```
/*GRADIENT101.C:CODED BY JUNSATO
SINCE 2019.8.14.*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <math.h>
#include <dos.h>
#include <limits.h>
#include <windows.h>

#define PI 3.1415926535897932384

double formula(int dim,double *var);
double *mallocdoublevector(int vsize);
double **mallocdoublematrix(int nline,
int nrow);
void freematrix(double **mtx,int nline);

/*CONJUGATE GRADIENT TEST*/
main(void)
{
   FILE *fout;
   char non[10],str[256];
   int i,j,k,m,n;
   double df,f1,f2,fa,c1,c2,c3,alpha,beta,
   gamma,vsize,eps;
   /*COORDINATES*/
   double *x1,*x2,*xa,*xx,*dx;
   /*GRADIENT VECTOR*/
   double *u1,*u2,*ua,*fgrad1,*fgrad2;
   double **cmtx; /*MATRIX*/

   fout=fopen("gradtest.txt","w");

   m=2; /*DIMENSION OF VECTOR*/
   n=m;
   gamma=1.0;
   eps=0.001;

   x1=mallocdoublevector(m);
   x2=mallocdoublevector(m);
   xa=mallocdoublevector(m);
   xx=mallocdoublevector(m);
   dx=mallocdoublevector(m);

   u1=mallocdoublevector(n);
   u2=mallocdoublevector(n);
   ua=mallocdoublevector(n);
   fgrad1=mallocdoublevector(n);
   fgrad2=mallocdoublevector(n);

*(x1+0)=1.0; *(x1+1)=1.0;
*(dx+0)=0.001; *(dx+1)=0.001;
f1 = formula(m,x1);
```

↑

５回出てくるこの *formula* を自分が指標(目的関数)にしたい関数やソフトウェアに置き換える。

```
for(i=0; i<m; i++)
{
   for(j=0; j<m; j++)
   {
      if(j==i) *(xx+j)=(*(x1+j))+(*(dx+j));
      else     *(xx+j)=(*(x1+j));
   }
   df = formula(m,xx);
   *(fgrad1+i)=gamma*(f1-df)/(*(dx+i));
   *(u1+i)=*(fgrad1+i);
}

fprintf(stderr,"INITIAL CONDITION¥n");
fprintf(fout,"f=%8.3f
{x1, x2}={%8.3f, %8.3f}¥n",
f1,*(x1+0),*(x1+1));
gets(non);

k=0;
while(1)
{
   k++;
   for(i=0; i<m; i++)
   {
     *(xa+i)=(*(x1+i))+(*(u1+i));
   }
   fa = formula(m,xa);
   for(i=0; i<m; i++)
   {
      for(j=0; j<m; j++)
      {
         if(j==i) *(xx+j)=(*(xa+j))+(*(dx+j));
         else     *(xx+j)=(*(xa+j));
      }
      df = formula(m,xx);
      *(ua+i)=-gamma*(fa-df)/(*(dx+i))
              +(*(fgrad1+i));
   }

   c1=0.0;
   c2=0.0;
   for(i=0; i<m; i++)
```

```
      {
         c1+=(*(fgrad1+i))*(*(fgrad1+i));
         c2+=(*(u1+i))*(*(ua+i));
      }
      alpha=c1/c2;
      vsize=0.0;
      for(i=0; i<m; i++)
      {
         *(x2+i)=(*(x1+i))+alpha*(*(u1+i));
         *(fgrad2+i)=(*(fgrad1+i))
                        -alpha*(*(ua+i));
         vsize+=(*(fgrad2+i))*(*(fgrad2+i));
      }

      f2 = formula(m,x2);

      fprintf(fout,"STEP %d f=%8.3f
       {x1, x2}={%8.3f, %8.3f}¥n",
       k,f2,*(x2+0),*(x2+1));
      fprintf(stderr,"grad f(x)
= %8.3f %8.3f¥n",*(fgrad2+0),*(fgrad2+1));
      fprintf(stderr,"VECTOR SIZE
= %9.5f¥n",vsize);
      gets(non);

      if(vsize<eps) break;

      c3=0.0;
      for(i=0; i<m; i++)
c3+=(*(fgrad2+i))*(*(fgrad2+i));
      beta=c3/c1;

      for(i=0; i<m; i++)
      {
         *(x1+i)=*(x2+i);
         *(u2+i)=(*(fgrad2+i))+beta*(*(u1+i));
         *(u1+i)=*(u2+i);
         *(fgrad1+i)=*(fgrad2+i);
      }
   }

   return NULL;
}/*main*/

double formula(int dim,double *var)
{
   double f;

   f=1.0*(*(var+0))*(*(var+0))
    +1.0*(*(var+1))*(*(var+1))
    +1.0*(*(var+0))*(*(var+1))
    -2.0*(*(var+0))
     -0.0*(*(var+1));

   return f;
}/*formula*/

double *mallocdoublevector(int vsize)
/*MALLOC DOUBLE VECTOR.*/
{
   double *v;

   v=(double *)malloc(vsize*sizeof(double));

   return v;
}/*mallocdoublevector*/

double **mallocdoublematrix(int nline,int
nrow)
/*MALLOC DOUBLE MATRIX.*/
{
   int i;

   double **mtx;
   mtx=(double **)malloc(nline*sizeof(double
*));
   for(i=0;i<nline;i++) *(mtx+i)=(double
*)malloc(nrow*sizeof(double));

   return mtx;
}/*mallocdoublematrix*/

void freematrix(double **mtx,int nline)
/*FREE MATRIX.*/
{
   int i;

   for(i=0;i<nline;i++) free(*(mtx+i));
   free(mtx);

   return;
}/*freematrix*/
```