信号与系统实验报告

名 称：数字信号卷积和的实现及应用

学 院：计软智学院

专 业：人工智能专业

学 号：58122310

姓 名：唐梓烨

日期：2024 年 04 月 04 日

评分：

一、实验目的

1. 掌握 Matlab 中相关函数的使用，程序代码编制与调试的流程。
2. 熟悉卷积和的运算规则及其意义，加深对离散时间信号分析的理解。

二、实验任务

1. 完成实验内容全部题目，分析解决调试代码过程中出现的问题。
2. 认真完成本次实验小结，思考卷积和的应用。

三、主要设备、软件平台

1. 硬件：计算机
2. 软件：Matlab

四、实验内容

1. 输出杨辉三角。
    1) 函数编写
    2) 控制台输出
    3) 循环语句、条件语句
    4) 程序运行、调试

代码：

```
function pascalTriangle(rows)
% 初始化一个空矩阵来存储杨辉三角
    triangle = zeros(rows);

     % 填充杨辉三角
    for i = 1:rows
        for j = 1:i
            if j == 1 || j == i
                triangle(i,j) = 1;
            else
                triangle(i,j) = triangle(i-1,j-1) + triangle(i-1,j);
            end
        end
    end

    % 以正三角形式输出杨辉三角到控制台
    for i = 1:rows    % 只输出矩阵下三角
        fprintf('%*s', 2*(rows-i), ''); % 每一行的前面用空格填充
        for j = 1:i
            fprintf('%4d', triangle(i,j));
```

```
        end
        fprintf('\n');
    end
end
```

控制台输出：

```
>> pascalTriangle(10)
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
  1   9  36  84 126 126  84  36   9   1
```

2. 编程实现信号 $x(n)$，$h(n)$间的卷积和运算函数 $my\_\mathrm{cov}(x,h)$，并绘制出下列信号卷积和波形。

1) $x(n)=[1,2,3,...,10]$， $h(n)=[1,1]$

2) $x(n)=[1,2,3,9,4,5,6,0,7,8]$， $h(n)=[-1,2,-1]$

3) $x(n)=[1,2,3,...,10]$， $h(n)=[1,2,3,..,10]$

实现的卷积和函数：

```
function [result,len_result] = my_cov(x, h)
    % 获取输入序列的长度
    len_x = length(x);
    len_h = length(h);

    % 计算卷积和结果的长度
    len_result = len_x + len_h - 1;

    % 初始化卷积和结果数组
    result = zeros(1, len_result);

    % 利用双重 for 循环计算卷积和
    for n = 1:len_result
```

```
            for k = max(1, n - len_h + 1):min(len_x, n)
                result(n) = result(n) + x(k) * h(n-k+1);
            end
        end
        disp(result)
    end
```

计算例题要求的卷积和：

```
x1 = [1:10];
h1 = [1,1];
[result1,len_result1] = my_cov(x1,h1);
x2 = [1,2,3,9,4,5,6,0,7,8];
h2 = [-1,2,-1];
[result2,len_result2] = my_cov(x2,h2);
x3 = x1;
h3 = x1;
[result3,len_result3] = my_cov(x3,h3);

% 输出三幅图
subplot(3, 1, 1);
plot(1:len_result1, result1);
xlabel('n');
ylabel('y');
title('Convolution of x=[1,2,...,10],h=[1,1]');
subplot(3, 1, 2);
plot(1:len_result2, result2);
xlabel('n');
ylabel('y');
title('Convolution of x=[1,2,3,9,4,5,6,0,7,8],h=[-1,2,-1]');
subplot(3, 1, 3);
plot(1:len_result3, result3);
xlabel('n');
ylabel('y');
title('Convolution of x=[1,2,...,10],h=[1,2,...,10]');
```

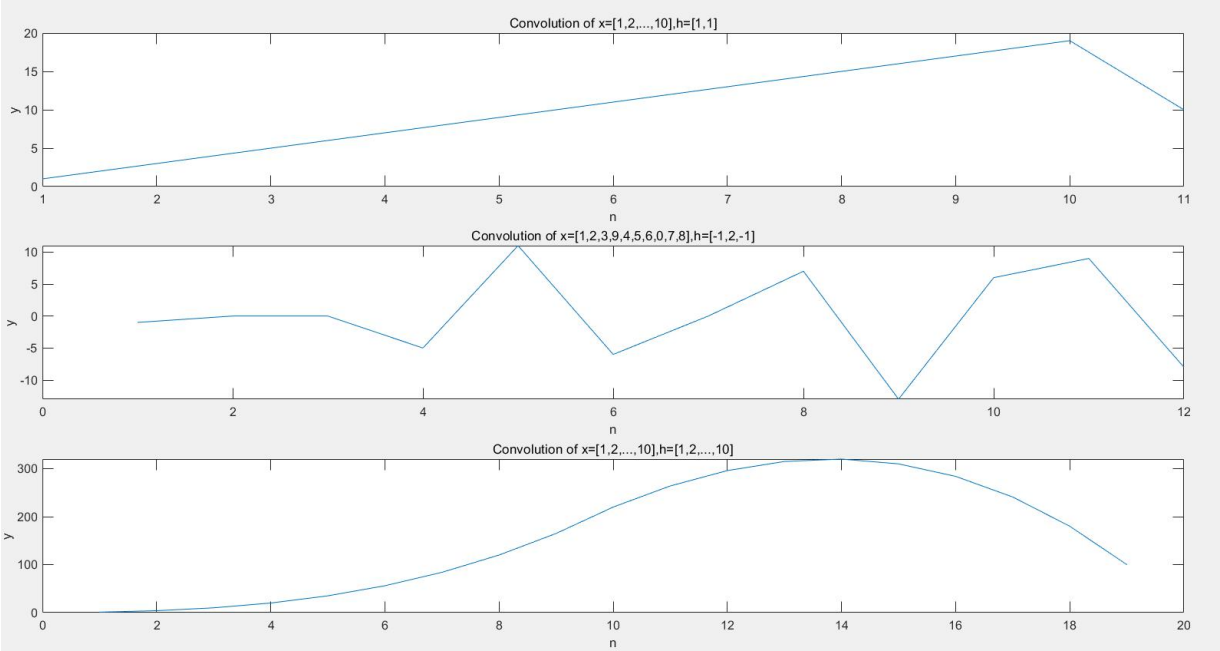
控制台输出：

```
>> cov_calculate
     1     3     5     7     9    11    13    15    17    19    10

    -1     0     0    -5    11    -6     0     7   -13     6     9    -8

     1     4    10    20    35    56    84   120   165   220   264   296   315   320   310   284   241   180   100
```

输出图像：



3. 设计游戏时，若对小怪使用一次技能的效果是“小怪会在接下来 5 秒内持续掉血，每秒掉血量分别为[5 4 3 2 1]”；如果间隔 1 秒连续发动 3 次技能，请绘制出每次攻击后小怪的累计掉血量情况。

代码：

```
% 初始化小怪的初始血量
initial_health = 100;

% 初始化每次攻击的掉血效果
damage_per_second = [5, 4, 3, 2, 1];

% 模拟 3 次技能攻击，分别在第一秒，第二秒，第三秒进行攻击
attack = [1,1,1];

% 通过卷积计算每秒损失的血量，使用第二问定义的 my_cov 函数
[damage,len_damage] = my_cov(attack,damage_per_second);

% 定义每秒血量值，并用每秒损失血量迭代计算剩余血量
health = zeros(1,len_damage);
for i = 1:len_damage
    if i == 1    % 第一秒血量=初始血量-在第一秒损失的血量
        health(i) = initial_health - damage(i);
    else       % 第 i 秒血量=第 i-1 秒血量-在第 i 秒损失的血量
        health(i) = health(i-1) - damage(i);
    end
```

```
    end
    % 补充在第零秒的血量并输出血量数列
    health = [initial_health,health]

    % 将血量数列可视化
    plot(0:len_damage, health);
    xlabel('S');
    ylabel('HP');
```

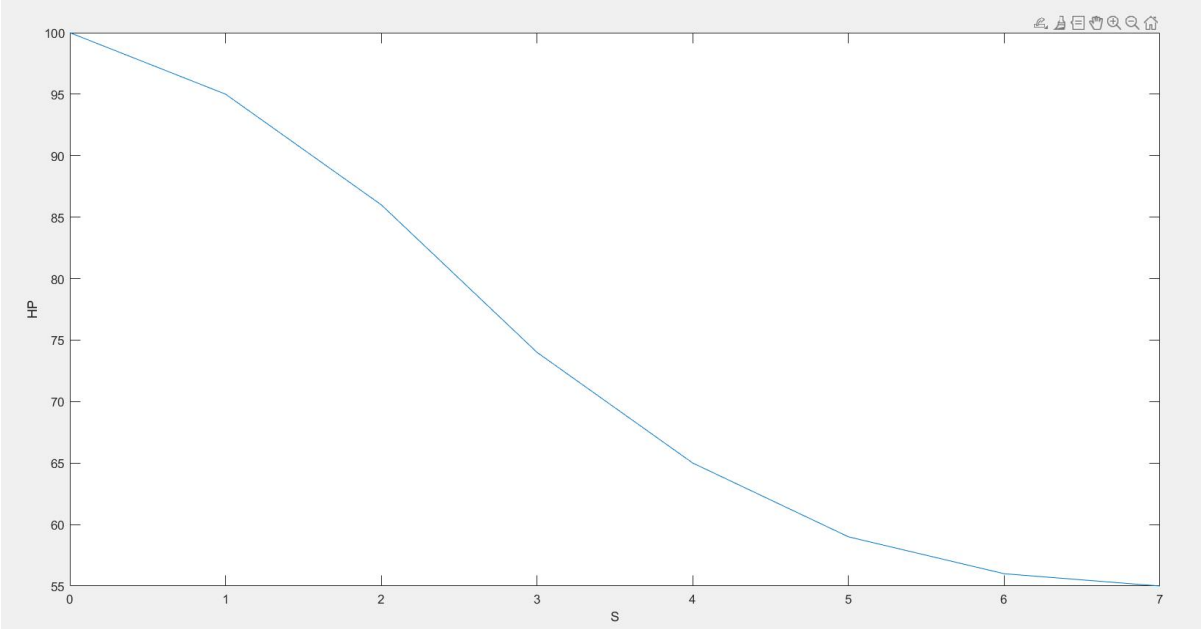
控制台输出：

```
>> HP
     5     9    12     9     6     3     1


health =

   100    95    86    74    65    59    56    55
```

血量随时间变化的可视化：



五、探究拓展

1. 给定一个如下所示的二维矩阵，实现其自身的卷积运算。

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

分析：

在实现矩阵的卷积时，若要保证输出矩阵与输入矩阵大小相同，那么在计算卷积时就要将被卷积矩阵进行填充。有零填充、常数填充、复制填充等等。

下面我将实现常数填充（在填充部分填入常数），和复制填充（在填充部分填入矩阵的边界值）。

常数填充函数：

```
% 参数释义 input_matrix：被卷积矩阵，padding_height：要填充的高度，取决于卷积核的高度，padding_width：要填充的宽度，取决于卷积核的宽度，padding_value：填充的常数值
function padded_matrix = constant_padding(input_matrix, padding_height, padding_width, padding_value)
    % 获取输入矩阵的大小
    [input_height, input_width] = size(input_matrix);

    % 计算填充后的矩阵大小
    padded_height = input_height + 2 * padding_height;
    padded_width = input_width + 2 * padding_width;

    % 初始化填充后的矩阵
    padded_matrix = zeros(padded_height, padded_width);

    % 将输入矩阵复制到填充后的矩阵的中心位置
    padded_matrix(padding_height + 1:padding_height + input_height, padding_width + 1:padding_width + input_width) = input_matrix;

    % 对填充区域进行填充
    % 填充上方
    padded_matrix(1:padding_height, :) = padding_value;
    % 填充下方
    padded_matrix(end-padding_height+1:end, :) = padding_value;
    % 填充左侧
    padded_matrix(:, 1:padding_width) = padding_value;
    % 填充右侧
    padded_matrix(:, end-padding_width+1:end) = padding_value;
end
```

复制填充函数：

```
% 参数释义 input_matrix：被卷积矩阵，padding_height：要填充的高度，取决于卷积核的高度，padding_width：要填充的宽度，取决于卷积核的宽度
function padded_matrix = replicate_padding(input_matrix, padding_height, padding_width)
    % 获取输入矩阵的大小
    [input_height, input_width] = size(input_matrix);
```

```
        % 计算填充后的矩阵大小
        padded_height = input_height + 2 * padding_height;
        padded_width = input_width + 2 * padding_width;

        % 初始化填充后的矩阵
        padded_matrix = zeros(padded_height, padded_width);

        % 将输入矩阵复制到填充后的矩阵的中心位置
        padded_matrix(padding_height + 1:padding_height + input_height, padding_width + 1:padding_width + input_width) = input_matrix;

        % 复制边界值
        % 复制上方边界
        for i = 1:padding_height    % 一行行复制
            padded_matrix(i, :) = padded_matrix(padding_height + 1, :);
        end
        % 复制下方边界
        for i = 1:padding_height
            padded_matrix(end - i + 1, :) = padded_matrix(end - padding_height, :);
        end
        % 复制左侧边界
        for j = 1:padding_width    % 一列列复制
            padded_matrix(:, j) = padded_matrix(:, padding_width + 1);
        end
        % 复制右侧边界
        for j = 1:padding_width
            padded_matrix(:, end - j + 1) = padded_matrix(:, end - padding_width);
        end
    end
```

定义了矩阵卷积函数（调用了前面两个填充函数）：

```
    function result = matrix_cov(input_matrix, conv_kernel, padding_method)
        % 获取输入矩阵和卷积核的大小
        [input_height, input_width] = size(input_matrix);
        [kernel_height, kernel_width] = size(conv_kernel);

        % 计算填充的大小
        padding_height = floor(kernel_height / 2);
        padding_width = floor(kernel_width / 2);

        % 根据填充方法对输入矩阵进行填充
        if strcmp(padding_method, 'constant')
```

```
            padded_matrix = constant_padding(input_matrix, padding_height, padding_width, 0)
        elseif strcmp(padding_method, 'replicate')
            padded_matrix = replicate_padding(input_matrix, padding_height, padding_width)
        else
            error('Unknown padding method');
        end

        % 初始化卷积结果矩阵
        result = zeros(input_height, input_width);

        % 执行卷积操作
        for i = 1:input_height
            for j = 1:input_width
                % 计算(i, j)点的卷积结果
                conv_result = 0;
                for m = 1:kernel_height
                    for n = 1:kernel_width
                        % 矩阵对应相乘后累加计算卷积结果
                        conv_result = conv_result + padded_matrix(i+m-1, j+n-1) * conv_kernel(m, n);
                    end
                end
                % 将卷积结果填入结果矩阵中
                result(i, j) = conv_result;
            end
        end
    end
```

实现例题的函数（调用了前面的矩阵卷积计算函数）：

```
input_matrix = [0 1 0; 1 1 1; 0 1 0];
conv_kernel = input_matrix;

% 使用常数填充
result_constant = matrix_cov(input_matrix, conv_kernel, 'constant')

% 使用边界复制填充
result_replicate = matrix_cov(input_matrix, conv_kernel, 'replicate')
```

控制台输出：
常数填充（以零填充为例）的填充后矩阵和卷积结果：

```
padded_matrix =

     0     0     0     0     0
     0     0     1     0     0
     0     1     1     1     0
     0     0     1     0     0
     0     0     0     0     0


result_constant =

     2     2     2
     2     5     2
     2     2     2
```

复制填充的填充后矩阵和卷积结果：

```
padded_matrix =

     0     0     1     0     0
     0     0     1     0     0
     1     1     1     1     1
     0     0     1     0     0
     0     0     1     0     0


result_replicate =

     2     3     2
     3     5     3
     2     3     2
```

六、实验小结

本次实验我熟悉了 matlab 的基本操作。包括对应用界面的熟悉、对数列和矩阵的一些基本操作、for 循环和 if 语句的书写格式和对 matlab 控制台，函数，脚本之间概念的理解。

在实验实现过程中，我进一步加深了对卷积的理解。实验 2 从实现课内的卷积和计算公式出发，实验 3 让我尝试将卷积应用在实际问题中。探究实验则让我们进行了矩阵卷积的尝试，矩阵卷积在 CNN 中是很重要的操作，并且在此次实验中我还自学到了要对被卷积函数进行填充才能保证输出矩阵和输入矩阵大小相同，并且尝试写了常数填充和复制填充的算法，加深了我对它们的理解。