Tarea 05

Juan Diego Murcia Porras jmurciap@unal.edu.co
Carlos Enrique Nosa Guzmán cnosa@unal.edu.co
Brayan Alejandro Romero Castro brromeroc@unal.edu.co

El procedimiento de regularización mejor conocido, es llamado regularización de Tikhonov, éste calcula una solución del problema de mínimos cuadrados amortiguado:

$$\min_f \|g - Kf\|_2^2 + \alpha^2 \|f\|_2^2 \tag{1}$$

1. Pruebe que (1) es equivalente al problema lineal de mínimos cuadrados

$$\min_f \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} K \\ \alpha I \end{bmatrix} f \right\|_2^2 \tag{2}$$

Solución
Con este nuevo enfoque, el primer término evalúa qué tan cerca está Kf de nuestro vector g, mientras que el segundo término castiga grandes inestabilidades (la inestabilidad afecta en gran medida a la norma euclídea). Note que el vector por dentro de la norma en (2) se puede representar para cualquier vector f como

$$\begin{bmatrix} g - Kf \\ -\alpha I f \end{bmatrix} = \begin{bmatrix} g - Kf \\ -\alpha f \end{bmatrix}.$$

Sin embargo, la norma euclídea de este vector al cuadrado es igual a la suma de los cuadrados de sus componentes. Por lo tanto, podemos dividir la suma correspondiente al vector de más arriba en dos: la correspondiente a las componentes de $g - Kf$, y la correspondiente a los componentes de $-\alpha f$. En otras palabras,

$$\left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} K \\ \alpha I \end{bmatrix} f \right\|_2^2 = \left\| \begin{bmatrix} g - Kf \\ -\alpha I f \end{bmatrix} \right\|_2^2 = ||g - Kf||_2^2 + || - \alpha f||_2^2 = ||g - Kf||_2^2 + \alpha^2 ||f||_2^2.$$

En conclusión, las funciones a minimizar en (1) y (2) son exactamente las mismas, y concluimos que los problema de minimización son equivalentes (se está minimizando la misma función). Nótese que la dimensión de la matriz I en la ecuación (2) es la misma que la de la matriz K.

2. Muestre que si K tiene una descomposición de valores singulares $K = U\Sigma V^T$ entonces (2) puede escribirse en el problema de mínimos cuadrados equivalente:

$$\min_{\hat{f}} \left\| \begin{bmatrix} \hat{g} \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma \\ \alpha I \end{bmatrix} \hat{f} \right\|_2^2 \tag{3}$$

Donde $\hat{f} = V^T f$, y $\hat{g} = U^T g$.
Solución

Lema 1: Sean U, V matrices ortogonales de dimensión n. Tenemos que

$$A = \begin{bmatrix} U & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & V \end{bmatrix}$$

es una matriz ortogonal. En efecto,

$$A^T = \begin{bmatrix} U^T & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & V^T \end{bmatrix},$$

dada la simetría de la matriz. Por ende,

$$AA^T = \begin{bmatrix} U & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & V \end{bmatrix} \begin{bmatrix} U^T & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & V^T \end{bmatrix} = \begin{bmatrix} UU^T & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & VV^T \end{bmatrix} = \begin{bmatrix} I_{n\times n} & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n} & I_{n\times n} \end{bmatrix} = I_{2n\times 2n}.$$

Al ser A cuadrada, concluimos que $AA^T = I_{2n\times 2n}$. □

Lema 2: Sea U una matriz ortogonal de dimensión n y $x \in \mathbb{R}^n$. Luego

$$||Ux||_2^2 = ||x||_2^2.$$

En efecto, para $y \in \mathbb{R}_\kappa$, tenemos que

$$||y||_2^2 = \langle y, y \rangle = y^T\ y,$$

donde $\langle \cdot, \cdot \rangle$ denota el producto interno. Así.

$$||Ux||_2^2 = \langle Ux, Ux \rangle = (Ux)^T\ (Ux) = x^T\ U^T U\ x = x^T\ x = ||x||_2^2,$$

donde la penúltima desigualdad se tiene gracias a que U es ortogonal. □

Si $K = U\Sigma V^T$, entonces, por el Lema 1,

$$\begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix}$$

es una matriz ortogonal. Así, usando el Lema 2,

$$\begin{aligned}
\left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} K \\ \alpha I \end{bmatrix} f \right\|_2^2 &= \left\| \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} \left(\begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} K \\ \alpha I \end{bmatrix} f \right) \right\|_2^2 \\
&= \left\| \begin{bmatrix} U^T g \\ 0 \end{bmatrix} - \begin{bmatrix} U^T K \\ \alpha V^T \end{bmatrix} f \right\|_2^2 \\
&= \left\| \begin{bmatrix} U^T g \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma V^T \\ \alpha V^T \end{bmatrix} f \right\|_2^2 \\
&= \left\| \begin{bmatrix} U^T g \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma \\ \alpha I \end{bmatrix} V^T f \right\|_2^2 \\
&= \left\| \begin{bmatrix} \hat{g} \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma \\ \alpha I \end{bmatrix} \hat{f} \right\|_2^2,
\end{aligned}$$

donde $\hat{f} = V^T f$ y $\hat{g} = U^T g$. Así, ambas funciones son la misma. Es equivalente minimizar respecto a f y respecto a $\hat{f}$, porque dado f podemos hallar $\hat{f}$ como $\hat{f} = V^T f$, y dado $\hat{f}$ podemos hallar f como $f = V\hat{f}$.

3. Determine una fórmula para resolver (3).

Solución

Considere el problema de encontrar el mínimo de la función

$$\phi(x) = \left\| \begin{bmatrix} \hat{g} \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma \\ \alpha I \end{bmatrix} x \right\|_2^2$$

Note que esta es una función escalar, es decir toma vectores y nos regresa números reales, además note que la norma de un vector describe una función diferenciable en todo su dominio además de ser un paraboloide en $\mathbb{R}^{n+1}$, por ende se puede demostrar que esta es una función convexa [1] y tiene un máximo o mínimo global (en este caso mínimo), de esta forma podemos derivar normalmente e igualar a 0.

Haciendo un análisis análogo, al que se hizo en el primer punto sabemos que minimizar la anterior función es equivalente a minimizar la función:

$$\hat{\phi}(x) = ||\hat{g} - \Sigma x||^2 + \alpha^2 ||x||^2$$

que utilizando el hecho de que la norma 2 al cuadrado de un vector es el producto punto de el consigo mismo, llegamos a:

$$\hat{\phi}(x) = \hat{g}^T g - x^T \Sigma^T \hat{g} - \hat{g}^T \Sigma x + x^T \Sigma^T \Sigma x + \alpha^2 x^T x$$

ahora derivemos $\hat{\phi}(x)$ respecto a x, para ello recordemos que si vamos a derivar $x^T \cdot a$, donde a es un vector $n \times 1$, su derivada es a (basta expandir el producto punto y tomar el gradiente para ver que esto es cierto). Además si

tenemos $x^T Ax$ con A una matriz simétrica de tamaño $n \times n$, entonces su derivada es $2Ax$. También note que $(\hat{g}^T \Sigma x)^T = x^T \Sigma^T \hat{g}$, como ambos son escalares llegamos a que son iguales. Aplicando todo lo anterior, llegamos a:

$$\hat{\phi}(x) = \hat{g}^T g - 2x^T \Sigma^T \hat{g} + x^T \Sigma^T \Sigma x + \alpha^2 x^T x$$

derivando

$$\frac{d\hat{\phi}}{dx} = -2\Sigma^T \hat{g} + 2\Sigma^T \Sigma x + 2\alpha^2 x$$

igualando a 0, encontramos que:

$$0 = -2\Sigma^T \hat{g} + 2\Sigma^T \Sigma x_{min} + 2\alpha^2 x_{min}$$

Despejando x_{min} obtenemos

$$x_{min} = (\Sigma^T \Sigma + \alpha^2 I)^{-1} \Sigma^T \hat{g}$$

Note que la inversa de la última parte existe pues es una matriz diagonal con entradas no nulas en ella, la anterior formula soluciona el problema de mínimos cuadrados amortiguado.

4. Muestre que la solución al problema

$$\min_f ||g - Kf||^2 \tag{4}$$

Puede escribirse como

$$f_{ls} = V\Sigma^\dagger U^T g \approx \sum_{i=0}^{n} \frac{u_i^T g}{\sigma_i} v_i \tag{5}$$

Solución

Tomando $\phi(x) = ||g - Kx||^2$, para el caso en el que todos los valores singulares son mayores a 0 y por tanto K es invertible , (lo cual es común cuando tenemos matrices provenientes de imágenes) tenemos de manera análoga al problema anterior, que la solución a esto es:

$$x_{min} = (K^T K)^{-1} K^T g \tag{6}$$

Como $K = U\Sigma V^T$, obtenemos que:

$$x_{min} = K^{-1}(K^T)^{-1}(K^T)g = (U\Sigma V^T)^{-1} g$$

como $V^{-1} = V^T$ y $U^{-1} = U^T$ obtenemos que:

$$x_{min} = V\Sigma^{-1} U^T g = \sum_{i=0}^{n} \frac{u_i^T g}{\sigma_i} v_i$$

En el caso en el que algunos σ_i sean 0 no podemos tomar la inversa de la ecuación (6), en ese caso cuando derivamos llegamos a la siguiente ecuación:

$$(K^T K) x_{min} = K^T g \tag{7}$$

la cual es equivalente (usando descomposición SVD en K) a la ecuación:

$$\Sigma^T \Sigma V^T x_{min} = \Sigma^T U^T g \tag{8}$$

ahora supongamos que hay r valores singulares no nulos, entonces podemos ver a la matriz Σ por bloques así:

$$\Sigma = \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(n-r)} \end{bmatrix}$$

En este caso Σ_r si es invertible pues es una matriz $r \times r$ con diagonal no nula, y su inversa en este caso es la matriz que en su diagonal tiene a $1/\sigma_i$, luego si multiplicamos por esta matriz a ambos lados de la ecuacion (8) obtenemos:

$$\begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^T & 0 \\ 0 & 0 \end{bmatrix} \Sigma V^T x_{min} = \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^T & 0 \\ 0 & 0 \end{bmatrix} U^T g$$

Como $\Sigma_r^T = \Sigma_r$, obtenemos que lo anterior es:

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V^T x_{min} = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} U^T g$$

donde I_r es la identidad de tamaño r, nuevamente multiplicando por la inversa por bloques nos queda:

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} V^T x_{min} = \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} U^T g.$$

Particionando a U, V por bloques y haciendo el producto indicado obtenemos:

$$\begin{bmatrix} V_r^T \\ 0 \end{bmatrix} x_{min} = \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_r^T \\ 0 \end{bmatrix} g$$

Donde V_r es la matriz de tamaño $n \times r$ que contiene a las primeras r columnas de V, al igual que U_r, multiplicando a ambos lados por $\begin{bmatrix} V_r & 0 \end{bmatrix}$, nos queda:

$$\begin{bmatrix} V_r & 0 \end{bmatrix} \begin{bmatrix} V_r^T \\ 0 \end{bmatrix} x_{min} = \begin{bmatrix} V_r & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_r^T \\ 0 \end{bmatrix} g$$

lo cual es equivalente a:

$$V_r V_r^T x_{min} = \begin{bmatrix} V_r & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_r^T \\ 0 \end{bmatrix} g$$

como $V_r V_r^T \approx I_n$ (es casi el producto entre V y V^T su inversa) obtenemos que:

$$x_{min} \approx \begin{bmatrix} V_r & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_r^T \\ 0 \end{bmatrix} g = \sum_{i=0}^{r} \frac{u_i^T g}{\sigma_i} v_i.$$

Parte de programación

El objetivo en esta sección es recuperar el texto que se encuentra distorsionado en una imagen borrosa. Como datos para trabajar el problema tenemos los siguientes:

```
load('DatosProy.mat')
```

Observamos las variables contenidas en este conjunto de datos:

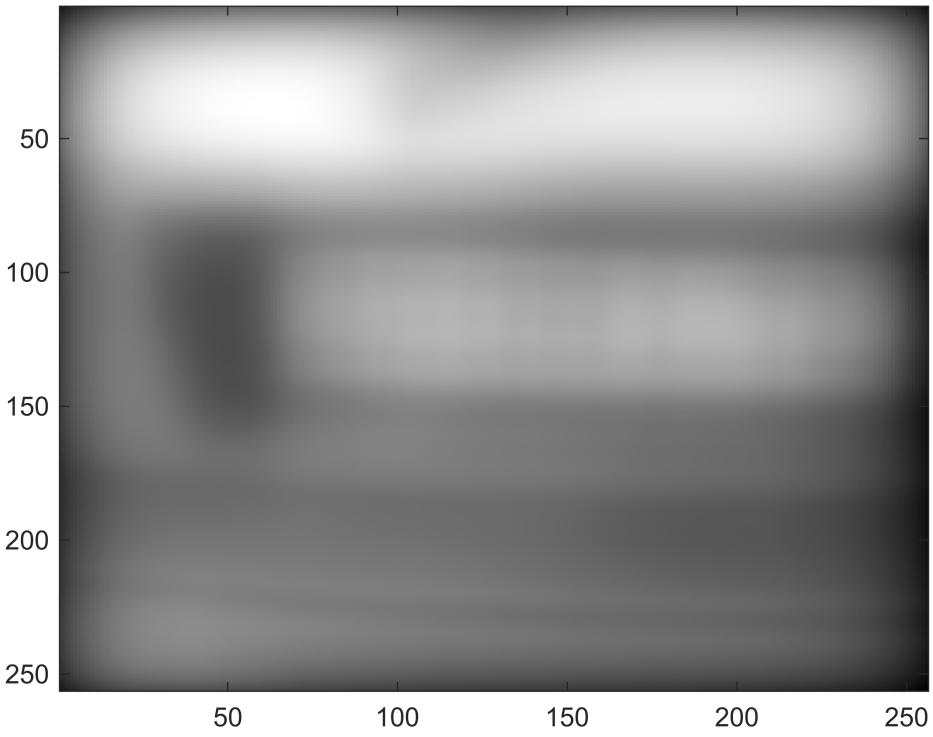
```
whos
```

```
  Name        Size              Bytes  Class     Attributes

  A         256x256            524288  double
  B         256x256            524288  double
  G         256x256            524288  double
```

Nótese que tenemos la imagen resumida en la matriz G, y otras dos matrices A y B tales que contienen la información del "obturador" Kde la siguiente manera $K = A \otimes B$, en donde $\otimes$ representa el producto de Kronecker. Las dimensiones de A, B y G son 256×256, y de la matriz K son $256^2 \times 256^2$.

La imagen distorsionada se muestra a continuación.

```
imagesc(G), colormap(gray)
```



Para restaurar esta imagen usamos dos métodos: Regularización de Tikhonov y Descomposición en valores singulares truncada.

Recordemos que el problema es encontrar la imagen f de la ecuación $K \cdot f = g$ donde K es el obturador y g es la imagen borrosa. Considere g como el vector de dimensiones $256^2 \times 1$ obtenido de redimensionar la matriz G.

```
%Orden de la matriz G
n = 256;
%Redimensión de G
g =  reshape(G, n^2, 1)
```

```
g = 65536×1
   48.3383
   51.1216
   53.8958
   56.6217
   59.3018
   61.9131
   64.4612
   66.9284
   69.2934
   71.5343
      ⋮
```

La matriz K tiene dimensiones demasiado grandes para ser manejadas por Matlab. Si calculamos el producto de Kronecker entre las matrices A y B nos queda una matriz con un poco más de 4×10^9 entradas que, según Matlab, ocupa un aproximado de 32 GB; esto excede el máximo tamaño de memoria para una matriz que es de aproximadamente 7.8 GB.

```
> kron(A,B)
equested 256x256x256x256 (32.0GB) array exceeds maximum array size preference (7.8GB). This might cause MATLAB to become unresponsive.

rror in kron (line 35)
  K = reshape(A.*B, ma*mb, na*nb);

elated documentation
```

Figura. Error al intentar operar la matriz K en Matlab.

Debido a que necesitamos la descomposición en valores singulares del obturador K para aplicar los métodos, afrontamos este problema haciendo uso del siguiente teorema:

Teorema 1. Si $A = U_A \Sigma_A V_A^T$, $B = U_B \Sigma_B V_B^T$ y $K = A \otimes B$ entonces $K = U \Sigma V^T$ en donde $U = U_A \otimes U_B$, $V = V_A \otimes V_B$ y $\Sigma = \Sigma_A \otimes \Sigma_B$.

Por otra parte, para facilitar la multiplicación entre una matriz, que es igual al producto de Kronecker entre otras dos matrices más pequeñas, y un vector usamos el siguiente resultado:

Teorema 2. Sean $A, B \in \mathbb{R}^{n \times n}$ y $g \in \mathbb{R}^{n^2 \times 1}$. Si $G = \textit{reshape}(g, n, n)$, entonces el producto $(A \otimes B)g$ puede ser calculado como

$$(A \otimes B)g = \textit{reshape}(BGA^T, n^2, 1)$$

Demostración (Teorema 2). Sea $g_i = g((i-1)n+1 : in, 1) \in R^{n\times 1}$ para $i = 1, 2, \dots, n$, por ende,

$$\mathbf{g} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}_{n^2\times 1} \qquad \mathbf{G} = \begin{pmatrix} | & | & & | \\ | & | & & | \\ g_1 & g_2 & \cdots & g_n \\ | & | & & | \\ | & | & & | \end{pmatrix}_{n\times n}$$

Por otra parte,

$$(A\otimes B)g = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ a_{21}B & \dots & a_{2n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nn}B \end{pmatrix} \cdot \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} a_{1k}Bg_k \\ \sum_{k=1}^{n} a_{2k}Bg_k \\ \vdots \\ \sum_{k=1}^{n} a_{nk}Bg_k \end{pmatrix}$$

$$= reshape\left(\begin{pmatrix} | & | & & | \\ \sum_{k=1}^{n} a_{1k}Bg_k & \sum_{k=1}^{n} a_{2k}Bg_k & \dots & \sum_{k=1}^{n} a_{nk}Bg_k \\ | & | & & | \end{pmatrix}, n^2, 1\right)$$

$$= reshape\left(B\begin{pmatrix} | & | & & | \\ \sum_{k=1}^{n} a_{1k}g_k & \sum_{k=1}^{n} a_{2k}g_k & \dots & \sum_{k=1}^{n} a_{nk}g_k \\ | & | & & | \end{pmatrix}, n^2, 1\right)$$

$$= reshape\left(B\begin{pmatrix} | & | & & | \\ g_1 & g_2 & \cdots & g_n \\ | & | & & | \end{pmatrix}\begin{pmatrix} a_{11} & \dots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{nn} \end{pmatrix}, n^2, 1\right)$$

$$= reshape\left(BGA^T, n^2, 1\right)$$

■

A continuación se hace la descomposición en valores singulares de las matrices A y B.

```
%Descomposición SVD de A y B
[UA, SA , VA] = svd(A);
[UB, SB , VB] = svd(B);
```

Analicemos los valores singulares de la matriz K. Por el Teorema 1 sabemos que $\Sigma = \Sigma_A \otimes \Sigma_B$, además que cada una de esta matrices es diagonal, por tanto, para calcular la matriz Σ simplemente consideramos las entradas de la diagonal y en el código tratamos estos valores como un vector de tamaño $n^2 \times 1$ llamado *diagS.* Es sencillo notar que:

$$diag\Sigma = (1_{n\times 1} \otimes diag\Sigma_A) \ .* \ (diag\Sigma_B \otimes 1_{n\times 1})$$

en donde $1_{n\times 1}$ es el vector de tamaño $n \times 1$ con todas sus componentes iguales a 1y $.*$ representa la operación de Matlab *element wise multiplication*.

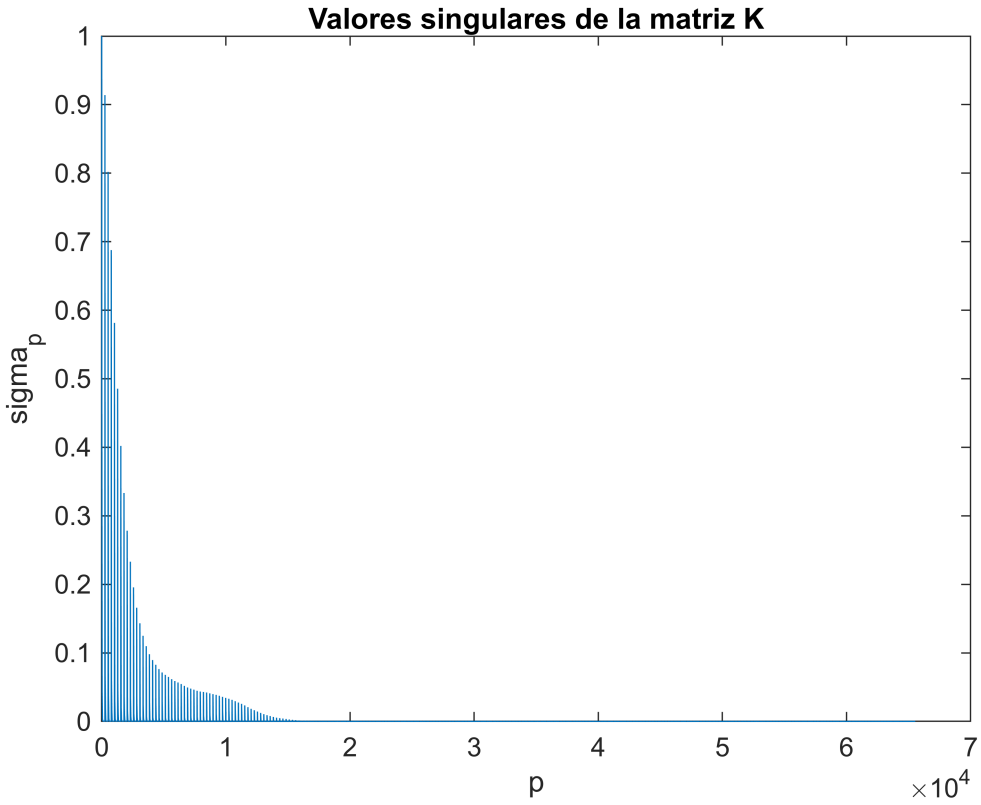
$$\Sigma = \Sigma_A \otimes \Sigma_B = \begin{pmatrix} \sigma_1^A \begin{pmatrix} \sigma_1^B & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n^B \end{pmatrix} & \dots & 0 \begin{pmatrix} \sigma_1^B & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n^B \end{pmatrix} \\ \vdots & \ddots & \vdots \\ 0 \begin{pmatrix} \sigma_1^B & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n^B \end{pmatrix} & \dots & \sigma_n^A \begin{pmatrix} \sigma_1^B & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n^B \end{pmatrix} \end{pmatrix}$$

Figura. Matriz de valores singulares de K.

```
%Diagonal de la matriz de valores singulares de K
diagS = kron(diag(SA),ones(n,1)).*kron(ones(n,1),diag(SB));
```

Para visualizar el comportamiento de los valores singulares asociados a Kgraficamos el escalamiento $(\sigma_1 = 1)$ de cada valor singular.

```
plot(diagS/diagS(1))
box on
xlabel('p')
ylabel('sigma_p')
title('Valores singulares de la matriz K')
```



Puede observarse que los valores singulares decaen a cero sin una separación significatica entre valores consecutivos, por lo tanto, este es un problema *mal propuesto discreto*, como se definió anteriormente.

Regularización de Tikhonov

Sea α un número real positivo. Debemos encontrar un vector f que esté en el conjunto solución del problema

$$mín_f ||g - Kf||_2^2 + \alpha^2 ||f||_2^2$$

Este problema es equivalente a

$$\text{mín}_{\hat{f}} ||\hat{g} - \Sigma \hat{f}||_2^2 + \alpha^2 ||\hat{f}||_2^2$$

donde $\hat{f} = V^T f$ y $\hat{g} = U^T g$. Según lo demostrado anteriormente, se encuentra que una solución a este problema es

$$\underset{\text{min}}{\hat{f}} = (\Sigma^T \Sigma + \alpha^2 I)^{-1} \Sigma^T \hat{g}$$

Por ende, $f_{mín} = V \hat{f}_{mín}$.

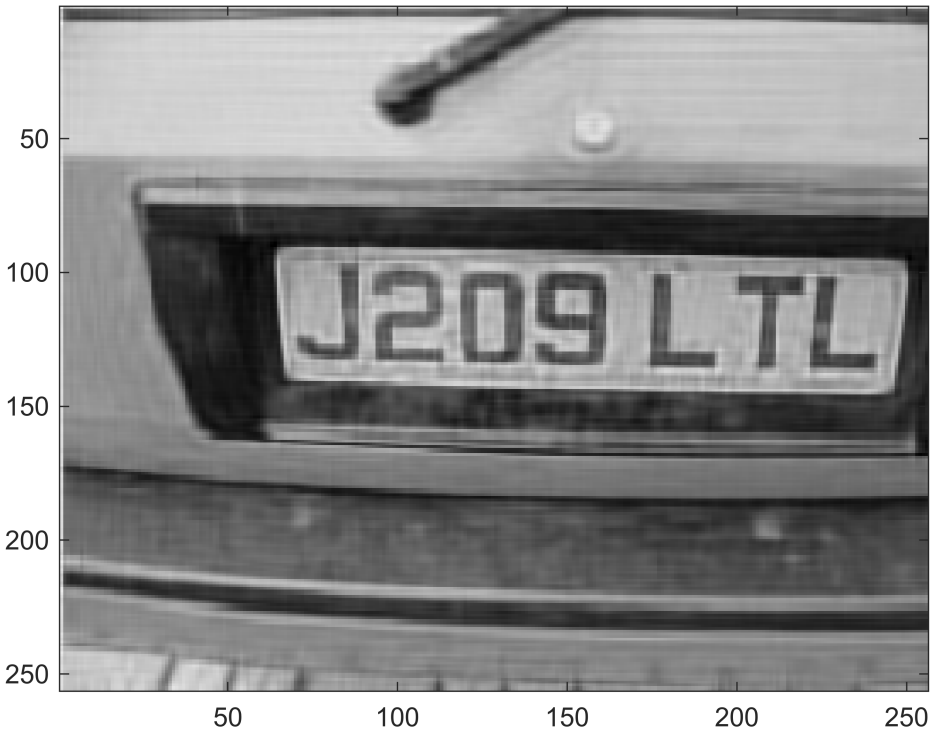
Escogemos el valor de α manualmente, prefiriendo el valor que sirve para que la imagen sea más nítida.

Probamos con $\alpha \in \{10^{-15}, 10^{-5}, 1\}$

```
%Valor de alpha
alpha = 1E-15;
%g_shombrero = U^{T}g
ghat = reshape(UB'*G*UA,n^2,1);
%Inversa de la matriz (Sigma'Sigma + alpha^2 I)
Invalpha =1./(diagS.*diagS + alpha^2 * ones(n^2,1));

%Solución en términos de f_sombrero
fhatmin = Invalpha .* reshape(SB'*reshape(ghat,n,n)*SA,n^2,1);
%Solución en términos de f
fmin = reshape(VB*reshape(fhatmin,n,n)*VA',n^2,1);
%Redimensión de fmin
Fmin = reshape(fmin,n,n);

%Imagen obtenida
imagesc(Fmin), colormap(gray)
```

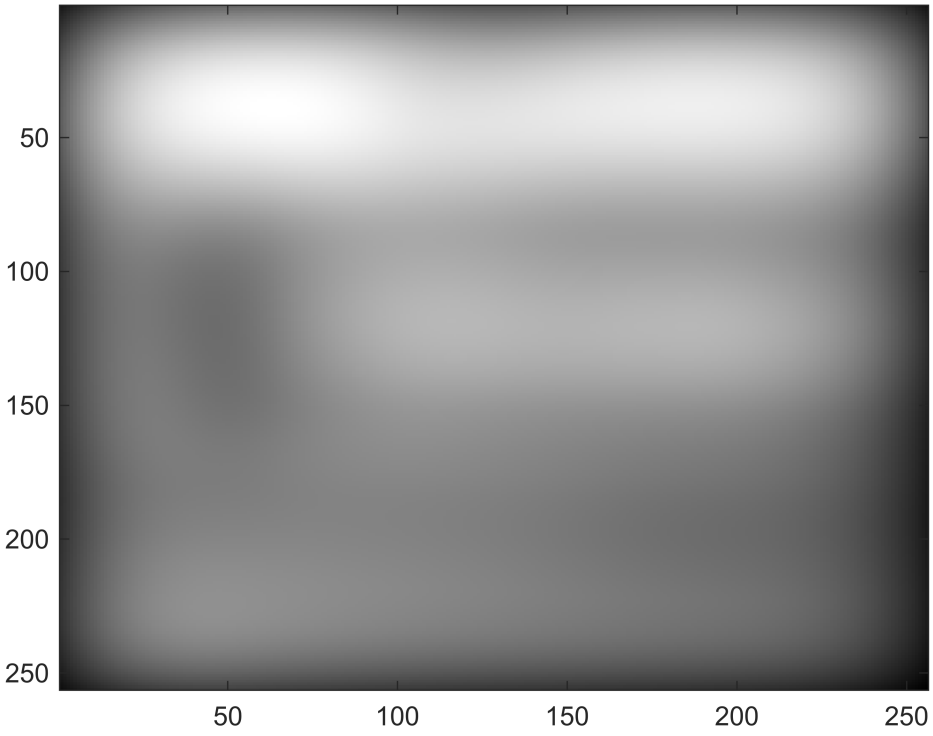

```
%Valor de alpha
alpha = 1E-5;
%g_shombrero = U^{T}g
ghat = reshape(UB'*G*UA,n^2,1);
%Inversa de la matriz (Sigma'Sigma + alpha^2 I)
Invalpha =1./(diagS.*diagS + alpha^2 * ones(n^2,1));
```

```
%Solución en términos de f_sombrero
fhatmin = Invalpha .* reshape(SB'*reshape(ghat,n,n)*SA,n^2,1);
%Solución en términos de f
fmin = reshape(VB*reshape(fhatmin,n,n)*VA',n^2,1);
%Redimensión de fmin
Fmin = reshape(fmin,n,n);

%Imagen obtenida
imagesc(Fmin), colormap(gray)
```




```
%Valor de alpha
alpha = 1;
%g_shombrero = U^{T}g
ghat = reshape(UB'*G*UA,n^2,1);
%Inversa de la matriz (Sigma'Sigma + alpha^2 I)
Invalpha =1./(diagS.*diagS + alpha^2 * ones(n^2,1));

%Solución en términos de f_sombrero
fhatmin = Invalpha .* reshape(SB'*reshape(ghat,n,n)*SA,n^2,1);
%Solución en términos de f
fmin = reshape(VB*reshape(fhatmin,n,n)*VA',n^2,1);
%Redimensión de fmin
Fmin = reshape(fmin,n,n);

%Imagen obtenida
```

```
imagesc(Fmin), colormap(gray)
```



Según las imágenes obtenidas vemos que mientras más pequeño es el valor de α obtenemos una imagen más nitida.

Descomposición en valores singulares truncada

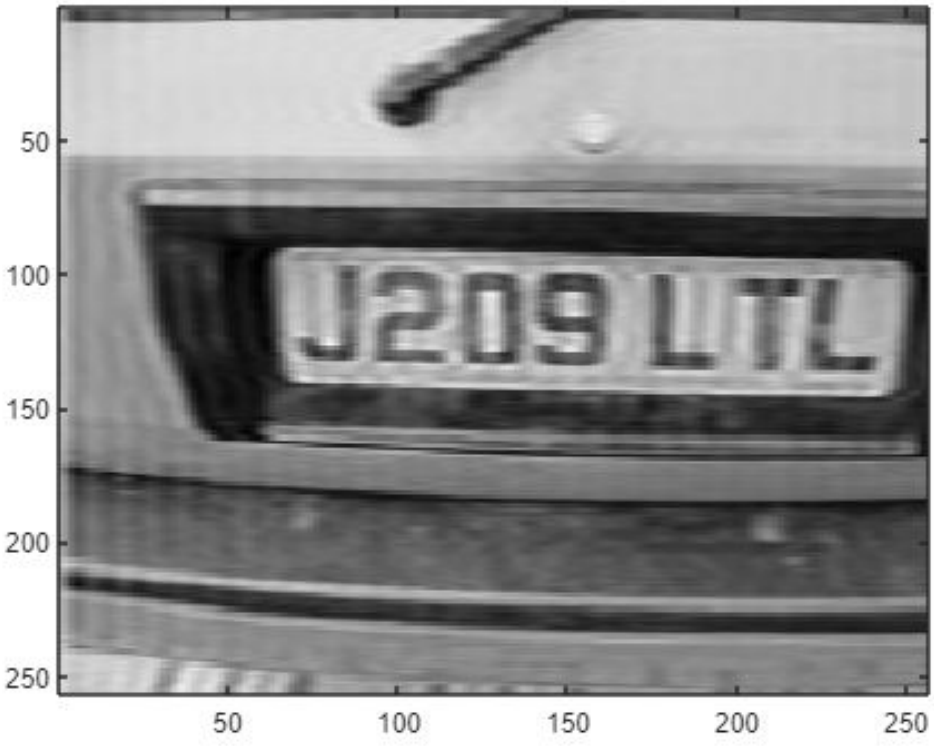
Para hallar la solución por medio de este método, tomamos un valor p en el conjunto $\{1, \dots, n^2\}$ y hacemos la sumatoria $f_{ls} = \sum_{i=1}^{p} \frac{u_i v_i^T}{\sigma_i} g$. A continuación tomamos $p = 15000$.

```
% Valor en donde se trunca la sumatoria
p = 15000;

%Inicialización: Vector solución
fls = zeros(n^2,1);
%Inicialización: Vector canónico i-ésimo
% en R^{n^2}
ei = zeros(n^2,1);
%Inicialización: Columna i-ésima de U
Up = zeros(n^2,1);
%Inicialización: Columna i-ésima de V
Vp = zeros(n^2,1);

%Sumatoria
```

```
for i = 1:p
    ei = zeros(n^2,1);
    ei(i) = 1;
    Ep = reshape(ei,n,n);
    Up = reshape(UB*Ep*UA',n^2,1);
    Vp = reshape(VB*Ep*VA',n^2,1);
    fls = fls + (1/diagS(i))*(Up'*g)*Vp;
end
% Redimensión de la imagen
Fls = reshape(fls,n,n);
```

El truncamiento de la sumatoria hasta $p = 15000$ arroja como resultado la siguiente imagen.

```
imagesc(Fls), colormap(gray)
```



Una manera más eficiente en términos de tiempo pero equivalente en precisión es la siguiente: Sabemos que $f = V\Sigma^{+}U^{T}g$, de esto se sigue,

$$
\begin{aligned}
f &= V\Sigma^{+}U^{T}g \\
&= \sum_{i=1}^{n^2}\frac{v_i u_i^T}{\sigma_i}g = \sum_{i=1}^{n^2}\frac{Ve_i e_i^T U^T}{\sigma_i}g && v_i = Ve_i,\ u_i = Ue_i \\
&= V\sum_{i=1}^{n^2}\frac{e_i e_i^T}{\sigma_i}\hat{g} = V\sum_{i=1}^{n^2}\frac{e_i^T\hat{g}}{\sigma_i}e_i && \hat{g} = U^T g \\
&= V\sum_{i=1}^{n^2}\frac{\hat{g}_i}{\sigma_i}e_i && \hat{g}_i = e_i^T\hat{g} \\
&= V\begin{pmatrix} \hat{g}_1/\sigma_1 \\ \hat{g}_2/\sigma_2 \\ \vdots \\ \hat{g}_{n^2}/\sigma_{n^2} \end{pmatrix}
\end{aligned}
$$

Para hacer el truncamiento de la sumatoria hasta un valor p hacemos el siguiente cálculo basados en lo anterior,

$$
f_{ls} = \sum_{i=1}^{\overset{0\leqslant p\leqslant n^2}{\uparrow\ p}}\frac{v_i u_i^T}{\sigma_i}g = = V\begin{pmatrix} \hat{g}_1/\sigma_1 \\ \vdots \\ \hat{g}_p/\sigma_p \\ 0 \\ \vdots \\ 0 \end{pmatrix}
$$

El vector que aparece en el cálculo de f_{ls} en la anterior ecuación se puede calcular en Matlab como

$$
\hat{g}\ .*\ [1\ \ ./\ \ diagS(1:p,1); zeros(n^2-p,1)]
$$

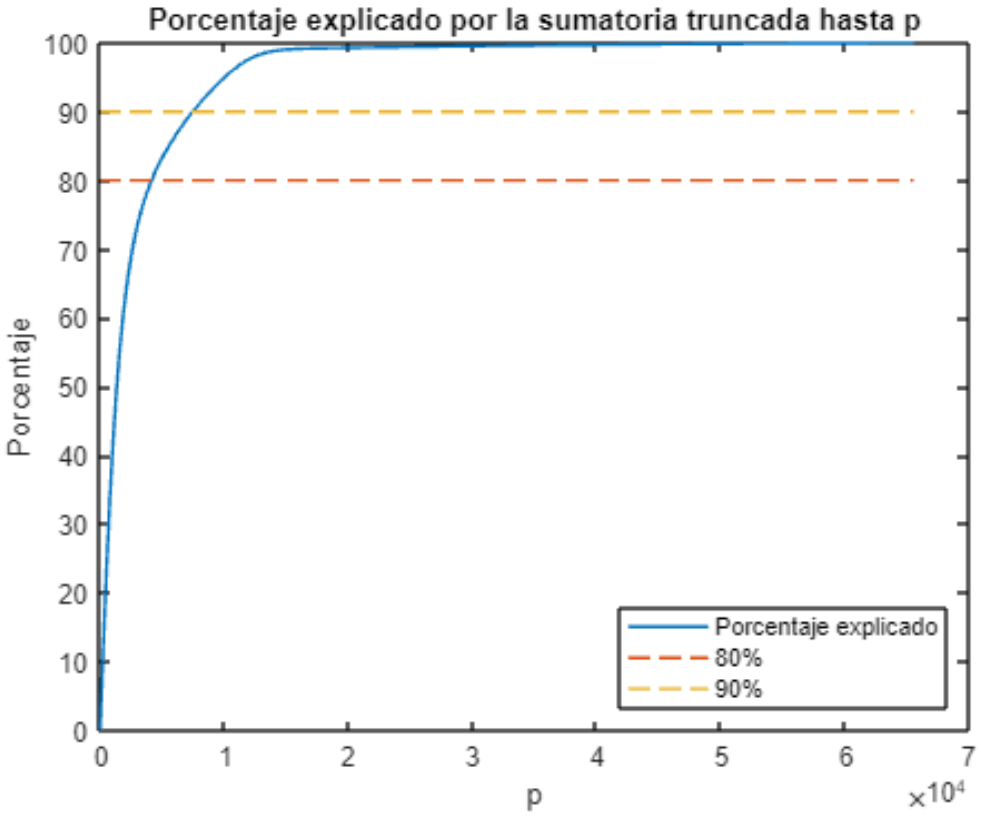
donde $.*$y $./$ representan operaciones *element wise.*

Nótese que este proceso es mucho más sencillo de manejar dado que no es necesario involucrar una estructura repetitiva (*for*) en el código.

Para escoger el valor de p indagamos sobre el porcentaje que representa la sumatoria truncada hasta p sobre la suma de todos los valores singulares de la matriz K.

```
step = 0:n:n^2;
```

```
sizestep = n+1;
Porcentaje_explicado = zeros(sizestep,1);
for i= 1:sizestep
    Porcentaje_explicado(i) = 100*sum(diagS(1:step(i)))/sum(diagS);
end

plot(step, Porcentaje_explicado, step, 80+0*step,'--',step, 90+0*step,'--')
box on
xlabel('p')
ylabel('Porcentaje')
title('Porcentaje explicado por la sumatoria truncada hasta p')
[a,b]=legend('Porcentaje explicado','80%','90%','Location','southeast');
```




```
set(b(1),'Color','b')
set(b(2),'Color','r')
set(b(3),'Color','k')
```

Como se puede evidenciar

- alrededor de $p = 4200$ se alcanza el 80% de la suma total,
- en aproximadamente $p = 7500$ se alcanza el 90% de la suma total, y
- si p es mayor o igual a 15000 se supera el 99% del valor de la sumatoria.

```
% Porcentaje para p=4200
```

```
disp(100*sum(diagS(1:4200))/sum(diagS))
```

```
    80.8603
```

```
% Porcentaje para p=7500
disp(100*sum(diagS(1:7500))/sum(diagS))
```

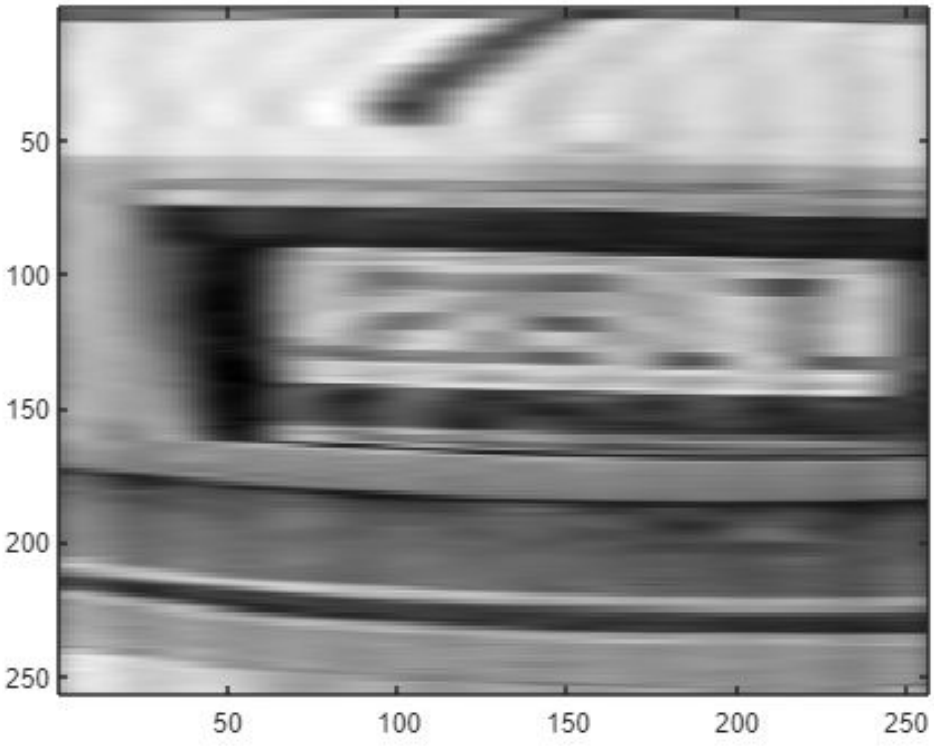
```
    90.4838
```

```
% Porcentaje para p=15000
disp(100*sum(diagS(1:15000))/sum(diagS))
```

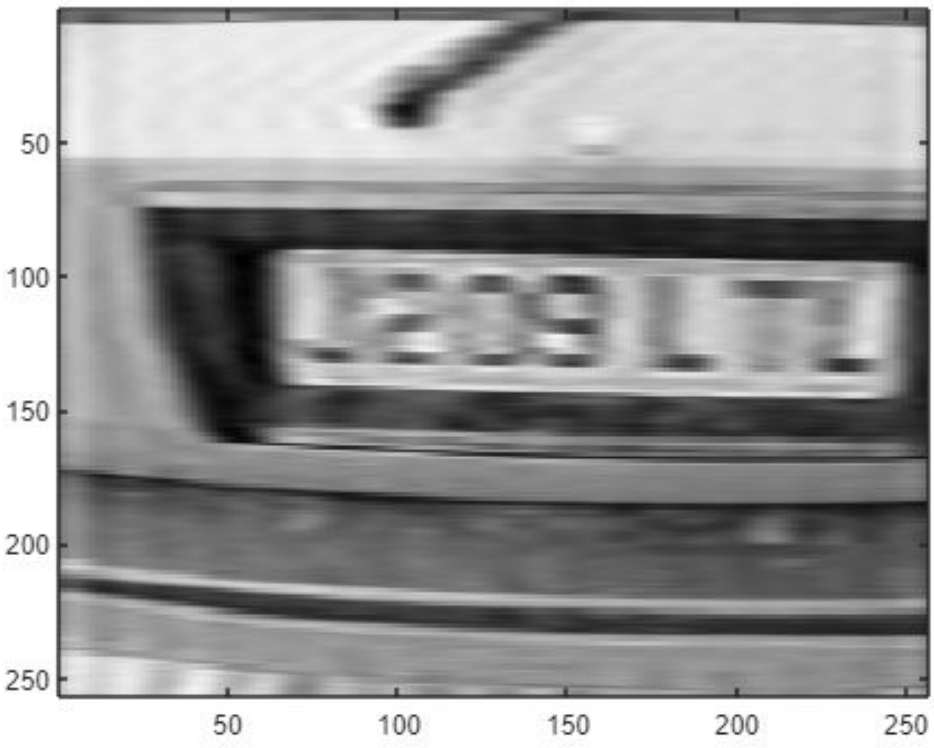
```
    99.0726
```

Para cada uno de estos valores, graficamos la imagen obtenida

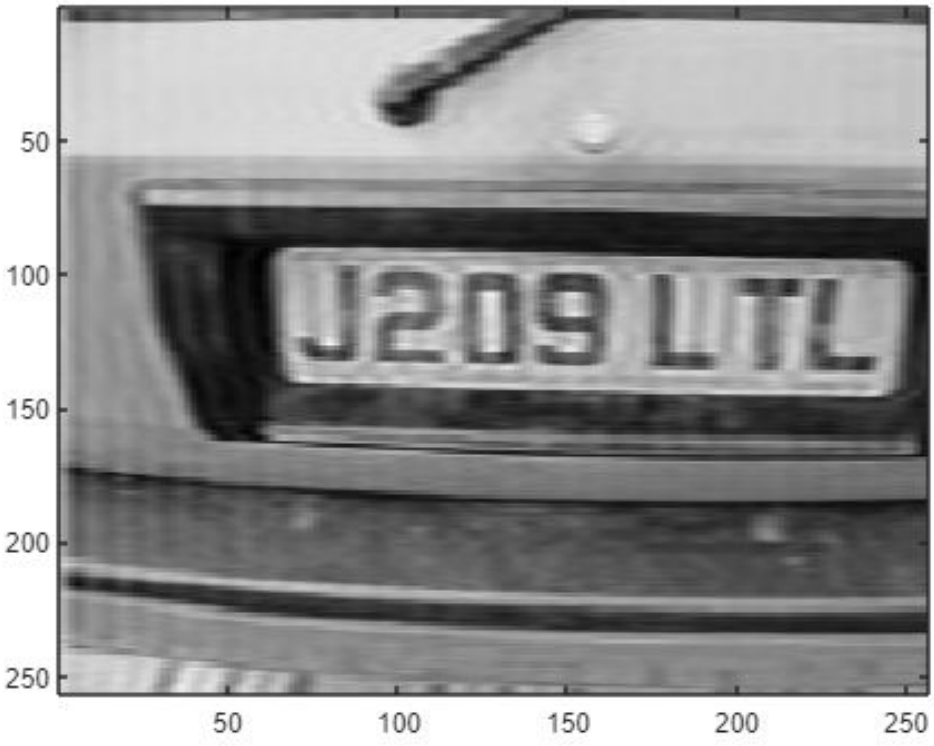
```
%Con un 80% de porcentaje explicado
p = 4200;

% Vector truncado
wp = ghat.* [1./diagS(1:p,1) ; zeros(n^2-p,1)];
% Solución de mínimos cuadrados
fls0 = reshape(VB*reshape(wp,n,n)*VA',n^2,1);
%Redimensión de fls0
Fls0 = reshape(fls0,n,n);

%Imagen obtenida
imagesc(Fls0), colormap(gray)
```



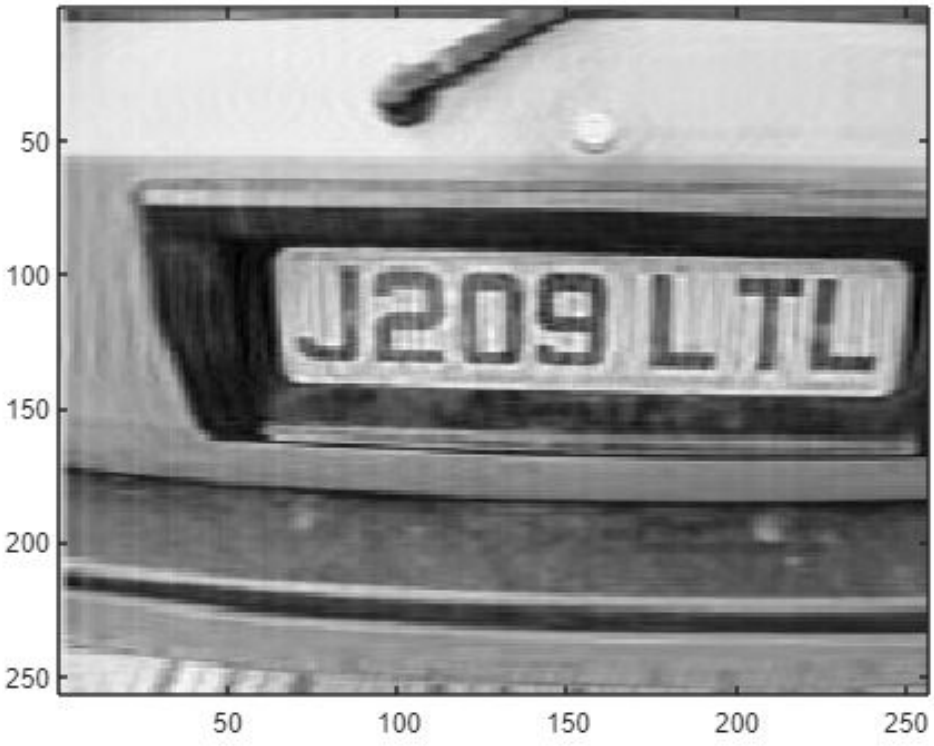

```
%Con un 90% de porcentaje explicado
p = 7500;

% Vector truncado
wp = ghat.* [1./diagS(1:p,1) ; zeros(n^2-p,1)];
% Solución de mínimos cuadrados
fls0 = reshape(VB*reshape(wp,n,n)*VA',n^2,1);
%Redimensión de fls0
Fls0 = reshape(fls0,n,n);

%Imagen obtenida
imagesc(Fls0), colormap(gray)
```




```
%Con un 99% de porcentaje explicado
p = 15000;

% Vector truncado
wp = ghat.* [1./diagS(1:p,1) ; zeros(n^2-p,1)];
% Solución de mínimos cuadrados
fls0 = reshape(VB*reshape(wp,n,n)*VA',n^2,1);
%Redimensión de fls0
Fls0 = reshape(fls0,n,n);

%Imagen obtenida
imagesc(Fls0), colormap(gray)
```




```
%Con un 99.45% de porcentaje explicado
p = 25000;

% Vector truncado
wp = ghat.* [1./diagS(1:p,1) ; zeros(n^2-p,1)];
% Solución de mínimos cuadrados
fls0 = reshape(VB*reshape(wp,n,n)*VA',n^2,1);
%Redimensión de fls0
Fls0 = reshape(fls0,n,n);

%Imagen obtenida
imagesc(Fls0), colormap(gray)
```



Nótese que mientras se aumenta el valor de truncamiento p la imagen obtenida es más nítida. Cabe también destacar que entre $p = 15000$ y $p = 25000$ no hay mucha diferencia del resultado.

```
%Cambio porcentual de la solución cuando esta p=15000 y pasa a p=25000
100*norm(reshape(VB*reshape(ghat.* [1./diagS(1:25000,1) ;
zeros(n^2-25000,1)],n,n)*VA',n^2,1)-reshape(VB*reshape(ghat.*
[1./diagS(1:15000,1) ; zeros(n^2-15000,1)],n,n)*VA',n^2,1))/
norm(reshape(VB*reshape(ghat.* [1./diagS(1:15000,1) ;
zeros(n^2-15000,1)],n,n)*VA',n^2,1))
```

```
ans = 4.0802
```

Para finalizar, por ambos métodos concluimos que el texto en la imagen es **J209 LTL**. A simple vista el método de regularización arroja una imagen 'mejor' en términos de nitidez en contraste con el método SVD truncado. A lo largo de esta parte de programación se tomaron los valores de α y de p de manera manual, por ende, para hacer una comparación completamente justificada entre ambos métodos es necesario tener un critero preciso sobre qué tan 'buena' es la imagen obtenida dependiendo de la sensibilidad de los parámetros de cada método.