Help

```
#include <math.h>
#include "mathtools.h"

/*Number of points of grid of levels l for Multig
    rid Algorithm*/
int nn(int l)
  {
  int kk,i;
  kk=2;
  for (i=1;i<l+1;i++)
    kk=kk*2;
  return(kk-1);
  }

/* Double abs()*/
double dabs(double x)
{
  if (x>0.) return x;
  else return (-x);
}

/* One-dimensional Normal Law. Density function *
    /
double nd(double x)
{
  return(exp(-SQR(x)/2.)/(sqrt(2.*PI)));
}

/*One-Dimensional Normal Law. Cumulative distribu
    tion function. */
/*Abramowitz, Milton et Stegun, Handbook of Mathe
    maticalFunctions, 1968, Dover Publication, New York
    , page 932 (26.2.18).Precision 10-7*/
double N(double x)
{
  const double p= 0.2316419;
  const double b1= 0.319381530;
  const double b2= -0.356563782;
  const double b3= 1.781477937;
  const double b4= -1.821255978;
```

```
  const double b5= 1.330274429;
  const double one_over_twopi= 0.39894228;

  double t;

  if(x >= 0.0)
    {
      t = 1.0 / ( 1.0 + p * x );
      return (1.0 - one_over_twopi * exp( -x * x
    / 2.0 ) * t * ( t *( t * ( t * ( t * b5 + b4 ) +
     b3 ) + b2 ) + b1 ));
    }
  else
    {/* x < 0 */
      t = 1.0 / ( 1.0 - p * x );
      return ( one_over_twopi * exp( -x * x / 2.0
     ) * t * ( t *( t * ( t * ( t * b5 + b4 ) + b3 )
     + b2 ) + b1 ));
    }
}



/*One-Dimensional Normal Law. Cumulative distribu
    tion function. */
/*Abramowitz, Milton et Stegun, Handbook of Mathe
    matical Functions,Precision 3x10-3*/
/*
double erf(double x)
{
  const double c1 = 0.196854;
  const double c2 = 0.115194;
  const double c3 = 0.000344;
  const double c4 = 0.019527;
  double t;

  if (x >= 0.0)
    {
      t = 1 + x * (c1 + x * (c2 + x * (c3 + x *
    c4)));
      t= t * t;
```

```
      t = t * t;
      return 1.0 - 0.5 / t;
    }
  else
    {
      t = 1 - x * (c1 - x * (c2 - x * (c3 - x *
    c4)));
      t = t * t;
      t = t * t;
      return 0.5 / t;
    }
}
*/
/*Two-Dimensional Normal Law. Cumulative distribu
    tion function.
  Drezner-Mathematics of Computation 32(1-1978)
    pp.277-279*/
static double ff(double x, double y, double aa,
    double bb, double r)
{
  return exp(aa*(2.*x-aa) + bb*(2.*y - bb) + 2.*
    r*(x-aa)*(y-bb));
}


static double NN1( double a, double b, double r)
{
  static const double u[4] = {0.3253030,0.4211071
    ,0.1334425,0.006374323};
  static const double v[4] = {0.1337764,0.6243247
    ,1.3425378,2.2626645};
  double aa,bb,m;
  int i,j;

  aa = a/(sqrt(2.*(1.-r*r)));
  bb = b/(sqrt(2.*(1.-r*r)));
  m = 0.0;

  for ( i = 0; i <= 3; i++)
    for ( j = 0; j <= 3; j++)
      m = m + u[i]*u[j]*ff(v[i], v[j], aa, bb, r)
```

```
    ;

  m = m*(sqrt(1.0- r*r))/PI;
  return (m);
}

static double NN2( double a, double b, double r)
{
  double res=0.0;

  if(( a <= 0.0 ) && ( b <= 0.0 ) && ( r <= 0.0 )
    )
    {
      res =  NN1(a, b, r);
    }
  else if (( a <= 0.0 ) && ( b >= 0.0 ) && ( r >=
     0.0 ))
    {
      res =  N(a) - NN1(a, -b, -r);
    }
  else if (( a >= 0.0 ) && ( b <= 0.0 ) && ( r >=
     0.0 ))
    {
      res = N(b) - NN1(-a, b, -r);
    }
  else if (( a >= 0.0 ) && ( b >= 0.0 ) && ( r <=
     0.0 ))
    {
      res = N(a) + N(b) - 1.0 + NN1(-a, -b, r);
    }
  return (res);
}


static double sgn( double x )
{
  double res=0.0;

  if ( x > 0.0 )
    {
      res = 1.;
```

```
    }
  else if (x < 0.0 )
    {
      res = -1.;
    }
  else if (x == 0.)
    {
      res = 0.0;
    }

  return (res);
}


double NN( double a, double b, double r)
{
  double r1,r2,d,res;

  if (r+PRECISION>=1)
    {
      res=N(MIN(a,b));
    }
  else
    if (r-PRECISION<=-1)
      {
    if (a>-b)
      {res=N(a)-N(-b);}
    else
      {res=0;}
      }
    else
      if( a*b*r <= 0.0 )
    {
      res = NN2(a,b,r);
    }
      else
    {
      r1 = (r*a - b)*sgn(a)/sqrt(a*a - 2.0*r*a*
    b + b*b);
      r2 = (r*b - a)*sgn(b)/sqrt(a*a - 2.0*r*a*
    b + b*b);
```

```
      d = (1.0- sgn(a)*sgn(b))/4.0;
      res = NN2( a, 0.0, r1 ) + NN2( b, 0.0, r2
    ) - d;
    }

  return (res);
}

#define NRANSI
#define SWAP(a,b) temp=(a);(a)=(b);(b)=temp;
#define M 7
#define NSTACK 50

static int istack[NSTACK];

void Sort(unsigned long n, double *arr)
{
  unsigned long i,ir=n,j,k,l=1;
  int jstack=0;
  double a,temp;

  for (;;) {
  if (ir-l < M) {
    for (j=l+1;j<=ir;j++) {
    a=arr[j];
    for (i=j-1;i>=1;i--) {
      if (arr[i] <= a) break;
      arr[i+1]=arr[i];
    }
    arr[i+1]=a;
    }
    if (jstack == 0) break;
    ir=istack[jstack--];
    l=istack[jstack--];
  }
  else {
    k=(l+ir) >> 1;
    SWAP(arr[k],arr[l+1])
    if (arr[l+1] > arr[ir]) {
      SWAP(arr[l+1],arr[ir])
      }
```

```
    if (arr[l] > arr[ir]) {
    SWAP(arr[l],arr[ir])
      }
    if (arr[l+1] > arr[l]) {
    SWAP(arr[l+1],arr[l])
      }
    i=l+1;
    j=ir;
    a=arr[l];
    for (;;) {
    do i++; while (arr[i] < a);
    do j--; while (arr[j] > a);
    if (j < i) break;
    SWAP(arr[i],arr[j]);
    }
    arr[l]=arr[j];
    arr[j]=a;
    jstack += 2;
    if (jstack > NSTACK){
    printf("SORTING ERROR{n");
    exit(0);
    };
    if (ir-i+1 >= j-l) {
    istack[jstack]=ir;
    istack[jstack-1]=i;
    ir=j-1;
    }
    else {
    istack[jstack]=j-1;
    istack[jstack-1]=l;
    l=i;
    }
  }
  }

}
#undef M
#undef NSTACK
#undef SWAP
#undef NRANSI
```

References