卷积演示系统

杨跃浙 WA2214014

杨昀川 WA2214022

设计思路

MATLAB 实现

- 线性卷积 在 MATLAB 4 中，可以使用 conv 函数来实现线性卷积。
  - y = conv(x, h)
- 圆周卷积 在 MATLAB 中，可以使用 cconv 函数来实现圆周卷积，代码如下：
  - y = cconv(x, h, N)
- 或通过快速傅里叶变换（FFT）来实现高效计算：
  - y = ifft(fft(x, N) .* fft(h, N))

- 1.使用 MATLAB 中的 conv 函数直接计算两个信号的线性卷积。 开发函数验证卷积的性质，如交换律和结合律。
- 2. 圆周卷积的实现：实现时域的圆周卷积计算，使用 MATLAB 的 cconv 函数。实现频域的圆周卷积计算，通过 FFT 和 IFFT 来演示卷积的频域等价性。
- 3. GUI 设计： •设计一个简洁的用户界面，包括输入框、计算按钮和图形显示区域。界面应允许用户选择卷积类型（线性或圆周），并输入自定义的信号参数。
- 4. 结果分析：在 GUI 中展示卷积结果的图形，包括原始信号和卷积后的信号。提供选项比较线性卷积和圆周卷积的结果，以及在不同参数下的表现。

线性卷积

- 线性卷积 设两个序列 x(n) 和 h(n)，其线性卷积定义为：

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)$$

- 线性卷积计算过程可以分为以下几步：
- 1. 翻褶: 将 h(m) 进行时间翻转，得到 h(−m)。
- 2. 移位: 将翻转后的 h(−m) 进行平移，得到 h(n − m)。
- 3. 相乘: 对每个 m，计算 x(m) · h(n − m)。
- 4. 相加: 对所有的 m 求和，得到卷积结果 y(n)。

DATE:

②

①

翻转
平移加

⇒ 线性卷积

③

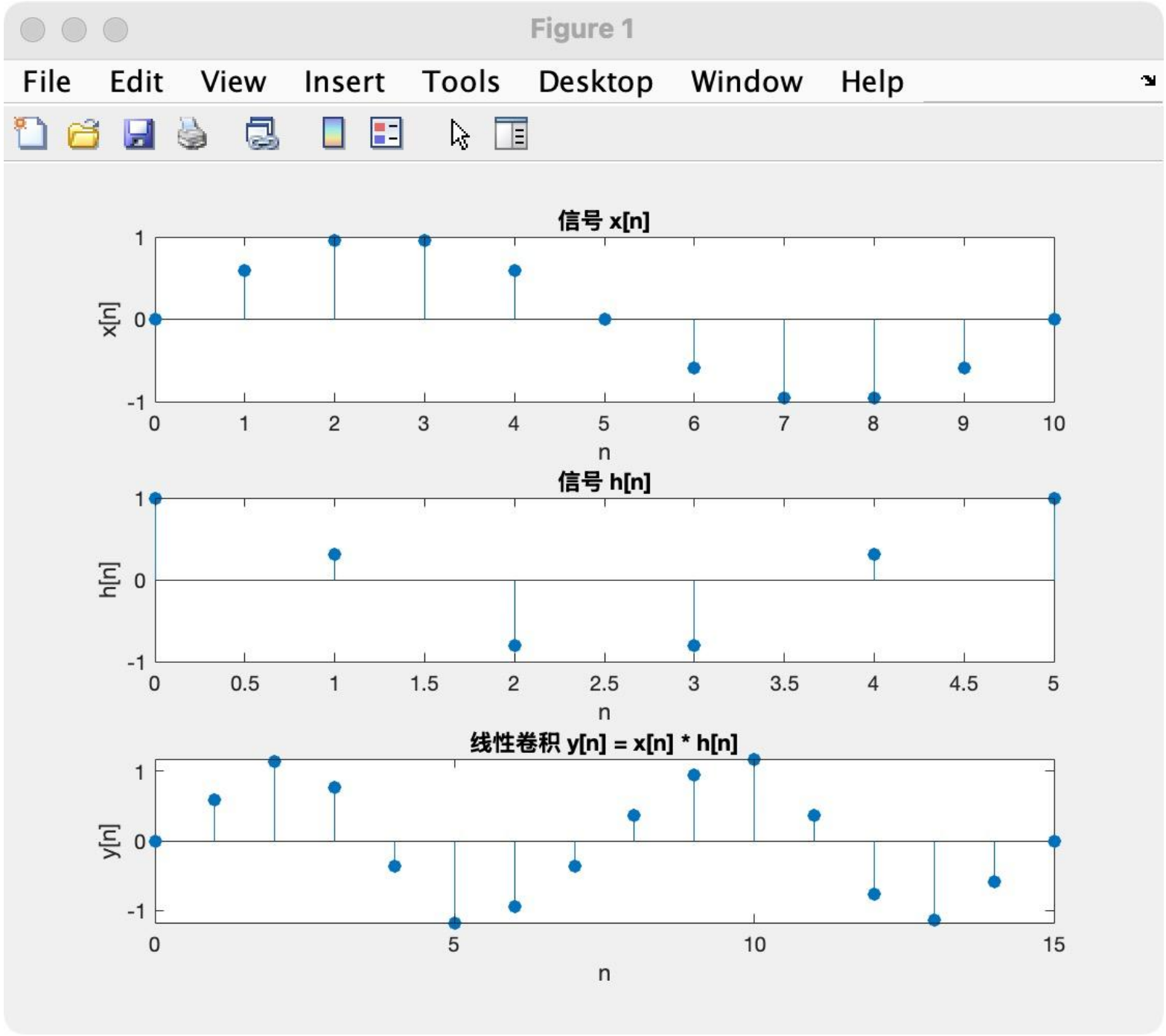
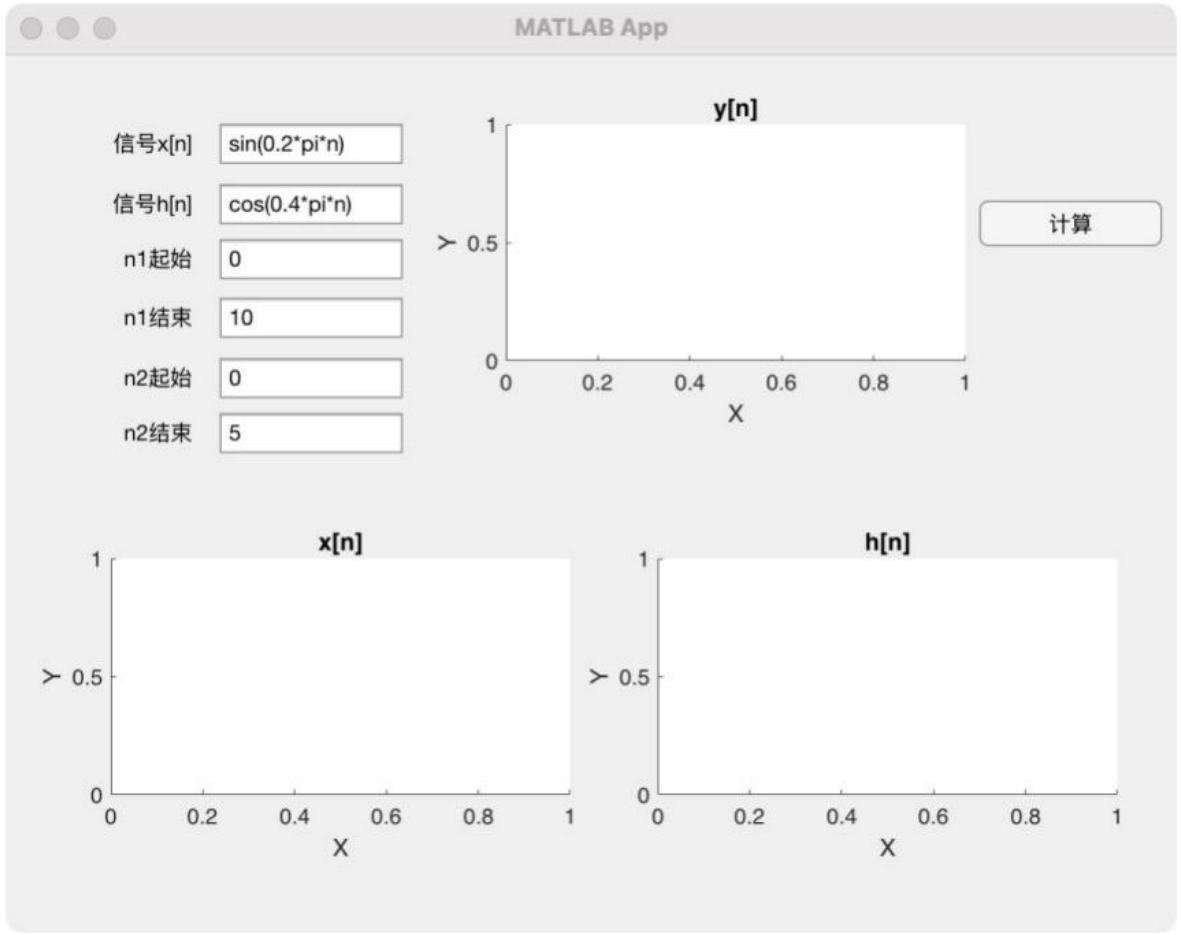
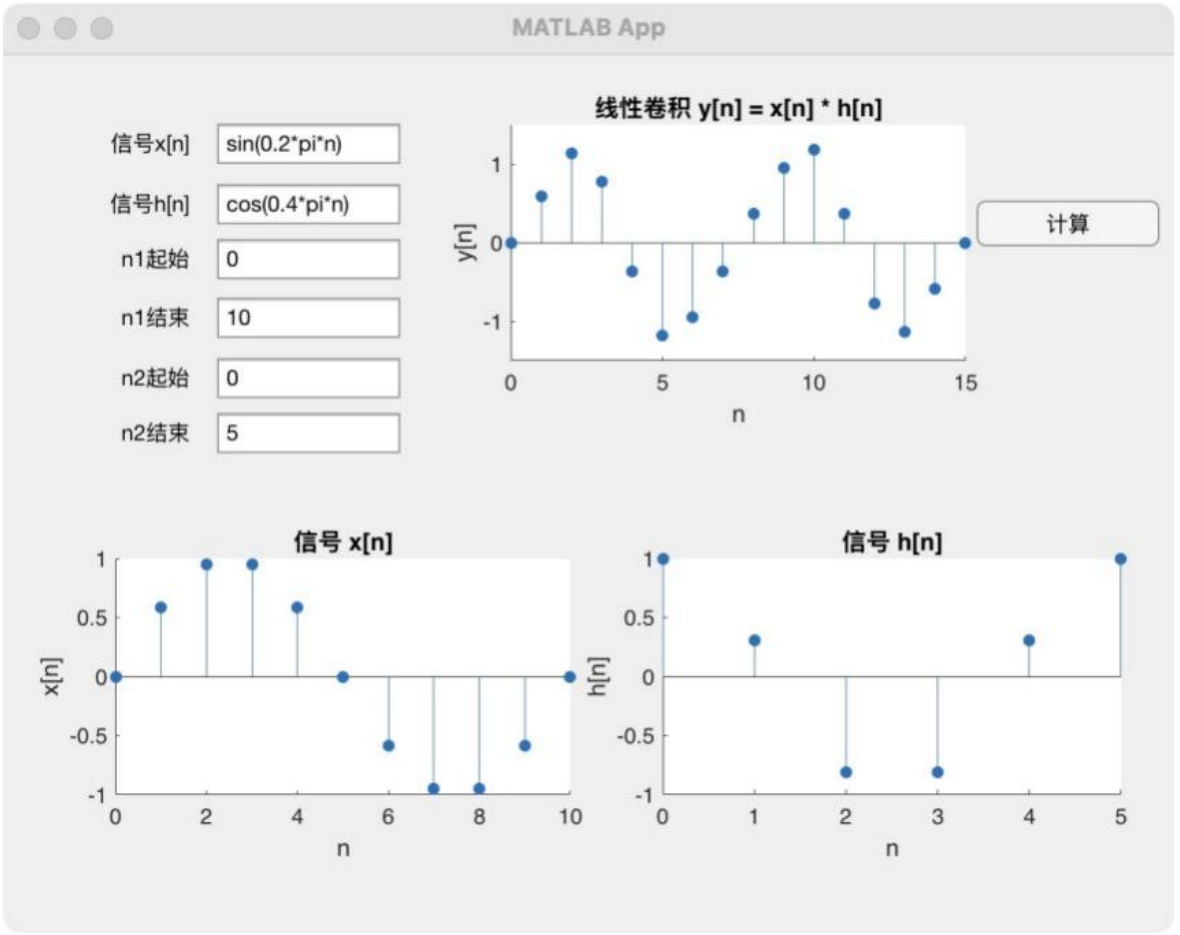


Figure 1
File Edit View Insert Tools Desktop Window Help
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n
线性卷积 y[n] = x[n] * h[n]
y[n]
n



Step 1:输入对应值

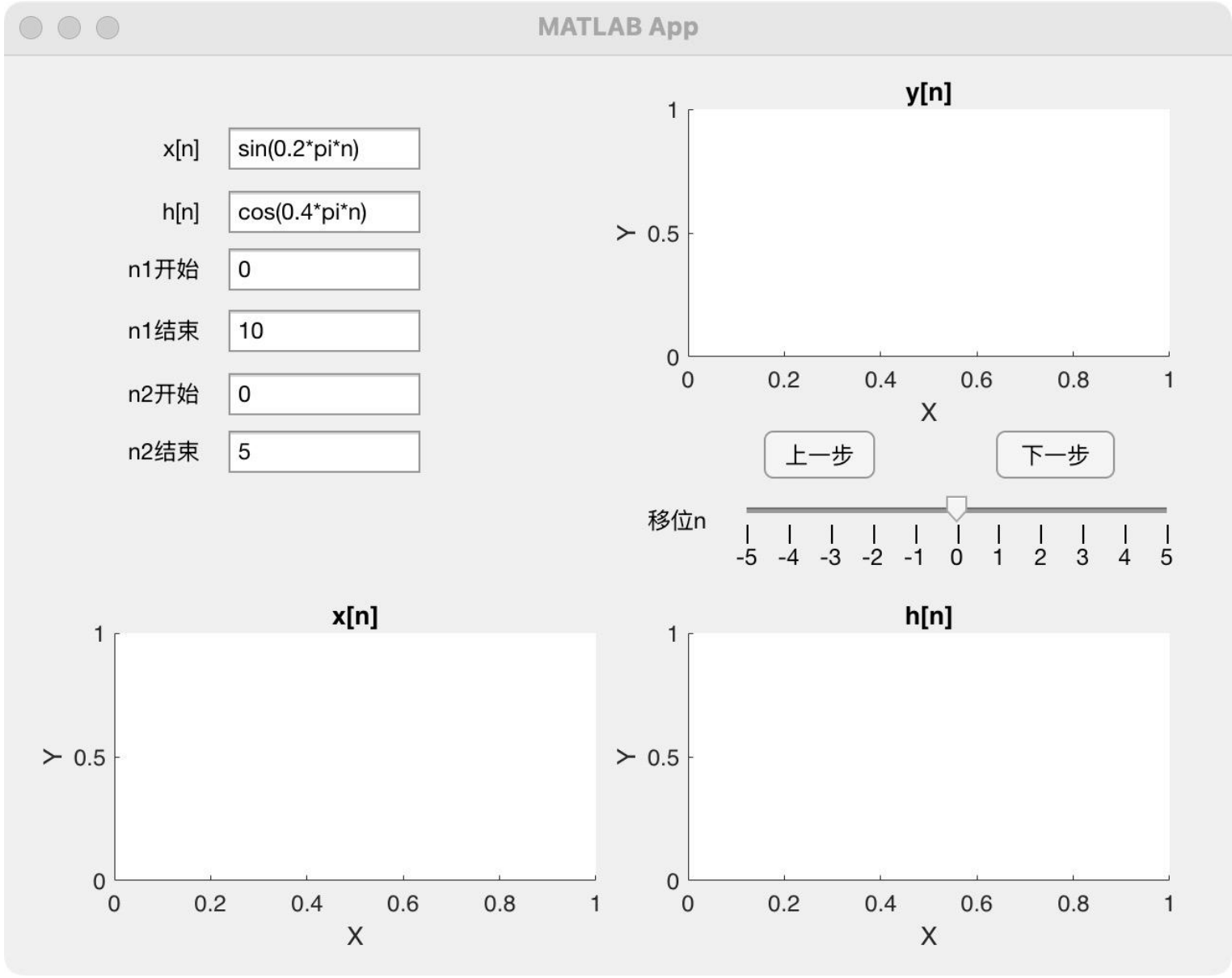


Step 2:单击计算按钮

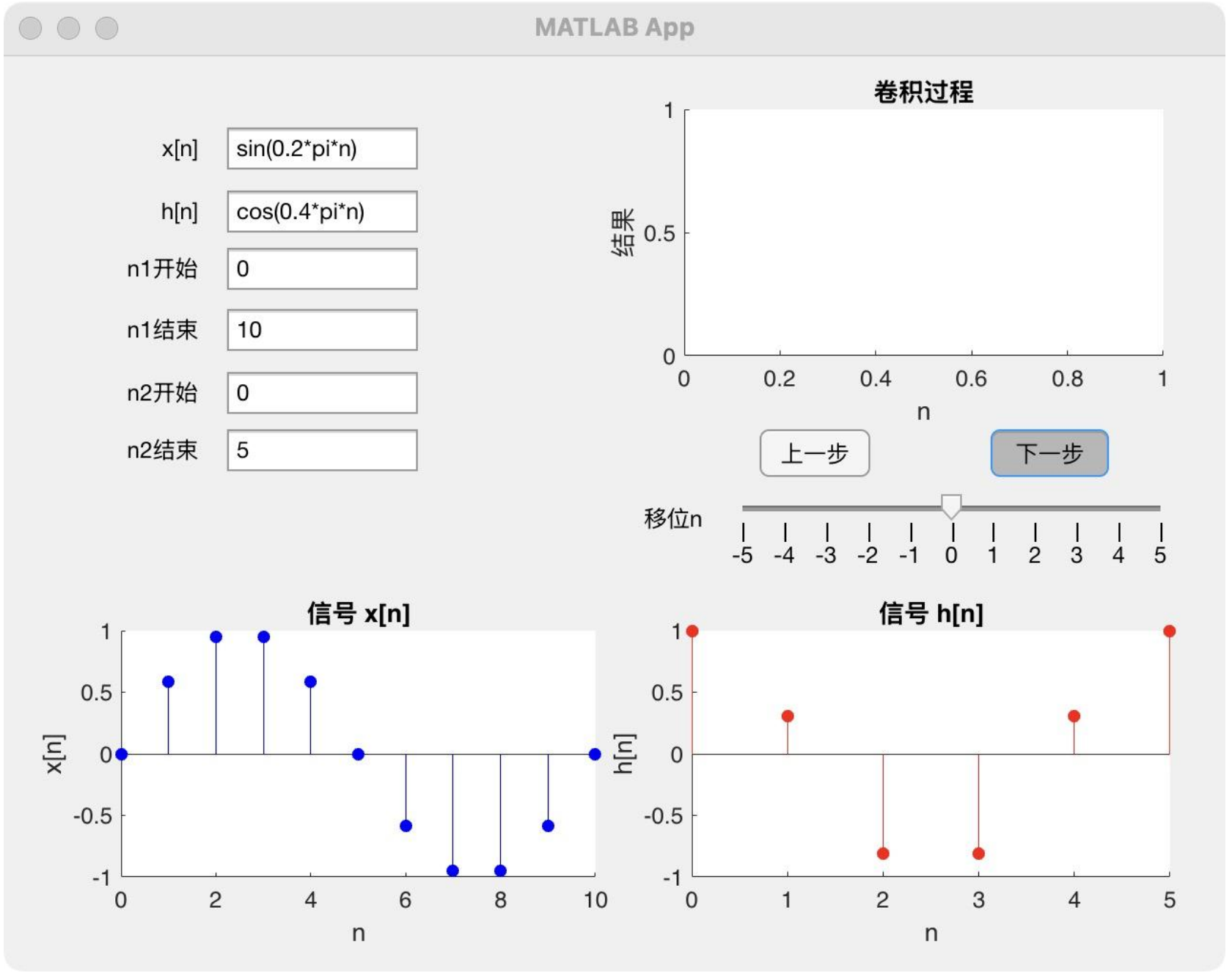
```
properties (Access = private)
    currentStep double = 0; % 当前步骤，初始值为 0
end
```

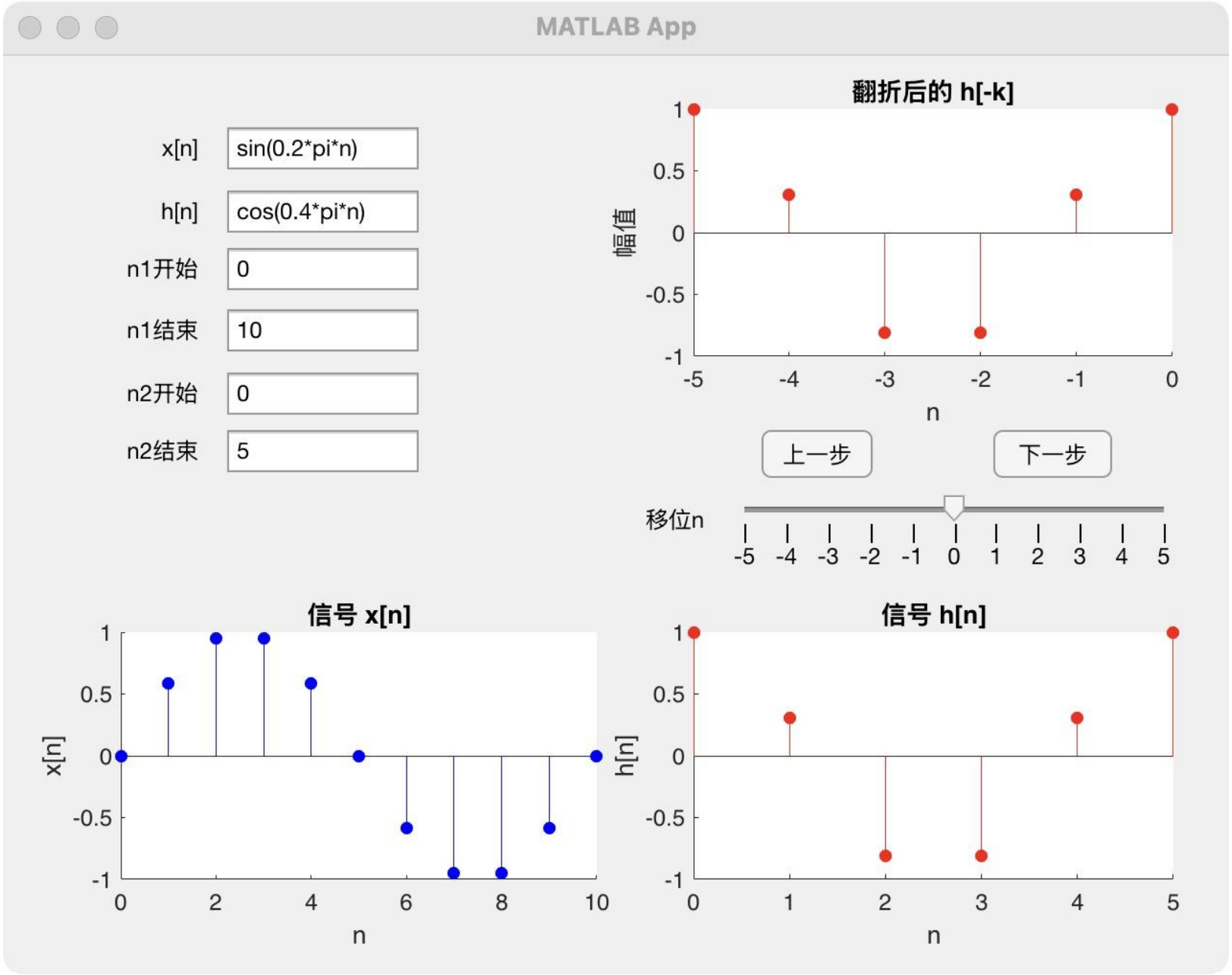
```
persistent x_signal h_signal n1_range n2_range n2_flip h_flip h_shi
% 第一步：绘制 x[n] 和 h[n]
if app.currentStep == 1
    % 获取时间范围
    n1_start = str2double(app.n1StartEditField.Value);
    n1_end = str2double(app.n1EndEditField.Value);
    n2_start = str2double(app.n2StartEditField.Value);
    n2_end = str2double(app.n2EndEditField.Value);

    % 定义时间轴
    n1_range = n1_start:n1_end;
    n2_range = n2_start:n2_end;

    % 计算信号 x[n] 和 h[n]
    n = n1_range;
    x_signal = eval(app.xExpressionEditField.Value);
    n = n2_range;
    h_signal = eval(app.hExpressionEditField.Value);
    % 绘制 x[n] 和 h[n]
    cla(app.UIAxes_x);
    stem(app.UIAxes_x, n1_range, x_signal, 'b', 'filled');
    title(app.UIAxes_x, '信号 x[n]');
    xlabel(app.UIAxes_x, 'n');
    ylabel(app.UIAxes_x, 'x[n]');

    cla(app.UIAxes_h);
    stem(app.UIAxes_h, n2_range, h_signal, 'r', 'filled');
    title(app.UIAxes_h, '信号 h[n]');
    xlabel(app.UIAxes_h, 'n');
    ylabel(app.UIAxes_h, 'h[n]');

    % 清空卷积结果图表
    cla(app.UIAxes_y);
    title(app.UIAxes_y, '卷积过程');
```

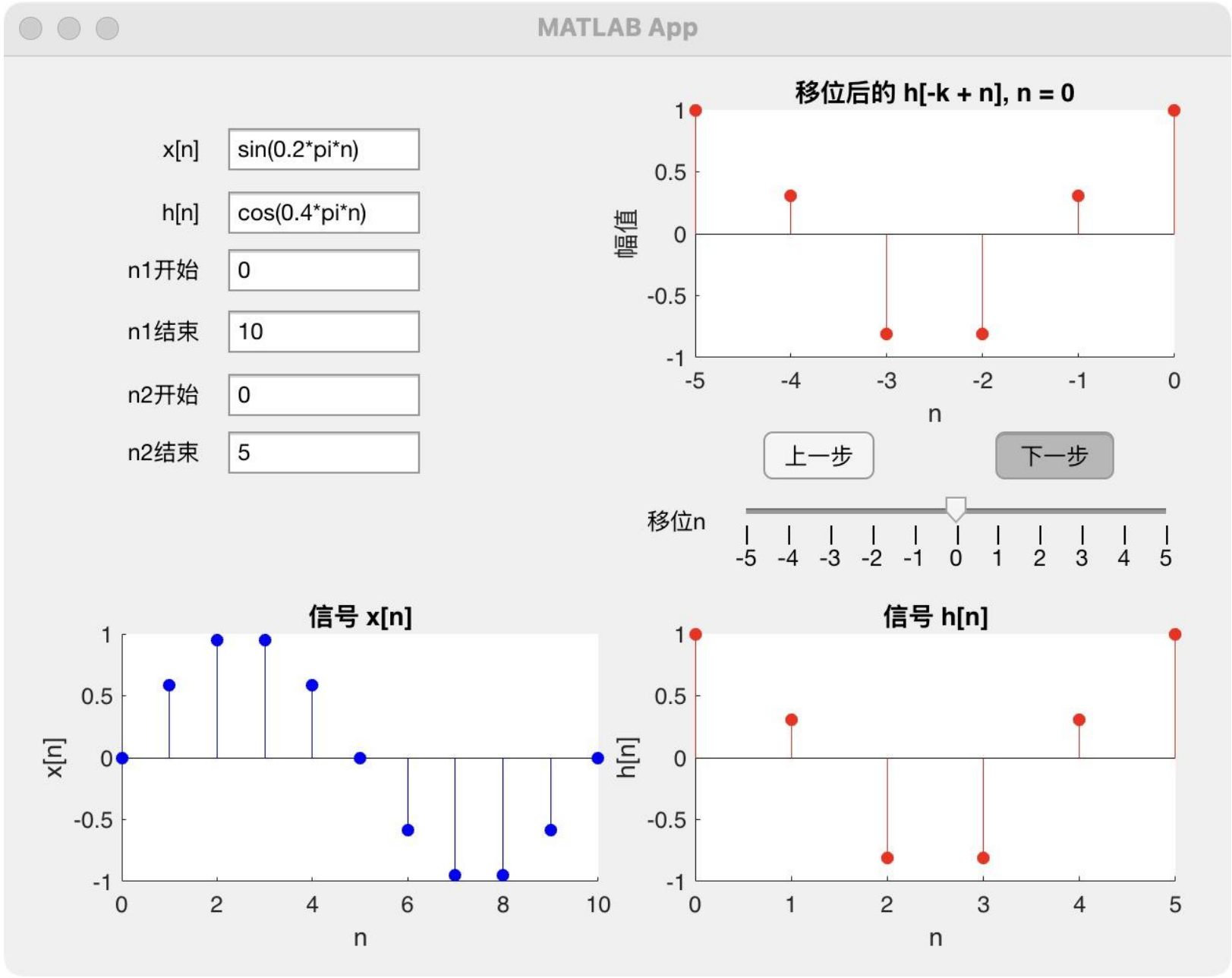
```
% 第二步：翻折 h[n]
elseif app.currentStep == 2
    h_flip = fliplr(h_signal); % 翻折 h[n]
    n2_flip = -fliplr(n2_range); % 翻折后的时间轴
    cla(app.UIAxes_y);
    stem(app.UIAxes_y, n2_flip, h_flip, 'r', 'filled');
    title(app.UIAxes_y, '翻折后的 h[-k]');
    xlabel(app.UIAxes_y, 'n');
    ylabel(app.UIAxes_y, '幅值');

% 第三步：移位 h[-k + n]（使用滑块控制移位量）
elseif app.currentStep == 3
    shift = round(app.nSlider.Value); % 获取滑块的移位量
    h_shift = n2_flip + shift;
    cla(app.UIAxes_y);
    stem(app.UIAxes_y, h_shift, h_flip, 'r', 'filled');
    title(app.UIAxes_y, ['移位后的 h[-k + n], n = ', num2str(shift)]);
    xlabel(app.UIAxes_y, 'n');
    ylabel(app.UIAxes_y, '幅值');

% 第四步：相乘并显示相乘结果
elseif app.currentStep == 4
    shift = round(app.nSlider.Value); % 获取滑块的移位量
    h_shift = n2_flip + shift;
    [n_overlap, idx_x, idx_h] = intersect(n1_range, h_shift); % 找到重叠区域
    x_overlap = x_signal(idx_x); % 获取 x 的重叠部分
    h_overlap = h_flip(idx_h);   % 获取 h 的重叠部分
    product = x_overlap .* h_overlap; % 相乘结果
    cla(app.UIAxes_y);
    stem(app.UIAxes_y, n_overlap, product, 'g', 'filled');
    title(app.UIAxes_y, ['相乘结果，移位 n = ', num2str(shift)]);
    xlabel(app.UIAxes_y, 'n');
    ylabel(app.UIAxes_y, '幅值');

      ylabel(app.UIAxes_y, '幅值');

  % 第五步：显示最终卷积结果
  elseif app.currentStep == 5
      y_conv = conv(x_signal, h_signal); % 计算卷积
      ny = (n1_start + n2_start):(n1_end + n2_end); % 卷积的时间范围
      cla(app.UIAxes_y);
      stem(app.UIAxes_y, ny, y_conv, 'm', 'filled'); % 显示卷积结果
      title(app.UIAxes_y, '卷积结果 y[n]');
      xlabel(app.UIAxes_y, 'n');
      ylabel(app.UIAxes_y, 'y[n]');
  end
```

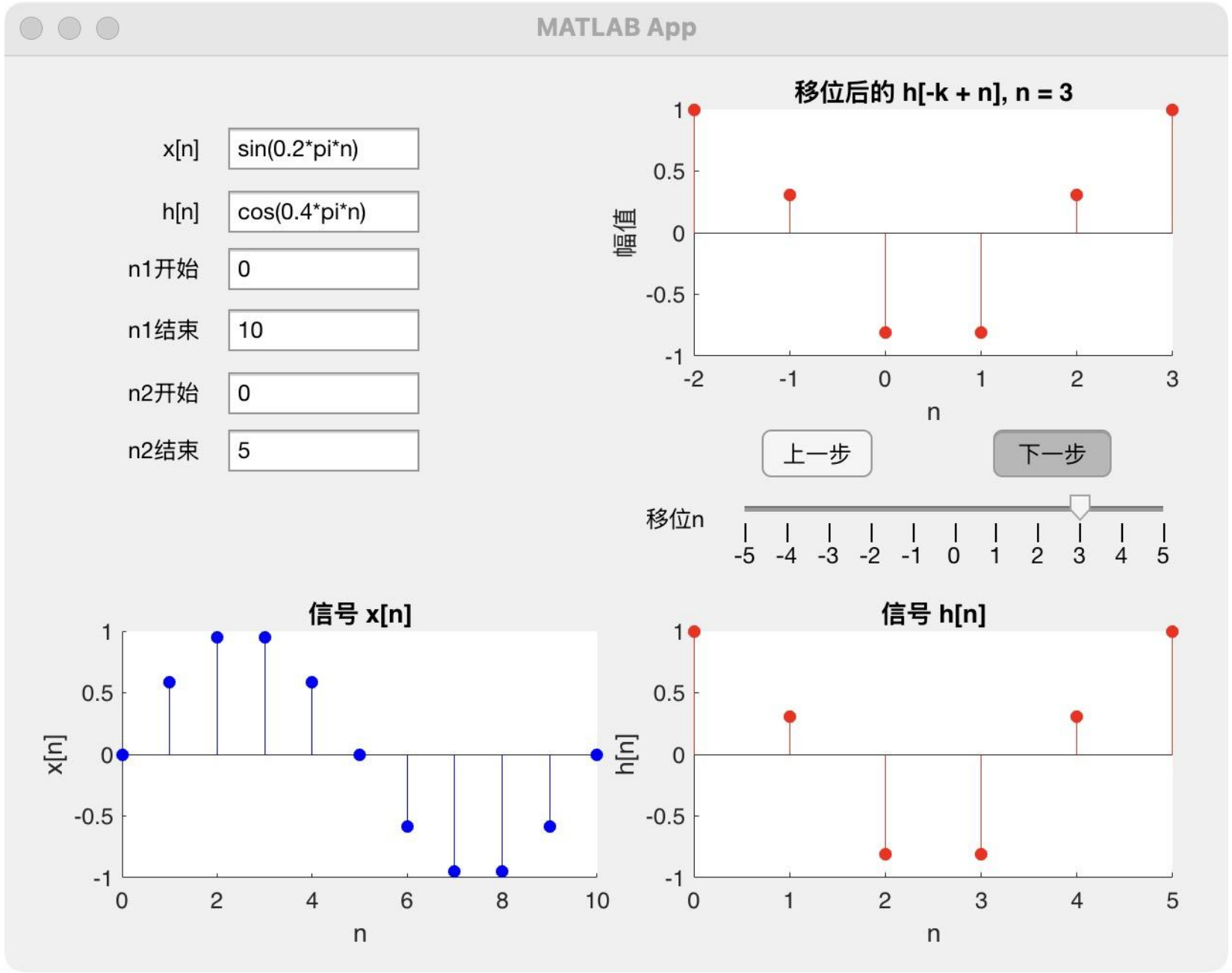

MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
y[n]
Y
X
上一步
下一步
移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
x[n]
h[n]

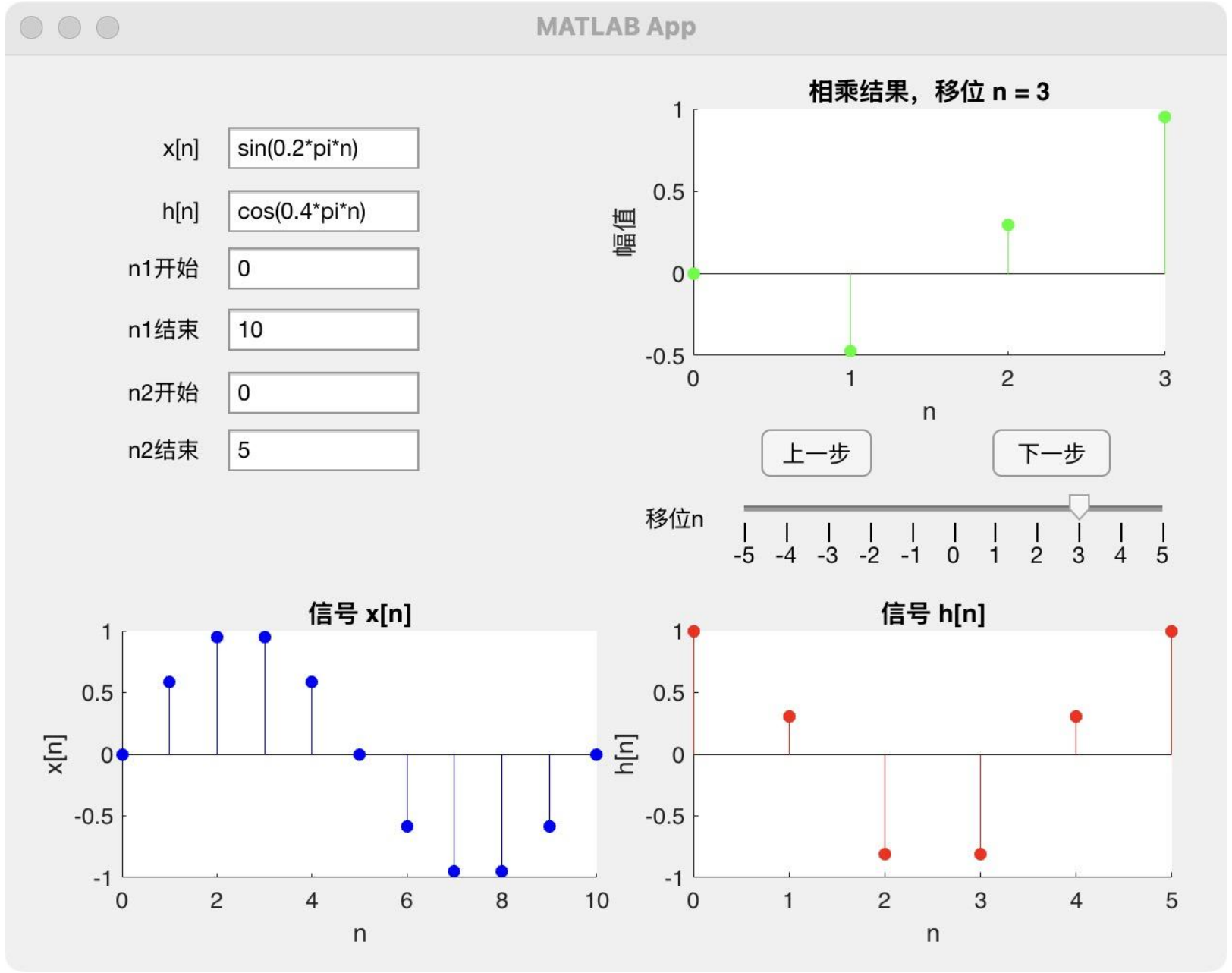
输入值

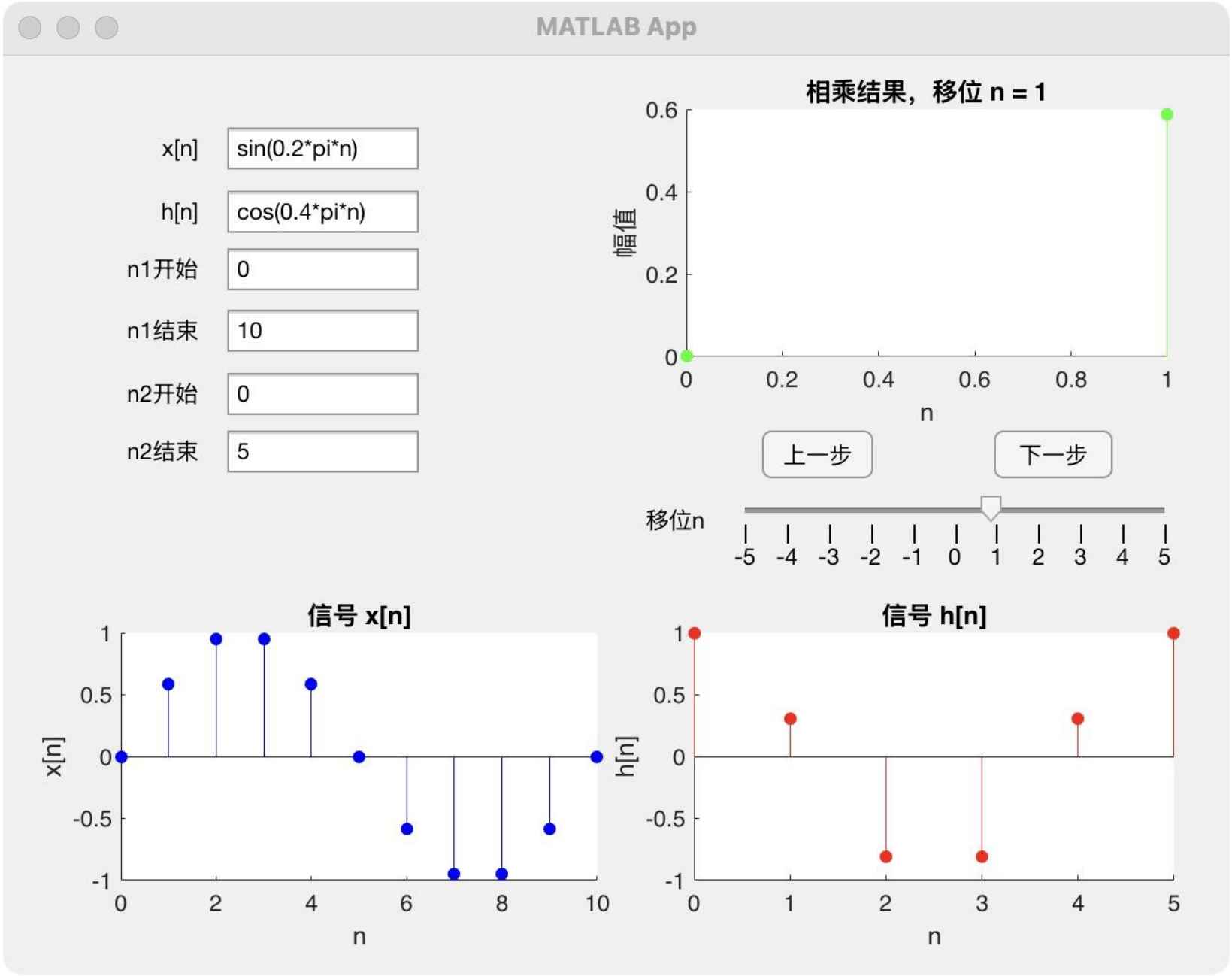

MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
卷积过程
结果
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n

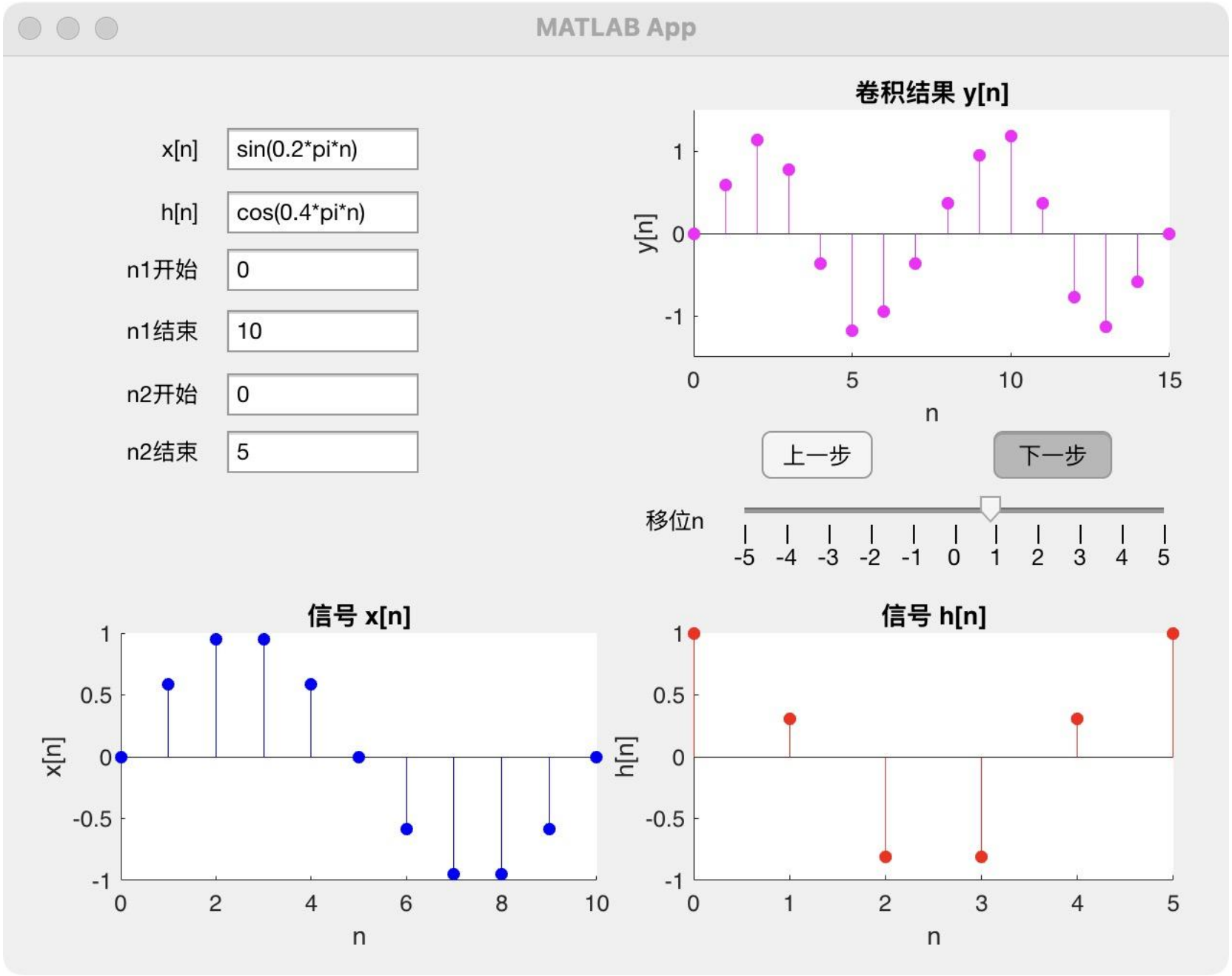

MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
翻折后的 h[-k]
幅值
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n


MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
移位后的 h[-k + n], n = 0
幅值
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n


MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
移位后的 h[-k + n], n = 3
幅值
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n


MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
相乘结果，移位 n = 3
幅值
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n


MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
相乘结果，移位 n = 1
幅值
n
上一步
下一步
移位n
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n


MATLAB App
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
卷积结果 y[n]
y[n]
n
上一步
下一步
移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
信号 x[n]
x[n]
n
信号 h[n]
h[n]
n

圆周卷积

- 圆周卷积的时域过程 设 x(n) 和 h(n) 为 N 点有限长序列，其圆周卷积定义为：

$$y(n) = \left[\sum_{m=0}^{N-1} x_1(m)x_2((n-m))_N\right] R_N(n)$$

- 圆周卷积计算过程包括以下步骤：
- 1. 补零: 将序列 x1(n) 和 x2(n) 补零至 N 点长度。
- 2. 周期延拓: x2(m) 进行周期延拓，得到 $x2((m))_N$。
- 3. 圆周翻褶: 将 $x2((m))_N$ 进行翻转，得到 $x2((-m))_N$。
- 4. 圆周移位: 将 $x2((-m))_N$ 按照 n 进行圆周平移，得到 $x2((n-m))_N R_N(m)$。
- 5. 相乘相加: 计算 $x1(m)x2((n-m))_N R_N(m)$ 并对所有 m 求和。

补零

圆周翻转移

圆周移位

圆周移位

相乘
相加
⇒ 圆周卷积

圆周卷积的频域过程

- 圆周卷积在频域中也有相应的计算方法。
- 通过离散傅里叶变换（DFT）将时域卷积转换为频域 乘积:
- DF T[x1(n)] = X1(k), DF T[x2(n)] = X2(k)
- 卷积的频域乘积为： Y (k) = X1(k) · X2(k)
- 通过逆傅里叶变换得到时域卷积结果:

$$y(n) = IDFT[Y(k)] = \left[\sum_{m=0}^{N-1} x_1(m)x_2((n-m))_N R_N(n)\right]$$

圆周卷积与线性卷积关系

- 当圆周卷积长度L>=N+M-1时，没有混叠，圆周卷积的前N+M-1个值为有效值，且与线性卷积相同，其余为0
- 当圆周卷积长度L<N+M-1时，产生混叠，圆周卷积的前(N+M-1)-L个值有混叠

```
  properties (Access = private)
      currentStep double = 0; % 当前步骤，初始值为 0
      inTimeDomain logical = true; % 默认为时域操作
  end
```

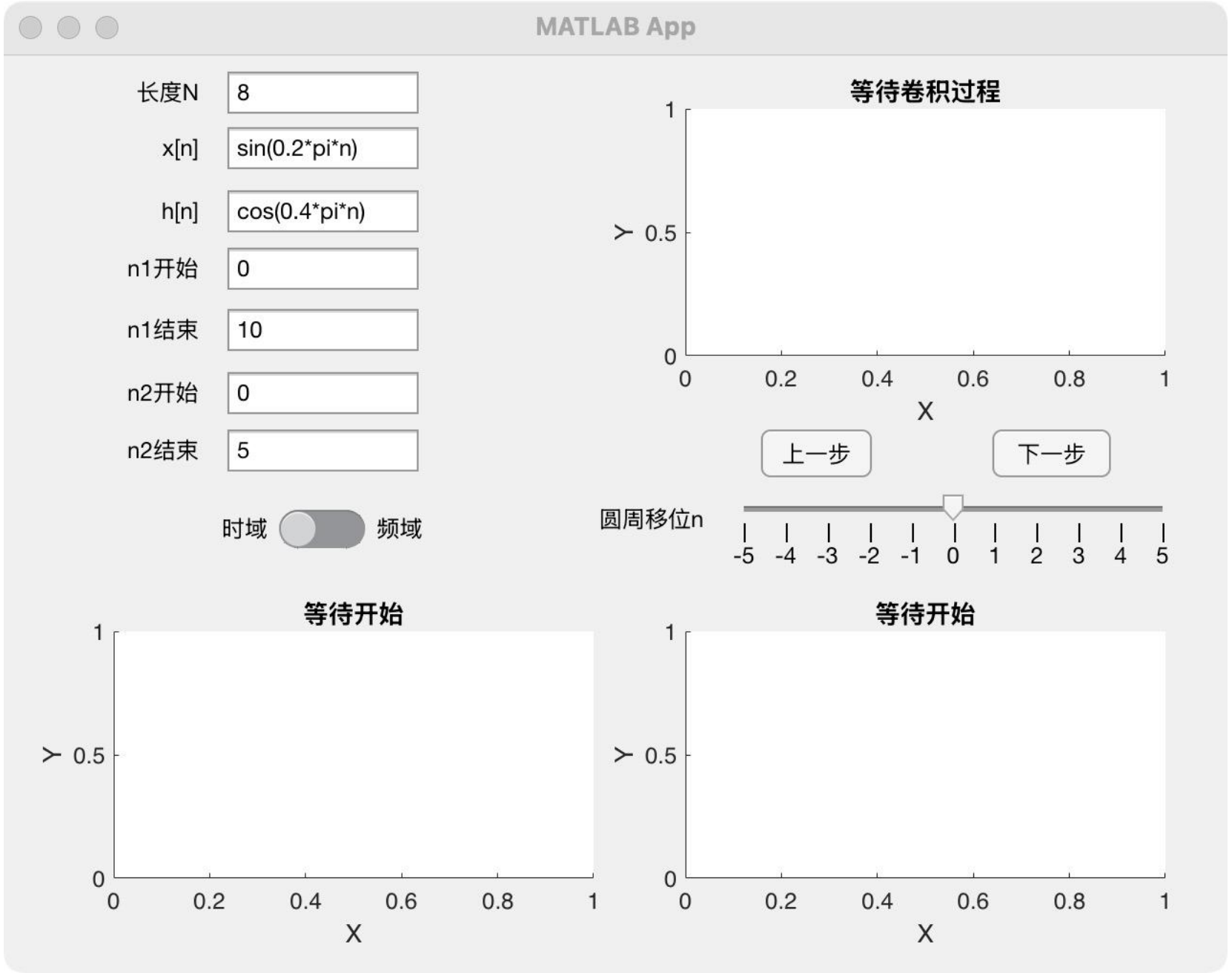
```
persistent x_signal h_signal n1_range n2_range h_flip h_shift n1_start n1_en

% 获取 N 的值（圆周长度），来自 NEditField
N = str2double(app.NEditField.Value);

% 初始状态，什么都不显示
if app.currentStep == 0
    cla(app.UIAxes_x);
    cla(app.UIAxes_h);
    cla(app.UIAxes_y);
    title(app.UIAxes_x, '等待开始');
    title(app.UIAxes_h, '等待开始');
    title(app.UIAxes_y, '等待卷积过程');

% 第一步：绘制原始信号
elseif app.currentStep == 1
    % 获取时间范围和信号
    n1_start = str2double(app.n1StartEditField.Value);
    n1_end = str2double(app.n1EndEditField.Value);
    n2_start = str2double(app.n2StartEditField.Value);
    n2_end = str2double(app.n2EndEditField.Value);
    n1_range = n1_start:n1_end;
    n2_range = n2_start:n2_end;

    % 计算信号 x[n] 和 h[n]
    n = n1_range;
    x_signal = eval(app.xExpressionEditField.Value); % 计算 x[n] 的表达式
    n = n2_range;
    h_signal = eval(app.hExpressionEditField.Value); % 计算 h[n] 的表达式

    % 绘制原始信号
    cla(app.UIAxes_x);
    stem(app.UIAxes_x, n1_range, x_signal, 'b', 'filled');
    title(app.UIAxes_x, '原始信号 x[n]');
    xlabel(app.UIAxes_x, 'n');
```

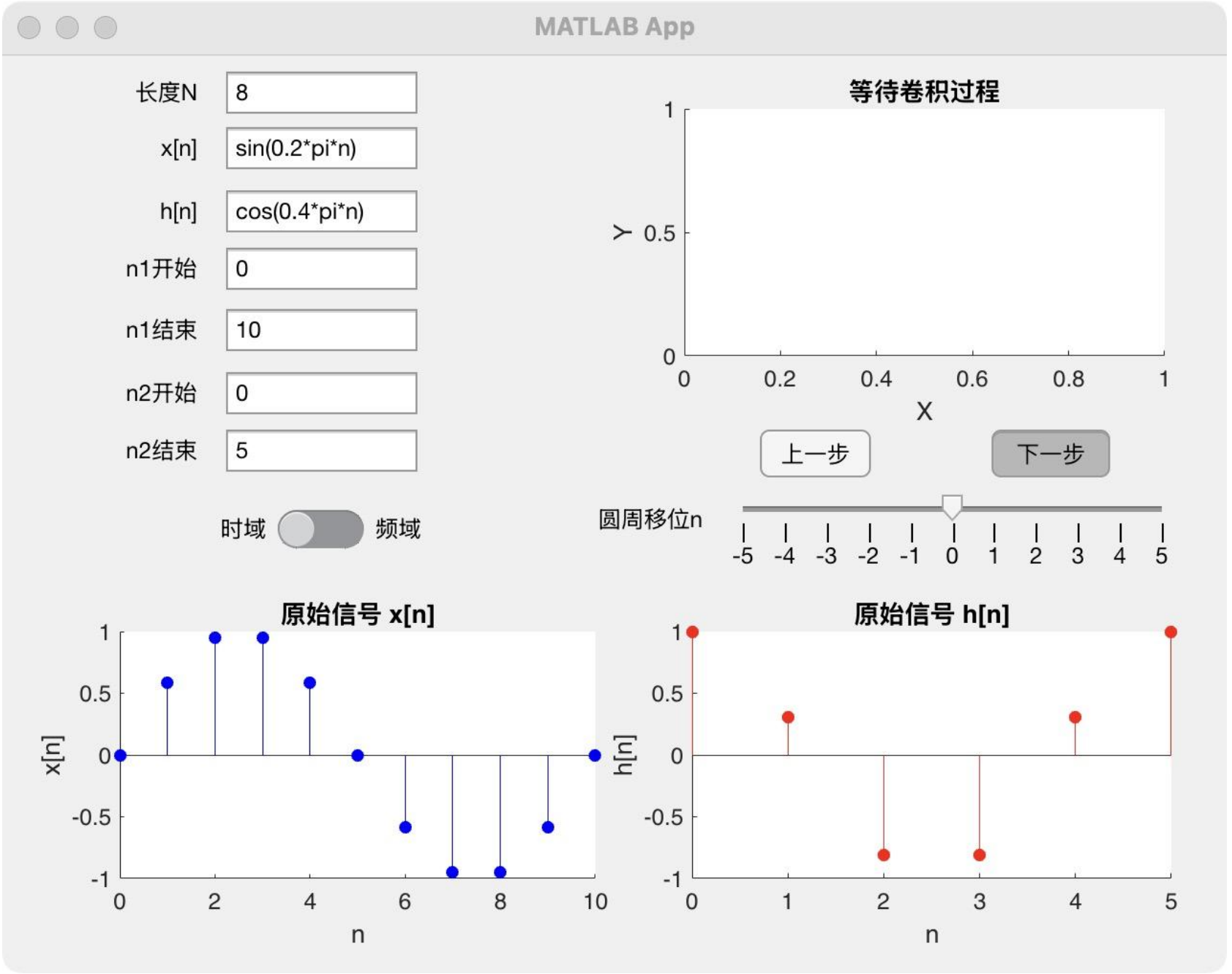
```
% 第二步：时域卷积 - 补零/截断+翻褶 或 频域卷积 - 计算 DFT
elseif app.currentStep == 2
    if app.inTimeDomain
        % 时域卷积 - 先补零或截断，再翻褶
        % 对 x_signal 和 h_signal 进行补零或截断到长度 N
        if length(x_signal) > N
            x_signal = x_signal(1:N);  % 截断 x_signal 到长度 N
        else
            x_signal = [x_signal zeros(1, N - length(x_signal))];  % 补零
        end

        if length(h_signal) > N
            h_signal = h_signal(1:N);  % 截断 h_signal 到长度 N
        else
            h_signal = [h_signal zeros(1, N - length(h_signal))];  % 补零
        end

        % 翻褶 h[n]
        h_flip = fliplr(h_signal);

        % 绘制补零/截断后的 x[n] 和 h[n]
        cla(app.UIAxes_x);
        stem(app.UIAxes_x, 0:N-1, x_signal, 'b', 'filled');
        title(app.UIAxes_x, '补零/截断后的 x[n]');
        xlabel(app.UIAxes_x, 'n');
        ylabel(app.UIAxes_x, 'x[n]');

        cla(app.UIAxes_h);
        stem(app.UIAxes_h, 0:N-1, h_flip, 'r', 'filled');
        title(app.UIAxes_h, '补零/截断并翻褶后的 h[n]');
        xlabel(app.UIAxes_h, 'n');
        ylabel(app.UIAxes_h, 'h[n]');
    else
        % 频域卷积 - 计算 DFT
```

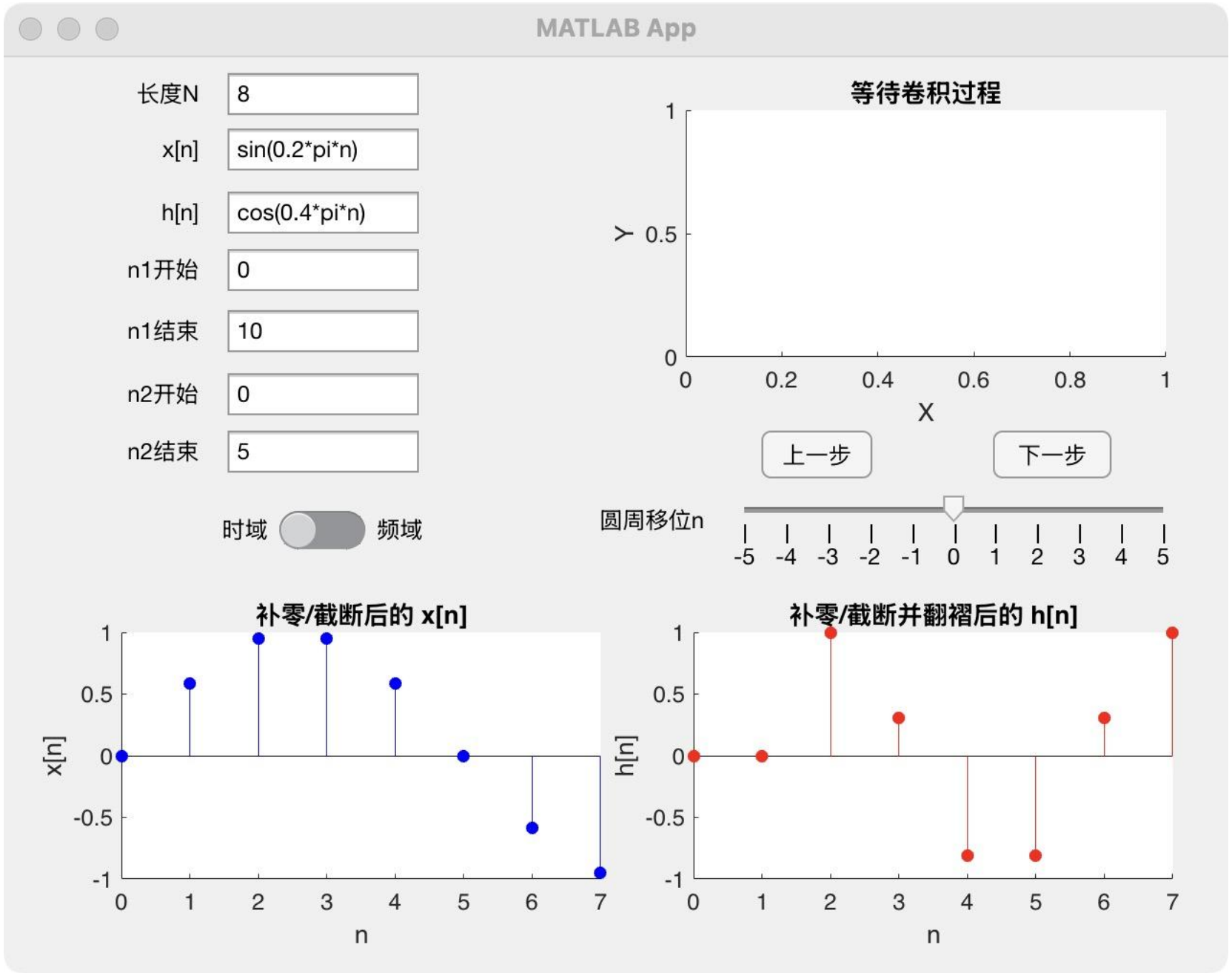
```
        xlabel(app.UIAxes_h, 'n');
        ylabel(app.UIAxes_h, 'h[n]');
    else
        % 频域卷积 - 计算 DFT
        X = fft(x_signal, N);
        H = fft(h_signal, N);

        % 绘制 DFT 结果
        cla(app.UIAxes_x);
        stem(app.UIAxes_x, 0:length(X)-1, abs(X), 'b', 'filled');  % 显示频域的幅度
        title(app.UIAxes_x, 'DFT of x[n]');
        xlabel(app.UIAxes_x, 'k');
        ylabel(app.UIAxes_x, '|X[k]|');

        cla(app.UIAxes_h);
        stem(app.UIAxes_h, 0:length(H)-1, abs(H), 'r', 'filled');  % 显示频域的幅度
        title(app.UIAxes_h, 'DFT of h[n]');
        xlabel(app.UIAxes_h, 'k');
        ylabel(app.UIAxes_h, '|H[k]|');
    end

% 第三步：时域卷积 - 圆周移位 或 频域卷积 - 点乘
elseif app.currentStep == 3
    if app.inTimeDomain
        % 时域卷积 - 圆周移位
        shift = round(app.nSlider.Value);
        h_shift = circshift(h_flip, shift);

        % 绘制移位后的 h[n]
        cla(app.UIAxes_y);
        stem(app.UIAxes_y, 0:length(h_shift)-1, h_shift, 'r', 'filled');
        title(app.UIAxes_y, ['圆周移位后的 h[n], 移位 n = ', num2str(shift)]);
        xlabel(app.UIAxes_y, 'n');
        ylabel(app.UIAxes_y, 'h[n]');
    else
        % 频域卷积 - 点乘
```

```
% 第四步：时域卷积 - 相乘 或 频域卷积 - IFFT
elseif app.currentStep == 4
    if app.inTimeDomain
        % 时域卷积 - 相乘
        shift = round(app.nSlider.Value);
        h_shift = circshift(h_flip, shift);

        % 检查 x_signal 和 h_shift 的长度是否相等
        if length(x_signal) == length(h_shift)
            product = x_signal .* h_shift;
        else
            error('x_signal 和 h_shift 的长度不一致，无法相乘');
        end

        % 绘制相乘结果
        cla(app.UIAxes_y);
        stem(app.UIAxes_y, 0:length(product)-1, product, 'g', 'filled');
        title(app.UIAxes_y, 'x[n] 和 h[n] 的相乘结果');
        xlabel(app.UIAxes_y, 'n');
        ylabel(app.UIAxes_y, '乘积');
    else
        % 频域卷积 - 计算 IFFT
        X = fft(x_signal, N);
        H = fft(h_signal, N);
        Y_freq = X .* H;
        y_circular = ifft(Y_freq, N);

        % 绘制 IFFT 结果
        cla(app.UIAxes_y);
        stem(app.UIAxes_y, 0:length(y_circular)-1, y_circular, 'm', 'filled');
        title(app.UIAxes_y, '圆周卷积结果（频域转换回时域）');
        xlabel(app.UIAxes_y, 'n');
        ylabel(app.UIAxes_y, 'y[n]');
    end
% 第五步：显示卷积结果
elseif app.currentStep == 5
```

```
    end
% 第五步：显示卷积结果
elseif app.currentStep == 5
    if app.inTimeDomain
        % 时域卷积
        y_circular = cconv(x_signal, h_signal, N);

        cla(app.UIAxes_y);
        stem(app.UIAxes_y, 0:length(y_circular)-1, y_circular, 'm', 'filled');
        title(app.UIAxes_y, '圆周卷积结果（时域）');
        xlabel(app.UIAxes_y, 'n');
        ylabel(app.UIAxes_y, 'y[n]');
    else
        % 频域卷积
        X = fft(x_signal, N);
        H = fft(h_signal, N);
        Y_freq = X .* H;
        y_circular = ifft(Y_freq, N);

        cla(app.UIAxes_y);
        stem(app.UIAxes_y, 0:length(y_circular)-1, y_circular, 'm', 'filled');
        title(app.UIAxes_y, '圆周卷积结果（频域）');
        xlabel(app.UIAxes_y, 'n');
        ylabel(app.UIAxes_y, 'y[n]');
    end
end
```


MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
Y
X
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
等待开始
等待开始


MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
Y
X
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
原始信号 x[n]
x[n]
n
原始信号 h[n]
h[n]
n


MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
Y
X
上一步
下一步
圆周移位n
补零/截断后的 x[n]
x[n]
n
补零/截断并翻褶后的 h[n]
h[n]
n

MATLAB App

长度N 8

x[n] sin(0.2*pi*n)

h[n] cos(0.4*pi*n)

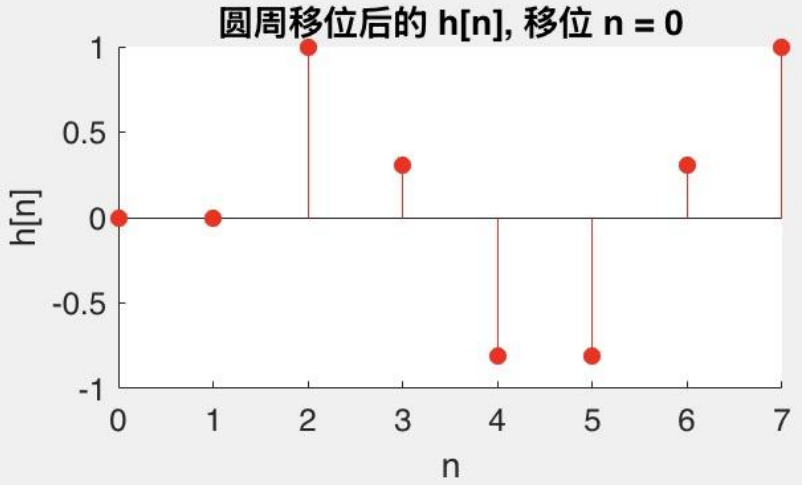
n1开始 0

n1结束 10

n2开始 0

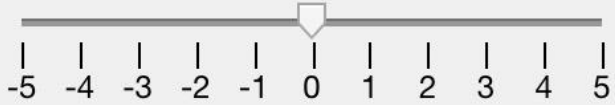
n2结束 5

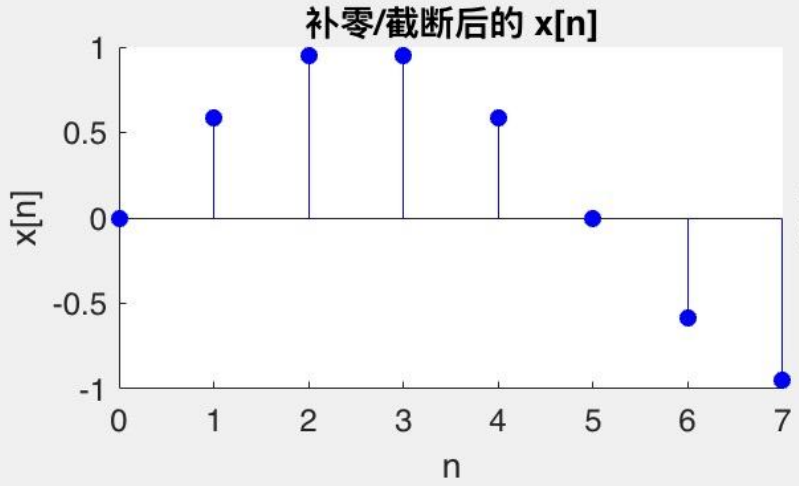
时域 频域

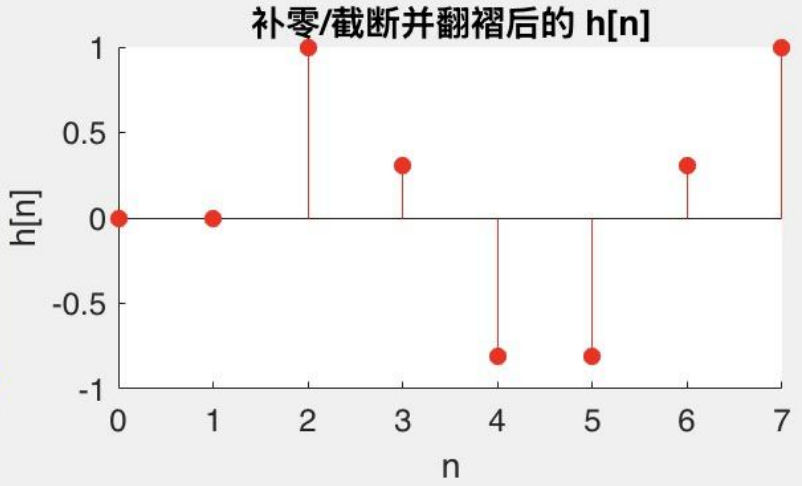


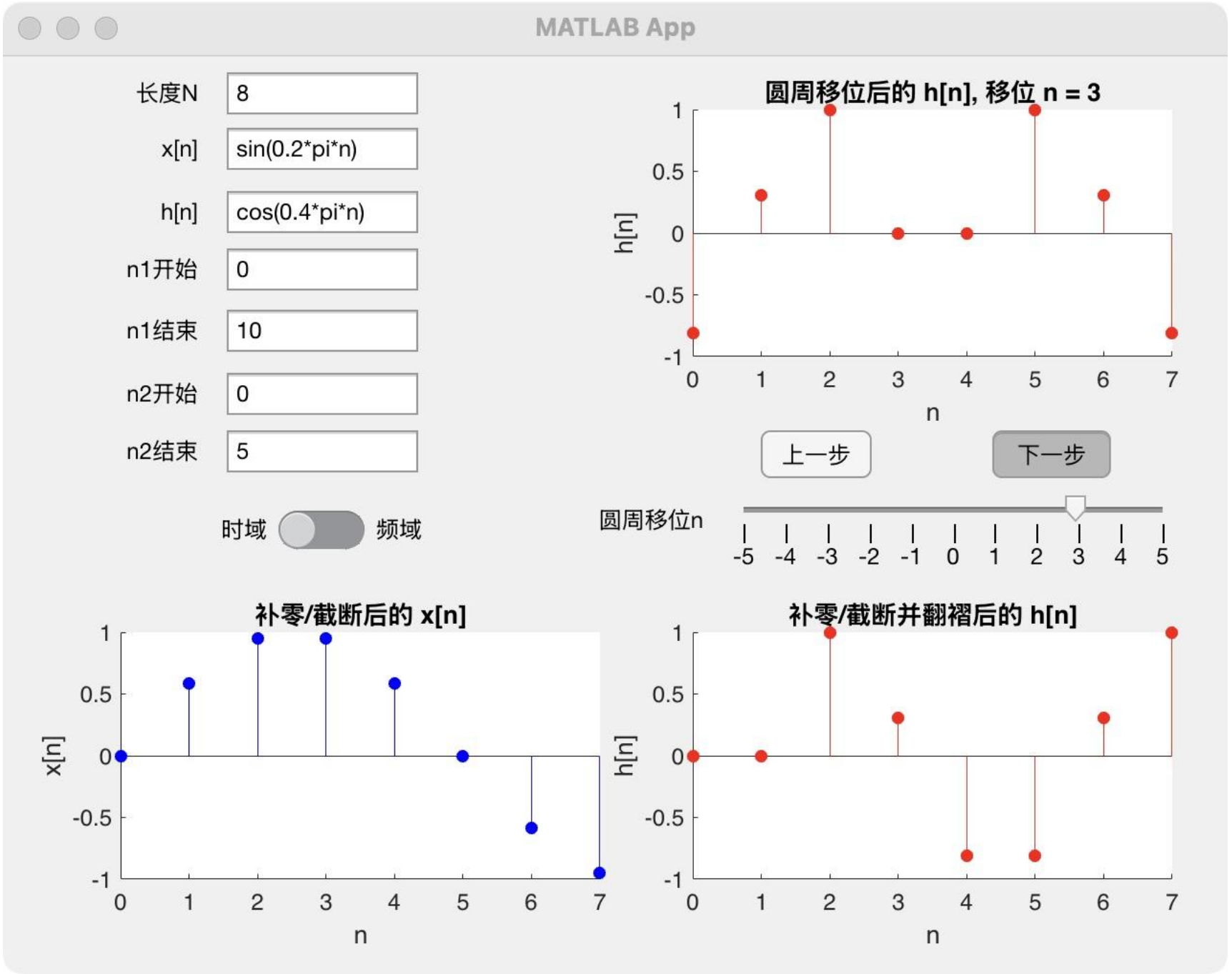
上一步 下一步

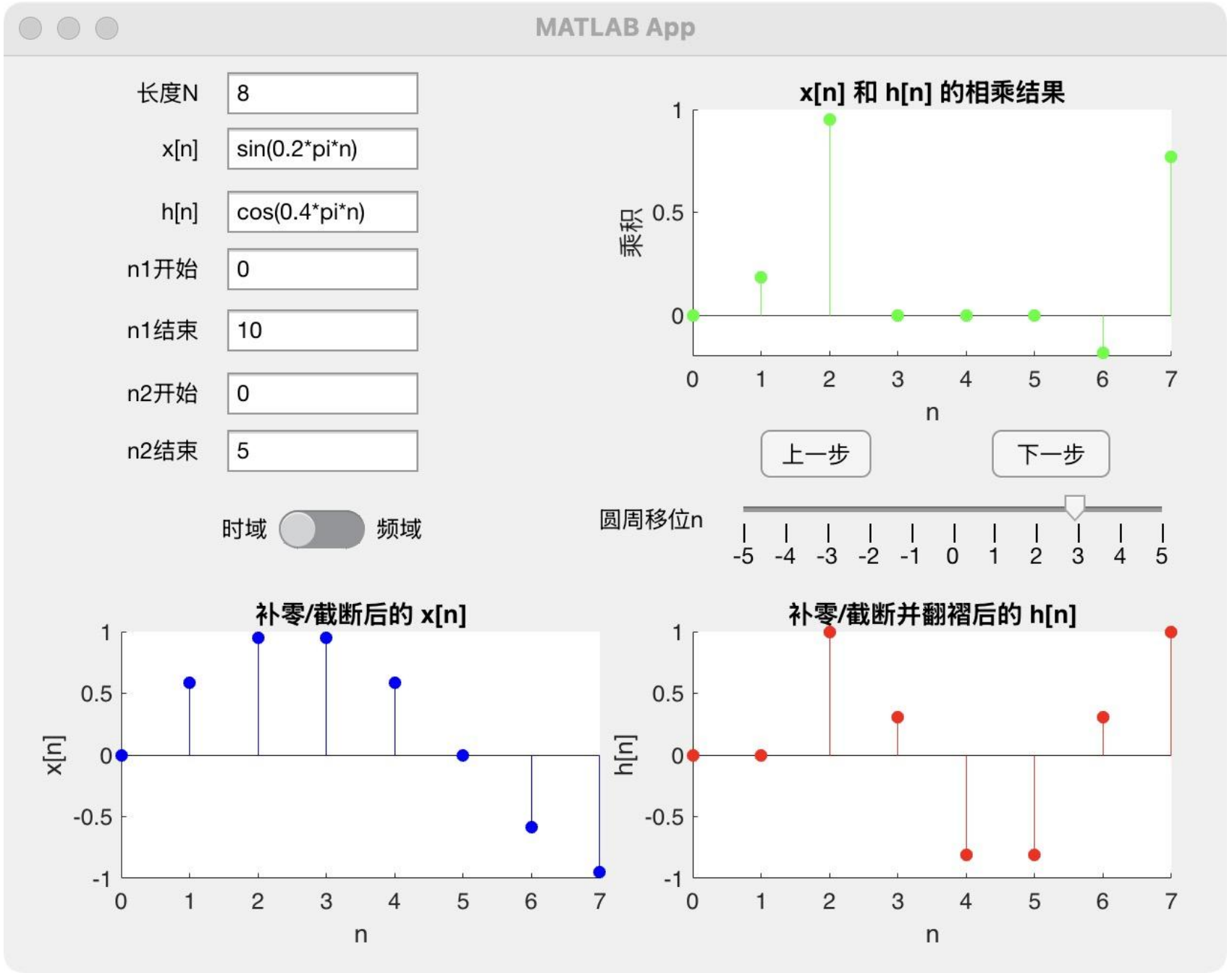
圆周移位n

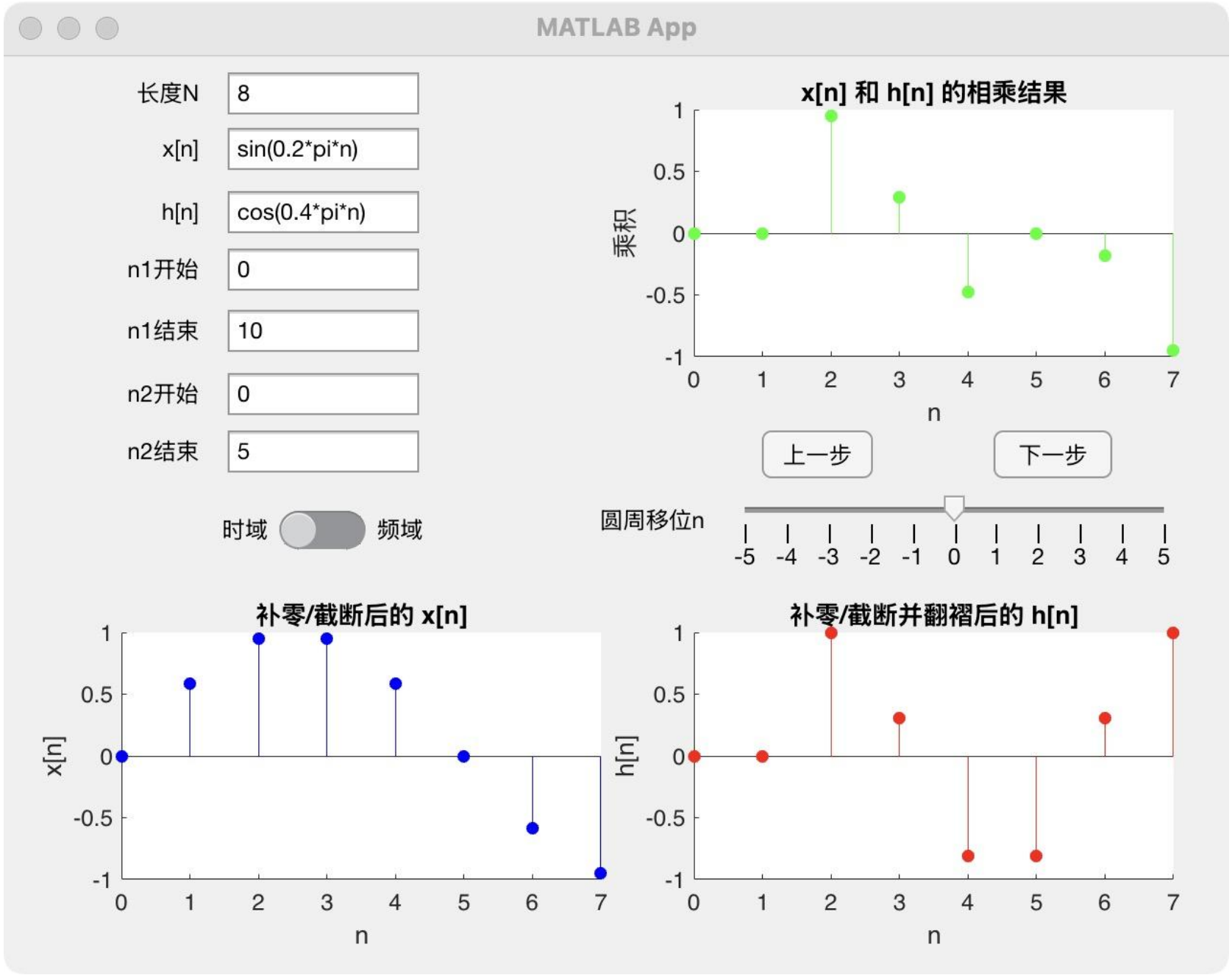


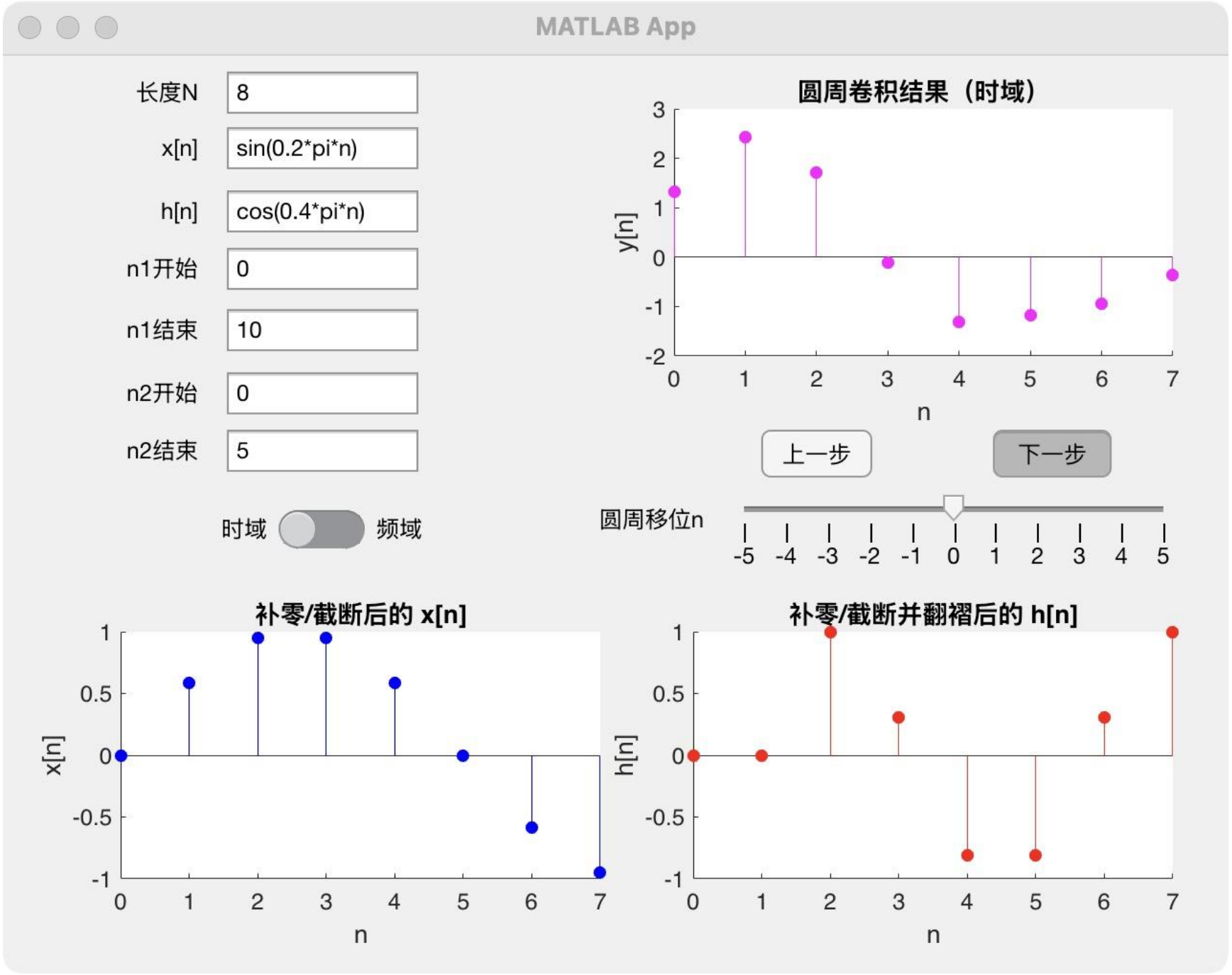


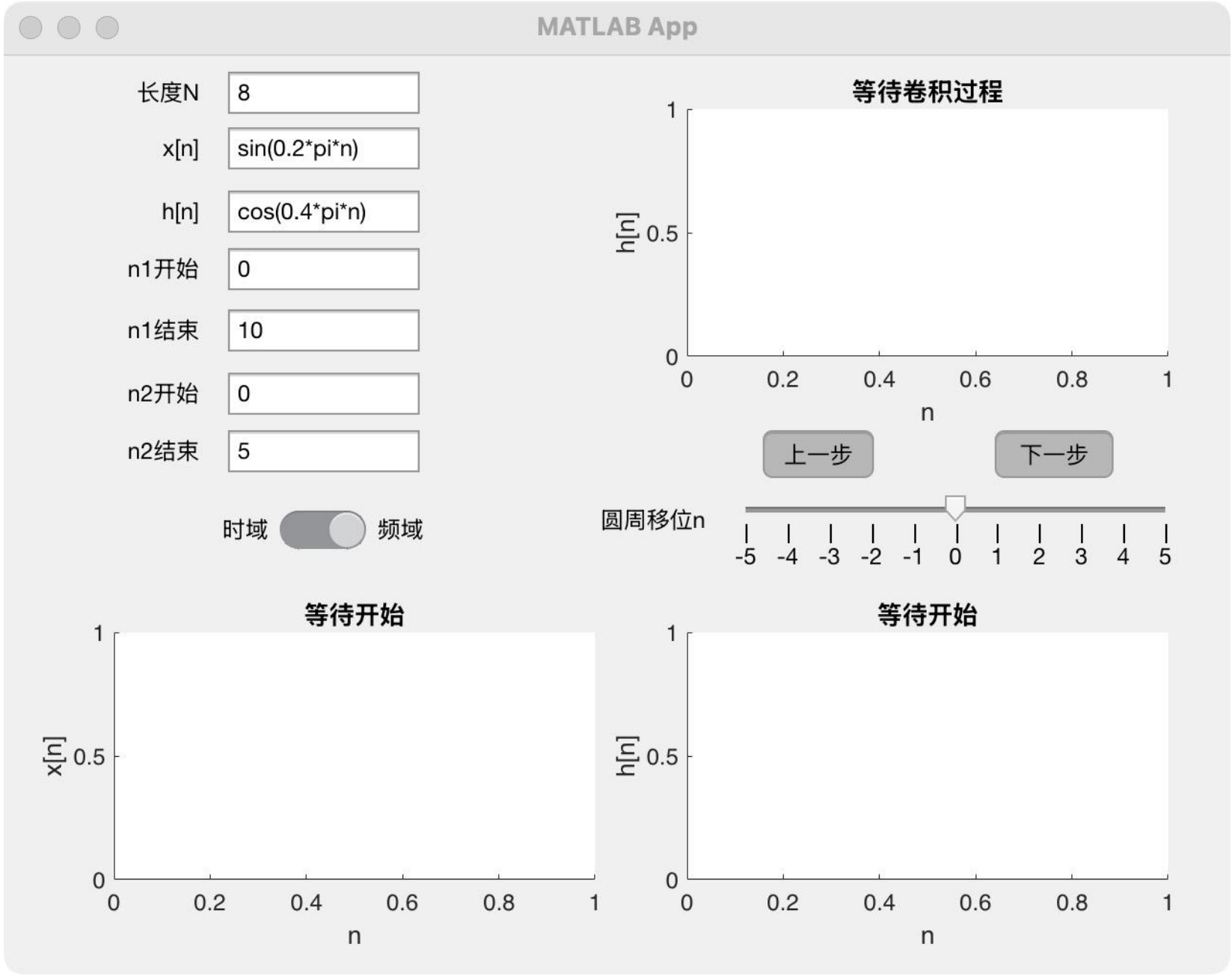


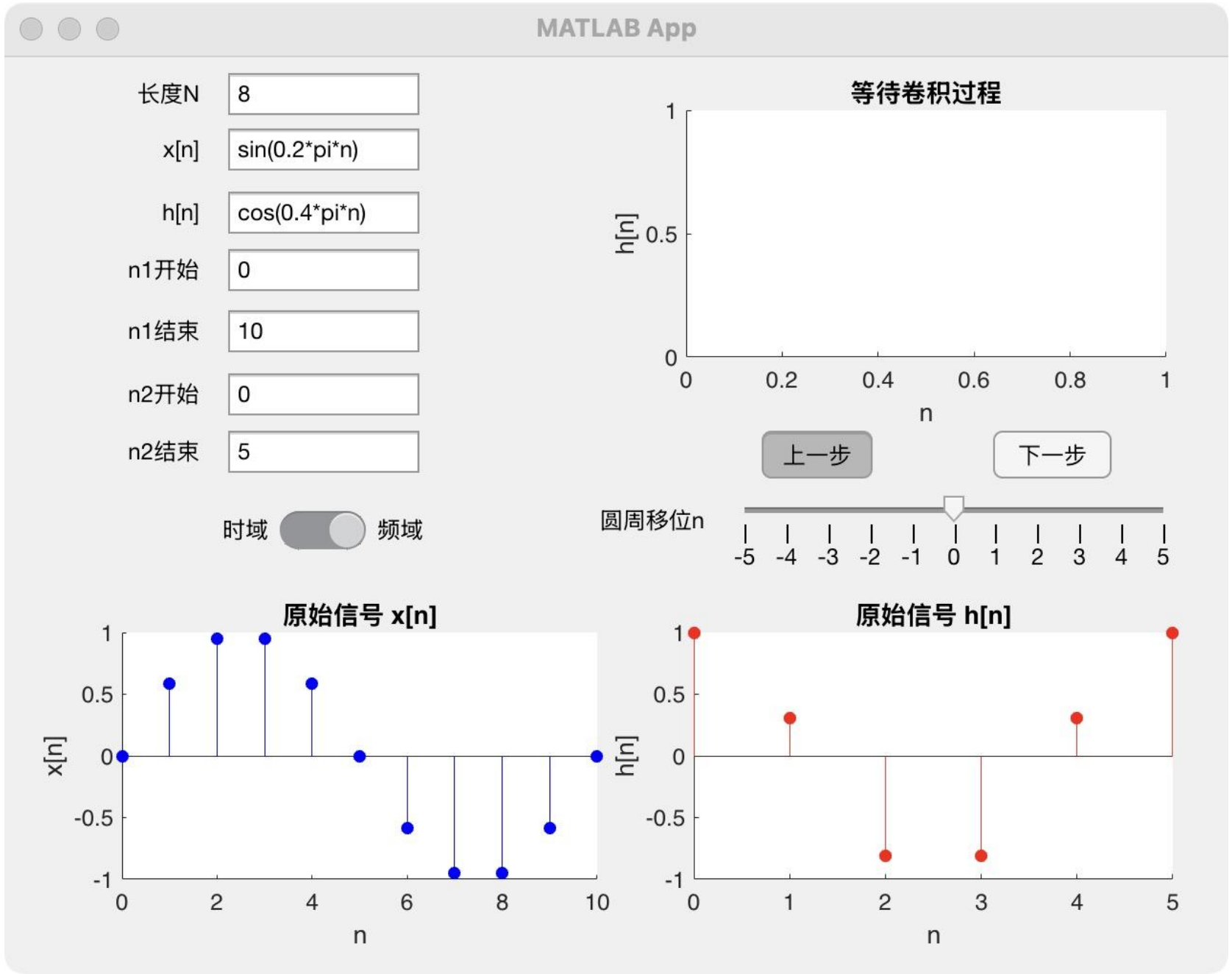

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
圆周移位后的 h[n], 移位 n = 3
h[n]
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
补零/截断后的 x[n]
x[n]
n
补零/截断并翻褶后的 h[n]
h[n]
n

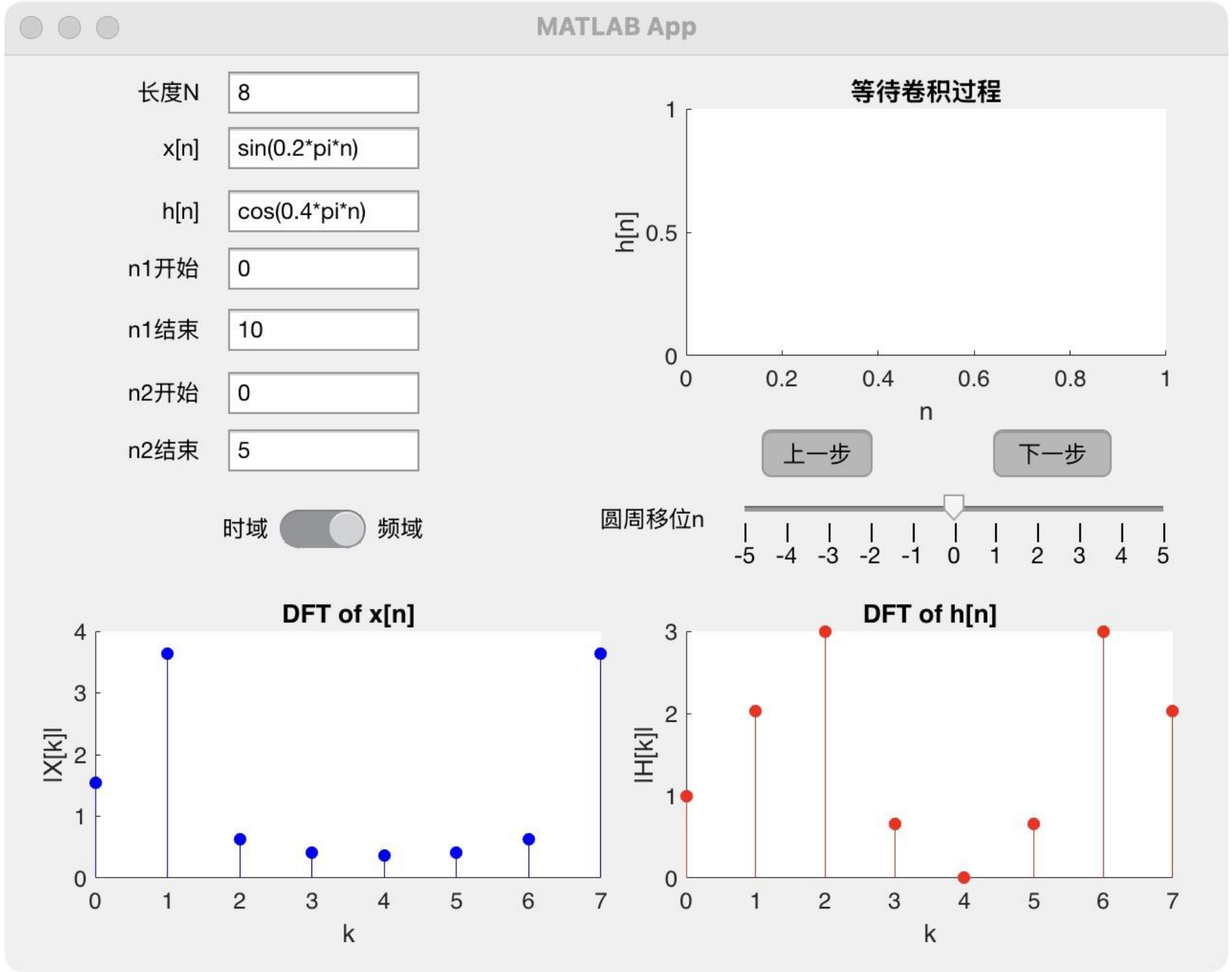

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
x[n] 和 h[n] 的相乘结果
乘积
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
补零/截断后的 x[n]
x[n]
n
补零/截断并翻褶后的 h[n]
h[n]
n

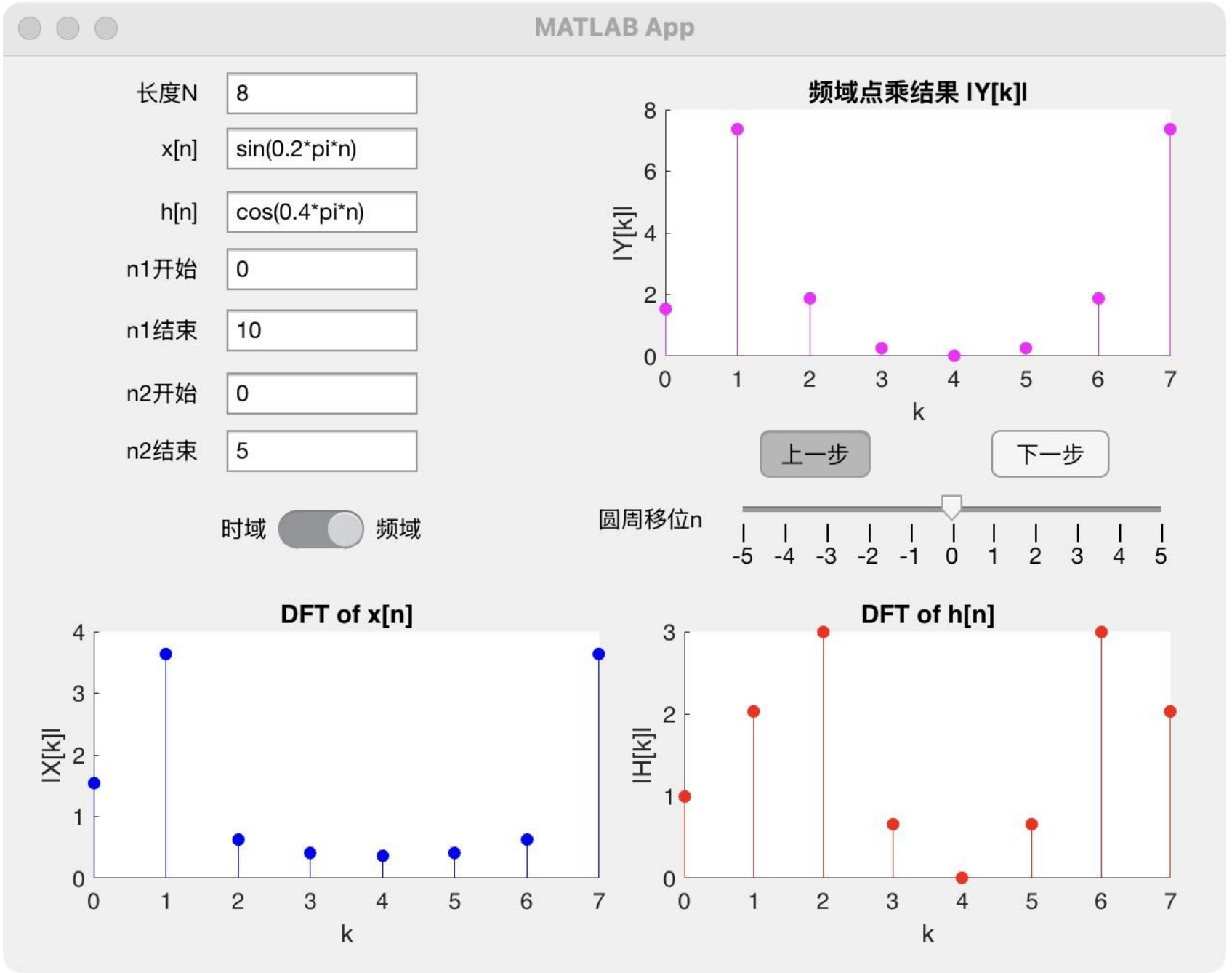

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
x[n] 和 h[n] 的相乘结果
乘积
n
上一步
下一步
圆周移位n
补零/截断后的 x[n]
x[n]
n
补零/截断并翻褶后的 h[n]
h[n]
n

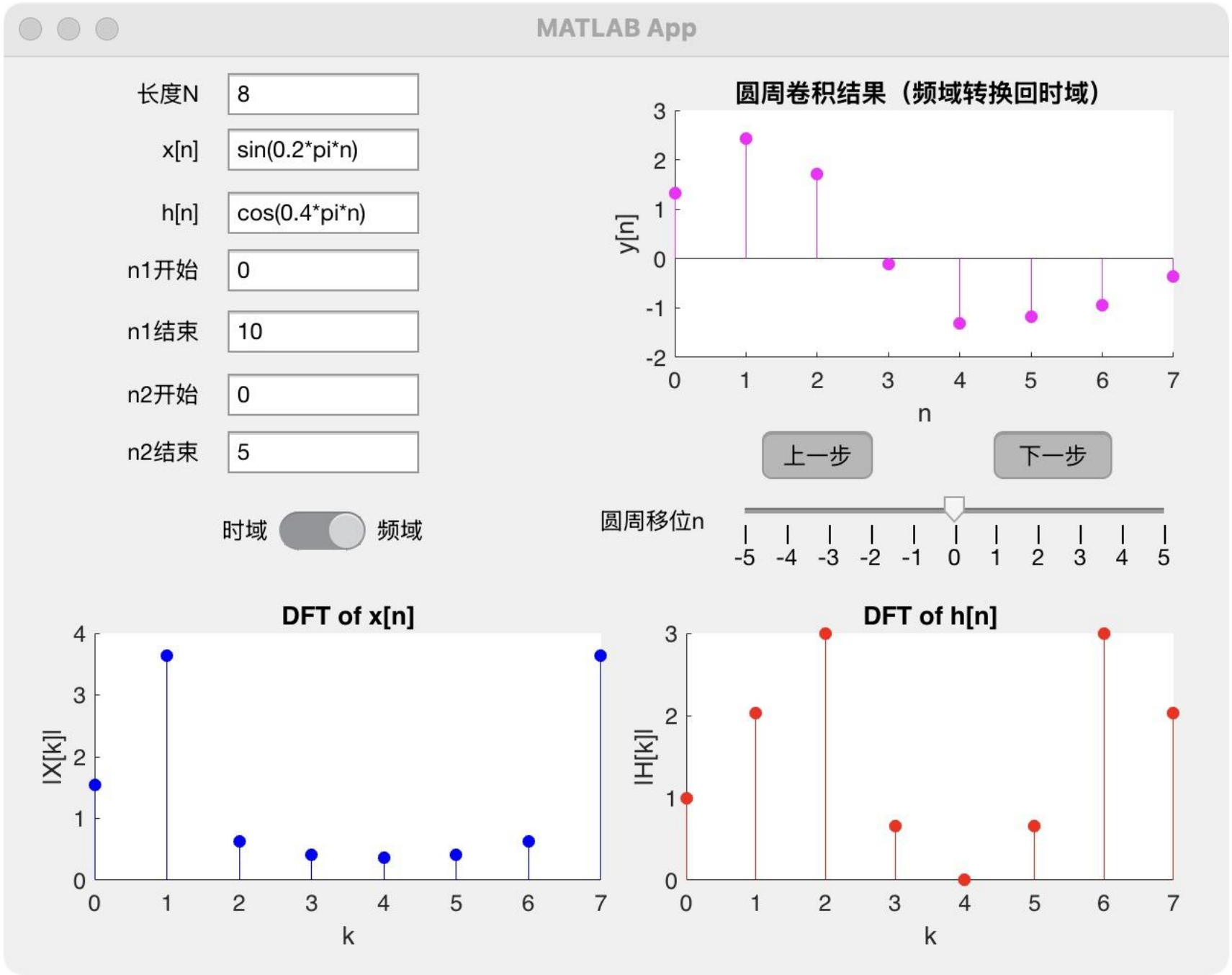

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
圆周卷积结果（时域）
y[n]
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
补零/截断后的 x[n]
x[n]
n
补零/截断并翻褶后的 h[n]
h[n]
n

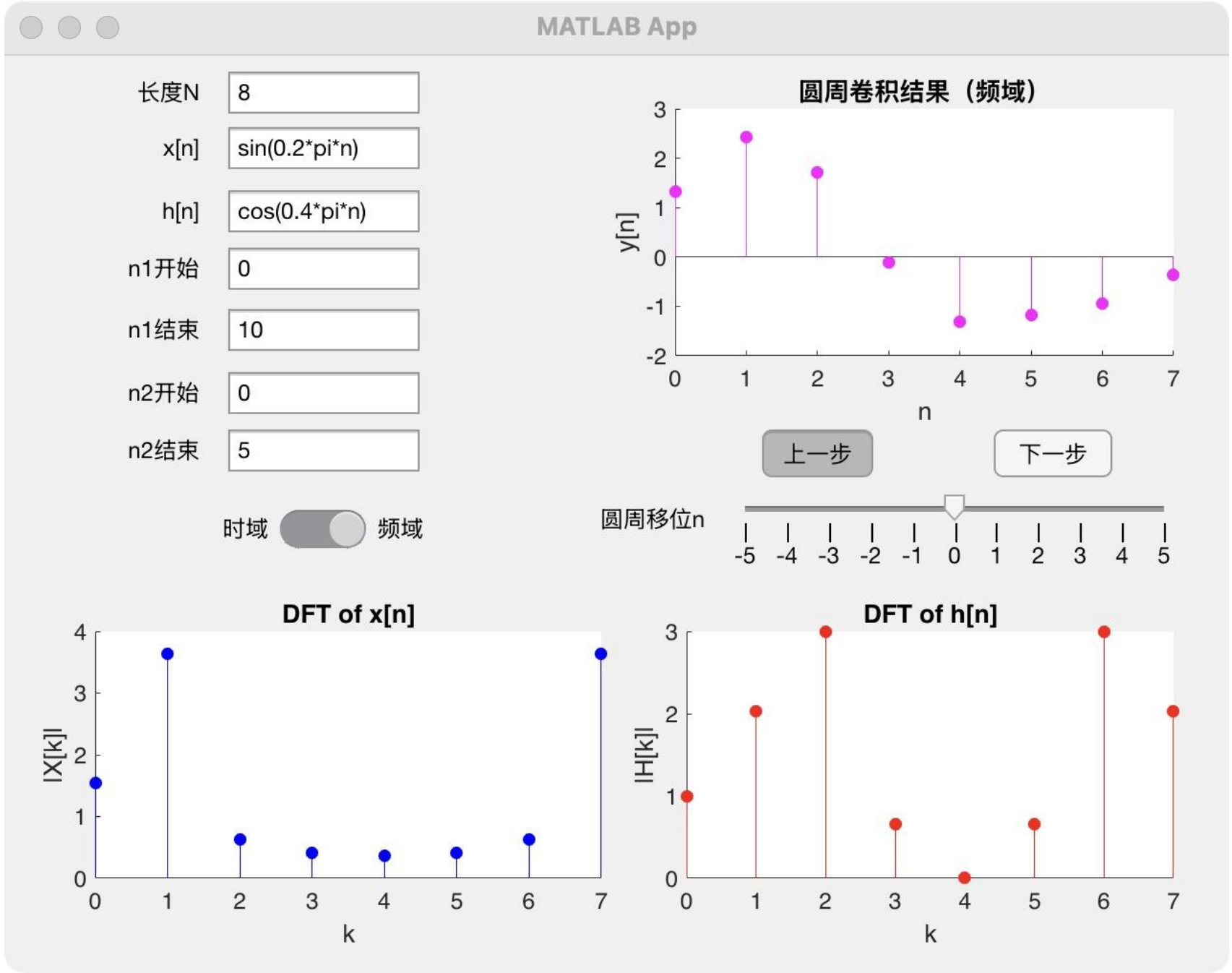

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
h[n]
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
等待开始
x[n]
n
等待开始
h[n]
n

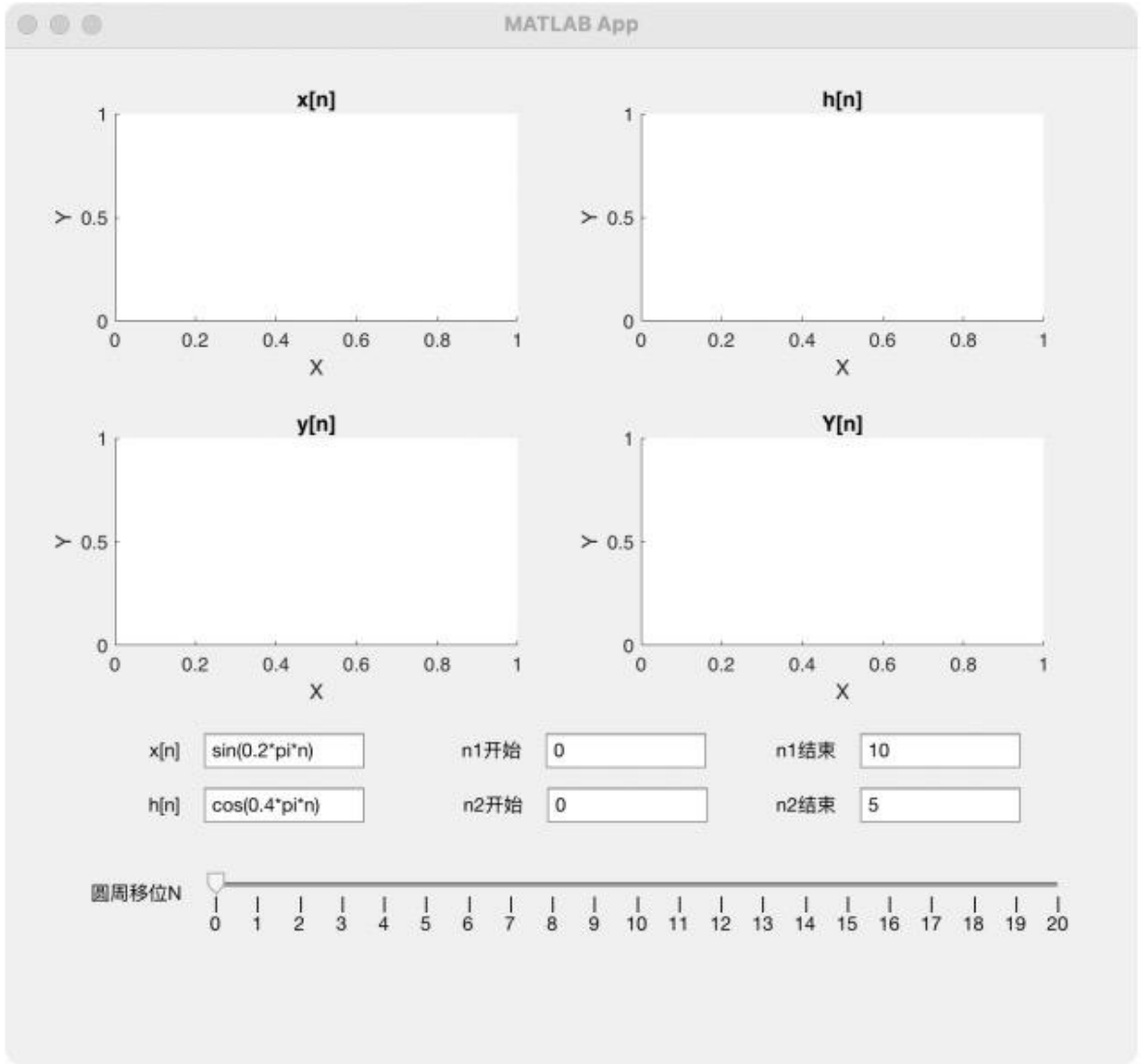

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
h[n]
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
原始信号 x[n]
x[n]
n
原始信号 h[n]
h[n]
n


MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
等待卷积过程
h[n]
n
上一步
下一步
圆周移位n
DFT of x[n]
|X[k]|
k
DFT of h[n]
|H[k]|
k

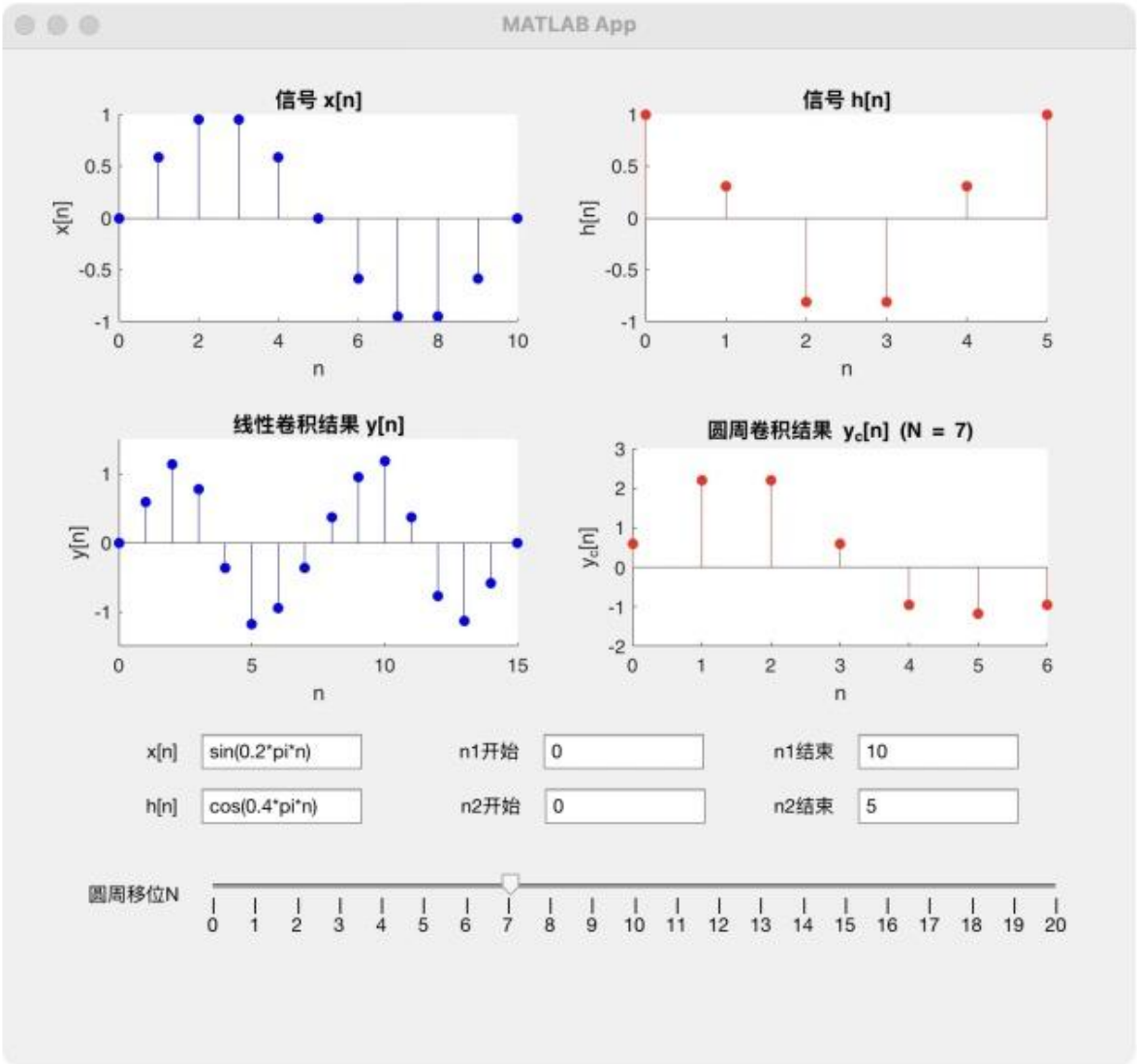

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
频域点乘结果 |Y[k]|
|Y[k]|
k
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
DFT of x[n]
|X[k]|
k
DFT of h[n]
|H[k]|
k


MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
圆周卷积结果（频域转换回时域）
y[n]
n
上一步
下一步
圆周移位n
DFT of x[n]
|X[k]|
k
DFT of h[n]
|H[k]|
k

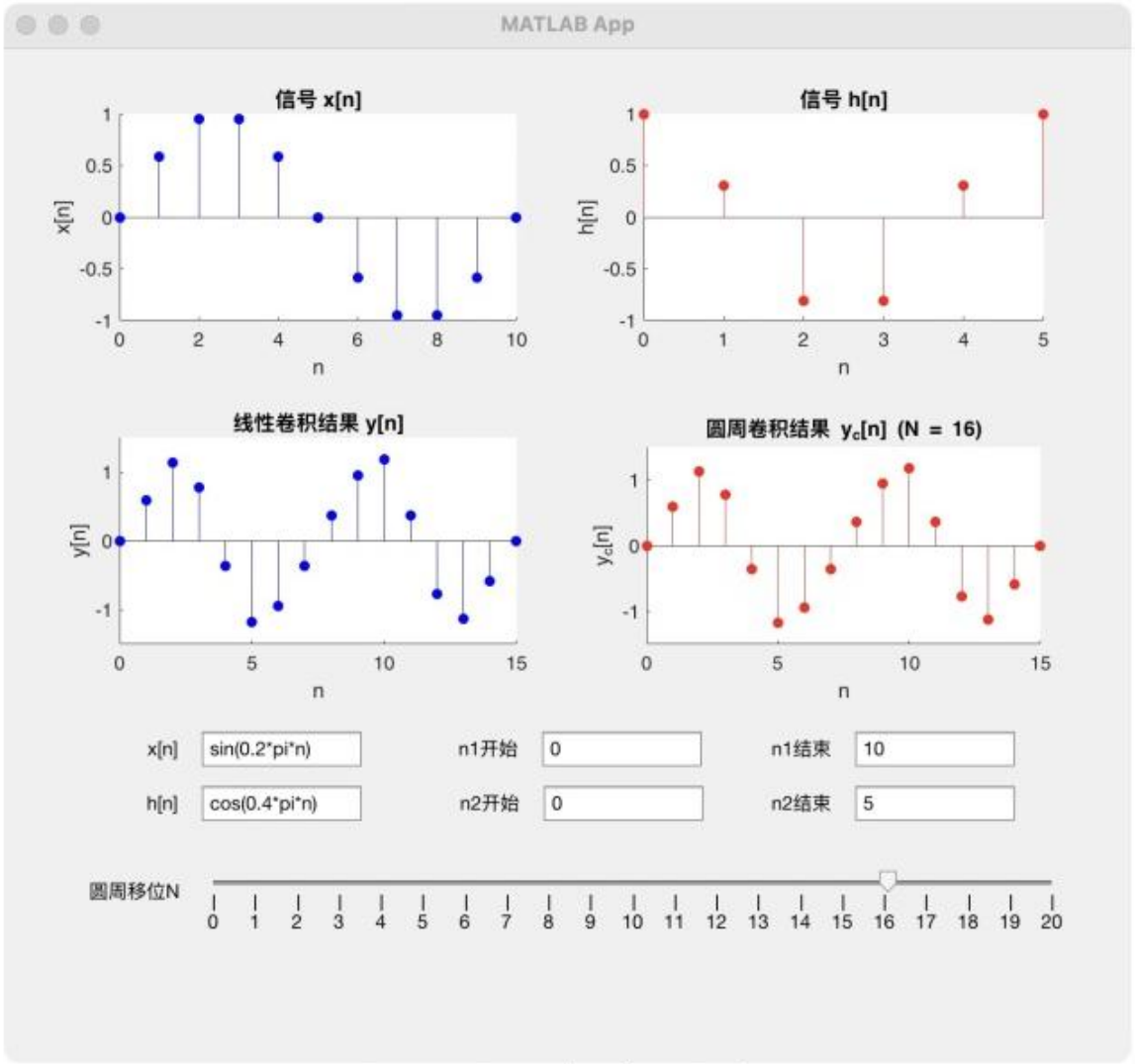

MATLAB App
长度N
8
x[n]
sin(0.2*pi*n)
h[n]
cos(0.4*pi*n)
n1开始
0
n1结束
10
n2开始
0
n2结束
5
时域
频域
圆周卷积结果（频域）
y[n]
n
上一步
下一步
圆周移位n
-5 -4 -3 -2 -1 0 1 2 3 4 5
DFT of x[n]
|X[k]|
k
DFT of h[n]
|H[k]|
k



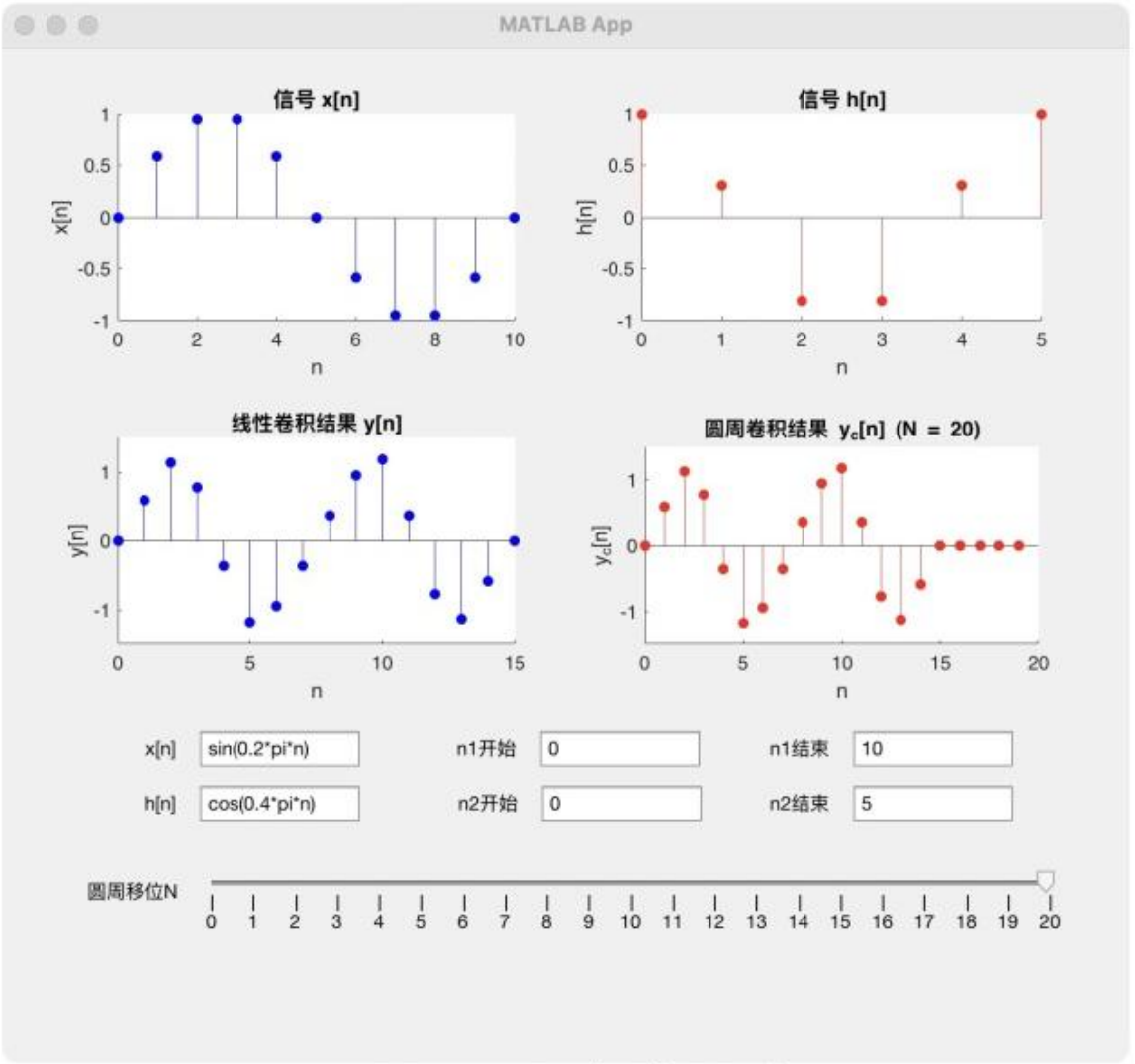
Step 1:输入对应值



Step 2:移动滑块
显示不同长度N下圆周卷积
和线性卷积的对比



Step 3:移动滑块
显示不同长度N下圆周卷积
和线性卷积的对比



Step 4:移动滑块
显示不同长度N下圆周卷积
和线性卷积的对比

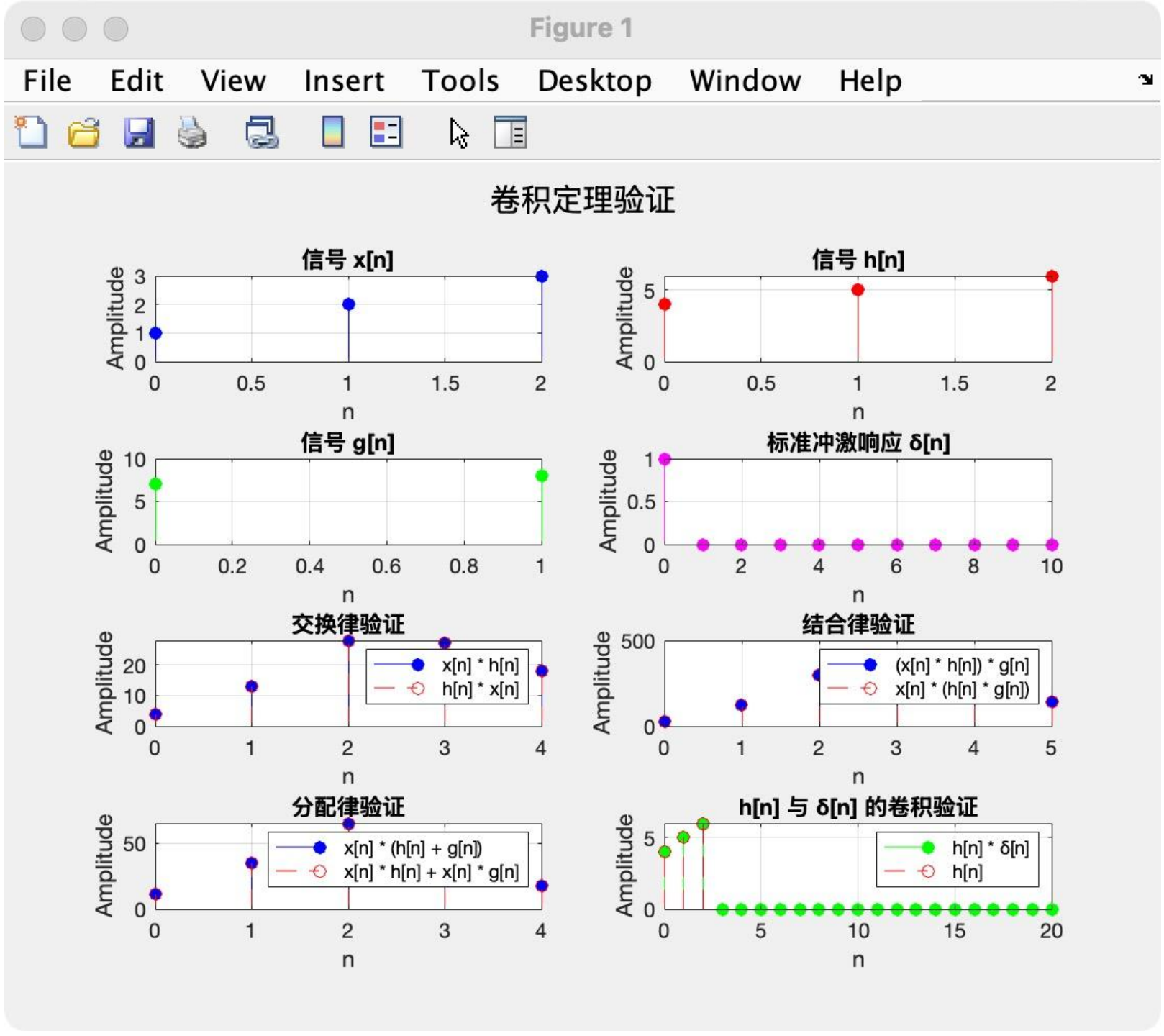


Figure 1
File Edit View Insert Tools Desktop Window Help
卷积定理验证
信号 x[n]
信号 h[n]
信号 g[n]
标准冲激响应 δ[n]
交换律验证
x[n] * h[n]
h[n] * x[n]
结合律验证
(x[n] * h[n]) * g[n]
x[n] * (h[n] * g[n])
分配律验证
x[n] * (h[n] + g[n])
x[n] * h[n] + x[n] * g[n]
h[n] 与 δ[n] 的卷积验证
h[n] * δ[n]
h[n]
Amplitude
n

THANKS