Table of Contents

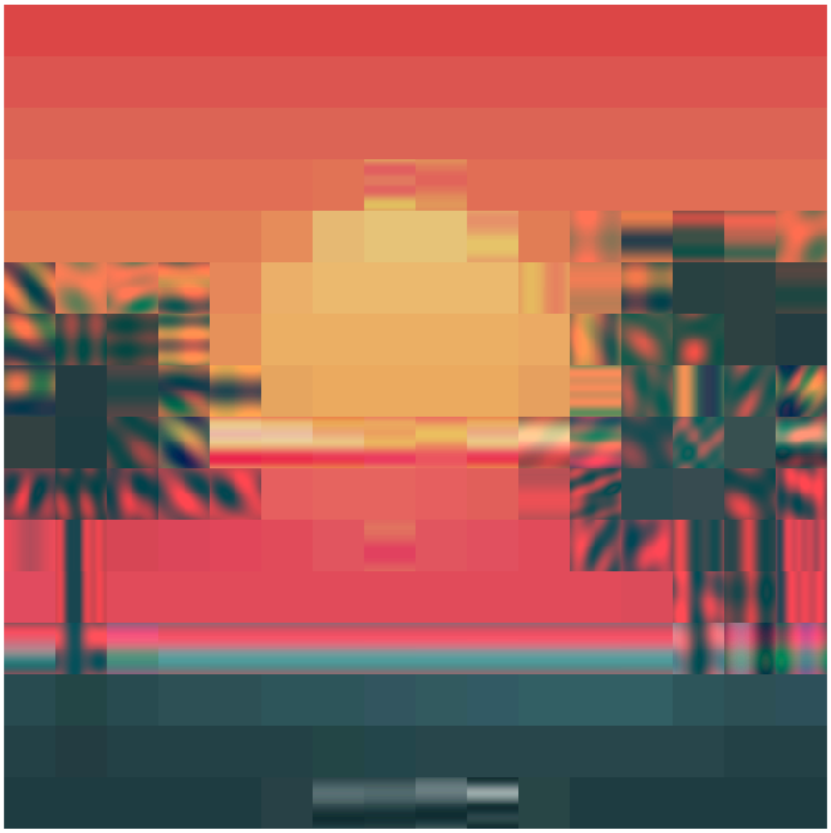

```
% Ronan McNally
% Digital Signal Processing ELEN 4810
% Final Project: Image Compression
```

Images: Sized to be divisible by 32

```
clear
clc

CS = imread("cartoonsunset.jpg");
CS = imresize(CS, [576 576]);
RS = imread("realsunset.jpg");
GUSTAV = imread("gustavklimt.jpg");
GUSTAV = imresize(GUSTAV, [1280 1280]);
MONALISA = imread("monalisa.jpg");
MONALISA = imresize(MONALISA, [1984 1984]);
```

Task 1 example

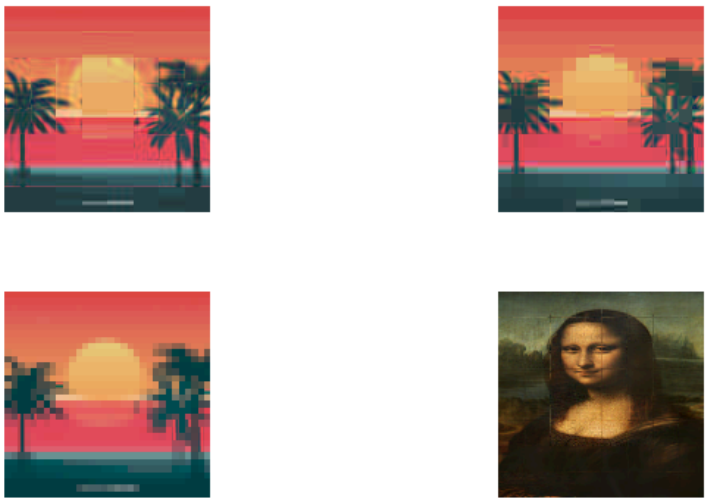
```
% and also used to compare to Task6 result

[Xcoeff_T1, NewIm_T1] = processImage(CS, 16, 16000);
imshow(uint8(abs(NewIm_T1)));
```



Task 2: Quantitative Evaluation

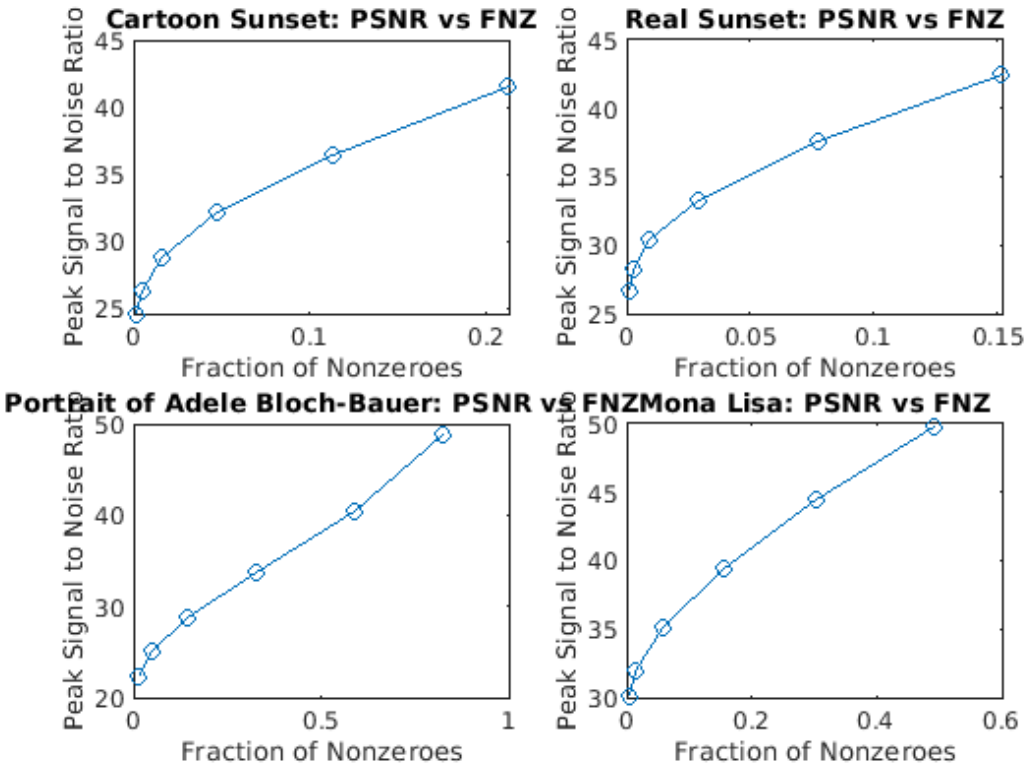
```
% I need new image and older image I for RMSE. RMSE used for getting
 PSNR.
% Then, Xcoeffs used to find FNZ

% for each of the four images, find the PSNR and FNZ of a respective
 tau.
% Then combine all images onto a plot. Choose 5 taus

% the tau's to choose are 1000, 2000, 4000, 8000, and 16000
% I will choose a window number of 8 as I feel that's a "fair middle
% ground"
% FIRST, the cartoon sunset

T2_taus = [500 1000 2000 4000 8000 16000];
CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];

for i = 1:6
    [X_T2, I_T2] = processImage(CS, 8, T2_taus(i));
    rmse_T2 = rmse(CS, I_T2);
```

```
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end

% Real sunset

RS_PSNR = [0 0 0 0 0 0];
RS_FNZ = [0 0 0 0 0 0];

for i = 1:6
    [X_T2, I_T2] = processImage(RS, 8, T2_taus(i));
    rmse_T2 = rmse(RS, I_T2);
    RS_PSNR(i) = psnr(rmse_T2);
    RS_FNZ(i) = fnz(X_T2);
end

GUSTAV_PSNR = [0 0 0 0 0 0];
GUSTAV_FNZ = [0 0 0 0 0 0];

for i = 1:6
    [X_T2, I_T2] = processImage(GUSTAV, 8, T2_taus(i));
    rmse_T2 = rmse(GUSTAV, I_T2);
    GUSTAV_PSNR(i) = psnr(rmse_T2);
    GUSTAV_FNZ(i) = fnz(X_T2);
end

MONALISA_PSNR = [0 0 0 0 0 0];
MONALISA_FNZ = [0 0 0 0 0 0];

for i = 1:6
    [X_T2, I_T2] = processImage(MONALISA, 8, T2_taus(i));
    rmse_T2 = rmse(MONALISA, I_T2);
    MONALISA_PSNR(i) = psnr(rmse_T2);
    MONALISA_FNZ(i) = fnz(X_T2);
end
```



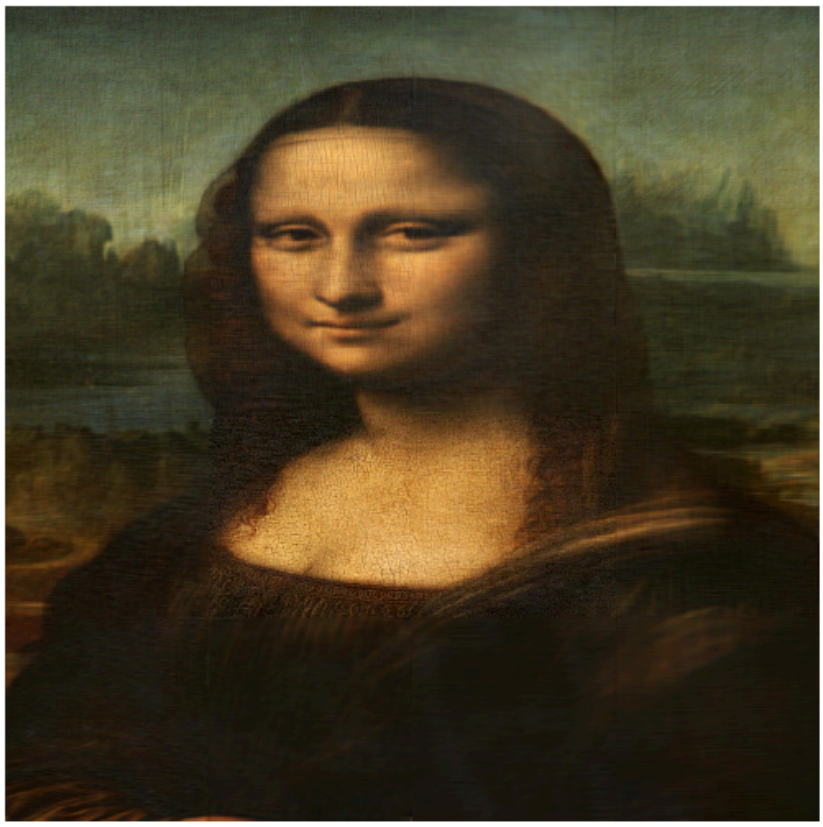
```
figure

subplot(2,2,1);
plot(CS_FNZ, CS_PSNR, "o-");
title("Cartoon Sunset: PSNR vs FNZ");
xlabel("Fraction of Nonzeroes");
ylabel("Peak Signal to Noise Ratio");

subplot(2,2,2);
plot(RS_FNZ, RS_PSNR, "o-");
title("Real Sunset: PSNR vs FNZ");
xlabel("Fraction of Nonzeroes");
ylabel("Peak Signal to Noise Ratio");

subplot(2,2,3);
plot(GUSTAV_FNZ, GUSTAV_PSNR, "o-");
title("Portrait of Adele Bloch-Bauer: PSNR vs FNZ");
xlabel("Fraction of Nonzeroes");
ylabel("Peak Signal to Noise Ratio");

subplot(2,2,4);
plot(MONALISA_FNZ, MONALISA_PSNR, "o-");
title("Mona Lisa: PSNR vs FNZ");
```

```
xlabel("Fraction of Nonzeroes");
ylabel("Peak Signal to Noise Ratio");

hold off
```



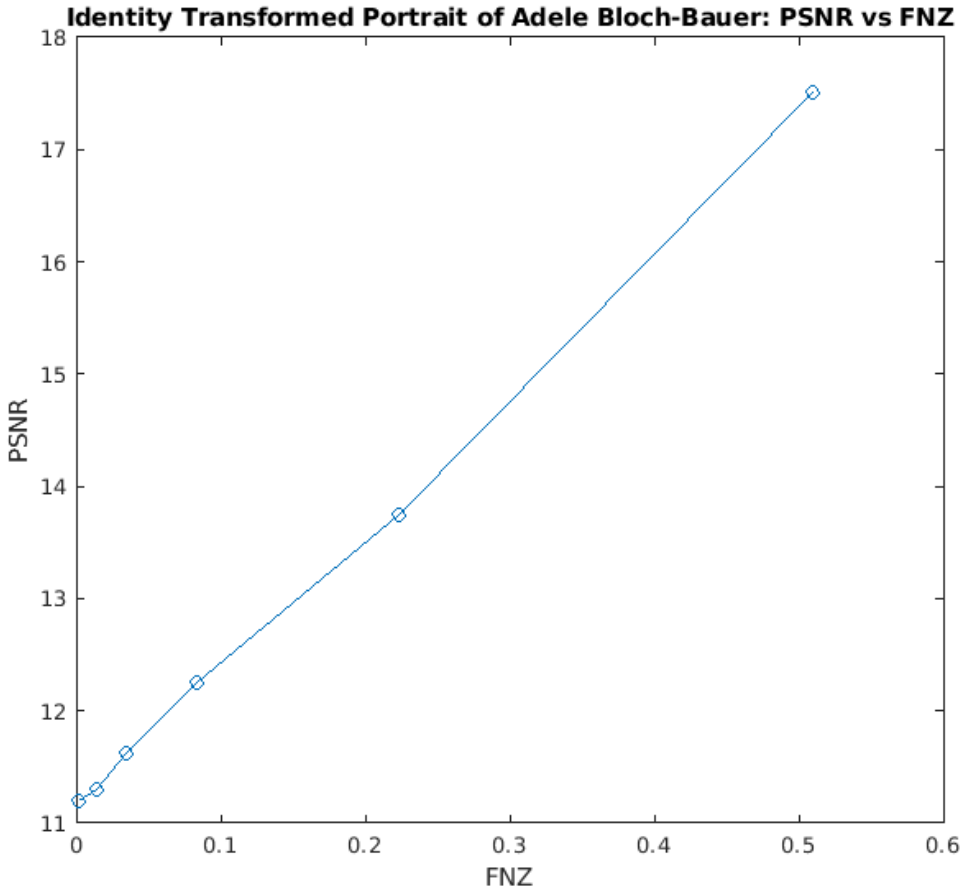
Task 3: Correlating tau to error and visible effect on error in photo

```
hold off

figure
[n1 m1] = processImage(MONALISA, 4, 16000);
% [n2 m2] = processImage(CS, 4, 4000);
% [n3 m3] = processImage(CS, 4, 8000);
% [n4 m4] = processImage(CS, 4, 16000);

t3_error=rmse(MONALISA,m1)
%imshow(int8(abs(m1)));
%imshow(int8(abs(m2)));
%imshow(int8(abs(m3)));
%imshow(int8(abs(m4)));


t3_error =

    6.5664
```



Task 4: Role of transformation and effect on PSNR vs FNZ curves

```
threshold = [40 80 120 160 200 240];
ML_PSNR = [0 0 0 0 0 0];
ML_FNZ = [0 0 0 0 0 0];

for i=1:6
    newML = MONALISA - threshold(i);
    ML_PSNR(i) = psnr(rmse(MONALISA, newML));
    ML_FNZ(i) = fnz(newML);
end

plot(ML_FNZ, ML_PSNR, "o-");
title("Identity Transformed Portrait of Adele Bloch-Bauer: PSNR vs
 FNZ");
xlabel("FNZ");
ylabel("PSNR");
```



Task 5: Increasing patch size and its effect on PSNR vs FNZ curves

```
figure

T5_patches = [4 8 16 32];
T2_taus = [500 1000 2000 4000 8000 16000];
CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];
for i = 1:6
    [X_T2, I_T2] = processImage(CS, T5_patches(1), T2_taus(i));
    rmse_T2 = rmse(CS, I_T2);
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end
subplot(3,2,1)
plot(CS_FNZ, CS_PSNR);
title("CS PSNR vs FNZ: Psize 144");
ylabel("PSNR");
xlabel("FNZ");
```

```
CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];
for i = 1:6
    [X_T2, I_T2] = processImage(CS, T5_patches(2), T2_taus(i));
    rmse_T2 = rmse(CS, I_T2);
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end
subplot(3,2,2)
plot(CS_FNZ, CS_PSNR);
title("CS PSNR vs FNZ: Psize 72");
ylabel("PSNR");
xlabel("FNZ");

CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];
for i = 1:6
    [X_T2, I_T2] = processImage(CS, T5_patches(3), T2_taus(i));
    rmse_T2 = rmse(CS, I_T2);
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end
subplot(3,2,3)
plot(CS_FNZ, CS_PSNR);
title("CS PSNR vs FNZ: Psize 36");
ylabel("PSNR");
xlabel("FNZ");

CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];
for i = 1:6
    [X_T2, I_T2] = processImage(CS, T5_patches(4), T2_taus(i));
    rmse_T2 = rmse(CS, I_T2);
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end
subplot(3,2,4)
plot(CS_FNZ, CS_PSNR);
title("CS PSNR vs FNZ: Psize 18");
ylabel("PSNR");
xlabel("FNZ");

CS_PSNR = [0 0 0 0 0 0];
CS_FNZ = [0 0 0 0 0 0];
for i = 1:6
    [X_T2, I_T2] = processImage(CS, T5_patches(5), T2_taus(i));
    rmse_T2 = rmse(GUSTAV, I_T2);
    CS_PSNR(i) = psnr(rmse_T2);
    CS_FNZ(i) = fnz(X_T2);
end
subplot(3,2,5)
plot(CS_FNZ, CS_PSNR);
title("CS PSNR vs FNZ: Psize 9");
```

```
ylabel("PSNR");
xlabel("FNZ");
```

```
Index exceeds the number of array elements (4).

Error in DSPfinalProjectRM4064 (line 218)
    [X_T2, I_T2] = processImage(CS, T5_patches(5), T2_taus(i));
```

Task 6: Extension--Orthobasis Learning

```
psize = 32;
Y = getPatches(CS, psize);
A = learn_orthobasis_msp(Y);

threshold_tau = 4000;

Xcoeffs_task6 = t6_encodeImage(Y, A, threshold_tau);
decodedImage = t6_decodeImage(Xcoeffs_task6, A, length(CS),length(CS),
 3, psize);

t6_FNZ = t6_FNZ_addition(CS, psize) + fnz(Xcoeffs_task6);
t6_psnr = psnr(rmse(CS,decodedImage));

[j, k] = processImage(CS, 18, 4000);
prev_psnr = psnr(rmse(CS, k));
prev_fnz = fnz(j);
```

Functions: Task 1 encoder and decoder of image alongside Task 2 functions

```
function [XCoeffs, filtImage] = processImage(I, p, tau)
for i = 1:p
    for j = 1:p
        N = length(I);
        Row = ((i-1) * (N/p)) + 1;
        Column = ((j-1) * (N/p)) + 1;

        imagemat = I(Row:Row+N/p-1, Column:Column+N/p-1, :);
        RC = imagemat(:,:,1);
        GC = imagemat(:,:,2);
        BC = imagemat(:,:,3);

        fRC = fft2(RC);
        fGC = fft2(GC);
        fBC = fft2(BC);

        RCindices = abs(fRC) > tau;
        GCindices = abs(fGC) > tau;
        BCindices = abs(fBC) > tau;

        RCfilt = fRC .* RCindices;
        GCfilt = fGC .* GCindices;
```

```
        BCfilt = fBC .* BCindices;
        Xcoeffpatch = cat(3, RCfilt, GCfilt, BCfilt);

        invRC = ifft2(RCfilt);
        invGC = ifft2(GCfilt);
        invBC = ifft2(BCfilt);
        newImagepatch = cat(3, invRC, invGC, invBC);

        XCoeffholder([Row:Row+N/p-1], [Column:Column+N/p-1],[1:3]) =
 Xcoeffpatch;

        Inewholder([Row:Row+N/p-1], [Column:Column+N/p-1],[1:3]) =
 newImagepatch;

    end
end
XCoeffs = XCoeffholder;
filtImage = Inewholder;
%end
imshow(uint8(abs(filtImage)))
end

function fiteval = psnr(RMSE)
M=255;
fiteval = 20*log10(M/RMSE);

end

function rootmeansquareerror = rmse(Inaught, Inew)

rootmeansquareerror = (sum(((abs(int32(Inaught) -
 int32(Inew))).^2),"all")/(length(Inaught)*length(Inaught)*3))^.5;


end

function totalfracnonzeros = fnz(X)

nonzeros = (X ~= 0);
numnonzeros = sum(nonzeros, "all");

totalfracnonzeros = numnonzeros/numel(X);

end



%%%%% functions for Task 6 %%%%%%

function patches = getPatches(I, psize)
    w = length(I);
    h = length(I);
    c = 3;
    patchnum = floor(w/psize)*floor(h/psize);
```

```
    count = 1;
    patches = zeros(psize*psize*3, patchnum);
    for i = 1:psize:(w-psize+1)
        for j = 1:psize:(w-psize+1)
            singlePatch = reshape(I(i:(i+psize-1), j:(j+psize-1),:),
[],1);
            patches(:, count) = double(singlePatch);
            count = count +1;
        end
    end
end


function t6_Xc = t6_encodeImage(Y, A, tau)
    Xi = A' * Y;
    indices = abs(Xi) > tau;
    Xci = Xi .* indices;
    t6_Xc = Xci;
end

function t6_I = t6_decodeImage(Xc, A, width, height, depth, psize)
    t6_Iholder = zeros(width, height, depth);
    Y = A * Xc;
    count = 1;
    for i = 1:psize:(width-psize+1)
        for j = 1:psize:(height-psize+1)
            t6_Iholder(i:(i+psize-1),j:(j+psize-1),:) =
 reshape(Y(:,count), psize, psize, depth);
            count = count + 1;
        end
    end
    t6_I = t6_Iholder;
end

% the following gets added to result of FNZ function for FNZ of task 6
function FNZ_addition = t6_FNZ_addition(I, psize)
    [width, height, depth] = size(I);
    FNZ_addition = ((psize*psize*depth)^2)/(width*height*depth);
end
```

Grazi to Prof. Wright for the provided code on courseworks

```
function A = learn_orthobasis_msp( Y )
% learn_orthobasis_msp
%
% Given an input Y of size n x N, finds an n x n orthogonal matrix A
 \in
% O(n) by solving the optimization problem
%
% \max_{A \in O(n)} || A' Y ||_4^4
```

```
%
% Inputs:
% Y -- n x N data matrix
%
% Outputs:
% A -- n x n matrix with orthogonal columns,
% for which A' Y is spiky (approximately sparse).
%
% Based on Zhai et. al. "Complete Dictionary Learning via L4 Norm
% Maximization over the Orthogonal Group"
%
%%%%%
MAX_ITER = 50;
DISPLAY = 1;
iter = 0;
obj = 0;
n = size(Y,1);
Atr = proj_orthogonal(randn(n,n));
allObj = [];
done = false;
while ~done
    iter = iter + 1;
    Atr = proj_orthogonal( (Atr * Y).^3 * Y' );
    obj = sum(sum( (Atr * Y).^4 ));
    allObj = [allObj, obj];
    if DISPLAY
        disp(['Iter ' num2str(iter) ' Obj ' num2str(obj)]);
        figure(1);
        plot(allObj,'LineWidth',3);
    end
    if iter >= MAX_ITER
        done = true;
    end
end
A = Atr'
end


function Q = proj_orthogonal( M )
```

proj_orthogonal

project a square matrix M onto the orthgonal group, in the Frobenius sense

min_{Q orthogonal} |Q-M|_F^2

Input: M -- n x n (square) matrix

Output: Q -- best orthogonal approximation to M

```
%%%%
[U,S,V] = svd(M);
Q = U*V';
end
```

Published with MATLAB® R2021a