Experiment-4

EE:2801 DSP-Lab

Indian Institute of Technology, Hyderabad

Jay Vikrant
EE22BTECH11025

I. Aim of the experiment

Simulate fixed point convolution and correlation between two signals in Matlab and C. Take input as x=0.3426 3.5784 2.7694 -1.3499 3.0349 0.7254 -0.0631 h=0.7147 -0.2050 -0.1241 1.4897 1.4090 Find out the mean square error for fixed point outputs and plot them.

II. Impelementation

1) we begin by converting floating-point arrays to fixed-point format; multiply each array element by 2^{12} and consider only the integer by rouund/fix in C/matlab respectively.
2) Upon creating the fixed-point arrays, perform convolution or correlation(i.e (fix-point)adding the (fix-point)product for shifted indices of x and h(operating signals)) to generate the output array, which remains in the fixed-point state.
3) The product and add has to be done separetely because of them being in different dimensions 2^{2*12} and 2^{12}
4) Convert the output array back to floating-point format
5) Calculate the error by taking the difference between each element of the fixed-point procedure output array and the array generated using the normal procedure.
6) Determine the average error by summing up all elements in the error array and dividing the total by its length.

NOTE:- The implementation of convolution and correlation(i.e convolution but folding one of the signal) are already done in Experiment 1. This is just the continuation of that experiment considering fixpoint format.

III. Code , Output and Plot of the simulations

Matlab simulation,

```
function main()

x = [0.3426 3.5784 2.7694 −1.3499 3.0349 0.7254 −0.0631];
h = [0.7147 −0.2050 −0.1241 1.4897 1.4090];

disp('convolution x and h by fix−point arithmetic is given by');
disp(fix _ convl(x,h));

disp('corss−correlation of x and h by fix−point arithmetic is given by');
disp(fix _ xcorrl(x,h));
```

```
disp('error of convolution from my function');
err_convl = mserr(fix_convl(x,h),float_convl(x,h),length(x)+length(h)-1);
disp(err_convl)

disp('error of correlation from my function');
err_xcorrl = mserr(fix_xcorrl(x,h),float_xcorrl(x,h),length(x)+length(h)-1);
disp(err_xcorrl);

%Plot
x = 1:1:length(x)+length(h)-1;
figure;
plot(x,err_convl,"g")
grid on;
figure;
plot(x,err_xcorrl,"b")
grid on;
end

function y = float_convl(x,h) % y here is the output

  l = length(x) + length(h) - 1;

  % Initialize output sequence
  y = zeros(1, l);

  % Perform convolution manually
  for n = 1:l
    for k = 1:length(x)
      if (n - k + 1) >= 1 && (n - k + 1) <= length(h)
        y(n) = y(n) + x(k) * h(n - k + 1); % This is just /sigma x(k)*h(n-k)
      end
    end
  end
end

function y = fix_convl(x,h) % y here is the output

  % Calculate output length
  l = length(x) + length(h) - 1;

  % Initialize output sequence
  y = zeros(1, l);
  Q = 12;
  % Perform convolution manually
  for n = 1:l
    for k = 1:length(x)
      if (n - k + 1) >= 1 && (n - k + 1) <= length(h)

        yf = fixpoint(y(n),Q);
```

```
            xf = fixpoint(x(k),Q); %x(k) taken in fixpoint

            hf = fixpoint(h(n - k + 1),Q); %x(n-k) taken in fixpoint

            xhf = xf*hf; %x(k)*h(n-k) product taken in fixpoint

            xh = xhf/(2^(2*Q)); %x(k)*h(n-k) product converted back

            xhff = fixpoint(xh,Q); %x(k)*h(n-k)=xh taken in fixpoint

            yf = yf + (xhff); % This is just /sigma x(k)*h(n-k) done in fixpoint

            y(n) = yf/(2^(Q)); % convolution in fixpoint converted back
        end
      end
    end
end

function x0 = flip_lr(x)
      l = length(x);
      x0 = zeros(1,l);
      for i = 1:l
            x0(l-i+1) = x(i);
      end
end

function y1 = fix_xcorrl(x,h) % y1 here is the output

    h0=flip_lr(h); %to rearange h(n) as h(-n)
    y1 = fix_convl(x,h0);

end

% cross_correlation

function y1 = float_xcorrl(x,h) % y1 here is the output

    h0=fliplr(h); %to rearange h(n) as h(-n)
    y1 = float_convl(x,h0);

end

function result = fixpoint(a, Q)
      result = fix(a * 2^Q);
end

function err = mserr(y_obs,y_acc,n)

err = zeros(1,n);
```

```
    for i = 1:n

        err(i) = (y_obs(i) - y_acc(i))^2 ;

    end
    disp('mse is')
    disp(sum((err))/(n));


end
```

The following got computed in Matlab,

```
>> main
convolution x and h by fix-point arithmetic is given by
    0.2446    2.4868    1.2034   -1.4653    7.9138    9.2302    1.3215    2.5410    5.3635    0.9280   -0.0886

corss-correlation of x and h by fix-point arithmetic is given by
    0.4824    5.5515    9.1890    1.7092    1.4336    7.6987    2.8706   -1.7698    2.0276    0.5310   -0.0449

error of convolution from my function
mse is
   7.5410e-07

   1.0e-05 *

    0.0052    0.0188    0.0028    0.0769    0.3019    0.1272    0.0689    0.0959    0.1227    0.0012    0.0081

error of correlation from my function
mse is
   7.5309e-07

   1.0e-05 *

    0.0091    0.0678    0.1811    0.0001    0.0584    0.2982    0.0254    0.1380    0.0335    0.0139    0.0031
```

The following got error plot was plotted in matlab,

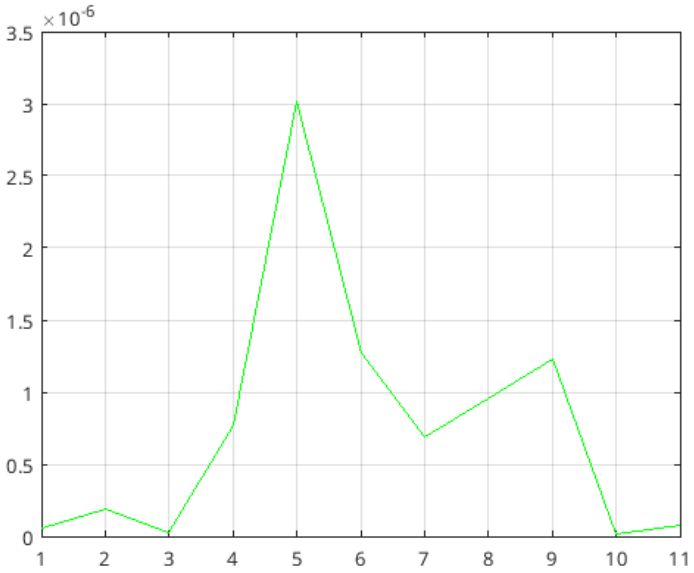

Fig. 6. Squared Error Plot of fix point convolution(square error VS index of each element of o/p)

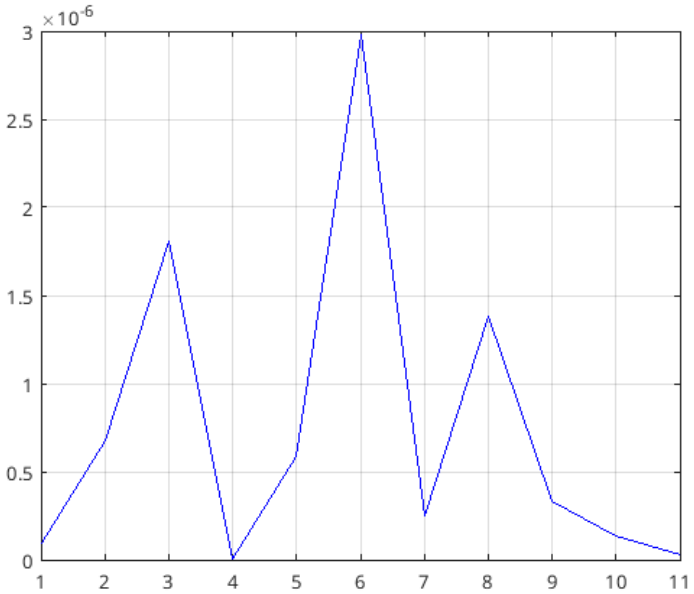

Fig. 6. Squared Error Plot of fix point correlation(square error VS index of each element of o/p)

C simulation,

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int fixpoint(float a, int Q)
{
    int y = round(a * pow(2, Q));
    return y;
}

float float_calc_conv(float *x, float *h, int l1, int l2,int index) // function to compute convolution
{
    int l = l1 + l2 - 1;
    float y[l];
    for (int n = 0; n < l; n++)
    {
        y[n] = 0;
        for (int k = 0; k < l1; k++)
        {
            if (n - k >= 0 && n - k < l2)
            {
                y[n] = y[n] + x[k] * h[n - k];
            }
        }
    }

    return y[index];
}

float fix_calc_conv(float *x, float *h, int l1, int l2,int index) // function to compute convolution
{
    int l = l1 + l2 - 1;

    float y[l],xh;

    int yf,xf,hf,xhf,xhff;

    for (int n = 0; n < l; n++)
    {
        y[n] = 0;
        int Q = 12;
        for (int k = 0; k < l1; k++)
        {
            if (n - k >= 0 && n - k < l2) /*keeping the h(n-k) sequnce same even though it has
                been folded and shifted */
            {
                yf = fixpoint(y[n],Q);
                xf = fixpoint(x[k],Q);
```

```
                hf = fixpoint(h[n-k],Q);

                xhf = xf*hf;
                xh = xhf/(pow(2,2*Q));

                xhff = fixpoint(xh,Q);

                yf = yf + xhff;
                y[n] = yf/(pow(2,Q));
            }
        }
    }

    return y[index];
}

float fliplr(float *x, int l1, int index) // function for folding the sequence i.e finding x(-n)
{
    float x0[l1];

    for (int i = 0; i < l1; i++)
    {
        x0[(l1 - 1) - i] = x[i];
    }
    return x0[index];
}

float fix_calc_corr(float *x, float *h, int l1, int l2,int index) // function to compute corss correlation
{
    float h0[l2];
    int l = l1 + l2 - 1;
    float y[l];
    for (int i = 0; i < l2; i++)
    {
        h0[i] = fliplr(h, l2, i);
    }
    for (int i = 0; i < l1 + l2 - 1; i++)
    {
        y[i]=fix_calc_conv(x, h0, l1, l2,i);
    }
    return y[index];

}

float float_calc_corr(float *x, float *h, int l1, int l2,int index) // function to compute corss correlation
{
    float h0[l2];
    int l = l1 + l2 - 1;
    float y[l];
    for (int i = 0; i < l2; i++)
```

```
    {
        h0[i] = fliplr(h, l2, i);
    }
    for (int i = 0; i < l1 + l2 - 1; i++)
    {
        y[i]=float_calc_conv(x, h0, l1, l2,i);
    }
    return y[index];

}

float error(float *x_obs,float *x_acc,int n,int index)
{
    float err[n];

    for(int i=0;i<n;i++)
    {
        err[i]=(pow((x_obs[i]-x_acc[i]),2));

    }
    return err[index];
}

float sum(float *x, int n) {

    float total = 0;

    // Loop through the array using a pointer, efficiently accessing elements
    for (int i = 0; i < n; i++) {
        total += *(x + i); // Dereference the pointer with offset to get each element
    }

    float avg = total/n;

    return avg;
}

int main()
{
    int l1 = 7;
    int l2 = 5;
    int l =l1 +l2 -1;
    float x[7] = {0.3426, 3.5784, 2.7694, -1.3499, 3.0349, 0.7254, -0.0631};
    float h[5] = {0.7147, -0.2050, -0.1241, 1.4897, 1.4090};

    printf("Convolution by fixpoint arithemetic of x and h resulted:\n");

    for (int i = 0; i < l1 + l2 - 1; i++)
    {
        printf("%.4f   ", fix_calc_conv(x, h, l1, l2,i));
```

```
}
printf("\n");

printf("Convolution␣by␣floatpoint␣arithemetic␣of␣x␣and␣h␣resulted:\n");

for (int i = 0; i < l1 + l2 − 1; i++)
{
    printf("%.4f␣␣␣", float_calc_conv(x, h, l1, l2,i));
}
printf("\n");

printf("Cross−correlation␣by␣fixpoint␣arithemetic␣of␣x␣and␣h␣resulted:\n");

for (int i = 0; i < l1 + l2 − 1; i++)
{
    printf("%.4f␣␣␣", fix_calc_corr(x, h, l1, l2,i));
}
printf("\n");

printf("Cross−correlation␣floatpoint␣arithemetic␣of␣x␣and␣h␣resulted:\n");

for (int i = 0; i < l1 + l2 − 1; i++)
{
    printf("%.4f␣␣␣", float_calc_corr(x, h, l1, l2,i));
}
printf("\n");

printf("the␣mean␣square␣error␣for␣convolution␣\n");

// for computing mse in convolution
float y_acc[l],y_obs[l],err[l];

for(int i = 0; i < l1 + l2 − 1; i++)
{
    y_obs[i]=float_calc_conv(x, h, l1, l2,i);
    y_acc[i]=fix_calc_conv(x, h, l1, l2,i);
}

for (int i = 0; i < l1 + l2 − 1; i++)
{
    err[i]=error(y_obs,y_acc,l,i);
    printf("%E␣␣␣", error(y_obs,y_acc,l,i));
}
printf("\n");

printf("The␣mse␣sum␣is␣%E",sum(err,l));

printf("\n");

// for computing mse in correlation
```

```
    printf("the mean square error for correlation\n");

    for(int i = 0; i < l1 + l2 - 1; i++)
    {
        y_obs[i]=float_calc_corr(x, h, l1, l2,i);
        y_acc[i]=fix_calc_corr(x, h, l1, l2,i);
    }

    for (int i = 0; i < l1 + l2 - 1; i++)
    {
        err[i]=error(y_obs,y_acc,l,i);
        printf("%E   ", error(y_obs,y_acc,l,i));
    }
    printf("\n");

    printf("The mse sum is %E",sum(err,l));

    printf("\n");

    return 0;
}
```

The following got computed in C,

```
[Running] cd "/home/jay/Desktop/Dsp-lab/C/" && gcc main4.c -o main4 -lm && "/home/jay/Desktop/Dsp-lab/C/"main4
Convolution by fixpoint arithemetic of x and h resulted:
0.2449   2.4868   1.2026   -1.4661   7.9155   9.2307   1.3208   2.5425   5.3643   0.9282   -0.0889
Convolution by floatpoint arithemetic of x and h resulted:
0.2449   2.4872   1.2032   -1.4662   7.9156   9.2314   1.3207   2.5420   5.3646   0.9281   -0.0889
Cross-correlation by fixpoint arithemetic of x and h resulted:
0.4827   5.5520   9.1902   1.7095   1.4324   7.7000   2.8711   -1.7705   2.0278   0.5312   -0.0449
Cross-correlation floatpoint arithemetic of x and h resulted:
0.4827   5.5523   9.1903   1.7093   1.4328   7.7005   2.8711   -1.7710   2.0282   0.5314   -0.0451
the mean square error for convolution
2.835288E-10   1.872550E-07   3.188798E-07   2.080611E-08   7.648850E-10   4.082722E-07   9.578699E-09   2.358643E-07   1.408282E-07   1.
798561E-08   1.657908E-09
The mse sum is 1.342176E-06
the mean square error for correlation
3.291256E-09   1.120507E-07   1.560784E-08   4.805999E-08   2.086754E-07   2.559593E-07   2.256115E-10   1.957051E-07   1.118911E-07   1.
660625E-08   3.086904E-08
The mse sum is 9.989416E-07
```

IV. Observations and Understanding

1) While implementing the fixed-point method, very minor discrepancy is noted. However, the average error amounts is of order 10^{-7} , aligning with our C code for convolution and correlation. This error is considerably small compared to the values in the input and output arrays. Consequently, employing the fixed-point method proves advantageous as it primarily involves integer numbers, contributing to memory efficiency.
2) In the MATLAB plots, it is evident that the error peaks at specific points, likely attributable to a higher number of operations (both addition and multiplication) required to derive those particular values.