
- partial_SVD.m
- USV_geometria.m
- valores_S_aproximacion.m

DESCRIPCIÓN DE LOS CÓDIGOS

El programa **compresionproyecto.m** comprime una imagen utilizando la técnica SVD, requiere la imagen y el número de valores singulares como dato de entrada.

El programa **schmidt.m** ortonormaliza vectores utilizando el procedimiento de Gram Smchmidt y la factorización QR.

El programa **Power_method_Z.m** calcula los auto valore y auto vectores de módulo máximo, este programa se utiliza dentro de **Def_potencias.m** el cual calcula k autovalores de una matriz simétrica A de orden n.

Utilizando el método de deflación de Hotelling en el programa **Def_Hotelling .m** se evalúa el autopar de módulo máximo de A de orden n.

main_eig_def.m se encarga de Evaluar el autopar de A dependiendo del valor de p, usando el método de Deflación de Hotelling.

El código **codigoSVD.m** se encarga de calcular la descomposición SVD, hace uso de todas las anteriores funciones para su cálculo.

El programa **error_relativo_SVD.m** permite encontrar los errores relativos y gráfica respectiva de la imagen, con la descomposición SVD, con distintos valores de k (valores singulares).

Con el programa **partial_SVD.m** se calcula la descomposición SVD.

El programa **USV_geometria.m** permite visualizar el efecto geométrico que tiene premultiplicar a la matriz V por A y a la matriz S por U, para matrices aleatorias de dimensión 2.

El programa **valores_S_aproximacion.m** permite mostrar y graficar los valores singulares de la matriz que representa cada imagen.

4. DISCUSIÓN DE RESULTADOS

Efecto geométrico

En la imagen 4.1 se puede observar el efecto geométrico que tiene multiplicar A con V y U con S. Se puede ver que AV tiene un máximo en v_1 y un mínimo en v_2. De esta manera se optimiza la función sobre el circulo unitario reduciendo el problema complejo a uno bastante estándar. V son los vectores unitarios que pertenecen a S, en otras palabras son pre imágenes de semiejes principales de AS modificados en dirección y sentido.

Los valores singulares describen la cantidad de estiramiento en las diferentes direcciones, entonces cuando se obtiene U*S la circunferencia unitaria descrita por vectores normalizados se deforma convirtiéndose en una elipse cuyos semiejes vienen dados por los vectores afectados por los valores singulares. Donde U son las direcciones de los semiejes principales de AS y numerados en correspondencia con los valeres singulares.

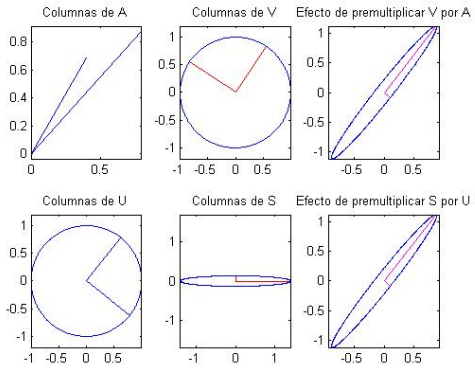


Figura 4.1: Comparación Efecto Geométrico de producto AV y US en una matriz 2*2.

La igualdad $AV = US$ se puede interpretar como que A toma los elementos de una base ortonormal (las columnas de V) y los manda a los generadores de la imagen de A que son los vectores columnas de U estirados o contraídos por los σ_j correspondientes.

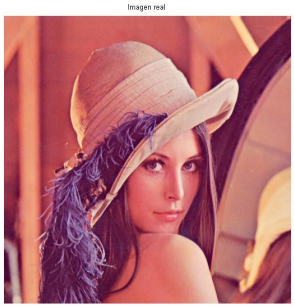
Si $m > n$ (más filas que columnas), al hacer el producto US se pierden las últimas columnas de U. Entonces, hubiera sido lo mismo quedarse con una $\hat{S}$ cuadrada (eliminando las filas nulas) y una $\hat{U} \epsilon \mathbb{C}^{m*n}$ (eliminando las columnas correspondientes). Esta es la forma reducida de la SVD y también podemos pensar a la SVD a partir de completar aquella.

Ahora, dado que los σ_j tienen distinta magnitud, el efecto de A será mayor en las direcciones dadas por los de mayor valor, es decir los primeros $(\sigma_1; \sigma_2, ...)$. Si nos quedamos sólo con las direcciones donde A tiene mayor efecto y eliminamos las otras, entonces no se perderá demasiada información, pero habremos almacenado muchos menos datos.

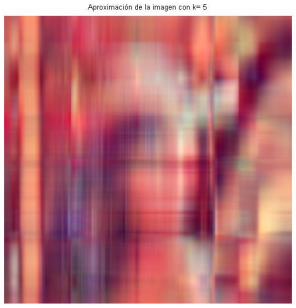
Imágenes con Diferentes Valores de p

En la imagen 4.2 se observa la aproximación de la imagen "$chica$". Es claro que con 5 valores de p la imagen aún no es distinguible, utilizando $p = 15$ la imagen es distinguible. Hay que tomar en cuenta que siendo la matriz original de la imagen de dimensiones $512 *$

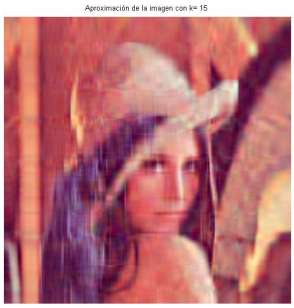
512 utilizar tan solo 15 valores singulares es bastante óptimo. Con 25 valores singulares se obtiene una imagen muy clara, ademas los detalles de fondo son distinguibles. Con $p = 35$ y 45 la imagen tiene muy pocas diferencias con la original. Esto muestra que la compresión SVD es muy eficiente.



(a) real

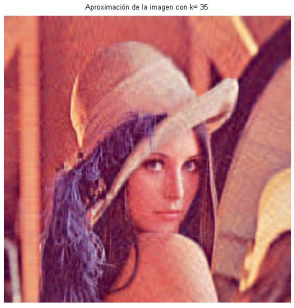


(b) p=5

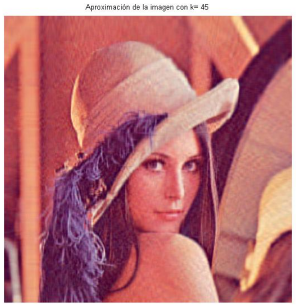


(c) p=15

(d) p=25



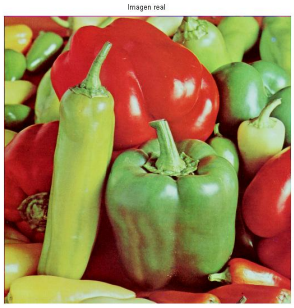
(e) p=35



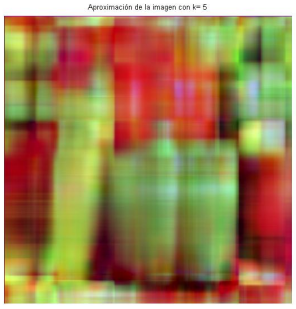
(f) p=45

Figura 4.2: Los resultados de usar diferentes números de p para aproximar la imagen "$chica$".

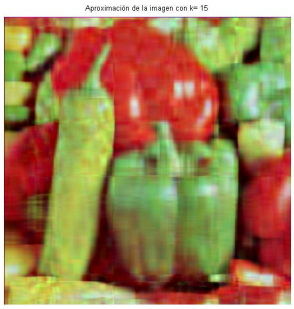
En la imagen 4.3 se aprecia la compresión de la imagen denominada "$vegetales$". Se puede observar que cuando se toma $p = 5$ la imagen no se puede reconocer. Cuando se incrementa a 15 la imagen es más distinguible. Con valores de 25, 35 y 45 la imagen aumenta considerablemente su calidad.



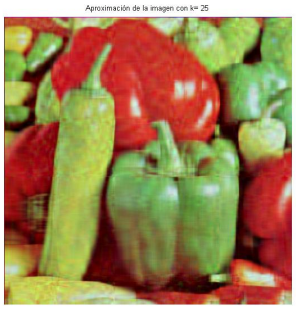
(a) real



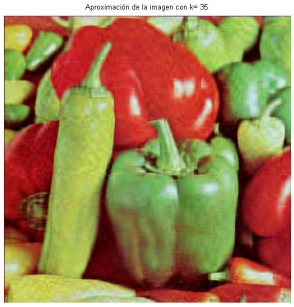
(b) p=5



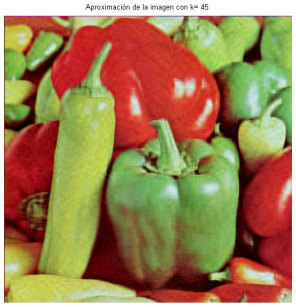
(c) p=15



(d) p=25



(e) p=35



(f) p=45

Figura 4.3: Los resultados de usar diferentes números de p para aproximar la imagen "$vegetales$".

Chica Rojo	Vegetal Rojo
365.6227	303.8652
43.8946	49.0642
33.9613	27.1902
27.6643	26.3689
23.2035	18.8253
22.5182	15.5336
16.1771	15.0075
15.2124	12.4492
12.0248	11.6937
11.4141	11.2532

Tabla 4.1: Comparación de valores singulares de las imágenes chica.tiff y vegetales.tiff referente al color rojo.

Chica Azul	Vegetal Azul
214.0025	138.6910
27.5215	42.5637
25.2558	26.9766
17.0589	26.6432
16.3318	21.9194
14.4995	19.6688
12.5013	18.4798
10.8959	15.2581
10.4112	13.1657
9.5104	12.9927

Tabla 4.2: Comparación de valores singulares de las imágenes chica.tiff y vegetales.tiff referente al color azul.

Chica Verde	Vegetal Verde
205.9599	238.6527
45.9632	82.3208
36.4562	56.3878
28.6373	51.7873
25.5174	37.8863
21.2405	29.6172
20.3553	23.2716
17.7172	21.8246
14.8978	21.1490
13.8124	19.7213

Tabla 4.3: Comparación de valores singulares de las imágenes chica.tiff y vegetales.tiff referente al color verde.

p (número valores singulares)	Error color rojo	Error color verde	Error color azul
5	0.0616	0.1031	0.0678
15	0.0207	0.0448	0.0284
25	0.0129	0.0278	0.0186
35	0.0094	0.0210	0.0145
45	0.0075	0.0165	0.0120

Tabla 4.4: Errores para la imagen chica.tiff con p=5, 15, 25, 35 y 45 valores singulares.

p (número valores singulares)	Error color rojo	Error color verde	Error color azul
5	0.0511	0.1241	0.1418
15	0.0246	0.0477	0.0608
25	0.0153	0.0280	0.0371
35	0.0113	0.0205	0.0247
45	0.0088	0.0154	0.0186

Tabla 4.5: Tabla de errores para la imagen vegetales.tiff con p=5, 15, 25, 35 y 45 valores singulares.

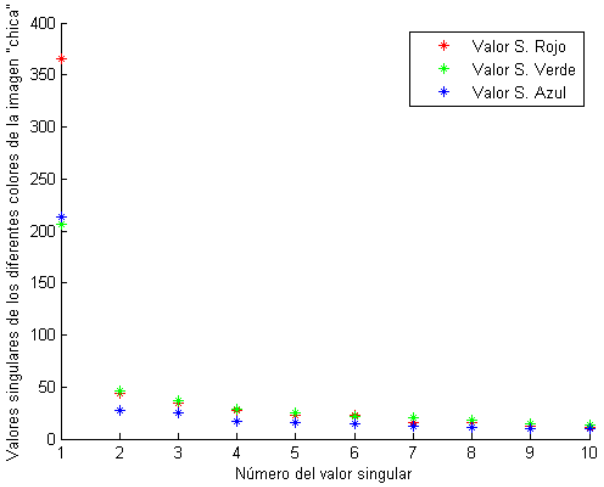


Figura 4.4: Gráfica de los valores singulares de cada color de la imagen "*chica*".

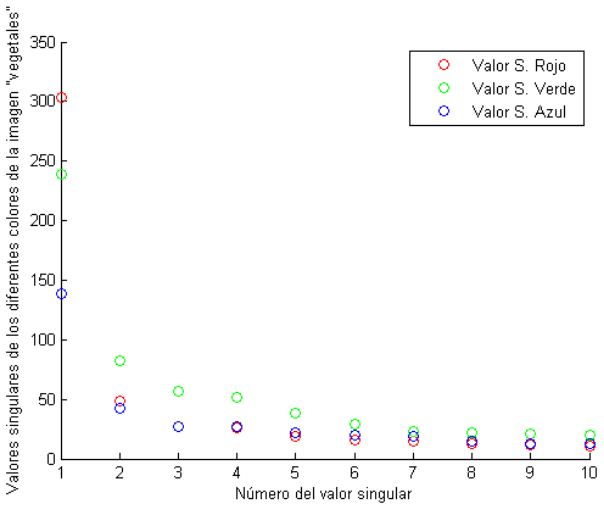


Figura 4.5: Gráfica de los valores singulares de cada color de la imagen "*vegetales*".

En la tabla 4.1 se puede observar que, a excepción del segundo valor singular, todos los valores singulares correspondientes al color rojo de la imagen "*chica*" (Figura 4.4) son mayores comparados con los de la imagen "*vegetales*" (Figura 4.5). Esto podría ser a que la imagen "*chica*" tiene más tonos de color rojo que la imagen "*vegetales*". Por ello en la imagen "*vegetales*" requerimos de menores valores singulares para generar una buena aproximación, debido a que se aproximan más rápido a cero en comparación con la imagen "*chica*".

De acuerdo a la tonalidad de la imagen se presentan los valores que determinan las combinaciones de colores. Así mismo, se observa que en las imágenes con mayor predominio de colores primarios tienen valores singulares menores a las imágenes que presentan una mezcla de estos colores. Es por eso que, al realizar la reducción de imágenes aquellas con valores singulares mayores perderán más información por la descomposición SVD, eso se puede constatar en las tablas 4.4-4.5 que muestran los errores en cada color. En un caso concreto, el color rojo de la imagen "*chica*" tiene un error de 0,0616 (Figura 4.6) comparado

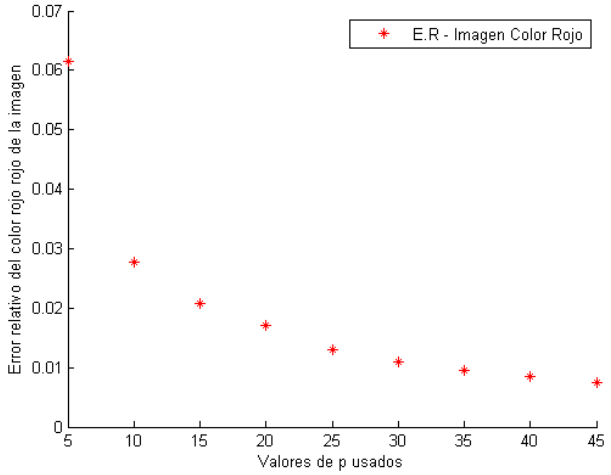


Figura 4.6: Gráfica del error relativo del color rojo de la imagen "*chica*".

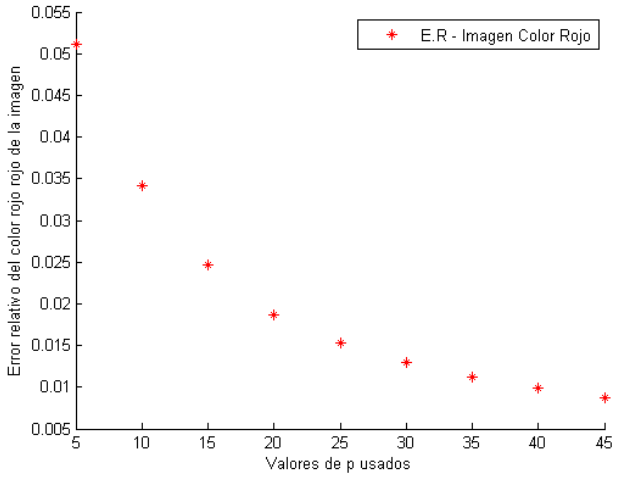


Figura 4.7: Gráfica del error relativo del color rojo de la imagen "*vegetal*".

al error de la imagen "*vegetales*" que tiene un error de 0,0511 (Figura 4.7). Este fenómeno es más notorio en la matriz de color azul en la que el error de la imagen "*chica*" es de 0,0678 mientras que en el error de la imagen "*vegetales*" es de 0,1418.

De este análisis se puede decir que la descomposición en valores singulares toma un tiempo considerable para que realice el proceso, desde el punto de vista computacional. Además, el problema de que su aplicación es fuertemente condicionada debido al exceso cálculos asociados. Además este método podría presentar complicaciones si los valores singulares son muy pequeños y sobrepasan el epsilon de la máquina, pero esté método presenta problemas cuando los valores singulares son 0.

La ventaja más importante de esta descomposición es que cualquier matriz puede ser descompuesta en valores singulares. También constituye uno de los métodos más baratos para compresión de imágenes. Además, utilizar compresión por valores singulares reduce el espacio de almacenamiento necesario, acelera el cálculo ya que utiliza una matriz de menor dimensión con respecto a la original. También, elimina información redundante.

5. CONCLUSIONES

En conclusión ciertas imágenes contienen valores singulares mayores que otras dependiendo de los tonos que presenten, y aplicando la descomposición SVD fue notable que en la matriz de cada color aquellos valores singulares mayores perdieron más información que los valores singulares menores y esto afectó a la nitidez de la imagen.

Además, la metodología empleada permitió facilitar los cálculos en el computador. Es así que, por ejemplo el método de deflación nos permite encontrar los autovalores de módulo mayor, evitando el proceso algebraico de calcularlos a partir del polinomio característico.

Es importante considerar que la imagen a comprimir da una matriz tridimensional por ser a color, por lo que se trabajó en bloques separados al realizar la descomposición SVD y finalmente se concatenó para obtener una imagen a color con menor volumen de datos pero distinguible.

REFERENCIAS

[1] Kincaid David, *Numerical Mathematics and Computing* , Sixth Edition, (2008).

[2] Butt Rizwan, *Introduction to Numerical Analysis Using Matlab* , Jones and Bartlett Publishers, (2010).

[3] Burden Richard, *Análisis Numérico* , Thomson and Learning, Séptima Edición.

[4] Faul C., *A Concise Introduction to Numerical Analysis* , University of Cambrigde UK, A Chapman and Hall Book, (2016).

APÉNDICE A LISTA DE PROGRAMAS COMPUTACIONALES

Lista de todos los programas utilizados para generar los resultados presentados en la sección 4.

```
% ------------------------------------------------------------------------
% Nombre del programa: compresionproyecto.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 12 de 2016
% Breve descripcion del programa: Permite la comprensión de imágenes
% mediante la técnica SVD
% Datos de entrada:
% imagen: Imagen a ser evaluado por el método SVD
% k: número de valores singulares
% Datos de salida:
% U1: Matriz ortogonal de tamaño mxm para el color rojo
% S1: Matriz diagonal con los valores singulares de tamaño mxn para el
% color rojo
% V1: Matriz ortogonal de tamaño nxn para el color rojo
%.........................................................................
% U2: Matriz ortogonal de tamaño mxm para el color verde
% S2: Matriz diagonal con los valores singulares de tamaño mxn para el
% color verde
% V2: Matriz ortogonal de tamaño nxn para el color verde
%.........................................................................
% U3: Matriz ortogonal de tamaño mxm para el color azul
% S3: Matriz diagonal con los valores singulares de tamaño mxn para el
% color azul
% V3: Matriz ortogonal de tamaño nxn para el color azul
% ------------------------------------------------------------------------

function [U1,S1,V1,U2,S2,V2,U3,S3,V3]=compresionproyecto(imagen,k)
% Lectura de la imagen
colorimage = imread(imagen);
% Guarda la imagen en formato .mat
save('imagen.mat','colorimage');
% Mensaje para el usuario
fprintf('Generando las gráficas espere...\n\n');
%.........................................................................
% Utilización del color rojo de la matriz imagen
bwimage1 = colorimage(:, :, 1);
% Imagen como ingreso, y retorna una imagen de clase double
original1 = im2double(bwimage1);
% Valores de U,S y V para la matriz de color rojo, utilizando el esquema
% SVD para valores singulares
[U1,S1,V1]=partial_SVD(original1,k);
% Descomposición parcial de la imagen en rojo
AN1 = U1*S1*V1';
%-------------------------------------------------------------------------
%.........................................................................

%.........................................................................
% Utilización del color verde de la matriz imagen
```

```
bwimage2 = colorimage(:, :, 2);
% Imagen como ingreso, y retorna una imagen de clase double
original2 = im2double(bwimage2);
% Valores de U,S y V para la matriz de color verde, utilizando el esquema
% SVD para valores singulares
[U2,S2,V2]=partial_SVD(original2,k);
% Descomposición parcial de la imagen en verde
AN2 = U2*S2*V2';
%--------------------------------------------------------------------------
%..........................................................................

%..........................................................................
% Utilización del color azul de la matriz imagen
bwimage3 = colorimage(:, :, 3);
% Imagen como ingreso, y retorna una imagen de clase double
original3 = im2double(bwimage3);
% Valores de U,S y V para la matriz de color azul, utilizando el esquema
% SVD para valores singulares
[U3,S3,V3]=partial_SVD(original3,k);
% Descomposición parcial de la imagen en azul6
AN3 = U3*S3*V3';
%--------------------------------------------------------------------------
%..........................................................................

% Graficación de la imagen real y la aproximación de la imagen real
% concatenando sus tres colores (rojo,verde,azul)
figure
hold on
subplot(1,2,1)
imshow(colorimage);
title('Imagen real')
subplot(1,2,2)
% Concatenación de las tres aproximaciones de las matrices de los colores
% rojo, verde y azul
rgbImage = cat(3, AN1,AN2,AN3);
imshow(rgbImage);
title(['Aproximación de la imagen con k= ' num2str(k)])
%--------------------------------------------------------------------------


% ------------------------------------------------------------------------
% Nombre del programa: Power_method_Z.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 10 de 2016
% Breve descripcion del programa: Evalua el autopar de módulo máximo con el
% método de las potencias
% Datos de entrada:
% A: Matriz simétrica de orden n
% tol: tolerancia, como criterio de parada
% maxit: número máximo de iteraciones permitido
% x0: autovector supuesto inicial
% Datos de salida:
% lambda: aproximación del autovalor de módulo máximo
% x: aproximación al autovector asociado
% i: iteraciones requeridas para la convergencia
```

```
% ------------------------------------------------------------------------
function [lambda,x,i]=Power_method_Z(A,tol,maxit,x0)
% Tamaño de la matriz A
[m,n]=size(A);
% Inicializar lamba en infinito
lambda0=inf;
% Norma Frobenius
naf=norm(A,'fro');
% Comprobación que la matriz A es cuadrada
if m~=n
    disp('A no es cuadrada');
    return;
end;
% Inicializar la convergencia en 0
conv=0;
% Inicializar el número de iteraciones requeridas para la convergencia
i=0;

% Bucle que continua hasta que el método alcance su convergencia
%-------------------------------------------------------------------------
while conv==0 && i < maxit
    i=i+1;
    % Multipliación sucesiva de la matriz A por x0
    y = A*x0;
    % Comprobar que la norma de y es menor a una tolerancia
    if norm(y) < 1e-10
        lambda=0;
        return;
    end;
    % Normalización de x
    x = y/norm(y);
    % Cálculo del valor de lambda
    lambda=x'*A*x;
    % Condicional para comprobar que el método converge o no
    if ((norm(x-x0)/naf < tol) || (abs(lambda-lambda0)/naf < tol))
        conv=1;
    end
    % Reasignación de la aproximación del autovalor y del autovector
    x0=x;
    lambda0=lambda;
end;
%-------------------------------------------------------------------------
if i==maxit
    %disp('Numero maximo de iteraciones alcanzado');
else
    %fprintf('it = %3i, lambda = %12.5e\n',i,lambda);
end;
%-------------------------------------------------------------------------


% ------------------------------------------------------------------------
% Nombre del programa: codigoSVD.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 12 de 2016
% Breve descripcion del programa: Permite calcular la descomposición SVD
```

```
% Datos de entrada:
% A: matriz de orden n
% k: número de valores singulares
% Datos de salida:
% U1: Matriz ortogonal de tamaño mxm
% S1: Matriz diagonal con los valores singulares de tamaño mxn
% V1: Matriz ortogonal de tamaño nxn
% ------------------------------------------------------------------------

function [U1,S1,V1]=codigoSVD(A,k)
% Vector que contiene las columnas de la matriz U, cuando aún no es
% ortogonalizada
vectoru1=[];
% Formar una matriz simétrica por medio de A'*A
A1=A'*A;
% Evalua el autopar de la matriz dependiendo del valor de k, usando el
% método de Deflación de Hotelling
[AD1, autoval1, autovec1] = main_eig_def(A1,k);
% Bucle que permite calcular los valores singulares para la matriz
%-------------------------------------------------------------------------
for j=1:k
    % Valores singulares, son las raíces de los autovalores de la matriz
    valores_singulares1=sqrt( autoval1(1:j));
end
%-------------------------------------------------------------------------
% Formación de la matriz diagonal S1 con los valores singulares
S1=diag(valores_singulares1);
% Ortonormalizar las columnas  de los autovectores calculados por medio de
%  Gram-Schmidt
[V1,R1]=gschmidt(autovec1);
% Bucle que permite calcular las columnas para la matriz U1 con la fórmula
% u1=A*v1/(valor singular)
%-------------------------------------------------------------------------
for i=1:k
u1=(A*V1(:,i))/((valores_singulares1(i)));
vectoru1=[vectoru1 u1];
end
%-------------------------------------------------------------------------
vectoru1;
%Ortonormalizar las columnas de la matriz U1
[U1,R]=gschmidt(vectoru1);
%.........................................................................
%-------------------------------------------------------------------------



% ------------------------------------------------------------------------
% Nombre del programa: Def_Hotelling.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 10 de 2016
% Breve descripcion del programa: Evalua el autopar de módulo máximo de A
% de orden n, por medio del método de deflación de Hotelling
% Datos de entrada:
% A: Matriz simétrica de orden n
% Datos de salida:
% ADe: Matriz similar que representa una deflación de A por el método de
```

```
% Hotelling. Si el espectro de A es {l1,l2, ...,ln} tal que:
% |l1| > |l2| >= ....|ln| entonces el espectro de AD es % {0,l2,l3,...,ln}
% lambda: Autovalor
% v1: Autovector
% ------------------------------------------------------------------------

function [ADe,lambda,v1] = Def_Hotelling (A)
% Tamaño de la matriz
[m,n]= size(A);
% Evaluar que sea una matriz cuadrada
if n ~=m
    disp('Matriz A debe ser cuadrada');
    return;
end;
% Tolerancia para la convergencia
tol=1e-14; maxit=1000;
% Vector random de tamaño igual a las columnas de una matriz A
x0=randn(n,1);
% Utilización power method
[lambda,x]=Power_method_Z(A,tol,maxit,x0);
% Normaliza el vector v1
v1=x/norm(x);
% Proceso de deflacion
ADe = A-lambda*(v1*v1');
%------------------------------------------------------------------------


function [autoval] = Def_potencias (A,k);
% Este programa calcula k autovalores de una matriz simétrica A de
% orden n ( k <= n)
% Input:
%        A:  Una matriz simétrica de orden n
%        k:  numero de autovalores deseados

[m,n]= size(A);
if n ~=m disp('Matriz A debe ser cuadrada'); return; end;
if k > n disp('k debe ser menor o igual a n'); return; end;

tol=1e-14; maxit=1000; cont=0; nn=n;
autoval=[];
while nn  > 2 & cont < k
    x0=randn(nn,1);
    % Se calcula el próximo autovalor
    [lambda,x,i]=Power_method_Z(A,tol,maxit,x0);
    autoval = [autoval; lambda]; cont=cont + 1;
    % Se construye la matriz de Householder
    e1=eye(nn,1);u=x-norm(x)*e1;
    H=eye(nn)-2*(u*u')/(u'*u);
    % Se aplica transformación de similaridad
    B=H'*A*H;
    % Preoceso de deflacion
    A = B(2:nn,2:nn); nn=nn-1;
end

% En este punto o bien se han calculado los k autovalores deseados o
% se tiene una matriz deflactada de 2x2 cuyos autovalores se obtienen
% directamente de la fórmula conocida.
```

```
if cont==k
    %fprintf('%3i autovalores calculados exitosamente \n',k);
    return;
end
% Se calculan los dos últimos autovalores de la matriz deflactada de 2x2
% Si A es de 2x2 su polinomio característico es:
% lam^2 - (A11+A22)*lam + det(A) y sus raices pueden calcularse usando la
% ecuación cuadrática.

b=A(1,1)+A(2,2); c=A(1,1)*A(2,2)-A(1,2)^2;;
lambda = (b + sqrt(b^2 - 4*c))/2;
%fprintf('it = %3i, lambda = %12.5e\n',1,lambda);

autoval = [autoval; lambda]; cont=cont+1;
lambda = (b - sqrt(b^2 - 4*c))/2;

%fprintf('it = %3i, lambda = %12.5e\n',1,lambda);

autoval = [autoval; lambda]; cont=cont+1;
%fprintf('%3i autovalores calculados exitosamente \n',cont);



% ------------------------------------------------------------------------
% Nombre del programa: gschmidt.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 10 de 2016
% Breve descripcion del programa: Permite ortonormalizar vectores por medio
% de Gram-Schmidt con la descomposición QR
% Datos de entrada:
% V: Matriz de vectores
% Datos de salida:
% Q: Matriz ortogonal (columnas ortonormales)
% R: Matriz triangular superior
% ------------------------------------------------------------------------
function [Q,R]=gschmidt(V)
% Tamaño de la matriz
[m,n]=size(V);
% Vector de ceros en R, con el mismo tamaño que las columnas V
R=zeros(n);
% Normaliza la primera columna de la matriz V
R(1,1)=norm(V(:,1));
Q(:,1)=V(:,1)/R(1,1);
% Bucle que permite ortonormalizar el resto de columnas de la matriz V
%-------------------------------------------------------------------------
for k=2:n
R(1:k-1,k)=Q(:,1:k-1)'*V(:,k);
% Substracción de la proyección
Q(:,k)=V(:,k)-Q(:,1:k-1)*R(1:k-1,k);
R(k,k)=norm(Q(:,k));
% Normaliza para el siguiente vector qj
Q(:,k)=Q(:,k)/R(k,k);
end
%-------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------
% Nombre del programa: main_eig_def.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 10 de 2016
% Breve descripcion del programa: Evalua el autopar de A dependiendo del
% valor de p, usando el método de Deflación de Hotelling
% Datos de entrada:
% A: Matriz simétrica de orden n
% p: Variable que determina el número de autopares mostrados
% Datos de salida:
% ADe: Matriz similar que representa una deflación de A por el método de
% Hotelling. Si el espectro de A es {l1,l2, ...,ln} tal que:
% |l1| > |l2| >= ....|ln| entonces el espectro de AD es % {0,l2,l3,...,ln}
% lambda: Autovalor
% v1: Autovector
% ------------------------------------------------------------------------

function [ADe, autoval, autovec] = main_eig_def(A,p)
% Tamaño de la matriz A
[m,n]=size(A);
% Variable contenedoras de todos los autovalores y autovectores evaluados
autoval =[];
autovec=[];
% Bucle que permite calcular p autovalores y autovectores
%-------------------------------------------------------------------------
for i=1:p
% Uso del método de Deflación Hotelling
[ADe,lambda,v1] = Def_Hotelling (A);
% Vector con todos los p lambdas evaluados
autoval = [autoval, lambda];
% Vector con todos los p autovectores
autovec = [autovec, v1];
% Intercambio de la matriz ADe  similar por la matriz A
A = ADe;
end
%-------------------------------------------------------------------------
end


% ------------------------------------------------------------------------
% Nombre del programa: error_relativo_SVD.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 12 de 2016
% Breve descripcion del programa: Permite encontrar los errores relativos y
% gráfica respectiva de la imagen, con la descomposición SVD, con distintos
% valores de k (valores singulares)
% Datos de entrada:
% imagen: Imagen a ser evaluado por el método SVD
% Datos de salida:
% kvector: Número de valores singulares utilizados en la aproximación de la
% matriz
% errorR: Error relativo de la imagen con el color rojo
```

```
% errorV: Error relativo de la imagen con el color verde
% errorA: Error relativo de la imagen con el color azul
% Gráfica del error relativo de la imagen
% -------------------------------------------------------------------------

function [kvector,errorR,errorV,errorA]=error_relativo_SVD(imagen)
% Vector de los errores relativos de los diferentes colores de la imagen
errorR=[];
errorV=[];
errorA=[];
kvector=[];
% Lectura de la imagen
colorimage = imread(imagen);
% Guarda la imagen en formato .dat
save('imagen.dat','colorimage');
% Mensaje para el usuario
fprintf('Generando las gráficas espere...\n\n');

% Bucle que permite calcular el error relativo de la imagen a diferentes
% valores de k
%==========================================================================
for k=5:5:45
%.....................................................................
% Utilización del color rojo de la matriz imagen
bwimage1 = colorimage(:, :, 1);
% Imagen como ingreso, y retorna una imagen de clase double
original1 = im2double(bwimage1);
% Valores de U,S y V para la matriz de color rojo, utilizando el esquema
% SVD para valores singulares
[U1,S1,V1]=partial_SVD(original1,k);
% Descomposición parcial de la imagen en rojo
AN1 = U1*S1*V1';
%---------------------------------------------------------------------------
%.....................................................................

%.....................................................................
% Utilización del color verde de la matriz imagen
bwimage2 = colorimage(:, :, 2);
% Imagen como ingreso, y retorna una imagen de clase double
original2 = im2double(bwimage2);
% Valores de U,S y V para la matriz de color verde, utilizando el esquema
% SVD para valores singulares
[U2,S2,V2]=partial_SVD(original2,k);
% Descomposición parcial de la imagen en verde
AN2 = U2*S2*V2';
%---------------------------------------------------------------------------
%.....................................................................

%.....................................................................
% Utilización del color azul de la matriz imagen
bwimage3 = colorimage(:, :, 3);
% Imagen como ingreso, y retorna una imagen de clase double
original3 = im2double(bwimage3);
% Valores de U,S y V para la matriz de color azul, utilizando el esquema
% SVD para valores singulares
[U3,S3,V3]=partial_SVD(original3,k);
% Descomposición parcial de la imagen en azul6
AN3 = U3*S3*V3';
```

```
%-------------------------------------------------------------------------
%.........................................................................
    hold on
    %Error relativo del color rojo de la imagen
    error_rojo=norm((original1-AN1),2)/norm(AN1,2);
    errorR=[errorR error_rojo];
    %Error relativo del color verde de la imagen
    error_verde=norm((original2-AN2),2)/norm(AN2,2);
    errorV=[errorV error_verde];
    %Error relativo del color azul de la imagen
    error_azul=norm((original3-AN3),2)/norm(AN3,2);
    errorA=[errorA error_azul];
    % Graficación de los errores respectivos
    plot(k,error_rojo,'r*')

    % Valores de k usados
    kvector=[kvector k];
end
%=========================================================================
% Transposición de los errores, para tener obtener error tipo columna
errorR=errorR';
errorV=errorV';
errorA=errorA';
% Transposición del vector k, para tener obtener k tipo columna
kvector=kvector';
% Añadir leyenda y nombre a los ejes de la gráfica
legend('E.R - Imagen Color Rojo')
xlabel('Valores de p usados')
ylabel('Error relativo del color rojo rojo de la imagen')
%-------------------------------------------------------------------------


% -------------------------------------------------------------------------
% Nombre del programa: codigoSVD.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 12 de 2016
% Breve descripcion del programa: Permite calcular la descomposición SVD
% Datos de entrada:
% A: matriz de orden n
% k: número de valores singulares
% Datos de salida:
% U1: Matriz ortogonal de tamaño mxm
% S1: Matriz diagonal con los valores singulares de tamaño mxn
% V1: Matriz ortogonal de tamaño nxn
% -------------------------------------------------------------------------

function [U1,S1,V1]=partial_SVD(A,k)
% Vector que contiene las columnas de la matriz U, cuando aún no es
% ortogonalizada
vectoru1=[];
% Formar una matriz simétrica por medio de A'*A
A1=A'*A;
% Evalua el autopar de la matriz dependiendo del valor de k, usando el
% método de Deflación de Hotelling
[AD1, autoval1, autovec1] = main_eig_def(A1,k);
```

```
% Bucle que permite calcular los valores singulares para la matriz
%-------------------------------------------------------------------------
for j=1:k
    % Valores singulares, son las raíces de los autovalores de la matriz
    valores_singulares1=sqrt( autoval1(1:j));
end
%-------------------------------------------------------------------------
% Formación de la matriz diagonal S1 con los valores singulares
S1=diag(valores_singulares1);
% Ortonormalizar las columnas  de los autovectores calculados por medio de
%  Gram-Schmidt
[V1,R1]=gschmidt(autovec1);
% Bucle que permite calcular las columnas para la matriz U1 con la fórmula
% u1=A*v1/(valor singular)
%-------------------------------------------------------------------------
for i=1:k
u1=(A*V1(:,i))/((valores_singulares1(i)));
vectoru1=[vectoru1 u1];
end
%-------------------------------------------------------------------------
vectoru1;
%Ortonormalizar las columnas de la matriz U1
[U1,R]=gschmidt(vectoru1);
%.........................................................................
%-------------------------------------------------------------------------



% -------------------------------------------------------------------------
% Nombre del programa: USV_geometria.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 13 de 2016
% Breve descripcion del programa: Permite visualizar el efecto geométrico
% que tiene premultiplicar a la matriz V por A y a la matriz S por U, para
% matrices aleatorias de dimensión 2.
% Datos de entrada:
% No existen
% Datos de salida:
% A: Matriz aleatoria de orden 2
% Gráficas del efecto geométrico de permutiplicar las matrices V por A y la
% matriz S por U
% -------------------------------------------------------------------------

% Crea una matriz aleatoria de dimensión 2
A = rand(2);
% Implementar la descomposición SVD, encontrar las matrices U,S,V
[U,S,V]=partial_SVD(A,2);
% Mensajes
fprintf('Generando las gráficas...\n\n');
fprintf('La matriz generada es: \n')
% Mostrar la matriz A
A
% Crear una figura, donde se ubicarán todas las gráficas correspondientes
% al efecto geométrico de premultiplicar las matrices anteriormente
% explicadas
        figure
```

```
%Generación de la gráfica  de la matriz A
        subplot(2,3,1)
        a1=A(:,1);
        a2=A(:,2);
        plot([0,a1(1)],[0,a1(2)],'b')
        hold on
        plot([0,a2(1)],[0,a2(2)],'b')
        title('Columnas de A')
        axis equal
% Generación de la gráfica de la matriz U
        subplot(2,3,4)
        u1=U(:,1);
        u2=U(:,2);
        plot([0,u1(1)],[0,u1(2)],'b')
        hold on
        plot([0,u2(1)],[0,u2(2)],'b')
        t=-pi:0.01:pi;
        x=cos(t);
        y=sin(t);
        hold on
        plot(x,y)
        title('Columnas de U')
        axis equal

% Generación de la gráfica de la matriz V
        subplot(2,3,2)
        v1=V(:,1);
        v2=V(:,2);
        plot([0,v1(1)],[0,v1(2)],'r')
        axis equal
        hold on
        plot([0,v2(1)],[0,v2(2)],'r')
        t=-pi:0.01:pi;
        x=cos(t);
        y=sin(t);
        hold on
        plot(x,y)
        title('Columnas de V')

% Generación de la gráfica de la matriz S
        subplot(2,3,5)
        s1=S(:,1);
        s2=S(:,2);
        plot([0,s1(1)],[0,s1(2)],'r')
        hold on
        plot([0,s2(1)],[0,s2(2)],'r')
        title('Columnas de S')
        axis equal
        m=size(x,2);
        X=zeros(2,m);
        for j=1:1:m
            X(1,j)=x(j);
            X(2,j)=y(j);
            F=S*X;
            plot(F(1,j),F(2,j))
        end
        axis equal
```

```
        % Efecto de premultiplicar la matriz V por la matriz A
        subplot(2,3,3)
        B=A*V;
        b1=B(:,1);
        b2=B(:,2);
        plot([0,b1(1)],[0,b1(2)],'m')
        hold on
        plot([0,b2(1)],[0,b2(2)],'m')
        title('Efecto de premultiplicar V por A')
        % Gráfica de la elipse, cuyos semiejes son b1 y b2
        m=size(x,2);
        X=zeros(2,m);
        for j=1:1:m
            X(1,j)=x(j);
            X(2,j)=y(j);
            E=A*X;
            plot(E(1,j),E(2,j))
        end
        axis equal

        % Efecto de premultiplicar la matriz S por la matriz U
        subplot(2,3,6)
        C=U*S;
        c1=C(:,1);
        c2=C(:,2);
        plot([0,c1(1)],[0,c1(2)],'m')
        hold on
        plot([0,c2(1)],[0,c2(2)],'m')
        title('Efecto de premultiplicar S por U')
        % Gráfica de la elipse cuyos semiejes son c1 y c2
        m=size(x,2);
        X=zeros(2,m);
        for j=1:1:m
            G=U*F;
            plot(G(1,j),G(2,j))
        end
        axis equal
%-------------------------------------------------------------------------


% -------------------------------------------------------------------------
% Nombre del programa: valores_S_aproximacion.m
% Autores: Richard Torres - Karen Sanchez - Vanessa Hinojosa - Diana Pereira
% Email de los autores:
% richard.torres@yachaytech.edu.ec - karen.sanchez@yachaytech.edu.ec-
% vanessa.hinojosa@yachaytech.edu.ec - diana.pereira@yachaytech.edu.ec
% Fecha de elaboracion: Julio 14 de 2016
% Breve descripcion del programa: Permite mostrar y graficar los valores
% singulares de la matriz que representa cada imagen.
% Datos de entrada:
% imagen1: Imagen 1 a ser evaluado por el método SVD
% imagen2: Imagen 2 a ser evaluado por el método SVD
% k: número de valores singulares
% Datos de salida:
% puntos: número del valor singular calculado
% VSR1: Valores singulares de la primera imagen con el color rojo
% VSV1:Valores singulares de la primera imagen con el color verde
% VSA1:Valores singulares de la primera imagen con el color azul
```

```
%.....................................................................
% VSR2:Valores singulares de la segunda imagen con el color rojo
% VSV2:Valores singulares de la segunda imagen con el color verde
% VSA2:Valores singulares de la segunda imagen con el color azul
% ---------------------------------------------------------------------

function [puntos,VSR1,VSV1,VSA1,VSR2,VSV2,VSA2]=valores_S_aproximacion(imagen1,imagen2,k
% Evaluación de la matrices de las imagenes por medio de la compresión SVD
[U1,S1,V1,U2,S2,V2,U3,S3,V3]=compresionproyecto(imagen1,k);
[U11,S11,V11,U22,S22,V22,U33,S33,V33]=compresionproyecto(imagen2,k);
% Número de los valores singulares
puntos=[];
% Valores singulares de la imagen 1
%=====================================================================
VSR1=diag(S1);
VSV1=diag(S2);
VSA1=diag(S3);
%=====================================================================

% Valores singulares de la imagen 2
%=====================================================================
VSR2=diag(S11);
VSV2=diag(S22);
VSA2=diag(S33);
%=====================================================================
figure
% Bucle que permite graficar los puntos de los valores singulares y los
% valores singulares de las matrices de las imagenes 1 y 2
%.....................................................................
for i=1:k
    puntos=[puntos; i];
    hold on
    plot(i,S1(i,i),'r*')
    plot(i,S2(i,i),'g*')
    plot(i,S3(i,i),'b*')


end
legend('Valor S. Rojo','Valor S. Verde','Valor S. Azul')
xlabel('Número del valor singular')
ylabel('Valores singulares de los diferentes colores de la imagen "chica"')
figure
for j=1:k

    hold on
    plot(j,S11(j,j),'ro')
    plot(j,S22(j,j),'go')
    plot(j,S33(j,j),'bo')

end
%.....................................................................
% Añadir leyenda y nombre a los ejes de la gráfica
legend('Valor S. Rojo','Valor S. Verde','Valor S. Azul')
xlabel('Número del valor singular')
ylabel('Valores singulares de los diferentes colores de la imagen "vegetales"')
%---------------------------------------------------------------------
```