ECE637 : Digital Image Processing : Lab1

Sriranga R

January 19, 2023

1 C Programming

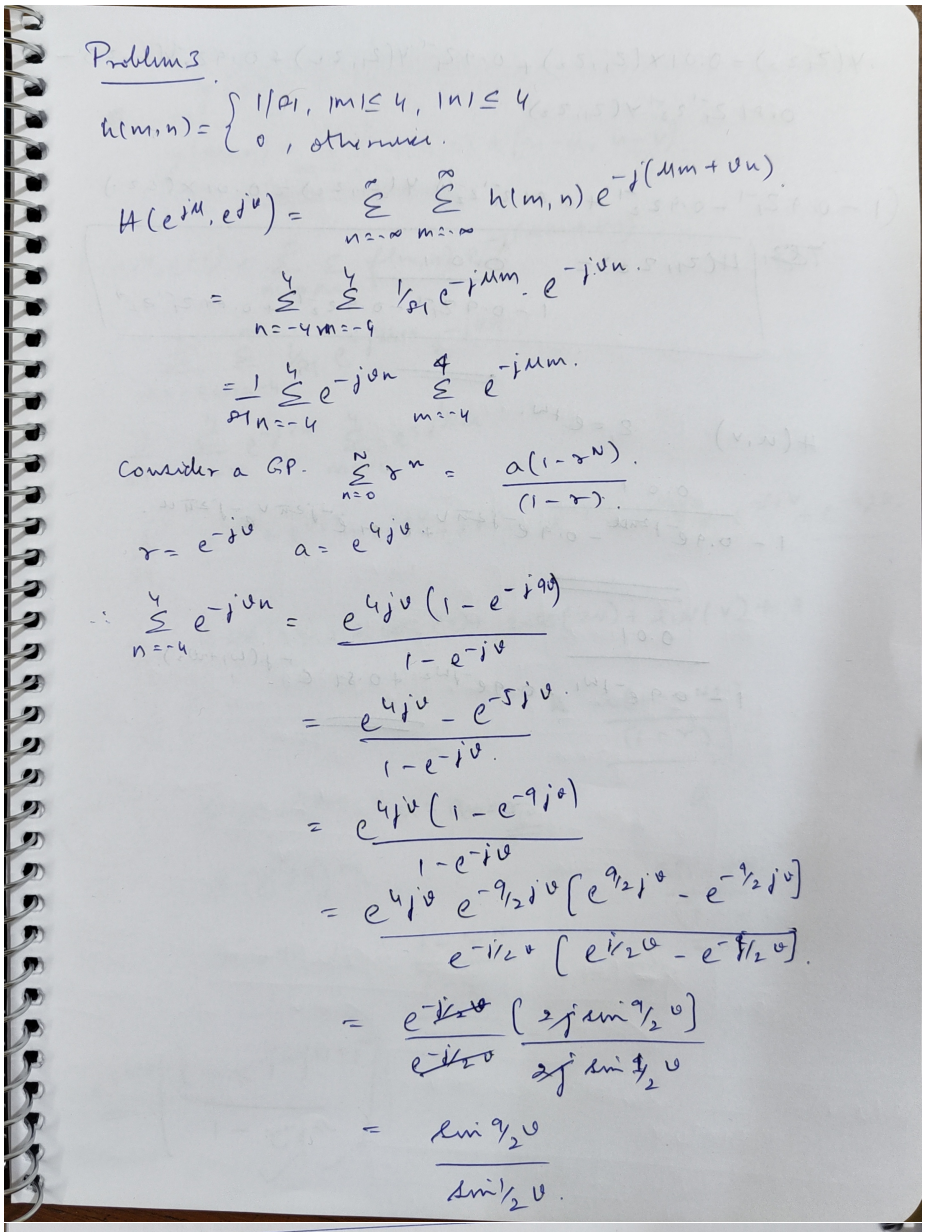
No submission due for this.

2 Display and Export Images in Python

No submission due for this.

3 FIR Low Pass Filter

3.1 Derivation of the analytical expression for $H(e^{j\mu}, e^{j\nu})$



Problem 3

$$h(m,n) = \begin{cases} 1/81, & |m| \le 4, \ |n| \le 4 \\ 0, & \text{otherwise.} \end{cases}$$

$$H(e^{j\mu}, e^{j\nu}) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h(m,n)\, e^{-j(\mu m + \nu n)}$$

$$= \sum_{n=-4}^{4} \sum_{m=-4}^{4} \frac{1}{81} e^{-j\mu m} \cdot e^{-j\nu n}$$

$$= \frac{1}{81} \sum_{n=-4}^{4} e^{-j\nu n} \sum_{m=-4}^{4} e^{-j\mu m}$$

Consider a GP. $\sum_{n=0}^{N} r^n = \frac{a(1-r^N)}{(1-r)}$.

$r = e^{-j\nu}$ $\quad a = e^{4j\nu}$

$$\therefore \sum_{n=-4}^{4} e^{-j\nu n} = \frac{e^{4j\nu}(1 - e^{-j9\nu})}{1 - e^{-j\nu}}$$

$$= \frac{e^{4j\nu} - e^{-5j\nu}}{1 - e^{-j\nu}}$$

$$= \frac{e^{4j\nu}(1 - e^{-9j\nu})}{1 - e^{-j\nu}}$$

$$= \frac{e^{4j\nu} e^{-9/2 j\nu} \left[e^{9/2 j\nu} - e^{-9/2 j\nu}\right]}{e^{-1/2 j\nu}\left[e^{1/2 j\nu} - e^{-1/2 j\nu}\right]}$$

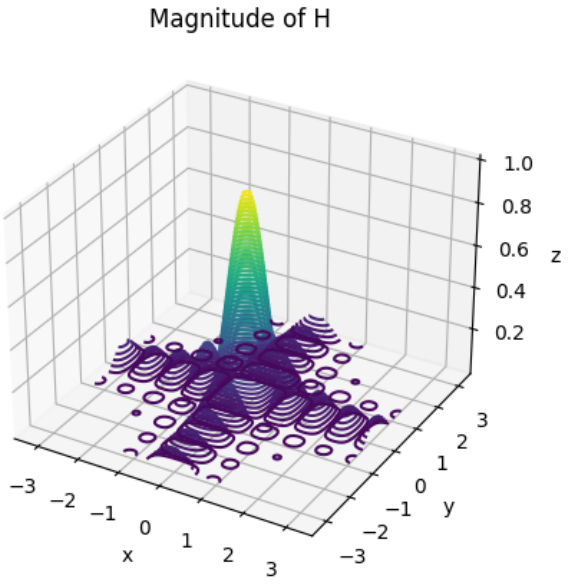
$$= \frac{e^{-1/2 j\nu}}{e^{-1/2 j\nu}} \frac{\left[2j \sin 9/2\, \nu\right]}{2j \sin 1/2\, \nu}$$

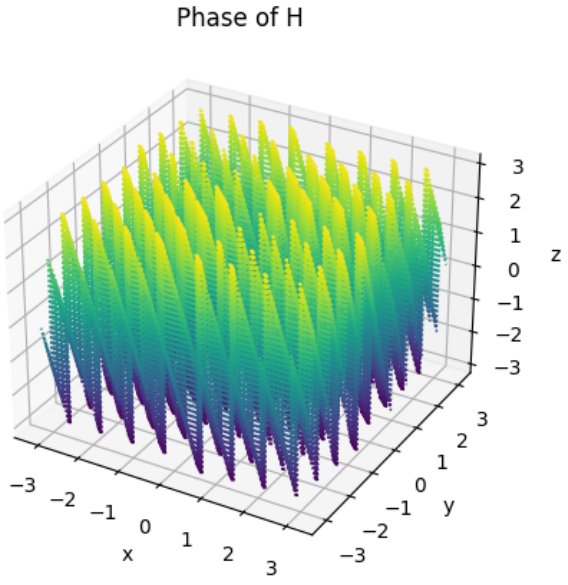
$$= \frac{\sin 9/2\, \nu}{\sin 1/2\, \nu}$$

$$\therefore \frac{1}{81} \sum_{n=-4}^{4} e^{-j\nu n} \sum_{m=-4}^{4} e^{-j\mu m}$$

$$= \frac{1}{81} \frac{\sin 9/2\, \nu}{\sin 1/2\, \nu} \cdot \frac{\sin 9/2\, \mu}{\sin 1/2\, \mu}.$$

3.2 Plot of $|H(e^{j\mu}, e^{j\nu})|$





3.3 Original Image - img03.tif

3.4 The filtered color image.

3.5 C Code

```
1
2 #include <math.h>
3 #include "tiff.h"
4 #include "allocate.h"
```

```
#include "randlib.h"
#include "typeutil.h"
#include "defs.h"

void error(char*);
int8_t filterVal(struct TIFF_img* img, int row, int col);

int main(int argc, char** argv)
{
    FILE *fp;
    struct TIFF_img input_img, color_img;

    if ( argc != 2 ) error( argv[0] );

    /* open image file */
    if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
        fprintf ( stderr, "cannot open file %s\n", argv[1] );
        exit ( 1 );
    }

    /* read image */
    if ( read_TIFF ( fp, &input_img ) ) {
        fprintf ( stderr, "error reading file %s\n", argv[1] );
        exit ( 1 );
    }

    /* close image file */
    fclose ( fp );
    //int32_t output_height = input_img.height + (2 * y_pad);
    //int32_t output_width = input_img.width + (2 * x_pad);
    get_TIFF ( &color_img, input_img.height, input_img.width, 'c' )
    ;
    printf("Input Height = %d, Input width = %d\n", input_img.
    height, input_img.width);

   double** filter = (double**)calloc(9,sizeof(double*));
   for(int i = 0; i < 9; i++)
   {
    filter[i] = (double*)calloc(9, sizeof(double));
   }

   for(int i = 0; i < 9; i++)
   {
    for(int j = 0; j < 9; j++)
    {
        filter[i][j] = 1.0/81;
    }
   }
   convolution2D(&input_img, &color_img, 9, 9, filter);

  if ( ( fp = fopen ( "color_output.tif", "wb" ) ) == NULL ) {
      fprintf ( stderr, "cannot open file color.tif\n");
      exit ( 1 );
  }

  /* write color image */
  if ( write_TIFF ( fp, &color_img ) ) {
```

```
      fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
      exit ( 1 );
  }

   fclose ( fp );

    free_TIFF ( &(input_img) );
    free_TIFF ( &(color_img) );

    return 0;
}

void error(char *name)
{
    printf("usage:  %s  image.tiff \n\n",name);
    printf("this program reads in a 24-bit color TIFF image.\n");
    printf("It then horizontally filters the green component, adds
    noise,\n");
    printf("and writes out the result as an 8-bit image\n");
    printf("with the name 'green.tiff'.\n");
    printf("It also generates an 8-bit color image,\n");
    printf("that swaps red and green components from the input
    image");
    exit(1);
}
```

defs.c consists the implementation of convolution2D.

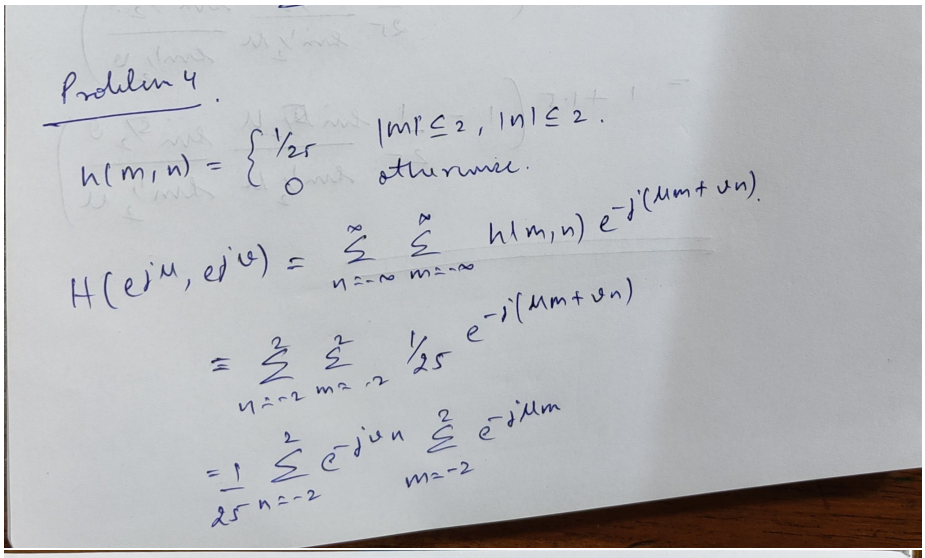
```
#include "defs.h"

// Convolution in 2D
// y[m,n] =
// Sum(u)
// {
//      Sum(v)
//      {
//          h(u,v) * x(m-u, n-v)
//      }
// }

void convolution2D(struct TIFF_img* inputImg, struct TIFF_img*
    outputImg, int filterHeight, int filterWidth,
                    double** filter)
{
    int fhIter = (filterHeight - 1)/2;
    int fwIter = (filterWidth - 1)/2;
    int imageHeight = inputImg->height;
    int imageWidth = inputImg->width;

    double red, green, blue;

    for(int i = 0; i < imageHeight; i++)
    {
        for(int j = 0; j < imageWidth; j++)
        {
            red = 0;
            green = 0;
            blue = 0;
```

```
            for(int row = -fhIter; row <= fhIter; row++)
            {
                for(int col = -fwIter; col <= fwIter; col++)
                {
                    int index_x = i - row;
                    int index_y = j - col;
                    if(index_x < imageHeight && index_y < imageWidth && index_x >= 0 && index_y >= 0)
                    {
                        red += filter[row + fhIter][col + fwIter] * inputImg->color[0][index_x][index_y];
                        green += filter[row + fhIter][col + fwIter] * inputImg->color[1][index_x][index_y];
                        blue += filter[row + fhIter][col + fwIter] * inputImg->color[2][index_x][index_y];
                    }
                }
            }
            outputImg->color[0][i][j] = limitPixel(red);
            outputImg->color[1][i][j] = limitPixel(green);
            outputImg->color[2][i][j] = limitPixel(blue);
        }
    }
}

uint8_t limitPixel(double pixel)
{
    uint8_t colorVal;
    if(pixel > 255)
    {
        colorVal = 255;
    }
    else if(pixel < 0)
    {
        colorVal = 0;
    }
    else
    {
        colorVal = (uint8_t)pixel;
    }
    return colorVal;
}
```

4 FIR Sharpening Filter

4.1 Derivation of the analytical expression for $H(e^{j\mu}, e^{j\nu})$

4.2 Derivation of the analytical expression for $G(e^{j\mu}, e^{j\nu})$



Problem 4.

$$h(m,n) = \begin{cases} 1/25 & |m| \le 2, |n| \le 2 \\ 0 & \text{otherwise} \end{cases}$$

$$H(e^{j\mu}, e^{j\nu}) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h(m,n)\, e^{-j(\mu m + \nu n)}$$

$$= \sum_{n=-2}^{2} \sum_{m=-2}^{2} \frac{1}{25} e^{-j(\mu m + \nu n)}$$

$$= \frac{1}{25} \sum_{n=-2}^{2} e^{-j\nu n} \sum_{m=-2}^{2} e^{-j\mu m}$$

$$\sum_{n=-2}^{2} e^{-j\nu n} = \frac{e^{2j\nu}(1 - e^{-5j\nu})}{1 - e^{-j\nu}}$$

$$= \frac{e^{2j\nu}\, e^{-5/2 j\nu}(e^{5/2 j\nu} - e^{-5/2 j\nu})}{e^{-1/2\nu}(e^{j/2\nu} - e^{-1/2\nu})}$$

$$= \frac{e^{-j/2\nu}\, 2j \sin(5/2\nu)}{e^{-j/2\nu}\, 2j \sin(1/2\nu)}$$

$$\therefore H(e^{j\mu}, e^{j\nu}) = \frac{1}{25} \frac{\sin(5/2\mu)}{\sin(1/2\mu)} \frac{\sin(5/2\nu)}{\sin(1/2\nu)}.$$

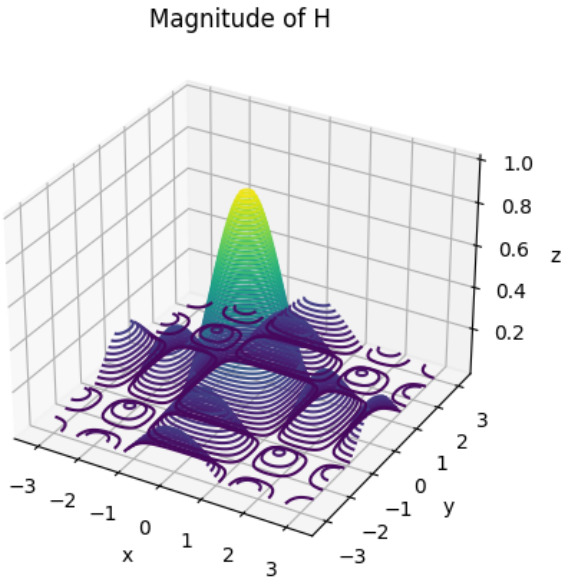
$$\rightarrow g(m,n) = \delta(m,n) + \lambda(\delta(m,n) - h(m,n))$$

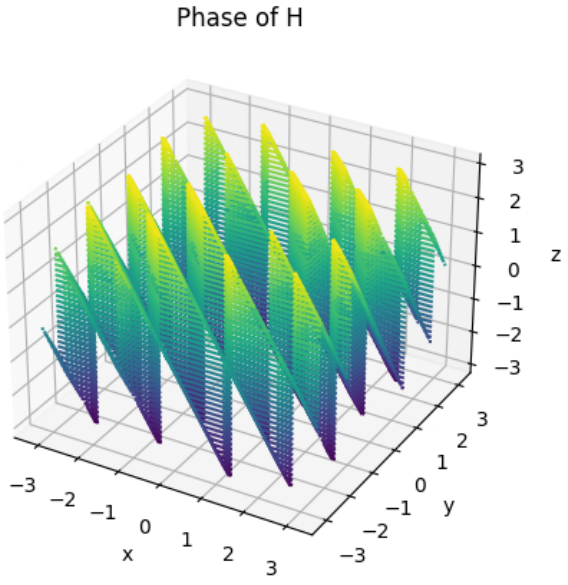
$$\therefore G(e^{j\mu}; e^{j\nu}) = 1 + 1.5(1 - H(e^{j\mu}, e^{j\nu}))$$

$$= 1 + 1.5\left(1 - \frac{1}{25} \frac{\sin 5/2\mu}{\sin 1/2\mu} \frac{\sin 5/2\nu}{\sin 1/2\nu}\right)$$

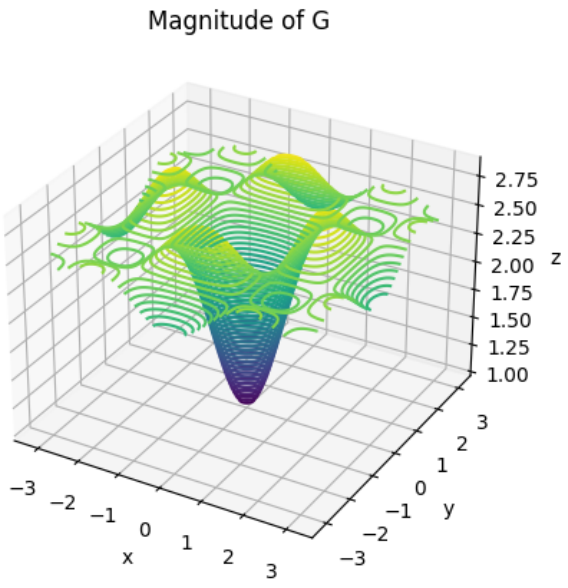
$$= 1 + 1.5\left(1 - \frac{1}{25} \frac{\sin 5/2\mu}{\sin 1/2\mu} \frac{\sin 5/2\theta}{\sin 1/2\mu}\right)$$

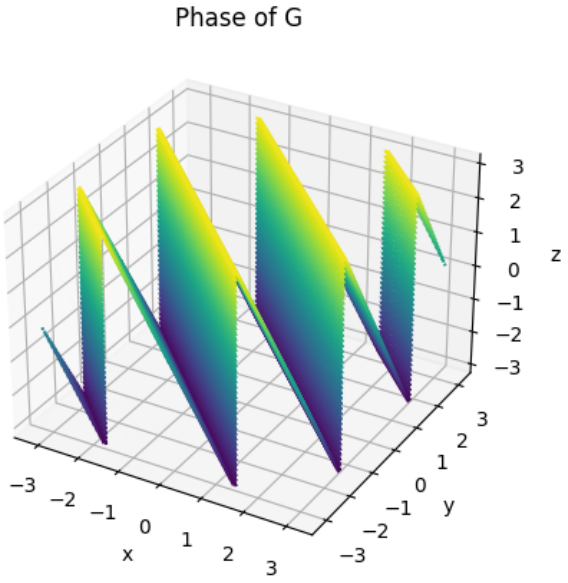
4.3 Plot of $|H(e^{j\mu}, e^{j\nu})|$





4.4 Plot of $|G(e^{j\mu}, e^{j\nu})|$





4.5 The input color image imgblur.tif.

4.6 The output sharpened color image for $\lambda = 1.5$

4.7 C code.

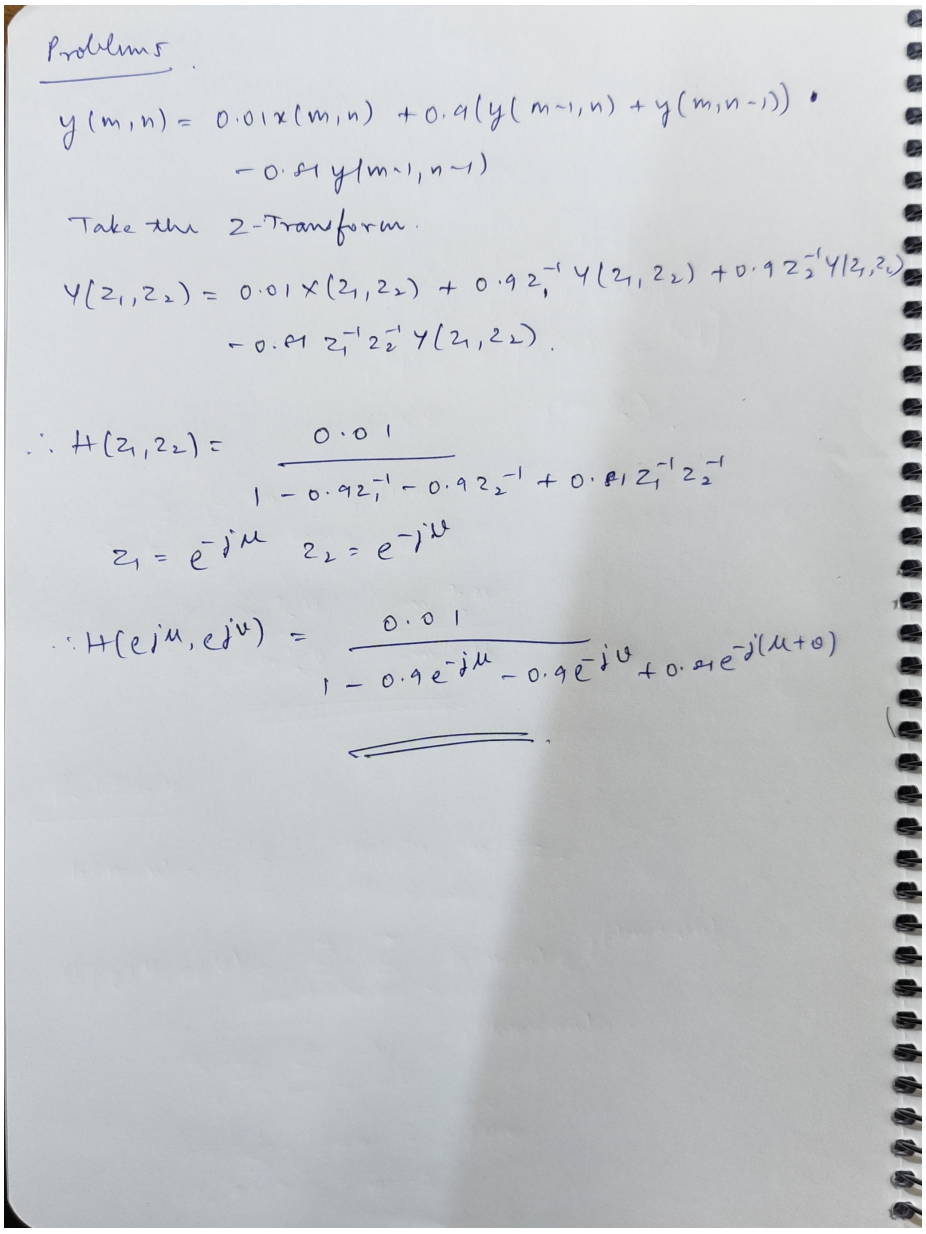
```
1
2 #include <math.h>
3 #include "tiff.h"
4 #include "allocate.h"
```

```
#include "randlib.h"
#include "typeutil.h"
#include "defs.h"

void error(char*);
int8_t filterVal(struct TIFF_img* img, int row, int col);

int main(int argc, char** argv)
{
    FILE *fp;
    struct TIFF_img input_img, color_img;

    if ( argc != 3 ) error( argv[0] );
    float lambda = atof(argv[2]);

    /* open image file */
    if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
        fprintf ( stderr, "cannot open file %s\n", argv[1] );
        exit ( 1 );
    }

    /* read image */
    if ( read_TIFF ( fp, &input_img ) ) {
        fprintf ( stderr, "error reading file %s\n", argv[1] );
        exit ( 1 );
    }

    /* close image file */
    fclose ( fp );
    //int32_t output_height = input_img.height + (2 * y_pad);
    //int32_t output_width = input_img.width + (2 * x_pad);
    get_TIFF ( &color_img, input_img.height, input_img.width, 'c' )
    ;

   double** filter = (double**)calloc(5,sizeof(double*));
   for(int i = 0; i < 5; i++)
   {
    filter[i] = (double*)calloc(5, sizeof(double));
   }
    double delta = 0;
   for(int i = 0; i < 5; i++)
   {
    for(int j = 0; j < 5; j++)
    {
        if(i == 2 && j == 2)
        {
            delta = 1.0;
        }
        else
        {
            delta = 0.0;
        }
        filter[i][j] = delta + lambda * (delta - (1.0/25));
    }
   }
    convolution2D(&input_img, &color_img, 5, 5, filter);
  if ( ( fp = fopen ( "color_sharp.tif", "wb" ) ) == NULL ) {
```

```
      fprintf ( stderr, "cannot open file color.tif\n");
      exit ( 1 );
  }

  /* write color image */
  if ( write_TIFF ( fp, &color_img ) ) {
      fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
      exit ( 1 );
  }

   fclose ( fp );

    free_TIFF ( &(input_img) );
    free_TIFF ( &(color_img) );

    return 0;
}

void error(char *name)
{
    printf("usage:  %s  image.tiff \n\n",name);
    printf("this program reads in a 24-bit color TIFF image.\n");
    printf("It then horizontally filters the green component, adds
    noise,\n");
    printf("and writes out the result as an 8-bit image\n");
    printf("with the name 'green.tiff'.\n");
    printf("It also generates an 8-bit color image,\n");
    printf("that swaps red and green components from the input
    image");
    exit(1);
}
```

5 IIR Filter

5.1 Derivation of the analytical expression for $H(e^{j\mu}, e^{j\nu})$



Problem 5

$$y(m,n) = 0.01x(m,n) + 0.9(y(m-1,n) + y(m,n-1)) - 0.81y(m-1,n-1)$$

Take the 2-Transform.

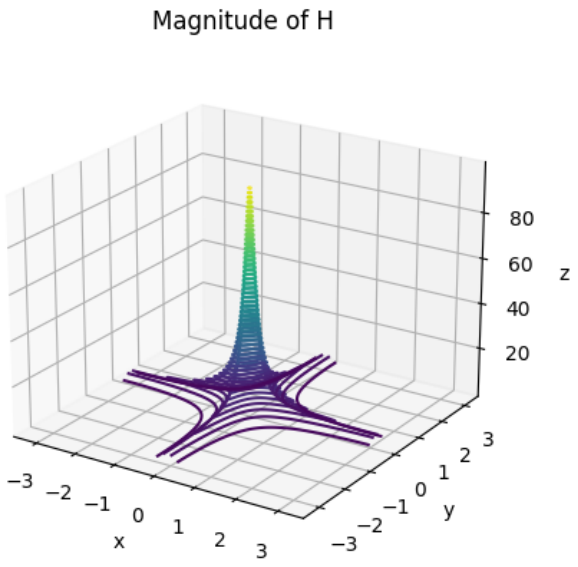
$$Y(z_1,z_2) = 0.01X(z_1,z_2) + 0.9z_1^{-1}Y(z_1,z_2) + 0.9z_2^{-1}Y(z_1,z_2) - 0.81z_1^{-1}z_2^{-1}Y(z_1,z_2).$$

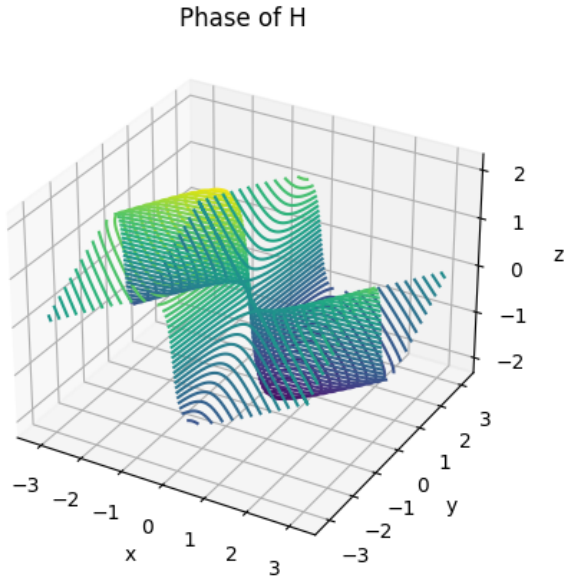
$$\therefore H(z_1,z_2) = \frac{0.01}{1 - 0.9z_1^{-1} - 0.9z_2^{-1} + 0.81z_1^{-1}z_2^{-1}}$$

$$z_1 = e^{-j\mu} \quad z_2 = e^{-j\nu}$$

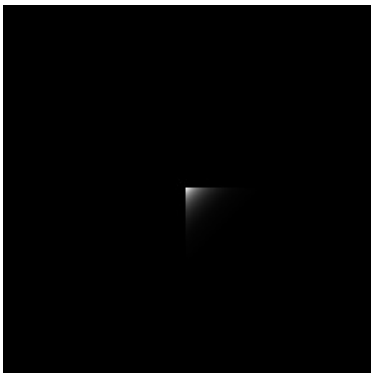
$$\therefore H(e^{j\mu}, e^{j\nu}) = \frac{0.01}{1 - 0.9e^{-j\mu} - 0.9e^{-j\nu} + 0.81e^{-j(\mu+\nu)}}$$

5.2 Plot of $|H(e^{j\mu}, e^{j\nu})|$

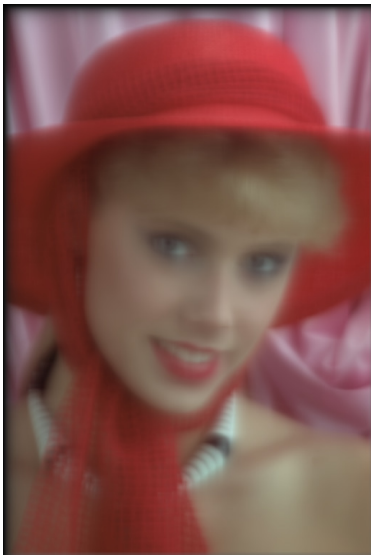




5.3 An image of the point spread function



5.4 The filtered output color image.



5.5 C code

```
1
2  #include <math.h>
3  #include "tiff.h"
4  #include "allocate.h"
5  #include "randlib.h"
6  #include "typeutil.h"
7  #include "defs.h"
8
9  void error(char*);
10 int8_t filterVal(struct TIFF_img* img, int row, int col);
11
```

```
int main(int argc, char** argv)
{
    FILE *fp;
    struct TIFF_img input_img, color_img;

    if ( argc != 2 ) error( argv[0] );

    /* open image file */
    if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
        fprintf ( stderr, "cannot open file %s\n", argv[1] );
        exit ( 1 );
    }

    /* read image */
    if ( read_TIFF ( fp, &input_img ) ) {
        fprintf ( stderr, "error reading file %s\n", argv[1] );
        exit ( 1 );
    }

    /* close image file */
    fclose ( fp );
    get_TIFF ( &color_img, input_img.height, input_img.width, 'c' )
    ;

    double*** inter_img = (double***)calloc(3, sizeof(double**));
    for(int i = 0; i < 3; i++)
    {
        inter_img[i] = (double**)calloc(input_img.height, sizeof(
    double*));
        for(int j = 0; j < input_img.height; j++)
        {
            inter_img[i][j] = (double*)calloc(input_img.width,
    sizeof(double));
        }
    }
   for(int i = 0; i < input_img.height; i++)
   {
    for(int j = 0; j < input_img.width; j++)
    {
        inter_img[0][i][j] = 0.01 * input_img.color[0][i][j];
        inter_img[1][i][j] = 0.01 * input_img.color[1][i][j];
        inter_img[2][i][j] = 0.01 * input_img.color[2][i][j];

        if(i > 0)
        {
            inter_img[0][i][j]+= 0.9 * inter_img[0][i - 1][j];
            inter_img[1][i][j]+= 0.9 * inter_img[1][i - 1][j];
            inter_img[2][i][j]+= 0.9 * inter_img[2][i - 1][j];
        }
        if(j > 0)
        {
            inter_img[0][i][j]+= 0.9 * inter_img[0][i][j - 1];
            inter_img[1][i][j]+= 0.9 * inter_img[1][i][j - 1];
            inter_img[2][i][j]+= 0.9 * inter_img[2][i][j - 1];
        }
        if(i > 0 && j > 0)
        {
```

```
            inter_img[0][i][j] -= 0.81 * inter_img[0][i - 1][j - 1];
            inter_img[1][i][j] -= 0.81 * inter_img[1][i - 1][j - 1];
            inter_img[2][i][j] -= 0.81 * inter_img[2][i - 1][j - 1];
        }
    }

    for(int d = 0; d < 3; d++)
    {
        for(int i = 0; i < input_img.height; i++)
        {
            for(int j = 0; j < input_img.width; j++)
            {
                color_img.color[d][i][j] = limitPixel(inter_img[d][i][j]);
            }
        }
    }

    }
  if ( ( fp = fopen ( "color_IIR.tif", "wb" ) ) == NULL ) {
      fprintf ( stderr, "cannot open file color.tif\n");
      exit ( 1 );
  }

  /* write color image */
  if ( write_TIFF ( fp, &color_img ) ) {
      fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
      exit ( 1 );
  }

   fclose ( fp );

    free_TIFF ( &(input_img) );
    free_TIFF ( &(color_img) );

    return 0;
}

void error(char *name)
{
    printf("usage:  %s  image.tiff \n\n",name);
    printf("this program reads in a 24-bit color TIFF image.\n");
    printf("It then horizontally filters the green component, adds noise,\n");
    printf("and writes out the result as an 8-bit image\n");
    printf("with the name 'green.tiff'.\n");
    printf("It also generates an 8-bit color image,\n");
    printf("that swaps red and green components from the input image");
    exit(1);
}
```

6 Python code

6.1 Problem3

```
import sys
import numpy as np
import matplotlib.pyplot as plt
import scipy.fft
import scipy.signal
from matplotlib import cm

h = (1/81) * np.ones((9,9))
fft_out = np.fft.fft2(h, s=[1024,1024])
fft_out = np.fft.fftshift(fft_out)

#plt.plot(np.linspace(-512, 512, 1024), abs(fft_out[1]))
#plt.show()

x = np.linspace(-np.pi, np.pi, 1024)
y = np.linspace(-np.pi, np.pi, 1024)
X, Y = np.meshgrid(x,y)
Z = abs(fft_out)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Magnitude of H")
#plt.show()

##plt.plot(np.linspace(-np.pi, np.pi, 1024), np.angle(fft_out[1]))
#plt.show()

x = np.linspace(-np.pi, np.pi, 1024)
y = np.linspace(-np.pi, np.pi, 1024)
X, Y = np.meshgrid(x,y)
Z = np.angle(fft_out)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Phase of H")
plt.show()
```

6.2 Problem4

```
import sys
import numpy as np
import matplotlib.pyplot as plt
import scipy.fft
import scipy.signal
from matplotlib import cm

```

```
h = (1/25) * np.ones((5,5))

fft_out = np.fft.fft2(h, s=[1024,1024])
fft_out = np.fft.fftshift(fft_out)

#plt.plot(np.linspace(-512, 512, 1024), abs(fft_out[1]))
#plt.show()

x = np.linspace(-512, 512, 1024)
y = np.linspace(-512, 512, 1024)
X, Y = np.meshgrid(x,y)
Z = abs(fft_out)
fig = plt.figure(1)
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Magnitude of H")
#plt.show()

#plt.plot(np.linspace(-np.pi, np.pi, 1024), np.angle(fft_out[1]))
#plt.show()

x = np.linspace(-np.pi, np.pi, 1024)
y = np.linspace(-np.pi, np.pi, 1024)
X, Y = np.meshgrid(x,y)
Z = np.angle(fft_out)
fig = plt.figure(2)
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Phase of H")
#plt.show()


g = np.zeros((5,5))
lambda_var = 1.5
delta = 0;
for i in range(0,5):
    for j in range(0,5):
        if(i == 2 and j == 2):
            delta = 1
        else:
            delta = 0
        g[i][j] = delta + lambda_var*(delta - h[i][j])

fft_out = np.fft.fft2(g, s=[1024,1024])
fft_out = np.fft.fftshift(fft_out)

#plt.plot(np.linspace(-512, 512, 1024), abs(fft_out[1]))
#plt.show()

x = np.linspace(-512, 512, 1024)
y = np.linspace(-512, 512, 1024)
```

```
X, Y = np.meshgrid(x,y)
Z = abs(fft_out)
fig = plt.figure(3)
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Magnitude of G")
#plt.show()

#plt.plot(np.linspace(-np.pi, np.pi, 1024), np.angle(fft_out[1]))
#plt.show()

x = np.linspace(-np.pi, np.pi, 1024)
y = np.linspace(-np.pi, np.pi, 1024)
X, Y = np.meshgrid(x,y)
Z = np.angle(fft_out)
fig = plt.figure(4)
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title("Phase of G")
plt.show()
```

6.3 Problem5

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import math
import cmath

input_array = np.zeros((256,256))
input_array[127][127] = 1
output_array = np.zeros((256,256))

for i in range(0,256):
    for j in range(0,256):
        output_array[i][j] = 0.01 * input_array[i][j]
        if(i > 0):
            output_array[i][j] += 0.9 * output_array[i - 1][j]
        if(j > 0):
            output_array[i][j] += 0.9 * output_array[i][j - 1]
        if(i > 0 and j > 0):
            output_array[i][j] -= 0.81 * output_array[i - 1][j - 1]

im_save = Image.fromarray((255 * 100 * output_array).astype(np.
    uint8))
im_save.save("h_out.tif")


x = np.linspace(-np.pi, np.pi, 1024)
y = np.linspace(-np.pi, np.pi, 1024)
```

```
28
29 def genComplexArray(m,n):
30     return (np.arange(m) * 0j)[:, None] + np.arange(n)
31 z = genComplexArray(1024, 1024)
32 X, Y = np.meshgrid(x,y)
33
34 for m in range(0,len(x)) :
35     for n in range(0,len(y)) :
36         z[m][n] = 1 / (1 - (0.9 * cmath.exp(complex(0,-x[m]))) -
    (0.9 * cmath.exp(complex(0,-y[n]))) + 0.81 * (cmath.exp(complex
    (0,-(x[m] + y[n])))))
37 Z = abs(z)
38 fig = plt.figure(1)
39 ax = plt.axes(projection='3d')
40 ax.contour3D(X, Y, Z, 50, cmap='viridis')
41 ax.set_xlabel('x')
42 ax.set_ylabel('y')
43 ax.set_zlabel('z')
44 ax.set_title("Magnitude of H")
45
46 Z = np.angle(z)
47 fig = plt.figure(2)
48 ax = plt.axes(projection='3d')
49 ax.contour3D(X, Y, Z, 50, cmap='viridis')
50 ax.set_xlabel('x')
51 ax.set_ylabel('y')
52 ax.set_zlabel('z')
53 ax.set_title("Phase of H")
54
55 plt.show()
```