```
::::::::::::::
makefile
::::::::::::::
.SUFFIXES : .o .cpp .c

.cpp.o :
        g++ -c -g -I${HOME}/include $<

.c.o :
        gcc -c -g -I${HOME}/include $<  ${F}


#---------------------------------------------------------------
SRC = lut_conv.c                        \
      lut_comp.c                        \

lut_conv : lut_conv.o
        gcc lut_conv.o -o lut_conv -lm

lut_comp : lut_comp.o
        gcc lut_comp.o -o lut_comp -lm


run_lut_conv : lut_conv
         ./lut_conv 29

run_lut_comp : lut_comp
         ./lut_comp 29


DAT = lut_real.dat
BAT = batch_run_lut_comp


print :
        /usr/bin/more makefile ${SRC} ${BAT}  > lut.src.print
        /usr/bin/more ${DAT} *.out   > lut.out.print


tar :
        mkdir src
        cp makefile ${SRC} ${BAT} ${DAT} src
        tar cvf lut.tar src
        \rm -fr src


#---------------------------------------------------------------
clean_obj:
        \rm -f *.o
clean_exe:
        \rm -f lut_comp lut_conv
clean_out:
        \rm -f print.* *.out

clean : clean_obj clean_exe clean_out
        \rm -f *~ *# *.UA0 a.out



::::::::::::::
lut_conv.c
::::::::::::::
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#define MAX_LUT (1 << 10)
// #define STR_PR


//-----------------------------------------------------------------------------
//  Purpose: Convert lut_real.dat into fixed lut_fint.dat
//
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2013.10.09
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//
//  Inputs : lut_real.dat
//  Outputs: lut_fint.dat
//
//-----------------------------------------------------------------------------


//---------------------------------------------
//  lut_real.dat format
//    1234567890123456789
//  7.8539816339744830962E-01
//  %25.19e
//---------------------------------------------



int main (int argc, char *argv[])
{

   FILE   *fin, *fout;
   int    i, frac_bits;
   double tmp, err, nerr;


   double         A_real[MAX_LUT];
   long long int  A_fint[MAX_LUT];
   char           str[256];


   if (argc < 2) {
      frac_bits = 29;
   } else {
      frac_bits = atoi(argv[1]);
   }

   printf("number of bits for fractional part : %d \n", frac_bits);


   printf("sizeof(long long int) = %d \n", sizeof(long long int));
   printf("col #1 : index i\n");
   printf("col #2 : number in string \n");
   printf("col #3 : double type data \n");
   printf("col #4 : fixed point integer \n");
   printf("col #5 : hexadecimal number \n");
   printf("col #6 : error \n");
   printf("col #7 : normalized err\n");


   fin = fopen("lut_real.dat", "r");
   fout = fopen("lut_fint.dat", "w");


   if (fin == NULL) {
     perror ("Unable to open file fin \n");
     exit( EXIT_FAILURE);
   }

   if (fout == NULL) {
     perror ("Unable to open file fout \n");
     exit( EXIT_FAILURE);
   }



   i = 0;


   while (fscanf(fin, "%s", str) != EOF) {

     A_real[i] = atof(str);
     A_fint[i] = (long long int) (A_real[i] * pow(2, frac_bits));  // * 2^29
     err       = A_real[i] - (double) A_fint[i] / pow(2, frac_bits) ;  // / 2^29
     nerr      = err / A_real[i];

     printf("[%3d] ",    i);
     printf("%25s ",     str);
     printf("%25.19e ",  A_real[i]);
     printf("%16lld ",   A_fint[i]);
     printf("%018x ",    A_fint[i]);
     printf("%10e ",     err);
     printf("%10e ",     nerr);
     printf(" \n");

     fprintf(fout, " %20lld \n", A_fint[i]);

     i++;
   }


   fclose(fin);
   fclose(fout);

   return 0;

}

::::::::::::::
lut_comp.c
::::::::::::::
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_ANGLE (1 << 10)

//#define VERBOSE


//-----------------------------------------------------------------------------
```

```
//  Purpose: Error Analysis of the converted integer angles
//
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2013.10.09
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//
//  Inputs : lut_real.dat
//
//
//-----------------------------------------------------------------------------



//---------------------------------------------
//  lut_real.dat format
//    1234567890123456789
//  7.8539816339744830962E-01
//  %25.19e
//---------------------------------------------


int main (int argc, char *argv[])
{

   FILE *fin;

   int i, num, frac_bits;
   double err, nerr;

   double         A_real[MAX_ANGLE];
   long long int  A_fint[MAX_ANGLE];   // 8-byte integer
   double         A_freal[MAX_ANGLE];
   char           str[256];

   double A_delta[MAX_ANGLE];
   double cos_delta[MAX_ANGLE];
   double sin_delta[MAX_ANGLE];


   if (argc < 2) {
      frac_bits = 29;
   } else {
      frac_bits = atoi(argv[1]);
   }

   printf("number of bits for fractional part : %d \n", frac_bits);


#ifdef VERBOSE
   printf("sizeof(long long int) = %d \n", sizeof(long long int));
#endif


   //---------------------------------------------------
   fin = fopen("lut_real.dat", "r");
   if (fin == NULL) {
     perror ("Unable to open file fin \n");
     exit( EXIT_FAILURE);
   }


   i = 0;

   while (fscanf(fin, "%s", str) != EOF) {

     A_real[i]  = atof(str);
     A_fint[i]  = (long long int) (A_real[i] * pow(2, frac_bits));  // * 2^29
     A_freal[i] = (double) A_fint[i] / pow(2, frac_bits); // divide by 2^29
     err        = A_real[i] - A_freal[i] ;
     nerr       = err / A_real[i];

#ifdef VERBOSE
     printf("[%3d] ",    i);
     printf("%25s ",     str);
     printf("%25.19e ",  A_real[i]);
     printf("%16lld ",   A_fint[i]);
     printf("%018x ",    A_fint[i]);
     printf("%10e ",     err);
     printf("%10e ",     nerr);
     printf(" \n");
#endif

     i++;
   }
   num = i;
   //------------------------------------------------------------
   for (i=0; i<num; ++i) {
     A_delta[i] = A_real[i] - A_freal[i];

#ifdef VERBOSE
     if (i==0) printf("\n\n* A_delta \n");
     printf("[%2d] ", i);
     printf("A_real= %25.19e ",   A_real[i]);
     printf("A_freal= %25.19e ",  A_freal[i]);
     printf("A_delta= %15.10e ",  A_delta[i]);
     printf("\n");
#endif
   }




   //------------------------------------------------------------
   for (i=0; i<num; ++i) {
     cos_delta[i] = cos(A_real[i]) - cos(A_freal[i]);

#ifdef VERBOSE
     if (i==0) printf("\n\n* cos_delta \n");
     printf("[%2d] ", i);
     printf("cos(A_real)= %18.10e ",  cos(A_real[i]));
     printf("cos(A_freal)= %18.10e ", cos(A_freal[i]));
     printf("cos_delta= %18.10e ",    cos_delta[i]);
     printf("\n");
#endif
   }


   //------------------------------------------------------------
   for (i=0; i<num; ++i) {
     sin_delta[i] = sin(A_real[i]) - sin(A_freal[i]);

#ifdef VERBOSE
     if (i==0) printf("\n\n* sin_delta \n");
     printf("[%2d] ", i);
     printf("sin(A_real)= %18.10e ",  sin(A_real[i]));
     printf("sin(A_freal)= %18.10e ", sin(A_freal[i]));
     printf("sin_delta= %18.10e ",    sin_delta[i]);
     printf("\n");
#endif
   }
   double dA_avg   = 0.0;
   double dcos_avg = 0.0;
   double dsin_avg = 0.0;

   double dA_max   = -9999L;
   double dcos_max = -9999L;
   double dsin_max = -9999L;

   double dA_min   = +9999L;
   double dcos_min = +9999L;
   double dsin_min = +9999L;


   for (i=0; i<num; ++i) {
     A_delta[i]   = fabs(A_delta[i]);
     cos_delta[i] = fabs(cos_delta[i]);
     sin_delta[i] = fabs(sin_delta[i]);

     dA_avg   += A_delta[i];
     dcos_avg += cos_delta[i];
     dsin_avg += sin_delta[i];

     dA_min   = fmin(dA_min,   A_delta[i]);
     dcos_min = fmin(dcos_min, cos_delta[i]);
     dsin_min = fmin(dsin_min, sin_delta[i]);

     dA_max   = fmax(dA_max,   A_delta[i]);
     dcos_max = fmax(dcos_max, cos_delta[i]);
     dsin_max = fmax(dsin_max, sin_delta[i]);
   }

   dA_avg   /= num;
   dcos_avg /= num;
   dsin_avg /= num;


#ifdef VERBOSE
   printf("\n\n");
   printf("* fractional bits = %d \n", frac_bits);
#endif

   printf("|A_delta|   (min= %.2e avg= %.2e max= %.2e)\n",
          dA_min, dA_avg, dA_max);
   printf("|cos_delta| (min= %.2e avg= %.2e max= %.2e)\n",
          dcos_min, dcos_avg, dcos_max);
   printf("|sin_delta| (min= %.2e avg= %.2e max= %.2e)\n",
          dsin_min, dsin_avg, dsin_max);
   fclose(fin);


   return 0;

}

::::::::::::::
batch_run_lut_comp
::::::::::::::
#!/bin/bash


#-----------------------------------
N=29
while [ $N -lt 61 ]; do
  # echo N = $N
  ./lut_comp $N | sed 's/num.*part/nbits/; s/min= .*max=/max=/; s/  */ /;' | \
  sed 'N; s/\n/ /; N; s/\n/ /; N; s/\n/ /;'
  let N=N+1
done
```