1. Linearity and Shift Invariance

1. (a)
Both methods are linear. Note that for both methods, each output pixel can be written as a weighted sum of the four surrounding input pixels. Further, the weights are not influenced by the four surrounding input pixels (provided the magnification factor \( a \) is constant).

We can also apply the linearity test and see that for all pixels \( S_1 \) and \( S_2 \) (where the first subscript denotes pixels of two different images, and the second subscript denotes the four surrounding input pixels), we can write:

\[
S_1 + S_2 = (a)(b)(S_{1,1} + S_{2,1}) + (1-a)(b)(S_{1,2} + S_{2,2}) + (a)(1-b)(S_{1,3} + S_{2,3}) + (1-a)(1-b)(S_{1,4} + S_{2,4})
\]

\[
= (a)(b)S_{1,1} + (1-a)(b)S_{1,2} + (a)(1-b)S_{1,3} + (1-a)(1-b)S_{1,4}
\]

\[
+ (a)(b)S_{2,1} + (1-a)(b)S_{2,2} + (a)(1-b)S_{2,3} + (1-a)(1-b)S_{2,4}
\]

1. (b)
Neither is shift invariant since the size of the output image is different from the size of the input image. The output is not the same as shifted output by the same amount when the input image is shifted.

2. Separability

2. (a)
Nearest neighbor interpolation is separable because finding the nearest column first and then the nearest row or vice versa gives the same result as finding the nearest pixel among the neighboring four pixels.

Bilinear interpolation is also separable. This is shown in the following set of equations:

\[
S_{12} = (a)S_1 + (1-a)S_2
\]

\[
S_{34} = (a)S_3 + (1-a)S_4
\]

\[
S = (b)S_{12} + (1-b)S_{34} = (a)(b)S_{1,1} + (1-a)(b)S_{1,2} + (a)(1-b)S_{1,3} + (1-a)(1-b)S_{1,4}
\]

Note that in the first two equations, linear interpolation is applied row by row. The third equation, then, applies the linear interpolation column by column to the result of the row-by-row interpolation. It is observed that the resulting \( S \) is equivalent to the \( S \) given in the problem set.

2. (b)
For the nearest neighbor interpolation:

\[
A^T = B = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]
Note that here $a=2$, hence when the interpolation is applied row by row, the $1^{st}$, $3^{rd}$, $5^{th}$, and $7^{th}$ columns of the output image matrix takes the values of the $1^{st}$, $2^{nd}$, $3^{rd}$, and $4^{th}$ columns of the input image matrix, respectively. Same operation is also applied to the $2^{nd}$, $4^{th}$, $6^{th}$, $8^{th}$ columns of the output image matrix. The entire process is done by multiplying $B$ with the input image. With similar process, applied column by column by multiplying the resulting image with $A$ yields the interpolated rows of the output image.

For the bilinear interpolation:

$$A^T = B = \begin{bmatrix}
1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 \\
\end{bmatrix}$$

OR

$$A^T = B = \begin{bmatrix}
\frac{3}{4} & \frac{3}{4} & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{4} & \frac{3}{4} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & \frac{3}{4} & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{3}{4} & 1 & 0 \\
\end{bmatrix}$$

Note that the bilinear interpolation with $a=2$ can be implemented using two different matrices (or more). The difference is due to the freedom in selecting the sample center locations. In the first matrix, the center locations in the output grid are the $1^{st}$, $3^{rd}$, $5^{th}$, and $7^{th}$ rows and columns. A row-by-row operation with the first matrix assigns the $1^{st}$, $2^{nd}$, $3^{rd}$, and $4^{th}$ input columns to the $1^{st}$, $3^{rd}$, $5^{th}$, and $7^{th}$ output columns. The $2^{nd}$, $4^{th}$, and $6^{th}$ output columns take the average of the $1^{st}$ and $2^{nd}$, $2^{nd}$ and $3^{rd}$, $3^{rd}$ and $4^{th}$ columns of the input image respectively.

In the second matrix, a similar operation is done; however, the center locations in the output grid are different. Hence, after the row-by-row operation, the $2^{nd}$ output column is assigned as the sum of $\frac{3}{4}$ of the $1^{st}$ input column and $\frac{1}{4}$ of the $2^{nd}$ input column, the $3^{rd}$ output column is assigned as the sum of $\frac{3}{4}$ of the $1^{st}$ input column and $\frac{3}{4}$ of the $2^{nd}$ input column, etc.

3. MATLAB implementation

3. (a) (b) images and MATLAB code attached

3. (c)

Nearest neighbor interpolation produces image artifacts, specifically jags in the image. Hence, the edges in the image appear to have problems. Bilinear interpolation gives better results in terms of this kind of artifacts; however, the image is blurred due to the averaging operation.
a = 1.05, nearest-neighbor interpolation

a = 1.05, bilinear interpolation
a = 1.25, nearest-neighbor interpolation

a = 1.25, bilinear interpolation
a = 1.05, nearest-neighbor interpolation

a = 1.05, bilinear interpolation
a = 1.25, nearest-neighbor interpolation

a = 1.25, bilinear interpolation
clear;
close all;

FileName1 = 'cman';
FileName2 = 'mri';

for i = 1:2
    % read images
    eval(sprintf('FileName = FileName%d;',i));
    F = imread([FileName '.tif']);
imwrite(uint8(F),sprintf('%s.tif',FileName));

    for a = [1.05 1.25];
        % call function for magnifying image
        G1 = magnify(F,a,'nearest');
        G2 = magnify(F,a,'bilinear');

        % plot images
        figure;
        subplot(3,1,1);
        imagesc(F); colormap(gray); axis image;
title(sprintf('Original Image %s
'));
        subplot(3,1,2);
        imagesc(G1); colormap(gray); axis image;
title(sprintf('Magnified Image %s
 a=%.2f using nearest-neighbor interpolation',FileName,a));
        subplot(3,1,3);
        imagesc(G2); colormap(gray); axis image;
title(sprintf('Magnified Image %s
 a=%.2f using bilinear interpolation',FileName,a));

        % write magnified images
        imwrite(uint8(G1),sprintf('%s_.%2fn.tif',FileName,a));
        imwrite(uint8(G2),sprintf('%s_.%2fb.tif',FileName,a));
    end
end
function G = magnify(F,a,method)
% Magnify the input image F by factor of arbitrary a>1
% and return as image G
% method could be 'nearest'(nearest-neighbor interpolation)
% or 'linear'(bilinear interpolation)
% by Chuo-Ling Chang
% April, 2003

% get size of F and G
F = double(F);
[mF,nF] = size(F);
mG = ceil(a*mF); nG = ceil(a*nF);
G = zeros(mG,nG);

% compute relative pixel locations in G with respect to coordinate system of F
[XG,YG] = meshgrid(1:nG,1:mG);
XG = (XG-1)/(nG-1)*(nF-1)+1;
YG = (YG-1)/(mG-1)*(mF-1)+1;

% interpolation
switch method
% round to the nearest neighbor in F
    case 'nearest'
    G(:) = F(sub2ind([mF,nF],round(YG(:)),round(XG(:))));
% get the four nearest neighbors in F
    case 'bilinear'
    cXG = ceil(XG(:)); fXG = floor(XG(:));
cYG = ceil(YG(:)); fYG = floor(YG(:));
a = cXG-XG(:);
b = cYG-YG(:);
% weighted average according to distance
    G(:) = F(sub2ind([mF,nF],fYG,fXG)).*a.*b...+
            F(sub2ind([mF,nF],fYG,cXG)).*(1-a).*b...+
            F(sub2ind([mF,nF],cYG,fXG)).*a.*(1-b)...+
            F(sub2ind([mF,nF],cYG,cXG)).*(1-a).*(1-b);
end