**#UNDERSTANDING AND EXPLANATION OF FIXED-POINT OPERATIONS**

* Figure 1 shows that under the fixed-point arithmetic the total number of bits are divided into fraction part and integer part.

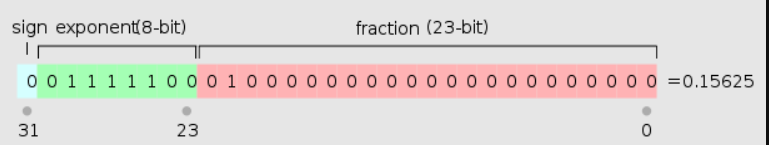
****

Figure 1 Fixed Point Notation

* The value Q shows the number of the bits dedicated for the fractional part.
* There exists a standard formula the helps to calculate the fixed point value of a floating point number.
* The Excess and low value of Q causes a trade-off between the accuracy.
* Our Aim is to vary the check the MSE(Mean Square Error) between the various Q values that can be set.
* As the Q value will increase the number of bits left for integer part of the number will decrease. Since the total number of the bits for this data type is fixed.
* Convolution operation has addition and multiplications in it. For this we write a function that can handle multiplication and addition of fixed point numbers and then complete the convolution.
* After this operation then again we will convert fixed point values to floating point values.
* We aim to calculate the error between fixed point convolution result and floating point convolution result.
* Take the mean square of this errors complete. Generate random numbers to do this.
* Complete this for 10000 iteration for various values of Q.
* As Q will increase it will error will be less until a point after which number of bits to express the integer part also become insufficient.

**#SOURCE CODE IN C THAT ENABLE US TO CALCULATE MSE FOR FIXED POINT OPERATION FOR VARIOUS Q VALUES**

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <stdio.h>

#define MAXIMLENGTH 200

//FUNCTION TO CALCULATE THE FLOATING POINT CONVOLUTION

float \*convolutionflt(float X[MAXIMLENGTH], float H[MAXIMLENGTH], float Y[MAXIMLENGTH], 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 ADD SCALAR IN FIXED POINT

int fxdAddScalar(int num1, int num2)

{

return num1+num2;

}

//FUNCTION TO SHIFT THE BIT POSITION TO CALCULATE 2^X

int bitshift(int num,int shift)

{

return num\*(1<<shift);

}

//FUNCTION TO CALCULATE THE FIXED POINT SCALAR 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 CALCULATE THE FIXED POINT CONVOLUTION

int \*convolutionfxd(int X[MAXIMLENGTH], int H[MAXIMLENGTH], int Y[MAXIMLENGTH], 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;

}

//MAIN FUNCTION TO CALCULATE MSE FOR VARIOUS Q

int main(){

//DECLARE INPUT AND OUTPUT ARRAY FOR FLOATING POINT CONVOLUTION

float X[MAXIMLENGTH];

float H[MAXIMLENGTH];

float Y[MAXIMLENGTH];

//DECLARE THE INPUT AND OUTPUT ARRAY FOR FIXED POINT CONVOLUTION

int Xfix[MAXIMLENGTH];

int Hfix[MAXIMLENGTH];

int Yfix[MAXIMLENGTH];

//DECLARE THE LENGTHS FOR INPUT AND OUTPUT ARRAYS OF THE CONVOLUTION

int xlen = 10;

int hlen = 25;

int ylen = xlen +hlen -1;

//DECLARE THE POINTERS TO PASS VALUE TO THE FUNCTION

float \*floatyConv;

int \*fixedyConv;

float fixedToFloatY[MAXIMLENGTH];

//SET THE ITERATION TO 10000 NUMBERS

int niter = 10000;

//DECLARE AND ARRAY THAT HAS Q VALUES FROM 7 TO 31

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};

//DECLARE AN ARRAY TO STORE THE ERRORS

float errors[MAXIMLENGTH];

// CALCULATE ERROR AND RELATED PARAMETERS FOR VARIOUS Q VALUES

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

float iterationError=0;

//DO THE BELOW OPERATION FOR 10000 TIMES

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)(10);

}

// 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);

}

//AVERAGE OUT THE ERROR FROM THE ABOVE OBTAINED

err = err/ylen;

iterationError += err;

}

errors[q] = iterationError/niter;

printf("Q VALUE IS : %d, Error VALUE IS : %f\n", qValue[q], iterationError/niter);

}

return 0;

}

**RESULT FOR MSE ERROR VALUE FOR RANDOM NUMBERS FROM 0 TO 10 AT VARIOUS Q VALUES**

Q VALUE IS : 7, Error VALUE IS : 0.121773

Q VALUE IS : 8, Error VALUE IS : 0.030446

Q VALUE IS : 9, Error VALUE IS : 0.007598

Q VALUE IS : 10, Error VALUE IS : 0.001904

Q VALUE IS : 11, Error VALUE IS : 0.000475

Q VALUE IS : 12, Error VALUE IS : 0.000119

Q VALUE IS : 13, Error VALUE IS : 0.000030

Q VALUE IS : 14, Error VALUE IS : 0.000007

Q VALUE IS : 15, Error VALUE IS : 0.000002

Q VALUE IS : 16, Error VALUE IS : 0.000000

Q VALUE IS : 17, Error VALUE IS : 0.000000

Q VALUE IS : 18, Error VALUE IS : 0.000000

Q VALUE IS : 19, Error VALUE IS : 0.000000

Q VALUE IS : 20, Error VALUE IS : 0.000000

Q VALUE IS : 21, Error VALUE IS : 0.000000

Q VALUE IS : 22, Error VALUE IS : 277.564362

Q VALUE IS : 23, Error VALUE IS : 63843.609375

Q VALUE IS : 24, Error VALUE IS : 48855.890625

Q VALUE IS : 25, Error VALUE IS : 45164.343750

Q VALUE IS : 26, Error VALUE IS : 44061.128906

Q VALUE IS : 27, Error VALUE IS : 43364.675781

Q VALUE IS : 29, Error VALUE IS : 43625.859375

Q VALUE IS : 30, Error VALUE IS : 43637.570312

Q VALUE IS : 31, Error VALUE IS : 43341.019531

**RESULT FOR MSE ERROR VALUE FOR RANDOM NUMBERS FROM 0 TO 1 AT VARIOUS Q VALUES**

Q VALUE IS : 7, Error VALUE IS : 0.042785

Q VALUE IS : 8, Error VALUE IS : 0.010785

Q VALUE IS : 9, Error VALUE IS : 0.002707

Q VALUE IS : 10, Error VALUE IS : 0.000676

Q VALUE IS : 11, Error VALUE IS : 0.000170

Q VALUE IS : 12, Error VALUE IS : 0.000042

Q VALUE IS : 13, Error VALUE IS : 0.000011

Q VALUE IS : 14, Error VALUE IS : 0.000003

Q VALUE IS : 15, Error VALUE IS : 0.000001

Q VALUE IS : 16, Error VALUE IS : 0.000000

Q VALUE IS : 17, Error VALUE IS : 0.000000

Q VALUE IS : 18, Error VALUE IS : 0.000000

Q VALUE IS : 19, Error VALUE IS : 0.000000

Q VALUE IS : 20, Error VALUE IS : 0.000000

Q VALUE IS : 21, Error VALUE IS : 0.000000

Q VALUE IS : 22, Error VALUE IS : 0.000000

Q VALUE IS : 23, Error VALUE IS : 0.000000

Q VALUE IS : 24, Error VALUE IS : 0.000000

Q VALUE IS : 25, Error VALUE IS : 0.000000

Q VALUE IS : 26, Error VALUE IS : 344.455688

Q VALUE IS : 27, Error VALUE IS : 612.976257

Q VALUE IS : 29, Error VALUE IS : 441.819061

Q VALUE IS : 30, Error VALUE IS : 437.971985

Q VALUE IS : 31, Error VALUE IS : 433. 44250