IMT2111 - Álgebra Lineal Numérica - 1er Semestre 2019
Profesor: Manuel Sánchez Uribe

Tarea 1

Nombre: Maximiliano Rivera
marivera3@uc.cl
9 de abril de 2019

Matlab Code

En esta sección se presentan los códigos implementados para esta tarea.

QR por el método de Gram-Schmidt clásico

```
% Created by Maximiliano Rivera, based on Numerical Linear Algebra by
% L.N. Trefetehn.

function [ Q, R ] = QRbyGSC(A)
% This function is for decompose a matrix A in matrices Q and R, where Q
% has orthonormal columns and R is upper triangular, such that A = QR. This
   is possible
% using the Gran-Schmidt Clasical algorithm

[m, n] = size(A);
Q = zeros(m ,n);
R = zeros(n, n);

for j=1:n
    v = A(:, j);
    r = Q'*v;
    v = v - Q*r;
    rjj = norm(v);
    v = v/rjj;
    Q(:, j) = v;
    R(1:j, j) = [r(1:j-1); rjj];


end
return

end
```

QR por el método de Gram-Schmidt modificado

```
% Created by Maximiliano Rivera, based on Numerical Linear Algebra by
% L.N. Trefetehn.

function [ Q, R ] = QRbyGSM(A)
% This function does QR decomposition with Gram–Schmidt modify algorithm

[m, n] = size(A);
Q = zeros(m ,n);
R = zeros(n, n);
for i=1:n
    vi = A(:, i);
    rii = norm(vi);
    qi = vi/rii;
    Q(:, i) = qi;
%     for j=i+1:n
%         vj = A(:, j);
%         rij = qi'*vj;
%         A(:, j) = vj - rij*qi;
%     end
    Vj = A(:, i+1:end);
    rij = qi'*Vj;
    A(:, i+1:end) = Vj - qi*rij;

    %We can construct R from the rii and rij previously calculated
    R(i:n, i) = [rii; rij'];
end
R = R';
return

end
```

QR por el método de Householder

```
% Created by Maximiliano Rivera, based on Numerical Linear Algebra by
% L.N. Trefetehn.

function [ Q, R ] = QRbyHouseholder(A)

[m, n] = size(A);
Q = eye(m);
for k = 1:n

    x = A(k:m, k);
    e1 = zeros(length(x), 1);
    e1(1) = 1;
    v = sign(x(1))*norm(x)*e1 + x;
    v = v/norm(v);
    F = eye(m-k+1, m-k+1) - 2*v*(v');

    A(k:m, k:n) = F*A(k:m, k:n);

    Qk = eye(m, m);
    Qk(end-(m-k):end, end-(m-k):end) = F;
    Q = Qk*Q; % forming the matrix Q' = Qk...Q2Q1
end
Q = Q';
R = A;
return
end
```

Pregunta 2

Maximiliano Rivera

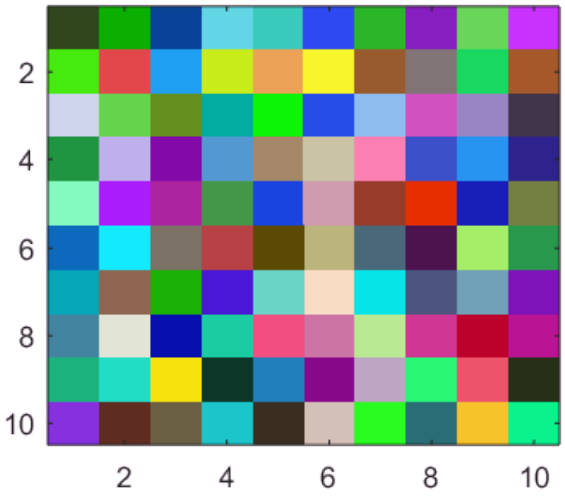
Problema 2.1

1. El costo de guardar cada elemento de A es de $m \cdot n$
2. El costo de guardar cada término de la SVD, es obtener todos los coeficientes de la sumatoria $\sum_{i=1}^{n} \sigma_i u_i v_i^*$, dando un total de $(m + n + 1) \cdot n$
3. El costo de guardar un término de la SVD es de $m + n + 1$, debido a que, con 1 término tenemos: $\sum_{i=1}^{1} \sigma_i u_i v_i^*$, con u_i de largo m, v_i largo n, y σ_i es un escalar.
4. El costo de guardar k elementos de la SVD es igual a $(m + n + 1) \cdot k$, por ende, el k para que se almacene la misma cantidad de información es $k = \frac{m \cdot n}{m + n + 1}$. Esto significa que, utilizando ese k, la descomposicion SVD presenta la misma información que la matriz previa.
5. El error relativo asociado a la imagen con k-términos es $\sum_{i=k+1}^{n} \sigma_i u_i v_i^*$

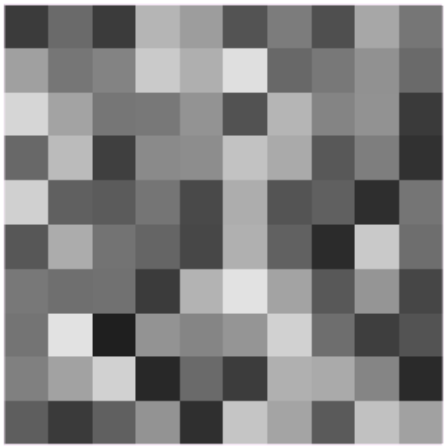
Problema 2.2

a) Se crea una matriz $C \in C^{10x10x3}$, para luego encontrar la escala de grises la imagen, quedando $C_{gray} \in C^{10x10}$

```
C = rand(10, 10, 3);
image(C);
```

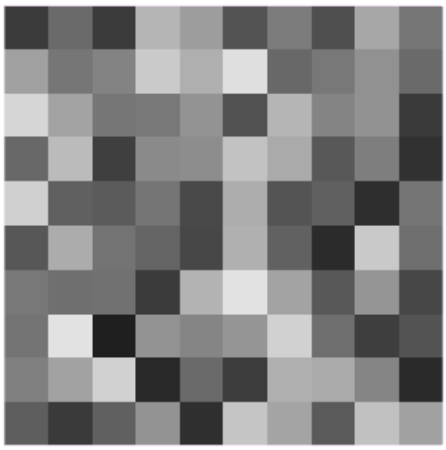


```
Cgray = rgb2gray(C);
% Cgray = rand(10, 10);
% imshow(Cgray 'CDataMapping','scaled'); colorbar;
% fig = figure();
% ax = axes(fig);
magnification = 6000;
imshow(Cgray, 'InitialMagnification', magnification);
```



b) Se descompone la matriz C_{gray} según la SVD.

```
[U,S,V] = svd(Cgray);
% image(U*S*V', 'CDataMapping','scaled'); colorbar;
imshow(U*S*V', 'InitialMagnification', magnification);
```

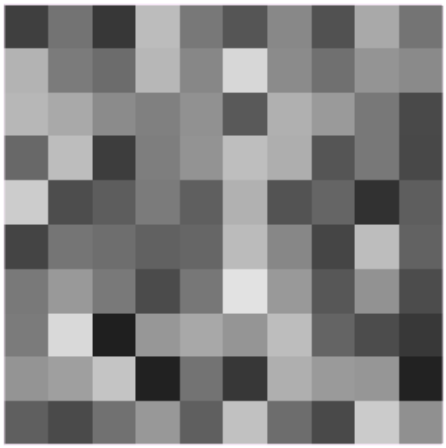


c) Se estima el k y luego se calcula el SVD para los primeros k términos.

```
[m, n] = size(Cgray);
k = round(m*n/(m + n+ 1))
```

```
k = 5
```

```
Sk = S(1:k, 1:k);
Uk = U(:, 1:k);
Vk = V(:, 1:k);
Cgray2 = Uk*Sk*Vk';
% image(Cgray2, 'CDataMapping','scaled'); colorbar;
imshow(Cgray2, 'InitialMagnification', magnification);
```

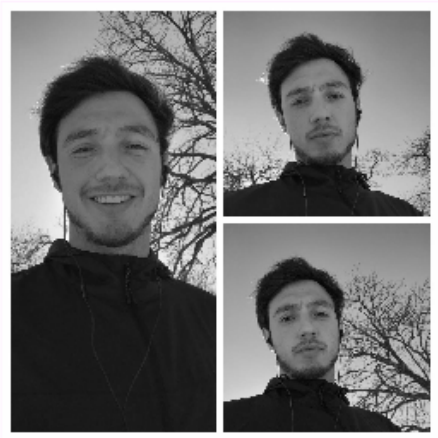


En esta última figura podemos notar que al utilizar una cantidad de valores singulares reducidos, la matriz resultante es levemente diferente a la matriz original. Esto se debe a la aproximaxión previamente calculada.

Problema 2.3

```
Max = im2double(imread('max.bmp'));
imshow(Max)
```

```
magnification2 = 100;
 Igray = rgb2gray(Max);
imshow(Igray, 'InitialMagnification', magnification2);
```



b) Descomposición SVD:

```
% Igray_double = double(Igray);
[UI,SI,VI] = svd(Igray);
% image(U*S*V', 'CDataMapping','scaled'); colorbar;
imshow(UI*SI*VI', []);
```

c)

```
[m, n] = size(Igray);
k = round(m*n/(m + n+ 1));
% k=300;
SIk = SI(1:k, 1:k);
UIk = UI(:, 1:k);
VIk = VI(:, 1:k);
Igray2 = UIk*SIk*VIk';
% image(Cgray2, 'CDataMapping','scaled'); colorbar;
imshow(Igray2, []);
```

Problema 2.4

```
Max1 = Max(:, :, 1);
Max2 = Max(:, :, 2);
Max3 = Max(:, :, 3);
% Igray_double = double(Igray);
[U1,S1,V1] = svd(Max1);
[U2,S2,V2] = svd(Max2);
[U3,S3,V3] = svd(Max3);
% image(U*S*V', 'CDataMapping','scaled'); colorbar;
[m, n] = size(Max1);
k = round(m*n/(m + n+ 1));
S1k = S1(1:k, 1:k);
U1k = U1(:, 1:k);
V1k = V1(:, 1:k);
S2k = S2(1:k, 1:k);
U2k = U2(:, 1:k);
V2k = V2(:, 1:k);
S3k = S3(1:k, 1:k);
U3k = U3(:, 1:k);
V3k = V3(:, 1:k);
NewMax = zeros(m, n, 3);
NewMax(:,:,1) = U1k*S1k*V1k';
NewMax(:,:,2) = U2k*S2k*V2k';
NewMax(:,:,3) = U3k*S3k*V3k';
imshow(NewMax)
```

Pregunta 3

Maximiliano Rivera

Los códigos utilizados están en la siguiente plataforma:

https://github.com/Marivera3/Algebra-Lineal-Numerica

Esto fueron implementados por mi, basado en los apuntes del libro texto guía L.N.Trefetehn.

Problema 3.1

La matriz a considerar es la siguiente

```
A = [1 2 3; 4 5 6; 7 8 7; 4 2 3; 4 2 2];
cond(A)
```

```
ans = 12.4508
```

```
[Q, R] = qr(A);
```

Para calcular el error se utiliza la norma 2. El error asociado a esta descomposición es:

```
e = norm(Q*R - A)
```

```
e = 2.8383e-15
```

```
rii = diag(R);
```

Luego, calculamos su descomposición QR con los siguiente algoritmos implementados:

- Factorización QR por método de Gram-Schmidt clásico.

```
[Q1, R1] = QRbyGSC(A);
```

El error asociado es:

```
e1 = norm(Q1*R1 - A)
```

```
e1 = 1.1802e-15
```

```
rii1 = diag(R1);
```

- Factorización QR por método de Gram-Schmidt modificado.

```
[Q2, R2] = QRbyGSM(A);
```

El error asociado es:

```
e2 = norm(Q2*R2 - A)
```

```
e2 = 1.1935e-15
```

```
rii2 = diag(R2);
```

- Factorización QR por método de Householder.

```
[Q3, R3] = QRbyHouseholder(A);
```

El error asociado es:

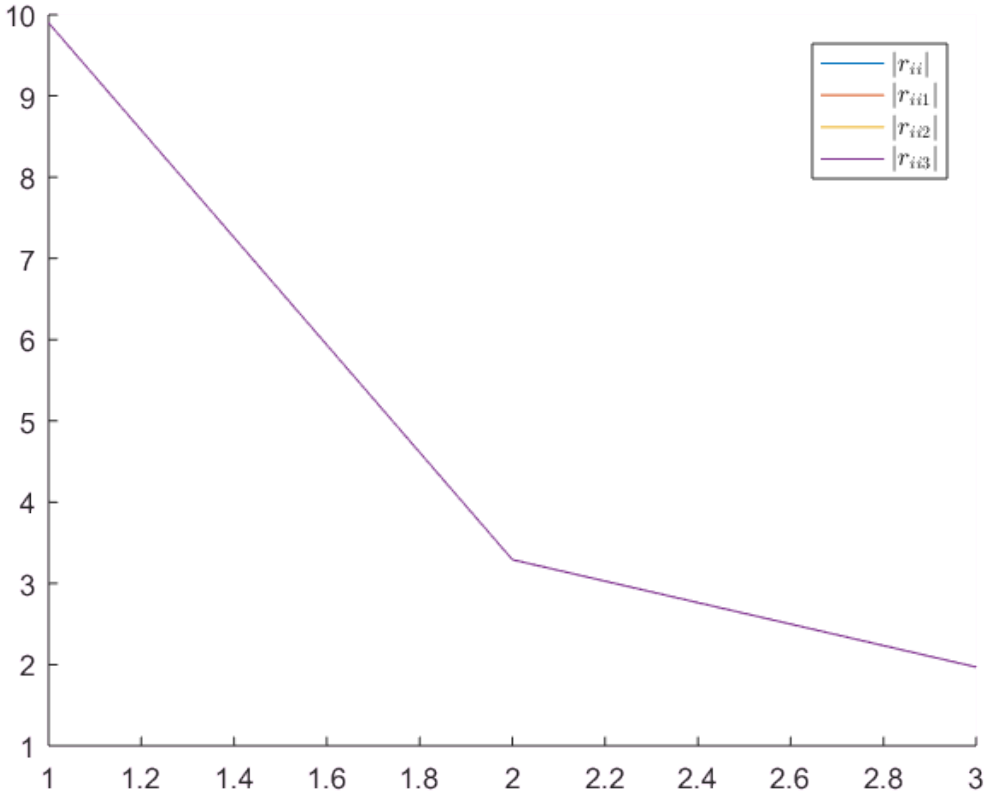
```
e3 = norm(Q3*R3 - A)
```

```
e3 = 2.8450e-15
```

```
rii3 = diag(R3);
```

El gráfico de las componentes diagonales de la matriz triagular superior es el siguiente:

```
fig = figure();
ax = axes(fig);
set(ax,'nextplot','add');
o = plot(ax, abs(rii));
o1 = plot(ax, abs(rii1));
o2 = plot(ax, abs(rii2));
o3 = plot(ax, abs(rii3));
leg = legend([o,o1, o2, o3],'$|r_{ii}|$','$|r_{ii1}|$', '$|r_{ii2}|$', '$|r_{ii3}|$');
set(leg,'interpreter','latex');
```



Donde los valores de r_{iiN}, con $N = 0, 1, 2, 3$ representan la diagonal de la matriz R respectiva. Del gráfico se concluye que estos valores son idénticos a excepto de un signo, esto se debe a que es posible multiplicar por un -1 en ambas matrices, por lo que, como también los errores relativos son pequeños, los métodos utilizados son equivalentes. Notemos que, por sobre todo esto, la matriz está bien condicionada.

Pregunta 3.2

Ahora se utiliza una matriz de Hilbert con $n = 20$. Esta matriz es conocida por ser mal condicionada.

```
H = hilb(20);
cond(H)
```

```
ans = 2.1065e+18
```

```
[Q, R] = qr(H);
```

Para calcular el error se utiliza la norma 2. El error asociado a esta descomposición es:

```
e = norm(Q*R - H)
```

```
e = 3.5079e-16
```

```
rii = diag(R);
```

Luego, calculamos su descomposición QR con los siguiente algoritmos implementados:

- Factorización QR por método de Gram-Schmidt clásico.

```
[Q1, R1] = QRbyGSC(H);
```

El error asociado es:

```
e1 = norm(Q1*R1 - H)
```

```
e1 = 3.2164e-17
```

```
rii1 = diag(R1);
```

- Factorización QR por método de Gram-Schmidt modificado.

```
[Q2, R2] = QRbyGSM(H);
```

El error asociado es:

```
e2 = norm(Q2*R2 - H)
```

```
e2 = 6.6324e-17
```

```
rii2 = diag(R2);
```

- Factorización QR por método de Householder.

```
[Q3, R3] = QRbyHouseholder(A);
```

El error asociado es:

```
e3 = norm(Q3*R3 - A)
```

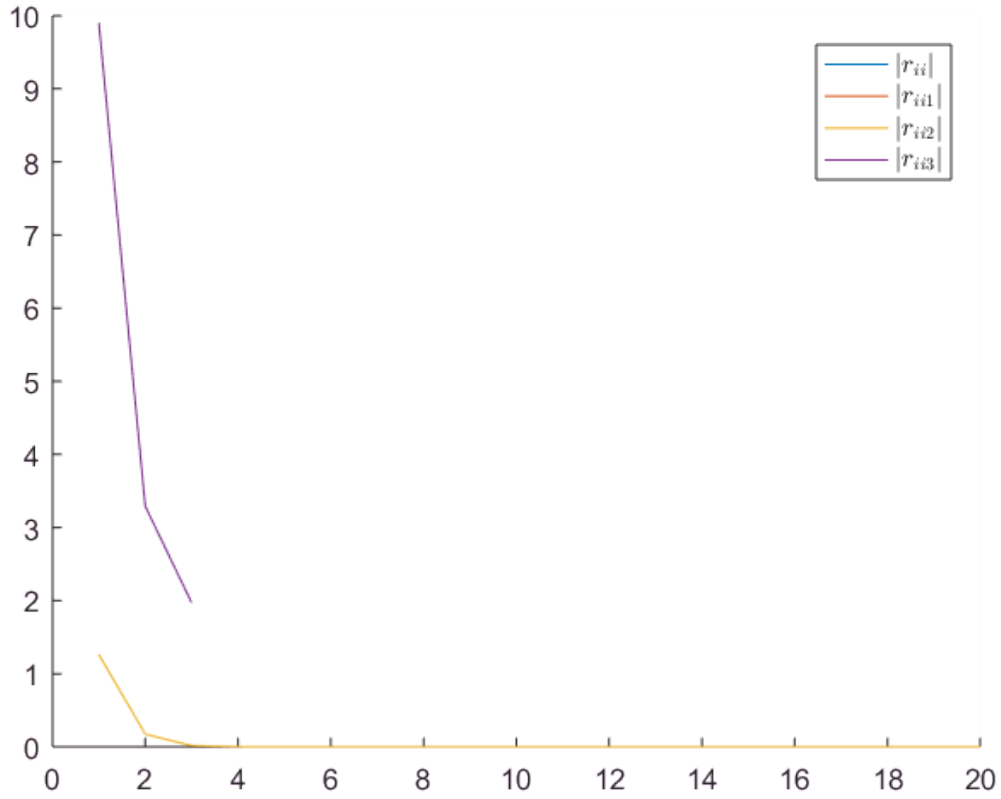
```
e3 = 2.8450e-15
```

```
rii3 = diag(R3);
```

El gráfico de las componentes diagonales de la matriz triagular superior es el siguiente:

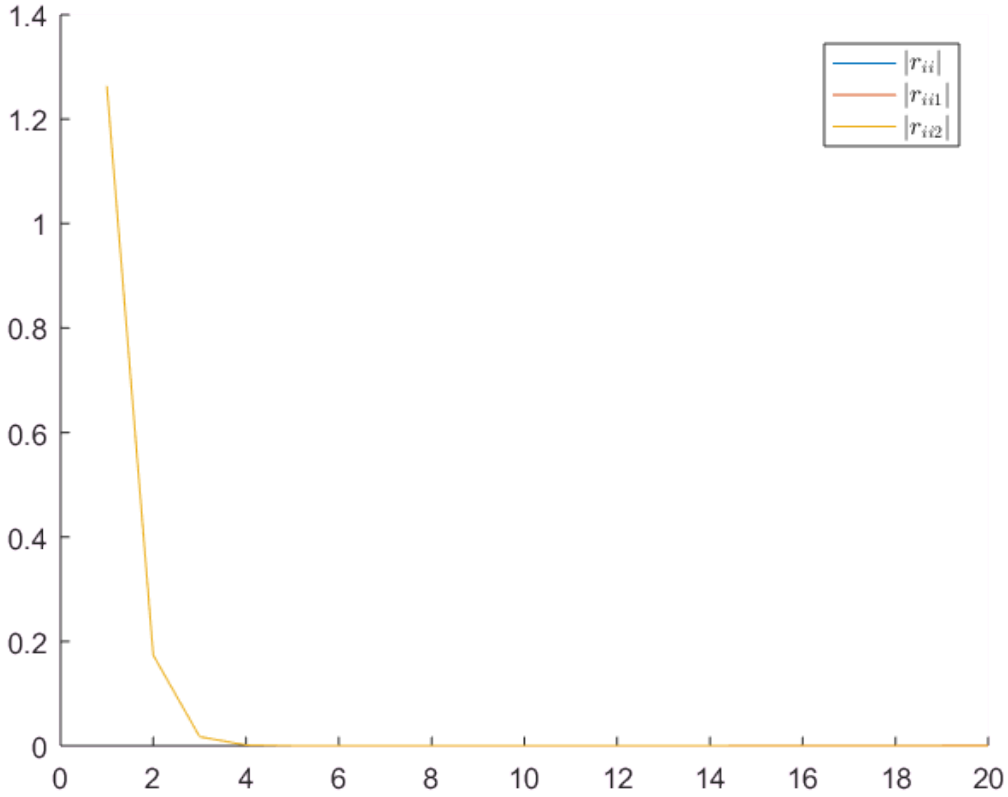
```
fig = figure();
ax = axes(fig);
set(ax,'nextplot','add');
```

```
o = plot(ax, abs(rii));
o1 = plot(ax, abs(rii1));
o2 = plot(ax, abs(rii2));
o3 = plot(ax, abs(rii3));
leg = legend([o,o1, o2, o3],'$|r_{ii}|$','$|r_{ii1}|$', '$|r_{ii2}|$', '$|r_{ii3}|$');
set(leg,'interpreter','latex');
```



Del gráfico se puede ver que los coeficientes de R obtenido por Householder difiere del resto. Se grafican los demas r_{ii}

```
fig2 = figure();
ax2 = axes(fig2);
set(ax2,'nextplot','add');
o = plot(ax2, abs(rii));
o1 = plot(ax2, abs(rii1));
o2 = plot(ax2, abs(rii2));
leg = legend([o,o1, o2],'$|r_{ii}|$','$|r_{ii1}|$', '$|r_{ii2}|$');
set(leg,'interpreter','latex');
```

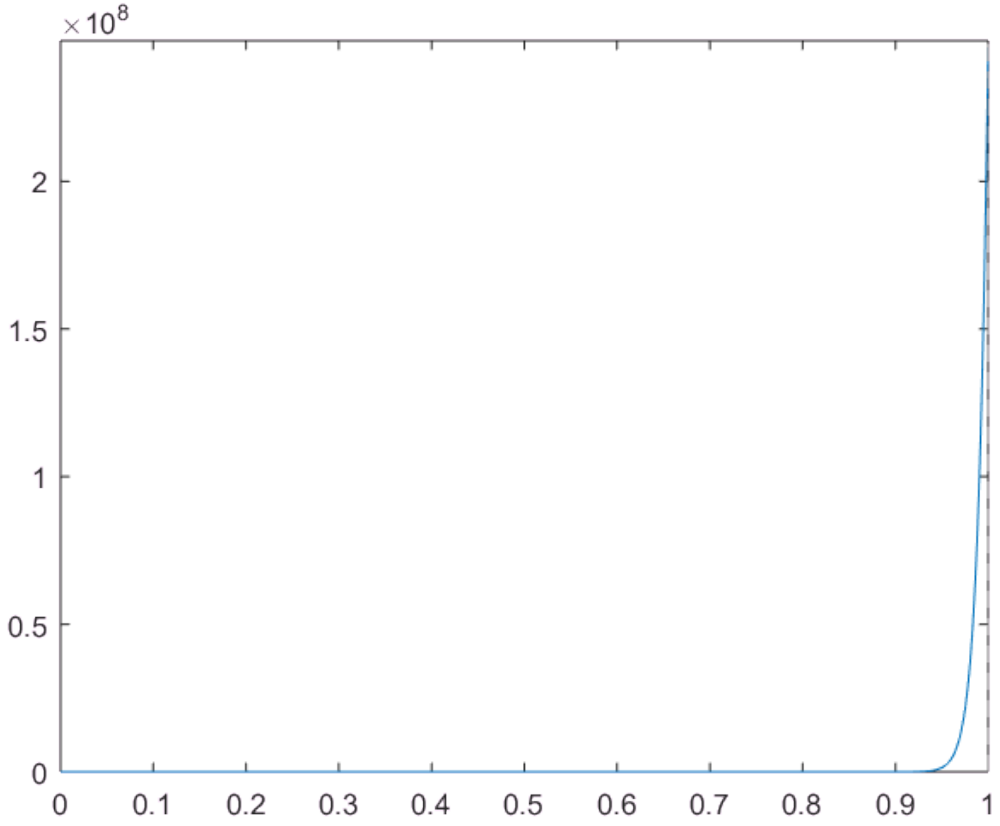


De este gráfico, podemos notar que el método de Gram-Schmidt es más estable que el método dado por Householder. Esto es facil de notar cuando se utiliza una matriz mal condicionada.

Pregunta 4

Maximiliano Rivera

```
f = @(t) e.^(sin(4*t))/2006.787453080206;
fplot(f, [0, 1])
```




```
m = 100; % Puntos
t = linspace(0,1, m);
b = f(t)';
% n = 15;
% x = linspace(0, 1, n);
```

- Método utilizando Ecuaciones Normales

Se utiliza la matriz de Vandermonde en el sistema $Ax = b$, el cual hay que minimizar.

Con x de la forma $x = (a_0, a_1, \dots, a_{14})^T$, A es la matriz de Vandermonde.

```
A = fliplr(vander(t));
A = A(:, 1:15);
```

```
Aprima = A'*A;
bprima = A'*b;
[R] = chol(Aprima)
```

```
Error using chol
Matrix must be positive definite.
```

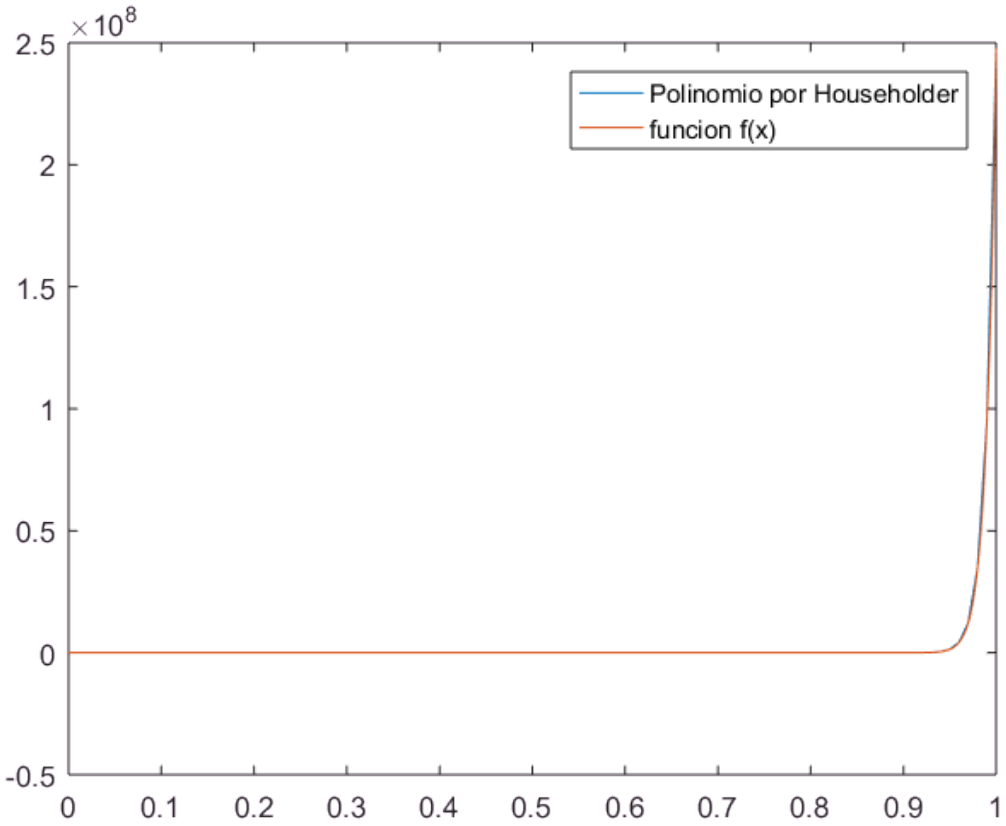
Como A^*A no es positiva definida, no presenta decomposición de Cholesky.

- Householder

```
A = vander(t);
[Q, R] = qr(A);
bprima = Q'*b;
coef = R\bprima;
```

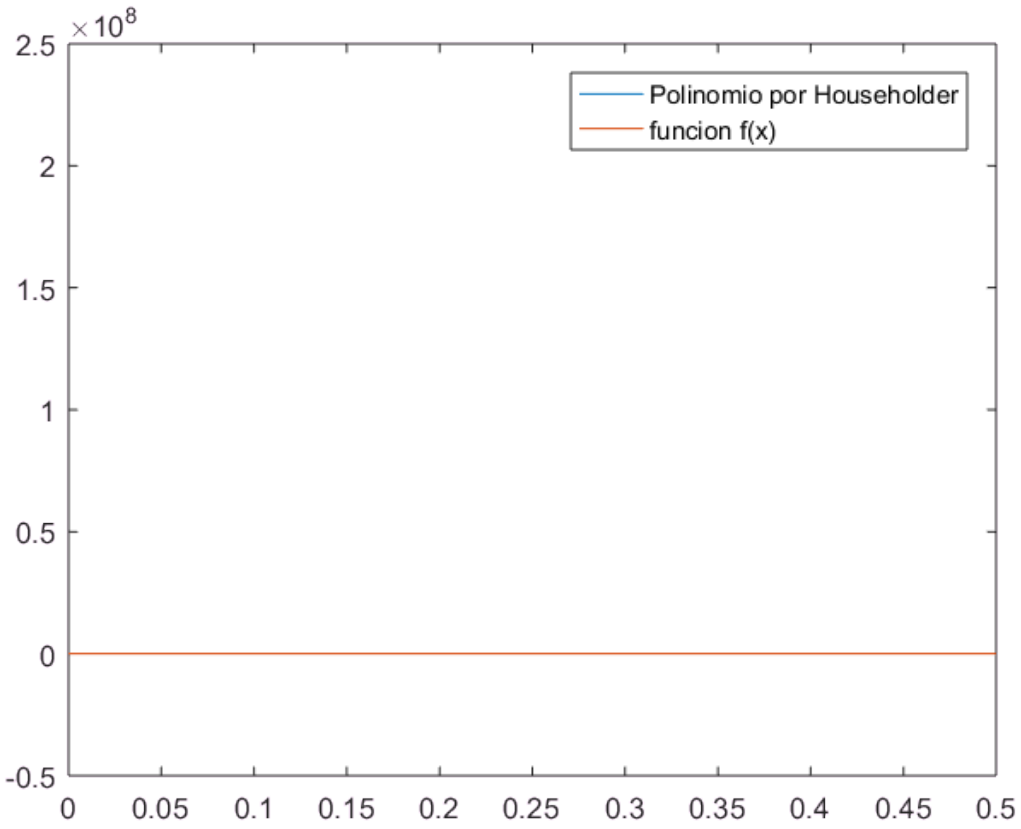
```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND
=  4.112408e-21.
```

```
coef_15_Householder = coef(1);
pol = polyval(coef, t);
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por Householder', 'funcion f(x)');
hold off;
```

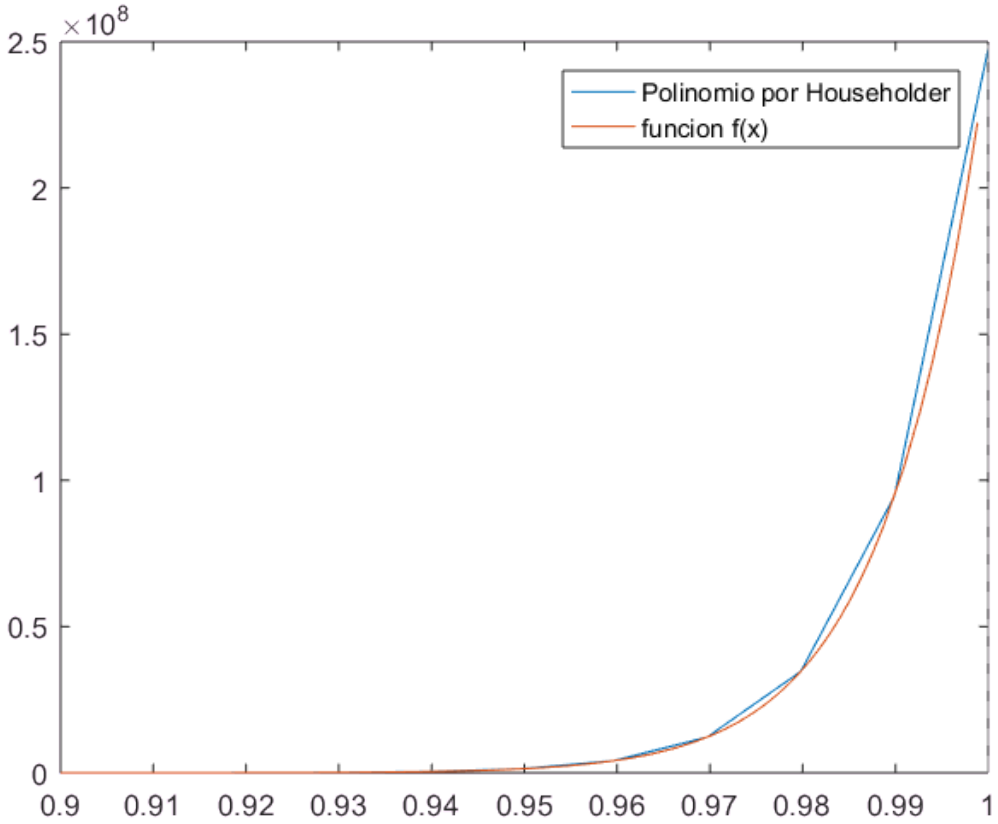


Podemos notar que el ajuste es preciso para $x < 0.9$ aproximadamente, luego a esto, existe una diferencia considerable.

```
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por Householder', 'funcion f(x)');
xlim([0 0.5])
hold off;
```




```
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por Householder', 'funcion f(x)');
xlim([0.9 1])
hold off;
```



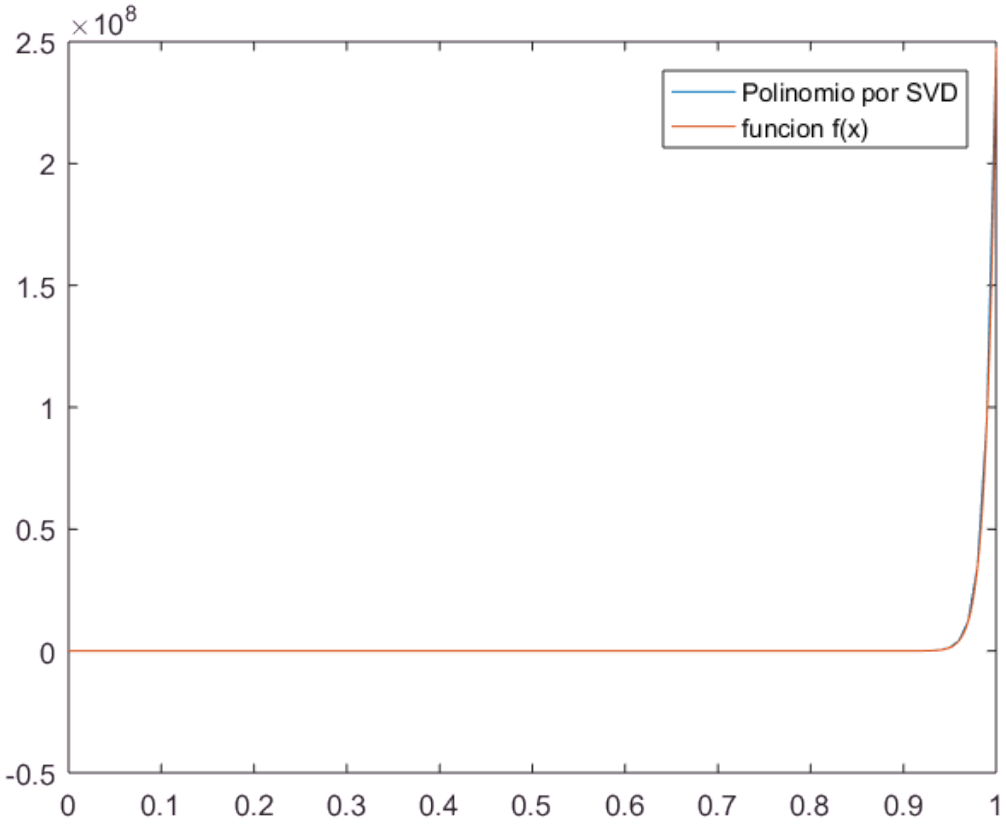
Esto se debe al gran crecimiento que presenta la función $f(x)$.

- SVD

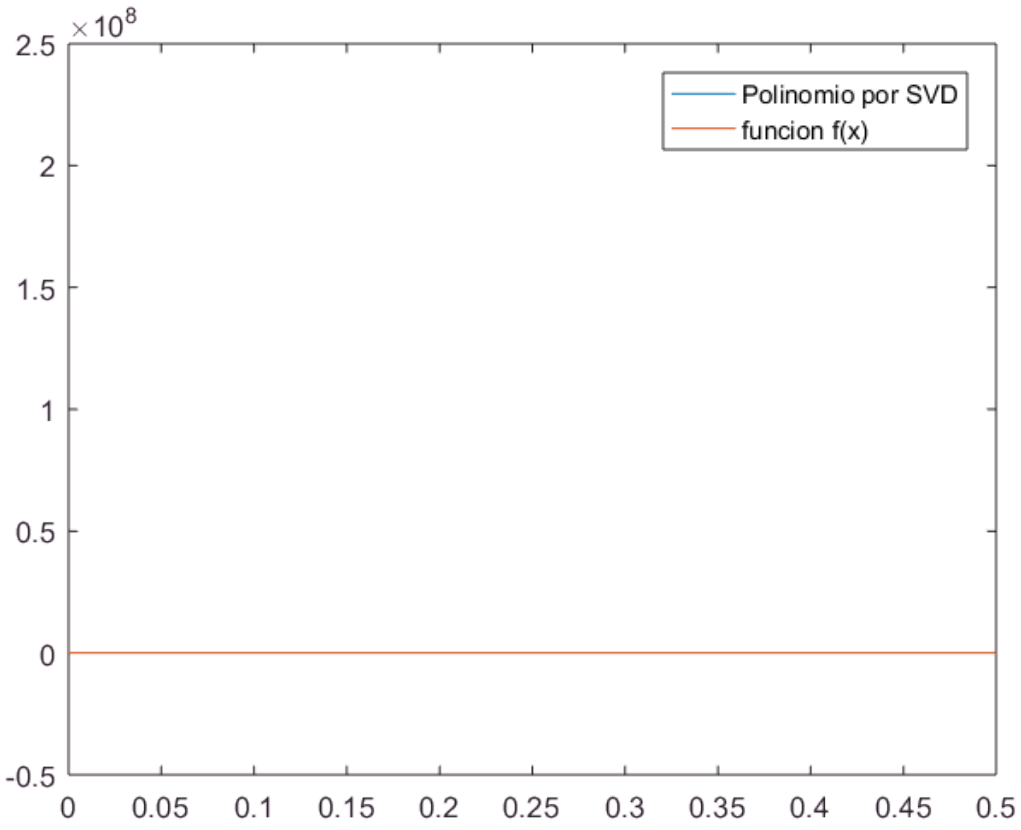
```
A = vander(t);
[U,S,V] = svd(A);
bprima = U'*b;
w = S\bprima;
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND
=  4.881463e-17.
```

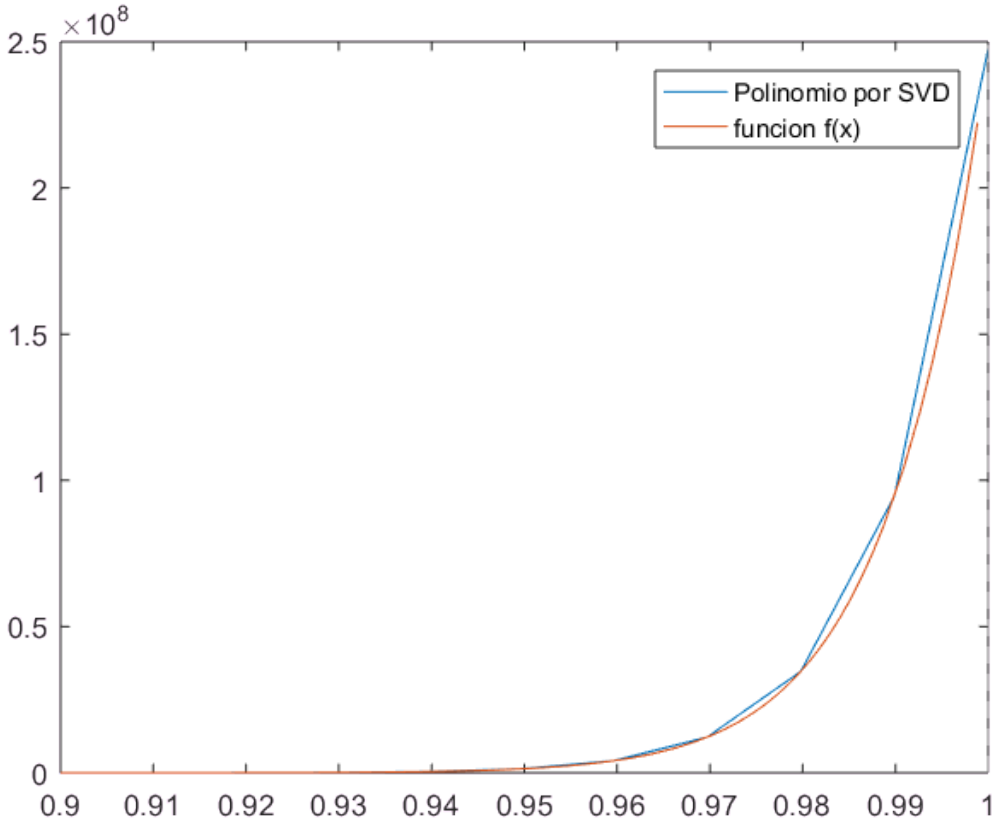
```
coef = V*w;
coef_15_SVD = coef(1);
pol = polyval(coef, t);
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por SVD', 'funcion f(x)');
hold off;
```




```
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por SVD', 'funcion f(x)');
xlim([0 0.5])
hold off;
```




```
plot(t, pol)
hold on;
fplot(f, [0, 1])
legend('Polinomio por SVD', 'funcion f(x)');
xlim([0.9 1])
hold off;
```



El comportamiento presente al utilizar descomposición SVD es similiar al por Householder.

Luego, se comparan los coeficientes 15 obtenidos.

```
coef_15_Householder
```

```
coef_15_Householder = 1.1592e+18
```

```
coef_15_SVD
```

```
coef_15_SVD = -4.8725e+12
```

Como se puede ver, los coeficientes 15 son considerablemente distintos, esto es debido a que la matriz de Vandermonde es mal condicionada, provocando inestabilidad en los métodos.