EXPERIMENT REPORT

of

Digital Signal Processing

CONVOLUTION GUI BY USING MATLAB

SUBMITTED BY

Xu Yining *2014141453202*

Professor Zhang Hong

School of Electronic and Information Engineering, Sichuan University

December, 2016

1 Linear Convolution

1.1 Background and Expression

1.1.1 Background

Linear convolution is an operation that describes the relationship between the input and output of a linear system in the time domain. This operation in linear system analysis and signal processing applications, often referred to as convolution. Linear convolution is the most common basic operation in digital signal processing, not only for system analysis but also for system design. If the impulse response of the filter is represented, the convolution is a linear filter and $y(n)$ is the response of the signal $x(n)$ through the filter.

1.1.2 Expression

In discrete time, convolution of two signals involves summing the product of two signals, where one of the signals is "flipped and shifted". For complex-valued function $x(n)$and $h(n)$defined on the *Z* integers, the discrete convolution of $x(n)$and $h(n)$ and g is given by:

$$x(n) * h(n) \stackrel{\text{def}}{=} \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

The convolution of two finite sequences is defined by extending the sequences to finitely supported functions on the set of integers. When the sequences are the coefficients of two polynomials, then the coefficients of the ordinary product of the two polynomials are the convolution of the original two sequences. This is known as the Cauchy product of the coefficients of the sequences. For linear convolution, if the lengths of the two sequences are N_1 and N_2, the total length of the convolution results should be $\mathrm{L} = N_1 + N_2 - 1$.

Thus when $x(n)$and $h(n)$ has finite support. For example, $x(n)$ in the set $[0, n]$, and $h(n)$ in the set $[0, m]$ (representing, for instance, a finite impulse response), a finite summation $y(n)$ may be used:

$$y(n) = x(n) * h(n) \stackrel{\text{def}}{=} \sum_{k=0}^{m+n-1} x(k)h(n-k) = \sum_{k=0}^{m+n-1} h(k)x(n-k)$$

For the length of $x(n)$ is m. And the length of $h(n)$ is n. That contribute to $y(n)$ is a sequence whose length is $m + n - 1$.

1.2 Code implementation Using MATLAB

There are three main methods to compute the convolution :

Method 1: Direct calculation. Method 2: Fast Fourier Transform (FFT). Method 3: Section convolution (sectioned convolution).Method 1 is the direct use of the definition to calculate the convolution, and 2 and 3 are used FFT to quickly calculate the convolution. Here we use method 1 to attain the convolution.

1.2.1 MATLAB Codes (*with Notes*)

```
function y = yn_conv(x,h)
% get the length of the two sequence
m = length(x);
n = length(h);
N = m + n -1;
if m >= n   % let the longest to flip
    t = x;
    x = h;
    h = t;
    t = m;
    m = n;
    n = t;
end
% padding zeros to make two sequence the same dimensions
b = zeros(1,n);
x = [b x b];
h = h(end:-1:1);
b = zeros(1,m + n);
h = [h b];
% shifting and cycling to attain the convolution sum
for k = 1:m+n-1
    h = circshift(h,[0,1]);
    y(k) = h * x';
end
end
```

1.2.2 Simulation Results and Illustrations

I create two sequence. Using both the function contain in MATLAB and the function I coded to make sure the function is effective. Also, I measured the time of each function costed.

```
x = [1 3 2 -1];
h = [1 -1 0 1];
n = length(x);
tic
y1 = conv(x,h);
toc
tic
y2 = yn_conv(x,h);
```

```
toc
```

The result of this text is as follows:

```
>> test
时间已过 0.095352 秒。
时间已过 0.690998 秒。
>> y1
y1 =
    1    2   -1   -2    4    2   -1
>> y2
y2 =
    1    2   -1   -2    4    2   -1
```

As we can see in this result.

2 Cyclic Convolution

2.1 Background and Expression

2.1.1 Background

Circular convolution is a convolution operation other than linear convolution, which is a kind of periodic convolution.

2.1.2 Expression

$x(n)$and $h(n)$are two finite sequences of length N, and DFT is denoted by $X(k)$ and $H(k)$, respectively. Forming the results

$$W(k) = X(k)H(k)$$

And the sequence $\omega(n)$ of length N is determined, and its DFT is $W(k)$.

First, we extend $\tilde{x}(n)$ and $\tilde{h}(n)$ into periodic sequences of period N, which are $\tilde{x}(n)$and $\tilde{h}(n)$ for the product of the periodic sequences corresponding to the Fourier series coefficients.

$$\widetilde{w}(n) = \sum_{m=0}^{N-1} \tilde{x}(n)\tilde{h}(n-m) \leftrightarrow \widetilde{W}(k) = \tilde{X}(k)\tilde{H}(k)$$

Periodic sequence $\widetilde{w}(n)$, the DFS coefficient is equivalent to the finite length sequence $\widetilde{W}(k)$ to cycle N extension.

Can be selected $\widetilde{w}(n)$ for one cycle $w(n)$:

$$w(n) = \widetilde{w}(n)R_N(n) = \sum_{m=0}^{N-1} \tilde{x}(n)\tilde{h}(n-m) = \sum_{m=0}^{N-1} x(m)h(< n-m >_N)$$

This step is called a cyclic convolution, denoted as

$$w(n)=x(n) \textcircled{N} h(n)$$

2.2 Code implementation Using MATLAB

2.2.1 MATLAB Codes (*with Notes*)

```
function y = yn_cconv(x, h)
% cyclic convolution need two sequence have the same length
if length(x)~=length(h)
    printf('err');
    return;
end
N = length(x);% length of the result sequence
y = zeros(1,N);
h = h(end:-1:1)';  % sequence fliping and transposing
for n = 1:N   % convolution sum
    y(n) = x*h;
    x = circshift(x, [0, 1]);  % shifting
end
y = y(end:-1:1);
end
```

2.2.2 Simulation Results and Illustrations

I create two sequence. Using both the function contain in MATLAB and the function I coded to make sure the function is effective. Also, I measured the time of each function costed.

```
x = [1 3 2 -1];
h = [1 -1 0 1];
n = length(x);
tic
y1 = cconv(x,h,n);
toc
tic
y2 = yn_cconv(x,h);
toc
```

The result of this text is as follows:

```
>> test
时间已过 0.003781 秒。
时间已过 0.005897 秒。
>> y1
y1 =
     5     4    -2    -2
>> y2
y2 =
     5     4    -2    -2
```

As we can see in this result.

3 Convolution GUI by Using MATLAB

3.1 Automatic Convolution

I set a button, which can automatic run the convolution when it is been clicked.

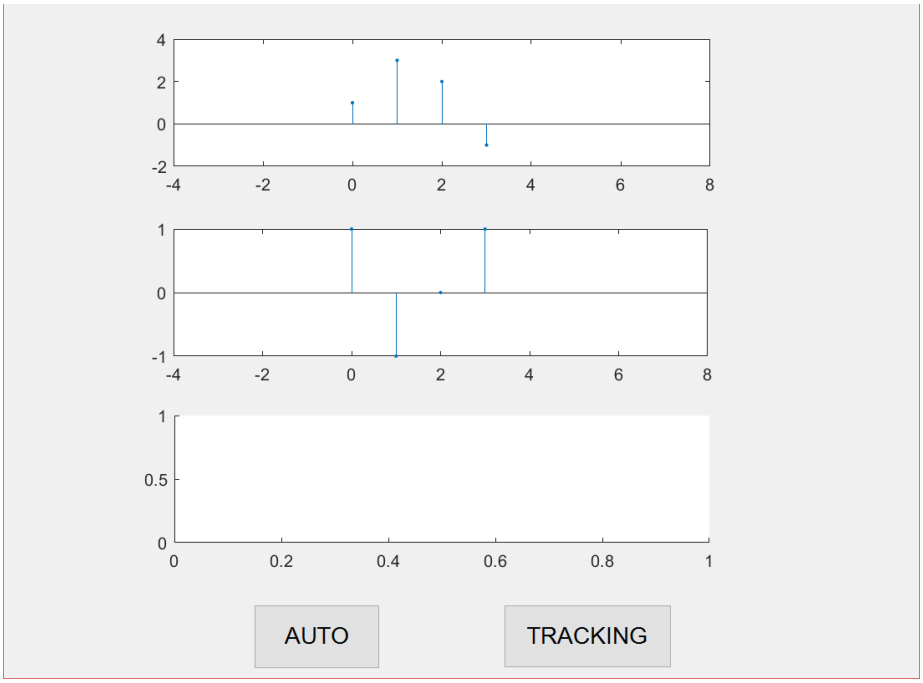
3.1.1 MATLAB Codes (*with Notes*)

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
x.index = [0 1 2 3];
x.value = [1 3 2 -1];
h.index = [0 1 2 3];
h.value = [1 -1 0 1];
m = length(x.value);
n = length(h.value);
N = n + m - 1;
handles.min = -n;
handles.max = m+n;
h1.value = h.value(end:-1:1);
b = zeros(1,n);
x.value = [b x.value b 0];
x.index = -n:n+m ;
b = zeros(1,m + n);
h1.value = [0 h1.value b];
h1.index = -n:n+m;
stem(handles.axes1,x.index,x.value,'.');
set(handles.axes1,'XLim',[-n n+m]);
handles.x = x;
handles.h = h1;
handles.m = m;
handles.n = n;
for k = 1:m+n
    value = circshift(h1.value,[0,k-1]);
    stem(handles.axes2,h1.index,value,'.');
    set(handles.axes2,'XLim',[-n n+m]);
    y.index = -n:m+n-1;
    y.value(find(y.index==k - 1)) = value * x.value';
    hold on
    stem(handles.axes3,y.index(find(y.index==k -
1)),y.value(find(y.index==k - 1)),'.','Color',[0,0,1]);
    set(handles.axes3,'XLim',[-n n+m]);
    ylim(handles.axes3,[-2,4]);
```

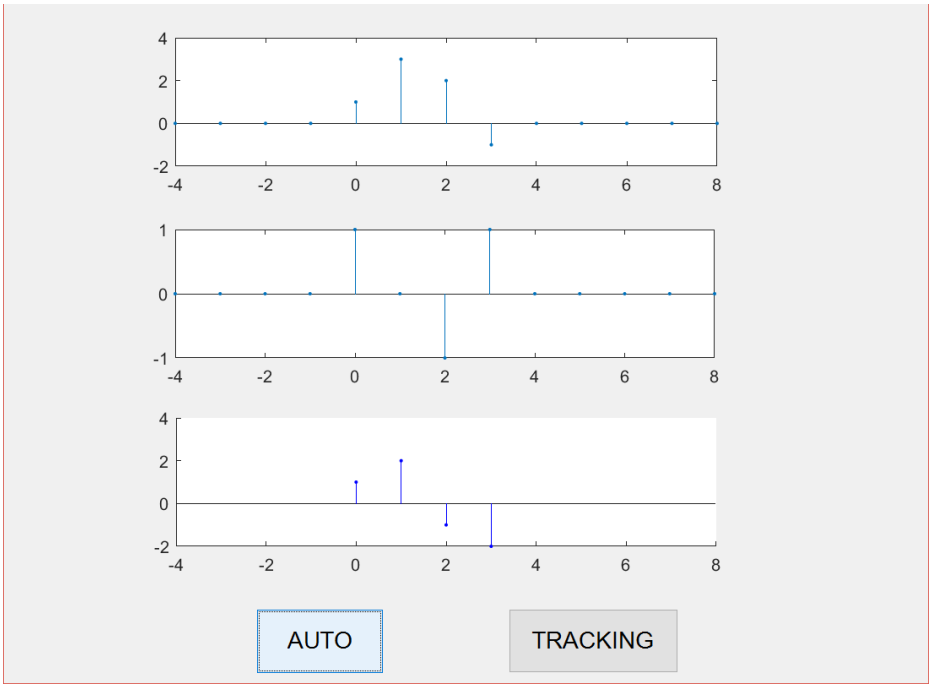
```
    hold off
    pause(0.5);
end
guidata(hObject, handles);
```

I put all the data stored in the handles inside, and updated in time to use when extracted. Loop statement, I will shift $h(n)$ each time, at the same time calculate the $x(n)$ and $h(n)$ transpose of the final result is that I shift to this point $y(n)$ value.

3.1.2 Simulation Results and Illustrations



This is the initial interface. And the follows is the interface that the linear convolution is working.



3.2 Tracking for Convolution

```
function pushbutton2_Callback(hObject, eventdata, handles)
handles.track = ~handles.track;
guidata(hObject, handles);

function figure1_WindowButtonMotionFcn(hObject, eventdata,
handles)
if handles.track == true
    m = handles.m;
    n = handles.n;
    x = handles.x;
    h1 = handles.h;
    currPt = get(handles.axes3, 'CurrentPoint');
    px = currPt(1,1);
    if(px>handles.min&&px<handles.max)
        axes(handles.axes3);
        cla;
        km = (px-handles.min)/(handles.max-
handles.min)*handles.max;
        for k = 1:km
            value = circshift(h1.value,[0,k-1]);
            stem(handles.axes2,h1.index,value,'.');
            set(handles.axes2,'XLim',[-n n+m]);
            y.index = -n:m+n-1;
            y.value(find(y.index==k - 1)) = value *
x.value';
            hold on
            stem(handles.axes3,y.index(find(y.index==k -
1)),y.value(find(y.index==k - 1)),'.','Color',[0,0,1]);
            set(handles.axes3,'XLim',[-n n+m]);
            ylim(handles.axes3,[-2,4]);
            hold off
        end
    end
end
```

Tracking is the same as automatic convolution. The differences is as follows:
Firstly, we need to define an index function for the mouse. The state of the mouse is also stored inside of handles. When we need to track, we clicked the button2 and it will update the state of the mouse. And if the state changed, the function is working.
Secondly, in the display interface, the mouse will have the corresponding coordinates of the place, we only need to determine the coordinates of the mouse, we can display the exactly results of the corresponding linear convolution.

3.3 Note

3.3.1 Update Data

If we want to use or change the data, which is storage in handles. We should add a line of the code like as follows, which use to update the data in handles.

```
guidata(hObject, handles);
```

3.3.2 To Judge

We need to determine the $h(n)$ shift after the flip to which point, to determine what we show the specific value of the convolution to which point, so we use the function *find()* to achieve.

```
y.value(find(y.index==k - 1)) = value * x.value'
stem(handles.axes3,y.index(find(y.index==k -
1)),y.value(find(y.index==k - 1)),'.','Color',[0,0,1]);
```

4 Applications and Conclusion

Convolution and related operations are found in many applications in science, engineering and mathematics. In image processing, digital signal processing.
In digital image processing convolutional filtering plays an important role in many important algorithms in edge detection and related processes. In statistics, a weighted moving average is a convolution. In electrical engineering, the convolution of one function (the input signal) with a second function (the impulse response) gives the output of a linear time-invariant system (LTI). Also Convolutional neural networks apply multiple cascaded convolution kernels with applications in machine vision and artificial intelligence.

Through this experiment, not only can it deepen our understanding of the principle of convolution, but also cultivate our ability of programming, independent thinking and problem-solving. At the same time, we also have a better understanding of the GUI. This practice for our future practice laid a more solid foundation. Thanks to the teachers for their support and help.