

【畳込み演算 Convolution】

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

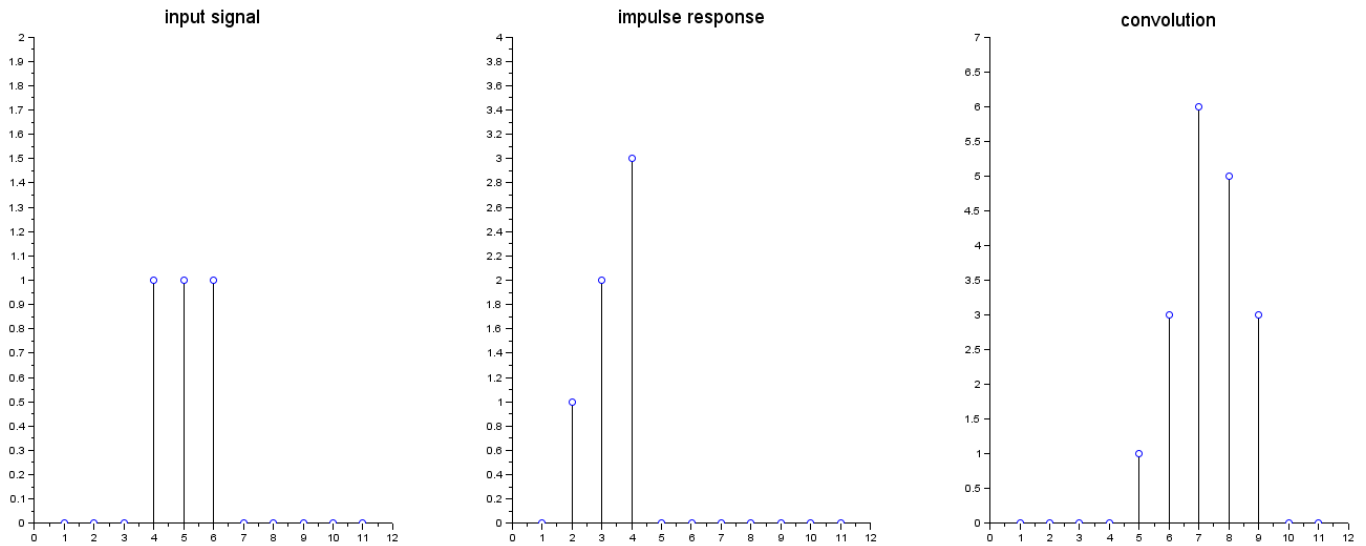


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//   畳込み演算
//   Convolution
//
//                               M.Tsutsui
////////////////////////////////////

clear;

funcprot(0);
function[y]=conv_1(x,h);//x:input signal , h:impulse response
//-----Convolution_Program_1-----//
    [size1,size2]=size(x);

    y=[]; //畳込み結果
    for k=1:1:size2;
        y=[y,sum(x(1:k).*flipdim(h(1:k),2))]; //[x1h1, x1h2+x2h1, x1h3+x2h2+x3h1, x1h4+x2h3+x3h2+x4h1, ...]
    end

endfunction

funcprot(0);
function[y]=conv_2(x,h);//x:input signal , h:impulse response
//-----Convolution_Program_2-----//
//結果はProgram_1と同一

    [size1,size2]=size(x);

    x_buf=[];

    for k=1:1:size2;
        x_buf=[x_buf,x(1:k)]; //[x1, x1,x2, x1,x2,x3, x1,x2,x3,x4, ...]
    end

```

```

end

h_buf=[];
for i=1:1:size2;
    h_buf=[h_buf,flipdim(h(1:i),2)]; //[h1, h2,h1 , h3,h2,h1, h4,h3,h2,h1 ...]
end

y_p=x_buf.*h_buf; //要素ごと乗算

y=[]; //畳み込み結果
for n=1:1:size2;
    y=[y,sum(y_p(1+1/2*n*(n-1):n+1/2*n*(n-1)))]; //サンプル抽出 加算
end

endfunction

x=[0 0 0 1 1 1 0 0 0 0]; //入力信号 例
h=[0 1 2 3 0 0 0 0 0 0]; //インパルス応答 例

subplot(1,3,1)
plot2d3(x);
plot(x,'o');
fg=gca();
fg.data_bounds(:,2)=[0;max(x)+1]; //縦軸調整
title('input signal','fontsize',4);

subplot(1,3,2)
plot2d3(h);
plot(h,'o');
fg=gca();
fg.data_bounds(:,2)=[0;max(h)+1];
title('impulse response','fontsize',4);

subplot(1,3,3)
plot2d3(conv_2(x,h));
plot(conv_1(x,h),'o');
fg=gca();
fg.data_bounds(:,2)=[0;max(conv_1(x,h))+1];
title('convolution','fontsize',4);

```

Source Code 2: Python

```

#-----
# Module Name:Convolution
# Author:m.tsutsui
#-----

#Library Import-----
from numpy import*
import math, numpy as np
import matplotlib.pyplot as plt
#-----

def conv(size):
    x_buf=[]

```

```

for i in range(1,size+1,1):
    x_buf.append(np.r_[x[0:i]])

h_buf=[]
for j in range(1,size+1,1):
    h_buf_p=h[0:j]
    h_buf.append(h_buf_p[::-1])

y=[]
for k in range(0,size,1):
    y.append(sum(array(x_buf[k])*array(h_buf[k])))

return y

if __name__ == '__main__':
    x=array([0 ,0, 0, 1, 1, 1, 0, 0, 0, 0]) #ex)input signal
    h=array([0, 1, 2, 3, 0, 0, 0, 0, 0, 0]) #ex)impulse response

    s_size=np.size(x,0)

    y=conv(s_size)

#_plot-----
    x_def=np.linspace(0,s_size,s_size)
    plt.figure(facecolor='w')

    plt.subplot(131)
    plt.plot(x_def,x,'o')
    plt.bar(x_def,x,width=0.03)
    plt.xlim([0,s_size+2])
    plt.ylim([0,max(x)+1])
    plt.grid()
    plt.title('input_signal')

    plt.subplot(132)
    plt.plot(x_def,h,'o')
    plt.bar(x_def,h,width=0.03)
    plt.xlim([0,s_size+2])
    plt.ylim([0,max(h)+1])
    plt.grid()
    plt.title('impulse_response')

    plt.subplot(133)
    plt.plot(x_def,y,'o')
    plt.bar(x_def,y,width=0.03)
    plt.xlim([0,s_size+2])
    plt.ylim([0,max(y)+1])
    plt.grid()
    plt.title('convolution')

    plt.show()

```