DSP Lab

ASSIGNMENT 4 — Fixed Point Convolution

Akhil Kumar Donka
EE22MTECH02003

April 10, 2022

Problem

To understand Q format along with the plot of Q format vs MSE (Mean Square Error) by computing fixed point convolutions on arbitrary sequences X and filter coefficients H.

Convolution Operation

Floating point convolution is given by :

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{1}$$

Whereas fixed point convolution is given by :

$$(x * h)[n] = scalar\_addition \sum_{k=-\infty}^{\infty} [scalar\_multiplication(x[k]h[n-k])] \tag{2}$$

While performing scalar multiplication, we have to take care of the generated Q format representation and accordingly convert the fixed point to desired Q format.

Plot observations

From the below graphs we can observe that mean square errors increases drastically after a certain Q value and this Q value is decided mainly by the range of numbers which signal is generating.

- Figure 1 is the plot when values in X (i.e signal) & H (i.e filter) are in range (0,1). For such values, we can observe error is nearly zero for values between Q = 14 to Q = 27. For values greater than this, overflow errors occur.
- Figure 2 is the plot when values in X (i.e signal) & H (i.e filter) are in range (0,5). For such values, we can observe error is nearly zero for values between Q = 14 to Q = 23. For values greater than this, overflow errors occur.
- Figure 3 is the plot when values in X (i.e signal) & H (i.e filter) are in range (0,10). For such values, we can observe error is nearly zero for values between Q = 14 to Q = 21. For values greater than this, overflow errors occur.

The errors are getting increased due to the event of overflow in representing bigger sized numbers. This clearly shows that there is a trade-off between fractional part precision and integer part precision. In the event of overflow, we reduce the Q-Value, and in case of loss of precision, increment the Q-Value. So to get the best of both the worlds, we can set values between $Q = 16$ to $Q = 19$.

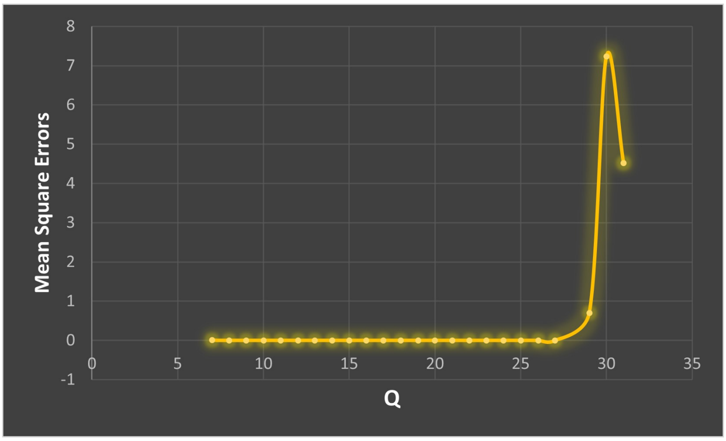

Figure 1: RMS Error vs Q (For values of signal b/w 0 & 1)

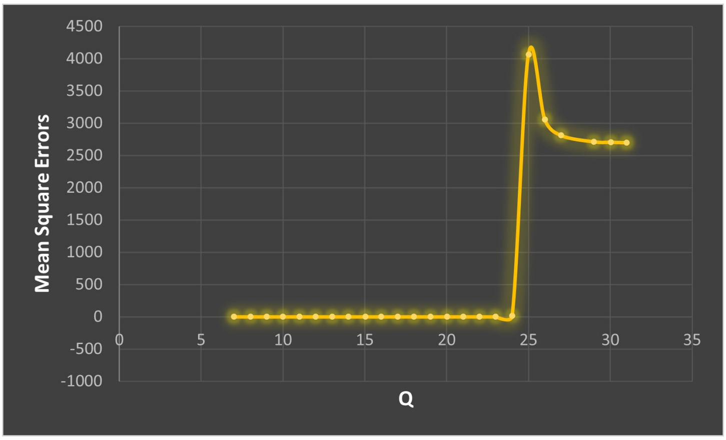

Figure 2: RMS Error vs Q (For values of signal b/w 0 & 5)

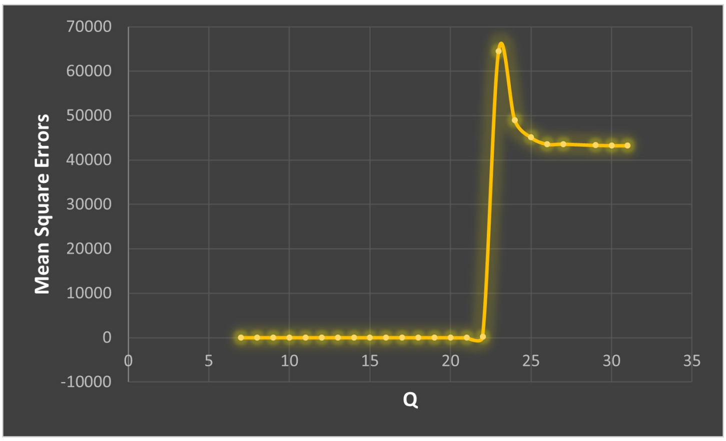

Figure 3: RMS Error vs Q (For values of signal b/w 0 & 10)

C Code

```
// importing standard libraries
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define MAXLEN 100

// function to perform fixed point addition
int fxdAddScalar(int num1, int num2)
{
    return num1+num2;
}

// converting a number from existing to new Q format
int bitshift(int num,int shift)
{
    return num*(1<<shift);
}

// function to perform fixed point multiplication
int fxdMulScalar(int num1, int q1, int num2, int q2, int resq)
{
    long long temp = (long long) num1*num2;
    int res;
    res = temp/(1<<(q1+q2-resq));
    return res;
}

// function to perform floating point convolution
float *convolutionflt(float X[MAXLEN], float H[MAXLEN], float Y[MAXLEN], int lenX,
                        int lenH, int lenY){
    for(int n=0;n<lenY;n++)
    {
        Y[n] = 0;
        for(int k = 0;k<=n;k++)
        {
        if((n-k)>=lenH || k>=lenX)
        continue;
```

```
        Y[n] = Y[n] + (X[k]*H[n-k]);
        }
    }
    return Y;
}

// function to perform fixed point convolution
int *convolutionfxd(int X[MAXLEN], int H[MAXLEN], int Y[MAXLEN], int lenX,
                    int lenH, int lenY, int Q)
{
    for(int n=0;n<lenY;n++)
    {
        Y[n] = 0;
        for(int k = 0;k<=n;k++)
        {
        if((n-k)>=lenH || k>=lenX)
        continue;
        Y[n] = fxdAddScalar(bitshift(Y[n],0),fxdMulScalar(X[k],Q,H[n-k],Q,Q));
        }
    }
    return Y;
}

int main(){
    // declaring float arrays for floating point convolution
    float X[MAXLEN];
    float H[MAXLEN];
    float Y[MAXLEN];

    // declaring integer arrays for fixed point convolution
    int Xfix[MAXLEN];
    int Hfix[MAXLEN];
    int Yfix[MAXLEN];

    // setting lengths of sequence and filter
    int xlen = 10;
    int hlen = 25;
    int ylen = xlen +hlen -1;
    float *floatyConv;
    int *fixedyConv;
    float fixedToFloatY[MAXLEN];
```

```
// declaring Q Values for which errors needs to be computed and # iterations
int niter = 10000;
int qValue[] = {7,8,9,10,11,12,13,14,15,16,17,
                18,19,20,21,22,23,24,25,26,27.28,29,30,31};
float errors[MAXLEN];

for(int q = 0;q<sizeof(qValue)/sizeof(qValue[0]);q++){

    float iterationError=0;

    for(int j =0; j<niter;j++){

        float err=0;

        // Generating random numbers for X
        for (int i=0;i<xlen;i++)
        {
            X[i] = ((float)rand()/RAND_MAX)*(float)(1);
        }

        // Generating random numbers for H
        for(int i=0;i<hlen;i++)
        {
            H[i] =  ((float)rand()/RAND_MAX)*(float)(1);
        }

        // Fixed point conversion for X
        for (int i=0;i<xlen;i++)
        {
            Xfix[i] = (int)(X[i]*(1<<qValue[q]));
        }

        // Fixed point conversion for H
        for (int i=0;i<hlen;i++)
        {
            Hfix[i] = (int)(H[i]*(1<<qValue[q]));
        }

        // floating point convolution
        floatyConv=convolutionflt(X, H, Y, xlen, hlen, ylen);
```

```
            // fixed point convolution
            fixedyConv=convolutionfxd(Xfix, Hfix, Yfix, xlen, hlen, ylen, qValue[q]);

            // fixed point convolution output to floating point values
            for(int i=0; i<ylen; i++){
                fixedToFloatY[i] = (float)fixedyConv[i]/(1<<qValue[q]);
            }

            for(int i=0; i<ylen; i++){
                err += pow((floatyConv[i] - fixedToFloatY[i]), 2);
            }

            // getting mean error of a sequence in an interation
            err = err/ylen;
            iterationError += err;
        }
        // storing error for each Q value averaged over all iterations
        errors[q] = iterationError/niter;
        printf("Q : %d, Error : %f\n", qValue[q], iterationError/niter);
    }

    return 0;
}
```