DSP LAB - Experiment 4

Fixed and Floating Point Convolution and Correlation

Ajay Krishnan K
EE22BTECH11003

February 14, 2024

Contents

1 Aim

To implement fixed and floating point convolution and correlation in MATLAB and C and compare the results by calculating the mean square of the difference between the fixed and floating point results and plotting the square of the difference.

2 Theory

2.1 Convolution

Convolution is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. In discrete time, the convolution of two sequences is given by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

2.1.1 Implementation in MATLAB

First, both the sequences are converted to fixed point using the custom floatToFixed function and then the convolution is performed using the convtaker function. Then qFactor multiplied fixed point array is converted back to fixed point by dividing by $(qFactor)^2$. Then the absolute square difference and mean square error is calculated. Subsequently, the absolute square of the difference between the fixed and floating point results is plotted.

2.1.2 Matlab Code

```
% CONVOLUTION

clc
clear
close all

input_x = [0.3426    3.5784    2.7694   -1.3499    3.0349
   0.7254   -0.0631];
fixed_input_x = floatToFixed(input_x);
h_array = [0.7147   -0.2050   -0.1241    1.4897    1.4090];
fixed_h_array = floatToFixed(h_array);

q_format = 2^12;
fixed_conv = conv_taker(fixed_input_x, fixed_h_array)/(q_format
   ^2);
float_conv = conv_taker(input_x, h_array);
disp("fixed convolution:");
```

```
disp(fixed_conv);
disp("float convolution:");
disp(float_conv);
float_fix_diff = absoluteDifferenceSquare(fixed_conv, float_conv)
    ;
disp("absolute error Square:");
disp(float_fix_diff);
float_fix_mse = meanSquareError(fixed_conv, float_conv);
disp("mean square : ");
disp( float_fix_mse);


figure;
stem(float_fix_diff);
title('Absolute Difference Square (Convolution)');
xlabel('Sample');
ylabel('Absolute Difference Square');

function mse = meanSquareError(array1, array2)
mse = 0;
for i = 1:length(array1)
    mse = mse + (array1(i) - array2(i))^2;
end
mse = mse / length(array1);
end

function abs_diff = absoluteDifferenceSquare(array1, array2)
abs_diff = abs((array1 - array2).^2);
end

function [conv] = conv_taker(signal, impulse)
L = length(signal) + length(impulse) - 1;
conv = zeros(1, L);
for n = 1:L
    for k = 1:length(impulse)
        m = n - k + 1;
        if 1 <= m && m <= length(signal)
            conv(n) = conv(n) + signal(m) * impulse(k);
        end
    end
end
end

function fixed_array = floatToFixed(input_array)
q_format = 2^12;
fixed_array = fix(input_array * q_format);
end
```

2.1.3 Output

```
fixed convolution:
0.2448     2.4869     1.2035    -1.4655     7.9142     9.2307
   1.3212     2.5412     5.3639     0.9281    -0.0887

float convolution:
    0.2449     2.4872     1.2032    -1.4662     7.9156     9.2314
          1.3207     2.5420     5.3646     0.9281    -0.0889

absolute error Square:
   1.0e-05 *

    0.0007     0.0096     0.0077     0.0572     0.1834     0.0439
          0.0229     0.0585     0.0595     0.0003     0.0026

mean square :
   4.0578e-07
```

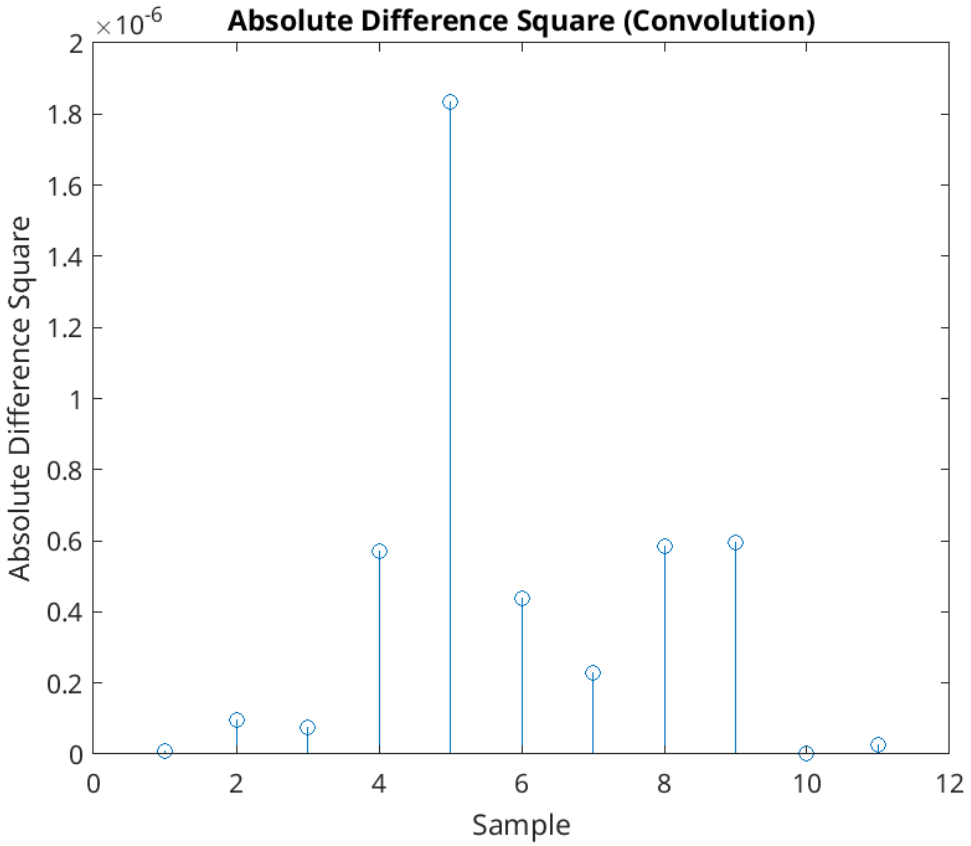

Figure 1: Square of difference between fixed and floating point convolution

2.1.4 Implementation in C

The fixed point convolution is implemented in C using the same logic as in MATLAB. The absolute square difference and mean square error is calculated and printed.

2.1.5 C Code

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

#define ARRAY_LENGTH 7
#define IMPULSE_LENGTH 5
#define Q_FORMAT pow(2, 12)

void conv_taker(float *signal, float *impulse, float *conv);
float meanSquareError(float *array1, float *array2);
void absoluteDifferenceSquare(float *array1, float *array2, float
    *abs_diff);
void floatToFixed(float *input_array, float *fixed_array);

int main() {
    float input_x[ARRAY_LENGTH] = {0.3426, 3.5784, 2.7694,
       -1.3499, 3.0349, 0.7254, -0.0631};
    float fixed_input_x[ARRAY_LENGTH];
    float h_array[IMPULSE_LENGTH] = {0.7147, -0.2050, -0.1241,
       1.4897, 1.4090};
    float fixed_h_array[IMPULSE_LENGTH];
    float fixed_conv[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_conv[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_fix_diff[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_fix_mse;

    floatToFixed(input_x, fixed_input_x);
    floatToFixed(h_array, fixed_h_array);

    conv_taker(fixed_input_x, fixed_h_array, fixed_conv);
    conv_taker(input_x, h_array, float_conv);

    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        fixed_conv[i] /= pow(Q_FORMAT, 2);
    }

    printf("fixed convolution:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%f ", fixed_conv[i]);
    }
    printf("\n");

    printf("float convolution:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%f ", float_conv[i]);
```

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

    absoluteDifferenceSquare(fixed_conv, float_conv,
       float_fix_diff);

    printf("absolute error Square:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%e ", float_fix_diff[i]);
    }
    printf("\n");

    float_fix_mse = meanSquareError(fixed_conv, float_conv);
    printf("mean square: %e\n", float_fix_mse);

    return 0;
}

void conv_taker(float *signal, float *impulse, float *conv) {
    int L = ARRAY_LENGTH + IMPULSE_LENGTH - 1;
    for (int n = 0; n < L; n++) {
        for (int k = 0; k < IMPULSE_LENGTH; k++) {
            int m = n - k;
            if (m >= 0 && m < ARRAY_LENGTH) {
                conv[n] += signal[m] * impulse[k];
            }
        }
    }
}

float meanSquareError(float *array1, float *array2) {
    float mse = 0;
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        mse += pow(array1[i] - array2[i], 2);
    }
    mse /= ARRAY_LENGTH + IMPULSE_LENGTH - 1;
    return mse;
}

void absoluteDifferenceSquare(float *array1, float *array2, float
    *abs_diff) {
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        abs_diff[i] = fabs(array1[i] - array2[i]) * fabs(array1[i
           ] - array2[i]);
    }
}

void floatToFixed(float *input_array, float *fixed_array) {
    for (int i = 0; i < ARRAY_LENGTH; i++) {
        fixed_array[i] = (int)(input_array[i] * Q_FORMAT);
    }
```

```
}
```

2.1.6 Output

```
    fixed convolution:
    0.244771 2.486940 1.203478 -1.465452 7.914201 9.230689
       1.321182 2.541230 5.363862 0.928139 -0.088746
    float convolution:
    0.244856 2.487249 1.203201 -1.466209 7.915555 9.231352
       1.320703 2.541995 5.364633 0.928089 -0.088908
    absolute error Square:
    7.209167e-09 9.577002e-08 7.662044e-08 5.719348e-07 1.833905e
       -06 4.393087e-07 2.291963e-07 5.853554e-07 5.952470e-07
       2.506795e-09 2.607464e-08
    mean square: 4.057389e-07
```

2.2 Correlation

Correlation is a statistical measure that expresses the extent to which two variables are linearly related. The correlation coefficient is a value that indicates the strength and direction of the relationship between variables. In discrete time, the correlation of two sequences is given by

$$r[k] = \sum_{n=-\infty}^{\infty} x[n]h[n+k]$$

2.2.1 Implementation in MATLAB

The fixed point correlation is implemented in MATLAB using the same logic as in the convolution, but the correlation function is used instead of the convolution function. The difference in both the functions is that the second sequence is not flipped before the correlation operation. The absolute square difference and mean square error is calculated and printed. The absolute square of the difference between the fixed and floating point results is plotted.

2.2.2 Matlab Code

```
% CORRELATION

clc
clear
close all

input_x = [0.3426    3.5784    2.7694   -1.3499    3.0349
   0.7254   -0.0631];
fixed_input_x = floatToFixed(input_x);
h_array = [0.7147   -0.2050   -0.1241    1.4897    1.4090];
```

```
fixed_h_array = floatToFixed(h_array);

q_format = 2^12;
fixed_corr = correlation_taker(fixed_input_x, fixed_h_array)/(
   q_format^2);
float_corr = correlation_taker(input_x, h_array);
disp("fixed correlation:");
disp(fixed_corr);
disp("float correlation:");
disp(float_corr);
float_fix_diff = absoluteDifferenceSquare(fixed_corr, float_corr)
   ;
disp("absolute error Square:");
disp(float_fix_diff);
float_fix_mse = meanSquareError(fixed_corr, float_corr);
disp("mean square : ");
disp( float_fix_mse);


figure;
stem(float_fix_diff);
title('Absolute Difference Square (Correlation)');
xlabel('Sample');
ylabel('Absolute Difference Square');

function mse = meanSquareError(array1, array2)
mse = 0;
for i = 1:length(array1)
    mse = mse + (array1(i) - array2(i))^2;
end
mse = mse / length(array1);
end

function abs_diff = absoluteDifferenceSquare(array1, array2)
abs_diff = abs((array1 - array2).^2);
end

function [correlation] = correlation_taker(signal, impulse)
    L = length(signal) + length(impulse) - 1;
    correlation = zeros(1, L);

    for n = 1:L
        for k = 1:length(impulse)
            m = n - k + 1;
            if 1 <= m && m <= length(signal)
                correlation(n) = correlation(n) + signal(m) *
                   impulse(length(impulse) - k + 1);
            end
        end
    end
    end
```

```
function fixed_array = floatToFixed(input_array)
q_format = 2^12;
fixed_array = fix(input_array * q_format);
end
```

2.2.3 Output

```
    fixed correlation:
    0.4826    5.5519    9.1893    1.7090    1.4334    7.6994
       2.8707   -1.7700    2.0278    0.5312   -0.0450

    float correlation:
        0.4827    5.5523    9.1903    1.7093    1.4328    7.7005
              2.8711   -1.7710    2.0282    0.5314   -0.0451

    absolute error Square:
       1.0e-05 *

        0.0015    0.0200    0.1118    0.0046    0.0309    0.1173
              0.0160    0.0925    0.0127    0.0022    0.0007

    mean square :
       3.7289e-07
```

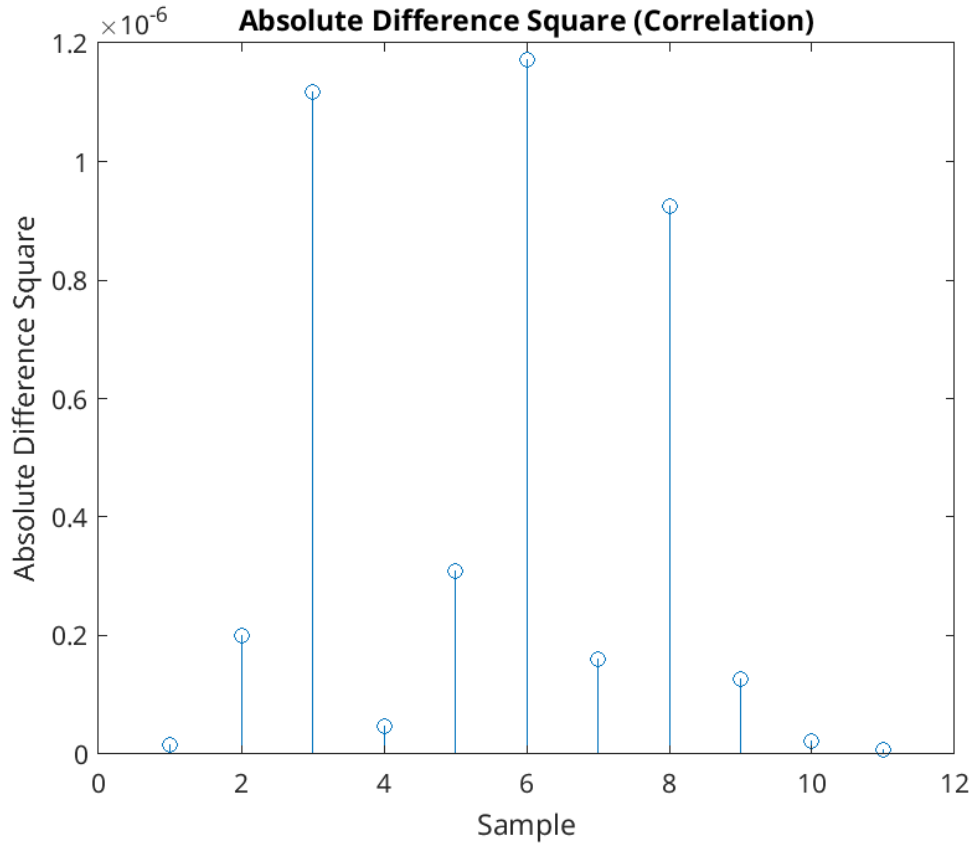

Figure 2: Square of difference between fixed and floating point correlation

2.2.4 Implementation in C

The fixed point correlation is implemented in C using the same logic as in MATLAB. The absolute square difference and mean square error is calculated and printed.

2.2.5 C Code

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

#define ARRAY_LENGTH 7
#define IMPULSE_LENGTH 5
#define Q_FORMAT pow(2, 12)

void correlation(float *signal, float *impulse, float *corr);
float meanSquareError(float *array1, float *array2);
void absoluteDifferenceSquare(float *array1, float *array2, float
    *abs_diff);
void floatToFixed(float *input_array, float *fixed_array);

int main() {
    float input_x[ARRAY_LENGTH] = {0.3426, 3.5784, 2.7694,
       -1.3499, 3.0349, 0.7254, -0.0631};
    float fixed_input_x[ARRAY_LENGTH];
```

```
    float h_array[IMPULSE_LENGTH] = {0.7147, -0.2050, -0.1241,
       1.4897, 1.4090};
    float fixed_h_array[IMPULSE_LENGTH];
    float fixed_corr[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_corr[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_fix_diff[ARRAY_LENGTH + IMPULSE_LENGTH - 1];
    float float_fix_mse;

    floatToFixed(input_x, fixed_input_x);
    floatToFixed(h_array, fixed_h_array);

    correlation(fixed_input_x, fixed_h_array, fixed_corr);
    correlation(input_x, h_array, float_corr);

    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        fixed_corr[i] /= pow(Q_FORMAT, 2);
    }

    printf("fixed correlation:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%f ", fixed_corr[i]);
    }
    printf("\n");

    printf("float correlation:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%f ", float_corr[i]);
    }
    printf("\n");

    absoluteDifferenceSquare(fixed_corr, float_corr,
       float_fix_diff);

    printf("absolute error Square:\n");
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        printf("%e ", float_fix_diff[i]);
    }
    printf("\n");

    float_fix_mse = meanSquareError(fixed_corr, float_corr);
    printf("mean square: %e\n", float_fix_mse);

    return 0;
}

void correlation(float *signal, float *impulse, float *corr) {
    int L = ARRAY_LENGTH + IMPULSE_LENGTH - 1;
    for (int n = 0; n < L; n++) {
        for (int k = 0; k < IMPULSE_LENGTH; k++) {
            int m = n - k ;
            if (m >= 0 && m < ARRAY_LENGTH) {
```

```
                corr[n] += signal[m] * impulse[IMPULSE_LENGTH - 1
                    - k];
            }
        }
    }
}

float meanSquareError(float *array1, float *array2) {
    float mse = 0;
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        mse += pow(array1[i] - array2[i], 2);
    }
    mse /= ARRAY_LENGTH + IMPULSE_LENGTH - 1;
    return mse;
}

void absoluteDifferenceSquare(float *array1, float *array2, float
    *abs_diff) {
    for (int i = 0; i < ARRAY_LENGTH + IMPULSE_LENGTH - 1; i++) {
        abs_diff[i] = fabs(array1[i] - array2[i]) * fabs(array1[i
           ] - array2[i]);
    }
}

void floatToFixed(float *input_array, float *fixed_array) {
    for (int i = 0; i < ARRAY_LENGTH; i++) {
        fixed_array[i] = (int)(input_array[i] * Q_FORMAT);
    }
}
```

2.2.6 Output

```
    fixed correlation:
    0.482602 5.551889 9.189254 1.709038 1.433385 7.699375
       2.870709 -1.769989 2.027810 0.531231 -0.045011
    float correlation:
    0.482723 5.552337 9.190310 1.709254 1.432830 7.700457
       2.871109 -1.770950 2.028167 0.531379 -0.045098
    absolute error Square:
    1.480672e-08 2.000534e-07 1.116554e-06 4.629932e-08 3.084648e
       -07 1.170602e-06 1.598624e-07 9.247925e-07 1.272165e-07
       2.183299e-08 7.425216e-09
    mean square: 3.725372e-07
```

3 Observations

- The absolute square of the difference between the fixed and floating point results is plotted in Figure 1 and Figure 2.

- From the plot, it is clear that the difference between the fixed and floating point results is very small, but peaks at certain points.
- This is because the fixed point representation has a limited range and precision, and the rounding, truncation and overflow errors accumulate over time and cause the error to increase at certain points.
- The mean square error is also very small, which indicates that the fixed point implementation is very close to the floating point implementation.
- The C and MATLAB implementations give the same results, which indicates that the fixed point implementation is consistent across different programming languages.

4 Conclusion

Finally, we can conclude that the fixed point implementation of convolution and correlation is very close to the floating point implementation, with a very small mean square error. This shows that fixed point implementation is a good approximation of floating point implementation, and can be used in applications where floating point arithmetic is not feasible. Even though floating-point processing can represent a wider range of numbers and is more precise than fixed-point processing, fixed-point processing is less expensive and easier to develop for.