**数字图像处理第四次作业**

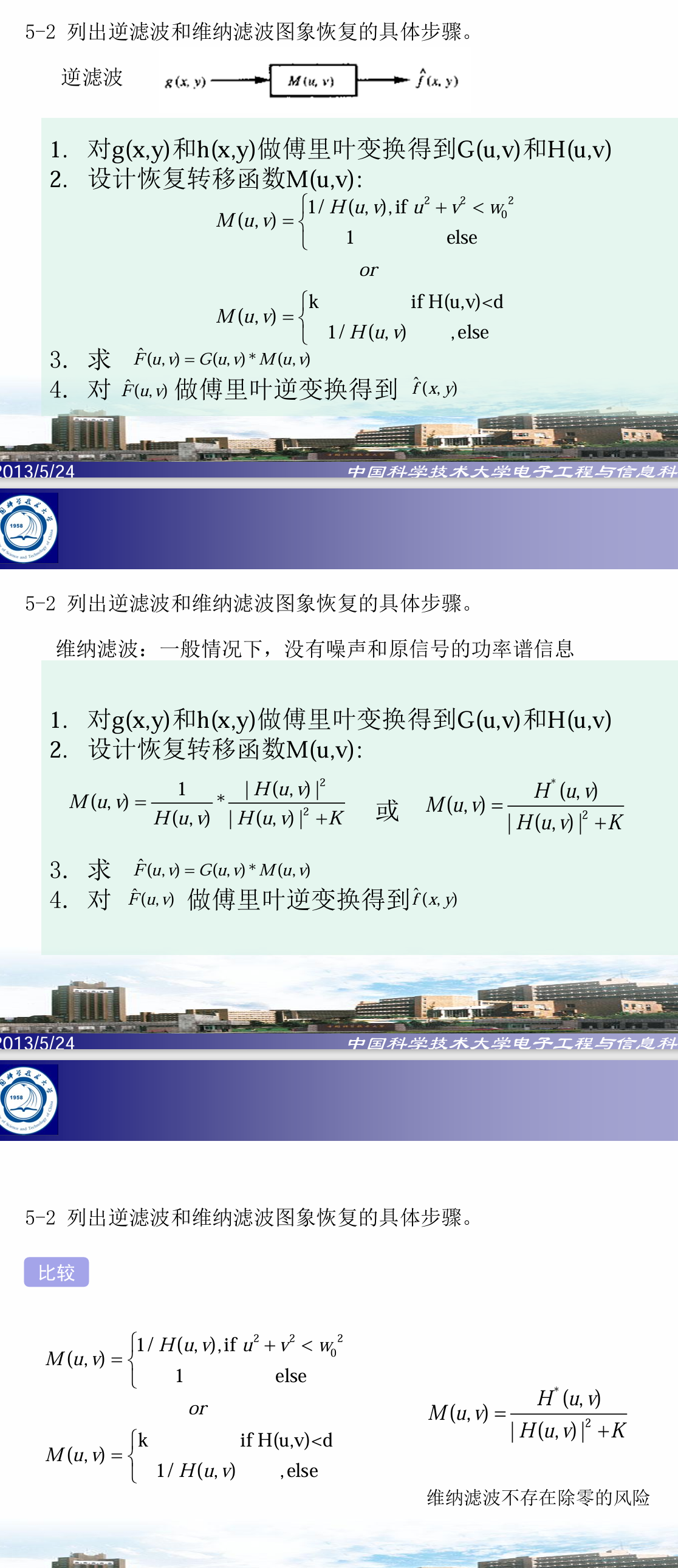
浮焕然 PB22061345

问题：

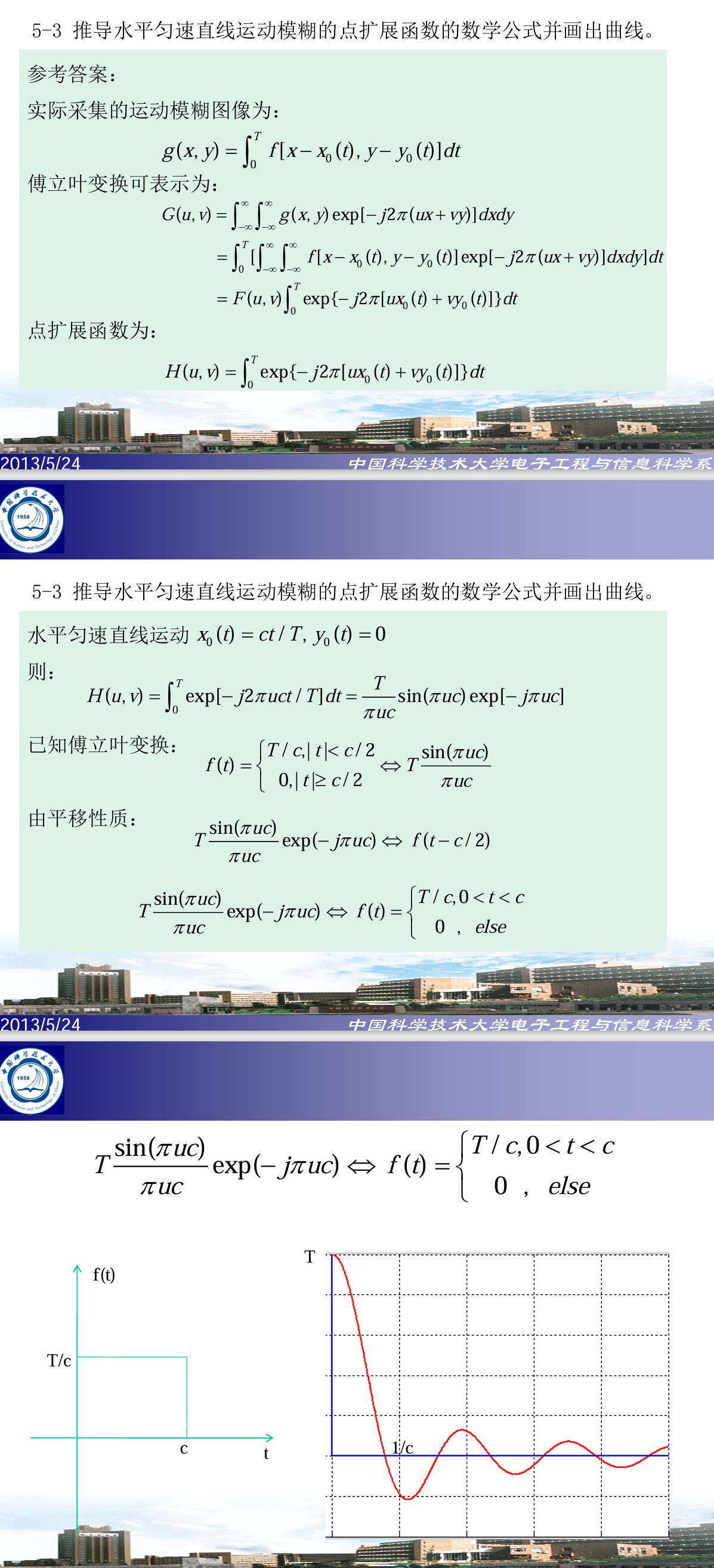
1. 写出逆滤波和维纳滤波图象恢复的具体步骤。

逆滤波：

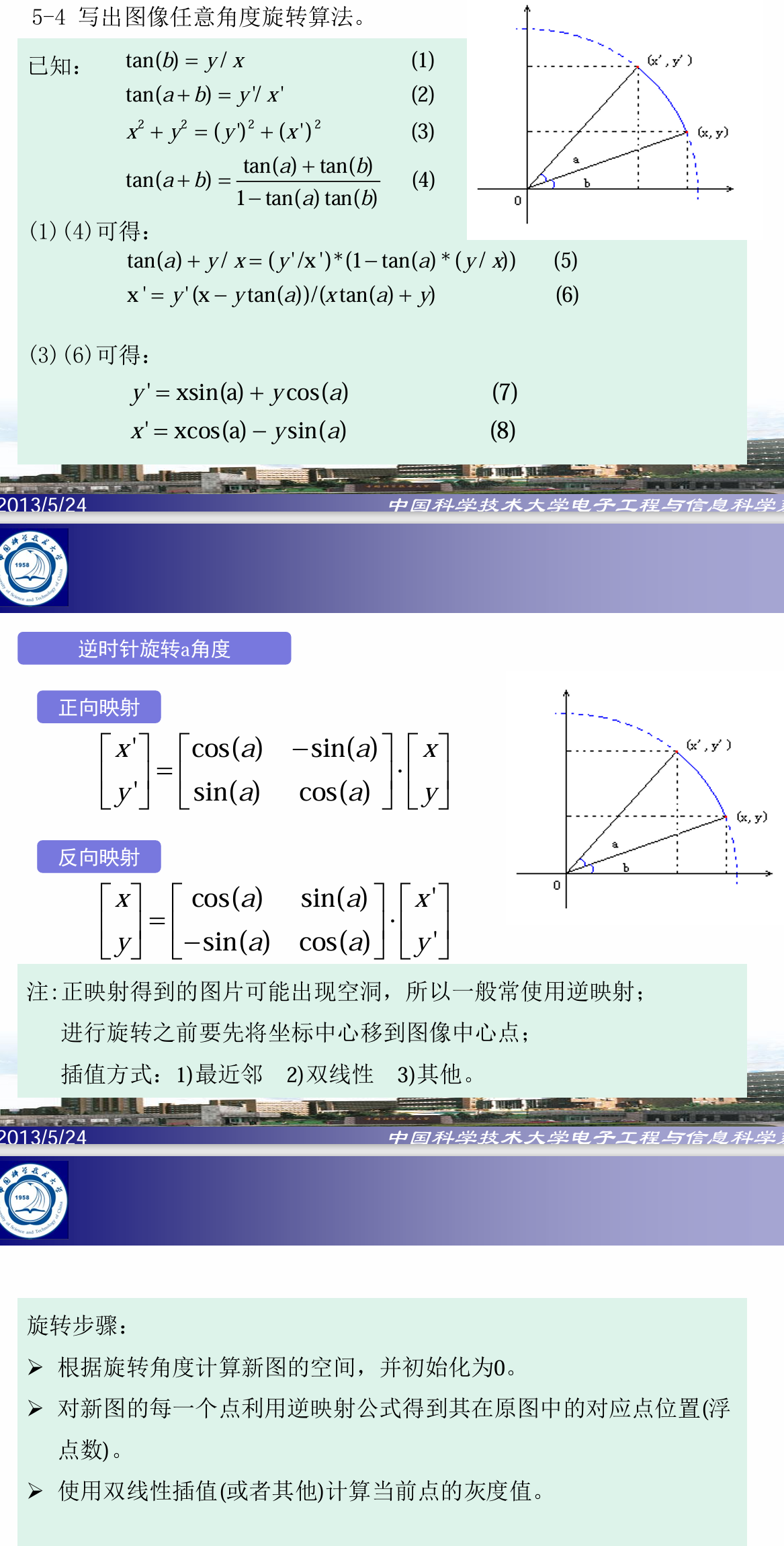
* 1. 对和h做傅里叶变换得到



1. 推导水平匀速直线运动模糊的点扩展函数的数学公式并画出曲线。



1. 编程实现lema.bmp的任意角旋转。



第五题代码如下（省略了图片读取、显示、保存部分）：

*# 添加高斯噪声*

def **add\_gaussian\_noise**(image, mean=0, std=5):

    gaussian\_noise = **np**.**random**.normal(mean, std, image.shape).astype(**np**.uint8)

    output = **np**.**zeros**(image.shape, **np**.uint8)

    for i in **range**(image.shape[0]):

        for j in **range**(image.shape[1]):

            temp = **int**(gaussian\_noise[i][j])+image[i][j]

            if temp<0:

                output[i][j] = 0

            elif temp>255:

                output[i][j] = 255

            else:

                output[i][j] = temp

    return output

*# 添加椒盐噪声*

def **add\_salt\_and\_pepper\_noise**(image, ratio):

    output = **np**.**zeros**(image.shape, **np**.uint8)

    for i in **range**(image.shape[0]):

        for j in **range**(image.shape[1]):

            rand = **random**.random()

            if rand < ratio:  *# salt pepper noise*

                if **random**.random() > 0.5:  *# change the pixel to 255*

                    output[i][j] = 255

                else:

                    output[i][j] = 0

            else:

                output[i][j] = image[i][j]

    return output

*# 局域平均滤波*

def **LocalAverageFilter**(imarray, k=3):

    height, width = imarray.shape[:2]

    edge = **int**((k-1)/2)

    output = **np**.**zeros**((height, width), dtype="uint8")

    for i in **range**(height):

        for j in **range**(width):

            if i <= edge - 1 or i >= height - 1 - edge or j <= edge - 1 or j >= width - edge - 1:

                output[i, j] = imarray[i, j]

            else:

                window = imarray[i - edge:i + edge + 1, j - edge:j + edge + 1]

                output[i, j] = **np**.**mean**(window)

    return output

*# 中值滤波*

def **MedianFilter**(imarray, k=3):

    height, width = imarray.shape[:2]

    edge = **int**((k-1)/2)

    output = **np**.**zeros**((height, width), dtype="uint8")

    for i in **range**(height):

        for j in **range**(width):

            if i <= edge - 1 or i >= height - 1 - edge or j <= edge - 1 or j >= width - edge - 1:

                output[i, j] = imarray[i, j]

            else:

                output[i, j] = **np**.**median**(imarray[i - edge:i + edge + 1, j - edge:j + edge + 1])

    return output

*# 添加高斯噪声*

gaussian\_noisy\_image = **add\_gaussian\_noise**(image,0,1)

*# 添加椒盐噪声*

salt\_pepper\_noisy\_image = **add\_salt\_and\_pepper\_noise**(image,0.1)

*# 对高斯噪声图像进行局域平均滤波和中值滤波*

gaussian\_mean\_filtered = **LocalAverageFilter**(gaussian\_noisy\_image)

gaussian\_median\_filtered = **MedianFilter**(gaussian\_noisy\_image)

*# 对椒盐噪声图像进行局域平均滤波和中值滤波*

salt\_pepper\_mean\_filtered = **LocalAverageFilter**(salt\_pepper\_noisy\_image)

salt\_pepper\_median\_filtered = **MedianFilter**(salt\_pepper\_noisy\_image)